

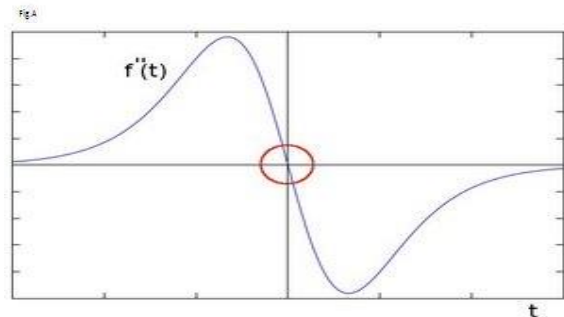
Procesarea unei imagini

- operatorul Laplacian (Positive/Negative) –

Introducere:

Operatorul Laplacian este un operator derivat folosit pentru a afla marginile unei imagini. Diferența majoră între operatorul Laplacian și alți operatori (Sobel, Prewitt) este că toate acestea sunt derivate de ordinul 1, iar Laplacianul este derivată de ordinul al doilea. Operatorul Laplacian poate fi pozitiv sau negativ, depinzând de valorile din kernel-ul matricei. Astfel, cel pozitiv are pe poziția din centru o valoare pozitivă, iar cel negativ are pe poziția centrală o valoare negativă. Restul elementelor vor fi -1 pentru Laplacianul pozitiv și 1 pentru Laplacianul negativ. Dacă colturile nu se iau în considerare, atunci în colțurile din kernel se pun zerouri.

$$\Delta^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$



Rolul aplicației:

Programul este scris în limbajul de programare Java, este rulat în mediul de dezvoltare Eclipse Mars 2 și efectuează operații cu fișiere și cu consola. Aplicația are ca sarcini preluarea de imagini ale căror locații sunt indicate în cadrul unui batch file (script ce utilizează comenzi din command prompt), trecerea acestora printr-un filtru Laplacian și, scrierea imaginii rezultate într-o locație indicată tot în batch file-ul de execuție. La fiecare execuție a programului pe fiecare imagine, se vor afișa timpii de execuție la fiecare stare prin care se trece.

0	-1	0
-1	4	-1
0	-1	0

The laplacian operator

-1	-1	-1
-1	8	-1
-1	-1	-1

The laplacian operator
(include diagonals)

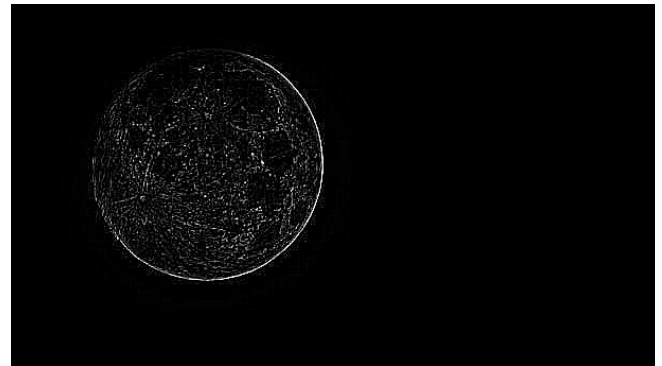
Modul de functionare:

Operatorul Laplacian amplifica discontinuitatile de nivel cu gri, astfel fiind redusa proeminenta zonelor în care nivelurile nu variaza semnificativ. Astfel se scot în evidenta marginile obiectelor din imagine.

Before



After



Descrierea structurala a aplicatiei:

Aplicatia este impartita in 2 pachete: unul ce continue programul principal (Main.java si Main class), iar celalalt cuprinde toate clasele ce implementeaza logica algoritmului, de la citire si scriere, pana la aplicarea operatorului Laplacian, si retinerea unor timpi de executie pentru observarea performantelor aplicatiei (Pixel, TablouDePixeli, Photo, Cadru, DetaliiExecutie, Filtru, OperatorLaplace).

Obiectele de baza utilizate in aplicatie sunt filtru si imagine. Clasa folosita pentru a crea obiectul imagine este clasa Pixel. Se contruieste prin agregare clasa abstracta TablouDePixeli, ce implementeaza interfaaa Cadru. Aceasta interfața contine functiile de obtinere a dimensiunilor si de redimensionare. Clasa Photo mosteneste clasa TablouDePixeli (deoarece o imagine este în teorie o matrice de pixeli). Aceasta implementeaza caracteristicile specifice unei imagini, cum ar fi metodele de scriere si citire de la locatiile specificate si timpul de încarcare. Obiectul filtru se obtine prin instantierea clasei abstracte Filtru. Aceasta clasa defineste funcția de aplicare a filtrului. De asemenea, implementează interfata DetaliiExecutie, care la randul ei, implementeaza calculul timpului de execuție. Clasa OperatorLaplace extinde clasa abstractă Filtru si are implementat algoritmul de transformare a pixelilor folosind filtrul Laplace.

In Main, se salveaza string-urile date ca parametri la executia fisierului batch în 2 variabile. Se instantiaza obiectul imagine folosindu-se constructorul care ia ca parametri path-ul imaginii respective. Se instantiaza un obiect de tip OperatorLaplace care ia ca parametru un obiect de tip Photo și returneaza un alt obiect tot de tip Photo. Mai apoi, dupa aplicarea filtrului, se folosește metoda de scriere a obiectului Photo care ia ca parametru un string continand path-ul în care noua imagine va fi scrisa si salvata.

Descrierea claselor utilizate:

Pixel: **clasa** ce implementeaza constructor cu parametri, gettere și settere;

TablouDePixeli: **clasa abstracta** ce implementeaza constructor cu parametri, afisare, obtinere handler pentru matricea de pixeli, resize, setare a unui pixel cu pozitia respectiva data ca parametru, respectiv obtinerea unui pixel, getter pentru dimensiunile matricei;

Photo: **clasa** ce implementeaza toate functiile mostenite de la TablouDePixeli, adaugand un constructor avand ca parametru locatia de incarcare, functie de scriere intr-un anumit path, resize, citirea unei noi imagini dintr-un path specificat ca parametru, obtinerea path-ului imaginii, obtinerea timpilor de extragere a pixelilor și de incarcare si salvare a imaginii;

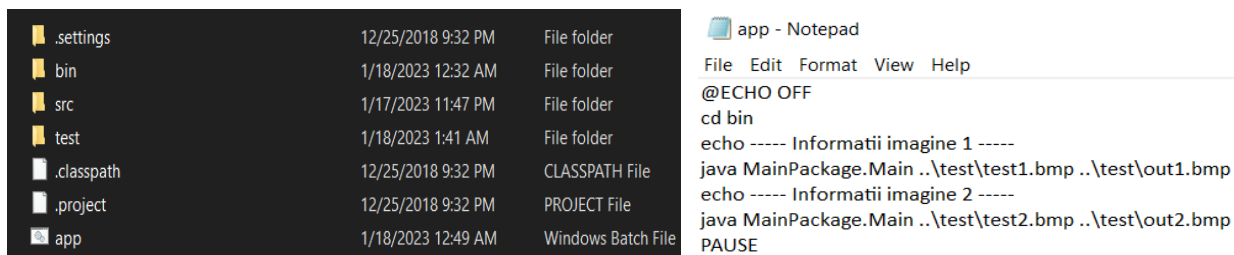
Cadru: **interfata** ce defineste functiile de obtinere a dimensiunilor si resize;

DetaliiExecutie: **interfata** ce defineste functia de obtinere a timpului de executie al procesarii;

Filtru: **clasa abstracta** ce defineste functia de aplicare a unui anumit filtru asupra unei imagini și implementarea functiei din interfata DetaliiExecutie;

OperatorLaplace: **clasa** ce implementeaza algoritmul pentru aplicarea filtrului Laplace asupra unei imagini.

Executie aplicatie:



The image shows a file explorer window on the left and a Notepad window on the right. The file explorer displays a directory structure with files and folders including .settings, bin, src, test, .classpath, .project, and app. The Notepad window, titled 'app - Notepad', shows the following text:

```
File Edit Format View Help
@ECHO OFF
cd bin
echo ----- Informatii imagine 1 -----
java MainPackage.Main ..\test\test1.bmp ..\test\out1.bmp
echo ----- Informatii imagine 2 -----
java MainPackage.Main ..\test\test2.bmp ..\test\out2.bmp
PAUSE
```

Run app.bat

Acest fisier functioneaza doar pe sistemele de operare Windows (de la versiunea XP pana la cea actuala). Pentru adaptare pe Linux sau MacOS, trebuie utilizate script-uri sau pipeline-uri scrise in bash sau in zsh.

Concluzie:

Timpul de executie al aplicatiei de la prima stare catre ultima (citire, executie, scriere) difera in functie de dimensiunile imaginii si spatiul de stocare. Cu cat avem o imagine mai mare/complexa, cu atat timpii de executie vor incepe sa creasca, fata de o imagine simpla ce ocupa putin spatiu.

Programarea folosind multithreading ar fi fost importanta in terminarea proiectului, dar nu sunt notiuni pe care am reusit sa le stapanesc bine, si incercarea implementarii tuturor cerintelor intr-un algoritm ca acesta a fost din pacate o provocare pe care nu am putut-o duce la capat. Totusi aplicatia functioneaza si fara multithreading, iar folosirea acestuia la aplicatii ce lucreaza cu date mici (cum ar fi imaginile) nu este o necesitate, nefiind posibila o diferenta vizibila la aplicare.

Resurse:

Java DIP – Laplacian Operator:

https://www.tutorialspoint.com/java_dip/applying_laplacian_operator.htm

Stackoverflow: <https://stackoverflow.com/>

Jpanel: <https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html>

ActionListener: <https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html>