JPEG Compression

Proiect realizat de:

Ipatiov Grigor

302310

05.06.2020

Link catre repository-ul codului sursa:

<https://github.com/GrigIp/jpeg-compression>

1. Introducere

Contextul temei

Se cere implementarea unui program ce va fi capabil de a lua o imagine, de a compresa aceasta imagine dupa formatul jpeg, iar mai apoi, decompresia si reconstructia imaginii aplicand algoritmul invers.

Probleme de rezolvat

- Modificarea valorilor pixelilor imaginii introduse pentru reducerea dimensiunilor datelor relevante pentru imagine.

- Codificarea noilor valori relevante.

- Aplicarea procedeului de compresie in mod invers pentru reconstructia imaginii.

Obiective propuse

- Compresia imaginilor in format jpeg

- Decompresia si reconstructia acestor imagini.

1. Fundamentare teoretica

“[JPEG] este o metoda de compresie. Aceasta devine o metoda standard de compresie in 1992 iar acronimul JPEG vine de la Joint Photographic Experts Group. JPEG este o metoda de compresie lossy.”

[Danoja Dias] - <https://medium.com/breaktheloop/jpeg-compression-algorithm-969af03773da>

“Principalul obiectiv al compresiei imaginii este de a stoca si a transmite datele intr-un mod efficient.”

“JPEG este primul standard international in termeni de compresie a imaginii. Acesta poate sa fie atata de tip lossy cat si de tip lossless.”

- <https://www.tutorialspoint.com/dip/Introduction_to_JPEG_compression.htm>

Conform paginii de Wikipedia “Lossless JPEG” a aparut in 1993 si joaca rol de aditie pentru JPEG-ul standard din 1992.”

- <https://en.wikipedia.org/wiki/Lossless_JPEG>

Lossless vs Lossy

“Compresia lossy se refera la acel tip de compresie in care unele date din fisierul original sunt pierdute. Procesul este unul ireversibil iar, odata convertit cu metoda lossy, nu exista cale de intoarcere. Cu cat compresia este mai repetata, cu atat mai multe date se voi degrada.”

“Unul dintre cele mai mari beneficii evidente este faptul ca fisierul rezultat are o marime semnificativ mai mica.”

“Compresia lossless se refera la compresia in care imaginea este redusa fara a se pierde din calitate. In general, aceasta este realizata prin stergerea metadatelor ce nu sunt necesare din fisiere JPEG sau PNG.”

- <https://www.keycdn.com/support/lossy-vs-lossless>

Descrierea metodei

- Primul pas al metodei de compresie este impartirea imaginii in blocuri de matrice de dimensinue 8 X 8.

- Schimbarea range-ului intensitatii pixelilor de la [0,255] la [-128,127]

- Aplicarea DCT (discrete cosinus transform) pe fiecare bloc al imaginii impartite.

- Aplicarea cuantizarii asupra blocurilor transformate cu ajutorul DCT cu ajutorul unei matrici standard folosita in compresia JPEG si numita maticea “Luminance”.

- Parcurgerea fiecarei matrici cuantizate in zig-zag si transformarea acesteia intr-un vector unidimensional.

- Aplicarea unei metode de compresie lossless pe fiecare vector astfel obtinut.

- Pentru decompresie se vor repeta pasii in mod invers, si se va reconstrui imaginea.

3. Proiectare si implementare.

Pentru a fi capabil sa manipulez datele ce definesc o imagine am ales sa lucrez cu biblioteca openCV pentru C++, biblioteca ce permite, prin functiile si tipurile de data, o manipulare usoara a imaginilor.

Astfel, dupa citirea imaginii din fisier, incep sa parcurg imaginea pe blocuri de dimensiune 8 X 8 iar fiecare pixel este stocat intr-o matrice secundara pe care se vor aplica transformarile. Cu toate acestea, datorita faptului ca imaginile nu se vor imparti mereu intr-un numar exact de blocuri, am grija sa nu ies din extremitatile imaginii, iar acolo unde acest lucru se intampla, in matricea secundara stochez valoarea 0.

Odata cu stocarea pixelilor in matricea secundara (imageBlock), voi scadea de asemenea 128 pentru schimbarea range-ului.

Urmatorii pasi ii reprezinta transformarea DCT si Cuantizarea. Astfel, conform formulelor de la DCT parcurg fiecare pixel al imaginii, componentele CI si CJ ale transformatei iau valori diferite, radical din 1/8 daca ma aflu pe prima linie respectiv, prima coloana si radical din 2/8 in restul cazurilor. Mai departe, se calculeaza suma produselor valorii fiecarui pixel inmultita cu cosinusurile din formula ce poate fi gasita la acest link (prima imagine):

<http://matlab.izmiran.ru/help/toolbox/images/transfo5.html>

In final, aceasta suma se inmulteste cu componentele CI si CJ, iar mai apoi, pentru cuantizare pixelul de la linia i si coloana j va fi impartit cu pixelul de la linia i si coloana j din matricea “Luminance”.

In urma acestor transformari se aplica parcurgerea in zig-zag a matricii iar datele sunt stocate intr-un vector unidimensional.

Pentru partea finala a algoritmului de compresie am ales ca metoda de encodare “Run-Length-Encoding” care presupune stocarea perechilor de tip valoare – numar de aparitii. Pentru acest tip de encodare am folosit o structura numita RLE\_Component ce are doua component, una pentru valoare si una pentru numar de aparente. Astfel, parcurg matricea pozitie cu pozitie, daca pe urmatoarea pozitie este aceeasi valoare ca pe cea anterioara, cresc contorul si merg mai departe. In caz contrar, salvez noua componenta intr-un vector de componente si reincep numaratoarea aparitiilor noii valori de la 1.

Datorita faptului ca la finalul matricii sunt foarte multe valori de 0 si a faptului ca stiu exact cate elemente trebuie sa fie in vector, putem omite salvarea numarului de 0-uri de la final fara a pierde date.

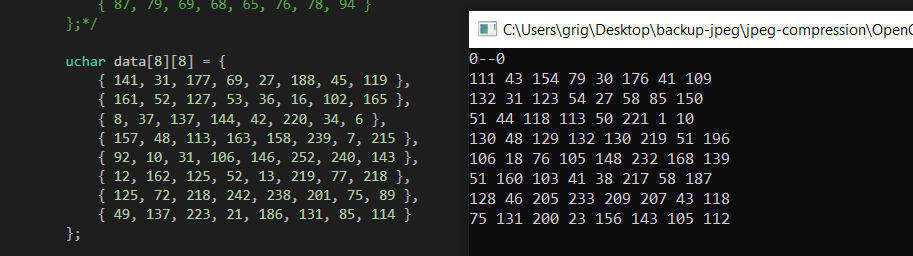
Pentru algoritmul de decompresie, am aplicat parcurgerea zig-zag din nou, de aceasta data stocand in matrice valoarea la care ma aflu in vectorul unidimensional.

Decuantizarea se face prin parcurgerea matricii, iar pentru fiecare pixel, acesta va fi inmultit cu pixelul aferent din matricea “Luminance”.

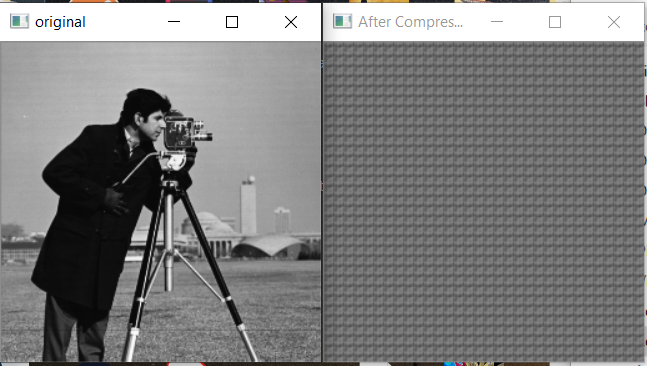
Inversa DCT-ului este asemanatoare cu DCT-ul dar de data asta componentele CI si CJ sunt calculate si folosite de fiecare data cand adaugam un nou termen la suma pixelului curent.

In final se adauga valoarea 128 fiecarui pixel pentru a reveni din nou in intervalul [0,255], iar fiecare bloc este utilizat pentru reconstructia imaginii.

1. Proiectare si implementare



Aplicand algoritmul de compresie si decompresie pe matricea 8X8 din stanga, in dreapta se poate observa rezultatul obtinut. Cu toate ca datele sunt apropiate ca valoare, se poate observa ca prin aplicarea unei metode de tip lossy, la reconstructia datelor acestea vor fi denaturate definitiv.



Cu toate ca pentru un singur bloc de 64 de pixeli compresia si decompresia pare ca functioneaza in modul in care ne asteptam, din pacate, algoritmul nu reuseste sa dea aceleasi rezultate si in momentul in care vine vorba de o imagine mai mare.

1. Concluzii

Algoritmul implementat pare sa ofere un rezultat rezonabil cand vine vorba despre analiza unui singur bloc de pixeli in comparatie cu rezultatele obtinute de acesta in urma procesarii unei imagini intregi.

1. Bibliografie

<http://matlab.izmiran.ru/help/toolbox/images/transfo5.html> - formulele pentru DCT si inversa DCT

<https://www.keycdn.com/support/lossy-vs-lossless> - informatii despre compresiile lossy si lossless

<https://medium.com/breaktheloop/jpeg-compression-algorithm-969af03773da>

<https://www.tutorialspoint.com/dip/Introduction_to_JPEG_compression.htm>

<https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348> - descrierea algoritmului de compresie

<https://www.geeksforgeeks.org/print-matrix-zag-zag-fashion/> - parcurgerea matricii zig-zag.