

Proiec Java

Logarithmic Gray Level Transform

Nume: Stoica Andreea
Grupa: 331AA

Introducere

Proiectul prezinta o modalitate de implementare al algoritmului de procesare de imagini **Logarithmic Gray Level Transform**, prin metode de programare orientatata pe obiect.

Descriere aplicatie

Aplicatia citeste de la tastatura numele fisierului pe care dorim sa il procesam, urmand sa construiasca calea prin care vom acesa **fisierul BMP**.

Pentru a putea prelucra la nivel de **pixel**, urmeaza citirea propriu zisa a imaginii. Procesarea imaginii va aplica algoritm specific transformarii logaritmice si ulterior va salva noua imagine obtinuta intr-un fisier de tip BMP.

De asemenea, proiectul tine cont de **timpii de executie** pentru fiecare etapa in parte. La sfarsit, vom calcula si timpul de executie pentru etapa totala de citire.

Consola aplicatiei:

Introduceti numele fisierului de intrare: **floare**

Introduceti numele fisierului de iesire: **out**

Etapa de citire informatii de identificare fisier dureaza: 4825

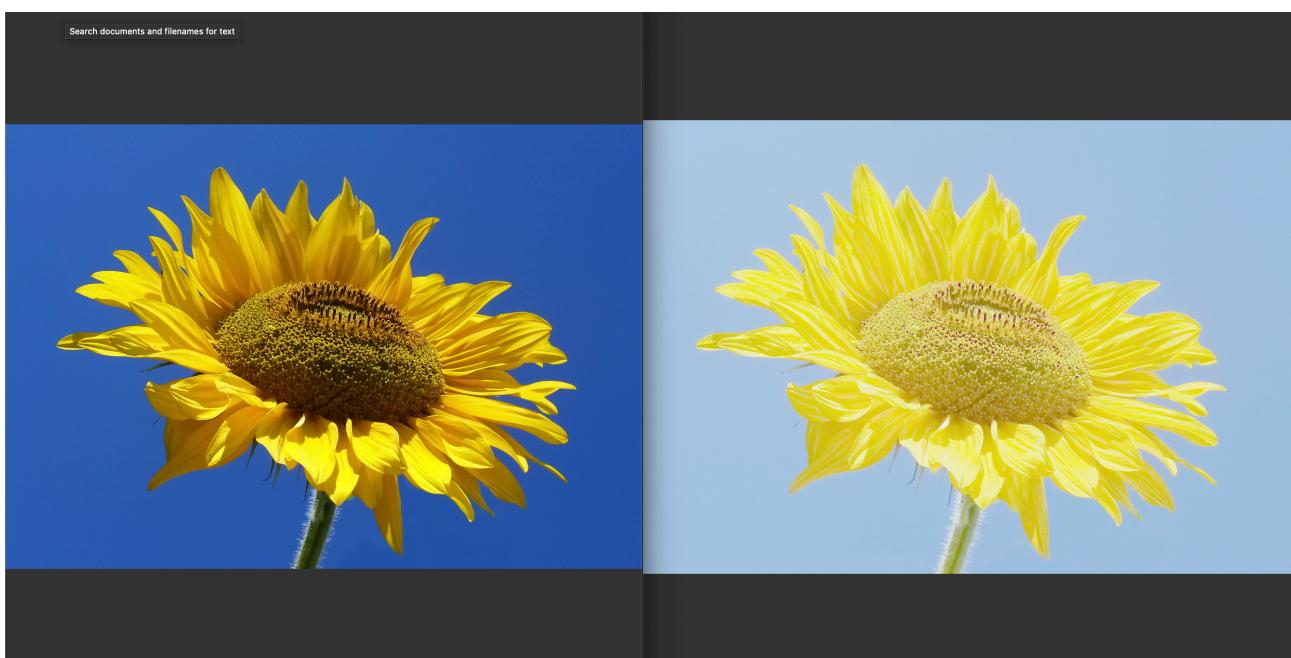
Etapa de citire a imaginii dureaza: 168

Procesarea imaginii dureaza: 740

Etapa de scriere in fisier dureaza: 74

Timpul pentru intreaga etapa de citire este: 4993

Aplicarea algoritmului



Imagine inainte de procesare

Imagine dupa procesare

Partea teoretica

Pentru a intelege mai bine utilitatea algoritmului de procesare, vom explica in linii mari principiul acestuia.

Logarithmic Gray Level Transform **modifica intensitatea culorii** dintr-o imagine prin **procesare la nivel de pixeli**. Se utilizeaza atunci cand este nevoie de schimbarea contrastului, pentru a ilumina imaginea. Acest procedeu are variatii aplicatii: imbunatatirea imaginilor medicale, a imaginilor capturate de catre sateliti, a imaginilor utilizate de senzori etc.

Transformata poate fi definita prin urmatoarea formula:

$$s = c \log(r + 1).$$

s este valoarea pixelului obtinut

c este o constanta ce determina gradul de iluminare

r este pixelul imaginii pe care dorim sa o prelucram

Se adauga valoarea 1 pentru a nu obtine un pixel infinit in cazul in care avem $\log(0)$. Astfel, valoarea minima care se va obtine va porni de la 1.

In timpul transformarii pixelii negri vor fi expandati, iar pixelii albi vor fi compresati, ceea ce va duce la o reglare a intensitatii luminii din fotografie si o imbunatatire a zonelor inchise.

Implementarea la nivel de cod va fi explicata in cadrul modulului ProcesareImagine.

Descriere structurala

Identificam urmatoarele etape esentiale in executia proiectului:

1. **CITIRE INFORMATII IDENTIFICARE FISIER SURSA SI FISIER DESTINATIE**
2. **CITIRE DIN FISIER**
3. **PROCESARE IMAGINE**
4. **SCRIERE IN FISIER**
5. **INREGISTRARE SI AFISARE TEMP EXECUTIE**
6. **CALCULARE TEMP ETAPA DE CITIRE COMPLETA**

Descriere functionala

1. Etapa va citi numele fisierelor sursa si destinatie de la tastatura. Aceasta etapa contine urmatoarele module: **Interogare**, **Scanare**, **Fisiere**.
2. Aceasta etapa va furniza ca parametru numele fisierului de intrare, urmand sa il citeasca in mod corespunzator. Module: **CitireFisier**
3. Aceasta etapa se ocupa de procesarea imaginii prin algoritmul dat. Module: ProcesareImagine
4. Aceasta etapa se ocupa de crearea unui fisier cu numele destinatiei si transferarea continutului imaginii procesate. Module: **ScriereFisier**, **Operatii**
5. Etapa este implementata pentru fiecare portiune de cod din modulul main, portiune corespunzatoare fiecarei etape mentionate mai sus. Module: **main**
6. Etapa va calcula suma primelor 2 etape de citire prin intermediul unui constructor ce poate avea mai multe variabile. Module: **SumaEtape**

Modulele vor fi explicate in detaliu, in continuare.

Descriere module

Fisiere - interfata ce contine metodele specifice unei **citiri** de String

Scanare - acest modul implementeaza **interfata** si deci rescrie metodele din Fisiere pentru a putea citi un String de la consola si pentru a putea returna un String; aceasta clasa implementeaza si conceptul de incapsulare prin **seter si geter** si protected String nume

Interogare - este modulul care se ocupa de afisarea in consola a intrebarilor legate de fisiere, dar si locul unde se obtine numele fisierelor propriu zise; aceasta clasa **mosteneste** clasa Scanare si cel mai important variabila nume; de asemenea, indeplineste principiul **polimorfismului** pentru ca rescrie metoda `getNum()`

main - aici vom reuni celelalte module pentru a forma etapele de executie

CitireFisier - se ocupa de citirea imaginii in format BMP; foloseste `java.awt.image.BufferedImage`; si implicit obiectul de tip **BufferedImage** ce stie sa lucreze cu pixeli din imagine

ProcesareImagine - separa pixelii in culorile principale componente: alb, rosu, verde si albastru; aplica transformata logaritmica la nivel de bit si construieste pixelul din cele 4 componente; la sfarsit, in imagine vom avea o matrice cu pixelii noi construiți

ScriereFisier - foloseste aceeași clasa `BufferedImage` pentru a scrie in fisierul destinatie pixelii procesati

SumaEtape - foloseste parametru variabil (**varargs**) pentru a calcula suma primelor 2 etape.

Operatii - este o **clasa abstracta** pentru exemplificare lucru cu clase abstracte; in aceasta se defineste o **metoda abstracta** ce este rescrisa in ProcesareImagine

Evaluare performante

Am evaluat performantele aplicatiei prin timpul de executie.

Etapa de citire informatiei de identificare fisier dureaza: 4825

Etapa de citire a imaginii dureaza: 168

Procesarea imaginii dureaza: 740

Etapa de scriere in fisier dureaza: 74

Se observa o discrepanță mare între timpul de citire al informațiilor de identificare și celelalte etape. Acest lucru rezultă din ierarhizarea excesivă Fisiere, Scanare, Interogare pe care am implementat-o pentru a exemplifica concepțele de POO - în special multimodularitatea cu cele 3 niveluri de moștenire.

Totuși, aplicația indeplinește în mod corect procesarea imaginii initiale.

Concluzii

Pentru a eficientiza timpul de lucru se pot reduce nivelurile de moștenire.

Aplicația a fost gândită pentru a invata concepțele de POO, timpul de execuție nefiind o prioritate.

Bibliografie

https://www.tutorialspoint.com/dip/gray_level_transformations.htm

<http://www.cs.uregina.ca/Links/class-info/425/Lab3/>

<https://www.tutorialspoint.com/index.htm>

<https://www.w3schools.com>

<https://stackoverflow.com>

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

<https://beginnersbook.com/2013/05/java-abstract-class-method/>

<https://scientistengineer.blogspot.com/2012/07/log-transformation.html> – cod procesare imagine