Universitatea din București

Facultatea de Matematică și Informatică

**Detectarea Marginilor**

Zotic Melania – Anemona

Bănesaru Denisa-Georgiana

Grupa 463

2024

# Cuprins

[Cuprins 2](#_Toc157415654)

[1. IMPORTANȚA DETECTĂRII MARGINILOR 3](#_Toc157415655)

[1.1. CONTEXT APLICAȚIILOR 3](#_Toc157415656)

[1.2. REZOLVAREA PROBLEMELOR ÎN PROCESAREA IMAGINILOR PRIN DETECTAREA MARGINILOR 4](#_Toc157415657)

[2. ABORDAREA TEHNICĂ A DETECTĂRII MARGINILOR 4](#_Toc157415658)

[3. DESCRIEREA CODULUI SI DESCRIEREA PROBLEMEI 8](#_Toc157415659)

[4. TEHNOLOGII FOLOSITE 9](#_Toc157415660)

[5. REZULTATE 11](#_Toc157415661)

[6. CONCLUZII 12](#_Toc157415662)

[7. BIBLIOGRAFIE 12](#_Toc157415663)

# IMPORTANȚA DETECTĂRII MARGINILOR

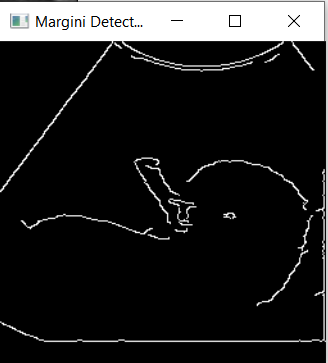
Detectarea marginilor este un pilon cheie în procesarea avansată a imaginilor, vitală pentru identificarea tranzițiilor de intensitate și culoare, facilitând astfel interpretarea detaliată a contururilor obiectelor.

## CONTEXT APLICAȚIILOR

Aplicații în Analiza Imaginilor Medicale: Detectarea marginilor este importantă în identificarea structurilor anatomice și a anomaliilor, cum ar fi tumori sau fracturi, prin analiza detaliată a imaginilor medicale.

Rolul în Recunoașterea Formelor: Această tehnică este esențială în extragerea caracteristicilor relevante ale obiectelor pentru sisteme de recunoaștere facială și interpretarea semnelor de circulație.

Importanța în Robotică și Vehicule Autonome: Detectarea marginilor este fundamentală în navigația și evitarea obstacolelor, permițând vehiculelor autonome și roboților să identifice clar contururile mediului înconjurător.



*Fig. 1 Ecografie cu ultrasunete Fig. 2 Imaginea ecografiei după aplicarea algoritmului de detectare a marginilor*

## REZOLVAREA PROBLEMELOR ÎN PROCESAREA IMAGINILOR PRIN DETECTAREA MARGINILOR

* Separarea Obiectului de Fundal: Importantă pentru identificarea clară a contururilor obiectelor.
* Segmentare Precisă: Permite o analiză detaliată și exactă a obiectelor individuale dintr-o imagine.
* Îmbunătățirea Recunoașterii Formelor: Ajută algoritmii să diferențieze eficient între diverse forme și modele într-un context vizual complex.

# ABORDAREA TEHNICĂ A DETECTĂRII MARGINILOR

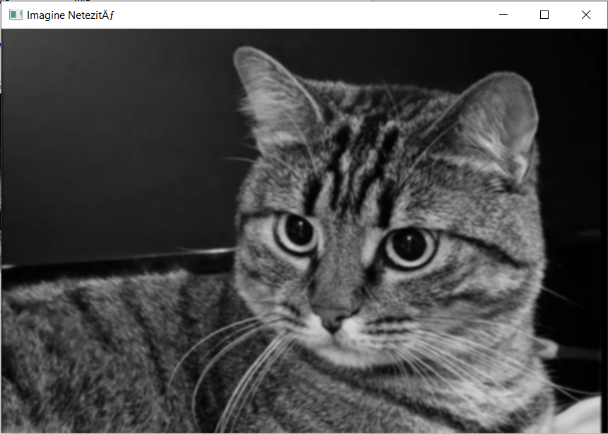
Cei mai cunoscuți algoritmi pentru detectarea marginilor:

OPERATORUL CANNY - cel mai des folosit in detectarea marginilor datorită performanței sale echilibrate. Acesta oferă o detectare precisă a marginilor, este robust la zgomot si produce margini subțiri.

**ETAPE :**

1. **Reducerea zgomotului**  - se netezește imaginea folosind un filtru Gaussian pentru a reduce zgomotul. Este un pas elementar deoarece elimina falsele margini create de zgomotul aleatoriu in imagine.

*Filtru Gaussian*



Imaginea originală Imaginea blurată cu filtru Gaussian

1. **Calcularea gradientului:** Imaginea netezită este apoi filtrată cu un nucleu Sobel atât în direcția orizontală, cât și în cea verticală pentru a obține prima derivată în direcția orizontală (Gx) și în direcția verticală (Gy). Din aceste două imagini, putem găsi gradientul marginii și direcția pentru fiecare pixel, după cum urmează:

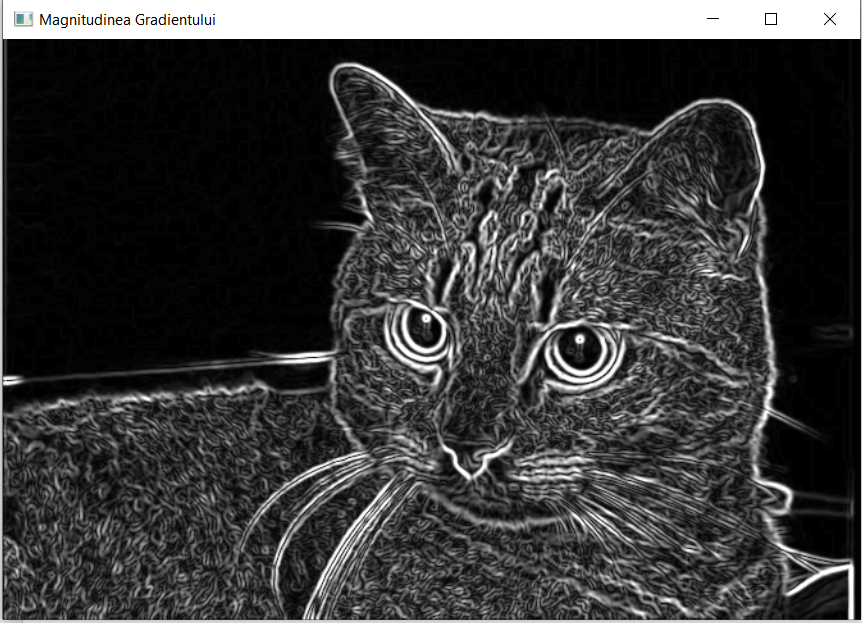
*Derivata pe orizontală*



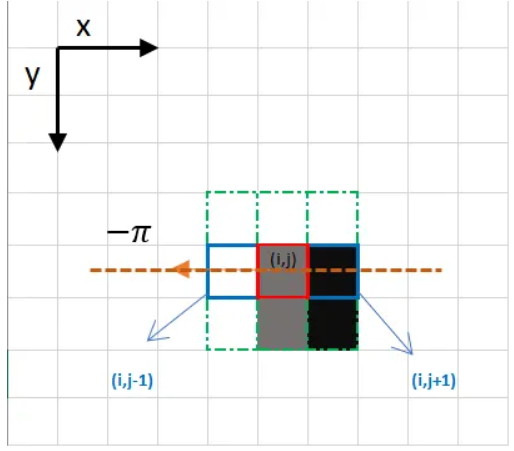
*Derivata pe verticală*



*Magnitudine(Mărimea muchiei)*

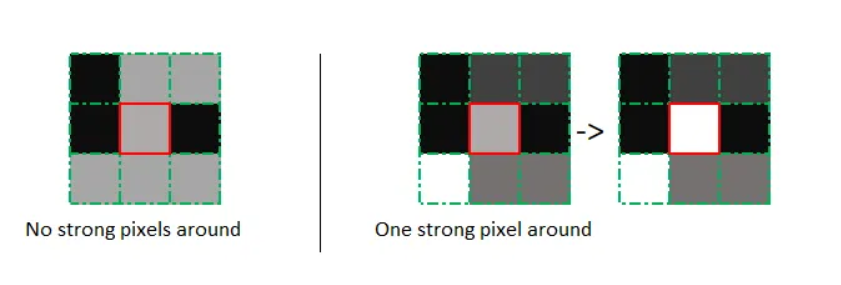


1. **Suprimarea Non-Maximelor** – acest pas reduce grosimea marginilor potențiale la grosimi de 1 pixel. Se realizează prin păstrarea doar a valorilor maxime ale gradientului și eliminarea celorlalte. Algoritmul parcurge toate punctele de pe matricea intensității gradientului și găsește pixelii cu valoarea maximă în direcțiile marginilor.

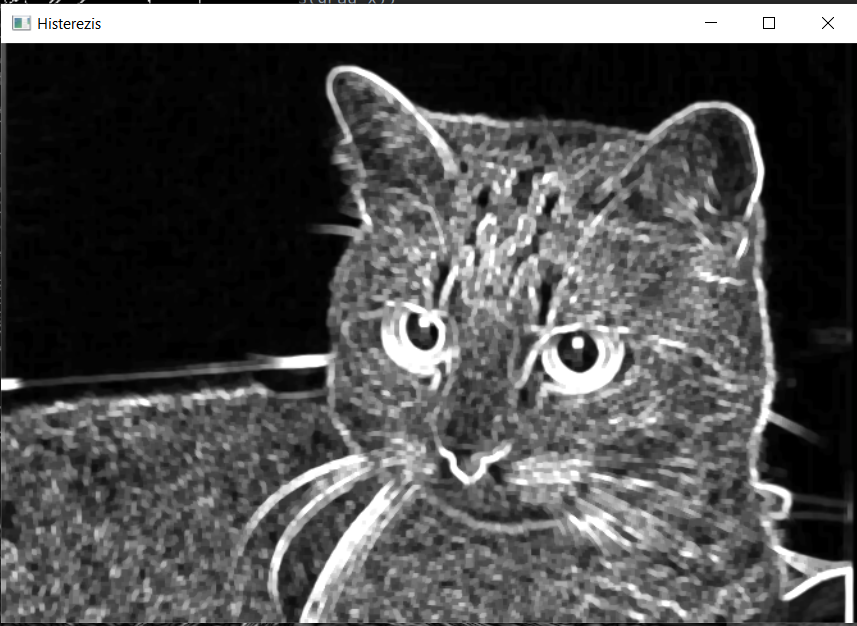


*Suprimarea Non-Maximelor*

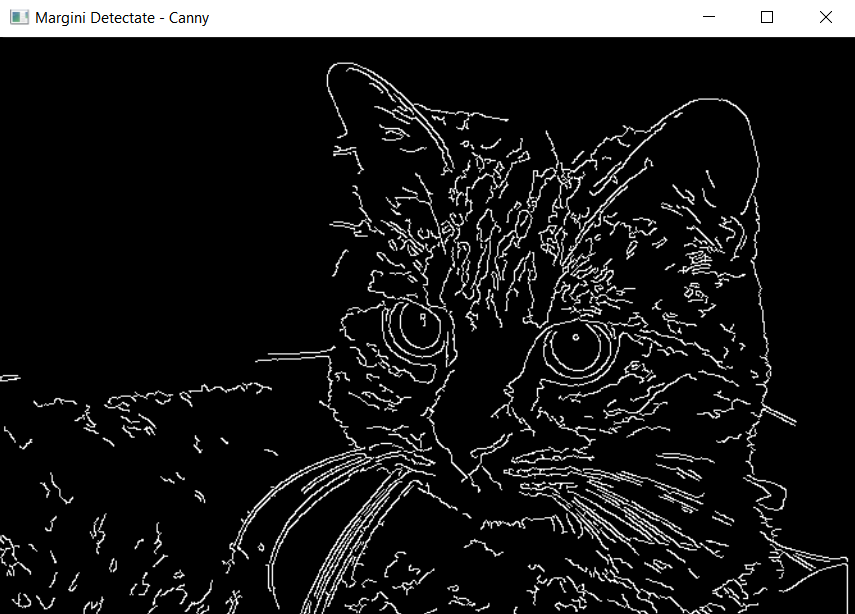
1. **Pragul Dublu:** Algoritmul aplică două praguri: unul inferior și unul superior. Punctele cu o magnitudine a gradientului mai mare decât pragul superior sunt considerate margini, iar cele sub pragul inferior sunt respinse. Punctele cu valori între aceste praguri sunt acceptate ca margini doar dacă sunt conectate la puncte puternice.
2. **Urmărirea Marginilor prin Histerezis:** consolidează detecția marginilor



Această etapă consolidează detecția marginilor, transformând pixelii slabi în margini puternice dacă sunt conectați la pixeli puternici, asigurând astfel continuitatea și claritatea marginilor în imagine.



*Imagine Histerezis*



*Imaginea finală*

OPERATORUL SOBEL - Acesta este unul dintre cei mai simpli și mai rapid de implementat algoritmi de detectare a marginilor. Sobel utilizează filtre de convoluție pentru a calcula gradientul de intensitate atât pe direcția orizontală, cât și pe cea verticală a imaginii.

OPERATORUL PREWITT - Asemenea celui Sobel, este utilizat pentru detectarea marginilor în procesarea imaginilor, aplicând o metodă distinctă de calcul al gradientului folosind un set de două nuclee (kernel-uri). Acesta reprezintă o alternativă eficientă la Sobel, generând rezultate comparabile, dar cu variații ușoare în detecția marginilor.

# DESCRIEREA CODULUI SI DESCRIEREA PROBLEMEI

Problema abordata este reprezentată de măsurarea distanțelor dintre două puncte alese de noi de pe conturul unui obiect.

În cazul nostru, obiectul poate fi reprezentat de imaginea unui organ scanat cu raze X.

Codul este compus din 3 functii importante:

def detectare\_margini\_canny(imagine):

return canny

Aceasta funcție apeleaza algoritmul Canny care primește ca parametru imaginea dorită și returnează imaginea cu contururi.

def click\_event(event, x, y, flags, param):

cv2.imshow("Imagine cu Contururi", imagine)

Funția *click\_event* ne va ajuta sa alegem cele ddouă puncte de pe conturul obiectului.

def calculare\_dimensiune(path\_imagine):

imagine = detectare\_margini\_canny(imagine\_originala)

cv2.setMouseCallback("Imagine cu Contururi", click\_event)

Funcția va primi ca parametru coordonatele punctelor și va calcula distanța euclidiana dintre cele două puncte.

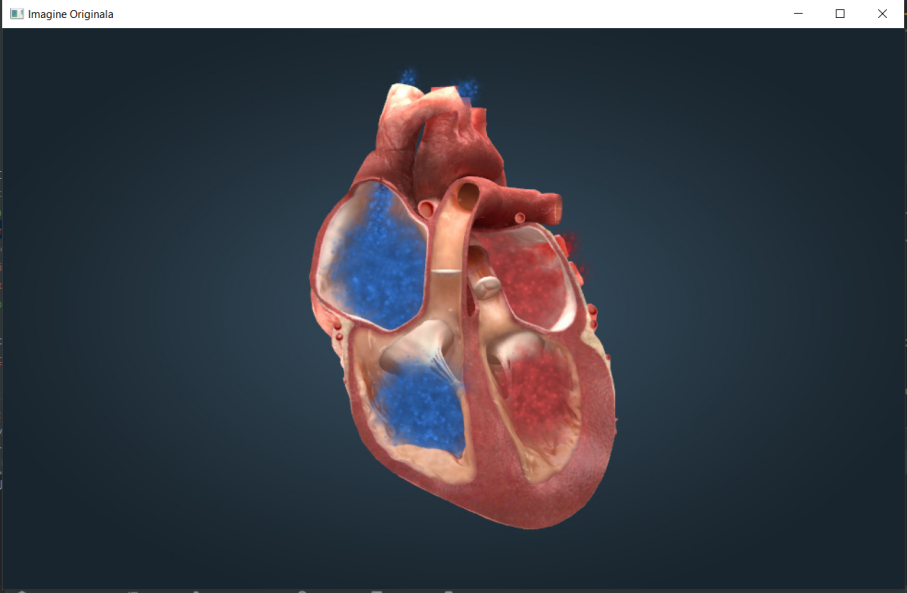
Apelarea funcției:

calculare\_dimensiune('inima.jpg')

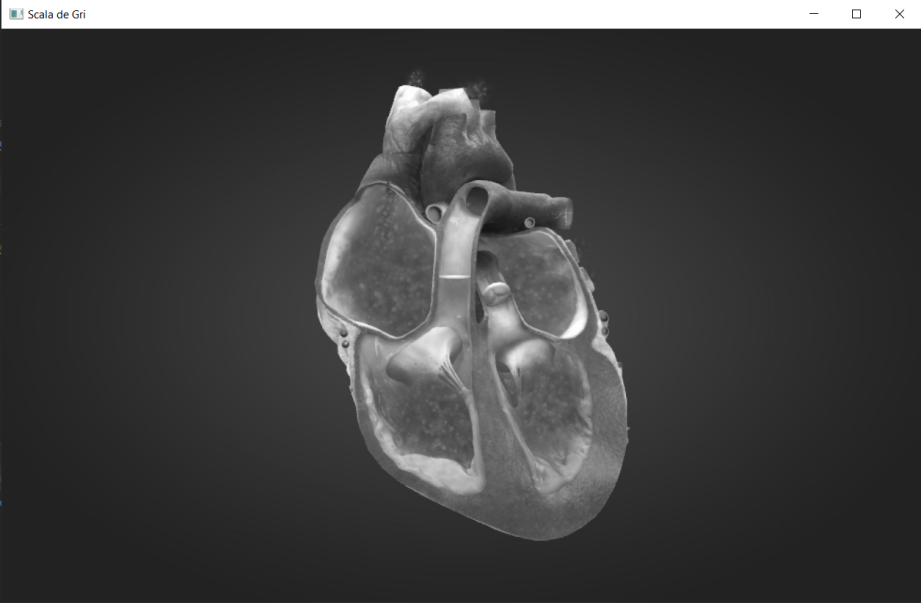
# TEHNOLOGII FOLOSITE

import cv2 - OpenCV oferă algoritmi optimizați atat clasici, cât și avansați de viziune computerizată și învățare automată.

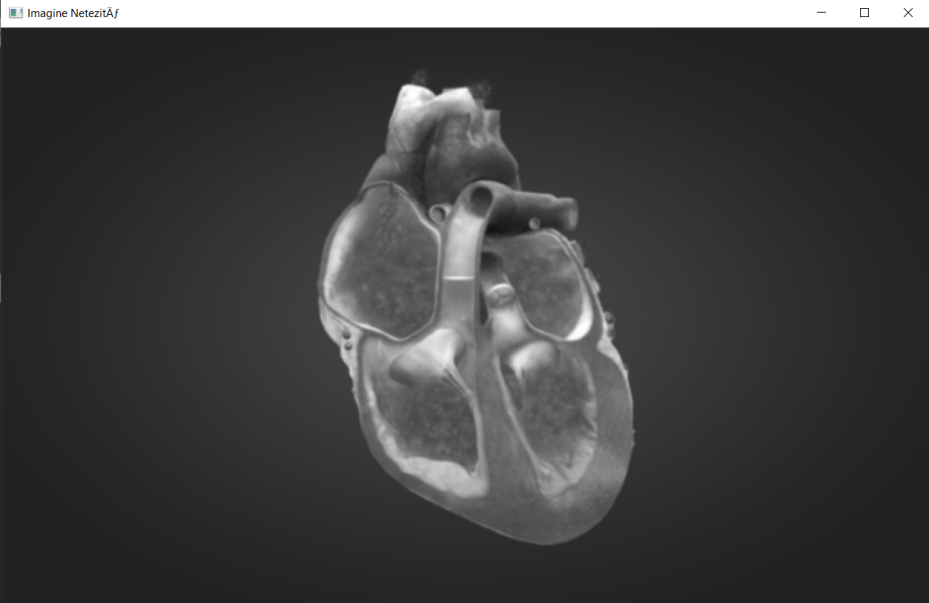
1. *cv2.imread - Citirea imaginii*



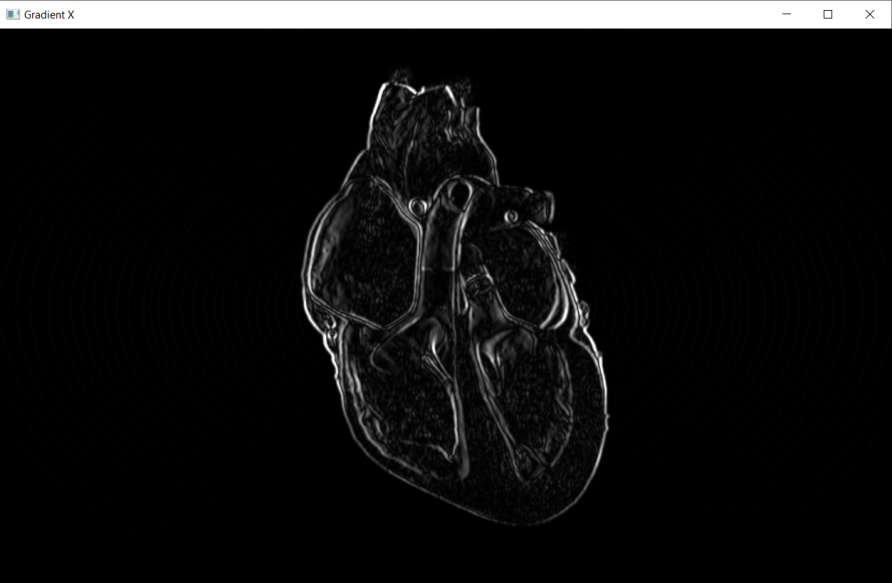
1. cv2.cvtColor - Converteste imaginea din color in gray



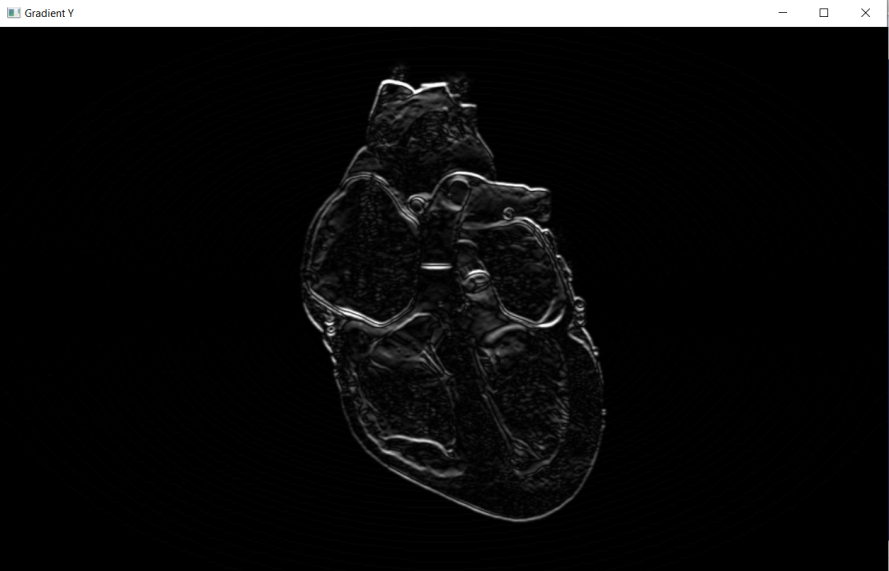
1. cv2.GaussianBlur - Estompeaza imaginea si reduce zgomotul



1. grad\_x = cv2.Sobel - Calculează derivata, gradientul intensității pe X



1. *grad\_y = cv2.Sobel - Calculează derivata, gradientul intensității pe Y*



1. cv2.cartToPolar - Calculează magnitudinea și unghiul gradientului



Pentru a obține cei mai buni parametrii, considerați ideali pentru determinarea marginilor am folosit augmentările de imagina și tunarea hiperparametrilor pentru algortimul Canny de detectare a marginilor. Am folosit un set de 30 de imagini din *Berkeley Segmentation Data Set and Benchmarks 500(BSDS500)* care este unul dintre cele mai cunoscute seturi de date pentru segmentarea și detectarea marginilor. Acesta conține imagini naturale și corespondenta lor cu segmentările realizate de om ce sunt considerate și “real-edges”.

Am definit funcțiile pentru aplicarea algoritmului Sobel și Canny, evaluarea imaginilor si definirea transformărilor de augmentare. Am definit apoi după încărcarea imaginilor si a marginii aferente un set de hiperparametri pentru algoritmul Canny si evaluez performanța algoritmului Sobel pe toate imaginile.

Algoritmul Canny este tunat folosind augmentări ale imaginilor și se evaluează performanța sa pe toate imaginile pentru fiecare set de hiperparametri. Se găsesc cei mai buni hiperparametri pentru Canny și scorul sau mediu de afișat.

Am comparat la final scorul mediu al algoritmului Sobel și scorul mediu al algoritmului Canny după găsirea celor mai buni parametri și am concluzionat că cel mai aproape de marginile detectate este Canny.

# REZULTATE

Acest scor reprezintă performanța algoritmului Sobel pentru setul de imagini dat și ground truth-uri. Scorul nu este unul foarte bun, acest fapt fiind datorat în mare de sensibilitatea la zgomot și la variația intensității luminii în imagine.



Acest scor reprezintă performanța algoritmului Canny cu cei mai buni parametrii găsiți prin tunarea acestuia. Scorul este semnificativ mai mare decât al lui Sobel . De asemenea, ne rezultă că pragul optim inferior este de 150 si cel superior de 200.



# CONCLUZII

Conform datelor și argumentelor prezentate, concluzia este că algoritmul Canny este cel mai aproape de marginile adevărate iar printre motivele care au dus la acest rezultat se enumeră: robust la zgomot, hiperparametrii optimi( prin tunarea hiperparametrilor s-au gasit valori care funcționează foarte bine pe setul de date actual), augmentarea imaginilor(rotație, flip orizontal si ajustarea luminozității și contrastului).

# BIBLIOGRAFIE

1. *Canny Edge Detection Step by Step in Python — Computer Vision -*[*https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123*](https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123) *Ultima accesare 15.01.2024*
2. *Canny Edge Detection -* [*https://docs.opencv.org/4.x/da/d22/tutorial\_py\_canny.html*](https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html) *Ultima accesare 15.01.2024*
3. *Edge Detection Algorithm of MRI Medical Image Based on Artificial Neural Network -* [*https://www.sciencedirect.com/science/article/pii/S1877050922014624?via%3Dihub*](https://www.sciencedirect.com/science/article/pii/S1877050922014624?via%3Dihub) *Ultima accesare 17.01.2024*
4. *Edge detection –* [*https://vincmazet.github.io/bip/detection/edges.html*](https://vincmazet.github.io/bip/detection/edges.html) *- Ultima accesare 17.01.2024*