IMAGE SMOOTHING

SECELEANU ALEXANDRA-ELENA
GRUPA 332AA

Introducere

Scopul proiectului este de a blura o imagine în format BMP de 24 de biți aflată într-un fișier și de a o scrie în alt fișier. Proiectul a fost realizat in JAVA, utilizand ca IDE Eclipse Mars.2 Release.

Blurarea se face prin utilizarea unei matrici de convoluție (kernel). În procesarea imaginilor, un kernel, o matrice de convoluție sau o mască este o matrice mică. Acesta se folosește pentru blurring, sharpening, ștanțare, detectare a marginilor și multe altele. Acest lucru se realizează prin efectuarea unei convoluții între un kernel și o imagine.

Expresia generală a unei convoluții este:

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x-dx, y-dy),$$

where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, ω is the filter kernel.

Convoluția este procesul de adăugare a fiecărui element al imaginii către vecinii săi locali, ponderați de kernel. Aceasta se referă la o formă de convoluție matematică. Trebuie notat faptul că operația matricei care se efectuează nu este o multiplicare matricială tradițională, în ciuda faptului că este similară cu *. De exemplu, dacă avem două matrice trei-trei, primul nucleu, iar cel de-al doilea o imagine, convoluția este procesul de răsturnare a rândurilor și a coloanelor kernel-ului și apoi de multiplicare a intrărilor și sumărilor similare la nivel locala.

Functionalitatile aplicatiei :

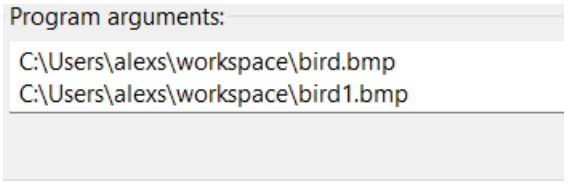
1. Preia de la tastatură sau din linia de comanda calea fișierului sursă în care se află poza ce urmează sa fie blurată.
2. Preia de la tastatură sau din linia de comanda calea fișierului destinație în care se va scrie poza blurată.
3. Citește imaginea (BMP) din fișierul sursă.
4. Transmite imaginea transformată în matrice din Producer în Consumer (câte ¼ din poză o dată).
5. Calculează timpul de citire.
6. Procesează imaginea.
7. Calculează timpul de procesare.
8. Scrie imaginea
9. Calculează timpul de scriere.
10. Afișează timpii de procesare.
11. Afișează imaginea inițială și pe cea procesată.

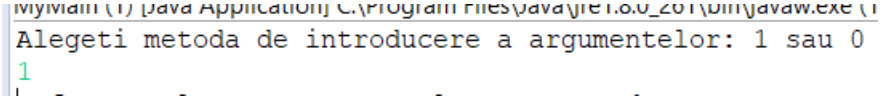
Scrierea path-ului in consola (apasam tasta 0) :

```
MyMain (1) [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (
Alegeti metoda de introducere a argumentelor: 1 sau 0
```

```
Alegeti metoda de introducere a argumentelor: 1 sau 0
0
Calea fisierului de unde se citeste poza
Fisier(BMP):
C:\Users\alexs\workspace\cat.bmp
Calea fisierului unde se va face scrierea
Fisier(BMP):
C:\Users\alexs\workspace\cat1.bmp
```

Scrierea path-ului in linia de comanda (apasam tasta 1)





Aplicația contine 6 clase Java:

- Producer
- Consumer
- ProdCons
- Smoothing
- Buffer
- Main

Și o interfață: Interface.

Clasa Producer transmite câte un sfert din imagine (din matrice) către Consumer. Moștenește clasa ProdCons.

```
1  package ImageSmoothing;
2
3  public class Producer extends ProdCons {
4      Producer() { // constructor fara parametri
5          super();
6      }
7      public Producer(Buffer buf) {
8          super(true, buf);// apelam constructorul clasei ProdCons
9          // am setat producer = true
10         System.out.println("A fost apelat constructorul pentru Producer");
11     }
12 }
13
```

Clasa Consumer preia datele (sfertul) de la Producer și moștenește clasa ProdCons

```
1  package ImageSmoothing;
2
3  public class Consumer extends ProdCons {
4      Consumer() { // constructor fara parametri
5          super();
6      }
7      public Consumer(Buffer buf) {
8          super(false, buf);// apelam constructorul clasei ProdCons
9          // am setat producer = false
10         System.out.println("A fost apelat constructorul pentru Consumer");
11     }
12 }
13
```

Clasa ProdCons folosește o variabilă de tip Boolean (producer) pentru a verifica dacă este activ Producer-ul sau Consumer-ul.

În cazul Producer-ului, aceasta împarte imaginea în 4 sferturi și transmite mai departeConsumer-ului pe rând câte un sfert.

În cazul Consumer-ului, se preia informația transmisă de Producer prin Buffer și o transmite mai departe spre procesare.

Aceasta clasa folosește incapsulare, este o clasă abstracta si moșteneste clasa Thread.

Astfel, se demonstrează cele 3 niveluri de moștenire.

Cu ajutorul funcției sleep(1000) si a output-urilor se evidențiază etapele comunicării:

```
C:\Users\alexs\workspace\bird1.bmp
A fost apelat constructorul pentru Producer
A fost apelat constructorul pentru Consumer
Producatorul a pus sfertul cu numarul 1 al imaginii.
Consumer-ul preia sfertul 1 din imagine.
Producatorul a pus sfertul cu numarul 2 al imaginii.
Consumer-ul preia sfertul 2 din imagine.
Producatorul a pus sfertul cu numarul 3 al imaginii.
Consumer-ul preia sfertul 3 din imagine.
Producatorul a pus sfertul cu numarul 4 al imaginii.
Consumer-ul preia sfertul 4 din imagine.
```

Clasa Smoothing:

- constructor care citește imaginea din fișier
- instanțiază kernel-ul
- matricea de convoluție prin instanțierea BufferedImageOp op = new ConvolveOp(kernel)
- modifică imaginea prin realizarea convoluției dintre imagine și matricea de convoluție folosind metoda imagineProcesata = op.filter(imagineInit, null)
- calculează timpii de procesare, citire și scriere

```
 Imaginea a fost procesata cu succes!
Citirea imaginii a durat: 0.031 secunde

Procesarea imaginii a durat: 0.007 secunde

Scrierea imaginii a durat: 0.031 secunde
```

Clasa Buffer evidențiază sincronizarea, reprezentând legătura dintre Producer șiConsumer. Producer-ul ia informație din matrice (imagineInit) și o pune în Buffer. Consumer-ul ia informația din Buffer și o transmite mai departe spre procesare (imaginePentruProcesare).

Clasa Main conține cele două moduri de preluare a argumentelor si include varargs în cadrul metodelor de preluare de la tastatura a argumentelor:

```
    else
    {
        //citire path fisier de editat
        display("Calea fisierului de unde se citeste poza","Fisier(BMP):");
        Scanner scanner1 = new Scanner(System.in);
        String in = scanner1.nextLine();

        //citire path fisier pentru scrierea imaginii procesate
        display("Calea fisierului unde se va face scrierea","Fisier(BMP):");
        Scanner scanner2 = new Scanner(System.in);
        String out = scanner2.nextLine();

        Smoothing mySmoothing = new Smoothing(in, out); // declar un obiect de tip Smoothing
        mySmoothing.startSmoothing(); // start aplicatie

        scanner.close(); // opresc scanner-ul pentru metoda de preluare a argumentelor
        scanner1.close(); // opresc scanner-ul pentru citirea path-ului fisierului de editat
        scanner2.close(); // opresc scanner-ul pemtru citirea path-ului fisierului in care scriu imaginea procesata
    }
}
```

Interface : conține metoda de pornire a aplicație

Bibliografie

1. https://en.wikipedia.org/wiki/Kernel_(image_processing)
2. https://www.w3schools.com/java/java_interface.asp
3. https://docs.oracle.com/javase/7/docs/api/java/awt/image/Kernel.html
4. https://docs.oracle.com/javase/7/docs/api/java/awt/image/ConvolveOp.html
5. https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImageOp.htm l
6. http://www.java2s.com/Code/Java/AdvancedGraphics/A3x3kernelthatblursanim age.htm
7. https://alvinalexander.com/blog/post/java/getting-rgb-values-for-each-pixel-in- image-using-java-bufferedi
8. https://www.geeksforgeeks.org/image-processing-java-set-1-read-write/
9. https://www.tutorialspoint.com/commons_cli/commons_cli_quick_guide.htm