ULB Sibiu

Filtrare liniară de netezire

Andrei-Nicolae Căluțiu

Sursă aplicație - Git

Procesarea Imaginilor| 2022

Cuprins

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 descrise la *"prezentarea problemei"*, anume crearea unei imagini cu aceleași proprietăți precum cea originală, pe care o compunem din cea originală, cu aplicarea nucleului asupra pixelilor de pe întreaga suprafață.

Prezentarea problemei

Procesul de convoluție presupune 3 pași:

1

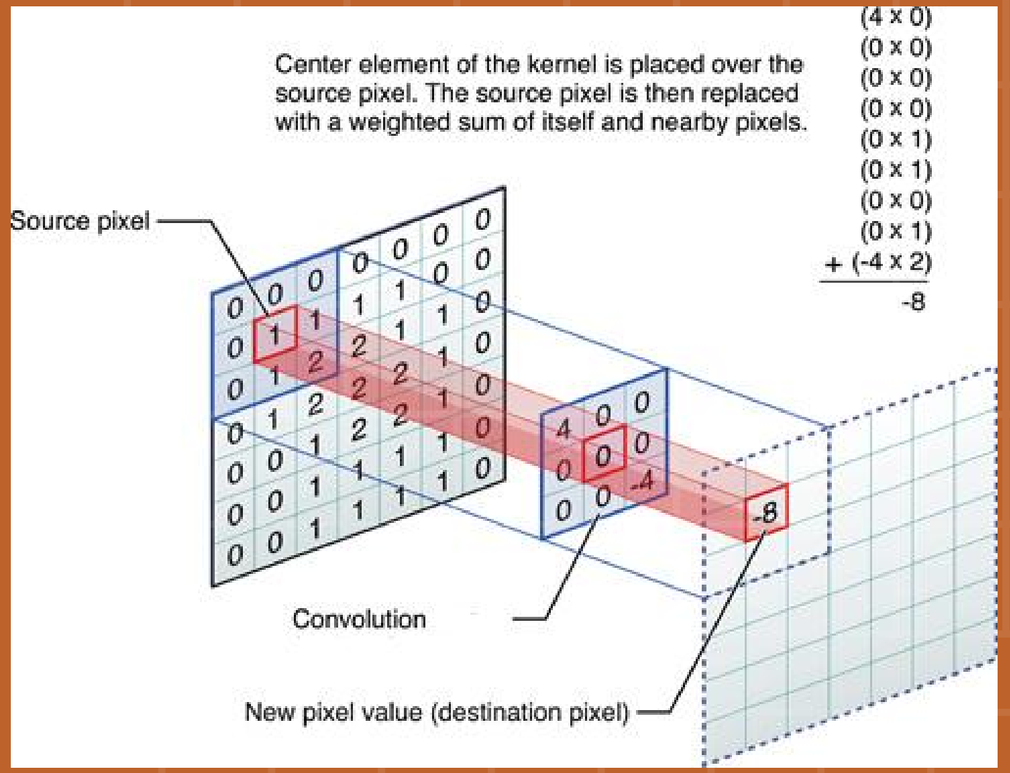
Matricea nucleului este aplicată peste fiecare pixel al imaginii originale (efectuând o verificare pentru a ne asigură că nucleul rămâne mereu suprapus imaginii și nu trece de marginile acesteia) apoi se multiplică valorile nucleului cu valorile imaginii peste care se oglindește.

2

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

3

Procesul este repetat pe suprafața întregii imagini.



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 dimnesiune, caz în care aplicația va genera un nucleu de mediere de această dimensiune.

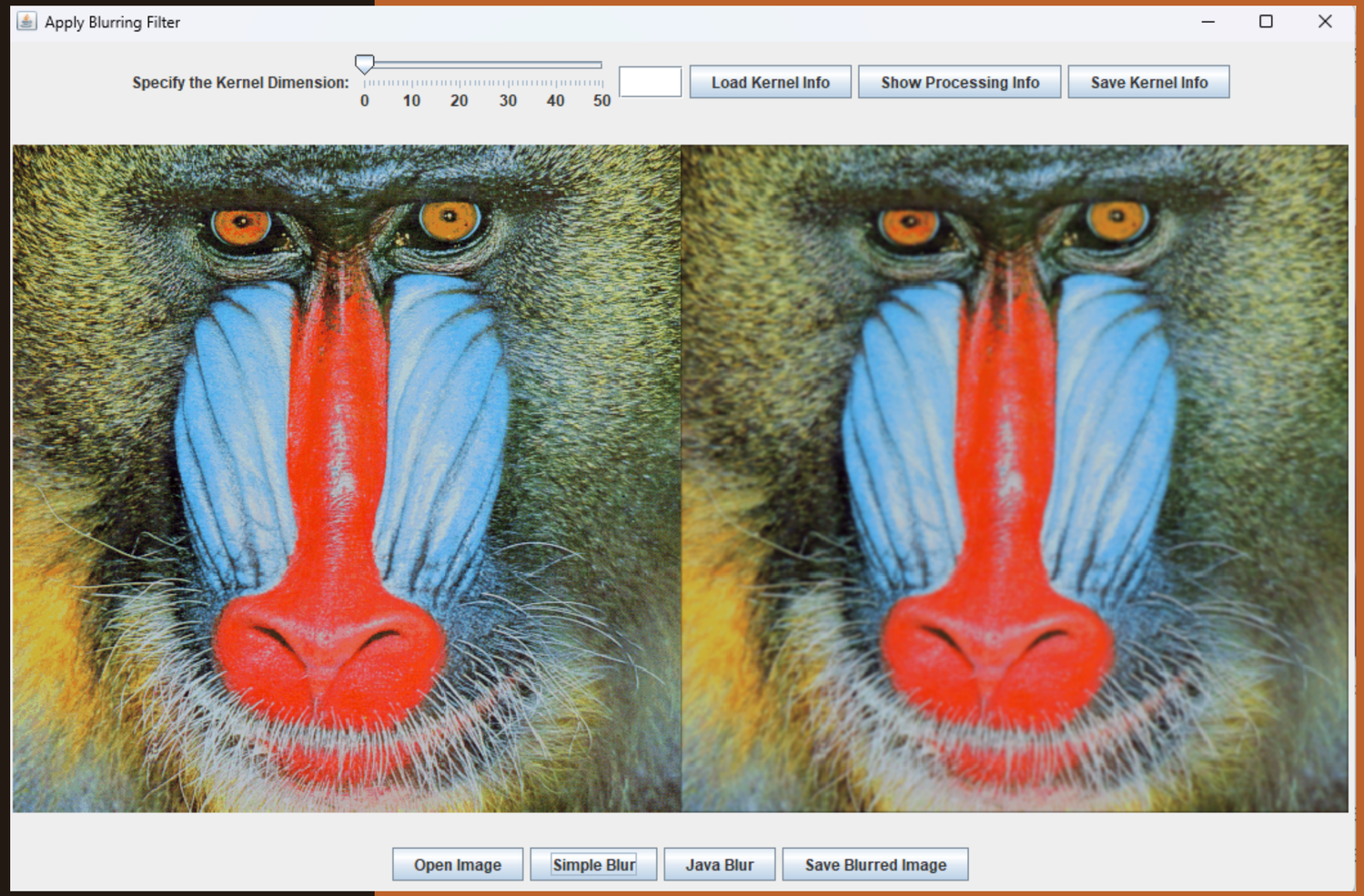
```
protected void onBlur(BufferedImage inputImage, Kernel kernel) {
    long startTime = System.nanoTime();
    BufferedImage blurredImage = new BufferedImage(inputImage.getWidth(), inputImage.getHeight(), inputImage.getType()
    int kernelWidth = kernel.getWidth();
    int kernelRadius = kernelWidth / 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) > inputIm
                            gray += 0;
                        else
                            gray += kernelData[kernelDataIndex] * inputImage.getRaster().getSample( x: x + kx,  y: y + k
                        kernelDataIndex++;
                    }
                blurredImage.getRaster().setSample(x, y, band, Utils.constrain(Math.round(gray)));
            }
    blurredImagePanel.setImage(blurredImage);
    elapsedTime = System.nanoTime() - startTime;
}
```

Sursă aplicație - Git

Prezentarea aplicației



Sursă aplicație - Git

Inițializare

În prima 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", "Java Blur", sau "Nice Blur" alegem varianta algoritmului de filtrare care să fie aplicată 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.

Rezolvarea situației marginilor nemodificate

Aplicarea unui efect de padding

În prima imagine, avem metoda prin care aplicăm imaginii originale un efect de umplere suplimentara, cu pixeli inutili, pentru a ne asigura că efectul de netezire se aplică pe întreaga suprafață esențială a imaginii.

În a doua imagine, avem apelul, în care după efectuarea procesării, afișăm doar o subimagine a celei blurate, din care eliminăm suprafață adăugată suplimentar prin padding.

Sursă aplicație - Git

```java
public static BufferedImage paddedImage(BufferedImage inputImage, int padding)
    if (padding == 0)
        return inputImage;
    BufferedImage processedImage = new BufferedImage(
            width: inputImage.getWidth() + padding * 2,
            height: inputImage.getHeight() + padding * 2,
           BufferedImage.TYPE_INT_ARGB);
    Graphics2D graph = (Graphics2D) processedImage.getGraphics();
    graph.drawImage(inputImage, padding, padding,  observer: null);
    return processedImage;
```

```java
protected void onNiceBlur(BufferedImage inputImage, Kernel kernel) {
    long startTime = System.nanoTime();
    BufferedImage paddingInputImage = Utils.paddedImage(inputImage, kernelSize);
    BufferedImage blurredPaddingImage = new BufferedImage(paddingInputImage.getWidth(),
            paddingInputImage.getHeight(), BufferedImage.TYPE_INT_ARGB);
    ConvolveOp convolveOp = new ConvolveOp(kernel, ConvolveOp.EDGE_ZERO_FILL,  hints: null);
    convolveOp.filter(paddingInputImage, blurredPaddingImage);
    blurredImagePanel.setImage(blurredPaddingImage.getSubimage(kernelSize, kernelSize,
            inputImage.getWidth(), inputImage.getHeight()));
    elapsedTime = System.nanoTime() - startTime;
```

ULB Sibiu

Sursă aplicație - Git

Mulțumesc!

Andrei-Nicolae Călușiu

Procesarea Imaginilor | 2022