FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Negative Image

Student: Buric Andrei-Alexandru

Grupa: 333AA

An universitar: 2021-2022

Negative Image

1. Introducere

Tema de procesare pe care am primit-o ca tema este “Negative Image”. Aceasta procesare consta in schimbarea culorilor unei poze care este salvata in format “bmp” in culori negative si desigur crearea unei noi imagini pe care o vom salva desigur tot in format “bmp”.

2. Descrierea modulelor

Interfata numita **Interface** continue doar o metoda pe care o vom implementa intr-o clasa ulterioara.

Clasa abstract **DataReader** implementeaza interfata si realizeaza metoda pentru citirea de la tastatura a locatiei unde se afla fisierul, numele fisierului care trebuie prelucrat si numele fisierului final, aceste date fiind necesare in prelucrearea imaginii. Aceasta clasa ofera si o implementare a metodei abstracte din interfata. Inainte de fiecare citire desigur ca trimitem un mesaj astfel incat utilizatorul sa stie ce are de introdus.

Clasa **ThreadsMaster** mosteneste clasa **DataReader** si are ca scop controlul cu privier la thread-urile Producer/Consumer. Avand in vedere toate functionalitatile, amintim urmatoarele: crearea thread-urilor, lansarea in executie a acestora si asteptarea finalizarii executiei lor. In aceasta clasa regasim si metoda varargs(cu numar de argumente variabil).

Clasa **Start** mosteneste clasa **ThreadsMaster** si in aceasta gasim metoda main care este punctul de start al aplicatiei. Clasa incepe cu o gettere si settere care o sa ne ajuta in aflarea dimensiunilor imaginii. In prima faza verificam daca la rularea programului am primit argumente astfel incat informatille de care avem nevoie: locatia imaginii, numele imaginii si numele noi imagini le putem extrage direct din argumente. Daca nu putem sa le extragem din argumente, afisam un mesaj in care specificam ca nu avem destule argumente si apelam **ReadData** functie care o sa-mi citeasca de la tastatura informatiile de care am nevoie pentru a executa programul: locatia fisierului, numele pozei si numele pozei rezultate.

Dupa ce toate am citit toate informatiile le afisez pentru a vedea si utilizatorul ce date a introdus. Cu aceste date, fie din terminal fie date de la tastatura, citim imaginea, setam dimensiunile fotografiei si suntem gata sa apelam metoda pentru pornirea thread-urilor. Desigur, dupa apelarea acestei metode, mai executam cateva teste, cum ar fi testarea metodei abstraca din interfata, testarea metodei abstraca din clasa abstraca si testarea varargs.

Clasa **Producer** extinde clasa **Start** si o implementeaza pe **Runnable**. Aceasta clasa transforma imaginea intr-o matrice de pixeli si fiecare pixel, respective fiecare element din matrice continue cele 3 componente R – red, G – green, B-blue). Dupa acest lucru, aceasta matrice este impartita in 4 sferturi. Cu ajutorul unei structure repetitive creeaza cate un sfert care este introdus in buffer unul cate unul. Dup ace se trimit sferturile unul cate unul se afiseaza pentru utilizator timpii executiei, precum si durata acestei operatiuni de trimitere.

Clasa **Consumer** extinde clasa **Start** si o implementeaza pe **Runnable**. Pentru inceput declaram toate variabilele de care avem nevoie pentru creearea matricii de pixeli si desigur pentru timpii pe care trebuie sa ii afisam. Cu ajutorul unei secvente repetitive extragem sfert cu sfert din

buffer si pe fiecare sfert pe care l-am extras, realicam o matrice de pixeli. Fiecare pixel avand valorile specific pentru cele 3 componente R – red, G – green si B – blue. Dupa ce am creeat matricea de pixeli din cele 4 sferturi, incep aceasta prelucrare a imaginii. Pentru fiecare pixel inversez cele 3 culori, rosu, verde si albastru. Dupa aceasta modificare le introduce din nou in matrice. Dupa afisarea timpilor de durata, trebuie ca noua poza creeata sa fie scrisa tot in fisierul de unde a fost citita, dar cu noul nume. Desigur ca dupa aceasta operatie, afisam din nou timpii de executie.

Clasa **Buffer** extinde clasa **Start** si este folosita pentru rezolvarea posibilelor probleme de tipul Deadlock care pot aparea asupra resursei comune si anume coloanal pe care o trimite producatorul spre consumator. In aceasta clasa regasim metodele de put, care pun in buffer si metodele de get care scot din buffer. Prin metodele implementate in clasa putem permite accesul la buffer numai de catre un thread la un moment dat.

3. Modalitate functionare algoritm

Conversia unei imagini in culori negative (complementare) se realizeaza verificant valoare fiecarei culori componente (R,G,B) ale fiecarui pixel din imaginea respective si calcularea opusului.

Pasii sunt urmatorii:

- Stocam imaginea pixel cu pixel intr-o matrice
- Cu metoda getRGB() calculam valoarea fiecarui pixel
- Vom creea un obiect de tip **Color** pentru a extrage valorile pentru rosu, verde si albastru din fiecare pixel folosint metodele getRed(), getGreen(), getBlue()
- Calculam noile valori scazand valorile curente din valoarea 255 (care reprezinta valoarea maxima a oricarei culori) si obtinem opusul culorii respective
- Vom creea un obiect noi de tip **Color** pentru a atrivui noile valori
- Setam noile valori RGB pentru fiecare pixel folosind setRGB()

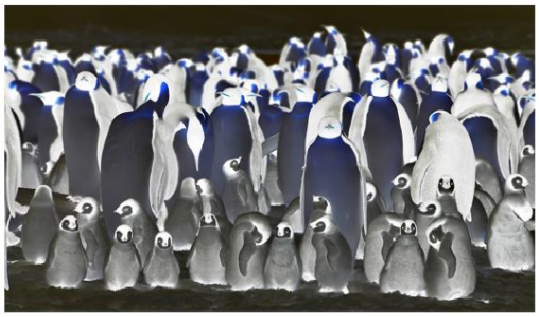
4. Rezultate obtinute

```
+------------------The parts were sent---------------------
| Start time: 1644934029926 ms
| Final time: 1644934030009 ms
| Working time = 83 ms

+-----------------The parts were received-------------------
| Start time: 1644934029777 ms
| Final time: 1644934030070 ms
| Working time = 293 ms

+-----------------The parts were modified-------------------
| Start time: 1644934030071 ms
| Final time: 1644934030173 ms
| Working time = 102 ms

+---------The parts were written in the new file-----------
| Start time: 1644934030173 ms
| Final time: 1644934030307 ms
| Working time = 134 ms
```



```
+------------------The parts were sent---------------------
| Start time: 1644934095689 ms
| Final time: 1644934095823 ms
| Working time = 134 ms

+----------------The parts were received-------------------
| Start time: 1644934095472 ms
| Final time: 1644934095927 ms
| Working time = 455 ms

+----------------The parts were modified-------------------
| Start time: 1644934095927 ms
| Final time: 1644934096034 ms
| Working time = 107 ms

+---------The parts were written in the new file-----------
| Start time: 1644934096034 ms
| Final time: 1644934096175 ms
| Working time = 141 ms
```

```
+------------------The parts were sent---------------------
| Start time: 1644934483508 ms
| Final time: 1644934483538 ms
| Working time = 30 ms

+-----------------The parts were received-------------------
| Start time: 1644934483475 ms
| Final time: 1644934483548 ms
| Working time = 73 ms

+-----------------The parts were modified-------------------
| Start time: 1644934483548 ms
| Final time: 1644934483608 ms
| Working time = 60 ms

+---------The parts were written in the new file-----------
| Start time: 1644934483608 ms
| Final time: 1644934483659 ms
| Working time = 51 ms
```

5. Concluzie

Putem obsera o diferenta la timpii de executie al fiecarei imagini, acest lucru este datorat faptului ca fiecare poza are dimensiuni total diferite. Totusi, aplicatia a reusit sa transforme fiecare imagine in forma ei negata cu success in ciuda numarului diferit de linii si de coloane.

6. Bibliografie

https://www.tutorialspoint.com/how-to-convert-a-positive-image-to-negative-tousing-opencv-librar