Universitatea Tehnică din Cluj-Napoca

Facultatea de Automatică și Calculatoare

Departamentul Calculatoare

Segmentarea imaginilor color

bazată pe K-means

Student: Lorand Sarkozi

Grupa: 30234

anul 2024

1. **Introducere**

Segmentarea imaginilor color bazată pe algoritmul K-means este o metodă populară în procesarea imaginilor, utilizată pentru a segmenta și grupa pixelii imaginii în clustere sau regiuni omogene în funcție de asemănarea culorilor lor. Această tehnica este esențială într-o varietate de aplicații, de la recunoașterea obiectelor și extragerea caracteristicilor în analiza imaginilor până la comprimarea datelor și îmbunătățirea calității imaginilor.

**1.2 Descrierea problemei abordate**

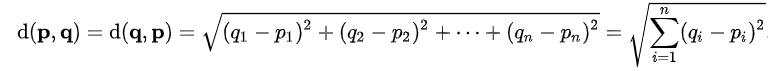
Algoritmul K-means este o abordare eficientă pentru segmentarea automată a imaginilor. Acesta utilizează un proces iterativ pentru a grupa pixelii în K clustere, unde K este numărul specificat de utilizator. Prin atribuirea fiecărui pixel la cel mai apropiat centroid al clusterului și recalcularea centroidului bazat pe media pixelilor atribuiți, algoritmul K-means poate fi utilizat pentru a segmenta imaginile în regiuni cu culori similare.

* 1. **Contextul problemei / Motivare / Utilitate**
* **Recunoaștere de obiecte:** Segmentarea imaginilor este un pas esențial în recunoașterea și localizarea obiectelor în imagini. Algoritmul K-means poate fi utilizat pentru a separa obiectele de fundal în imagini, facilitând astfel procesul de recunoaștere a obiectelor.
* **Îmbunătățirea calității imaginilor:** Prin segmentarea imaginilor în regiuni omogene, algoritmul K-means poate fi utilizat pentru a elimina zgomotul și artefactele din imagini, îmbunătățind astfel calitatea și claritatea acestora.
* **Comprimarea datelor:** Segmentarea imaginilor poate fi utilizată pentru a reduce dimensiunea datelor în aplicații de stocare și transmitere a imaginilor. Prin identificarea și reprezentarea eficientă a regiunilor semnificative ale imaginii, algoritmul K-means poate contribui la comprimarea datelor fără a compromite prea mult calitatea imaginii.
* **Analiza texturii și a modelelor:** Segmentarea imaginilor poate facilita analiza texturii și a modelelor în imagini. Prin gruparea pixelilor în regiuni cu texturi sau modele similare, algoritmul K-means poate fi utilizat pentru a extrage caracteristici semnificative și pentru a identifica tipare în imagini.

1. **Considerații teoretice**

Algoritmul K-means e o tehnica de clusterizare utilizata pentru a segmenta seturi de date in K grupuri numite clustere, unde K este valoarea data de catre utilizator. Procesul incepe prin selectarea aleatorie a K centroizi initiali pe ecran, fiecare reprezentand un cluster. Apoi fiecare punct din setul de date este atribuit clusterului cu cel mai apropiat centroid in functie de distanta euclidiana ca de exemplu.

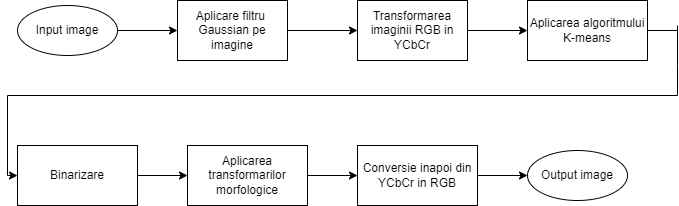
Dupa ce toate punctele sunt atribuite clusterelor, centroizii vor fi recalculate ca fiind valorile medii a punctelor din fiecare cluster. Acest process va fi recalculat pana cand nu vor mai exista schimbari semnificative in legatura cu pozitia centroizilor. Sau putem da de la noi un numar maxim de iteratii, iar cand acesta este ajuns algoritmul se va opri.



1. **Specificatii si implementare**

Pentru a putea sa cream imaginea dorita folosind K-means, trebuie sa respectam o serie

de etape cu ajutorul carora vom ajunge la imaginea dorita.



**3.1 Aplicarea filtrului Gaussian**

Pentru a aplica filtrul Gaussian pe imagine, am ales sa folosesc 3 metode.

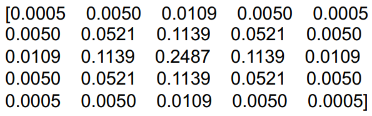
**1).void FilterCreation()**

* Aceasta functie primeste ca si input o matrice de 5x5, care va fi filtrul nostrum Gaussian, si cu ajutorul formulei afisate in cursul 5 am populat aceasta imagine. Avem o variabila sigma care este deviatia standard, care va accentua cat de blurry va fi imaginea.

**A graph of a function

Description automatically generated**

* Aplicand aceasta formula cu un factor de deviatie standard de sigma = 0.8, algoritmul nostru ne va genera matricea urmatoare:



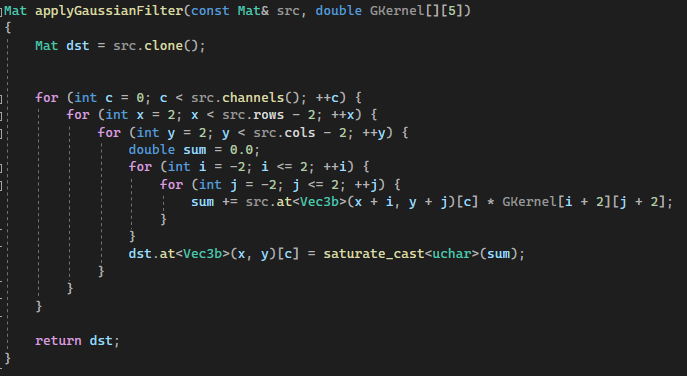
* La final normalizam valorile matricii
* Am implementat functia in acest fel:

A computer screen with many numbers and symbols

Description automatically generated with medium confidence

**applyGaussianFilter()**

* Functia primeste ca si input imaginea, si matricea filtru 5x5
* Se parcurg pixelii imaginii cu exceptia pixelilor de pe margine(de aceea incepem cu x = 2 -> src.cols – 2; y = 2 - > src.rows – 2).
* Pentru fiecare canal de culoare (c) din imaginea de intrare, se calculează suma ponderată a pixelilor dintr-o vecinătate de 5x5 folosind matricea ca filtru Gaussian. Această sumă ponderată se aplică pentru a obține valoarea pixelului de ieșire (dst) pentru fiecare canal.



**testGaussianFilter()**

* In aceasta functie cream filtrul Gaussian si il aplicam pe imaginea noastra folosind applyGaussianFilter()

A screen shot of a computer program

Description automatically generated

**3.2 Transformarea imaginii RGB in YCbCr**

* Pentru transformarea imaginii din RGB in YCbCr am folosit functia cvtColor.

**3.3 Aplicarea algoritmului K-means**

* Algoritmul K-means se bazeaza pe mai multe etape. In prima etapa trebuie sa ne alegem centroidele, eu am ales sa folosesc histograma si sa aleg K maxime locale, unde K este un numar dat de utilizator pentru numarul centroidelor.

**A computer code on a black background

Description automatically generated**

* Mai departe, vom apela functia k-means la care ii trimitem numarul centroidelor, numarul iteratiilor , imaginea, labels-ul, o matrice de valori, si centroidele, un vector de culori.
* Vom avea in interiorul functiei un for loop pe care il vom parcurge de I ori unde I este numarul iteratiilor, sau pana cand centroidele sunt in locurile in care converge si va da aceleasi valori ori de cate ori iteram.

**A screen shot of a computer program

Description automatically generated**

* In interiorul loop-ului vom asigna fiecare pixel la un centroid formand cluster-ele.

**A screen shot of a computer code

Description automatically generated**

* Dupa ce le-am assignat, vom da update la locul centroidelor, ca sa gasim locul ideal pentru acestea(adica culoarea potrivita).

**A computer screen shot of a code

Description automatically generated**

* La final cream un label cu valorile fiecarui pixel, niste tag-uri ale culorilor.

**3.4 Aplicarea transformarilor morfologice**

**3.4.1 Aplicarea eroziunii pe imagine color**

**Functia erodeLabels()**

* Conceptul acestei functii este sa facem o eroziune pe imagine astfel incat sa scapam de zgomote. Vom avea pentru label-ul pentru pixeli erodati valoarea -1. Cand vecinul unui pixel nu e de aceeasi culoare si nu e nici erodat, atunci pixelul current va fi si el erodat.
* Pentru vizitarea vecinilor ne folosim de functia elemStruct

**A computer screen with numbers and symbols

Description automatically generated**

**A computer screen shot of a program code

Description automatically generated**

**3.4.2 Aplicarea dilatarii pe imagine color**

**Functia dilateLabels ()**

* Ne folosim tot de labels, parcurgem matricea labels si cand gasim un pixel vecin care are -1 atunci si pixelul acela va lua culoarea pixelului current.

**A computer screen shot of a program code

Description automatically generated**

**4.Rezultate experimentale**

Am testat algoritmul k-means pe imaginile de test.

* Pentru imaginea cu fructi, pentru 8 centroizi, si 17 iteratii mi-a dat imaginea urmatoare:

A screenshot of a computer screen

Description automatically generated

* Pentru mana ridicata langa tabla, pentru 3 centroizi si 8 iteratii(fara noise cancelling):

A hand on a chalkboard

Description automatically generated

**Concluzii**

* Ce am mai putea inbunatatii la algoritmul K-means, este alegerea centroizilor de start. Am putea inpartii histograma pe K parti, iar din fiecare partes a alegem culoarea predominanta si sa o setam ca si centroid.

**Bibliografie**

[**https://en.wikipedia.org/wiki/K-means%2B%2B**](https://en.wikipedia.org/wiki/K-means%2B%2B)

[**https://drive.google.com/file/d/1Iedh22vPmnclnwT3xcKBnIUQ7QPoctPP/view**](https://drive.google.com/file/d/1Iedh22vPmnclnwT3xcKBnIUQ7QPoctPP/view)

[**https://docs.opencv.org/3.4/de/d4d/tutorial\_py\_kmeans\_understanding.html**](https://docs.opencv.org/3.4/de/d4d/tutorial_py_kmeans_understanding.html)