PROIECT PROCESARE DE IMAGINI –COMPRESIE SI DECOMPRESIE JPEG

Una dintre marile provocări ale procesării de imagini este lucrul cu cantități mari de date. Deoarece majoritatea imaginilor au rezoluții mari, împărțite pe mai multe canale. O imagine este compusă din pixeli, iar valoarea acestora le determină culoarea. Această abordare simplă fata de imagini este reflectată în formatul bitmap (.bmp) pentru imagini. Este rar folosit în zilele noastre, deoarece dimensiunea lui face dificil lucrul cu el.

Pentru a reduce dimensiunea mare a acestor imagini și pentru a le face mai ușor de folosit pentru aplicații de procesare (sau aplicații AI), aplicăm diferiți algoritmi de compresie pe aceste imagini bitmap de bază pentru a obține înapoi un fișier care este mai mic ca dimensiune, dar care conține totuși conținutul original al imaginii într-un mod pe care îl putem decoda.

Compresia imaginii poate fi fie cu pierderi, fie fără pierderi, în funcție de dacă pierdem sau nu informații despre imagine atunci când îi aplicăm compresie. Algoritmii de compresie au si o caracteristica numita compression ratio, exprimata in procente, care defineste cat de mare este imaginea comprimata in comparatie cu originala.

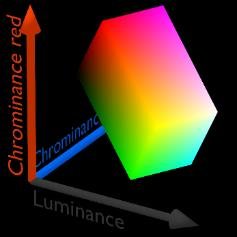
Algoritmii cu pierderi au, în general, rapoarte de compresie mai bune decât algoritmii fără pierderi, deoarece putem omite sau aproxima anumite caracteristici ale imaginii, reducand astfel dependenta de date explicite pentru imagine.

Unul dintre cele mai folositi algoritmi de compresie este compresia jpeg(Joint Photographic Experts Group). Este un algoritm lossy, deci pierdem calitate vizuala, dar poate ajunge la rapoarte de compresie foarte bune, dar acesta depinde de nivelul de detaliu dorit.

Algoritmul urmareste urmatorii pasi, incepand cu imaginea .bmp.

1. **Conversia RGB -> YCbCr.**

Fiindca ochiul uman este mai sensibil la intensitate decat culoare, incepem prin a converti spatiul de culoare al imaginii in Y(luminance) Cb (Chrominance-Blue) si Cr(Chrominance-Red).



Spatiul de culoare Y Cb Cr

Sursa: https://www.researchgate.net/figure/YCbCr-Color-Space-In-the-YCbCr-component-of-Y-is-a-luminance-brightness-whereas\_fig4\_323025324

Pentru a converti imaginea RGB in Y Cb Cr, aplicam urmatoarele operatii asupra canalelor RGB, care contin valori intre 0 si 255.

Aceste operatii vor rezulta in valori tot intre 0 si 255 pe care le putem folosi mai departe.

1. **Downsampling(Reducerea detaliului)**

Pentru a reduce mai departe mărimea fișierului, se reduce rezoluția canalelor Cb si Cr, ,astfel se reduce mărimea imaginii, păstrând totuși datele importante de luminance.

Astfel, vom imparti rezolutia canalelor Cr si Cb la 2, iar pentru a afisa imaginea ne va trebui upsampling facut pe canalele Cr si Cb pentru a aproxima pixelii lipsa din acele canale.

Upsamplingul se poate face in mai multe moduri:

* Nearest neighbor
* Interpolare Biliniara
* Interpolare Bicubica

1. **Împărțirea in blocuri**

Imaginea este împărțita in blocuri de 8x8 pixeli, pentru a aplica urmatorul pas(Discrete Cosine Transform), pe o subsectiune a imaginii si nu pe toata imaginea. Fiindca am impartit deja rezolutia canalelor Cb si Cr la 2, impartind-o mai departe la 8 vom reduce marimea acelor canale de 16 ori, ceea ce ajuta mult la comprimare.

1. **Transformata Cosinus Discreta**

Fiecare bloc 8x8 trece printr-o transformare discreta, ce transforma valorile pixelilor in componente de frecventa. Fiindca ochiul uman este mai sensibil la frecvente mici, compresia jpeg se concentreaza pe pastrarea acestora in detrimentul celor mai inalte.

Pentru fiecare bloc 8x8, aplicam urmatoarea transformata:

Unde:

f(x,y) este valoarea unui pixel la pozitia (x,y)

F(u,v) este componenta de frecventa a transformatei

C(u) si C(v) sunt componente de normalizare, egali cu daca u = 0 sau v = 0, altfel 1

1. **Cuantizare**

Coeficienții transformatei cosinus sunt împărțiți la un coeficient de cuantizare dintr-un tabel de cuantizare standard, apoi sunt rotunjite in sus la cel mai apropiat intreg.

Tabelele de cuantizare sunt definite pentru luminanta(Y), si pentru canalele Chroma(Cr, Cb).

In G avem componentele de frecventa din DCT, iar in Q avem valorile matricii de cuantizare.

1. **Scanare ZigZag**

Dupa cuantizarea blocurilor, matricea contine cele mai importante valori(frecventele mici) in coltul stanga sus, iar cele mai putin importante in coltul dreapta jos.

Scanarea Zig-Zag reordoneaza matricea intr-o lista unidimensionala, punand zerouri la final, dupa care putem aplica run-length encoding pentru a fi mai eficienti.

Run-length encoding grupeaza valorile repetate stocand separat de cate ori apare o valoare si valoarea in sine. Acest lucru ne ajuta mult deoarece la final, dupa scanarea zig-zag, vom avea multe valori similare ce pot fi astfel comprimate.

1. **Crearea fisierului**

Datele comprimate, impreuna cu metadate, sunt grupate impreuna intr-un fisier cu extensia .jpg sau .jpeg. Care poate apoi fi deschis si afisat prin aplicarea algoritmului invers(decompresie).