Segmentarea imaginilor color bazată pe region growing

Student: Alina MACAVEI, 30234

2025

# Introducere

Problema abordată:

Scopul proiectului este să împărțim o imagine în mai multe regiuni de culori asemănătoare . Practic ne dorim ca părțile din imagine care au culori asemănătoare să fie grupate împreună.

Problema este că într-o imagine pot fi mai multe variații de culoare și zgomot (pixelii care nu se potrivesc cu restul ).Vom implementa un algoritm care să identifice ce pixel face parte din aceeași zonă și care nu.

Am folosit metoda numită region growing care pornește de la un pixel și adaugă lângă el alți pixeli asemănători în funcție de culoare. Repetăm acest proces până când toată imaginea e împărțită în regiuni bine definite.

Util pentru detectarea obiectelor sau analiza de imagini , chiar dacă obiectele din imagine nu sunt perfect delimitate sau imaginea conține zgomot.

Contextul problemei:

Imaginile color sunt folosite peste tot în viața de zi cu zi, de la camerele foto ale telefonului, până la aplicații medicale, supraveghere video, etc. Dar pentru ca un calculator să înțeleagă ce e într-o imagine, trebuie mai întâi să o analizeze și să o împartă în părți care au sens pentru el.

Aici apare nevoiadesegmentare a imaginilor. Segmentarea înseamnă să împărțim o imagine în bucăți mai mici, numite regiuni, care conțin pixeli asemănători. De exemplu, dacă într-o poză avem cerul, un copac și o casă, vrem ca algoritmul să grupeze fiecare dintre aceste obiecte în regiuni separate.

Motivare:

Motivația pentru a folosi **region growing** este că e o metodă simplă și eficientă, care funcționează bine atunci când diferențele de culoare sunt clare între regiuni. Putem decide cât de asemănători trebuie să fie pixelii ca să îi punem în aceeași zonă, folosind un prag pe care îl alegem noi**.** Dacă pixelii sunt destul de asemănători între ei, sunt puși în aceeași regiune.

Utilitate:

Metoda abordată poate fi aplicată în mai multe domenii:

* medicină, pentru a găsi și izola zone de interes din imagini precum radiografii, ecografii sau scanări RMN
* siguranța rutieră, pentru ca mașinile autonome să poată recunoaște drumul, pietonii sau alte mașini din jur
* industria fabricării de produse, pentru a verifica automat dacă o piesă este defectă sau nu
* recunoașterea obiectelor, pentru ca un algoritm să înțeleagă ce este într-o imagine: o persoană, un animal, o casă etc

# Considerații teoretice

Metodă region growing presupune să pornim de la un pixel (numit *seed*) și să adăugăm lângă el alți pixeli dacă au culori asemănătoare. Continuăm așa până nu mai găsim pixeli care să se potrivească. Apoi, căutăm un alt *seed* și repetăm procesul. La final, obținem regiuni diferite în imagine, fiecare cu propriile caracteristici de culoare.

Pentru ca segmentarea să funcționeze bine, facem câțiva pași importanți înainte:

1. Filtrarea zgomotului: imaginile pot avea zgomot (pixeli defecți care strică analiza - purici). Așa că aplicăm un *filtru Gaussian*, care face imaginea mai netedă și ajută algoritmul să nu se încurce cu valori greșite.
2. Transformarea culorilor: imaginile sunt de obicei în format RGB, dar nu e mereu cel mai bun pentru analiză. Așa că transformăm imaginea în alt spațiu de culoare,  **L\*u\*v\***, care ne ajută să vedem mai bine diferențele între culori.
3. Calculul deviației standard: măsurăm cât de mult variază culorile în imagine (cât de împrăștiate sunt valorile) , ne dă o idee despre cât de tolerant trebuie să fie algoritmul când decide dacă doi pixeli sunt asemănători.
4. Algoritmul de region growing:
   * Se pornește de la un pixel neetichetat.
   * Se compară cu vecinii lui (cei 8 din jur).
   * Dacă diferența de culoare față de regiune este mică (sub un prag T), atunci pixelul e adăugat în acea regiune.
   * Se actualizează media regiunii și se continuă procesul.
5. Postprocesare: unele regiuni pot fi prea mici sau zgomotoase. Așa că, la final, aplicăm pași de *eroziune* **și** *dilatare*, adică ștergem și umplem anumite părți pentru a curăța imaginea segmentată.

Descrierea teoretică a algoritmului implementat:

1. Filtrul Gaussian:

Filtrul Gaussian este un filtru de netezire folosit în procesarea imaginilor pentru a elimina *zgomotul* (pixeli izolați / variații bruste care nu ar trebui să fie acolo) și pentru a *blura*imaginea într-un mod natural.

A black background with white text

AI-generated content may be incorrect.

Spre deosebire de media simplă (care face o medie egală a pixelilor din jur), filtrul Gaussian *pune mai multă „greutate” pe pixelii din centru*, ceea ce face ca tranzițiile să fie mai line și imaginea să nu devină atât de plată.

**A square of numbers with black numbers on it

AI-generated content may be incorrect.**



**273 = 14 + 26 + 16 + 26 +... + 7 + 4 + 1 + 4 + 1 - împărțim pentru normalizare**

***Pixelul central* (val= 41)** are cea mai mare influență în calculul noii valori.Pixelii din jurul lui au influență mai mică (26 sau 16), în funcție de cât de departe sunt.

Se face un fel de *medie ponderată* care păstrează contururile importante și elimină detaliile inutile (zgomotul).

**A person using a telescope

AI-generated content may be incorrect.**

**Cum funcționează?**

1. Parcurgi fiecare pixel al imaginii.
2. Pentru fiecare pixel, iei o zonă de **3x3** în jurul lui.
3. Înmulțești fiecare valoare din zonă cu coeficientul corespondent din filtrul Gaussian.
4. Aduni totul și împarți la **16** (suma tuturor coeficienților).
5. A graph of a bell curve

   AI-generated content may be incorrect.Rezultatul devine noua valoare a pixelului în imaginea finală.
6. Transformarea RGB-L\*u\*v\*

L\*u\*v\* este un spațiu de culoare complex , dar potrivit pt region growing în care:

* **L** măsoară luminozitatea (Lightness)
* **u** și **v** - cromaticitatea ( adică diferențele de nuanță)

Acest spațiu este util pentru că:

* este mai potrivit pentru a *compara culori*
* diferențele percepute între culori sunt mai corecte pentru ochiul uman

D65 - Daylight 6500K -lumina naturală de la amiază unde temperatura de fculoare e de 6500 K

Pași:

A black background with white text

AI-generated content may be incorrect.1. Se normalizează valorile RGB :

A number on a black background

AI-generated content may be incorrect.2. Se transformă în spațiul XYZ folosind o matrice :

A black background with white text

AI-generated content may be incorrect.3. Se calculează L(luminozitatea):

A black background with white text

AI-generated content may be incorrect.4. Se calculează u' și v' (intermediar) :

A black background with white text

AI-generated content may be incorrect.5. Se calculează u și v finale :

Folosești valorile constante:

Vei obține:

**L** între 0 și 100, **u** între -134 și +220, **v**  între -140 și +122

Dacă vrei să salvezi imaginea înapoi în format 8-bit (valori între 0–255), OpenCV face automat următoarea conversie:

A math equation on a black background

AI-generated content may be incorrect.

1. Calculul deviației standard pe canalele u și v

Deviația standard ne spune cât de **„***împrăștiat***e**” sunt valorile dintr-un canal de culoare față de media lor.Cu alte cuvinte, dacă valorile u sau v sunt foarte diferite între ele, atunci deviația standard e mare.Dacă sunt apropiate, deviația e mică.

A screenshot of a black background with white text

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

În algoritmul de region growing, avem nevoie de deviația standard ca să stabilim cât de tolerant să fie algoritmul când compară doi pixeli.

Dacă deviația e mare, înseamnă că valorile variază mult - algoritmul va permite mai multe diferențe.  
Dacă e mică, doar pixeli foarte asemănători vor fi acceptați într-o regiune.

1. Postprocesare morfologică (eroziune și dilatare)

După ce am realizat segmentarea imaginii, este posibil ca în imaginea rezultată să apară regiuni mici, izolate, sau zgomot . Pentru a curăța imaginea segmentată și a obține regiuni mai compacte și corecte, se aplică operații morfologice pe imaginea binară:

pixeli obiect: mulțimea pixelilor de interes (asupra cărora se aplică operațiile morfologice

pixeli fundal: complementul mulțimii pixelilor de obiect

Elementele structurale :

A diagram of a structure

AI-generated content may be incorrect.

**Eroziunea:**

Eroziunea are rolul de a elimina pixelii nepotriviți de la **marginea** **regiunilor**. Practic, **regiunile devin mai mici**, iar zgomotul (regiunile mici, izolate) poate fi eliminat.

Cum funcționează:

**La fiecare pas:**

* + Dacă toți pixelii de sub elementul structural sunt 1 (obiect) → pixelul central devine 1 (păstrat ca obiect).
  + Dacă cel puțin unul dintre ei este 0 (fundal**)** → pixelul central devine 0 (eliminat => devine fundal).

**Dilatarea:**

Dilatarea are rolul de a "repara" regiunile și de a umple spațiile goale dintre părțile unei regiuni.

Cum funcționează:

**La fiecare pas:**

* + Dacă **centrul elementului structural** se suprapune cu un pixel „1” (obiect), atunci **toți pixelii** din jurul acelui pixel (acoperiți de elementul structural) vor fi transformați în „1” în imaginea rezultată.

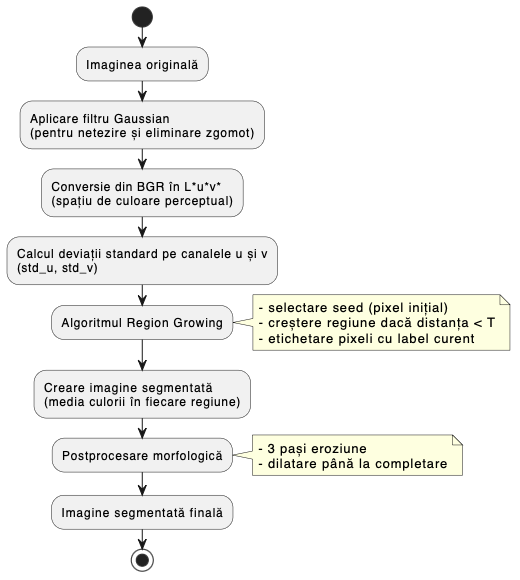
De obicei:

* Se aplică **3 pași de eroziune** pentru a elimina zgomotul
* Urmați de mai multe dilatări, pentru a reface structura regiunilor

Imaginea finală segmentată va avea regiuni:

* mai clare,
* fără pete mici izolate,
* fără goluri între regiuni,
* cu margini netede și coerente.

# Specificații de implementare



Detalii algoritmice:

* Filtrul Gaussian este implementat manual prin convoluție 5x5 cu kernel precalculat și normalizatA computer screen with numbers and symbols

  AI-generated content may be incorrect.
* Conversia BGR → Luv\* se face cu  ***cvtColor***  din OpenCV
* Deviațiile standard pentru u și v se calculează cu  ***meanStdDev***
* Pragul T este calculat astfel: ***T = scale \* sqrt(std\_u^2 + std\_v^2)***

******

* ***scale*** este introdus de utilizator și controlează granularitatea segmentării
* Distanța dintre pixeli este distanța Euclidiană în spațiul (u,v)
* Region growing folosește o coadă FIFO pentru a adăuga vecinii acceptați
* ***labels*** este o matrice CV\_32SC1 folosită pentru a marca regiunile
* ***makeSegmented*** calculează media culorilor din fiecare regiune pentru a produce o imagine segmentată coerentă
* ***postProcessLuv*** aplică 3 pași de eroziune urmate de dilatare până când nu mai există pixeli neetichetați

Ghid de utilizare:

* Se rulează aplicația executabilă
* Se selectează opțiunea 13 - Region Growing din meniu pentru segmentare
* Se alege o imagine color
* Se introduce un factor de scalare "Factor (val>0)= "
* Se afișează următoarele :
* imaginea originală : Source
* imaginea filtrată : Filtered
* imaginea cu regiuni etichetate : Labeled
* imaginea segmentată finală după postprocesare : Final

# Rezultate experimentale

Setul de date:

* Am folosit imagini BMP color din folderul de test oferit

( https://drive.google.com/drive/folders/1sFdJFAijbdoQht4N6fao\_huZyBdf88mP )

Exemple vizuale:

* Cu ***scale = 0.5*** obținem multe regiuni mici și detaliate
* Cu ***scale = 3.0*** obținem regiuni mai mari și mai netede
* ***Postprocesarea*** elimină regiuni foarte mici sau neclare și umple golurile dintre regiuni

Validare:

* Testele vizuale arată că regiuni omogene de culoare sunt bine identificate
* Contururile obiectelor mari sunt respectate (după filtrare Gaussiană)

# Concluzii

Realizări:

* Am implementat cu succes un algoritm complet de segmentare bazat pe region growing
* Am aplicat corect preprocesare (blur Gaussian, conversie culoare) și postprocesare morfologică
* Segmentarea se adaptează dinamic prin parametrul scale

Grad de împlinire:

* Toate obiectivele propuse au fost atinse
* Codul este modular, ușor de extins și reutilizat

Observații:

* Alegerea spațiului Luv\* a fost critică pentru acuratețea segmentării
* Filtrarea inițială reduce semnificativ artefactele din imagine

Direcții viitoare:

* Extindere la segmentare 3D sau video
* La funcția postProcessLuv aș putea adăuga un int max\_iterations = 100; // prevenitiv pt infinite loop la partea de dilatări succesive .
* Să identific si mai multe edge-case uri pe care să le tratez . De exemplu , ce sa fac în cazul în care are loc suprapunerea etichetelor (dacă un pixel este afectat de mai multe etichete în aceeași iterație). Totuși, în contextul aplicației, această situație este puțin probabilă datorită naturii secvențiale a procesării.
* **Optimizări posibile:**
* Paralelizarea procesului de region growing pentru a reduce timpul de execuție, în special pe imagini de dimensiuni mari.
* Utilizarea funcțiilor optimizate din OpenCV (ex. cv::dilate, cv::erode) în locul implementării manuale a operațiilor morfologice, pentru a îmbunătăți performanța.

Bibliografie:

* Material curs/laborator Procesarea Imaginilor
* OpenCV documentation: https://docs.opencv.org/
* Wikipedia: https://en.wikipedia.org/wiki/Region\_growing
* Rafael C. Gonzalez, Richard E. Woods - Digital Image Processing
* https://drive.google.com/file/d/1HNZGtbIeX6OeAzB\_lLEytH7Bw\_f84-Kv/view
* https://www.youtube.com/results?search\_query=region+growing+in+image+processing

\*\*\*\*

Menționez că , am atașat 2 variante , regionGrowingVar1.txt este prima versiune de proiect pe care am prezentat o în data de 08.05.2025 . Despre această versiune pot spune că funcționează relativ bine , însă partea de postprocesare are nevoie de îmbunătățiri , motiv pentru care am modifict funcțiile de **eroziune** si **dilatare** din varainta finală pe care o regăsiți în fișierul OpenCVApplication.cpp .

Aici avem o imagine de la prima versiune de proiect :

A screenshot of a computer screen

AI-generated content may be incorrect.

După o analiză detaliată a laboratorului "**7. Operații morfologice pe imagini binare"** am

identificat unde era problema :

În primul cod, eroziunea și dilatarea folosesc **dst = src\_labels.clone()** pentru a păstra valorile inițiale și a le modifica doar unde este necesar, în timp ce în al doilea cod, dst este inițializat cu zeros() , unde 0 reprezintă fundalul, iar valorile nenule reprezintă obiectele. În primul cod, eroziunea și dilatarea actualizează tmp și tmp\_labels direct în buclele interioare, ceea ce poate fi mai eficient, dar mai puțin clar, pe când al doilea cod separă calculul (eroded, dilated) de actualizare, oferind o structură mai modulară. Al doilea cod include verificări mai detaliate a parametrilor de intrare (segluv, labels, mean) și mesaje de eroare mai informative, precum și un contor de iterații pentru dilatare .

**REZULTATE FINALE :**

A hand on a chalkboard

AI-generated content may be incorrect.

Factor (val>0) = 0.7

A hand on a chalkboard

AI-generated content may be incorrect.

Factor (val>0) = 1

Screens screenshot of a computer

AI-generated content may be incorrect.

Factor (val>0) = 0.3

Screens screenshot of a computer

AI-generated content may be incorrect.

Factor (val>0) = 1

Screens screenshot of a computer

AI-generated content may be incorrect.

Factor (val>0) = 0.2

Screens screenshot of a computer screen

AI-generated content may be incorrect.

Factor (val>0) = 0.6

A person's hand and a person's hand

AI-generated content may be incorrect.

Factor (val>0) = 1.2

A person holding a hand

AI-generated content may be incorrect.

Factor (val>0) = 1

A screenshot of a computer screen

AI-generated content may be incorrect.

Factor (val>0) = 0.7