Image Sharpening

(convolution mask)

Colcer Alexia Ioana

Grupa 334AA

Introducere

Image Sharpening este un proces ce consta in cresterea contrastului dintre zonele luminoase si cele intunecate ale unei imagini pentru a accentua caracteristicile. Cu alte cuvinte, se mareste valoarea contrastului in anumite zone.

Descrierea aplicatiei cerute

Aplicatia, realizata in limbajul de programare Java, trebuie sa permita, mai intai, incarcarea unei imagini (adresa acesteia). Dupa aceea, aplicatia va procesa imagina, va efectua anumite operatii pentru fiecare bit al acesteia, urmand ca rezultatul sa fie o noua imagine, ce are caracteristici mult mai detaliate.

Parte teoretica

Pe langa teoria legata de ceea ce inseamna de fapt procesul de Sharpening, este necesara si definirea subclasei BufferedImage. Aceasta se gaseste in pachetul java.awt.image. Aceasta permite stocarea unei imagini. Este creeata o matrice cu dimensiuni ce variaza in functie de pixelii imaginii. Si in fiecare componenta sunt stocate valori pentru cele 3 culori RGB.

Convolution Mask

O gamă largă de tehnologii de procesare a imaginii utilizează operații multipixel cu măști de tip kernel convolution, în care fiecare pixel de ieșire este modificat de contribuțiile unui număr de pixeli de intrare adiacenți. Aceste tipuri de operațiuni sunt în mod obișnuit denumite convoluție sau convoluție spațială. In cazul efectului de sharpen se foloseste urmatoarea masca de 3x3 elemente:

```
kernelelem = { 0.0f, -1.0f, 0.0f,
              -1.0f,  5.f, -1.0f,
               0.0f, -1.0f,  0.0f};
```

Descrierea implementarii

Pentru a crea aceasta aplicatie am implementat mai intai o clasa abstracta cu numele ReadFile. Aceasta clasa nu poate fi instantiata. Ea contine doar metoda abstracta Read. Apoi am implementat clasa ReadImage, ce mosteneste clasa abstracta ReadFile. Aceasta contine metoda Read care se ocupa de citirea unei imagini sub forma de Buffered Image. Pentru a citi o imagine este nevoie sa se introduca adresa acesteia. In mod asemanator am implementat si clasele WriteFile(clasa abstracta) si WriteImage. Acestea contin metoda Write care se ocupa de scrierea imaginii la o anumita adresa.

Apoi am realizat interfata BaseImage. Aceasta contine o metoda abstracta(fara implementare) numita GetBuffer. Aceasta metoda este suprascrisa in clasa Image, unde ii este definita functionalitatea. Clasa Image are ca atribute o imagine, latimea si inaltimea acesteia dar si un vector tridimensional pentru stocarea culorilor. Aceasta este clasa pe care o mosteneste clasa ImageSharpen.

Aceasta clasa contine algoritmul pentru realizarea procesului de Image Sharpening. Fiecare pixel de culoare este recalculat in functie de valoarea pixelului curent si de valoarea pixelului de pe linia anterioara si coloana anterioara pentru a obtine o accentuare a imaginii.

In acest cod am folosit threaduri pentru a imbunatati eficienta procesarii imaginilor. Un thread (ReadImage) este utilizat pentru a citi o imagine dintr-un fisier și a o pune intr-o coada blocanta. Apoi, un alt thread preia si proceseaza aceasta imagine.

Practic, ReadImage citește imaginea si o plaseaza în readBuffer. ImageProcessor preia imaginea din readBuffer, o proceseaza si o plaseaza în processedBuffer. In final, clasa Main preia imaginea procesata din processedBuffer si continua cu scrierea sau alte operatiuni necesare. Aceasta abordare permite ca citirea si procesarea imaginii sa se desfasoare in paralel, reducand timpul de asteptare si imbunatatind performanta generala a programului.

Am adaugat si un notifyAll() pentru a notifica ImageProcessor cand o imagine este disponibila si un wait() pentru a aștepta pana cand o imagine este disponibila în buffer.

Rularea programului

Informatiile legate de adresa unei imagini si adresa rezultatului pot fi introduse de catre utilizator atat ca input de la tastatura la rularea programului cat si ca parametrii in linia de comanda.

Pentru rularea programului din linia de comanda, mai intai trebuie navigat pana in folderul ce contine programul. La mine: C:-> Java-> Workspace -> ImageSharpening -> src. Aici se ruleaza comanda java PackTest.Main.

• Daca nu se introduce niciun argument, atunci ni se va indica pe rand sa introducem adresele pentru sursa si pentru destinatie.

• Daca se introduce un singur argument – sursa, atunci, dupa citirea imaginii si prelucrarea acesteia ni se va indica sa introducem adresa destinatiei.

• Daca se introduc ambele argumente, programul va rula pana la final fara sa fie nevoie de alte interventii.

Rezultate

In urma prelucrarii unei imagini oarecare am obtinut urmatoarele rezultate:

Performantele obtinute sunt urmatoarele:

```
Dati adresa fisierului sursa: C:\Java\workspace\ImageSharpening\3.bmp
Timpul necesar procesarii imaginii este: 0.02775 secunde
Dati adresa fisierului destinatie: C:\Java\workspace\ImageSharpening\test.bmp
Writing
Writing complete
Timpul necesar scrierii imaginii este: 0.0210255 secunde
```

Dupa cum se poate vedea, imaginea a fost procesata in 0.02775 secunde. Scrierea acesteia, foarte rapida, a fost efectuata in doar 0.0210255 secunde.

In concluzie, procesarea si prelucrarea unei imagini este destul de rapida si cu rezultate impresionante.

Bibliografie

- http://www.informit.com/articles/article.aspx?p=1013851&seqNum=5
- https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html
- https://blog.udemy.com/java-interface-example