UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

GĂSIREA CONTURURILOR

Studenți

Tache Daria-Elena

Zăvoianu Catinca-Ioana

Coordonator științific

Rusu Cristian

București, ianuarie 2025

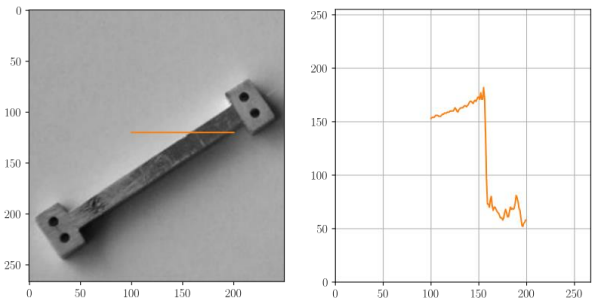
Cuprins

1. Introducere

Proiectul despre detecția marginilor (edge detection) se concentrează pe identificarea marginilor în imagini digitale, care sunt definite ca linii sau curbe unde luminozitatea imaginii suferă schimbări bruște sau întreruperi. Aceste schimbări pot semnala discrepanțe în adâncimea scenei, orientarea suprafeței, proprietățile materialului sau variațiile iluminării. Detecția marginilor este o tehnică esențială în procesarea imaginilor, fiind folosită în special pentru detectarea și extragerea trăsăturilor din imagini. Prin identificarea acestor margini, procesul de interpretare a imaginii poate fi simplificat, reducând volumul de date de procesat și filtrând informațiile mai puțin relevante. În ciuda importanței sale, în practică detecția marginilor poate întâmpina dificultăți, cum ar fi fragmentarea marginilor, lipsa unor segmente de margini sau apariția unor margini false care nu reflectă fenomene relevante.

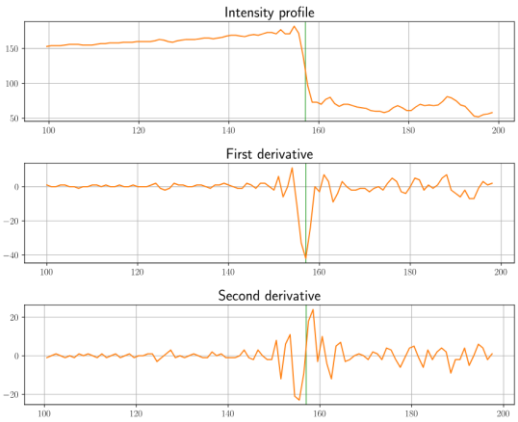
Problema abordată este una fundamentală în analiza imaginilor, fiind esențială pentru numeroase aplicații, de la recunoașterea obiectelor la realitatea augmentată. Această tehnică ajută la extragerea trăsăturilor structurale din imagini și facilitează reducerea complexității acestora, astfel încât să se poată identifica rapid informațiile esențiale. Detecția marginilor este utilizată într-o gamă largă de domenii, de la medicină (pentru imagistica medicală) până la automobilele autonome, unde identificarea precisă a marginii obiectelor din jur este esențială pentru siguranță.

Marginea unei imagini poate fi considerată ca o frontieră care delimitează două obiecte distincte. De obicei, aceasta este caracterizată printr-o variație rapidă a intensității pixelilor. Astfel, una dintre metodele matematice fundamentale utilizate pentru detectarea marginilor este analiza derivatelor funcției de intensitate a imaginii.



- intensitatea luminii de-a lungul liniei portocalii

Avantajul utilizării derivatelor: Pentru a detecta marginea unei imagini, se analizează derivata primei sau celei de-a doua ordini ale profilului de intensitate. O margine poate fi identificată prin analiza primei derivate, care evidențiază schimbările bruște ale luminozității în imagine. În mod similar, marginea poate fi detectată prin identificarea punctelor de trecere prin zero ale derivatei de ordinul al doilea. În cazul imaginilor digitale, derivatele sunt calculate folosind diferențe discrete, iar calculul acestora se realizează pe ambele axe (orizontală și verticală).



Definiții matematice:

Derivata de ordinul întâi, denumită „gradient”, măsoară variația intensității pixelilor și poate fi calculată prin convoluție cu un nucleu specific. Precum prima derivata este folosită pentru a calcula rata de schimbare a valorilor unei funcții, aici folosim o operație similară, pentru a afla rata de schimbare a intensității pixelilor.

Derivata de ordinul al doilea, denumită „Laplacian”, indică o schimbare de curbură și poate fi folosită pentru a detecta margini sau puncte de interes în imagini.

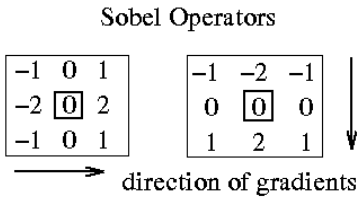
Operatorii pot fi organizați în două categorii mari, în funcție de abordarea matematică utilizată: operatori de gradient și operatori gaussieni. Fiecare categorie folosește tehnici matematice specifice pentru a identifica marginile imaginii.

2. Tipuri de Edge Detection

2.1 Prelucrarea imaginilor prin intermediul gradientului

2.1.1 Operatorul Sobel

Operatorul Sobel este o metodă clasică utilizată pentru calcularea aproximativă a gradientului unei imagini. Acesta este o extensie a metodei Gradient, care include un proces de netezire pentru a reduce influența zgomotului. Acesta folosește două măști 3x3 pentru a calcula magnitudinea gradientului si direcția la fiecare pixel:



Prima mască amplifică schimbările în intensitate de pe direcția orizontală. Valorile pozitive din partea dreaptă vor scoate în evidență zonele cu culori deschise din imagine, iar valorile negative din partea stângă vor scoate în evidență zonele închise la culoare. Cea de a doua mască funcționează similar cu cea anterioară, dar acționând pe direcția verticală. Aceste două măști funcționează în esență ca un filtru trece-sus pe imaginea ce trebuie prelucrată.

Magnitudinea gradientului rezultat si direcția, pot fi calculate cu ajutorul formulelor:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2} \qquad \mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Magnitude **Direction**

, unde G_x și G_y sunt gradientele în direcțiile orizontală și verticală, respectiv.

2.1.2 Pașii Algoritmului de detecție a muchiilor folosind operatorul Sobel:

1. Se convertește imaginea în tonuri de gri, deoarece operatorul Sobel funcționează pe imagini cu un singur canal.
2. Se calculează cele două gradiente (orizontal si vertical) convoluând pe rând imaginea cu măștile definite anterior.

3. Se calculează magnitudinea gradientului din gradientele pe direcțiile x și y folosind formula de mai sus.
4. Se aplică un prag pentru a clasifica pixelii ca fiind margini sau nu. Pixelii cu magnitudine peste prag sunt considerați margini.
5. Magnitudinea gradientului și gradientele individuale sunt normalizate în intervalul 0-255 pentru o vizualizare mai bună.
6. Normalizăm și afișăm imaginea detectată cu margini.

2.1.3 Avantaje și dezavantaje

- Avantaje ale Operatorului Sobel
    - Eficiență computațională: Simplu și rapid de implementat.
    - Detectarea marginilor netede: Este ideal pentru identificarea marginilor mai largi și mai uniforme.
- Limitări ale Operatorului Sobel
    - Sensibilitate la zgomot.
    - Detecție limitată a direcțiilor diagonale: Nu conservă bine punctele orientate diagonal.
    - Grosimea marginilor: Produce margini groase, care pot reduce precizia detecției.

2.2 Prelucrarea imaginilor prin intermediul unui filtru Gaussian

Operatorii gaussieni sunt utilizați pentru detectarea marginilor prin aplicarea unui **filtru Gaussian** asupra imaginii, urmată de calcularea gradientului. Acești operatori sunt mai sensibili la detalii fine și contribuie la reducerea zgomotului din imagine înainte de aplicarea operatorilor de gradient. Aplicarea unui filtru Gaussian ajută la **netezirea imaginii** și la estomparea micilor variații care nu sunt semnificative.

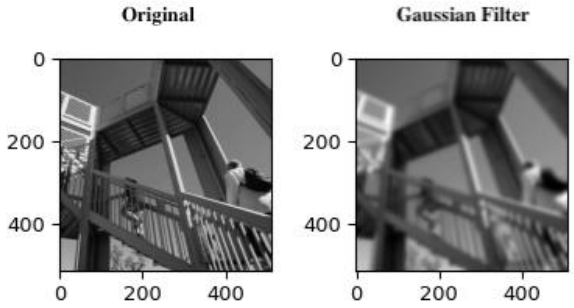
2.2.1 Operatorul Canny

Operatorul Canny este un algoritm utilizat pentru detectarea marginilor în procesarea imaginilor, având drept caracteristici cheie rata scăzută de eroare și detecție unică per muchie. Acesta aplică operația de convoluție pentru a evidenția contururile obiectelor și minimizează efectele zgomotului pentru rezultate mai precise.

2.2.2 Principiul Funcționării

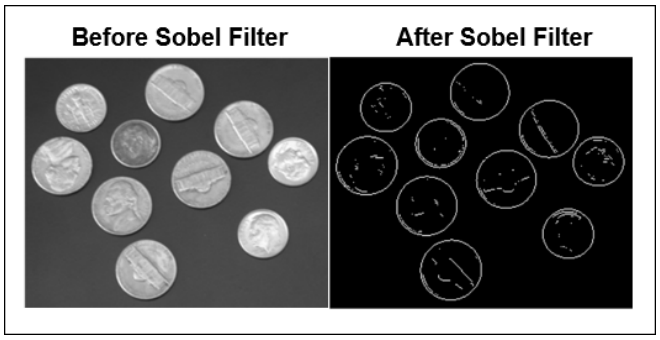
Algoritmul include mai multe etape, cum ar fi:

- Preprocesarea - constă în netezirea imaginii prin aplicarea unui kernel Gaussian, care acționează ca un filtru trece-jos, reducând variațiile bruște de intensitate.

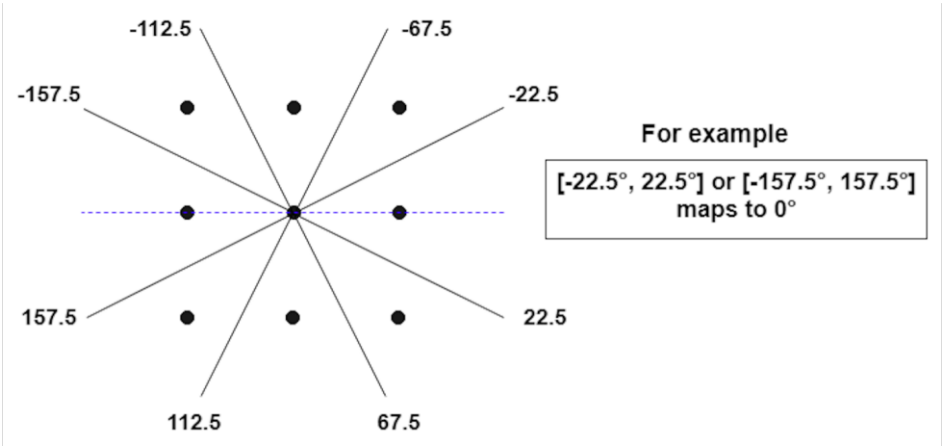


- Aplicarea unui operator derivativ 2D pentru a calcula magnitudinea și direcția gradientului și a identifica schimbările de intensitate. În funcție de precizia dorită, se pot folosi diferiți operatori:

- Sobel alculează prima derivată, utilizând un filtru trece-sus pentru a evidenția tranzițiile de intensitate.
- Laplacian – calculează a doua derivată, aplicând un filtru trece-sus mai strit, care detectează schimbările bruște de intensitate și oferă o detectare mai agresivă a imaginilor.



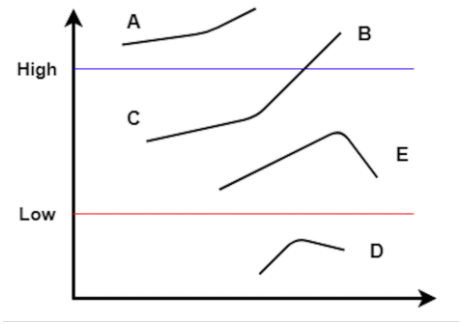
- Suprimarea non-maximală: Restrângerea pixelilor pe baza rezultatelor anterioare. Pentru fiecare pixel se realizează rotunjirea către cel mai apropiat vecin aparținând direcției gradientului (calculate anterior). Pixelul va fi păstrat doar în cazul în care punctul către care s-a realizat rotunjirea este punct de maxim, altfel, acesta va fi suprimat. Acest pas asigură un rezultat cât mai neted.



- Folosirea pragurilor de histerezis pentru a decide ce contururi sunt păstrate. O metodă comună de a determina aceste praguri este bazată pe histograma magnitudinii gradienului ce a fost calculată la pasul anterior. Cele două praguri sunt:

- Pragul superior – poate fi un procent de ex 70% - 90% din valoarea maximă a gradientului, iar orice valoare care depășește acest prag este considerată o margine puternică și este păstrată.

- Pragul inferior – de obicei este o fracțiune din pragul superior cum ar fi 30% - 50%. Pixelii cu valorile sub acest prag sunt eliminați, iar cei având valori intermediare sunt păstrați doar dacă sunt conectați la margini puternice.



2.2.3 Avantaje și Dezavantaje

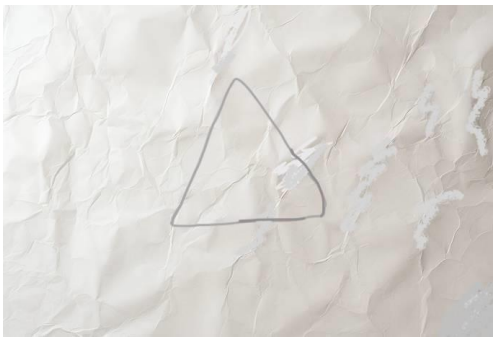
Câteva dintre avantajele folosirii operatorului Canny sunt detectarea precisă a marginilor și sensibilitatea redusă în ceea ce privește zgomotul.

Pe de altă parte, în funcție de cantitatea de detalii pe care o are imaginea, pot rezulta și răspunsuri zgomotoase în urma aplicării unui operator derivativ, ceea ce creează margini false. Cu toate că această metodă prezintă rezultate bune, o posibilă limitare este dată de timpul prelungit de execuție.

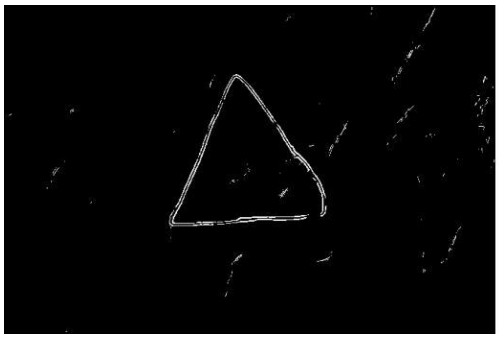
3. Comparații și Concluzii

Utilizarea operatorului Canny este recomandată în lucrul cu imagini naturale sau complexe, având zgomot ridicat, sau în sarcini de vizualizare computerizată ce necesită margini bine definite. Totuși, utilizarea acestui operator *nu* este indicată în aplicații ce rulează în timp real sau care au cerințe stricte de viteză, sau în cazul imaginilor artificiale simple, caz în care detectoarele mai rapide, precum Sobel, pot fi suficiente.

Experiment: Am luat imaginea de mai jos și am trecut-o o dată prin filtrul Sobel și o dată prin cel Canny.

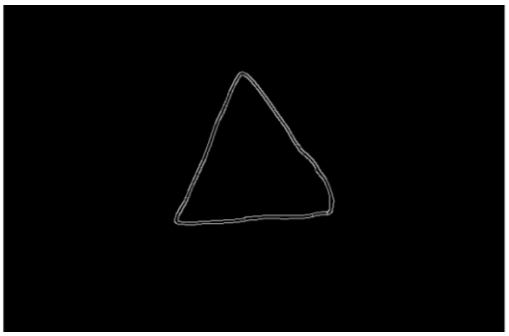


Cum imaginea are zgomot pe fundal, ne așteptăm ca filtrul Canny să ofere un rezultat exact (deoarece acesta elimină zgomotul gaussian), pe cand cel Sobel să ofere un rezultat mai aproximativ. Rezultatele au fost următoarele:



-detecție Sobel

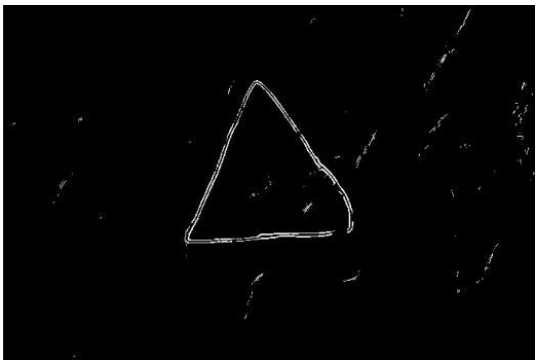
- 0.056056 secunde, sursa "sobel1.py"



-detecție Canny

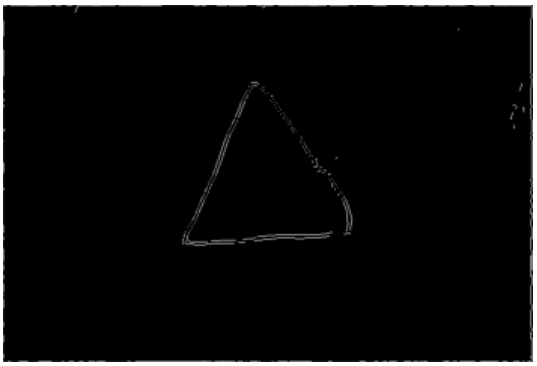
- 0.071177 secunde, sursa "canny.py"

Am testat “ground truth” și pentru coduri scrise fară a folosi funcțiile predefinite ale operatorilor, iar rezultatele au fost consistente:



- detecție Sobel

- 5.679149s, sursa "sobel_manual.py"



- detecție Canny

- 8.516952s, sursa "canny_manual.py"

După aceste rezultate, putem trage concluzia că în prezența zgomotului, operatorul Canny detecteaza mult mai clar marginile adevarate, dar Sobel este mai rapid. În funcție de prioritați și de nevoi, putem lua în considerare aceste observații și să facem o alegere potrivită.

5. Bibliografie

[1] Donald E. Knuth, „Structured Programming with *Go to* Statements", în *ACM Com- put. Surv.* 6.4 (Dec. 1974), pp. 261–301, ISSN: 0360-0300, DOI: 10.1145/356635. 356640, URL: https://doi.org/10.1145/356635.356640.

[2] Djemel Ziou and Salvatore Tabbone, Edge Detection Tehniques - An Overview.

URL: https://inria.hal.science/inria-00098446/

[3] O. R. Vincent, Clausthal University of Technology, Germany and University of Agriculture, Abeokuta, Nigeria O. Folorunso, Department of Computer Science, University of Agriculture, Abeokuta, Nigeria - A Descriptive Algorithm for Sobel Image Edge Detection

URL: https://proceedings.informingscience.org/

[4] The University of Edinburgh, Canny Edge Detector

URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm