Converting Color Image to Gray-Scale Image – Average method

Neagu Fabian-Florin

331AB

Cuprins:

1) Introducere

Proiectul realizat are ca scop implementarea unui algoritm prin intermediul caruia se poate reliza conversia unei imagini de tip color in imagine de tip Gray-Scale prin intermediul metodei mediei folosind limbajul de programare Java si mediul de dezvoltare Eclipse.

2) Exemplu functionalitate:

Imaginea aplicata:

Imaginea obtinuta:

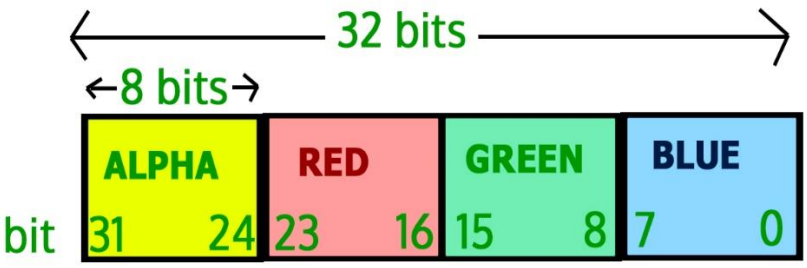
Performantele obtinute:

```
Introduceti Path-ul de input:
D:\AWJ - Aplicatii Web Suport Java\Final_form\input1.bmp
Introduceti Path-ul de output:
D:\AWJ - Aplicatii Web Suport Java\Final_form\output.bmp
Introduceti nivelul de alb-negru dorit( 1 - 10 ):
1
Executie inceputa !
Producer Thread: 1/4 din imagine.
Consumer Thread: 1/4 din imagine.
Producer Thread: 2/4 din imagine.
Consumer Thread: 2/4 din imagine.
Producer Thread: 3/4 din imagine.
Consumer Thread: 3/4 din imagine.
Producer Thread: 4/4 din imagine.
Consumer Thread: 4/4 din imagine.

Durata citire imagine din fisier: 4733 milisecunde
Durata prelucrare imagine Convert to Grey-Scale: 82 milisecunde
Durata scriere imagine in fisier: 213 milisecunde

Vizualizati fisierul output !
```

3) Descriere algortim Average-Method

Acest alogoritm are ca scop prelucrarea imaginii la nivel de pixel in scopul de a transforma o imagine initiala intr-o imagine alb-negru. Imaginea este divizata la nivel de pixeli, iar fiecare dintre acestia se vor prelucra astfel incat sa se ajunga la rezultatul dorit.

Imaginea este structurata ca o matrce de pixeli, iar structura fiecarui pixel este urmatoarea:



Pasii necesari implementarii acestui algoritm sunt:

A) Determinarea dimensiunilor imaginii in vederea parcurgerii matricei de pixeli ce compun imaginea(lungime si inaltimea imaginii)

```
int width = this.initialImage.getWidth(); //Latimea imaginii
int height = this.initialImage.getHeight(); //Inaltimea
imaginii
```

B) Extragerea fiecarui pixelul si aflarea valorilor A – Alpha, R – Red, G – Green, B – Blue ce compun pixelul respectiv in vederea prelucrarii ulterioare a acestor valori

```
int a = (pixel>>24)&0xff;//Valoara Alpha a pixelului
int r = (pixel>>16)&0xff;//Valoara Red a pixelului
int g = (pixel>>8)&0xff;//Valoara Green a pixelului
int b = pixel&0xff;//Valoara Blue a pixelului
```

C) Aflarea mediei valorilor R, G, B in vederea prelucrarii valorilor ce compun pixelul respectiv pentru obtinerea imaginii de tip alb-negru

```
int avg = (r+g+b)\(level+2);//Aplicare medie
```

Acest algoritm a fost modificat astfel incat sa permita modificare imaginii pe nivele de alb-negru(1- nivelul standard, cel mai putin intunecat, 9 – nivel maxim, cel mai intunecat).

D) Crearea noului pixel(Specifice imaginii alb-negru) si pozitionarea valorilor pe pozitiile specifice acestora

```
int newPixel = (a<<24) | (avg<<16) | (avg<<8) | avg;//Gasim valoarea noului pixel pentru a obtine imaginea dorita
```

E) Compunerea tuturor noilor pixeli pentru crearea noii imagini(imagine in Grey-Scale) , rezultatul fiind imaginea complet prelucrata la scara alb-negru.

```
for(int x=0;x<height;x++)//Parcurgere linii
          for(int y=0;y<width;y++)//Pracurgere coloane
          {
               newImg.setRGB(y,x,list.get(x).get(y));//Copiere valorile din lista in Valorile pixelilor imaginii noi
          }
```

Valorile pixelilor prelucrati au fost stocate intr-o matrice realizata cu ajutorul listelor.

4)Decriere generala program

Programul contine 3 parti principale:

- Citire imagine sursa din fisier
- Prelucrare imagine citita anterior
- Scriere imaginii prelucrate in fisier.

Fisierul cu care se va lucra va avea un format de tip BMP – 24bit BMP – RGB. Imaginea data va fi preluata din folder utilizand doua threaduri si bazandu-se pe scenariul Producator-Consumator. Mai exact, threadul Producator va prelua din fisier cate un sfert din imagine, sferturi ce vor fi preluate pe rand de catre cel de al doilea thread(Consumator) in vederea obtinerii imaginii complete.

Citire imagine sursa din fisier

Citirea din fisier este o etapa esentiala in realizarea aplicatiei propuse aceasta realizandu-se prin intermediul a doua Threaduri care comunica intre ele folosind scenariul Producator – Consumator. Producatorul genereaza flux de date (Citeste din fisier cate un sfert din imagine), iar consumatorul preia acest flux de date.

Pentru a fi accesibila ambelor fire de executie, vom incapsula variabila ce va contine bitii cititi din fisier de catre Producator intr-un obiect descris de clasa ImageMatrix si care va avea doua metode: put (pentru punerea pe rand a cate un vector de bytes cititi din fisier in buffer) si get (pentru obtinerea vectorilor respectivi din buffer).

Pentru a nu se accesa simultan resursa comuna, si anume, ImageMatrix, in momentul in care Producer pune valori in Buffer(ImageMatrix), aceasta resursa se blocheaza pentru a nu permite si consumatorului sa o acceseze. Cand Producatorul a terminat de pus valorile respective in buffer, acesta „se deblocheaza" permitand Consumatorului sa preia datele din acesta si blocand resursa pana la terminarea sarcinii sale. In plus, cele doua fire de executie sunt sincronizate, comunicarea dintre ele realizandu-se prin intermediul metodelor wait() si notify().

Prelucrare imagine

Odata citita din fisier, imaginea respectiva este supusa algoritmului de prelucrare descris, algoritm ce are ca scop transformarea imaginii initiale intr-o imagine alb-negru, cu nivelul respectiv.

Scriere imagine in fisier

Dupa prelucrarea imaginii, imaginea trebuie sa fie stocata intr-un fisier dat cu ajutorul unei functii ce lucreaza cu fisiere.

5)Decriere Clase si Metode folosite

1)Clasa AbstractClassProdCons

Aceasta clasa este de tip abstracta si extinde clasa Threada, ea fiind folosita pentru implentarea ulterioara a thread-urilor Consumer si Producer pentru crearea scenariului dorit in scopul citirii imaginii ce urmeaza a fi prelucrata din fisier.

Aceast clasa contine un obiect de tip ImageMatrix, obiect ce joaca rolul de buffer dintre Producator si consumator. Aceasta repreinta resursa comuna dintre cele doua thread-uri, buffer in care producatorul va pune valori, iar consumatorul va lua valori. In plus, se percepe exitenta unui obiect de tip Dimension care incapsuleaza dimensiunile imaginii(lungimea si inaltimea acesteia), date necesare atat pentru threadul consumer cat si pentru threadul producer.

De asemenea, aceast clasa este prevazuta cu un constructor cu parametri, metoda public void run() urmand a fi implementata in clasele Producer si Consumer, acestea extinzand clasa AbstractClassProdCons.

2)Clasa ImageMatrix

Aceasta clasa reprezinta resursa comuna impartita de Producer si Consumer. Printre variabilele acestei clase se numara un vector de bytes vect in care se vor pune succesiv bitii imaginii de catre producer. De asememea din acest vector se vor prelua bitii respectivi de catre consumer, tot procesul realizandu-se sincronizat.

```
private int dimensiune;// Dimeniunea imaginii necesara stabilirii dimensiunii bufferului
private final byte[] vect;  //Vectorul de bytes ce va stoca imaginea(imaginea va fi stocata sub forma vect de bytes)
private static boolean available;  //Daca este available sau nu resursa noastra
```

Variabila dimensiune reprezinta dimensiunea totala(lungime inamgine * inaltime imagine), aceasta fiind necasara deoarecel vectorul de biti va fi alocat in constructor chiar cu aceasta dimensiune, el fiind capabil sa stocheze toti bitii ce compun imaginea respectiva. In plus, variabila available este de tip boolean, aceast avand rolul de a realiza sincronizare dintre threaduri, indicand momontul in care Producerul are acces asupra resursei si cand Consumerul are acces la aceasta. In momentul in care Producerul pune valori in buffer, prin intermediul mecanismului de sincronizare, Consumerul trebuie sa astepta pana cand producerul elibereaza resursa respectiva si invers.

Functiile put si get vor fi folosite de clasele Producer si Consumer in interiorul metodelor run() ajutand la punerea/preluarea datelor in/din resursa comuna, si anume, ImageMatrix. Acetea sunt sincronizate, astfel incat sa se evite eventualele erori datorate problemelor de nesincronizare a thread-urilor.

Functia put are rolul de a alipi fragmentele de imagine la buffer. Mai exactm functia put primeste ca parametru un vector de biti ce corespund unui sfer de imagine, dar si un index care indica al al catelea sfert de imagine este acesta. Folosind aceste date functia alipeste acest nou fragment de imagine in buffer avand grija sa nu afecteze partea deja existenta in acesta. Dupa terminarea acestui proces, variabila available se face true, iar, prin intermediul metodei notifyAll(), se anunta threadu Consumer ca Producerul a terminat operatia asupra bufferului, Consumerul preluand conducerea acum.

```
public synchronized void put(byte [] vect1, int pozitie) // Functie put apelata de Producer
{
    while (available) {  // Daca Consumatorul inca ia valoarea
        try {
            wait();
            // Asteapta consumatorul pana termina de preluat valoarea
        } catch (InterruptedException e) {
            System.out.println("Eroare put-wait()");//Mesaj aparitie exceptie
            e.printStackTrace(); // Afisare detalii exceptie
        }
    }
    // Thread-ul Producer pune cate un sfert din imagine in Vectorul vect pentru a putea fi prelulata de Consumer
    for(int i = pozitie * dimensiune / 4; i < (pozitie + 1)*dimensiune/4; i++)
    {
        vect[i] = vect1[i - pozitie * dimensiune / 4]; // Se completeaza noile elemente(Nu se suprascriu cele vechi)
    }//Completez de la pozitie*dimensiune/4

    available = true;//Am realizat put-ul
    notifyAll(); // Se anunta terminarea functiei
}
```

Functia get are rolul de a extrage din buffer, mai exact din vectorul de bytes, valorile corespunzatoare sfertului din imagine respectiv. Cu ajutorul acestei functii, thread-ul consumator primeste sfertul de imagine in mod indirect de la Producer(pentru ca mai intai trece prin buffer si apoi ajunge la Consumer).Dupa terminare acestei operatiuni vaiabila available devine false, indicand faptul ca resursa comuna, bufferul, este din nou disponibil, iar cu ajutorul metodei notifyAll() Threadul Producer este instiinta ca poate incepe din nou operatiunea de citire a urmatorului sfert din imagine.

```
public synchronized byte[] get() // Functie get apelata de Consumer(ia valori din buffer)
{
    if (!available) { //Daca producatorul inca pune valoare in buffer
        try {
            wait(); // Asteapta producatorul sa termine de pus valorile
        } catch (InterruptedException e) {
            System.out.println("Eroare get-wait()"); //Mesaj aparitie exceptie
            e.printStackTrace(); // Afisare detalii exceptie
        }
    }
    available = false;  //Am realizat si terminat get-ul

    notifyAll(); // Se anunta terminarea functiei
    return vect; //Returnare buffer
}
```

3)Clasa Producer

Aceasta clasa extinde clasa abstracta AbstractClassProdCons, deci mosteneste deja bufferul de tip obiect ImageMatrix, precum si un obiect de tip Dimension ce incapsuleaza dimensiunile imaginii. De aseemenea, clasa mai contine si o varibila de tip File ce reprezinta fisierul din care Producerul va trebui sa citeasca cate un sfert de imagine, dar si o variabila de tip inputStream ce reprezinta un StreamInput de bytes folosit pentru citirea din fisier cu ajutorul metodei run.

```
public void run()
{

    int dim = getDimension();   //Dimensiunea fisierului
    try {
        is = new FileInputStream(inputFile);  //InputStream necesar citirii din fisier
    } catch (FileNotFoundException e1)
    {
        System.out.println("FileNotFoundException in Producer Thread");//Afisare mesaj eroare aparitie exceptie
        e1.printStackTrace();//Semnalare eroare de tip FileNotFound
    }

    for (int i = 0; i < 4; i++) // Structura repetitiva for cu 4 cicli specifici citirii fiecarui sfert din imagine
    {
        byte[] vect1 = new byte[dim/4+1]; //Vector in care se va extrage pe rand fiecare sfert din bitii fisierului
        try
        {
            is.read(vect1, 0, dim/4);//Se citeste in vectorul vect1 cate un sfert din imagine
            // Se citeste sferul de imagine corespunzator pasului la care suntem
        } catch (IOException e) {
            System.out.println("Eroare read-Producer");//Afisare mesaj eroare aparitie exceptie
            e.printStackTrace();  //Semnalare eroare de tip IO
        }
        this.buffer.put(vect1, i);// Producer trimite rezultatul catre consumer (Prin BUFFER)
        System.out.println("Producer Thread: " + (i+1) + "/4 din imagine.");
        try {
            Thread.sleep(1000); //Threadul Producer asteapta o secunda
        } catch (InterruptedException e) {
            System.out.println("Eroare put-Producer");//Afisare mesaj eroare aparitie exceptie
            e.printStackTrace();//Semnalare eroare de tip Intrerupere
        }
    }
}
```

In metoda run se realizeaza mai intai initializarea inputStream-ului necesar citirii din fisierul respectiv, iar apoi se realizeaza o structura repetitiva for cu 4 ciclii ce corespun citirii succesive a fiecarul sfert din bitii ce contin imaginea respectiva. Prin intermediul functiei is.read(vect1,0,dim/4) se citeste

cate un sfert din fisier, iar mai apoi, prin intermediul functiei put, detaliata la punctul anterior, se pun aceste valori citite in buffer pe pozitii corespunzatoare. In final, acesta intra in sleep, asteapta o secunda si se realizeaza sincronizare dintre cele doua threaduri astfel incat sa nu apara fenomenul de consistenta.

4) Clasa Consumer

Precum clasa detaliata anterior, si aceast clasa extinde abstracta AbstractClassProdCons, deci mosteneste deja bufferul de tip obiect ImageMatrix, precum si un obiect de tip Dimension ce incapsuleaza dimensiunile imaginii. De asemenea, se percepe existenta unui evctor de bytes ce are rolul de a stoca valorile(Sferturile de imagine) preluate din buffer, valori primite in mod indirect de la Producer.

```
public void run() //Consumer preia cate 1/4 din imagine de la Producer pe masura ce acesta scrie in Buffer
{
    int dim=this.getDimension(); //Dimensiunea totala a imaginii
    for (int i = 0; i < 4; i++) // Se primesc pe rand sferturile produse de Producer
    {
        byte[] aux=this.buffer.get();  //Vector auxiliar al informatiilor primite din buffer
        //Sferturile de imagine provenite din buffer(De la Prodeucer) trebuie puse in ordine si pe poitii corecte
        System.arraycopy(aux, i*dim/4, vect,i*dim/4, dim/4); //Consumer primeste sfertul de imagine de la Producer(din buffer)
        //vect = this.buffer.get();// Consumer primeste sfertul de imagine de la Producer(din buffer)
        System.out.println("Consumer Thread: " + (i+1) + "/4 din imagine."); //Se afiseaza statusul consumatorului corespunzator
            try {
                Thread.sleep(1000);  //Thread-ul Consumer asteapta o secunda
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                System.out.println("Eroare intrerupere Consumer sleep");//Afisare mesaj eroare aparitie exceptie
                e.printStackTrace(); //Afisare detalii exceptie
            }
    }
}
```

In interiorul metodei run se realizeaza aceiasi structura repetitiva cu 4 ciclii ce corespun primirii fiecarui sfert de imagine provenite de la Producer. Prin intermediul functiei get() se asigura preluarea vectorului de biti ce corespund sfertului de imagine respectiv si alipirea acestora la vectorul de bytes propriu clasei Consumer.

5) Interfata InitialImageInterface

Aceasta reprezinta interfata ce va fi implementata de catre clasa InitialImage, iar metodele specificate de acesta interfata sunt reprezentate de o metoda ce are rolul de a citi din fisier(readImageFromFile), o metoda care returneaza dimensiunile imaginii si una care returneaza imaginea in sine(obiect de tip BufferedImage).

6)Clasa InitialImage

Aceasta clasa implementeaza interfata detaliata la punctul precedent, ea reprezentand clasa imaginii prorpiu-zise. Aceasta contine o singura variabila de clasa, si anume, un obiect de tip BufferedImage ce reprezinta imaginea ce va fi citita din fisierul respectiv.

Deoarece imaginea nu se primeste ca paramteru, ci ea necesita a fi citita intodeauna dintr-un fisier, constructorul acestei clase primeste un singur parametru, si anume, fisierul din care se va realiza citirea imaginii respective. Constructorul acesta apeleaza mai departe o functie specializata ce are rolul de a citi propriu zis imaginea din fisierul transmis ca parametru, iar, la finalul executiei acestei functii, imaginea va fi stocata in variabila clasei de tip BufferedImage.

Mai departe, functia readImageFromFile() se ocupa de citirea propriu zisa a imaginii din fisier(folosind bineinteles scenariul Producer-Consumer enuntat mai sus). Aceasta functie initializeaza Thread-urile Producator si Consumator si le porneste, avand grija sa se realizeze mecanisme de sincronizare in mod corect. Se utilizeaza declaratiile synchronized ale functiilor put si get prezentate anterior, dar si functia join() ce are rolul de a bloca main-ul pana la executia thread-urilor. Dupa citirea succesiva a fiecarui sfert de imagine, in buffer se gaseste un vector de bytes care compun imaginea completa, iar, pentru a transforma acest vector de bytes(Citit din fisier) intr-un obiect de tip BufferedImage se foloseste urmatoarea secventa:

```
Producer producerThread = new Producer(buff, inputFile, d); // Thread 1 - Producer(Citeste din fisier si pune in buffer)
Consumer consumerThread = new Consumer(buff, d); // Thread 2 - Consumer(Primeste cea ce Producer citeste(la din buffer))
producerThread.start(); // Pornire thread Producer
consumerThread.start(); // Pornire thread Consumer

try {
    consumerThread.join();//Thread main este blocat pana cand se termina thread 2
    //producerThread.join();//Thread main este blocat pana cand se termina thread 2
} catch(Exception e) {
    System.out.println("Exceptie la join");//Afisare mesaj aparitie exceptie
    e.printStackTrace();//Afisare detalii exceptie aparuta
}

// Transform din vector de bytes in Buffered image!!!
//ByteArrayInputStream bais = new ByteArrayInputStream(buff.getVect());
ByteArrayInputStream bais = new ByteArrayInputStream(consumerThread.getVect());

try {
    img = ImageIO.read(bais); //Transformare din vector de biti in BufferedImage!!!
} catch (IOException e) {
```

In final aceasta imagine se pune in locul variabilei locale de clasa si astfel se rezolva citirea completa a imaginii din fisier impreuna cu transformarea vectorului de bytes in BufferedImage.

7)Interfata convertedImageInterface

Aceasta interfata are rolul de a specifica metodele ce urmaeaza a fi implementate in clasa ConvertedImage, si anume, o metoda de prelucrare a imaginii citite(Algoritmul Average Method), o metoda de scriere a imaginii prelucrate in fisier si o metoda de returnare a imaginii prelucrate(de tip BufferedImage).

8)Clasa convertedImage

Aceasta clasa mosteneste clasa initialImage si implementeaza interfata detaliata anterior. Aceasta mosteneste constructorul clasei parinte, metoda de citire, dar si imaginea de tip BufferedImage(care va reprezenta imaginea citita). Pe langa acestea, mai avem o variabila de clasa processedImage in care se va stoca imaginea convertita prin intermediul algoritmului respectiv.

Constructorul acestei clase primeste ca parametru fiserul din care se va citi imaginea initiala, aceasta functie apeland constructorul clase parinte care se va ocupa de tot procesul de citire. De asemenea, odata citita, imaginea este prelucrata prin intermediul algoritmului enuntat la inceputul documentatiei, creandu-se astfel imaginea finala(convertita).

```
for(int x=0;x<height;x++)//Parcurgere linii
{
    list.add(new ArrayList<>());//adaugare linie noua in lista
    for(int y=0;y<width;y++) //Parcurgere coloane
    {
        int pixel=this.initialImage.getRGB(y,x);//Pixelul din imagine de pe linia x si coloana y

        int a = (pixel>>24)&0xff;//Valoara Alpha a pixelului
        int r = (pixel>>16)&0xff;//Valoara Red a pixelului
        int g = (pixel>>8)&0xff;//Valoara Green a pixelului
        int b = pixel&0xff;//Valoara Blue a pixelului

        int avg = (r+g+b)/(level+2); //Metoda mediei

        int newPixel = (a<<24) | (avg<<16) | (avg<<8) | avg;//Gasim valoarea noului pixel pentru a obtine imaginea dorita

        list.get(x).add(newPixel);//adaugare pixel nou(prelucrat) in lista(linia x si coloana y)
    }
}
```

In final acesta se scrie intr-un fisier dat prin intermediul unei functii care primeste ca parametru fisierul respectiv.

9)Clasa de teste – Main

Aceast clasa se ocupa de achizia datelor de tastura pentru a identifica fisierul sursa, fisierul destinatie, dar si nivelul de al-negru dorit a fi aplicat

imaginii. De asemenea, metoda main se ocupa de comanda citirii, dar si de comenzile de prelucrare si de scriere a rezultatului final in fisier.

6)Performante si timpi de executie

Datorita intergrarii in procesul de citire a scenariului Producer-Consumer, timpul de necesar citirii imaginii din fisier este mai crescut decat timpul necesar executiei propriu zise a algoritmului ori scrierii simple a imaginii prelucrate in fisierul destinatie.

Astfel, pentru citirea diferitelor imagini in format BMP 24 bit RGB, s-au obtinut urmatoarele performante:

Exemplul 1:

```
Durata citire imagine din fisier: 4726 milisecunde
Durata prelucrare imagine Convert to Grey-Scale: 84 milisecunde
Durata scriere imagine in fisier: 214 milisecunde
```

Exemplul 2:

```
Durata citire imagine din fisier: 4460 milisecunde
Durata prelucrare imagine Convert to Grey-Scale: 165 milisecunde
Durata scriere imagine in fisier: 624 milisecunde
```

Exemplul 3:

```
Durata citire imagine din fisier: 4390 milisecunde
Durata prelucrare imagine Convert to Grey-Scale: 68 milisecunde
Durata scriere imagine in fisier: 126 milisecunde
```

7)Concluzie

In concluzie, scenariul Producer-Consumer implementat in cadrul citireii multithreading a unei imagini din fisier poate fi aplicat cu succes asupra oricarei imagini de tip BMP, iar convertirea acesteia la scara alb-negru prin intermediul Metodei Mediei reprezinta o modalitate facila si eficienta de a prelucra o imagine cu scopul de a ajunge la rezultatul dorit.

8)Bibliografie

https://stackoverflow.com/questions/9131678/convert-a-rgb-image-to-grayscale-image-reducing-the-memory-in-java

https://memorynotfound.com/convert-image-grayscale-java/

https://www.tutorialspoint.com/java_dip/grayscale_conversion.htm