



ULB Sibiu

# Filtrare liniară de netezire

**Andrei-Nicolae Căluțiu**

Sursă aplicație – Git

Procesarea Imaginilor| 2022



# Cuprins

- ✦ **Introducere**
- ✦ **Prezentarea problemei**
- ✦ **Fundamente teoretice**
- ✦ **Soluția propusă**
- ✦ **Prezentarea aplicației**

# Introducere

În procesarea imaginilor, procesul de filtrare liniară, numit și convoluție, este folosit la aplicarea efectelor de blurring, sau noise reduction, fie pentru a elimina anumite detalii, fie pentru extragerea de informații specifice, cum ar fi contururi, sau margini. Prin asta ne referim la eliminarea elementelor neimportante, adică acei pixeli prea mici în raport cu masca de filtrare aplicată. Acest proces de convoluție presupune transformarea unei imagini inițiale, prin aplicarea unui nucleu matricial asupra fiecărui pixel al imaginii și celor din vecinătatea lui. Dimensiunea matricii și valorile ei vor determina efectul transformării, fie un efect de blur, noise reduction, sharpening, sau edge detection.



Sursă aplicație – Git

# Fundamente teoretice

Vom folosi două abordări pentru aplicarea acestor filtre:

01

Utilizarea metodei *filter*, din cadrul clasei *ConvolveOp* al API-ului Java, care aplică procesarea de convoluție fiecărei componente din imaginea sursă, inclusiv componentei alfa, dacă există.

02

Dezvoltarea unei metode proprii, pentru aplicarea procesării descrie la "prezentarea problemei", anume crearea unei imagini cu aceleași proprietăți ca cea originală, pe care o compunem din cea originală, cu aplicarea nucleului asupra întregii suprafețe.





# Prezentarea problemei

Pag. 05

Procesul de convoluție presupune 3 pași:

1

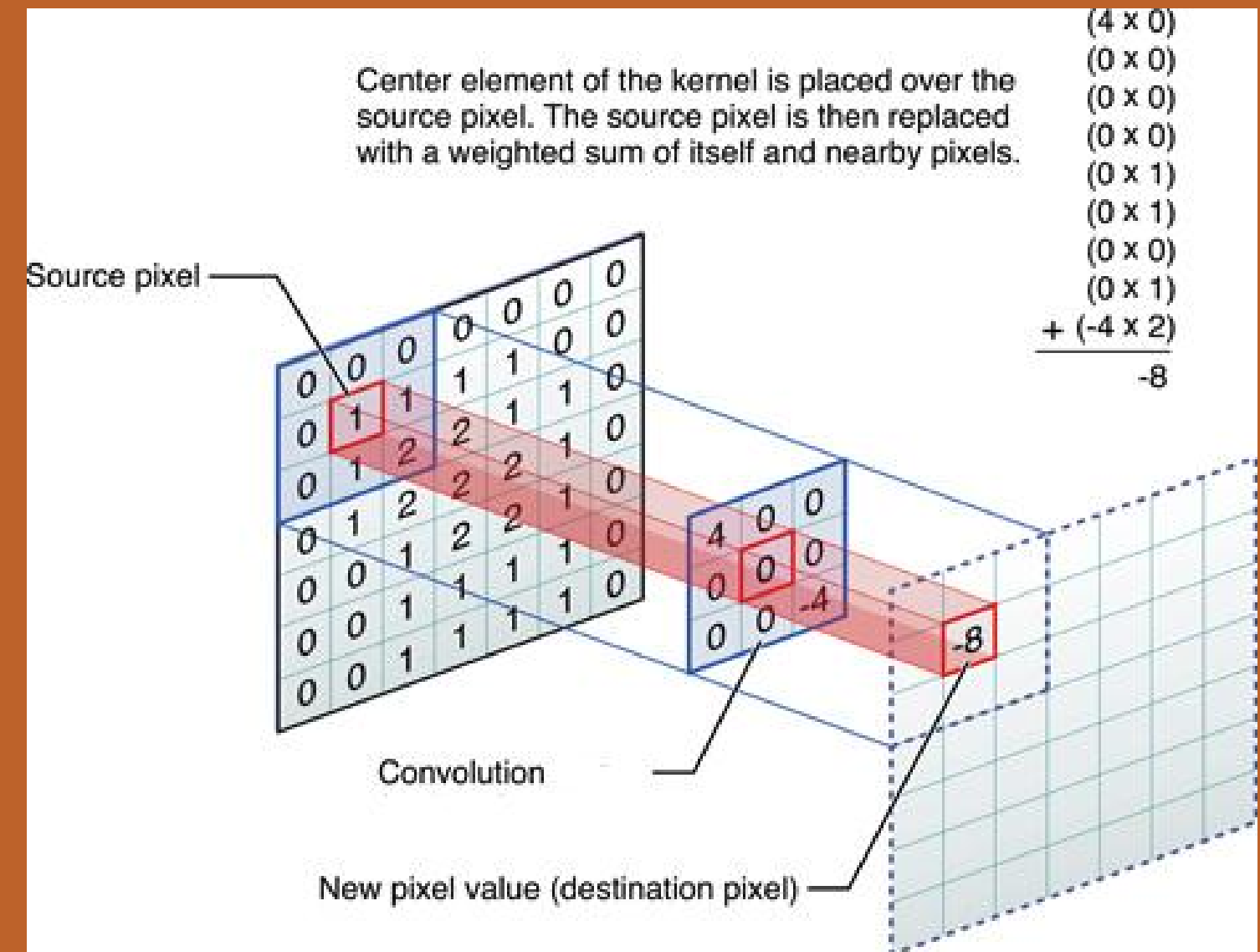
Matricea nucleului este aplicată peste fiecare pixel al imaginii originale (efectuând o verificare pentru a ne asigura că nucleul rămâne mereu suprapus imaginii și nu trece de marginile acesteia) apoi se multiplică valorile nucleului cu valorile imaginii peste care se oglindește.

2

Suma valorilor multiplycate este pusă apoi în locul pixelului central, în poza rezultată.

3

Procesul este repetat apoi, pe suprafața întregii imagini.



[medium.com/@bdhuma](https://medium.com/@bdhuma)

Sursă aplicație – Git

# Soluția propusă

## Aplicație grafică

Propunem dezvoltare unei aplicații Java cu interfață grafică, ce va permite alegerea algoritmului care să aplice filtrul de netezire, cel din API-ul Java sau cel scris de noi, dar și alegerea arbitrară a informațiilor nucleului de mască.

Acesta poate fi preîncărcat sub formă de fișier, sau putem alege o dimensiune pentru el, caz în care aplicația va genera un nucleu de mediere de această dimensiune.

Sursă aplicație – Git

```
protected void onBlur(BufferedImage inputImage, Kernel kernel) {
    long startTime = System.nanoTime();
    BufferedImage blurredImage = new BufferedImage(inputImage.getWidth(), inputImage.getHeight(), inputImage.getType());
    int kernelWidth = kernel.getWidth();
    int kernelRadius = kernelWidth / 2;
    float[] kernelData = kernel.getKernelData(null);
    int kernelDataIndex;

    for (int band = 0; band < inputImage.getRaster().getNumBands() && band < 3; band++)
        for (int y = 0; y < inputImage.getHeight(); y++)
            for (int x = 0; x < inputImage.getWidth(); x++) {
                float gray = 0;
                kernelDataIndex = 0;

                for (int ky = -kernelRadius; ky <= kernelRadius; ky++)
                    for (int kx = -kernelRadius; kx <= kernelRadius; kx++) {
                        if ((x + kx) < 0 || (x + kx) > inputImage.getWidth() - 1 || (y + ky) < 0 || (y + ky) > inputImage.getHeight() - 1)
                            gray += 0;
                        else
                            gray += kernelData[kernelDataIndex] * inputImage.getRaster().getSample(x + kx, y + ky, band);
                        kernelDataIndex++;
                    }
                blurredImage.getRaster().setSample(x, y, band, Utils.constrain(Math.round(gray)));
            }
    blurredImagePanel.setImage(blurredImage);
    elapsedTime = System.nanoTime() - startTime;
}
```



# Prezentarea aplicației

Pag. 07

Sursă aplicație – Git

## Inițializare

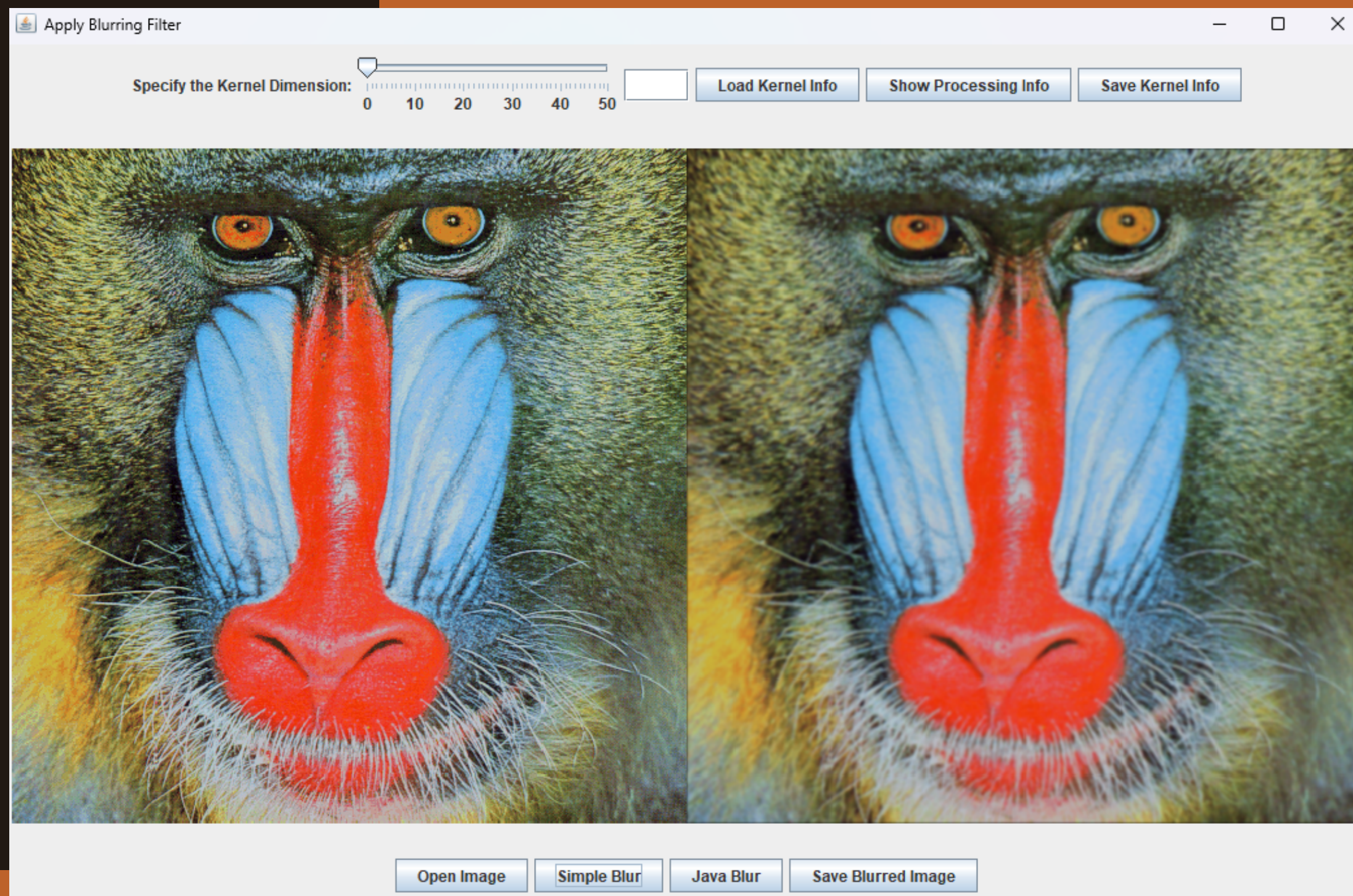
În primă fază trebuie să alegem o poză inițială de pe disk, care va fi afișată în panel-ul din stînga, apoi trebuie să încărcăm informațiile despre nucleu. Dacă nu avem unul existent, într-un fișier, putem alege cu slider-ul aplicației o dimensiune, pentru care aplicația va genera un nucleu de mediere

## Filtrare

În acest moment, cu butoanele "Simple Blur", sau "Java Blur" alegem care variantă a algoritmului de filtrare să se aplice asupra pozei aleasă. Imaginea filtrată va apărea în panel-ul din dreapta, iar cu butonul "Show Processing Info", putem afișa informații despre nucleul folosit, sau timpul de rulare.

## Salvare

În final, aplicația ne permite să salvăm atît imaginea filtrată, cît și informațiile despre nucelul folosit.





ULB Sibiu

Sursă aplicație – Git

# Mulțumesc!

**Andrei-Nicolae Căluțiu**

Procesarea Imaginilor | 2022