Mireanu Cosmin
331AB

Proiect JAVA

Image Sharpening (convolution mask)

➢ Introducere

O gamă largă de tehnologii de procesare a imaginii utilizează operații multipixel cu măști de kernel convoluție, în care fiecare pixel de ieșire este modificat de contribuțiile unui număr de pixeli de intrare adiacenți. Aceste tipuri de operații sunt în mod obișnuit denumite convoluții. În forma cea mai simplă, o operație de convoluție bidimensională pe o imagine digitală utilizează Box Convolution Kernel.

Proiectul are scopul de a face „sharpen” la o imagine in format BMP, aflata intr-un fisier. Acest lucru a fost realizat cu limbajul de programare Java, utilizand ca IDE Eclipe Mars 2 si IntelliJ IDEA, versiunea de Java folosita fiind 8.

➢ Partea Teoretica

O masca de convolutie este o matrice de numere care se foloseste pentru a efectua o operatie de convolutie asupra unei imagini. Convolutia este o operatie matematica care combina valorile din doua seturi de date pentru a produce un nou set de date.

In cazul imaginii, setul de date initial este matricea de pixeli a imaginii. Setul de date secundar este masca de convolutie. Convolutia are ca rezultat o noua matrice de pixeli, care este imaginea rezultata dupa aplicarea mastii de convolutie

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

g – imagine procesata

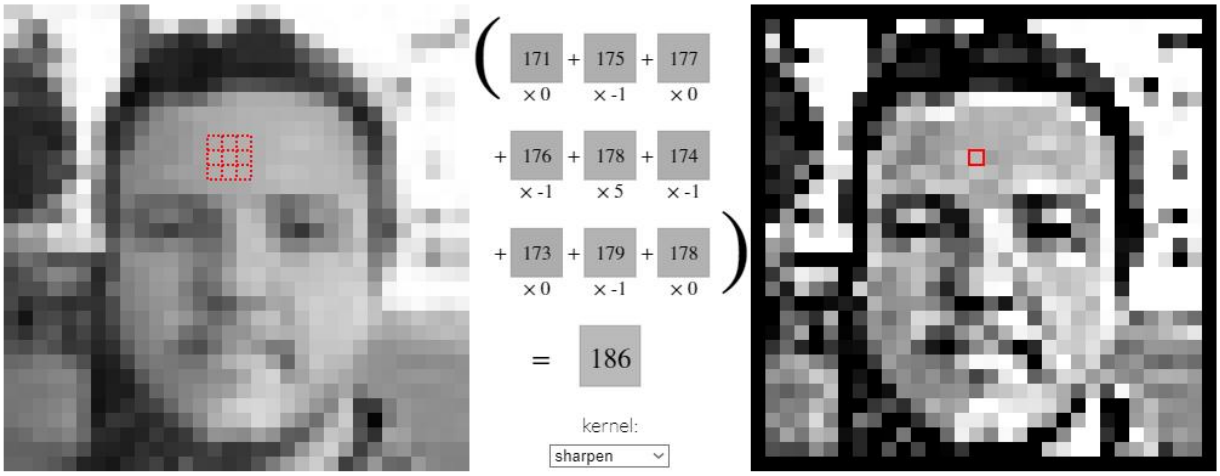
f – imagine initiala

w – kernel

In functie de matricea aleasa, se pot face mult mai multe operatii, precum smoothing, blur si edge detection, dar ne vom concentra pe sharpen.

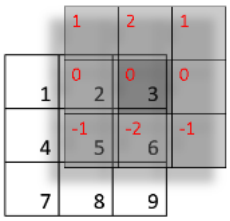
Convoluția este procesul de adăugare a fiecărui element al imaginii către vecinii săi locali, ponderați de kernel. Aceasta se referă la o formă de convoluție matematică. Trebuie notat faptul că operația matricei care se efectuează - convoluția - nu este o multiplicare matricială tradițională, în ciuda 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 prin care se inmulteste fiecare pixel curent cu o valoarea, din care se scade restul de pixeli vecini, inmultiti si ei cu o alta valore.



input image output image

Trebuie luate in calcul si elementele aflate pe margine sau colt, unde apar exceptii.



$$
\begin{aligned}
y[2,0] &= \sum_j \sum_i x[i,j] \cdot h[2-i, 0-j] \\
&= \quad x[1,-1] \cdot h[1,1] + x[2,-1] \cdot h[0,1] + x[3,-1] \cdot h[-1,1 \\
&\quad + x[1,0] \cdot h[1,0] \quad + x[2,0] \cdot h[0,0] \quad + x[3,0] \cdot h[-1,0] \\
&\quad + x[1,1] \cdot h[1,-1] + x[2,1] \cdot h[0,-1] + x[3,1] \cdot h[-1,-1 \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 \\
&\quad + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) \\
&= -17
\end{aligned}
$$

➢ Descrierea Functionala a Aplicatiei

Aplicatia urmareste urmatorii pasi:

1) Preia de la tastatura numele imaginii sursa
2) Preia de la tastatura numele imaginii care va fi rezultata
3) Preia de la tastatura valoarea de sharpen care va fi aplicata
4) Citeste imaginea
5) Se calculeaza timpul de citire
6) Se afiseaza imaginea originala
7) Se prelucreaza imaginea
8) Se calculeaza timpul de procesare

Mireanu Cosmin
331AB

9) Se afiseaza imaginea modificata
10) Se scrie imaginea
11) Se calculeaza timpul de scriere
12) Se afiseaza toti timpii de procesare

Scrierea in consola:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL                Run: Main
PS C:\Users\Cosmin\Desktop\Proiect\ImageSharpen>  c:; cd 'c:\Users\Cosmin\Desktop\Proiect\Imag
eSharpen'; & 'C:\Program Files\Java\jdk1.8.0_181\bin\java.exe' '-cp' 'C:\Users\Cosmin\Desktop\
Proiect\ImageSharpen\bin' 'packTest.Main'
Enter the input name of the image:
PozaNr1
Enter the output name of the image:
PozaNr1_sharp02
Enter the sharpness radius of the image:
0.2
Radius: 0.2
Displaying image...
Displaying sharpened image...
Reading image took 53 ms
Sharpening image took 219 ms
Writing image took 34 ms
```

Imaginea Input

Imaginea Output

➢ Descrierea Arhitecturala a Aplicatiei

Aplicatia consta in 7 clase, separate in 2 pachete, packTest si packWork:

1) Pipe
   In aceasta clasa am descris un pipe, am declarat 2 atribute si am creat constructorul fara parametrii.
2) TimeInterface
   Interfata, unde se regasesc cele 3 functii pentru afisarea timpului fiecarei etape
3) DisplayImage
   Clasa abstacta, implementeaza interfata si care are metoda displayImage, care afiseaza imaginea originala pe ecran.
4) ReadImage
   Clasa care extinde DisplayImage si are metode care citeste imaginea, cele legate de functionalitatea cu threads si suprascrie prima metoda de timp, readTime.
5) SharpenImage
   Clasa care extinde ReadImage si are metode care fac procesarea imaginii, sharpen, o functie constrain, de ajutor pentru a afla minimul dintre 3 valori, suprascrierea metodelor de display si sharpenTime.
6) WriteImage
   Clasa care extinde SharpenImage si are metoda de scrierea imaginii, cu numele dat, si suprascrierea metodei de writeTime.
7) Main
   Clasa principala, unde se citesc de la tastatura numele fisierului de intrare, celui de iesire, un nivel de sharpen si se apeleaza functiile de citire, display, sharpen, write si cele 3 de afisare a timpului.

➢ **Descrierea modulelor**

java.io.PipedInputStream: Această clasă reprezintă o intrare piped, care permite unui fir să citească date scrise de alt fir.

java.io.PipedOutputStream: Această clasă reprezintă o ieșire piped, care permite unui fir să scrie date care pot fi citite de alt fir.

java.io.IOException: Această clasă reprezintă o excepție care apare atunci când apare o eroare de I/O, cum ar fi închiderea unui flux care este încă deschis.

java.io.File: Această clasă reprezintă un fișier pe sistemul de fișiere local.

javax.imageio.ImageIO: Această clasă oferă un set de metode pentru citirea, scrierea și manipularea imaginilor.

java.awt.image.BufferedImage: Această clasă reprezintă o metoda de a manipula datele de tip imagine, care stochează datele imaginii în memorie.

javax.swing.ImageIcon: Această clasă reprezintă o pictogramă care conține o imagine.

javax.swing.JFrame: Această clasă reprezintă o fereastră care servește drept container principal pentru componentele GUI.

javax.swing.JLabel: Această clasă reprezintă o etichetă, care este o componentă GUI care afișează text sau o imagine.

javax.swing.WindowConstants: Această interfață oferă constante pentru diverse proprietăți legate de ferestre, cum ar fi operația de închidere implicită.

java.awt.BorderLayout: Această clasă reprezintă un manager de aspect care aranja componentele într-un aspect de bordură, împărțind containerul în cinci zone: nord, sud, est, vest și centru.

java.awt.Color: Această clasă reprezintă o culoare în spațiul de culori RGB (roșu, verde, albastru).

java.util.Scanner: Această clasă oferă metode pentru citirea datelor din diverse surse de intrare, cum ar fi fișierele text sau consola.

- **Performante**

In functie de marimea imaginii, timpii de executie pentru fiecare procedura creste direct proportional.

Citirea imaginii porneste de la aproximativ 0.053 secunde, timpul de prelucrare este in jur de 0.23 secunde, iar timpul de scriere este aproximativ 0.028 secunde.

➢ Concluzii

Acest proiect m-a ajutat sa imi consolidez cunostiintele despre Java si OOP, si sa prelucrez o imagine, sa ii aplic un filtru de sharpen, prin aplicarea unei matrici de convolutie.

➢ Bibliografie

- https://www.songho.ca/dsp/convolution/convolution2d_example.html
- https://www.songho.ca/dsp/convolution/convolution.html
- https://setosa.io/ev/image-kernels/
- https://www.w3schools.com/java/
- https://www.geeksforgeeks.org/image-processing-in-java-read-and-write/
- https://alvinalexander.com/blog/post/java/getting-rgb-values-for-each-pixel-in-image-using-java-bufferedi/