Stefan Andreea-Constanta

Grupa 332 AB

Documentatie proiect

Tema proiectului: Image resizing (Zooming +/-) – keeping aspect ratio. Pixel replication method.

1. Descrierea algoritmului de procesare

Algoritmul de replicare a pixelilor consta in copierea fiecarui pixel atat pe linie, cat si pe coloana de un numar de ori egal cu factorul de marire a imaginii.

Exemplu:

Avem o imagine de 2 randuri si 2 coloane si vrem sa o marim de doua ori utilizand aceasta metoda. Presupunem ca imaginea poate fi inteleasa ca urmatoarea matrice cu valorile pixelilor:

1 2

3 4

Se copiaza pixelii rand cu rand si coloana cu coloana, obtinandu-se urmatoarea matrice:

1 1 2 2

1 1 2 2

3 3 4 4

3 3 4 4

Se poate observa ca imaginea originala de 2x2 pixeli, a devenit acum o imagine de 4x4 pixeli, adica imaginea originala a fost marita de doua ori.

2. Descrierea arhitecturii structurale si functionale

Am realizat un proiect in care, prin metoda replicarii pixelilor, se mareste o imagine aleasa de utilizator. Imaginea de intrare si factorul de zooming (de cate ori se mareste imaginea) vor fi date ca parametri de intrare din linia de comanda (sau din Run Configurations in cazul in care este rulat in eclipse), iar numele imaginii de iesire (dupa procesare) va fi introdus de la tastatura.

Aplicatia contine urmatoarele clase:

- MainClass.java
- BmpFile.java
- ReadAndProcessBmpFile.java (implementeaza interfata)
- ReadBmpFile.java

- AbstractClass.java (mosteneste clasa Thread)
- Consumator.java (mosteneste clasa AbstractClass)
- Producator.java (mosteneste clasa AbstractClass)
- InputFileInterface.java (interfata)
- Utils.java

Clasa BmpFile.java

Aceasta clasa descrie un obiect de tip bmp. Contine informatiile specifice unui fisier bmp, care vor fi utilizate in cadrul procesarii: numele fisierului, un vector care contine headerul, latimea imaginii in pixeli, inaltimea imaginii in pixeli, dimensiunea unei linii de bytes (aceasta va reprezenta lungimea unui rand dintr-o matrice de bytes), dimensiunea imaginii fara header si fara padding, dimensiunea imaginii fara header, dimensiunea totala a paddingului si numarul total de bytes din fisier.

Clasa ReadAndProcessBmpFile.java

In aceasta clasa sunt implementate mai multe metode.

a) Metoda *public BmpFile processInputFile(String fileName)*

In aceasta metoda se declara un obiect de tip BmpFile, se citesc din fisierul de intrare primii 54 bytes care reprezinta headerul fisierului. Se seteaza dimensiunile pentru obiectul declarat, se calculeaza si se seteaza dimensiunea unei linii (din matricea in care se vor stoca bytes specifici imaginii, adica cei care reprezinta pixelii) si paddingul total pe care il are fisierul de intrare.

b) Metoda *public static int[][] transformBmpFileToBytesMatrix(BmpFile in)*

Aceasta metoda primeste fisierul de intrare ca parametru, citeste din fisier de la sfarsitul headerului pana la finalul fisierului si adauga valorile citite intr-o matrice care va reprezenta matricea de bytes ce definesc pixelii imaginii, iar ulterior afiseaza aceasta matrice.

c) Metoda *public static int[][] transformBytesMatrixToPixelsMatrix(BmpFile in, int[][] bytesMatrix, int scale)*

In aceasta metoda se transforma pe rand cate 3 valori din matricea de bytes intr-o singura valoare intreaga si se pun aceste valori rezultate intr-un vector. Se realizeaza aceasta transformare deoarece in fisier nu se gasesc pixelii sub forma unei singure valori corespunzatoare fiecarui pixel. In fisier un pixel este reprezentat de cate 3 valori consecutive care reprezinta culoarea acestuia. Mai departe, acest vector de pixeli realizat este folosit in algoritmul de copiere. Se copiaza fiecare valoare de un numar de ori egal cu parametrul scale (care este dat de utilizator la inceputul programului din linia de comanda) si toate valorile,

atat cele care existau in vector, cat si cele copiate se pun intr-un nou vector. Acest ultim vector este trecut ulterior intr-o matrice si se face afisarea acesteia.

d) Metoda *public static BmpFile initOutputBmpFile(BmpFile in, int[][] newPixelsMat, String outFileName)*

In aceasta clasa se calculeaza dimensiunile finale ale matricei de bytes care reprezinta pixelii imaginii de iesire (imaginii procesate) si paddingul care trebuie adaugat pe fiecare linie a matricei. Se transforma toate valorile din matricea creata anterior care reprezentau pixelii, inapoi in bytes (adica fiecare valoare se separa inapoi in cele 3 valori care reprezinta culorile). Se face afisarea matricei finale si se seteaza dimensiunile fisierului de iesire.

e) Metoda *public static BmpFile getOutputMaximizedBmpFile(BmpFile in, BmpFile out)*

In aceasta metoda se realizeaza headerul pentru fisierul nou (imaginea procesata), adica se trec pe pozitiile corespunzatoare dimensiunile noi ale imaginii procesate si se copiaza din headerul vechi valorile care nu se modifica la procesare. Se scrie headerul in fisierul de iesire si de asemenea, se scriu si valorile din matricea creata anterior cu bytes ce reprezinta imaginea de iesire. La finalul acesteia, pentru verificare, se citeste si afiseaza din fisierul nou creat headerul si se afiseaza dimensiunile noi ale imaginii procesate.

Clasa ReadBmpFile.java

Aceasta clasa functioneaza ca un buffer pentru implementarea multithreadingului. Aici sunt realizate cele doua metode get si set specifice consumatorului si producatorului, prin care producatorul transmite informatii, iar consumatorul le preia.

Clasele Producator.java si Consumator.java

Aceste clase mostenesc clasa AbstractClass.java si implementeaza constructori pentru obiectele Producator, respectiv Consumator.

Clasa AbstractClass.java

Aceasta clasa mosteneste clasa Thread. In aceasta clasa este implementata metoda run specifica threadurilor utilizate in proiect. In metoda run, in functie de variabila de tip boolean numita role, se pot diferentia doua cazuri: role=1 inseamna ca este activ threadul Producator, aici se declara un obiect de tip ReadAndProcessBmpFile, se apeleaza metoda *processInputFile* care realizeaza citirea headerului si setarea dimensiunilor necesare procesarii si acest obiect este transmis consumatorului. role=0 inseamna ca este activ threadul Consumator, acesta preia informatiile transmise de producator si apeleaza restul metodelor din clasa ReadAndProcessBmpFile pentru a realiza procesarea si scrierea unui fisier nou.

Cat timp este activ threadul Consumator, threadul Producator este in sleep. La finalul programului ambele threaduri mor.

Clasa Utils.java

Aceasta clasa contine metode utilizate in program.

a) Metoda *public static int[] toBytes(int i)*

Aceasta metoda realizeaza separarea unei valori intregi in 3 valori care reprezinta culorile unui pixel.

b) Metoda *public static int obtainPaddingByNrBytes(int nrBytes)*

In aceasta metoda se calculeaza paddingul care trebuie adaugat pe fiecare rand al matricei de bytes finala, adica cea cu imaginea procesata.

c) Metoda *public static void printOutMessages(String... strings)*

Aceasta metoda reprezinta o functie cu numar variabil de parametri care afiseaza mesaje.

d) Metoda *public static void printOutMessagesWithTimestamp(String... strings)*

Este tot o functie cu numar variabil de parametri, dar pe langa afisarea unui mesaj, aceasta afiseaza si ora momentului in care este folosita.

Clasa MainClass.java

Aceasta este clasa de test in care sunt declarate threadurile si sunt apelate metodele necesare pornirii acestora.

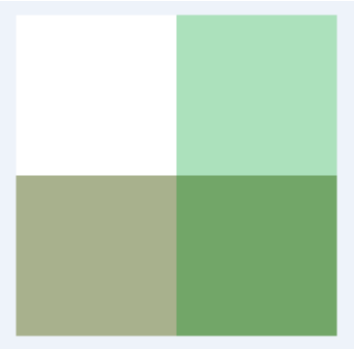
3. Rezultate procesare imagine

a) Imaginea de intrare:

Property	Value
Image	
Dimensions	2 x 2
Width	2 pixels
Height	2 pixels
Bit depth	24

Este o imagine foarte mica, de 2x2 pixeli.

Imaginea marita de 4 ori:



Property	Value
Image	
Dimensions	8 x 8
Width	8 pixels
Height	8 pixels
Bit depth	24

Se poate observa ca noua dimensiune este de 8x8 pixeli.

b) Imaginea de intrare:

Property	Value
Image	
Dimensions	3 x 3
Width	3 pixels
Height	3 pixels
Bit depth	24

Imaginea marita de 3 ori:

Property	Value
Image	
Dimensions	9 x 9
Width	9 pixels
Height	9 pixels
Bit depth	24

c) Imagine de intrare cu dimensiuni diferite (latime diferita de inaltime):

Property	Value
Image	
Dimensions	4 x 3
Width	4 pixels
Height	3 pixels
Bit depth	24

Imagine marita de 2 ori:

Property	Value
Image	
Dimensions	8 x 6
Width	8 pixels
Height	6 pixels
Bit depth	24

Imaginile au fost create in paint deoarece aveam nevoie de imagini de dimensiuni foarte mici pentru a le procesa. Am intampinat o problema la scrierea headerului pentru fisierul de iesire deoarece, daca poza rezultata este foarte mare (dimensiunea depaseste 255 de bytes), dimensiunea totala trebuie salvata pe mai multi octeti si nu am reusit sa implementez aceasta scriere.