Converting Color Image to Gray-Scale Image Average method

Procop Iuliana

Cuprins

Introducere

Scopul aplicatiei este de a converti o imagine color in format BMP intr-o imagine in nuante de gri. Utilizatorul aplicatiei poate specifica atat locatia si numele imaginii care doreste a fi convertita, cat si locatia si numele imaginii in tonuri de gri rezultate in urma conversiei. Acesta primeste mesaje de avertizare daca adresele sursa si destinatie nu sunt valide si informatii despre performantele aplicatiei.

Generalitati imagini

O imagine este o matrice de valori numerice, unde fiecare element din matrice este un pixel.

După tipul valorilor numerice ale matricei, imaginile pot fi:

- **Imagini scalare**, unde fiecare componentă a matricei este un scalar, acestea fiind imaginile monocrome, în care punctele au doar două valori posibile, în general alb-negru, pixelii acestora având valori unsigned 8-bit (0 – negru, 255 - alb).

- **Imagini vectoriale**, unde fiecare componentă este un vector de numere, în special imaginile color, vectorul având trei elemente RGB, ce reprezintă cele trei componente de bază ale oricărei culori.

Descrierea conceptului aplicatiei

Pentru realizarea conversiei unei imagini color în alb – negru prin intermediul Metodei Average, am transpus imaginea sub forma unei matrice, din care am extras fiecare pixel.

Astfel, pentru fiecare pixel am extras cele 3 valori: roșu (R), verde (G) și albastru (B). Tonul de gri al noului pixel va fi format prin compunerea valorilor rezultate prin împărțirea fiecărei valori R G B la media aritmetică a valorilor R G B.

Descrierea aplicatiei

1. **Clasa abstracta Image** este clasa ce implementeaza **Interfata**, astfel nu poate fi instantiata;
2. **Clasa FormatPixel** este o clasa ce mosteneste clasa **Image** si are ca atribut *format* ce specifica daca imaginea este color sau in tonuri de gri. Valoarea acestuia e stabilita prin metoda *verificare;*
3. **Clasa Greyscale** este clasa care mosteneste clasa **FormatPixel**. In metoda *greey* este verificata valoarea atributului format si in functie de aceasta, este convertita imaginea in greyscale;
4. **Clasele ReadF si WriteF** sunt clasele pentru citirea si scrierea path-ului in care se gaseste imaginea color sau deja convertita;
5. **Clasa Main** este o clasa in care se gaseste metoda *main* care instantiaza obiectele de tip greey si BufferedImage;

Conceptele POO – incapsulare, mostenire, polimorfism, abstractizare

Incapsulare

ReadF.java | WriteF.java | Interfata.java | Image.java | Greyscale.java | Main.java | FormatPixel.java

```java
package pack02;

import java.awt.image.BufferedImage;

abstract public class Image implements Interfata{
    private BufferedImage img;
    private String path;

    // setter pentru path-ul fisierului
    public void setPath(String path) {
        this.path = path;
    }

    // getter pentru path-ul fisierului
    public String getPath() {
        return this.path;
    }

    // setter pentru imagine
    public void setImage(BufferedImage image) {
        this.img = image;
    }

    // getter pentru imagine
    public BufferedImage getImage() {
        return img;
    }

    public void greey() {
    }
}
```

Mostenire

```java
public class FormatPixel extends Image {
```

```java
public class Greyscale extends FormatPixel{
```

Polimorfism

Conceptul de polimorfism este folosit in proiect astfel:

In clasa **Greyscale** sunt create cele doua functii cu acelasi nume, dar cu parametri diferiti, una dintre ele realizand conversia unei imagini color in Grayscale prin metoda Average, iar cealalta are ca scop returnarea numarului de pixeli ai imaginii.

Prima functie „greey” este cea care realizeaza conversia imaginii si nu are parametri:

```
// metoda ce converteste o imagine color in Greyscale
public void greey() {

    // daca imaginea este color, o converteste in Greyscale
    if(this.getFormat() == 0)
    {
        //se obtine imaginea
        BufferedImage image = this.getImage();

        // se citesc latimea si inaltimea
        int width = image.getWidth();
        int height = image.getHeight();

        for(int i = 0; i < height; i++)
        {
          for(int j = 0; j < width; j++)
          {

            int p = image.getRGB(j, i); // se citeste valoarea rgb a pixeului

            int r = (p>>16)&0xff; // variabila ce stocheaza valoarea culorii rosii a pixelulu
            int g = (p>>8)&0xff; // variabila ce stocheaza valoarea culorii verzi a pixelului
            int b = p&0xff; // variabila ce stocheaza valoarea culorii albastre a pixelului

            // se calculeaza media celor 3 valori
            int avg = (r + g + b) / 3;

            // se inlocuiesc valorile rgb ale pixelului cu media calculata anterior
            p = (avg<<16) | (avg<<8) | avg;
            image.setRGB(j, i, p);

          }

    }

    // daca imaginea este deja Greyscale, asta duce la afisarea unui mesaj de avertisment si
    else {
        System.out.println("Imaginea este deja Greyscale!");
        System.exit(0);
    }
}
```

Cea de-a doua functie „greey” care indica numarul de pixeli ai imaginii este:

```
public int greey(BufferedImage img) {

    int width = img.getWidth();    // se obtine latimea
    int height = img.getHeight(); // se obtine inaltimea
    return width * height; //returneaza numarul de pixeli ai imaginii

}
```

Apelul celor doua metode in main produce o data aplicarea conversiei Grayscale cu metoda Average asupra unei imagini, iar urmatoarea consta in scrierea in consola a numarului de pixeli ai imaginii.

```
// daca se trece de verificare, incepe procesarea imaginii
start = System.nanoTime(); // se memoreaza momentul de inainte de procesare
grey.greey();
System.out.println("Timp procesare: " + (System.nanoTime() - start) + " nanosecunde"); //


int nrpixeli = grey.greey(img);
System.out.println("Numarul de pixeli ai imaginii este: " + nrpixeli);
```

Abstractizarea

Abstractizarea consta in nepermiterea anumitor caracteristici sa fie vizibile in exterior. Astfel, se creeaza un fel de interfata comuna pe care se bazeaza o anumita categorie de obiecte.

```
abstract public class Image implements Interfata{
```

Lucru cu fisiere si tratarea erorilor

Pentru citire:

```java
package pack02;

import java.awt.image.BufferedImage;

// clasa pentru citirea path-ului in care se afla imaginea
public class ReadF {
    public static BufferedImage read(String path) {
        BufferedImage img = null;
        try{
            File f = new File(path);
            img = ImageIO.read(f);
          }catch(IOException e){
            System.out.println("Citire imagine nereusita!"); // tratare eroare de citire
            System.exit(0);
          }
        return img;
    }
}
```

Pentru scriere:

```java
package pack02;

import java.awt.image.BufferedImage;

// clasa pentru scrierea path-ului in care se afla imaginea
public class WriteF {
    public static void write(BufferedImage img, String path) {
        try{
            if(!path.endsWith(".bmp")) {
                System.out.println("Fisierul trebuie sa contina extensia .bmp!"); // tratare eroa
                System.exit(0);
            }
            File f = new File(path);
            ImageIO.write(img, "bmp", f);

        }catch(IOException e){
            System.out.println("Scriere nereusita!"); // tratare eroare de scriere
            System.exit(0);
        }
    }
}
```

Mod de functionare

Aplicatia care realizeaza convertirea unei imagini color in Grayscale folosind metoda mediei aritmetice incepe prin scrierea de catre utilizator in linia de comanda a adresei imaginii folosite pentru convertirea acesteia.

Astfel, se apeleaza metoda de citire din clasa **ReadF** in care este tratata exceptia pentru care adresa imaginii este invalida prin afisarea unui mesaj in consola: „Citire imagine nereusita!”.

In urma introducerii path-ului se verifica formatul pixelilor imaginii („**grey.verify()**”). Daca imaginea este deja in tonuri de gri, un mesaj este afisat in consola si executia programului se incheie: „Imaginea este deja Grayscale!”. Daca imaginea este color, ea va fi pregatita pentru a fi convertita in Greyscale prin metoda **Average**. Astfel, daca se trece de etapa de verificare, incepe procesarea imaginii in Grayscale cu metoda specificata.

Metoda **Average** este descrisa de relatia tonuri de gri = (R + G + B) / 3. Aceasta consta in obtinerea tonurilor de gri prin calculul mediei aritmetice intre valorile pentru pixelii de culoare rosie (R), pixelii de culoare verde (G) si cei de culoare albastra (B).

Urmatorul pas al aplicatiei consta in citirea destinatiei in care va fi scrisa imaginea convertita. Se apleaza metoda de scriere din clasa **WriteF** in care este tratata, de asemenea, cazul de eroare in care adresa poate fi invalida si este afisat un mesaj in consola: „Scriere nereusita!”. In aceasta metoda se mai trateaza si cazul in care fisierul nu are extensia „.bmp”, se va afisa un mesaj de eroare: „Fisierul trebuie sa contina extensia .bmp!”.

Pentru fiecare etapa este afisat in consola tipul de executie al acesteia, exprimat in nanosecunde.

Avand in vedere faptul ca fiecare componenta a codului este comentata, se pot gasi informatii suplimentare in privinta anumitor detalii neprecizate, codul sursa fiind gasit in arhiva proiectului.

Rezultatul obtinut in urma conversiei