

# **Image Smoothing**

Proiect – APLICAȚII WEB CU SUPTOR JAVA

## Cuprins

Introducere .....	3
Partea Teoretică .....	3
Descrierea Funcţională a Aplicaţiei .....	4
Descrierea Arhitecturală a Aplicaţiei .....	4
MainClass .....	5
ReadImage .....	5
SmoothImage .....	6
WriteImage .....	7
TimeInterface .....	7
Bibliografie.....	8

## Introducere

Proiectul are scopul de a blura o imagine în format BMP aflată într-un fișier și de a o scrie în alt fișier. Acest lucru a fost realizat în Java, iar ca modalitate de procesare a imaginii am folosit masca de convoluție.

## Partea Teoretică

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. 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:**

$$g(x, y) = (\omega * f)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x - s, y - t),$$

*g*-image procesată; *f*-image inițială; *w*-kernel

În funcție de valorile elementului, un kernel poate provoca o gamă largă de efecte, cele de smoothing având și ele variații:

<b>Box blur</b> (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
<b>Gaussian blur 3 × 3</b> (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
<b>Gaussian blur 5 × 5</b> (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

**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 matricii care se efectuează - convoluția - 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 însumărilor similare la nivel local.

## Descrierea Funcțională a Aplicației

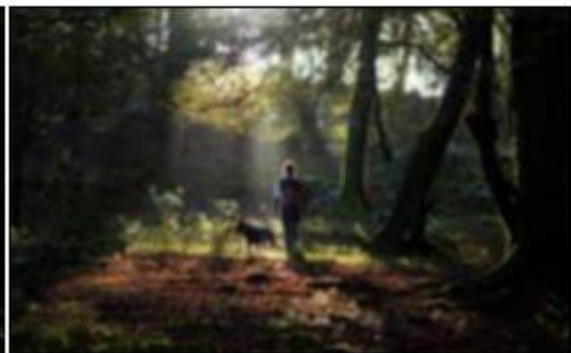
Aplicația urmărește următorii pași:

1. Preia de la tastatură calea fișierului sursă, în care se află poza ce urmează să fie blurată.
2. Preia de la tastatură calea fișierului destinație, în care se va scrie poza blurată.
3. Citește imaginea (**BMP**) din fișierul sursă.
4. Calculează timpul de citire
5. Procesează imaginea
6. Calculează timpul de procesare
7. Scrie imaginea
8. Calculează timpul de scriere
9. Afișează timpii de procesare

INPUT



OUTPUT



## Descrierea Arhitecturală a Aplicației

Aplicația constă în **4 clase Java**:

- MainClass
- ReadImage
- SmoothImage
- WriteImage

De asemenea, aceasta conține și o interfață: TimeInterface

## MainClass

În această clasă sunt preluate datele de la tastatură(input/output) şi se realizează citirea, procesarea şi scrierea pozei din fişierul sursă prin instanţierea clasei *WriteImage* şi apelarea metodei *writeImageToFile* a acesteia.

Mai mult, această clasă conţine metoda *display* ce are parametrii de tipul *varargs* ce este folosită pentru scrierea textului în consolă.

De asemenea se afişează şi timpul de citire, procesare şi scriere prin apelarea metodelor *readTime*, *smoothImageTime* şi *writeTime*.

## ReadImage

Această **clasă abstractă** are scopul de a **citi imaginile** din fişierul introdus de la tastatură. De asemenea, aceasta **implementează interfaţa *TimeInterface*** pentru a folosi metodele de calcul al timpului din clasa respectivă.

Pentru a citi imaginea, această clasă importă **clasa Java *java.awt.image.BufferedImage***. Clasa Java *BufferedImage* este o subclasă a clasei *Image*. Este folosită pentru a manipula datele de tip *image*.

**Clasa *ReadImage*** conţine următoarele metode:

- **public *BufferedImage* getImage()**

o constructor

- **public void readImageFromFile (String fileName)**

o citeşte poza din fişierul sursă folosind *ImageIO.read(file)*, ce este o metodă din cadrul bibliotecii *java.awt.image.BufferedImage*

o calculează timpul de citire

- **public long readTime()**

o scrie timpul de citire

- **public abstract long smoothImageTime ()**

o metodă abstractă

- **public abstract long writeTime()**

o metodă abstractă

## SmoothImage

Această **clasă abstractă** are scopul de a **procesa imaginea citită**. Această clasă moşteneşte clasa **ReadImage**.

Clasa *ReadImage* importă următoarele clase Java:

- *java.awt.image.BufferedImage* (folosită pentru a manipula datele de tip imagine)
- *java.awt.image.BufferedImageOp* (descrie operaţiuni single-input / single-output efectuate asupra obiectelor *BufferedImage*)
- *java.awt.image.ConvolveOp* (implementează o convoluţie de la sursă la destinaţie - convoluţia utilizând un kernel de convoluţie este o operaţie spaţială care calculează pixelul de ieşire de la un pixel de intrare, înmulţind kernelul cu surround-ul pixelului de intrare - acest lucru permite ca pixelul de ieşire să fie afectat de vecinătatea imediată într-un mod care poate fi specificat matematic cu ajutorul unui kernel – operează asupra obiectelor *BufferedImage*)
- *java.awt.image.Kernel* (defineşte o matrice care descrie modul în care un pixel specificat şi pixelii din jur afectează valoarea calculată pentru poziţia pixelilor în imaginea de ieşire a unei operaţii de filtrare - originea X şi originea Y indică elementul matricei kernelului care corespunde poziţiei pixelilor pentru care se calculează o valoare de ieşire)

Clasa **SmoothImage** conţine următoarele **metode**:

- **public SmoothImage ()**

o constructor care citeşte imaginea din fişier utilizând metoda *getImage* din clasa *ReadImage*

- **public void smooth()**

o instantiaza kernel-ul

o matricea de convoluţie prin instanţierea *BufferedImageOp* *op = new ConvolveOp(kernel)*

o modifică imaginea prin realizarea convoluţiei dintre imagine şi matricea de convoluţie folosind metoda *this.imageS = op.filter(super.getImage(), null);*

o calculează timpul de procesare

- **public BufferedImage getImageS()**

o returnează imaginea procesată

- **public long smoothImageTime()**

o afişează timpul de procesare

## WriteImage

Această clasă are scopul de a scrie imaginea procesată. Această clasă **moşteneşte** clasa **SmoothImage**.

**Clasa WriteImage** conţine următoarele metode:

- **public WriteImage ()**

- o constructor ce foloseşte returnează imaginea procesată folosind metoda *getImageS* moştenită din clasa *SmoothImage*

- **public static BufferedImage getImageW()**

- o getter ce returnează imaginea

- **public WriteImage(String fileName)**

- o setter ce returnează imaginea procesată

- o citeşte imaginea din fişier folosind metoda *readImageFromFile* moştenită din clasa *ReadImage*

- o procesează imaginea citită folosind metoda *smooth* moştenită din clasa *SmoothImage*

- o returnează imaginea procesată folosind metoda *getImageS* moştenită din clasa *SmoothImage*

- **public void writeImageToFile(String fileName) throws FileNotFoundException, IOException**

- o scrie în fişier folosind metoda **FileOutputStream(fileName)**

- o calculează timpul de scriere în fişier

- o *FileNotFoundException* semnalează că o încercare de a deschide fişierul denotat de un nume de cale specificat a eşuat. Această excepţie va fi aruncată de constructorii *FileInputStream*, *FileOutputStream* şi *RandomAccessFile* când nu există un fişier cu numele căii specificate. Acesta va fi, de asemenea, aruncat de către aceşti constructori dacă fişierul există, dar din anumite motive este inaccesibil, de exemplu atunci când se face o încercare de a deschide un fişier de citire numai pentru scriere.

- o *IOException* semnalează că sa produs o excepţie I / O de un fel.

- **public long writeTime()**

- o afişează timpul de scriere

## TimeInterface

Interfaţa conţine 3 metode abstracte implementate de clasa *ReadImage* şi moştenite de clasele *SmoothImage* si *WriteImage*.

## Bibliografie

1. [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_smoothing\)](https://en.wikipedia.org/wiki/Kernel_(image_smoothing))
2. [https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)
3. <https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html>
4. <https://docs.oracle.com/javase/7/docs/api/java/io/FileNotFoundException.html>
5. <https://docs.oracle.com/javase/7/docs/api/java/awt/image/Kernel.html>
6. <https://docs.oracle.com/javase/7/docs/api/java/awt/image/ConvolveOp.html>
7. <https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImageOp.html>
8. <http://www.java2s.com/Code/Java/Advanced-Graphics/A3x3kernelthatblursanimage.htm>
9. [Smoothing Binary Image using 3X3 filter mask, Java Implementation | Nayef's Blog \(wordpress.com\)](#)