Converting Color Image to Gray-Scale Image
- Average method -

Aplicații web cu suport Java

Student: Drăgan Pavel

Introducere

Imaginile color sunt o parte esențială a mediului nostru vizual, dar uneori este necesar să le convertim în imagini grayscale pentru diferite scopuri. Pentru această temă am explorat implementarea conversiei imaginilor color la grayscale folosind metoda mediei (average method). Proiectul are la bază un set de clase Java care acoperă diverse aspecte ale acestui proces.

Descrierea aplicației cerute

Aplicația are rolul de a face transforma o imaginea color (RGB) la una GrayScale folosind metoda mediei. Imaginile sursă trebuie să aibă formatul 24bit BMP. De asemenea pentru procesare trebuie să fie folosiți doar algoritmi low-level, aplicația să conțină concepte POO și să respecte „Coding standards". Utilizatorul va introduce de la tastatură în lina de comandă path-ul fișierelor din care trebuie luate imaginile sursă dar și destinația imaginilor grayscale. În cazul în care nu se află imagini în fișierul menționat se va afișa un mesaj care să specifice acest lucru. Pe lângă funcționalitatea de transformare a tuturor imaginilor din fișierul menționat, utilizatorul mai are posibilitatea de a specifica ce poze anume să fie convertite și să le și afișeze instant (aici am implementat cerința cu varargs). Aplicația este și modulară, fiind împărțită în clase cu ierarhii.

Partea teoretică

Metoda medie (average method) pentru a transforma o imagine color într-una grayscale se bazează pe ideea simplă de a combina componentele de culoare ale fiecărui pixel într-o singură valoare, care reprezintă nivelul de gri al acelui pixel. Acest proces de conversie presupune că intensitățile Red (Roșu), Green (Verde) și Blue (Albastru) au greutăți egale în determinarea culorii finale a pixelului grayscale. Deci putem parcurge pixel cu pixel imaginea noastră făcând media dintre valorile celor 3 culori.

Prin parcurgerea acestui proces pentru fiecare pixel al imaginii color, obținem o imagine grayscale în care fiecare pixel reflectă nivelul mediu de gri al componentelor sale color. Este important de menționat că această metodă presupune că intensitățile Red, Green și Blue au aceeași importanță în determinarea nivelului de gri al unui pixel, ceea ce poate produce rezultate satisfăcătoare în multe cazuri, dar nu capturează în mod necesar toate subtilitățile imaginii color originale.

Descrierea structurală

Proiectul este divizat în mai multe clase:

• ImageInterface: O interfață care definește metodele de afișare și citire a imaginilor.

• Image: O clasă abstractă care implementează interfața și oferă funcționalități de bază pentru manipularea imaginilor.

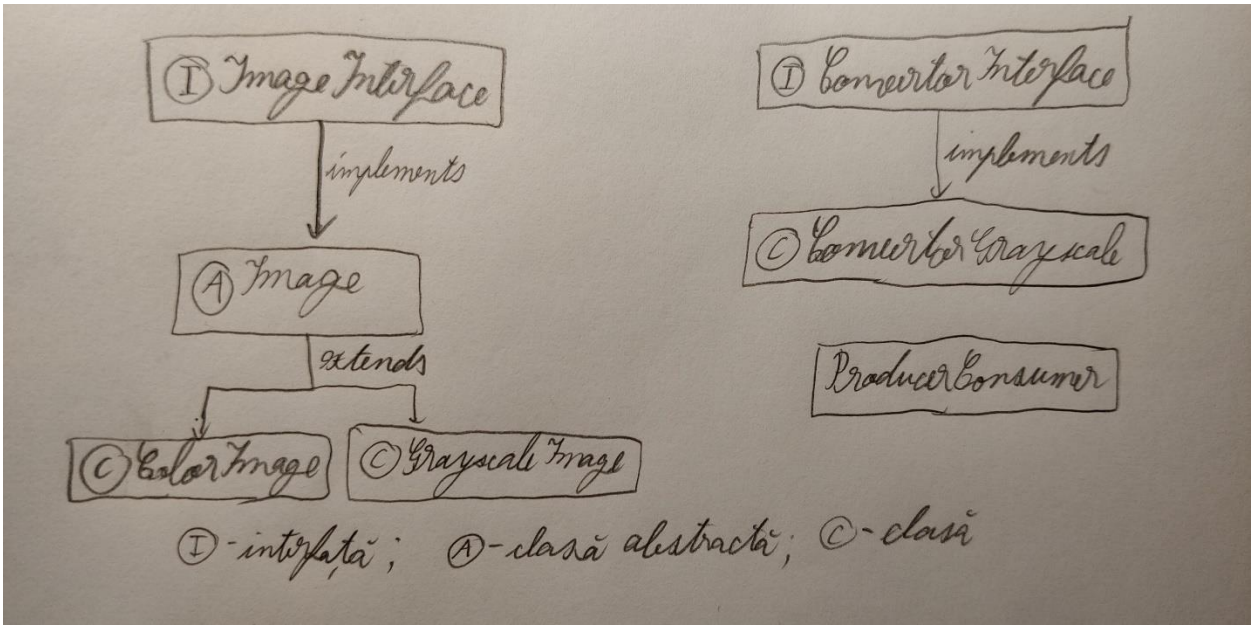
• ColorImage: O subclasă a clasei Image care reprezintă imagini color.

• GrayscaleImage: O subclasă a clasei Image care reprezintă imagini grayscale și adaugă funcționalități specifice acestora.

• ConvertorInterface: O interfață care definește metodele pentru conversia imaginilor.

• ConvertorGrayscale: O clasă care implementează ConvertorInterface și realizează conversia la grayscale folosind metoda mediei.

• ProducerConsumer: O clasă care implementează un model producător-consumator pentru gestionarea imaginilor.



Descrierea implementării

• Interfata ImageInterface:

- Metode abstracte:

afisare(): O metoda abstracta pentru afisarea imaginii.

citire(String numeFisier): O metoda abstracta pentru citirea imaginii dintr-un fisier.

• Clasa ColorImage:

- Mostenire: Mosteneste clasa Image, extinzandu-i functionalitatea pentru a manipula imagini color.

- Constructori:

Constructor implicit: Creaza o instanta a clasei fara a specifica o imagine.

Constructor cu parametru BufferedImage: Primeste o imagine color si o atribuie variabilei membru din clasa parinte.

• Clasa GrayscaleImage:

- Mostenire: Mosteneste clasa Image, extinzandu-i functionalitatea pentru a manipula imagini grayscale.

- Variabila ImagineGrayscale:

Tip: BufferedImage

Rol: Reprezinta imaginea grayscale asociata instantei de clasa.

- Constructor cu parametri: Primeste o imagine color si una grayscale, atribuindu-le variabilelor corespunzatoare.

- Metoda afisare(): Descriere: Implementeaza afisarea imaginii grayscale pe ecran, creand un obiect JFrame si adaugand imaginea in interiorul acestuia.

• Interfata ConvertorInterface:

- Metode abstracte:

convertToGrayscale(ColorImage coloredImage): O metoda abstracta pentru convertirea unei imagini color in grayscale.

convertMultipleToGrayscale(ColorImage... coloredImages): O metoda abstracta pentru convertirea mai multor imagini color in grayscale.

• Clasa ConvertorGrayscale:

- Implementare: Implementeaza interfata ConvertorInterface, oferind functionalitati pentru convertirea imaginilor color in grayscale.

- Constructor implicit: Creaza o instanta a clasei.

- Metode:

convertToGrayscale(ColorImage coloredImage): Converteste o imagine color in grayscale folosind metoda medie si returneaza rezultatul sub forma unei instante de GrayscaleImage.

convertMultipleToGrayscale(ColorImage... coloredImages): Converteste mai multe imagini color in grayscale folosind metoda medie si returneaza rezultatele sub forma unei liste de instante de GrayscaleImage.

• Clasa ProducerConsumer:

- Implementare: Ofera functionalitate pentru producerea si consumarea imaginilor intr-un mediu multi-threaded.

- Variabile:

buffer: O coada de imagini.

capacity: Capacitatea maxima a cozii.

convertor: Un obiect de tip ConvertorGrayscale pentru conversia imaginilor.

ColorImageFolder si GrayscaleImageFolder: Folderele pentru imagini color, respectiv grayscale.

- Metoda produce(): Produce imagini color dintr-un folder, le adauga in coada, si notifica consumatorul.

- Metoda consume(): Consuma imagini color din coada, le convertește la grayscale, și le salveaza intr-un folder specific.

Evaluarea performanței

În Main-ul nostru avem o serie de etape care ne ajută să convertim imaginile și să verificăm performanța aplicației dezvoltate.

Aplicația începe prin obținerea unui path pentru folderul ce conține imaginile color și un path pentru folderul în care se vor salva imaginile grayscale. Aceste informații sunt introduse de utilizator în consolă.

Se creează un obiect ProducerConsumer și se furnizează căile specificate de utilizator. Aceasta este clasa responsabilă pentru gestionarea producției și consumului de imagini între firul producător și cel consumator.

Se măsoară timpul de început al aplicației folosind System.nanoTime() pentru a determina durata totală de execuție a programului.

Se lansează două fire de execuție distincte - unul pentru producție și unul pentru consum - pentru a permite procesarea concurentă a imaginilor.

Firul de producție (producerThread) este responsabil pentru citirea imaginilor color din folderul specificat și adăugarea lor în coada de imagini pentru procesare.

Firul de consum (consumerThread) preia imaginile color din coadă, le convertește la grayscale și le salvează în folderul destinat.

Programul principal așteaptă ca ambele fire de execuție să își încheie execuția folosind metoda join(), asigurându-se că procesarea tuturor imaginilor este finalizată înainte de a afișa timpul total de execuție.

La final, timpul total de execuție al programului este afișat în consolă.

```
long startTime = System.nanoTime(); // Obtinem momentul de start al aplicatiei pentru a măsura durata totală de execuție.
```

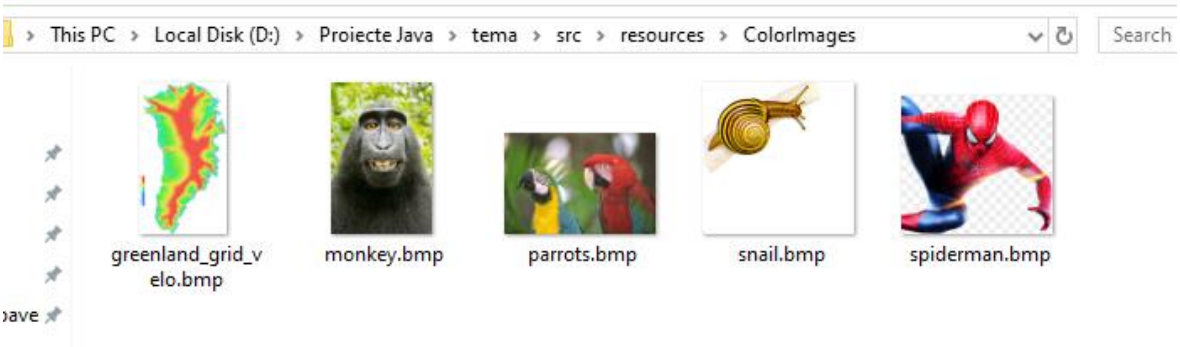
```
System.out.println("Timpul dupa citire identificare informatii fisiere: " + (System.nanoTime() - startTime));
```

```
// Afișează timpul după procesarea tuturor imaginilor.
System.out.println("Timp dupa procesarea tuturor imaginilor: " + (System.nanoTime() - startTime));
```

```
<terminated> MyMain (2) [Java Application] D:\Java2\bin\javaw.exe (17 ian. 2024, 04:14:45)
Introduceti path-ul in care se afla imaginile color:
./src/resources/ColorImages
Introduceti path-ul in care vreti sa se afle imaginile grayscale:
./src/resources/GrayscaleImages
Timpul dupa citire identificare informatii fisiere: 11531864300
Producer produced the image - 0
Producer produced the image - 1
Producer produced the image - 2
Consumer consumed
Consumer consumed
Consumer consumed
Producer produced the image - 3
Producer produced the image - 4
Consumer consumed
Consumer consumed
Timp dupa procesarea tuturor imaginilor: 20046430200
```

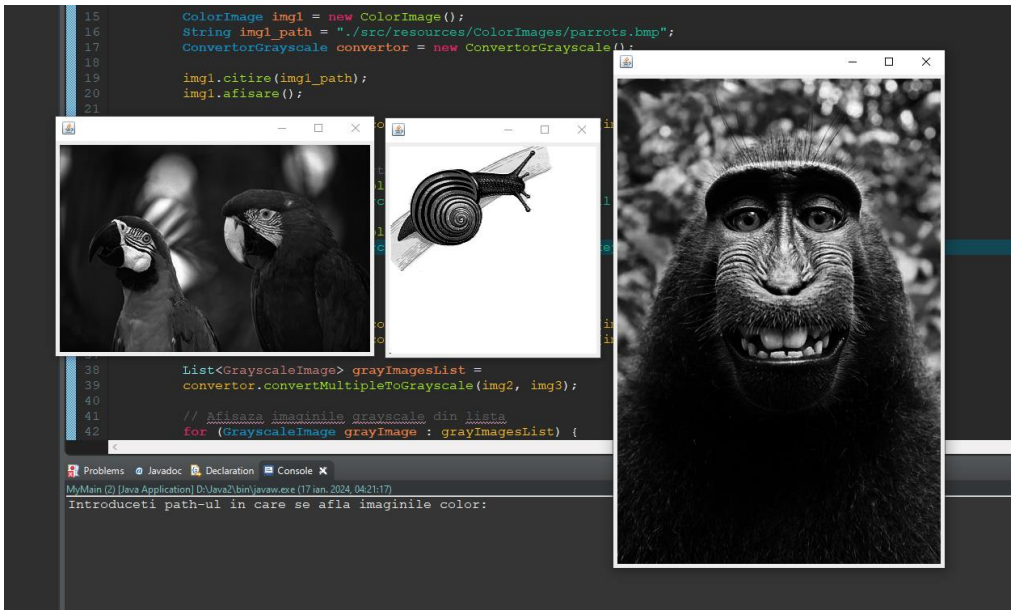
Rezultate

Folderul cu imagini color:



Folderul destinație pentru imaginile grayscale după rularea programului:

Ce se afișază dacă apelăm direct metodele care convertesc doar o imagine sau un număr de imagini dorit de utilizator:



Concluzii

În concluzie, acest program reprezintă o implementare eficientă a conversiei imaginilor color la imagini grayscale folosind un model de producător-consumator într-un mediu multi-threaded. Structurat modular și interactiv, programul demonstrează aplicarea conceptelor de programare orientată pe obiect și manipulare a imaginilor. Prin măsurarea timpului de execuție, interacțiunea cu utilizatorul și adaptabilitatea la diferite scenarii, acesta oferă o soluție flexibilă și extensibilă pentru procesarea eficientă a imaginilor în mediul Java.

Bibliografie

https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-space-conversion/

https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-grayscale-image-in-java

https://www.geeksforgeeks.org/image-processing-in-java-colored-image-to-grayscale-image-conversion/