# **Raportul tehnic sub forma unui articol – Split and Merge pentru segmentarea imaginilor**

**Membrii echipei**

**Manager**: Farcas Rares *(e-mail: raresfarcas68@gmail.com)*

**Dezvoltator**: Lucan Valentin Alexandru (*e-mail: alexlucan2000@gmail.com)*

**Nume îndrumător**: Camelia Costina Florea

**Data predării proiectului (zi/lună/an)**:

## Sinteza lucrării (Abstract)

Lucrarea realizată are ca scop aplicarea unui algoritm care realizează o segmentare de tip “divizare si unificare” asupra unei imagini pe nivele de gri. Parametrii pentru segmentare, adică definirea uniformității într-o regiune, alături de imaginea pe nivele de gri reprezintă datele de intrare. Prin implementarea algoritmului de “divizare si unificare” si utilizând aceste date de intrare se dorește ca datele de ieșire să conțină imaginea împreună cu histograma sa înainte si după aplicarea algoritmului. Algoritmul are ca scop împărțirea imaginii în mai multe regiuni, dacă aceste regiuni obținute nu sunt omogene, se continuă împărțirea până când se obțin regiuni omogene. Regiunile similare se unifică, iar apoi se divizează regiunile care nu au fost unificate. Imaginea segmentată conține mai multe regiuni, pixelii fiecărei regiuni având aceeași intensitate, culoare sau textură. Pentru implementarea algoritmului se utilizează limbajul de programare Python utilizat în mediul de dezvoltare Google Colab.

## Introducere

Segmentarea imaginilor se bazează pe extragerea caracteristicilor din imagini, si anume determinarea zonelor din imagine care conțin pixeli cu caracteristici similare , culoare sau textură. Acesta este un proces care implică partiționarea unei imagini in mai multe astfel de regiuni, verificând un anumit criteriu (ex.: culoarea). Unirea acestor regiuni trebuie sa restabilească imaginea inițială. De regulă segmentarea urmărește extragerea, identificarea sau recunoașterea unui anumit obiect dintr-o imagine.

În general metodele de segmentare pot fi divizate în patru categorii:

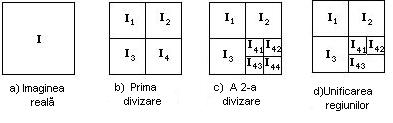
* segmentarea prin divizare
* segmentarea prin divizare si unificare
* segmentarea prin extindere
* Segmentarea prin extindere si unificare

Segmentarea poate fi implementata utilizând metodele: ce se bazează pe analiza intensității fiecărui pixel în parte, bazate pe analiza regiunilor sau bazate pe muchii.

## Fundamentare teoretica

Algoritmul „split & merge” este o procedură iterativă care include în fiecare iterație divizare și unificare. Se pleacă de le regiuni de mărime medie, de exemplu pătrate de mărime fixă, se unifică regiunile similare și apoi se divizează regiunile care nu au fost unificate :

* dacă o regiune este neomogenă (P(R) = FALSE ), ea este divizata în 4 regiuni.
* dacă două regiuni adiacente Ri, Rj sunt similare (P(Ri U Rj) = TRUE ), ele sunt unificate.
* algoritmul se termină atunci când nu mai este posibilă nici divizarea, nici unificare.



Figură Principiul "Split & Merge"

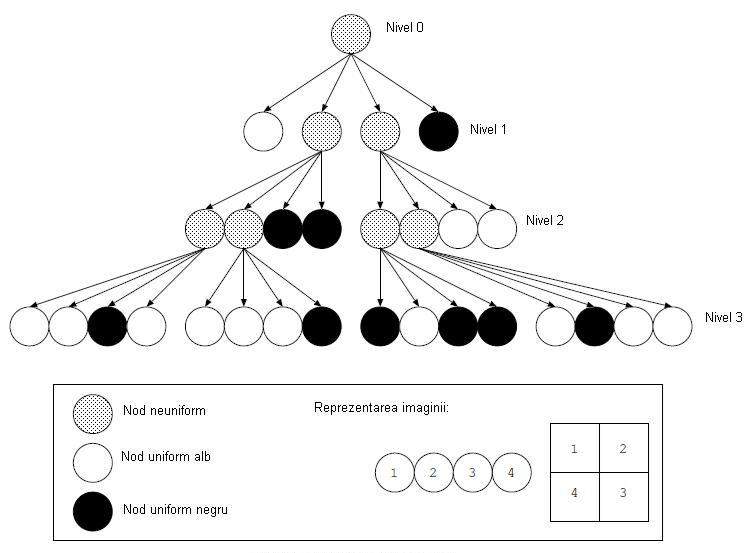
Algoritmul este recursive, dacă regiunea curentă nu este omogenă, ea este divizată în 4 și procedura este apelată recursiv. Dacă regiunea este omogenă, se calculează media sa și se compară cu mediile regiunilor vecine (pe cele 4 laturi ale regiunii), verificând posibilele unificări. Astfel, daca diferența dintre media regiunii curente si media unei regiuni adiacente este sub o valoare de prag, atunci cele doua regiuni se unifica: se calculează media valorilor pixelilor celor doua regiuni si se atribuie pixelilor celor doua regiuni in imaginea de ieșire. Daca nici o unificare nu este posibila, se atribuie pixelilor regiunii din imaginea de ieșire valoarea medie a pixelilor săi, dacă mai multe unificări sunt posibile, se alege cea mai bună, adică unificarea cu regiunea vecină care are valoarea medie cea mai apropiată de valoarea medie a regiunii curente.



Figură 2 Exemplificarea procesului de segmentare prin divizare si contopire a imaginii

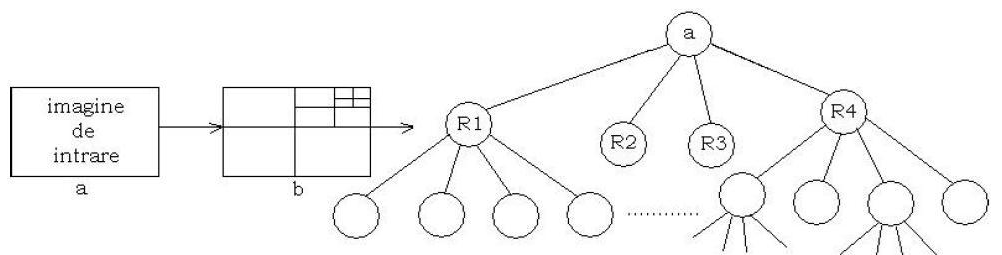
Procesul de segmentare este un pas extrem de important pentru extragerea de informații despre calitatea imaginii. Fiecare regiune segmentată este conectată la vecinii săi si oferă informații calitative, cum ar fi mărime, culoare sau formă.

Divizarea recursivă a imaginii în regiuni din ce în ce mai mici utilizând algoritmul de segmentare prin divizare recursivă duce la o reprezentare de tip piramidală.



Figură Structura de tip piramidală

Această structura piramidală poate fi corelata cu reprezentarea printr-un arbore cuadric. Astfel in structura reprezentată fiecare nod este de tip părinte și are patru fii corespunzător celor patru regiuni divizate.



Figură Structura unui arbore cuadric

## Implementarea soluției adoptate

Mediul de programare in care s-a realizat implementarea algoritmului este Google Colaboratory, iar librăriile utilizate sunt:

* cv2 pentru a citi imaginea pe nivele de gri
* matplotlib.pyplot pentru a afisa imaginea originala si cea rezultata in urma execuției algoritmului, împreuna cu histograma specifica fiecărei imagini
* numpy pentru a utiliza funcțiile de array

Algoritmul ales pentru realizarea segmentării “Split and Merge” este un algoritm prin divizare recursivă. Acesta pornește de la premisa că întreagă imagine este omogenă. Dacă acest lucru nu este adevărat, imaginea este divizată în patru regiuni, care mai departe acestea vor deveni regiuni părinte în structura piramidală ce urmează a fi constituită. Procedura de divizare este aplicată recursiv până când se obțin numai regiuni omogene.

Explicarea pașilor implementați:

1. .Verificarea omogenități pentru regiunea curenta.

 if omogen(a, N1, M1, N2, M2, T)==0

1. Dacă regiunea curentă nu este omogenă, ea este divizată în patru, iar procedura este aplelată recursiv:

Se divide regiunea in 4 zone egale si se reapelează funcția divideReg cu 8 parametrii (imaginea de intrare, imaginea de ieșire, coordonata X al coltului de sus, coordonata Y al coltului de sus, coordonata X al coltului de jos, coordonata Y al coltului de jos, pragul pentru criteriul de omogenitate si numărul de regiuni rezultate din divizarea recursiva.

divideReg(a, b, N1, M1, (N1+(N2-N1) / 2),(M1+(M2-M1) / 2), T, N)

divideReg(a, b, (N1+(N2-N1) / 2), M1, N2,(M1+(M2-M1) / 2), T, N)

divideReg(a, b, N1, (M1+(M2-M1) / 2), (N1+(N2-N1) / 2), M2, T, N)

divideReg(a, b, (N1+(N2-N1) / 2), (M1+(M2-M1) / 2), N2, M2, T, N)

1. Dacă regiunea este omogenă, se calculează media sa şi se compară cu mediile regiunilor vecine

Dacă diferența dintre media regiunii curente si media unei regiuni adiacente este sub o valoare de prag prestabilită, atunci cele două regiuni se unifică: se calculează media valorilor pixelilor celor două regiuni și se atribuie pixelilor celor două regiuni în imaginea de ieșire

def omogen(a, N1, M1, N2, M2, T):

    Lmax=255

    max = 0

    min = Lmax

    for i in range(int(M1), int(M2)):

        for j in range(int(N1), int(N2)):

            if (a[i][j]< min).any():

                min = a[i][j]

            if (a[i][j]> max).any():

                max = a[i][j]

    if (np.array(max-min) < np.array(T)).any()  :

        return 1

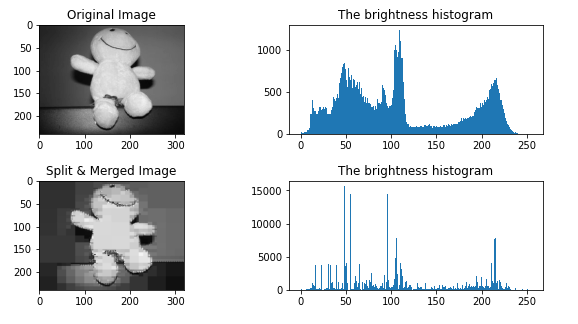
    else:

        return 0

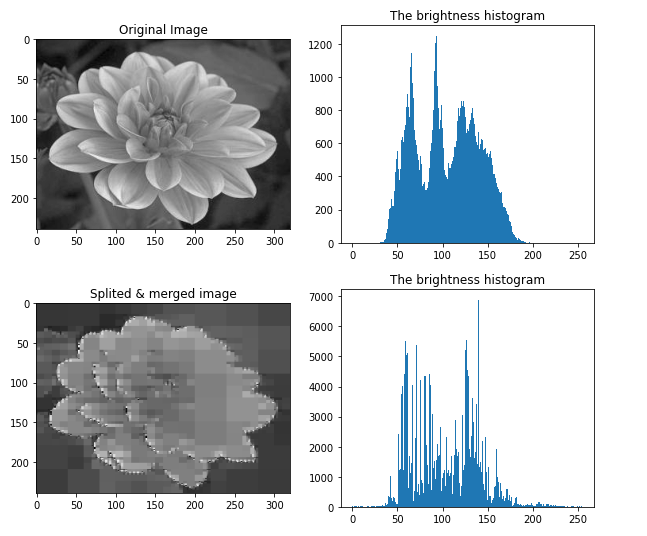
1. Dacă nici o unificare nu este posibilă, se atribuie pixelilor regiunii din imaginea de ieșire valoarea medie a pixelilor săi
2. Dacă mai multe unificări sunt posibile, se alege cea mai bună, adică unificarea cu regiunea vecină care are valoarea medie cea mai apropiată de valoarea medie a regiunii curente.

## Rezultate experimentale

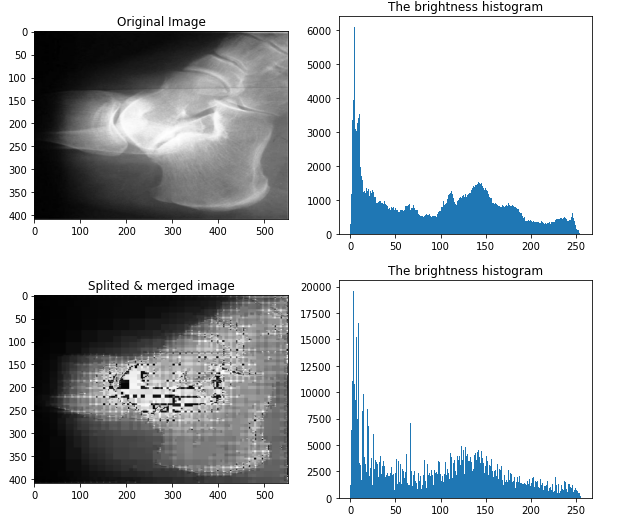
In urma aplicării algoritmului se poate observa pe histograma imaginii ca numărul de pixeli de aceeași culoare creste drastic si este redus numărul de nuanțe din imagine.



Figură Rezultatul aplicarii algoritmului pentru imaginea smiley.jpg



Figură Rezultatul apligarii algoritmulului cu un prag T=60 pentru imaginea floare\_sa\_gr\_fc.jpg



Figură Rezultatul apligarii algoritmulului cu un prag T=20 pentru imaginea image001.jpg

## Concluzii

In concluzie, algoritmul de divizare si unire consta in unirea unor regiuni învecinate in urma segmentării imaginii in funcție de un criteriu de omogenitate, astfel se obțin partiții din imagine care au toți pixelii de aceeași culoare. Acest algoritm ar putea fi utilizate de exemplu pentru separarea obiectelor de fundal sau pentru a identifica anumite obiecte dintr-o imagine. Însă algoritmul are un dezavantaj major: poate produce regiuni adiacente similare, care nu corespund aceluiași nivel se divizare.

## Bibliografie

* 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and

Informatics -Marian Bakoš - Active Contours and their Utilization at Image Segmentation – 2007

## https://pyimagesearch.com/2021/01/23/splitting-and-merging-channels-with-opencv/\

## https://www.projectpro.io/recipes/split-and-merge-images-channels-opencv

## https://www.geeksforgeeks.org/splitting-and-merging-channels-with-python-opencv/

## ImageProcLab1\_ToDo.ipynb – Laborator 1 PNI

## Prelucrarea numerica a imaginilo Cap.8 Analiza de imagini si recunoasterea formelor

## Anexe

Import Google Drive to project

from google.colab import drive

drive.mount('/content/drive')

Import libraries

import cv2

import matplotlib.pyplot as plt

import numpy as np

Upload image

InputIm\_FileName="/content/drive/MyDrive/floare\_sa\_gr\_fc.jpg"

InOriginalImg=cv2.imread(InputIm\_FileName,0)

Implementare algoritm, afisare imagine originala si imagine divizata si unificata impreuna cu histograma

emptyimg = np.zeros([240,320,3],dtype=np.uint8) #definire imagine goala

emptyimg.fill(255) # or img[:] = 255

N=0;

def divideReg(a, b, N1, M1, N2, M2, T, N):

# a = input img

# b = output img

# N1,M1 = coordonate colt stanga sus

# N2,M2 = coordonate colt dreapta jos

# T =  pragul folosit in criteriul de omogenitate

# N =   memoreaza numarul de regiuni rezultate din divizarea recursiva

    if omogen(a, N1, M1, N2, M2, T) == 0 and (N2-N1) > 1 and (M2-M1) >1 :

        # dividem regiunea in 4 zone egale si apelam functia recursiv pentru cele 4 zone

        divideReg(a, b, N1, M1, (N1+(N2-N1) / 2),(M1+(M2-M1) / 2), T, N)

        divideReg(a, b, (N1+(N2-N1) / 2), M1, N2,(M1+(M2-M1) / 2), T, N)

        divideReg(a, b, N1, (M1+(M2-M1) / 2), (N1+(N2-N1) / 2), M2, T, N)

        divideReg(a, b, (N1+(N2-N1) / 2), (M1+(M2-M1) / 2), N2, M2, T, N)

    else:

      # regiunea este omogena

        N += 1

        s = 0

#calcularea mediei pentru regiunea omogena

        for i in range(int(M1), int(M2)):

            for j in range(int(N1), int(N2)):

                s+=  a[i][j]

        s = s / ((N2-N1)\*(M2-M1))

#asignarea valorii medii pentru regiunea omogena

        for i in range(int(M1), int(M2)):

            for j in range(int(N1), int(N2)):

                b[i][j] = s

def omogen(a, N1, M1, N2, M2, T):

    Lmax=255

    max = 0

    min = Lmax

    for i in range(int(M1), int(M2)):

        for j in range(int(N1), int(N2)):

            if (a[i][j]< min).any():

                min = a[i][j]

            if (a[i][j]> max).any():

                max = a[i][j]

    if (np.array(max-min) < np.array(T)).any()  :

        return 1

    else:

        return 0

outputimg = emptyimg

divideReg(InOriginalImg,outputimg,0,0,320,240,80,N)

plt.figure(figsize=(10,2))

plt.subplot(1,2,1)

plt.imshow(InOriginalImg,cmap='gray')

plt.title('Original Image')

plt.subplot(1,2,2)

plt.hist(InOriginalImg.ravel(),256,[0,255])

plt.title('The brightness histogram')

plt.show()

plt.figure(figsize=(10,2))

plt.subplot(1,2,1)

plt.imshow(outputimg,cmap='gray')

plt.title('Split & Merged Image')

plt.subplot(1,2,2)

plt.hist(outputimg.ravel(),256,[0,255])

plt.title('The brightness histogram')

plt.show()