

Proiect Aplicatii Web cu suport Java: Image Sharpening (Convolution Mask)

Neleptcu Daniel-Andrei 332AB

12.01.2024

Cuprins

Introducere	2
Descriere functionalitate aplicatie	2
Descriere clase	2
Respectare cerinte	5
Rezultate	7
Bibliografie	8

Introducere

Tema aleasa presupune modificarea unei imagini BMP cu scopul aplicarii filtrului de sharpness. Aplicarea filtrului se face prin aplicarea unui Kernel de convolutie ale carui elemente se inmultesc cu fiecare element corespunzator din matricea originala de pixeli. Fiecare rezultat al acestor inmultiri se aduna la o suma care va reprezenta pixelul final. In general Kernel-urile defera prin dimensiune, dar acestea sunt de obicei patratice cu numar impar de linii/coloane, suma valorilor din Kernel rezultand 1. Convolutia cu Kernel poate duce la mai multe efecte in functie de valorile alese pentru kernel. Am ales sa folosesc un kernel de tipul:

$$\text{Kernel} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Descriere functionalitate aplicatie

Un fisier BMP este o imagine salvata in formatul de imagine raster Bitmap (BMP). Contine date de imagine necomprimate, acceptand imagini monocrome si color la adancimi de biti de culoare variabile, impreuna cu metadate de imagine. Fisierele BMP sunt utilizate in mod obisnuit pentru stocarea fotografiilor digitale 2D. Aplicatia foloseste ca sursa o imagine BMP 24bit. Pentru a rula aplicatia se apeleaza in consola *"java -jar numeaplicatie.jar inputPath outputPath"*, unde *inputPath* si *outputPath* sunt locatiile fisierului sursa, respectiv unde dorim sa introducem fisierul final. In lipsa celui de-al doilea parametru fisierul final se va salva in directorul aplicatiei *jar* cu numele *"sharpenOutput.bmp"*.

```
C:\Users\zuch3e\Desktop>java -jar projectJava.jar hope.bmp output.bmp
Am intrat in ImageReader
```

Exemplu rulare aplicatie

Descriere clase

Proiectul este structurat in doua pachete: *packTest* si *packWork*. Pachetul *packTest* contine clasa de test a aplicatiei *MainClass* de unde este comanda aplicatia. Clasa *packWork* contine clasele ce implementeaza algoritmii si logica necesara in vederea producerii rezultatului dorit.

MainClass

In aceasta clasa se initializeaza toti parametrii necesari pornirii aplicatiei, se citesc din *args* valorile path-urilor de input si output, se creeaza legaturile intre pipe-uri (pentru legatura Consumer-WriterResult), obiectul buffer comun claselor Producer-Consumer, respectiv se pornesc threadurile corespunzatoare ce vor executa cerinta in continuare.

ImageReader(Producer)

In clasa *ImageReader* se face legatura cu clasa *ImageWriter* prin intermediul buffer-ului comun in care se va plasa informatie. Aceasta clasa extinde clasa *startProcess*. Obiectul instantiat din aceasta clasa isi incepe executia in supra-scrierea metodei *run* a clasei *Thread* in care se calculeaza timpul de executie a metodei *processImage()*, practic timpul petrecut in clasa *ImageReader*. In aceasta metoda se declara un obiect de tipul *ImageIO* din *inputPath-ul* specificat, se extrag dimensiunile si pixelii. Pixelii sunt plasati intr-un vector ce este trimis prin buffer catre clasa *ImageSharpen*.

Buffer

Aceasta metoda reprezinta spatiul de stocare comun dintre clasele *ImageReader* si *ImageSharpen*. Clasa are metode sincronizate de introducere si citire in, respectiv din buffer, ce asteapta intai ca o valoare sau un vector sa fie prima data introdus pana sa incerce sa il citeasca.

ImageSharpen(Consumer)

In aceasta clasa se face legatura cu clasa *ImageReader* prin intermediul lui *buffer*, respectiv cu clasa *WriterResult* prin intermediul Pipe-urilor. Asemanator clasei *ImageReader*, clasa suprascrie metoda *run* in care se calculeaza timpul de executie al metodei *processImage*. In aceasta metoda se asteapta un timp prestabilit conform cerintei, se citesc dimensiunile si pixelii din *buffer*. Dupa citirea informatiilor necesare incepe procesul de convolutie al matricei de pixeli cu Kernelul mentionat in **Introducere**. Am ales sa nu folosesc metoda *zero-padding* in tratarea muchiilor, ci pastrarea acestora ca muchiile din matricea de pixeli initiala, deoarece prin aceasta metoda nu se supralumineaza pixelii ca in cazul metodei *zero-padding*. Inainte de inceperea convolutiei se transmit in *pipe* dimensiunile matricei. Dupa aplicarea efectului se transmite pixel cu pixel, valoarea modificata in urma filtrului catre *WriterResult* prin intermediul *pipe-ului* comun dintre acestea.

InterfataTimp

In aceasta interfata se implementeaza functionalitatea de baza pentru masurarea timpului, metodele fara implementare, ce vor fi implementate de clasa Timp.

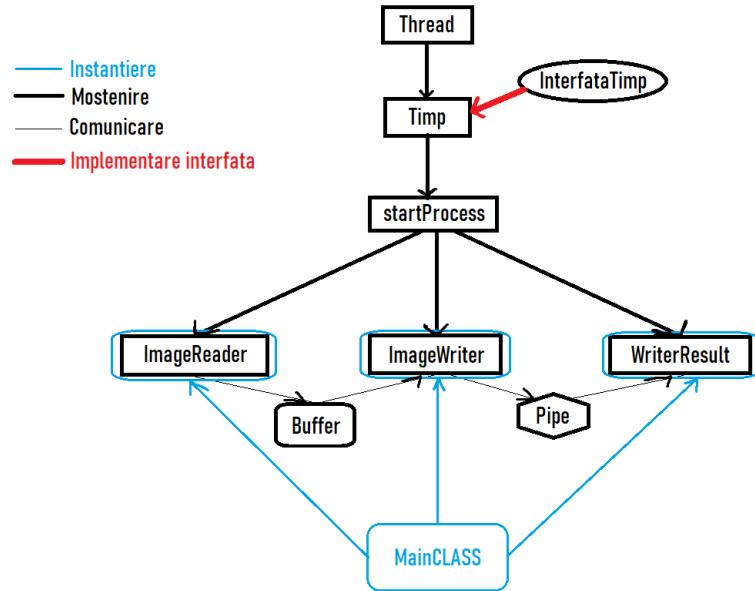
Timp

In aceasta clasa sunt implementate metode pentru a putea seta un punct de pornire al masurarii timpului, $Tstart$, un punct de oprire al masurarii timpului, $Tstop$, respectiv o metoda pentru a putea calcula diferenta dintre aceste doua metode si a o converti la o valoare reprezentata in secunde, $getTime$.

StartProcess

Aceasta este o clasa abstracta ce descrie o metoda numita $processImage()$ ce va fi implementata in fiecare clasa care este mostenita de ea (cele 3 clase ce implementeaza prelucrarea imaginii).

Relatii Clase



Relatii Clase

Respectare Cerinte

1. Am folosit doar fisiere BMP 24bit.
2. Am implementat manual aplicarea convolutiei pentru sharpness.
3. Am folosit incapsulare prin setarea atributelor cu parametrul private. Mostenirea poate fi observata in diagrama din pagina anterioara. Polimorfismul este prezent prin metodele implementate in clasa *Timp*, respectiv prin existenta metodei *put()* din clasa *buffer*. Abstractizarea a fost realizata prin implementarea clasei abstracte *startProcess*, respectiv prin implementarea metodei *processImage* in cele 3 clase ce o mostenesc.
4. Am respectat "Coding Standards" prin: inceperea numelui claselor cu litera mare, inceperea numelui metodelor cu litera mica, cuvantul public este inaintea cuvantului static, numirea parametrului din main *args*, fiecare cuvant dintr-un identifier incepe cu litera mare mai putin primul in cazul metodelor.
5. Am folosit fisiere pentru input si output. Pe langa acestea am pastrat o ordine clara a utilizarii parantezelor, respectiv spatierii dintre variabile, operanzi si operatori, atribuirii si indentari.
6. Am folosit args pentru transmiterea parametrilor din consola la rularea fisierului *jar*.
7. Aplicatia este modulara cu cel putin 3 niveluri de mostenire (se pot observa in diagrama din pagina anterioara).
8. Am folosit *varargs* la metoda *put()* din *buffer*. Aceasta putea fi folosit si pentru cazul in care se transmite vectorul de pixeli.
9. Am folosit constructori in cele 3 clase instantiatе de *MainClass*
10. Am folosit un bloc de initializare in clasa *Buffer* si un bloc de initializare static pentru initializa numarul de instante cu 0 in clasa *Timp*.
11. Include interface prin interfata *InterfataTimp* ce este implementata de clasa *Timp*.
12. Include Clasa Abstracta prin clasa *StartProcess* care este extinsa de cele 3 clase instantiatе de *MainClass*
13. Exceptiile sunt tratate folosind blocuri try-catch.
14. Aplicatia este structurata in 2 pachete: *packTest* cu clasa de testare *MainClass* si pachetul *packWork* cu restul claselor.

15. Aplicatia contine Producer-Consumer(*ImageReader* - *ImageSharpen* cu thread-uri. Transmiterea se face prin obiectul buffer, linie cu linie, nu se poate astepta o secunda dupa fiecare segment transmis (linie), deoarece o imagine are foarte multe linii si s-ar astepta mult. Am folosit elemente de sincronizare in cadrul clasei *Buffer* si am inceput procesarea dupa transmiterea informatiei integrale.
16. Am folosit comunicatie prin *Pipes* pentru clasele *ImageSharpen*, *WriterResult*, transmit informatia pixel cu pixel. Deoarece este nerealist sa transmit 4 segmente de informatie prin 4 transmiteri in cadrul unei imagini cu dimensiuni realistice, am ales sa transmit informatia pixel cu pixel si sa afisez un mesaj de fiecare data cand se ajunge la o impartire fictiva a imaginii in 4 segmente, la nivel de linii.
17. Se respecta etapele de executie ale aplicatiei conform cerintei.

Rezultate

Unicorn:



Inainte de Sharpen

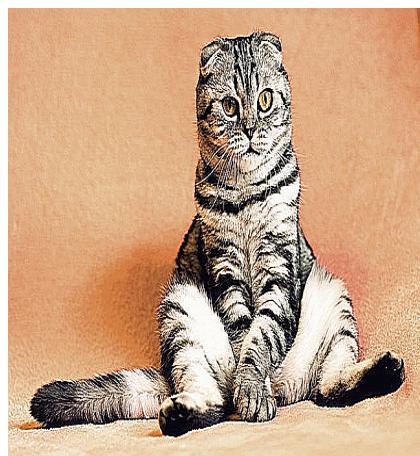


Dupa Sharpen

Pisica:



Inainte de Sharpen



Dupa Sharpen

Bibliografie

Click - Wikipedia Convolutie cu kernel

Click - Curs AWJ 2023-2024 Moodle

Click - Poza Unicorn

Click - Poza Pisica