Proiect Aplicații web cu suport java

-Image smoothing-

Enache Alexandra-Anamaria

334AA

Image Smoothing

Procesul de smoothing sau blur este un proces prin care se reduce zgomotul și detaliile unei imagini, aceasta devenind neclara, dar își păstrează totuși conținutul. În cea mai mare parte, dorim în continuare să putem spune în general ce este în interiorul imaginii. Fiecare pixel din imagine trebuie modificat în așa fel încât să fie diferit de cel anterior, dar să nu fie complet deformat încât să facă conținutul imaginii nedescifrabil. În practică acest lucru înseamnă că fiecare pixel din imaginea sursă se împrăștie și se amestecă în pixelii înconjurători.

Pentru a realiza efectul de smoothing asupra unei imagini se aplica o mască de convoluție (nucleu/ kernel) fiecarui pixel din imagine, mai exact asupra celor trei canale de culori a fiecărui pixel (roșu, verde si albastru). Kernel – ul este în esență o matrice bidimensională care conține numere utilizate pentru manipularea imaginii, care este, de asemenea, o matrice bidimensionala de pixeli și este ceea ce determină proprietățile convoluției. Convoluția este procesul de adăugare a fiecărui element al imaginii la vecinii săi locali, ponderat de nucleu.

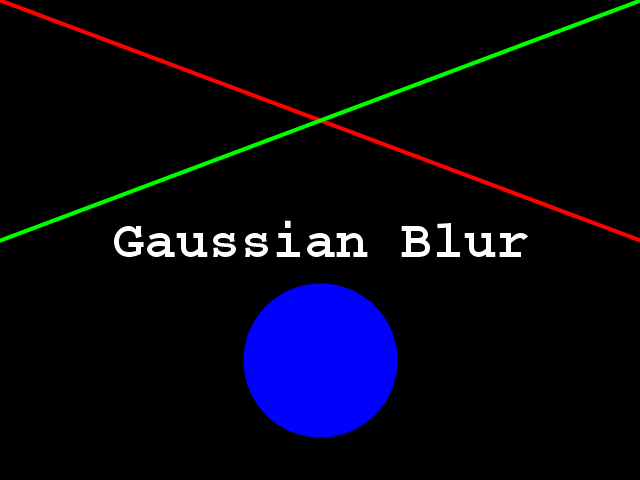
În cadrul proiectului am folosit un nucleu cu dimensiunea 3x3 pentru efectul de smoothing.

Imaginile care sunt introduse pentru a fi convertite trebuie sa aibă format 24bit BMP-RGB. Acest format de stocare al imaginii combină culorile roșu, verde si albastru pentru fiecare pixel. Fiecare pixel este reprezentat sub forma unui întreg în care primii 8 biți corespund culorii albastru, urmatorii 8 biți corespund culorii verde, iar ultimii 8 biți corespund culorii roșu.

Pentru rularea programului se va introduce doar numele pozei (fără path sau extensie) asupra căreia se dorește să se aplice filtrul smoothing, apoi numele cu care se va dori salvarea imaginii blurate.

Exemplu de smoothing:

Original image

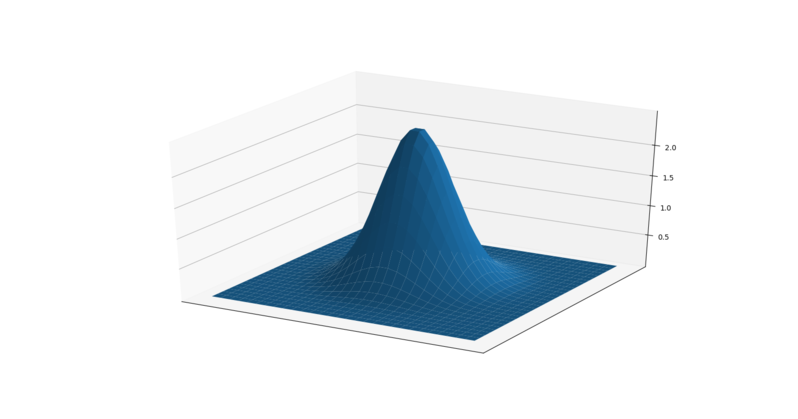


Before smoothing



Pentru efectul de smoothing, considerăm un grup dreptunghiular de pixeli care înconjoară pixelul de filtrat. Acest grup de pixeli, numit nucleu, se deplasează împreună cu pixelul care este filtrat. Pentru ca pixelul filtrului să fie întotdeauna în centrul nucleului, lățimea și înălțimea nucleului trebuie să fie impare. Pentru a aplica acest filtru pixelului curent, se calculează o medie ponderată a valorilor culorilor pixelilor din nucleu. Pixelii cei mai apropiați de centrul nucleului au o greutate mai mare decât cei îndepărtați de centru. Această medie se face de la un canal la altul, iar valorile medii ale canalului devin noua valoare pentru pixelul filtrat. Nucleele mai mari au mai multe valori luate în calcul în medie, ceea ce duce la un efect de blur mai pronunțat.

Aplicarea filtrului Gaussian



Structura proiectului

Programul este format din 6 clase:

* Main – clasa principala
* Buffer – clasa buffer pentru imagine
* Producer – clasa care moștenește un thread
* Consumer – clasa care moștenește un thread
* ImageInterface – interfața
* Image - clasa care prelucrează imaginea
* Main – este clasa principală a proiectului. De aici se începe rularea programului si pornirea thread-urilor.
* Buffer – clasa cu ajutorul căreia manipulăm pixelii imaginii cu ajutorul metodelor get() și put(). Metoda put() este cea care se ocupă de punerea pixelilor în matricea imaginii, iar metoda get() este cea care returnează pixelii imaginii.
* Producer – este thread-ul care citeste o pătrime din imagine prin intermediul metodei get() din clasa Buffer. După fiecare sfert de imagine citit, thread-ul intră în starea de sleep pentru 1 secundă.
* Consumer – este thread-ul care preaia informatia transmisă de Producer, asamblând imaginea originală după terminarea celor 2 thread-uri. De asemenea, de aici se apelează metoda blurImage() care va produce efectul de smoothing (blur) asupra imaginii.
* ImageInterface – este interfața în care avem metodele principale de procesare a imaginii (citire/ scriere pixeli) prin intermediul thread-urilor.
* Image – este clasa care implementează metodele get() si put() ale interfeței si totodată are si alte metode care asigură prelucrarea imaginii. Metodele getWidth() si getHeight() returnează lungimea si lățimea imaginii. Metodele getImage() si setImage() returnează imaginea, respectiv pune o imagine care este dată ca parametru într-un buffer. Cu ajutorul metodei putPixel se construiește matricea de pixeli a imaginii. Metodele blurImage() si convolvePixel() sunt cele care se ocupa efectiv de efectul de smoothing asupra imaginii procesate, fiind cele care adaugă masca de convoluție asupra fiecărui canal de culori (RGB) al fiecărui pixel din imagine.

Performante

Kerner-ul pe care l-am folosit pentru masca de convoluție are dimensiunea de 3x3 tocmai pentru a scădea complexitatea programului, deoarece chiar daca un nucleu cu o dimensiune mai mare accentuează efectul de smoothing, de asemenea crește și complexitatea programului, devenind un algoritm încet. Complexitatea algoritmului este de O(Kw\*Kh\*Iw\*Ih), unde Kw, Kh – lungimea, respective lățimea nucleului, iar Iw si Ih sunt dimensiunile imaginii procesate. De asemenea, timpul necesar citirii si încărcării imaginii în memorie sunt mai lungi decât procesare propriu-zisă a imaginii.

Exemplu de rulare a programului:

Imaginea originală - > tiger



Imaginea după rularea programului și aplicarea efectului de smoothing –> out\_tiger

