UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Procesare de imagini

Proiect – Superpixeli

Îndrumător: asist. Prof. Mircea Paul Mureșan

Sudent: Ciupe David-Robert

Introducere

Segmentarea superpixelilor este un proces esențial în domeniul procesării de imagini și al analizei vizuale. Este o metodă de a împărți o imagine în mai multe părți sau regiuni denumite "superpixeli". Un superpixel este o colecție de pixeli adiacenți cu caracteristici similare, cum ar fi de exemplu culoarea. Conceptul de superpixeli este esențial în îmbunătățirea eficienței algoritmilor de procesare a imaginilor și în reducerea complexității acestora, fără a pierde semnificativ din informațiile relevante.

Segmentarea superpixelilor poate fi utilizată în diverse aplicații precum segmentarea obiectelor, recunoașterea formelor, analiza texturii, detectarea contururilor, și multe altele. Un avantaj major al utilizării superpixelilor, în locul pixelilor individuali, este faptul că aceștia pot îmbunătăți precizia multor algoritmi de analiză a imaginilor.

Algoritmi

Voi enumera mai jos câțiva algoritmi pentru detecția pixelilor:

1. Simple Linear Iterative Clustering (SLIC): Acesta este probabil cel mai cunoscut algoritm de segmentare superpixel. El funcționează prin adaptarea metodei k-means la problema de segmentare superpixel.
2. Quick Shift: Quick Shift este un alt algoritm de segmentare superpixel. Spre deosebire de SLIC, Quick Shift nu presupune o formă sau dimensiune regulată a superpixelilor. El funcționează prin construirea unui arbore de estimare a densității și apoi „mutarea” fiecărui pixel către cel mai apropiat vecin cu o densitate mai mare.
3. Watershed Transform: Transformarea Watershed este o tehnică clasică de segmentare a imaginilor, care poate fi de asemenea folosită pentru detectarea superpixelilor. Ea funcționează prin tratarea imaginii ca pe un peisaj 3D, unde intensitatea pixelilor reprezintă înălțimea, și apoi „umplerea” acestui peisaj de la „văile” sale.
4. Felzenszwalb's Efficient Graph-Based Segmentation: Acest algoritm funcționează prin construirea unui graf cu pixelii ca noduri și diferențele de intensitate ca ponderi ale muchiilor, apoi segmentarea acestui graf în regiuni coerente.
5. Turbopixels: Algoritmul Turbopixels se bazează pe curgere de fluid pentru a segmenta o imagine în superpixeli. Acesta funcționează prin inițializarea unui set de „semințe” în imagine, apoi lăsarea acestora să se extindă ca o curgere de fluid până când întreaga imagine este acoperită.

Simple Linear Iterative Clustering (SLIC)

Algoritmul SLIC (Simple Linear Iterative Clustering) este o metodă populară pentru segmentarea superpixelilor într-o imagine. Acest algoritm este conceput pentru a fi atât eficient, cât și precis, generând superpixeli compacți și regulati, ceea ce este adesea util în multe aplicații de procesare a imaginilor.

Descriere generală a pasilor algoritmului SLIC:

1. **Inițializarea centrelor**: În primul rând, imaginea este împărțită în k regiuni (unde k este numărul dorit de superpixeli), și un centroid este inițializat la centrul fiecărei regiuni.
2. **Atribuirea pixelilor la superpixeli**: pentru fiecare pixel din imagine, se calculează distanța la toți centroizii din vecinătate (distanța fiind calculată în spațiul de culoare și poziție). Pixelul este atribuit grupului a cărui centroid este cel mai aproape (se face o etichetare). Vecinătatea este definită de o regiune 2S x 2S în jurul centroidului, unde S este dimensiunea aproximativă a unui superpixel (adică, latura pătratului care ar avea o arie egală cu numărul total de pixeli în imagine împărțit la numărul de superpixeli).
3. **Actualizarea centroizilor**: După atribuirea tuturor pixelilor la un superpixel, centroizii sunt actualizați ca fiind media tuturor pixelilor atribuiți acelui superpixel.
4. **Iterare**: Pașii de atribuire și actualizare sunt repetati până când algoritmul converge. Adică, până când modificările în compoziția superpixelilor între două iterații consecutive sunt sub un anumit prag. În cazul meu, am iterat algoritmul de 10 ori.

În imaginea de mai jos este prezentat un pseudocod al algoritmului SLIC:

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Implementare

Funcția main().
-Deschidem imaginea inițială și o afisăm:

-Se inițializeaza parametrii.

N – nr de pixeli din imagine.

K – nr de superpixeli.

m – factor pentru distanta.

S – marimea superpixelilor (nr de pixeli dintr-un superpixel).

nx, ny – nr de superpixeli pe directia x, y.

dx, dy – dimensiunile superpixeli(width, height).

-Se adaugă un filtru de blur pe imaginea inițială:

Acest blur ajuta la:

1. **Reducerea zgomotului**: Blur-ul poate atenua zgomotul aleatoriu din imagine, cum ar fi zgomotul de sare și piper sau zgomotul gaussian.
2. **Eliminarea detaliilor nedorite**: În unele cazuri, detaliile foarte mici din imagine pot fi irelevante pentru segmentarea în superpixeli. Prin aplicarea unui efect de blur, aceste detalii

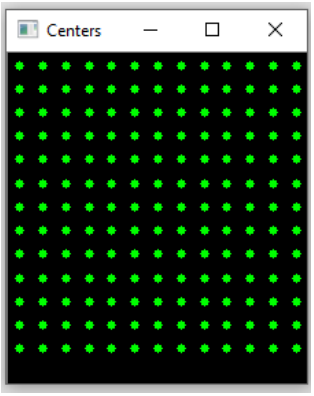
mici pot fi estompate, făcând ca algoritmul să se concentreze mai mult pe caracteristicile mai mari și mai relevante ale imaginii.

3. **Îmbunătățirea performanței algoritmului**: Blur-ul poate contribui la îmbunătățirea performanței algoritmului prin simplificarea imaginii, ceea ce poate duce la o convergență mai rapidă a algoritmului și la obținerea unor rezultate mai stabile.

-calculateSuperpixels(Mat img): Aceasta este funcția principală care implementează algoritmul de segmentare a imaginii în superpixeli.

Transformă imaginea din formatul inițial în formatul float (CV_32F) și normalizează valorile pixelilor între 0 și 1 prin împărțirea lor la 255. Convertesc imaginea din spațiul de culoare BGR (Blue, Green, Red) în spațiul de culoare Lab. Spațiul de culoare Lab este mai adecvat pentru analiza culorilor decât BGR sau RGB. Inițializarea centrelor superpixelilor.

DrawCenters(img_lab): Desenează centrele superpixelilor pe imagine:



Pentru un număr de iterații (în acest caz, 10), funcția calculează superpixelii.

a. Selecționează o fereastră de pixeli în jurul centrului. Dimensiunea ferestrei este determinată de S, care este mărimea aproximativă a superpixelului.

b. Apoi, pentru fiecare pixel din fereastra selectată, calculează distanța de la pixel la centrul superpixelului. Dacă această distanță este mai mică decât distanța anterioară stocată în matricea dists pentru acel pixel sau dacă pixelul nu a fost încă atribuit unui superpixel, atunci actualizează matricea dists cu noua distanță și matricea labels cu indicele centrului superpixelului.

c. calculează noile centre ale superpixelilor cu ajutorul funcției calculateNewCenters().

```
float dist(Point2i p1, Point2i p2)
```

Această funcție calculează distanța dintre două puncte într-un spațiu bidimensional care conține atât caracteristici spațiale, cât și de culoare. Fiecare punct este reprezentat de poziția sa pe imagine (componentele x și y) și culoarea sa în spațiul de culoare Lab (componentele L, a, b).

$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2},$$

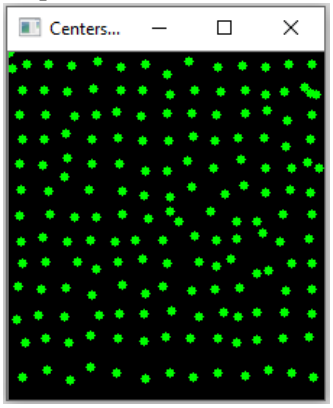
```
void calculateNewCenters()
```

Aceasta este o funcție pentru a recalcula pozițiile centrelor superpixelilor după fiecare iterație a algoritmului.

În următoarele două bucle for se parcurg toți pixelii imaginii. Pentru fiecare pixel, se obține eticheta superpixelului din matricea labels. Poziția pixelului este adăugată la poziția curentă a centrului superpixelului corespunzător în vectorul new_centers și se mărește numărul de pixeli al acelui superpixel în vectorul counts.

În următoarea buclă for, fiecare centru al superpixelului din new_centers este împărțit la numărul de pixeli al acelui superpixel pentru a obține poziția medie. Acesta devine noua poziție a centrului superpixelului.

După cele 10 iterari, acestea vor fi noile poziții ale centrelor:

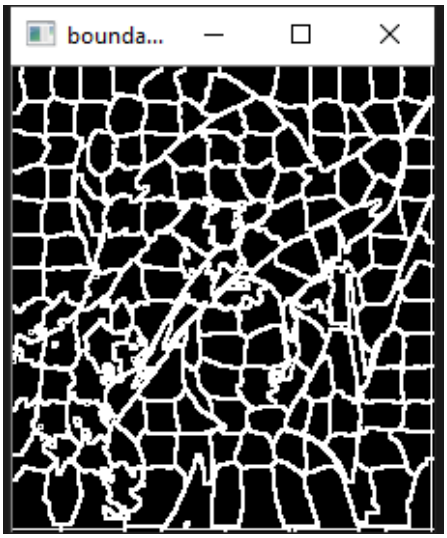



```
void calculateBoundaries(Mat& grad)
```

Această funcție este responsabilă pentru calcularea și desenarea granițelor dintre superpixeli.

Se aplică filtrul Sobel pe matricea labels_float. Filtrul Sobel este folosit în general pentru detectarea marginilor într-o imagine. În urma aplicării filtrului Sobel, avem acum două matrici care descriu modul în care intensitatea pixelilor se schimbă în ambele direcții. Dar pentru a obține o imagine a granițelor, avem nevoie de o singură valoare pentru fiecare pixel care descrie măsura schimbării de intensitate în acel punct. Aceasta este obținută prin combinarea gx și gy într-o singură matrice grad folosind funcția magnitude(gx, gy, grad). Aceasta calculează radicalul sumei pătratelor gx și gy pentru fiecare pixel.

Dupa calcularea marginilor superpixelilor aceasta este imaginea:



```
Mat colorSuperpixels(Mat img)
```

Acesta este un cod care colorizează fiecare superpixel cu media culorilor pixelilor săi. Calcularea sumei culorilor în fiecare superpixel. Calcularea culorii medii pentru fiecare superpixel. Crearea și colorarea imaginii de ieșire.

După colorarea fiecarui superpixel cu culoarea medie imaginea devine:

Bibliografie

https://infoscience.epfl.ch/record/177415?ln=en

https://www.youtube.com/watch?v=uDhcpwxCufU

https://darshita1405.medium.com/superpixels-and-slic-6b2d8a6e4f08