Факултет за информатички науки
и компјутерско инженерство (ФИНКИ)

Семинарска работа по предметот

Дигитално процесирање на слика

Тема:

Компресија на слики

Проф:

Д-р Ивица Димитровски

Изработил:

Даниел Илиевски 223021

Скопје, јуни 2024 година

Содржина

Вовед

Компресијата на слики се смета како една од клучните работи за брзиот напредок во областа на дигитално процесирање на слики. Во секојдневниот живот, како што нашата употреба и зависност од паметните уреди и компјутери продолжува да расте, така и нашата потреба на складирање на големи количини на податоци расте. На пример, сликите бараат многу простор за складирање и време на пренос. До некој степен, овој проблем може да се реши со поголема брзина на интернет како и дискови за трајно складирање на податоци со поголем капацитет, меѓутоа релативно високата цена е еден голем ограничувачки фактор. Компресија на слики е всушност процесот на намалување на потребната количина на податоци за преставување на дигитални слики и е една од најмоќните и најкорисни техники во дигиталното процесирање на слика. Во компресија на слики, исто така е ставен акцент и на тоа да имаме што е можно помала загуба на квалитет и информации од самата слика. Во оваа семинарска работа ќе се запознаеме подетално за тоа што претставува компресија на слики, кои се најпознатите алгоритми за компресија на слики кои се користат ширум светот, а на крај ќе видиме и практичен пример преку код на еден едноставен алгоритам за компресија на слики.

Што претставува компресија на слики?

Како што кажавме, компресија на слики е процес кој ја прави големината на датотеката на дигиталната слика помала. Ова најчесто се постигнува со остранување на бајти информации од сликите или преку користење на алгоритам за компресија на слики кој всушност ја презапишува самата датотека на сликата, на начин така што би зафаќал помалку простор.

За подетално разбирање, дигиталната слика всушнст преставува дводимензионална матрица. Ова ни овозвожува да користиме линеарна алгебра за да правиме трансформации врз сликата. Вредностите на матрицата ја преставуваат осветленоста на секој пиксел и се цели броеви меѓу 0 (црно) и 255 (бело). Тие се преставени преку 8 битен код. Доколку би користеле слики во боја, тие би ги преставиле како тридимензионални матрици каде што би биле преставени интензитетите на црвената, зелената и сината боја.

Еден систем за компресија на слики, круцијално е да има енкодирачка и декодирачка компонента. Енкодирачката компонента ја зема оригиналната слика како влез, ја процесира и назад враќа компресирана слика како излез. За разлика од неа, декодирачката компонента

е тотално обратна, односно како влез ја зема компресираната слика додека пак на излез враќа слика идентична на оригиналната, декомпресирана слика.

Зошто е важна компресијата на слики?

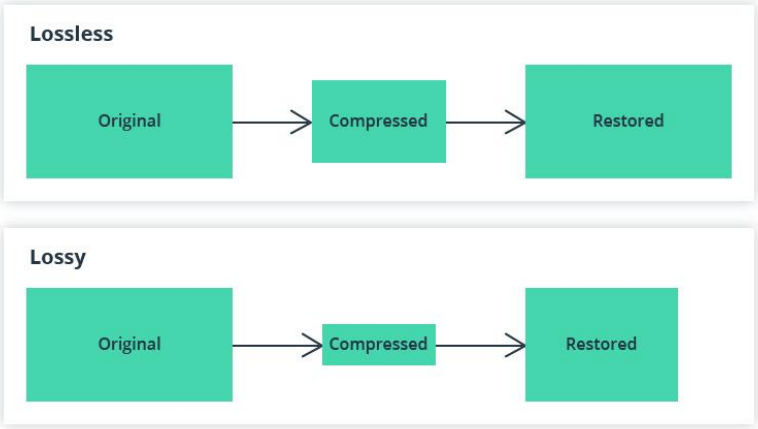
Подобро корисничко искуство на веб страни: Постојат голем број на апликации и веб страници каде компресијата на слики се користи за да се зголеми нивната ефикасност и перформанси, односно самата слика да биде побрзо вчитана кога некој корисник ќе има интеракција со истите. Ова исто така има круцијална улога доколку интернет конекцијата е побавна. Тоа може да бидат веб страни од најразчилен тип како од здравствена индустрија, е-продавници, владини и училишни, галерии, социјални мрежи итн.

Брзина на пренос: Ова е тесно поврзано со претходното. Имено, помалите датотеки побрзо се прикачуваат на интернет, пример кога сакаме да испратиме слики по мејл или да објавиме на социјалните мрежи.

Ефикасност при складирање: Компресираните сликите завземаат помалку простор при складирање, а тоа овозможува повеќе слики да се складираат на диск. Особено клучен фактор играат сликите направени од дигиталните фотоапарати кои имаат прилично висока големина, па е потребно да се изврши некој алгоритам за компресија за да се намали истата.

Генерална поделба на алгоритми за компресија на слики

Алгоритмите за компресија на слики може да се поделат на алгоритми за компресија на слики без загуба (lossless image compression algorithms) и алгоритми компресија на слики со загуба (lossy image compression algorithms)



Што преставуваат алгоритми за компресија на слики со загуба? (Lossy image compression algorithms)

Алгоритмите за компресија на слики со загуба се алгоритми за компресија кои ја намалуваат големината на датотеката на сликата, така што се задржуваат најзначајните информации од сликата. Всушност, овие алгоритми резултираат со трајно бришење на некои податоци. Самиот креатор може да одлучи колава би била стапката на компресија. При декомпресија на сликите, тие податоци не се враќаат. Најчесто, овие изгубени податоци не се видливи за човековото око. Сепак, колку повеќе се компресира сликата, толку повеќе се губи квалитет, а со тоа и станува поприметливо.

Тука се воведува и терминот дисторзија (rate distortion) кој е метрика во овој вид на компресија на слики и ја преставува разликата помеѓу оригиналната и реконструираната слика.

Голем број од сликите прикачени на интернет се во формат на датотека кој користи комбинација на алгоритми за компресија на слики со загуба. Ова придонесува за побрзо вчитување на самите слики. Исто така, вакви алгоритми се користат и за аудио и видео датотеки каде загубата на одредена количина на податоци е прифатлива и незабележителна од поголемиот дел на корисниците. Познати формати на датотеки кои користат вакви алгоритми се:

JPEG (Joint Photographic Experts Group): Овој формат на датотеки е еден од најкористените кога станува збор за компресија на слики со загуба. Имено, составен е од повеќе алгоритми, кои можат да ја компресираат сликата во размер 10:1 со минимална загуба во квалитетот. Поновите верзии на JPEG се JPEG 2000 и JPEG XR, но сеуште не се поддржани од поголемиот дел на пребарувачи.

WebP: Иако овој формат поддржува и компресија на слики без загуба, сепак почесто е користен за компресија на слики со загуба. Креиран е од Google со цел да ги замени JPEG, PNG, и GIF форматите на датотеки.

Што преставуваат алгоритми за компресија на слики без загуба? (Lossless image compression algorithms)

Алгоритмите за компресија на слики без загуба се алгоритми каде откако сликата ќе се декомпресира, квалитетот на сликата останува потполно ист, односно немаме никаква

загуба на податоци. Со други зборови кажано, секој бит на податок останува ист како во оригиналната датотека пред да биде компресирана. Иако може да се намали големината на сликата за 30-40%, сепак ова намалување е прилично помало споредено со компресија на слики со загуба.

Познати формати на датотеки кои користат вакви алгоритми се:

GIF (The Graphics Interchange File): Овој формат се користи за статични слики, едноставни анимации и кратки клипови. Тој е познат по неговата способност да копресира слики со ограничен број на бои (до 256 бои).

PNG (Portable Network Graphics): Најчесто се користи на веб како замена за JPEG и WebP.

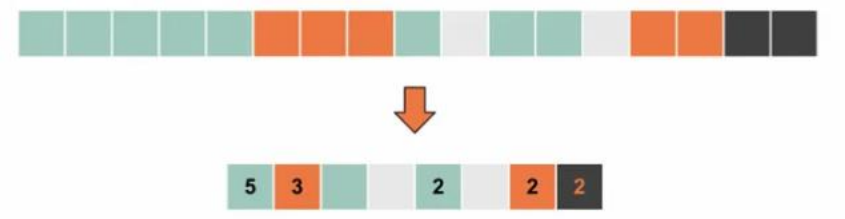
Кои се најпознатите алгоритми за компресија на слики?

Алгоритми за компресија без загуба:

Run-Length Encoding (RLE)

Run-Length Encoding преставува едноставен lossless алгоритам за компресија на слики кој енкодира повторувачки пиксели. Имено, тој ги идентификува последователните идентични пиксели и ги заменува со пар вредност на пикселот и бројач колку пати се повторува истиот. Тој прилично лесно се имплементира и работи добро за слики со големи површини на иста боја. Недостаток на овој алгоритам е тоа дека не е препорачливо да се употребува за слики каде има големи варијации на боите или покомплексни шеми. Таму ефективноста се намалува, бидејќи во некои ситуации, самото зачувување на овие парови може да го надмине бенефитот од компресирање.

Се користи опционално во BMP форматот и во финалните чекори на JPEG.



Entropy encoding

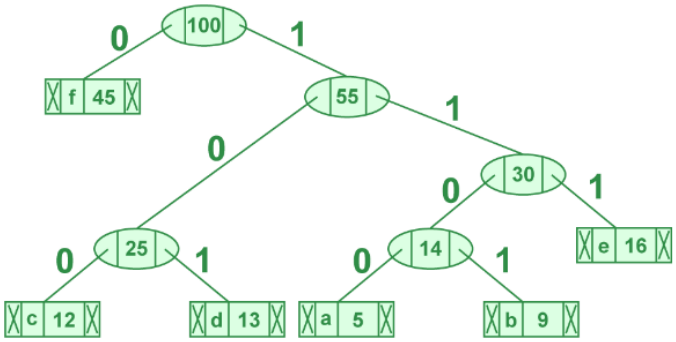
Huffman Encoding

Huffman Encoding, кој преставува тип на Entropy Coding, спаѓа во групата на lossless алгоритми за компресија на слики, кој всушност пикселите кои што се појавуваат почесто, се преставени со пократки секвенци на битови, додека пак оние што се појавуваат поретко се претставени со подолги секвенци.

На почеток, се анализира фреквенцијата на секоја вредност на пикселите во сликата. Користејќи ја оваа анализа се конструира хофманово дрво. Ова преставува бинарно дрво, во кое оние вредности на пиксели кои што се појавуваат почесто, се сместени поблиску до коренот. Хофмановите кодови се доделени на тој начин така што се доделува „0“ на кодот одејќи налево во дрвото и обратно „1“ одејќи надесно. Ова резултира со уникатни кодови за секоја вредност на пикселите. При компресија, секој пиксел во сликата е заменет со соодвениот хуфманов код.

Недостаток на овој алгоритам е тоа што хофмановото дрво треба се чува заедно со компресираните податоци

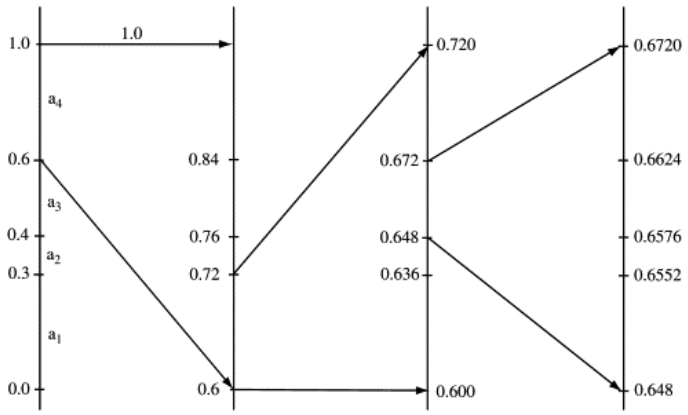
Се користи во PNG форматот како и во JPEG.

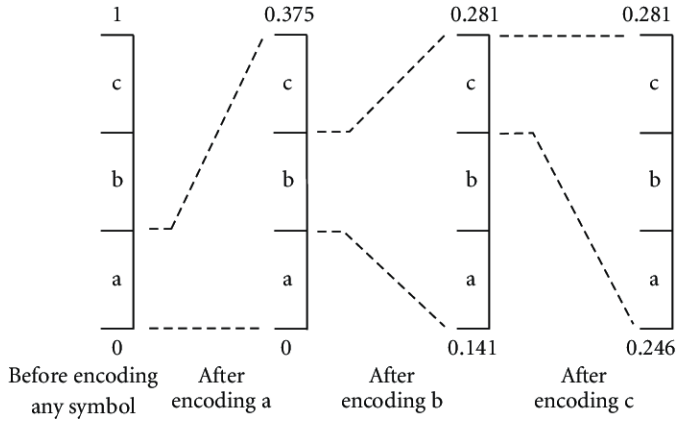


Arithmetic Encoding

Arithmetic Encoding е исто така тип на Entropy Coding, кој спаѓа во групата на lossless алгоритми. Имено, работи на тој начин што претставува цела секвенца од пиксели како една аритметичка вредност во опсегот [0, 1) преку итеративно прецизирање на интервалот врз основа на веројатностите на вредностите на пикселите.

Не се користи често бидејќи заокружувањето на децималните броеви може да доведе до грешки.





Lempel–Ziv–Welch (LZW)

LZW (Lempel-Ziv-Welch) е алгоритам за компресија без загуба кој работи со користење на речник за мапирање на повторувачки секвенции од пиксели со пократки кодови. Алгоритмот започнува со иницијализација на речник кој ги содржи сите можни вредности на пиксели (пр. За 8-битни слики, би имал 256 стартни парови клуч и вредност). Имено, секоја вредност на пиксел е поврзана со уникатен код.

Алгоритмот ги чита вредностите на пикселите од сликата еден по еден и пробува да идентификува секвенци од пиксели. Кога ќе пронајде нова, невидена секвенца таа се додава во речникот со нов код. Така при компресирањето, секоја секвенца која што ја запишал во речникот ја заменува со кодот доделен за истата. Ако имаме големи идентични секвенци од пиксели, тие ќе се заменат со кратки кодови, што значително ја намалува големината на датотеката.

Се користи во GIF и PNG форматот.

Алгоритми за компресија со загуба:

Transform Coding

Transform coding е преставник на алгоритмите за компресија на слики со загуба. Имено, служи за математичко преставување на пикселите од сликата користејќи помалку информации, односно трансформирање од просторен домен во фреквенциски домен. Фреквенцискиот домен, со други зборови, претставува стапка на промена на вредностите на пикселите. Овие фреквенциски коефициенти се делат на ниски и високи. Ниските фреквенциски коефициенти (познати и како low frequency values) ги преставуваат мазните површини додека пак високите фреквенциски коефициенти (познати како high frequency values) ги преставуваат деталите и рабовите во сликата. Повеќето слики имаат повеќе ниски, а помалку високи фреквенциски коефициенти. На овој начин, најзначајните детали од сликата (како најголемите промени во боја или интензитет) се претставени со помал сет на коефициенти кои ефикасно ги опишуваат суштинските делови од сликата. Ова значи дека можеме да ги задржиме битните податоци од сликата, уништувајќи ги помалку битните, а со тоа и намалување на големината на фајлот. Transform coding може да се изведе со повеќе техники како Discrete Cosine Transform или Discrete Wavelet Transform.

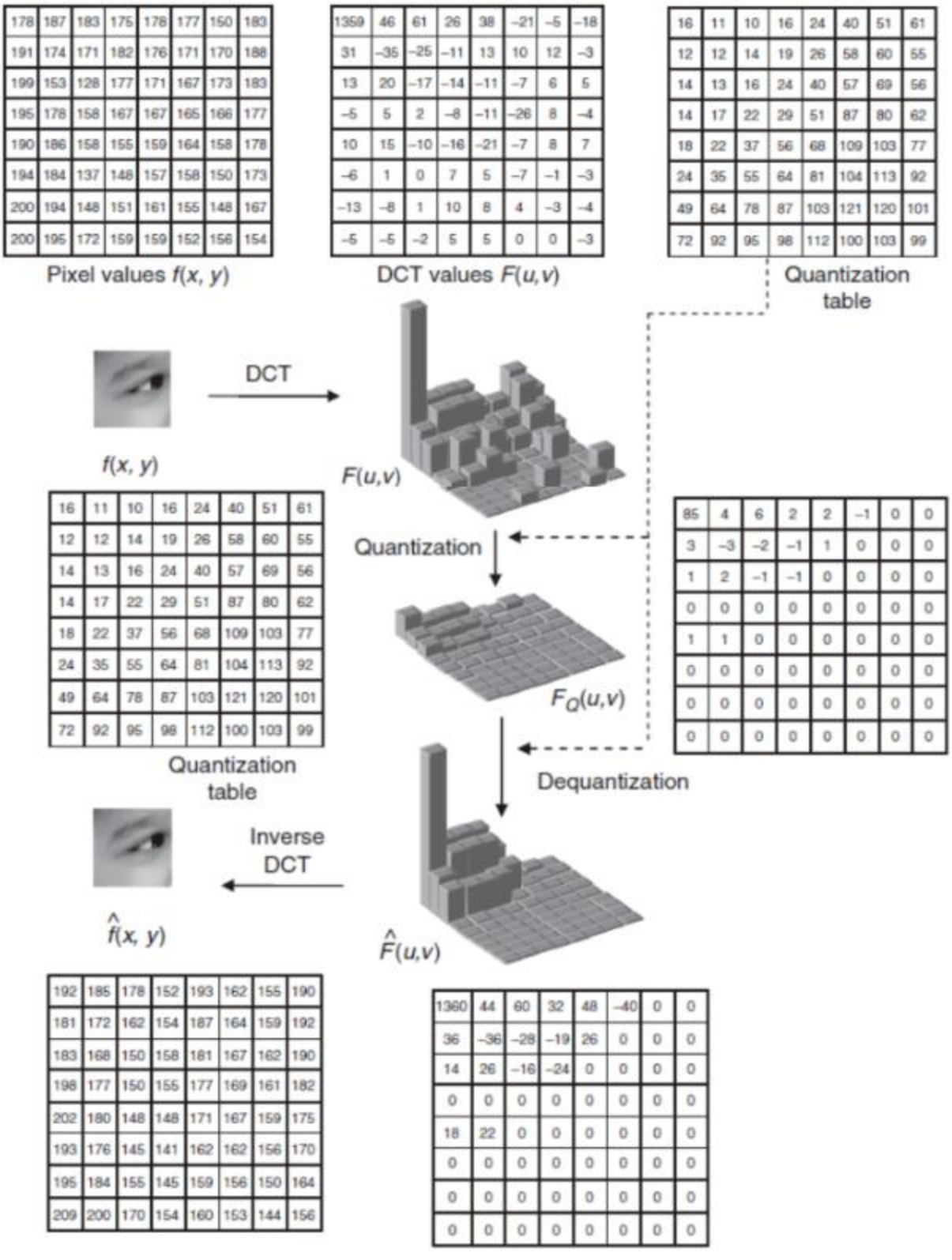
Transform Coding користејќи Discrete Cosine Transform

Започнува на тој начин што целата слика се дели на мали блокови со големина N×N, обично 8x8 пиксели. Се применува DCT формулата на секој блок каде всушност се случува трансформацијата од просторен во фреквенциски домен.

Потоа, се спроведува квантизација на фреквенциските коефициенти добиени од DCT трансформацијата. Квантизацијата се врши со делење на секој фреквенциски коефициент со одредена квантизациска вредност, која е одредена од квантизацискатa матрица. Коефициентите потоа се заокружуваат на најблиската цела бројка. Со ова имаме намалување на точноста на коефициентите, со што се воведува загуба на податоци. Високите фреквенции, кои претставуваат фини детали и шумови во сликата, се претвораат во мали или нулеви вредности, додека ниските фреквенции, кои претставуваат главни детали во сликата, останууваат релативно непроменети.

Квантизацискатa матрица е дизајнирана така што поголемите вредности во неа се користат за високите фреквенции, што води до поголемо намалување на тие вредности, односно поголема компресија и поголема загуба на податоци, додека помалите квантизациски вредности задржуваат повеќе детали, но со помала компресија. Овој чекор значително ја намалува големината на податоците.

Се користи во JPEG форматот.


178 187 183 175 178 177 150 183
191 174 171 182 176 171 170 188
199 153 128 177 171 167 173 183
195 178 158 167 167 165 166 177
190 186 158 155 159 164 158 178
194 184 137 148 157 158 150 173
200 194 148 151 161 155 148 167
200 195 172 159 159 152 156 154
Pixel values f(x, y)
1359 46 61 26 38 −21 −5 −18
31 −35 −25 −11 13 10 12 −3
13 20 −17 −14 −11 −7 6 5
−5 5 2 −8 −11 −26 8 −4
10 15 −10 −16 −21 −7 8 7
−6 1 0 7 5 −7 −1 −3
−13 −8 1 10 8 4 −3 −4
−5 −5 −2 5 5 0 0 −3
DCT values F(u,v)
16 11 10 16 24 40 51 61
12 12 14 19 26 58 60 55
14 13 16 24 40 57 69 56
14 17 22 29 51 87 80 62
18 22 37 56 68 109 103 77
24 35 55 64 81 104 113 92
49 64 78 87 103 121 120 101
72 92 95 98 112 100 103 99
Quantization table
DCT
f(x, y)
F(u,v)
Quantization
F_Q(u,v)
85 4 6 2 2 −1 0 0
3 −3 −2 −1 1 0 0 0
1 2 −1 −1 0 0 0 0
0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Dequantization
16 11 10 16 24 40 51 61
12 12 14 19 26 58 60 55
14 13 16 24 40 57 69 56
14 17 22 29 51 87 80 62
18 22 37 56 68 109 103 77
24 35 55 64 81 104 113 92
49 64 78 87 103 121 120 101
72 92 95 98 112 100 103 99
Quantization table
Inverse DCT
f̂(x, y)
F̂(u,v)
192 185 178 152 193 162 155 190
181 172 162 154 187 164 159 192
183 168 150 158 181 167 162 190
198 177 150 155 177 169 161 182
202 180 148 148 171 167 159 175
193 176 145 141 162 162 156 170
195 184 155 145 159 156 150 164
209 200 170 154 160 153 144 156
1360 44 60 32 48 −40 0 0
36 −36 −28 −19 26 0 0 0
14 26 −16 −24 0 0 0 0
0 0 0 0 0 0 0 0
18 22 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

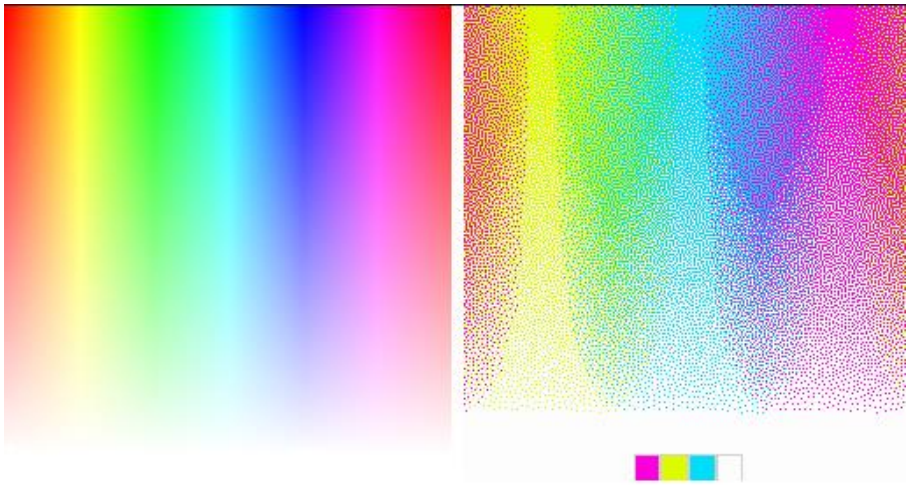
Quantization

Квантизација е типичен алгоритам од групата на lossy алгоритми. Се користи за намалување на бројот на бои во сликата. Се скенира и се избираат најпогодните бои кои ќе бидат репрезентативни за целокупната слика. Ова може да се направи според дистрибуцијата на боите или нивната важност. На пример, алгоритмите како K-means или Median Cut се користат за да ги поделат боите во групи и да ја изберат најдоброто репрезентативна боја за секоја група.

- **K-means**: Овој алгоритам работи со центроиди (средни точки) на групи пиксели со слични бојни вредности. Сликата се дели на кластери според сличноста на вредностите на пикселите, а потоа се избира центроидот (средната вредност) на секој кластер како репрезентативна боја.
- **Median Cut**: Овој метод се базира на разделување на пикселите во сликата врз основа на медијаната во RGB просторот. Вредностите на пикселите се преставени на оска, каде се избира медијаната. По нејзиното одбирање, отприлика пола од пикселите одат на една страна а пола на друга. Постапката продолжува се додека не се постигне потребниот број на бои.

Откако ќе се идентификуваат репрезентативните бои, секоја оригинална боја во сликата се заменува со најблиската репрезентативна боја. Ова се прави со цел да се намали бројот на различни бои, без да се загуби значителноста на изгледот на сликата. Пример ако оригиналната слика има милиони бои, секој пиксел ќе биде заменет со најблиската боја од палетата со репрезентативни бои. Оваа палета се зачувува заедно со податоците на сликата, овозможувајќи правилно прикажување на боите. Овој процес на квантизација на бои ја намалува сложеноста на податоците на сликата, овозможувајќи ефикасна компресија без значителна загуба на квалитетот.

Се користи во GIF.



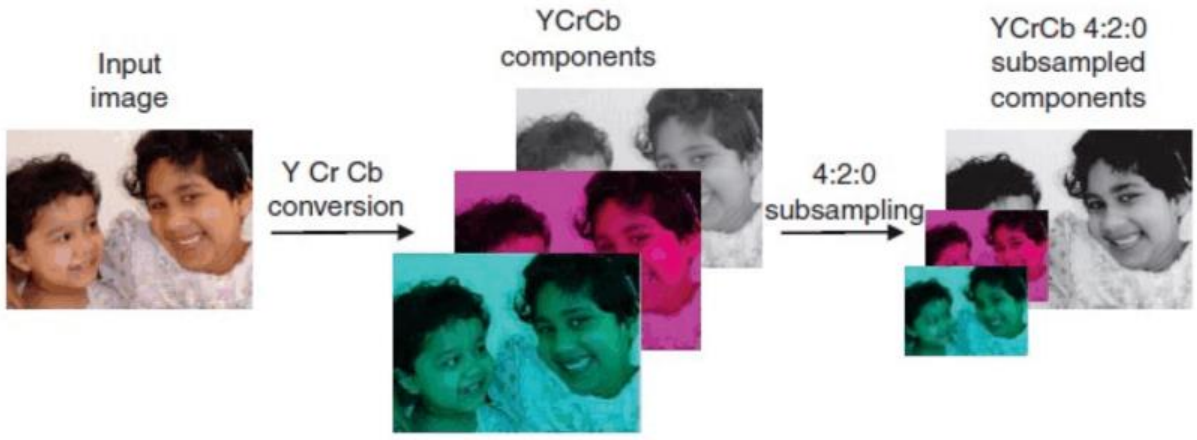
Chroma subsampling

Chroma subsampling исто така спаѓа во групата на lossy алгоритми. Во дигиталните слики, бојата е најчесто преставена со комбинирање на 3 компоненти: црвена, зелена и сина. Со други зборови, сликите се најчесто преставени со RGB просторот на бои. Меѓутоа, човекото око е посензитивно на промени во светлината отколку на промени во бојата. Па така, сликата можеме да ја преставиме од RGB во YCbCr просторот на бои каде Y се однесува на осветленост (лума компонента), а Cb и Cr се хроминансни вредности (хрома компоненти), каде Cb индицира разлика помеѓу сина компонента и некоја рефефентна вредност, додека пак Cr индицира разлика помеѓу црвена компонента и некоја референтна вредност. Тоа значи дека намалување на резолуциите на компонентите за боја а задржување на резолуцијата на компонентата за светлина може значително да ја намали големината на сликата без да биде видно променет нејзиниот квалитет. Ова е исто познато како субсемплирање. Најпознати стандарди или шеми за субсемплирање се 4:4:4, 4:2:2 и 4:2:0

- **4:4:4**: Нема субсемплирање. Ова значи дека сите три компоненти (Y, Cb, Cr) имаат иста резолуција.

- **4:2:2**: Хроминансните компоненти (Cb и Cr) се семплирани со половина од резолуцијата хоризонтално. Ова значи дека за секои два пиксели во секој ред, осветленоста (Y) е семплирана два пати, но хроминансните компоненти (Cb и Cr) само еднаш.

- **4:2:0**: Хроминансните компоненти се семплирани со половина од резолуцијата и хоризонтално и вертикално. Ова значи дека за секои четири пиксели (два по хоризонтала и два по вертикала), осветленоста (Y) е семплирана четири пати, но хроминансните компоненти (Cb и Cr) само еднаш.



Предности и недостатоци при користење на алгоритми за компресија на слики без загуба

Главната предност при користење на алгоритми за компресија на слика без загуба е тоа што квалитетот на сликата може да вратен имајќи компресиран фајл со помала големина. Во JPEG и PNG форматите, ова е направено со бришење на непотребна мета дата. За ситуации каде што е важно да се зачува квалитетот на фајлот, оваа lossless компресија е подобар избор.

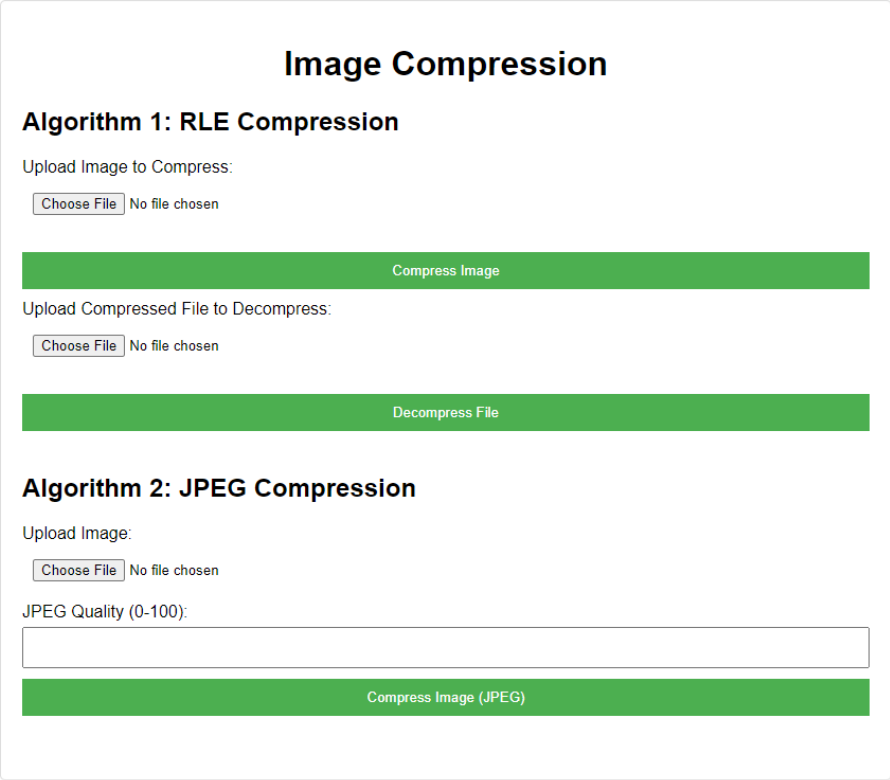
Недостатокот од овие алгоритми е тоа што резултираат со помали стапки на компресија и може да бараат поголема процесирачка моќ и време за да се компресира и декомпресира сликата.

Предности и недостатоци при користење на алгоритми за компресија на слики со загуба

Користењето на алгоритми со компресија на слики со загуба резултира со значително намалена големина на фајлот што е главниот бенефит. Најчесто алгоритмите овозможуваат самите корисници да ја одберат стапката на компресија.

Недостатокот е тоа што резултира со губење на квалитетот, кое можеби не е соодветно за некои ситуации. Колку повисока стапка на компресија, толку повеќе губиток на квалитет имаме. Дополнително, оригиналниот фајл со оригиналниот квалитет не може да биде вратен по компресирањето.

Практичен пример на два едноставни алгоритми за компресија на слики



Алгоритам 1 (Run Length Encoding s compression):

Подолу прикажаната слика преставува функција која служи за компресирање на сликата и нејзино зачувување. Ги земаме димензиите на сликата, и според нив пиксел по пиксел ја измнинуваме. Доколку претходниот пиксел е ист со претходниот го зголемуваме бројачот, во спротивно ја додаваме вредноста на пикселот и бројот на појавувања во листа, а бројачот го ресетираме.

Потоа, ја отвараме датотеката каде сакаме да ги запишеме компресираните податоци. Дополнително, доколку не постои таков фајл, се креира и отвара со привилегии за запишување. Прво ја запишуваме висината и ширината на сликата, а потоа ги измнинуваме компресираните податоци и запишуваме во фајлот, во облик: број на појавувања на пикселот и вредност на пикселот

```python
def rle_compress_and_save(image, filename):
    compressed_data = []
    height, width = image.shape

    for i in range(height):
        run_length = 1
        current_pixel = image[i, 0]

        for j in range(1, width):
            if image[i, j] == current_pixel:
                run_length += 1
            else:
                compressed_data.append((run_length, current_pixel))
                current_pixel = image[i, j]
                run_length = 1

        compressed_data.append((run_length, current_pixel))

    try:
        with open(filename, 'w') as f:
            f.write(f"{height} {width}\n")
            for run_length, pixel_value in compressed_data:
                f.write(f"{run_length} {pixel_value}\n")
        print(f"Compressed data saved to '{filename}'")
    except Exception as e:
        print(f"Error saving compressed data: {e}")
```

Втората функција служи за читање на компресираните податоци и декомпресирање. Почетокот на оваа функција е сосема инверзна на претходната, каде податоците запишани во фајл ги читаме и додаваме во листата како tuples. Претходно ги читаме висината и ширината. Потоа се прави празна numpy array каде ќе биде зачувана декомпресираната слика. Па така, во јамка изминуваме низ компресираните податоци кои се состојат од вредност на пиксел и број на појавување. При секое изминување го зголемуваме бројачот за тековната колона, а доколку стигнеме до крајот на редицата, тековниот бројач за колона се ресетира, додека пак тековниот бројач за редица се зголемува за 1.

```
2 usages
def load_and_rle_decompress(filename):
    compressed_data = []
    try:
        with open(filename, 'r') as f:
            height, width = map(int, f.readline().strip().split())
            for line in f:
                run_length, pixel_value = map(int, line.strip().split())
                compressed_data.append((run_length, pixel_value))
    except Exception as e:
        print(f"Error loading compressed data: {e}")
        return None

    decompressed_image = np.zeros( shape: (height, width), dtype=np.uint8)
    current_row = 0
    current_col = 0

    for run_length, pixel_value in compressed_data:
        for _ in range(run_length):
            decompressed_image[current_row, current_col] = pixel_value
            current_col += 1
            if current_col == width:
                current_col = 0
                current_row += 1
                if current_row == height:
                    break

    return decompressed_image
```

Откако ги објаснивме двете фунцкии, следуваат функциите кои реагираат на соодветните рути /rle-compress и /rle-decompress, односно одговараат на HTTP POST барања.

Во функцијата rle_compress_image прво, се проверува дали постои датотека со име image во барањето. Доколку не постои, се враќа порака "No image part". Потоа, се добива датотеката од барањето и се проверува дали има име. Ако нема, се враќа порака "No selected file". Доколку постои датотека, се креира целосен пат до каде ќе се зачува во UPLOAD_FOLDER и датотеката се зачувува на таа локација. Потоа, сликата се чита и се креира целосен пат за компресираната датотека која ќе се зачува во COMPRESSED_FOLDER со име на оригиналната слика плус _compressed_data.txt. Сликата се компресира користејќи ја погоре објаснетата функцијата rle_compress_and_save и се зачувува на компресираниот пат. На крај, компресираната датотека се испраќа назад како attachment.

```python
@app.route( rule: '/rle-compress', methods=['POST'])
def rle_compress_image():
    if 'image' not in request.files:
        return "No image part"

    file = request.files['image']
    if file.filename == '':
        return "No selected file"

    if file:
        filename = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(filename)

        image = cv2.imread(filename, flags: 0)
        compressed_file_path = os.path.join(COMPRESSED_FOLDER, file.filename + '_compressed_data.txt')
        rle_compress_and_save(image, compressed_file_path)
        return send_file(compressed_file_path, as_attachment=True)
```

На многу сличен начин работи и другата функција rle_decompress_file.

```python
@app.route( rule: '/rle-decompress', methods=['POST'])
def rle_decompress_file():
    if 'text_file' not in request.files:
        return "No text file part"

    file = request.files['text_file']
    if file.filename == '':
        return "No selected file"

    if file:
        filename = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(filename)

        decompressed_image = load_and_rle_decompress(filename)
        if decompressed_image is None:
            return "Error decompressing file"
        decompressed_image_path = os.path.join(DECOMPRESSED_FOLDER, 'decompressed_image.bmp')
        cv2.imwrite(decompressed_image_path, decompressed_image)
        return send_file(decompressed_image_path, as_attachment=True)
```

Алгоритам 2 (JPEG compression):

Вториот алгоритам е дополнителен и користи готови функции од библиотеката Open CV2. Всушност се дава можност да се промени квалитетот на JPG сликата односно да се компресира дополнително со веќе постоечкиот алгоритам за компресирање кој го користи JPEG.

```python
@app.route( rule: '/jpeg-compress', methods=['POST'])
def jpeg_compress_image():
    if 'image' not in request.files or 'jpeg_quality' not in request.form:
        return "No image or JPEG quality part"

    file = request.files['image']
    jpeg_quality = int(request.form['jpeg_quality'])

    if file.filename == '':
        return "No selected file"

    if file:
        filename = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(filename)

        compressed_file_path = os.path.join(COMPRESSED_FOLDER, 'compressed_' + file.filename)
        try:
            jpeg_compress(filename, compressed_file_path, jpeg_quality)
            return send_file(compressed_file_path, as_attachment=True)
        except Exception as e:
            return str(e)
```

```python
def jpeg_compress(image_path, output_path, jpeg_quality):
    if not (0 <= jpeg_quality <= 100):
        raise ValueError("JPEG quality must be between 0 and 100")

    image = cv2.imread(image_path)
    if image is None:
        raise FileNotFoundError(f"The image at {image_path} could not be found or read.")

    success = cv2.imwrite(output_path, image, params: [cv2.IMWRITE_JPEG_QUALITY, jpeg_quality-5])
    if not success:
        raise IOError(f"Failed to save the compressed image to {output_path}")

    print(f"Image saved as {output_path} with JPEG quality {jpeg_quality}")
```

Кодот е прилично сличен на претходниот алгоритам.

Заклучок

Како заклучок, компресијата на слики игра витална улога во современото дигитално процесирање на слики, со значително влијание на различни аспекти од користењето на интернет и дигиталните медиуми. Првичната предност на компресијата е во значителното намалување на големината на датотеките, што не само што ги олеснува преносот и складирањето на слики, туку и значително го подобрува перформансот на веб страници и други платформи. Оваа технологија го овозможува поефикасниот пренос на податоци, што е критично за брзата и ефикасна комуникација во денешното друштво.

Со развојот на компресионите алгоритми, постојат различни пристапи за постигнување компромис меѓу квалитетот на сликата и просторот за складирање. Lossless методите осигуруваат висок квалитет без загуба на информации, додека lossy методите применуваат различни техники за редуцирање на големината на датотеките со минимален губиток на визуелни детали. Овие опции овозможуваат разнообразни примени во зависност од потребите на корисниците и специфичните услови на употреба.

Со непрекинатиот напредок на технологиите, се очекува компресијата на слики да продолжи да се развива, поттикната од потребата за подобрување на ефикасноста и оптимизација на ресурсите. Во иднина, нови напредни алгоритми за компресија можат да доведат до револуција во начинот на кој се менаџираат и споделуваат сликите во дигиталната ера, истовремено обезбедувајќи постојано унапредување на корисничкото искуство и технолошките можности.