Mazilu Gabriela
334AA

Aplicatii Web cu suport Java

Documentatie

Logarithmic Gray Level Transform

Mazilu Gabriela
Grupa 334AA

Mazilu Gabriela
334AA

1. Introducere

În cadrul proiectului am implementat posibilitatea de editare foto pentru un fișier sursă de tip BMP de 24bits.

Avantajul utilizării unui astfel de format pentru imagine este acela că tipul de fișier .bmp (BITMAP) este folosit pentru a stoca imagini la nivel digital, indiferent de sistemul de operare în care ajunge.

Astfel, ținând cont de funcționalitatea programului, și anume aceea de a prelucra imagini, acest format foto este cel mai avantajos, intrucat se poate preta la orice mediu de lucru.

2. Descrierea aplicației cerute

Scopul aplicației implementate este acela de a transforma formatul imaginii, oricare ar fi acela, într-unul Alb-Negru.

O astfel de transformare se face la nivelul fiecărui pixel al pozei, prin modificarea intensității celor trei nuanțe de culori: Roșu – Verde – Albastru.

Așadar prin aplicarea unei formule logaritmice asupra fiecărei culori, în urma modificării acestea vor actualiza pixelul din care au fost extras, conducând la modificarea setului de culori pentru imaginea originală.

3. Partea teoretică

Formula algoritmică aplicată asupra setului RGB este următoarea: **culoare_nouă = constantă * log(|culoare_veche| + 1)**, unde:

- culoare_nouă = noua valoare a culorii din RGB pentru pixelul studiat
- constanta = o valoare ce nu trebuie să depășească un anumit prag astfel încât la final culoare_nouă să fie cuprinsă între 0 (ce corespunde culorii negru) și 255 (ce corespunde culorii alb).
- culoare_veche = vechea valoare a culorii din RGB pentru pixelul studiat

Adunarea cu 1 (unu) în cadrul logaritmului se face ca în măsura în care culoare_veche = 0, atunci logaritmul ar avea ca rezultat infinit, ceea ce nu se află în intervalul necesar pentru valorile RGB.

4. Descrierea implementarii

Aplicația include clasa principală finalImage, situată în pachetul test, care incapsulează funcționalitățile cheie pentru procesarea imaginilor. Această clasă conține metodele esențiale readImage, writeImage și grayImage. finalImage extinde clasa abstractă imageGray din pachetul proiect, care impune un cadru pentru implementarea metodelor necesare prelucrării imaginilor, asigurând astfel conformitatea cu principiile programării orientate pe obiecte.

În cadrul programului, la nivelul executării cerinței, aceasta se face în felul următor:

- Încărcare imagine;
- Memorarea fiecărui pixel al acesteia într-un array de tip byte;
- Citirea pixel cu pixel și determinarea fiecărei valori RGB;
- Logaritmizarea fiecărei valori din RGB pentru fiecare pixel;
- Implementarea fiecărui pixel după actualizarea valorilor RGB;
- Salvarea imaginii rezultat într-un fișier selectat;
- Afisare rezultate timp de procesare fiecare etapa

5. Descrierea structurala – arhitecturala si functionala a aplicatiei

1. Clasa finalImage (package packTest):
    - Rol: Aceasta este clasa principală care extinde imageGray. Se ocupă de fluxul principal al procesării imaginii, de la citirea acesteia până la salvarea versiunii procesate.

- Funcționalități: - Citirea imaginii: Se citește imaginea dintr-o cale specificată și se extrage informația de header și body.
  - Transformare în grayscale: Aplică transformarea - Logarithmic Gray Level.
  -Salvare Imagine: Salvează imaginea modificată într-o locație specificată.

2. Clasa imageGray (package packWork):
   - Rol: Clasă abstractă care definește structura de bază pentru procesarea imaginii.
   - Funcționalități: Metode abstracte pentru setarea dimensiunilor și transformarea imaginii.
     Stocare cale imaginii.

3. Clasa imgProcessing (package packWork):
   - Rol: Aceasta clasa păstrează o imagine BufferedImage ca variabilă statică, utilizată de celelalte clase pentru procesare.
   - Funcționalități: Stocarea imaginii ce urmează a fi procesată.

4. Interfața interfaceGray (package packWork):
   - Rol: Definește o interfață pentru notificarea finalizării procesării.
   - Funcționalități: Metoda showGray pentru a anunța finalizarea procesului.

5. Clasa projectGray (package packWork):
   - Rol: Implementează interfața interfaceGray și afișează un mesaj la finalizarea programului.
   - Funcționalități: Implementează showGray pentru afișarea mesajului de final.

Mazilu Gabriela
334AA

Metodele implementate:

- Constructorii:

finalImage(): Constructor fără parametri.

finalImage(int height, int width, byte[] header, byte[] body): Constructor cu parametri pentru inițializarea înălțimii, lățimii, header-ului și body-ului imaginii.

finalImage(int height, int width, byte[] header, byte[] body, String path):

Constructor cu parametri suplimentari, inclusiv calea (path).

- Metode Setter:

setImg(int w, int h): Setează lățimea și înălțimea imaginii.

setWidth(int w): Setează lățimea imaginii.

setHeight(int h): Setează înălțimea imaginii.

- Metode Getter:

getWidth(): Returnează lățimea imaginii.

getHeight(): Returnează înălțimea imaginii.

Metoda readImage citește pixel cu pixel imaginea din path-ul sursă, introdus de la tastatură de către utilizator, iar din header-ul imaginii memorează width-ul (lățimea) și height-ul (înălțimea).

Metoda writeImage scrie pixel cu pixel imaginea în path-ul introdus de la tastatură de către utilizator.

Metoda readPixels citește pixelii imaginii pentru a putea fi folosiți mai departe de metoda grayImage. Mai exact, readPixels inițializează buffer-ul de lucru pentru valorile RGB ale fiecărui pixel.

Mazilu Gabriela
334AA

Metoda showGray este folosita pentru a afisa un mesaj la finalizarea procesarii imaginii.

Metoda grayImage aplica transformarea Logarithmic Gray Scale asupra fiecărui pixel al imaginii, modificând valorile RGB prin scalare algoritmică bazată pe formula dată.

Tratarea exceptiilor:

```
public static finalImage readImage() throws FileNotFoundException, IOException{
finalImage Img = new finalImage(height, width, header, body);
try{
    f = new File(path);
    img = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
    img = ImageIO.read(f);
    rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
    System.out.println("Path-ul catre imaginea sursa a functionat corespunzator.");
    }catch(IOException e){
    System.out.println("Error: "+e);
    }
double stopTime = System.nanoTime();
double elapsedTime = (stopTime - startTime) / 10000000000.0;
System.out.println("-> Timpul total de executie pentru incarcarea imaginii a fost de: " + elapsedTime + " secunde");
return Img;
```

```
public static void writeImage(finalImage modifiedImage) throws IOException{
    double startTime = System.nanoTime();
    Scanner keyboard = new Scanner(System.in);
    System.out.println("Path-ul catre destinatia unde va fi salvata poza modificata:");
    path=keyboard.next();
    keyboard.close();
    File f = new File(path);
    try{
          f = new File(path);
          ImageIO.write(img, "bmp", f);
          System.out.println("Proces incheiat cu succes.");
          }catch(IOException e){
          System.out.println("Error: "+e);
    }
    double stopTime = System.nanoTime();
    double elapsedTime = (stopTime - startTime) / 10000000000.0;
    System.out.println("-> Timpul total de executie pentru salvarea imaginii a fost de: " + elapsedTime + " secunde");
}
```

Varargs:

```
private static boolean varArgs(String ... arguments){
    long start = System.currentTimeMillis();
    if(arguments.length == 3) {
        String parameter1 = arguments[0].toLowerCase();
        if(!parameter1.equals("and")&&!parameter1.equals("or")&&!parameter1.equals("xor"))
            return false;
    }
    return false;
```

6. Conceptele de Programare Orientată Obiect utilizate

Moștenire: Clasa finalImage moștenește toate atributele și metodele din clasa părinte imageGray. Clasa nefiind constuiră ca privata conduce la posibilitatea de extragere a datelor din clasa copil, acestea fiind vizibile acolo.

Polimorfism: Clasa imageGray este construita ca fiind de tipul abstract, ea având metodele abstracte writeImage() și grayImage (). Aceste doua metode, implementate in clase copil, vor avea forme diferite in funcție de clasa în care sunt implementate.

Încapsulare: Toate atributele din clasa imageGray sunt de tipul protected(pentru a putea fi vizibile in clasele copil). In clasa copil myImage, toate atributele sunt private, pentru a nu putea fi accesibile din exterior. Pentru aceasta se definesc setteri si getteri pentru atributele necesare.

Abstractizare: Orice clasă care moștenește clasa imageGray va trebui să îi implementeze metodele abstracte.

Mazilu Gabriela
334AA

7. Evaluarea performanțelor

La nivelul performanțelor, timpul de execuție al programului este dependent doar de dimensiunea imaginii, formatul fiind obligatoriu unul de tip BMP de 24bits.

La nivelul output-ului foto, aceasta ia diferite nuanțe de Alb-Negru, depinzând de valoarea constantei din calculul logaritmic.

Cu cât constanta este mai mică cu atât rezultatul va fi și el mai mic, deci mai apropiat de 0, valoarea RGB pentru negru.

Cu cât constanta este mai mare cu atât rezultatul va fi și el mai mare, deci mai apropiat de 255, valoarea RGB pentru alb.

Așadar, nuanța de Alb-Negru pe care o va lua imaginea inițială depinde de valoarea constantei, care în cadrul programului implementat este standard, și nu poate fi citită de la tastatură întrucât și aceasta are un anumit interval de valori în care se poate afla.

8. Concluzii

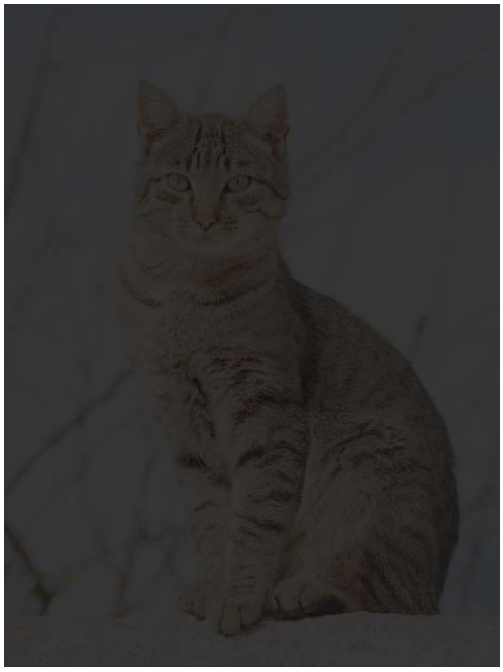
Așadar, algoritmul implementat are rolul de a aplica filtrul GrayLevel asupra imaginii de test. O astfel de transformare se face la nivelul fiecărui pixel al pozei, prin modificarea intensității celor trei nuanțe de culori: Roșu – Verde – Albastru.

Așadar prin aplicarea unei formule logaritmice asupra fiecărei culori, în urma modificării, acestea vor actualiza pixelul din care au fost extras, conducând la modificarea setului de culori pentru imaginea originală.

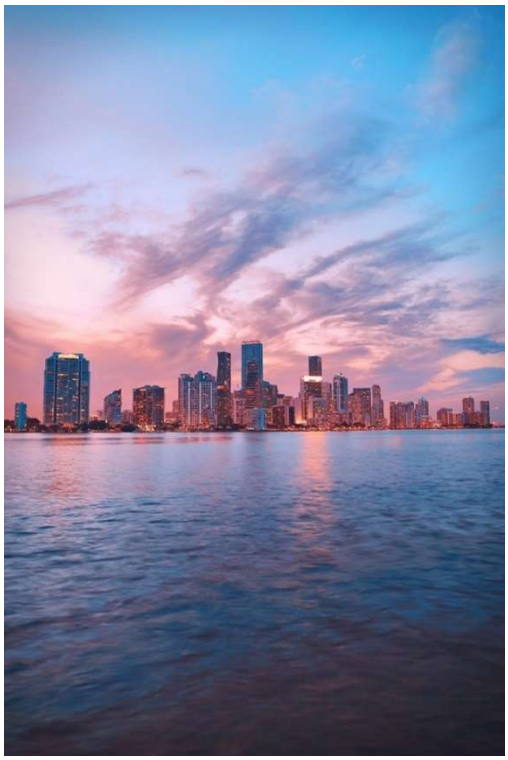
Mazilu Gabriela
334AA

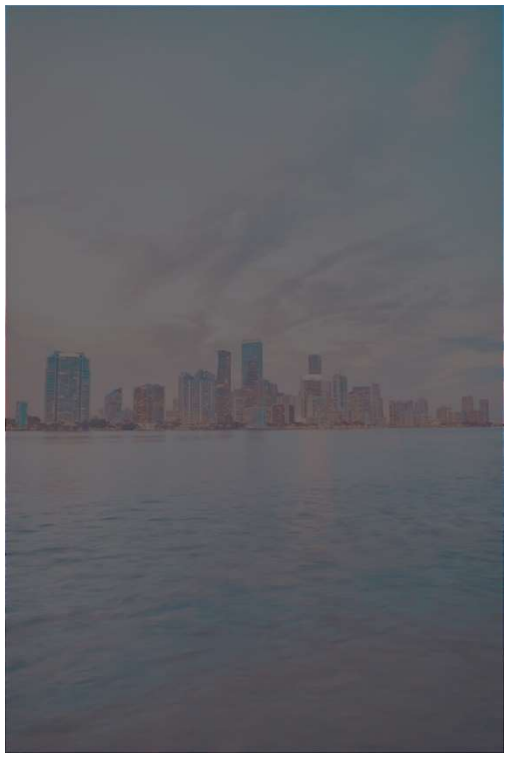
Imaginea originala:

Imaginea prelucrata:



Constanta = 10





Constanta = 20

9. Bibliografie

https://www.tutorialspoint.com/dip/gray_level_transformations.htm

https://www.dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-grayscale-image-in-java

https://homepages.inf.ed.ac.uk/rbf/HIPR2/pixlog.htm

https://ro.wikipedia.org/wiki/BMP_(format_fișier)

10. Documentatie cod sursa

Clasa finalImage:

```
package packTest;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.awt.Color;
import java.io.FileNotFoundException;
import java.awt.image.BufferedImage;
import java.io.FileOutputStream;
import javax.imageio.ImageIO;

import packWork.imageGray;
import packWork.projectGray;

import java.util.Scanner;
public class finalImage extends imageGray {
    private static int width;
```

```
  private static int height;
  byte[] header;
 byte[] body;
  static int rgb_buffer[][][];

  public finalImage(){}

  public finalImage(int height, int width, byte[] header, byte[] body) {
          this.height = height;
    this.width = width;
    this.header = header;
    this.body = body;
  }

  public finalImage(int height, int width, byte[] header, byte[] body, String path) {
          this.height = height;
    this.width = width;
    this.header = header;
    this.body = body;
    this.path = path;
  }

  //Setters
  public void setImg(int w, int h){
          this.width=w;
          this.height=h;
  }
  public void setWidth(int w){
          this.width=w;
  }
  public void setHeight(int h){
          this.height=h;
  }

  //Getters
  public int getWidth(){
          return this.width;
  }
  public int getHeight(){
          return this.height;
  }
```

```
    public static void main(String args[])throws IOException{
            double startTime = System.nanoTime();
            finalImage image = readImage(); //incarcare imagine
            image.grayImage(); //Modificarea setului de culori Rosu-Verde-Albastru prin
scalare algoritmica asupra imaginii
            writeImage(image); //Imagine finala
            double stopTime = System.nanoTime();
            double elapsedTime = (stopTime - startTime) / 10000000000.0;
      System.out.println("-> Timpul total de executie al programului a fost de: " +
elapsedTime + " secunde");
      projectGray p = new projectGray();
      p.showGray();
    }


    //Incarcarea imaginii, pixel cu pixel, in cadrul unui array de tip byte[]
    public static finalImage readImage() throws FileNotFoundException, IOException{
            double startTime = System.nanoTime();

    System.out.println("==================================================
==========");
            System.out.println("          Logarithmic Gray Level Transform");
            System.out.println("
==================================================");
            System.out.println("    Proiect realizat de: Mazilu Gabriela 334AA");

    System.out.println("==================================================
==========");
            Scanner keyboard = new Scanner(System.in);
            System.out.println("Path-ul catre imaginea asupra careia se aplica modificarile:");
            path=keyboard.next();
            File f = new File(path);
            while ( !f.exists() ){
        System.out.println("Path-ul introdus nu este corect!");
        path=keyboard.next();
        f = new File(path);
      }
            byte[] b = Files.readAllBytes(Paths.get(path));

      byte[] header = new byte[54];
      for ( int i = 0 ; i < 54 ; i ++ )
        header[i] = b[i];
```

```
        width =
    ((header[21]&0xFF)<<24)|((header[20]&0xFF)<<16)|((header[19]&0xFF)<<8)|((header[18]
    &0xFF));//row

        height=
    ((header[25]&0xFF)<<24)|((header[24]&0xFF)<<16)|((header[23]&0xFF)<<8)|((header[22]
    &0xFF));//col

        // Memorarea pixelilor in array-ul de tip byte[], body
        byte[] body = new byte[3*width*height];
        for ( int i = 54 ; i < 3*width*height ; i++ )
           body[i-54] = b[i];

        //Incarcare imagine
        finalImage Img = new finalImage(height, width, header, body);
                try{
                        f = new File(path);
                        img = new BufferedImage(width, height,
    BufferedImage.TYPE_INT_ARGB);
                        img = ImageIO.read(f);
                        rgb_buffer = new int[3][img.getHeight()][img.getWidth()];
                        System.out.println("Path-ul catre imaginea sursa a functionat
    corespunzator.");
                        }catch(IOException e){
                        System.out.println("Error: "+e);
                        }
                double stopTime = System.nanoTime();
                double elapsedTime = (stopTime - startTime) / 10000000000.0;
        System.out.println("-> Timpul total de executie pentru incarcarea imaginii a fost de: " +
    elapsedTime + " secunde");
        return Img;
       }

       //Citire pixel cu pixel din imagine si determinarea pentru fiecare dintre acestia a valorilor
    RGB
       public void readPixels(){
                double startTime = System.nanoTime();
                for(int row = 0; row < img.getHeight(); row++){
                        for(int col = 0; col < img.getWidth(); col++){
                                Color c = new Color(img.getRGB(col,row));
                                rgb_buffer[0][row][col] = c.getRed();
                                rgb_buffer[1][row][col] = c.getGreen();
                                rgb_buffer[2][row][col] = c.getBlue();
```

```
            }
                }
                double stopTime = System.nanoTime();
                double elapsedTime = (stopTime - startTime) / 10000000000.0;
        System.out.println("-> Timpul total de executie pentru citirea pixelilor a fost de: " +
    elapsedTime + " secunde");
    }

    //Modificarea setului de culori Rosu-Verde-Albastru prin scalare algoritmica
    //Scalarea se face conform formulei: NouaCuloare = constanta * log(vechea culoare + 1)
    public void grayImage(){
            double startTime = System.nanoTime();
            readPixels();
            for(int row = 1; row < img.getHeight()-1; row++){
                    for(int col = 1; col < img.getWidth()-1; col++){

                            float a = (float) 20;


                            int newr = (int) (a * Math.log(rgb_buffer[0][row][col] + 1));
                    int newg = (int) (a * Math.log(rgb_buffer[1][row][col] + 1));
                    int newb = (int) (a * Math.log(rgb_buffer[2][row][col] + 1));

                            Color c=new Color(newr,newg,newb);
                            img.setRGB(col,row,c.getRGB());
                        }
            }
            double stopTime = System.nanoTime();
            double elapsedTime = (stopTime - startTime) / 10000000000.0;
        System.out.println("-> Timpul de executie pentru aplicarea Logarithmic Gray Scale
    Transform a fost de: " + elapsedTime + " secunde");
    }

    //Salvare rezultat
    public static void writeImage(finalImage modifiedImage) throws IOException{
            double startTime = System.nanoTime();
            Scanner keyboard = new Scanner(System.in);
            System.out.println("Path-ul catre destinatia unde va fi salvata poza modificata:");
            path=keyboard.next();
            keyboard.close();
            File f = new File(path);
            try{
                f = new File(path);
```

```
                ImageIO.write(img, "bmp", f);
                System.out.println("Proces incheiat cu succes.");
                }catch(IOException e){
                System.out.println("Error: "+e);
            }
            double stopTime = System.nanoTime();
            double elapsedTime = (stopTime - startTime) / 10000000000.0;
        System.out.println("-> Timpul total de executie pentru salvarea imaginii a fost de: " +
elapsedTime + " secunde");
    }
}
```

Clasa imageGray:

```
package packWork;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.awt.image.BufferedImage;
public abstract class imageGray extends imgProcessing
{
   protected static String path; //Path-ul catre zona de unde trebuie preluata imaginea,
respectiv publicat rezultatul
    protected abstract void grayImage(); //Modificarea setului de culori Rosu-Verde-Albastru
prin scalare algoritmica
    protected static void writeImage() throws IOException {}
    protected abstract void setWidth(int w); //determinarea latimii
    protected abstract void setHeight(int h); //determinatea inaltimii
    protected void setImg(int w, int h){}

    //Creare varArgs
    private static boolean varArgs(String ... arguments){
      long start = System.currentTimeMillis();
      if(arguments.length == 3) {
         String parameter1 = arguments[0].toLowerCase();

if(!parameter1.equals("and")&&!parameter1.equals("or")&&!parameter1.equals("xor"))
            return false;
      }
      return false;
   }
}
```

Mazilu Gabriela
334AA

Clasa imgProcessing:

```
package packWork;

import java.awt.image.BufferedImage;
public class imgProcessing {
	public static BufferedImage img;
}
```

Clasa interfaceGrey:

```
package packWork;

//Interface pentru apelare showGray = clasa ce anunta finalizarea programului

public interface interfaceGray {

	public void showGray();

}
```

Clasa projectGray:

```
package packWork;

//Clasa pentru afisarea momentului de finalizare rulare a programului.

public class projectGray implements interfaceGray{

	public void showGray()
	{
		System.out.println("Program incheiat.");

	}

}
```