Универзитет у Крагујевцу

Факултет инжењерских наука

Александра Д. Нешић

Реализација алгоритма за компресију слика без губитака

мастер рад

Крагујевац, 2020.

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Електротехника и рачунарство

Ниво студија: Мастер академске студије

Предмет: Методе формирања и обраде дигиталне слике

Број индекса: 449/2019

Александра Д. Нешић

Реализација алгоритма за компресију слика без губитака

мастер рад

Комисија за преглед и одбрану

1. Др Маријана Гавриловић Божовић, доцент – ментор

2. _____ Датум одбране: _____

3. _____ Оцена: _____

Задатак мастер рада

У оквиру мастер рада кандидаткиња ће проучавати методе које се користе за компресију дигиталне слике. Рад ће се састојати од теоријског дела, где ће кандидаткиња дати преглед резултата из области и изложити основне алгоритме које се користе у компресији слике са и без губитака, са акцентом на компресију без губитака. Допринос кандидаткиње ће бити и у самостално написаним функцијама које ће вршити компресију слике без губитака за један изабрани алгоритам.

Препоручена литература:

[1] Др Миодраг В. Поповић, Дигитална обрада слике, Академска мисао, Београд, 2006.

[2] Rafael C. Gonzales, Digital Image Processing, 2008.

Крагујевац, октобар 2020.

Ментор:

Др Маријана Гавриловић Божовић

Садржај

Abstract

This thesis presents the realization of a lossless image compression algorithm. The reason for the development of compression algorithms is the efficient storage of data on disk and their transmission through communication channels. Lossless compression algorithms are used for data that would make losses unusable, mainly for textual data and sometimes for audio and image compression purposes, in any case where losses are not allowed. The degree of compression achieved by lossless algorithms is usually not sufficient to accommodate and transmit digital audio, image and film, because the size of this data is enormous. In these cases, lossy algorithms are commonly used. It can be shown that by allowing very little loss in the file being compressed, enormous degrees of compression can be achieved without the loss being even noticeable to our senses. The development of lossy algorithms is of the utmost importance as it is the driver of the multimedia revolution, which has had a decisive impact on increasing the scope and use of both computers and information itself. Despite the great importance of lossy algorithms, lossless algorithms still have a huge significance for all sensitive data where losses are not allowed (text files, medical data and images, etc.).

This thesis builds on the implementation of a lossless image compression algorithm using a technique called Huffman encoding. This technique makes it easy to implement and uses less memory.

Keywords: : Huffman Encoding, Compression, Lossless

Резиме

Овај рад представља реализацију алгоритма за компресију слика без губитака. Разлог развоја алгоритама компресије је ефикасно складиштење података на диск и њихов пренос кроз комуникационе канале. Алгоритми компресије без губитака се користе код података које би губици учинили неупотребљивим, углавном код текстуалних података, а понекад за потребе компресије звука и слике, у сваком случају када губици нису дозвољени. Степен компресије који се постиже алгоритмима без губитака најчешће није довољан за смештање и пренос дигиталног звука, слике и филма, јер је величина ових података огромна. У тим случајевима уобичајено се користе алгоритми са губицима. Може се показати да дозвољавањем јако малих губитака у фајлу који се компресује, могу да се постигну огромни степени компресије,
а да по декомпресији губици не буду ни приметни за наша чула. Развој алгоритама са губицима је од изузетног значаја јер представља покретач мултимедијалне револуције, која је имала пресудан утицај на повећање обима и начина коришћења како рачунара, тако и самих информација. И поред великог значаја алгоритама са губицима, алгоритми без губитака и даље имају огроман значај и примену за све осетљиве податке где губици нису дозвољени (текстуални фајлови, медицински подаци и слике, итд).

Овај рад се заснива на реализацији алгоритма за компресију слика без губитака помоћу технике која се зове Huffman-ово кодирање. Ова техника омогућава једноставну имплементацију и користи мање меморије.

Кључне речи: Huffman-ово кодирање, Компресија, Без губитака

1. Увод

Компресије података почеле су озбиљније да се развијају осамдестих година прошлог века, јер су тада меморија и диск били страховито скупи и много се скупоценог простора трошило на складиштење података у развијеном, тј. некомпресованом облику. Идеја компресије података је на први поглед неприродна. Међутим, користи се чињеница да подаци које људи користе садрже много непотребног садржаја, а по теорији информација може тачно да се израчуна колико износи тај вишак и одбаци оно што је редундантно. Алгоритми компресије заснивају се управо на тој чињеници. Фајлови се у компресованом облику чувају на диску и шаљу кроз комуникационе канале. Да би се могли користити, морају се претходно декомпресовати, тј. вратити у првобитан облик. Клод Шенон је у свом раду из 1948. године "Математичка теорија комуникације" формулисао теорију компресије података [1]. Сама теорија не показује како дизајнирати или имплементирати ове алгоритме. Ипак, ова теорија пружа основна упутства како доћи до оптималних перформанси. С обзиром на велику потребу за компресијом слике, поступци за компресију се интезивно развијају последњих двадесетак година и истраживања у овој области су врло интезивна и данас. Развијене су методе којима је могуће извршити компресију мирне слике чак и до 50 пута без знатног утицаја на квалитет репродуковане слике. У случају секвенце слика степен компресије може бити и већи.

Алгоритми компресије се могу поделити у две категорије: алгоритми компресије без губитака и алгоритми компресије са губицима. Код компресије без губитака добија се фајл који ће после декомпресије бити идентичан оригиналу. Постоји јако много алгоритама компресије без губитака, а два која се најчешће употребљавају су Хофманов кôд и LZW (Lempel Ziv Welch) алгоритам. Оба алгоритма раде на принципу да карактере или низ карактера који се чешће појављују, кодирају краћим кодним речима.

Све методе компресије слике засноване су на коришћењу *просторне редундантности информација* које садржи дигитална слика. Редундантност информација у слици постоји због знатне просторне корелације пиксела и може се описати на различите начине. *Статистичка редундантност* је повезана са расподелом интензитета пиксела слике и може се елиминисати коришћењем ентропијских техника из теорије информација. Елиминацијом статистичке редундантности постиже се компресија без губитака. Редундантност информација у слици се може искористити и за *предикцију* вредности пиксела на основу његовог локалног суседства. Предикцијом се врши декорелација података слике, чиме се смањује њена ентропија. Предиктивне технике се користе и за компресију слика без губитака, и за компресију са губицима.

Најпознатији програм за компресију података без губитака је ZIP, а користи се најчешће за компресовање текстуалних података. Степен компресије који се може постићи алгоритмима без губитака није довољан за смештање на диск и пренос дигиталног звука, слике и филма кроз комуникационе канале. Основни проблем је величина ових фајлова. Из ових разлога почели су да се развијају алгоритми са губицима, који при компресији дозвољавају губитке у фајлу, тако да декомпресовани фајл неће бити једнак оригиналу.

Компресија са губицима је погодна за податке који потичу од дигитализованих слика или звука, јер је дигитална презентација већ и сама само апроксимација. Може се показати да дозвољавањем малих губитака може да се постигне висок степен компресије, а да губици неће бити приметни, или неће бити значајни за потребе наших чула.

Ипак, ако се овакви алгоритми за компресију са губицима и накнадну декомпресију врше узастопно на истим подацима, може доћи до значајног пада квалитета података.

У сваком случају, развој ових алгоритама је револуционаран и заслужан за експоненцијални развој мултимедија. Код компресије звука појава MP3 формата довела је до мултимедијалне револуције. Дигиталне слике су постале природни тип података, јер је створен JPEG стандард за компресију дигиталних слика. JPEG се често користи зато што захтеви за мултимедијом и компресијом слика расту експоненцијално. Међутим, JPEG алгоритам није јединствен стандард, већ скуп правила и препорука са много подесивих параметара. Базиран је на дискретној косинусној трансформацији, која је најједноставнија за хардверску имплементацију и применљив је на широк спектар дигиталних слика, тј. није ограничен резолуцијом слике, системом презентације боја, комплексношћу сцене, итд. JPEG се користи само за компресију статичне слике (једна слика), али је створен сличан стандард MPEG који се бави компресијом покретних слика (низа статичних слика). JPEG и MPEG алгоритми представљају алгоритме компресије са губицима.
Овај рад је базиран на реализацији Хофмановог алгоритма за компресију слика без губитака. Алгоритам је реализован у развојном окружењу MATLAB.

2. Дигитална слика

Слика представља репрезентацију објекта, особе или сцене добијене помоћу неког оптичког уређаја као што је огледало, сочиво или камера. Ова репрезентација је дводимензионална (2D), иако представља неку пројекцију из реалног света - тродимензионални (3D) објекат или сцену. Дигитална слика је нумеричка репрезентација дводимензионалне слике помоћу коначног броја тачака које називамо пикселима. Сваки пиксел је представљен помоћу једне или више нумеричких вредности.

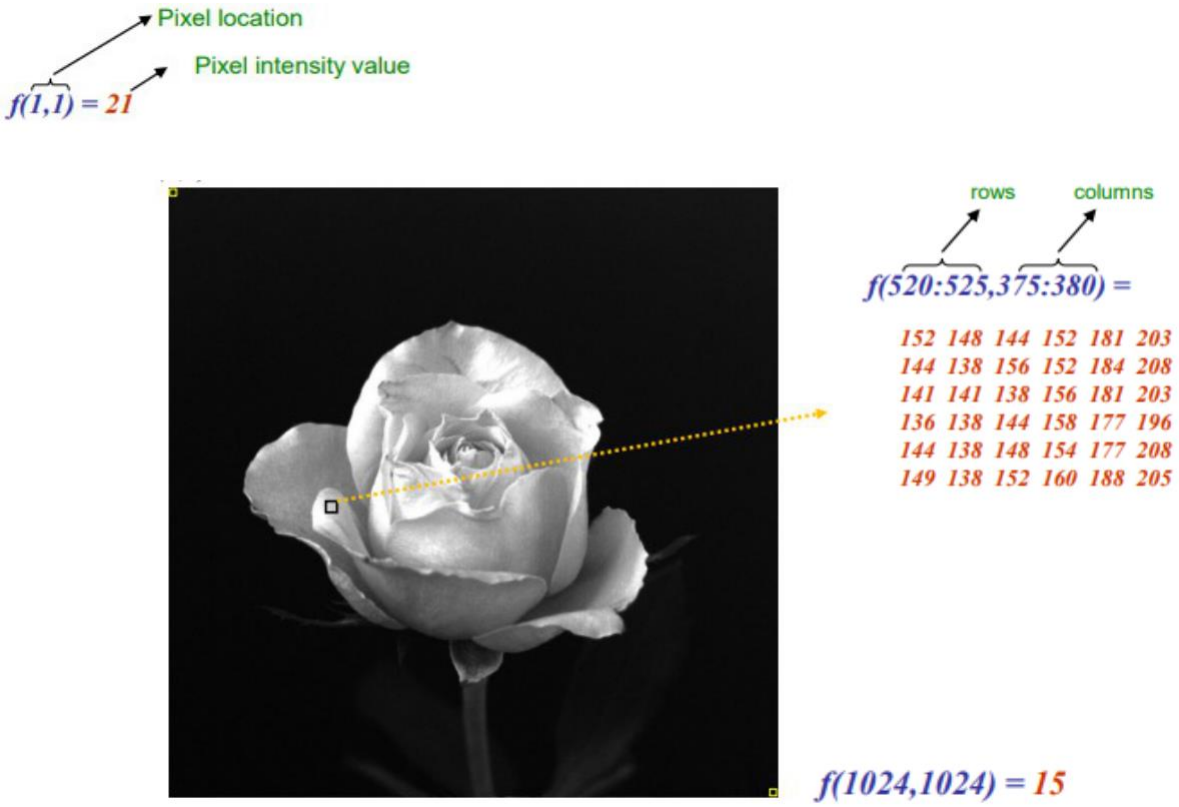
Дигитална слика се представља дводимензионалном реалном матрицом. Са $f(x, y)$ ћемо означавати слику димензије $M \cdot N$, где су x и y просторне координате (x представља број врсте $(0, \ldots, M-1)$, y број колоне $(0, \ldots, N-1)$) и амплитуда f у било ком пару координата (x, y), назива се интензитет слике у тој координати простора.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots & f(0,N-1) \\ f(1,0) & f(1,1) & \ldots & f(1,N-1) \\ \ldots & \ldots & \ldots & \ldots \\ f(M-1,0) & f(M-1,1) & \ldots & f(M-1,N-1) \end{bmatrix}$$

Вредност дводимензионалне функције $f(x, y)$ у пикселу са координатама (x_0, y_0) означава се са $f(x_0, y_0)$ и назива се интензитет слике у том пикселу. Максималне и минималне вредности интензитета зависе од типа слике и конвенције.

Слика 1: *Прва дигитална слика икада, Russell Kirsch 1957. направио је дигиталну слику (176x176) скенирањем фотографије свог тромесечног сина*

Узимамо у обзир да је следећа слика 1024x1024 пиксела (слика 2) 2Д функција или матрица са редовима и колонама. У 8-битном представљању вредности интензитета пиксела се мењају између 0 (црно) и 255 (бело).



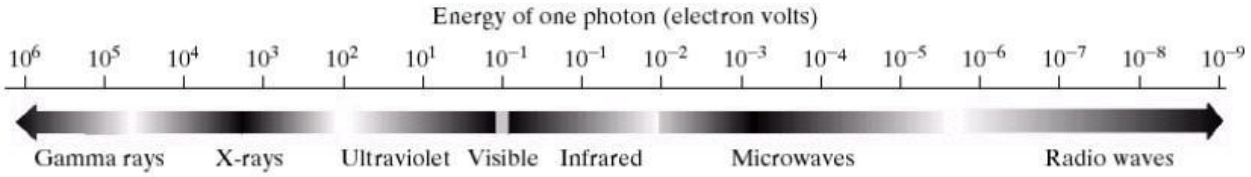
Слика 2: *Представљање слике 1024x1024 пиксела*

2.1 Дигитална обрада слике

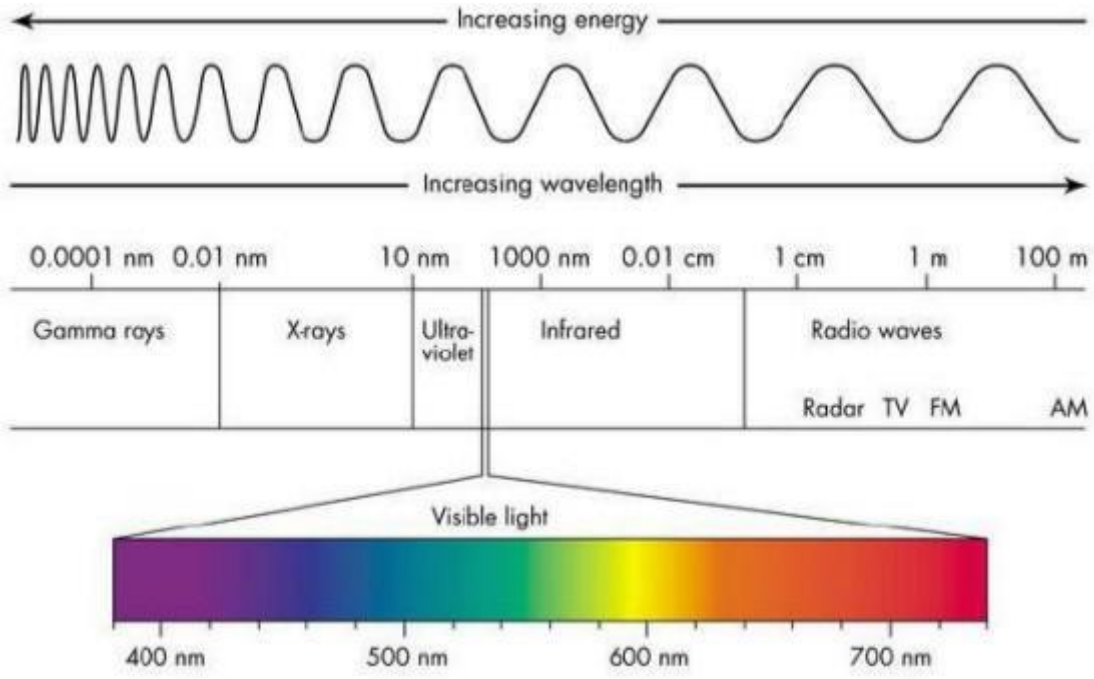
Дигитална обрада слике односи се на обраду дигиталне слике коришћењем дигиталних рачунара.

За формирање слике било на сензору било у људском оку је потребна енергија. Та енергија може да има различите форме па отуда може да се формира слика у различитим деловима електромагнетног (ЕМ) енергетског спектра.

Спектрални опсези су груписани према енергији по фотону у распону од гама зрака (највише енергије) до радио таласа (најнижа енергија).



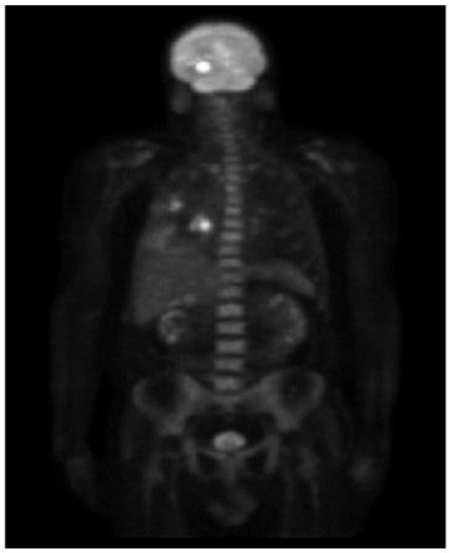
Слика 3: *Електромагнетни спектар уређен према енергији по фотону*

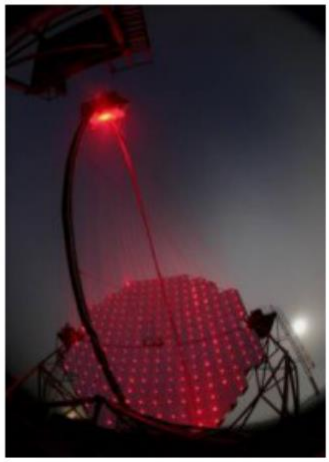


Слика 4: *Електромагнетни спектар; Видљиви спектар је приказан зумирано да би се олакшало појашњење, али имати на уму да је видљиви спектар веома уски део ЕМ спектра*

Дигиталне слике засноване на ЕМ спектру:

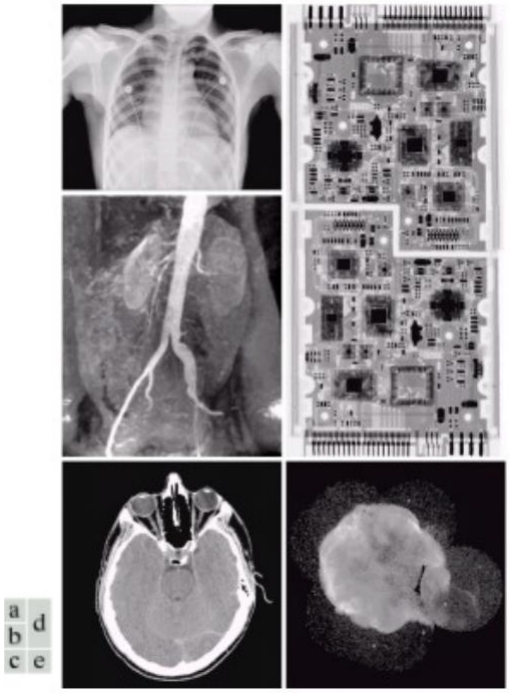
- **Слике у опсегу гама зрачења:** Користи се у нуклеарној медицини и астрономским посматрањима;





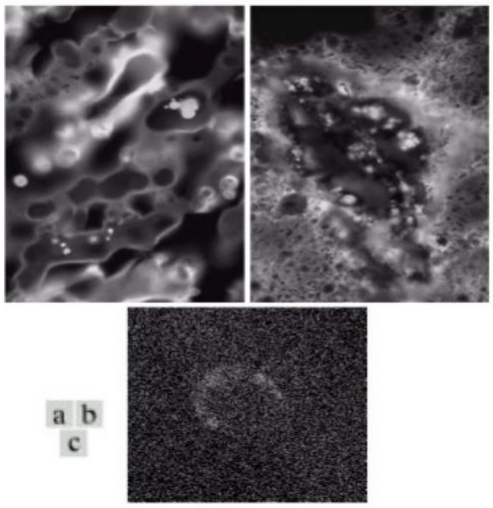
Слика 5: *Гама зрачење, лево је у нуклеарној медицини, десно је Черенковљев телескоп*

- **Слике у опсегу X зрачења:** Користи се у медицинској дијагностици, у индустрији и астрономији.



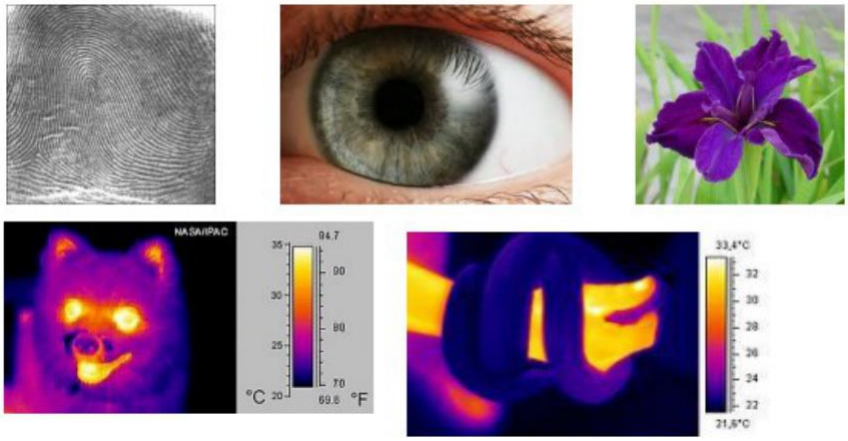
Слика 6: *Примери слика формираних X зрачењем: Грудни кош, Аортни ангиограм, ЦТ главе, Штампана плоча, Цигнус петља ((а) и (ц) Dr. David R.Pickens Dept. of Radiology & Radiological Sciences Vanderbilt University Medical Center; (б) Dr. Thomas R.Gest, Division of Anatomical Sciences University of Michigan Medical School; (д) Mr. Joseph E. Pascente, Lixi, Inc.; (е) NASA)[2]*

- **Слике у опсегу UV (ултраљубичасто) зрачења:** Примене ултраљубичастог зрачења укључује микроскопију, ласере, биолошко сликање и астрономију.



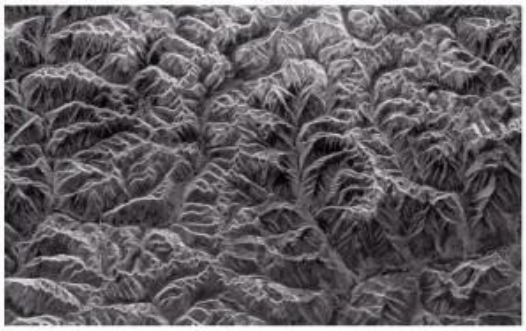
Слика 7: *Примери слика формираних УВ зрачењем: Нормални кукуруз, Запрљан кукуруз, Цигнус петља ((а) и (б) Dr. Michael W. Davidson Florida State University, (ц) NASA)[3]*

- **Слике у опсегу видљиве светлости и инфрацрвеном појасу:** апликације укључују све слике добијене од стране камера, микроскопа и праћења услова околине.



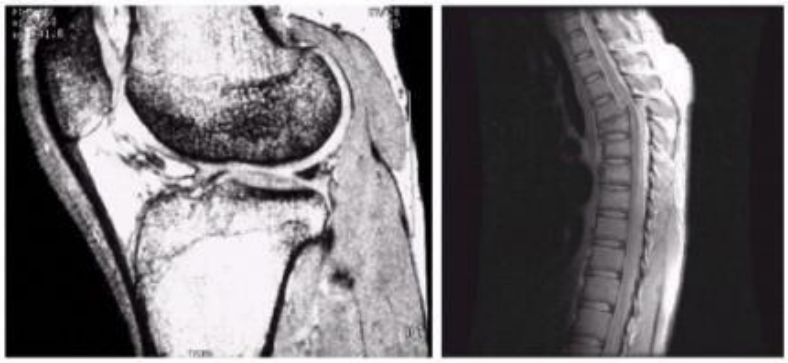
Слика 8: *Примери слика видљиве светлости, инфрацрвена 'топлотна' слика и змија око руке*

- **Слике у микроталасном опсегу:** Апликације које укључују радар, укључујући војне примене и заштиту животне средине.



Слика 9: *Свемирска радарска слика планина на југоистоку Тибета*

- **Слике у радио таласном опсегу:** Апликације које укључују медицинско снимање (Магнетна резонанца – МРИ) и апликације у астрономији.



Слика 10: *МРИ слике човека, лево колено, десно кичма*

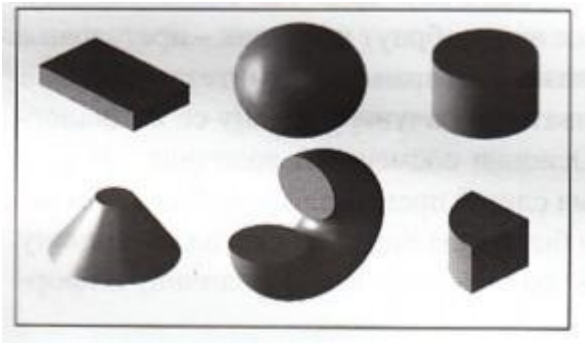
2.2 Векторска и растерска графика

2.2.1 Векторска графика

Вектор (векторска линија) као појам у графици означава одсечак који има своју дужину и смер. Векторска графика означава начин 'цртања' помоћу тих векторских линија. Свака линија садржи три податка – дужину, смер и податак о боји линије. Векторске линије се користе за креирање векторских објеката (различитих геометријских фигура или објеката), при чему се за дефинисање објеката основним подацима (дужина, смер, боја линије) додаје и четврти податак – боја испуне објекта.

Векторским објектом сматра се сваки спој једне или више линија које су затворене – почетна тачка линије уједно је и завршна тачка. Према томе, у векторској графици , на основу једноставних математичких формула, рачунар памти највише четири податка за сваки објекат. Због тога, такве слике и цртежи заузимају мало простора на медијумима за смештај података (хард диск, ЦД, ДВД, флеш...). Осим тога, величина вектора мења се математичк променом вредности његове дужине и смера, што не утиче на квалитет приказа графике.

Векторска графика донедавно се користила углавном за израду једноставних цртежа, шема, логотипа и сл, али су последњих година направљени нови векторски програми који су знатно побољшали могућности векторске графике и по много чему је приближили квалитету растерске графике. Нова област примене векторске графике је Веб графика где је потребно направити квалитетну слику која ће заузимати што мање простора (а ту је векторска графика свакако у предности на растерску). Једини прави недостатак векторске графике у односу на растерску је немогућност приказивања фотореалистичних слика (прелази и нијансе појединих боја).



Слика 11: *Векторски објекти*

2.2.2 Битмапирана (растерска графика)

Растерска графика је 'цртање' помоћу матрице тачака-пиксела, при чему сваки пиксел посебно носи информацију о боји коју репродукује. У односу на векторску графику, растерска има низ недостатака. Величина слике добијене на овај начин и њен квалитет зависе од броја пиксела који је чине. Повећање растерске слике постиже се увећањем димензија постојећих пиксела или додавањем нових, а смањивање слике умањивањем или одузимањем пиксела. Тим поступком добије се физички већа или мања слика, али са израженим опадањем њеног квалитета.
Осим зависности квалитета од величине, слике направљене растерском графиком заузимају много више меморијског простора од слика направљених векторском графиком због тога што сваки пиксел може приказати само једну боју, али садржи податке и о свим бојама које се могу приказати. И поред ових недостатака, значајна предност растерске графике је могућност приказивања фотореалистичних слика и сложених цртежа са финим детаљима.

Слика 12: *Пиксели у растерској графици*

Векторска графика:

- Базира се на принципу геометрије односно вектора;
- сваки вектор има своју почетну тачку, смер и завршну тачку, дужину..;
- служи за цртање геометријских облика : круг, квадрат, троугао, неправилни облици;
- можемо је у бесконачности повећати или смањити без губитака на квалитету јер се базира на математичким функцијама.

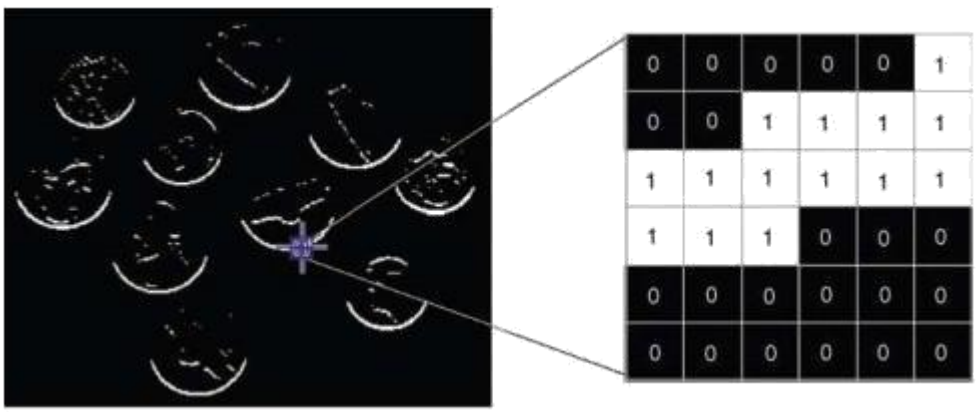
Растерска графика:

- базира се на пикселу (на екрану) или на тачки (на штампаном отиску);
- растер је мрежа хоризонталних и вертикалних линија које стварају поља која зовемо пикселима;
- растерска слика се састоји од пиксела различитих боја и нијанси;
- повећавањем слике долази до губитка квалитета јер растерска слика зависи од густине пиксела па је не можемо у бесконачности повећавати.

2.3 Представљање слике у рачунару

Бинарне (1-битне) слике

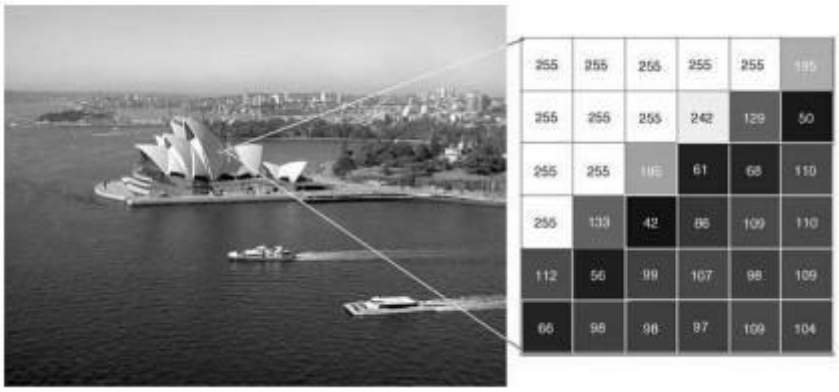
Бинарне слике су кодиране као матрица и користи се 1 бит по пикселу где 0 означава црну а 1 белу боју. Главна предност овакве репрезентације је мала меморија потребна са памћење слика. То су углавном слике које садрже текст или једноставне графичке приказе (слика 13).



Слика 13: *Бинарна слика*

Монохроматске (8-битне) слике

Монохроматске слике такође су кодиране као матрице и то са 8 бита по пикселу. Вредност пиксела 0 означава црну, а вредност 255 белу боју. Све вредности између представљају одређену нијансу сиве боје (слика 14).



Слика 14: *Монохроматска слика*

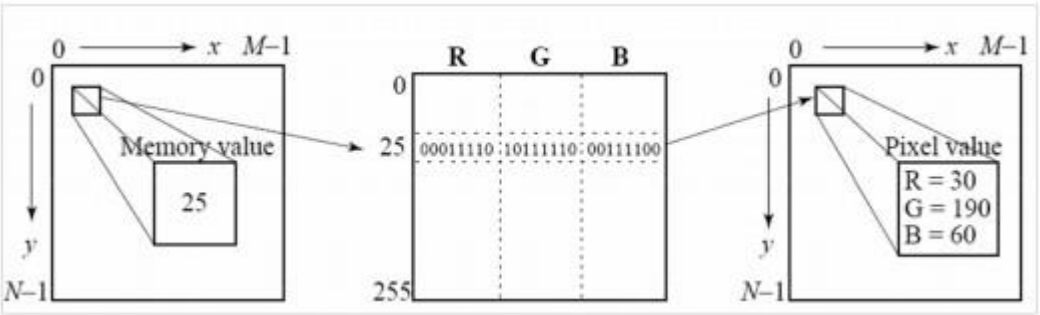
Слике у боји

За репрезентацију слике у боји потребне су три компоненте, односно три вредности светлости, које одговарају интензитетима светлости у спектралним опсезима који приближно одговарају црвеној, зеленој и плавој боји. Разлог за ово лежи у људској перцепцији боје која је заснована на постојању различитих ћелија осетљивих на светлост у наведеним спектралним опсезима. За опис боје се користе различити модели. Дисплеји углавном користе РГБ (**енгл.** RGB red, green, blue) модел код којег компоненте (колор-компоненте) садрже вредности пиксела у црвеном, зеленом и плавом делу спектра.
Слике у боји је могуће представити на два начина, као битмапиране и као индексиране слике. Битмапиране слике за репрезентацију сваког пиксела користе три бајта, односно, 3 × 8 = 24 бита. На овај начин је могуће представити 256^3 = 16.777.216 боја. Ове слике су у литератури на енглеском познате и као *truecolor* слике. Сама слика се меморише коришћењем три матрице од којих свака садржи вредности интензитета пиксела за једну компоненту. Алтернативно, могуће је користити и тродимензионални низ, при чему две димензије одговарају просторним координатама, а трећа колор-компонентама. Слика димензија 640×480 пиксела сада заузима 640×480×3 бајтова = 900 кБ. Пример слике у боји и њених колор-компонената приказан је на слици 15. Поједине колор-компоненте су представљене као сиве слике, при чему су вредности пиксела посматране као интензитети. Често се 24-битне слике меморишу као 32-битне, а додатни бајт се користи за α-канал којим се може нпр. описати транспарентност слике у случајевима када пиксели две слике заузимају исте просторне координате.

Битмапиране слике у боји заузимају три пута више меморије од сивих слика исте резолуције. Медутим, ако се узме у обзир да нису све од више од 16 милиона боја расположивих у 24-битној репрезентацији једнако заступљене у свакој слици, долазимо до закључка да је могуће слику у боји представити и коришћењем мање од 3 бајта по пикселу, уз евентуални губитак визуелног квалитета. Индексиране слике за репрезентацију сваког пиксела користе један бајт уз коришћење прегледне (*lookup*) табеле за чување информација о бојама. У овом контексту се прегледна табела назива и колор-мапа или палета боја.

Када се за репрезентацију пиксела користи један бајт, тада прегледна табела има 256 ставки што одговара палети од 256 боја. Сада су за репрезентацију слике потребни матрица димензија једнаких резолуцији слике и прегледна табела. Сваки елемент матрице заузима 1 бајт што значи да прегледна табела има 256 ставки које заузимају по три бајта – сваки бајт одговара једној колор-компоненти, Р, Г и Б. У матрици слике се за сваки пиксел чува кодна вредност боје — индекс реда у палети. При одређивању боје пиксела потребно је прочитати вредност реда у палети, а онда из палете и саму боју пиксела. Овај процес је илустрован на слици 16. Квалитет индексиране слике је обично нижи од одговарајуће битмапиране слике у боји.

Слика 15: *Слика Лена у боји и њене Р, Г и Б компоненте*



Слика 16: *Палета боја за индексиране 8-битне слике у боји*

2.4 Појам резолуције

Појам резолуције је једноставан, али начин на који се он тумачи може бити збуњујући. Просторна резолуција је мера каолико фино уређај апроксимира континуалну, аналогну слику користећи дискретне елементе - пикселе. Дакле, концепт резолуције је уско повезан са одабирањем и фреквенцијом одабирања. Постоје два начина специфицирања резолуције. За штампаче и скенере резолуција означава број тачака по јединици дужине (dots per inch – dpi). Канцеларијски штампачи обично (2003. година) имају резолуцију од 600dpi, а бољи штампачи од 1200 до 2700dpi. У зависности од технологије и намене штампачи могу имати и много веће резолуције. У свету видео уређаја, резолуција се нормално специфицира тако што се даје величина рама измерена у пикселима тј. пиксел димензијама. На пример, PAL рам је 768x576 пиксела, а NTSC рам је 640x480 пиксела [4]. Јасно је да ако се знају физичке димензије монитора, тј. ТВ екрана може се резолуција специфицирана на горњи начин транслирати у јединице изражене тачкама по јединици дужине. За видео је много природније да се резолуција изрази у виду броја пиксела јер је тај број непроменљив за одређени видео стандард и не зависи од физичких димензија екрана монитора. Исти резон важи и за дигиталне камере.
Рачунарски монитори су базирани на истој технологији као и видео (ТВ) монитори тако да је код њих резолуција изражена преко величине слике као што је то код ВГА 640x480 или 1024x768. Ипак, понекад се резолуција монитора изражава помоћу тачака по инчу, зато што се код рачунара појављује тенденција да се ова вредност фиксира. Као што је напоменуто, слика је дводимензионални вектор вредности пиксела тако да она свакако има димензије изражене у пикселима. Насупрот улазно излазним уређајима она нема физичких димензија. У одсуству додатних информација физичка величина слике када се она приказује зависи од резолуције урађаја на коме се приказује. На пример квадрат са страницом дужине 128 пиксела када се приказује на уређају резолуције од 72 dpi биће 45mm ширине. Иста слика приказана без скалирања на монитору са резолуцијом од 115 dpi биће око 28mm. По аналогији, иста слика одштампана на штампачу са 600dpi биће широка само 5mm. Генерално важи формула:

физичке димензије[inch] = **димензије у пикселима**[pix] / **резолуција уређаја**[pix/inch]

где се резолуција уређаја мери у пикселима по јединици дужине.(Ако је резолуција уређаја изражена у пикселима по инчима физичке димензије биће изражене у инчима).
Слике имају своје природне димензије. На пример: величина слике пре него што је скенирана, или величина канваса слике када се користи неки од графичких алата (Photoshop). Често се жели да се та природна величина слика задржи и да се она прикаже са тим димензијама на дисплеју монитора или пак штампачу. Другим речима, често постоји намера да се битмапирана слика прикаже у својој природној величини тј. да се не повећа или смањи услед утицаја резолуције уређаја. Ка том циљу, велики број формата фајлова меморише податак о тзв. оригиналној резолуцији слике. Та резолуција се изражава у пикселима по инчу (ppi) да би се разликовала од резолуције уређаја. Обично се та меморисана резолуција поклапа са резолуцијом уређаја на коме је слика направљена или дигитализована. На пример, ако је слика скенирана са 600dpi, меморисана резолуција слике биће 600ppi. Како су пиксели слике генерисани по тој резолуцији, физичке димензије слике се могу лако израчунати користећи димензије пиксела и резолуцију слике. Тада софтвер који је задужен за приказивање слике омогућава да се слика прикаже са оригиналним димензијама јер се она скалира са фактором који је одређен количником *резолуција уређаја/резолуција слике*.
То је јасније објашњено на следећем примеру. Слика димензија 6x4 инча се скенира уређајем од 600 dpi те ће њена резолуција бити 600 ppi. Дакле битмапирана слика ће бити димензија 3600x2400 пиксела. Приказана на 72 dpi монитору слика ће бити величине 50x33.3 инча. Да би била величине оригинала мора се пре приказивања скалирати фактором 72/600 = 0.12. У случају да је резолуција слике мања од резолуције уређаја на коме се она приказује слика се мора интерполирати са додатним пикселима.

Тај процес увек доноси губитак квалитета приказане слике. У супротном ситуација је различита. Наиме, ако је резолуција слике већа од резолуције уређаја на коме се она приказује онда се врши одбацивање сувишних пиксела да би се изједначио број пиксела слике и број пиксела који уређај може да подржи. То доводи до једног привидног парадокса, а то је да је субјективни квалитет такве слике увек бољи од квалитета слике која оригинално има исту резолуцију као и уређај на коме се приказује. То значи да ће, на пример, слика скенирана са 600dpi боље изгледати на дисплеју од 72dpi него слика приказана на истом дисплеју али претходно скенирана са 72dpi. То је зато што скенер врши одабирање и ако је његова резолуција мање онда су растојања између одабраних пиксела већа тако да се могу изгубити неки детаљи који постоје у оригиналној слици. Ако се скенирање врши са већом резолуцијом пиксели тако дигитализоване слике садрже већу количину информација него у претходном случају и те информације се у процесу пододабирања могу искористити. На пример боја пиксела се може одредити као просечна вредност суседних пиксела. То чини да су прелази боја у слици која се приказује много блажи, а линије мање изрецкане.

По дефиницији техника којом се слике генеришу са већом резолуцијом од резолуције уређаја на којој ће бити приказане назива се преодабирање (*oversampling*), а техника којом се резолуција слике која се приказује смањује на величину једнаку резолуцији уређаја на којој се приказује назива се пододабирање (*downsampling*). Слике које су добијене помоћу технике *oversampling/downsampling* су квалитетније једино ако софтвер који врши пододабирање користи додатне информације присутне у слици која је преодабирана. Интернет читачи (*Explorer* и *Netscape browsers*) нису добри у том послу тако да се пододабирање мора извршити претходно помоћу рецимо *Photoshop -a*. Подаци које се једном одбаце или занемаре више се никада не могу повратити. Следи закључак да је потребно задржати високу резолуцију битмапираних слика, и смањујући резолуцију само тада када је то потребно за њено приказивање на монитору. Ипак, мане високе резолуције су понекад велике. Слике велике резолуције садрже више пиксела и на тај начин ангажују више простора на диску и узимају више времена приликом трансфера преко рачунарске мреже. Величина слике расте са квадратом резолуције и то ствара проблеме тако да и поред предности у погледу квалитета, у пракси се тежи да слике буду оне резолуције које одговарају техничким карактеристикама уређаја на којима се меморишу, обрађују, приказују, преносе и користе. Као репер који се користи за одређивање довољног квалитета битмапиране слике односно величина резолуције је резолуција просечног монитора. Али чак и са резолуцијама од 72 и 96ppi фајлови слика су превелики за пренос преко мрежа. Да би се редуковала њихова величина, а да се при томе сачува довољан квалитет потребно је користити технике компресије података.

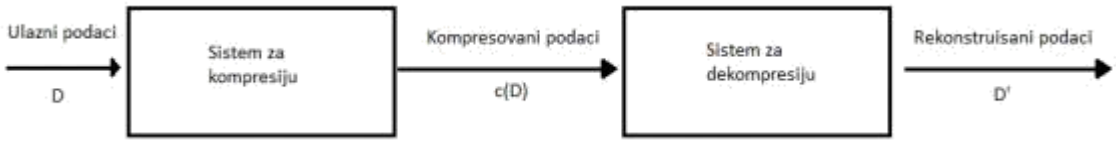
3. Компресија

У данашње време све су већи захтеви за преносом што веће количине података. То се покушава реализовати хардверским решењима, која ипак почињу да досежу своје реалне границе (ограничења која намећу саме технолошке могућности), па се прибегава софтверским решењима која теже да слику смање на што мању величину, да би се могло пренети више слика у јединици времена, преко истог ограниченог преносног система.

Слике се у дигиталном облику могу представити на више различитих начина, а један од најпростијих је да се вредности свих пиксела чувају у фајлу на основу неке конвенције (нпр. полазећи од пиксела који се налази у горњем левом углу слике). Последица коришћења овакве репрезентације слика је велики меморијски простор који је потребан за њихово чување. Да би се смањио меморијски простор за чување слика користе се технике компресије слика. Ове технике такође смањују трошкове комуникације приликом преноса велике количине података. Осим ових добрих особина, постоје и негативне стране компресије. Када се приступа неком податку код компресованих база података, понекад је потребно декомпресовати целу базу да бисмо дошли до жељеног податка. Таква ситуација се јавља код база чији се подаци ређе мењају (на пример, у статистици). Такође, уколико дође до грешке на једном биту компресованог кода, декодер може преостале битове да погрешно интерпретира чиме се долази до лоших података. У многим системима додатна комплексност приликом додавања процеса компресије може значајно да повећа трошкове система и умањи његову ефикасност.

Технике компресије слика су данас присутне у популарним апликацијама и уређајима као што су ДВД плејери или дигиталне камере. Добри алгоритми компресије слика су у могућности да слику у великој мери компресују, а да деградација слике буде минимална што се тиче визуелног квалитета.

У апстрактном смислу, компресију података можемо дефинисати као метод који од улазних података D генерише крађу репрезентацију тих података c(D), са мањим бројем битова у односу на D. Обрнути процес се назива декомпресија и у том процесу се од компресованих података c(D) генеришу или реконструишу подаци D’ (слика 17). Понекад се системи за компресију и декомпресију заједнички зову **CODEC** (*COmpression - DEcompression*)[5].



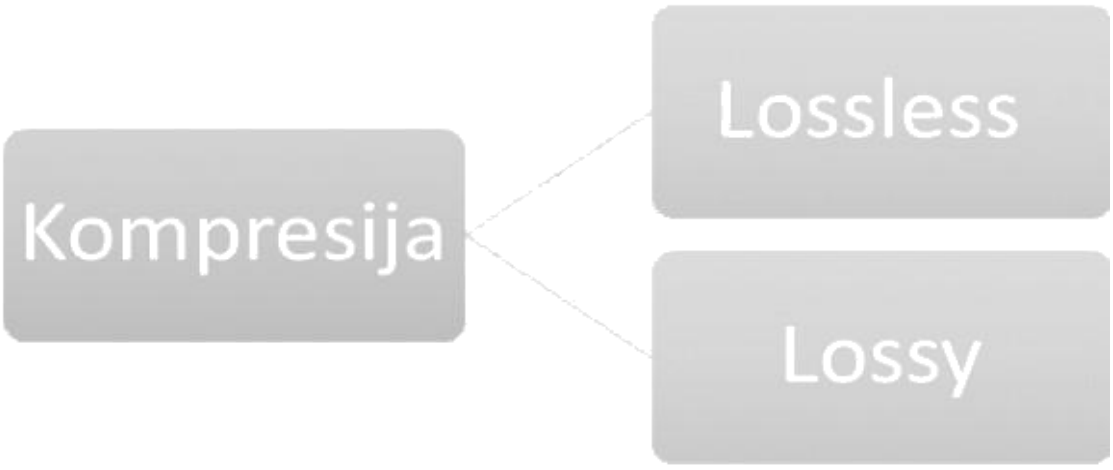
Слика 17: *CODEC*

Компресија података односи се на процес смањења количине података потребних за представљање дате количине информација. Имати на уму да подаци и информације нису исто. Подаци се односе на начин на који се информације чувају.
Различите количине података могу представљати исту количину информација. Понекад у оквиру података постоје делови који немају релевантне информације, или понављају познате информације. Стога се каже да постоји сувишност података – редунданција. Прекомерност података је централни концепт компресије слике и може се математички дефинисати.

Типови компресије

а) Компресија без губитака (*losseless*) - када су резултујући подаци D' у потпуности идентични у односу на оригиналне податке D;
б) Компресија са губицима (*lossy*) - када су резултујући подаци D' апроксимација оригиналних података D.

Алгоритми компресије без губитка обично користе статистичку редунданцију за приказивање податка без губитака информација, тако да је процес реверзибилан.
Компресија са губицима је начин сажимања података са унапред прихватљивим малим губицима. Користи се углавном код мултимедијалних података (звук и видео). Најпознатији формати су JPG (слике), MP3 (аудио) и MPEG (видео). Методе са губицима заснивају се на моделима људске перцепције (више компресују оне атрибуте слике који мање доприносе укупном изгледу). Оне узрокују деградацију слике у сваком кораку (сваким следећим кораком компресије/декомпресије слика се деградира), али најчешће омогућују далеко већи проценат компресије него методе без губитака.



Слика 18: *Типови компресије*

Предности компресије података:

- Потребно мање простора на диску;
- Брже писање и читање;
- Бржи пренос датотека;

Мане компресије података:

- Смањење квалитета;
- Променљиви стандарди.

3.1 Перформансе компресије

Типични модел за компресију података може да се опише следећим корацима:

- Одбацивање редундантних података;
- Редукција ентропије;
- Кодирање ентропије.

Редунданса у подацима може да се јави у више облика. На пример, суседни пиксели на слици су међусобно корелисани. Под корелацијом се подразумева да су вредности суседних пиксела веома сличне осим, на пример, у деловима слике који садрже наглу промену боје. Композиција речи у природном говору прати исти модел на основу одређене граматике, а такође се између речи праве паузе. У овим случајевима редунданса у подацима може да се смањи како би се постигла компресија.

Смањење редундансе у подацима обично се постиже трансформацијом података из једне репрезентацију у другу. Популарне технике смањивања редундансе су предикција узорка података на основу неког модела, трансформација из просторног у фреквентни домен као код дискретне косинусне трансформације (*DCT*), декомпозиција оригиналног скупа података у различите подгрупе као код дискретне вејвлет трансформације (*DWT*).

Код компресије без губитака овај корак је комплетно реверзибилан.

Следећи корак код компресије са губицима је да се смањи ентропија како би било потребно мање битова за меморисање или пренос података. То се постиже одбацивањем мање значајних информација на основу неких критеријума. Процес није реверзибилан јер није могуће потпуно повратити изгубљене податке помоћу инверзног процеса. Овај корак се примењује код компресије са губицима и завршава се обично применом неке од техника квантизације. Након тога, квантизирани коефицијенти се кодирају. С обзиром да је ентропија квантизираних података мања од ентропије оригиналних података, потребно је мање битова за њихово представљање чиме је постигнута компресија.

Основни проблем компресије слика је да се смањи количина података (битова) који су потребни за представљање слике. Математички гледано, компресија података представља трансформацију (кодирање) дводимензионалног низа пиксела у неки статистички некорелисани скуп података. У процесу декодирања реконструише се оригинална слика (код компресије без губитака) или се добија апроксимација оригиналне слике (код компресије са губицима).

У теорији информација, редундантност можемо дефинисати као разлику броја битова који су коришћени за пренос неке поруке и броја битова стварне информације у тој поруци. Процес компресије података елиминише нежељену редундансу. Ако исту информацију можемо репрезентовати на више начина, онда можемо да кажемо да она репрезентација која захтева више података садржи редундантне податке.
Да би разликовали податак од информације посматрамо следећи пример: исту информацију (број 4) можемо представити помоћу различитих репрезентација (у растућем редоследу у односу на број битова):

- Бинарни еквивалент број 4 (100): 3 бита;
- Римски број IV кодиран помоћу ASCI карактера за I и V: 16 битова;
- Реч ’quatro’ што значи ’четири’ на португалском језику, кодиран такође помоћу ASCI карактера: 48 битова;
- Некомпресована бинарна (црно-бела) слика димензије $64 \cdot 64$ на којој се налазе четири бела квадрата на црној позадини: 32768 битова.

Као и у многим другим системима, перформансе играју главну улогу у избору одређеног алгоритма компресије. Избор одговарајућег алгоритма компресије углавном зависи од потреба саме апликације: већа или мања компресија, субјективни и објективни квалитет реконструисаних података, сложеност алгоритма, брзина извршења.

3.1.1 Фактор компресије

Најпопуларнија метрика за одређивање перформанси алгоритма компресије је фактор компресије. Он представља однос броја битова оригиналних података и броја битова компресованих података. Постатрамо слику димензије $256 \cdot 256$ за чије је представљање потребно 65536 бајта, ако је за сваки пиксел потребно по један бајт. Ако је за представљање ове слике након компресије потребно 4096 бајта (16 пута мање од оригиналне), онда фактор компресије представљамо као 16 : 1.
Математичка дефиниција редундантности података дата је следећом формулом:

$$R = 1 - \frac{1}{CR}$$

где параметар **CR** представља **фактор компресије** и дефинише се као:

$$CR = \frac{b_1}{b_2}$$

b_1 и b_2 су бројеви битова у два различита скупа података који представљају исту информацију. Формула за редундантност података представља релативну редундансу првог скупа података (b_1). Кад је $b_2 = b_1$, тада је CR = 1 и R = 0, па можемо рећи да први скуп података (b_1) не садржи редундантне податке. Када је $b_2 << b_1$, онда CR → ∞ и R → 1, па добијамо високу редундансу и као последицу велику компресију података. Коначно, када је $b_2 >> b_1$, CR → 0 и R → −∞, можемо закључити да други скуп садржи много више података од првог и овај случај није пожељан у процесу компресије. CR и R узимају вредности из интервала [0, +∞] и [−∞, 1], респективно. Фактор компресије вредности 100 (или 100:1) значи да први скуп података има 100 јединица информације (битова) за сваку јединицу информације другог (компресованог) скупа. Одговарајућа редунданса (0.99 у овом случају) значи да је 99% података редундантно.

Једна варијација фактора компресије је просечан број битова по узорку. Ако је 65536 пиксела слике компресовано на 4096 бајта, онда кажемо да је компресија 0.5 бита по пикселу.

3.1.2 Мере компресије

Квалитет компресије може се представити на основу субјективних и објективних оцена. Најчешће се субјективна оцена дефинише преко просечне оцене посматрача (*mean observer score - MOS*). Постоје различити статистички начини да се израчуна MOS, а један од најпростијих је да се случајно изаберу статистички значајан број посматрача који би визуелно оценили квалитет компресије. На пример, те оцене могу бити од 1 до 5, при чему 5 представља најбољи а 1 најгори квалитет. Све оцене се сабирају а потом деле са бројем узорака (посматрача) чиме се добија субјективна оцена компресије. Оштећења слике која су видљива након декомпресије називају се артифакти компресије.

Не постоји универзално прихваћена објективна оцена квалитета компресије, али најчешће се користи средња квадратна грешка (*Root Mean Square - RMS*).

Нека је $f(x, y)$ оригинална а $f'(x, y)$ реконструисана слика. За све x и y, грешка $e(x, y)$ између $f'(x, y)$ и $f(x, y)$ је:

$$e(x, y) = f'(x, y) - f(x, y)$$

а укупна грешка између слика (димензија M · N):

$$E = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f'(x, y) - f(x, y)]$$

RMS грешка (e_{RMS}) између *f '(x,y)* и *f(x,y)* се рачуна на следећи начин:

$$e_{RMS} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f'(x,y) - f(x,y)]^2}$$

Код компресије слика RMS се често представља и преко еквивалентне реципрочне вредности

- PSNR (*Peak Signal to Noise Ratio*) дефинисане са:

$$PSNR = 10log_{10}\frac{(2^B - 1)^2}{e_{RMS}}$$

где B представља број битова по пикселу, па у случају кад је сваки пиксел представљен са 8 битова добијамо:

$$PSNR = 10log_{10}\frac{255^2}{e_{RMS}} = 20log_{10}\frac{255}{\sqrt{e_{RMS}}} = 20log_{10}255 - 10log_{10}e_{RMS}$$

PSNR се изражава у децибелима (dB), што је PSNR вредност већа, то је реконструисана слика боља у погледу визуелног квалитета. Обично добро реконструисане слике имају вредност PSNR од 30dB и више.

4. Компресија са губицима

Компресија са губицима значи да се неки подаци изгубе приликом декомпресије. Компресија са губицима темељи се на претпоставци да тренутне датотеке података чувају више информација него што их људска бића могу уочити. Дакле, небитни подаци се могу уклонити.

Алгоритми компресија са губицима жртвују неке податке у фајлу, тако да се по декомпресији добија фајл који није једнак оригиналу. Већ је поменуто да смештање на диск и пренос кроз комуникационе канале звука, слика и покретних слика захтева много веће степене компресије од оних који се могу добити код компресије без губитака, и да се управо за те потребе користе компресије са губицима. Та идеја је наизглед лоша, али се показује да није тако, и долази се до интересантних закључака. Ако се ради о музици или слици, онда промена од једног малог дела процента неће да буде ни приметна, а ако и буде приметна неће да буде значајна за наша чула. Музика и слика, а самим тим и филм, јер он представља низ статичних слика, су информације које не морају да буду 100% верно репродуковане. Са друге стране, то су информације које теку у времену, тако да је јако тешко приметити шта се догодило. Требало је само уочити да код ових информација постоје велике резерве и да губитке можемо дозволити. Још једна срећна околност је да се показује да се могу постићи велики степени компресије ако дозволимо јако мало губитака.

Као што је већ истакнуто, значај ових алгоритама је огроман, јер су они покренули експоненцијални развитак дигиталног звука, слике и филма. Представник револуције дигиталног звука је MP3 формат, који је већ сада поприлично застарео, није ни најсавршенији, али је он ту револуцију направио. Помоћу MP3 формата на један CD стаје око петнаест албума и они могу да се размењују кроз комуникационе канале. Код компресије слика захтеви за сабијањем података су још израженији него код звука. Овде је измишљен формат JPEG за компресију дигиталних слика. Код компресије покретних слика направљен је формат MPEG, који представља неку варијанту JPEG-а.

4.1 JPEG стандард

Значај дигиталних слика као компјутерских података препознат је од стране интернационалних организација за стандарде ISO И ITU-T (CCITT). Сарадњом ове две организације створен је JPEG (енгл. *Joint Photographics Experts Group*) [6] стандард за компресију дигиталних слика.

JPEG стандард укључује четири основне методе за компресију.
Прве три се темеље на дискретној косинусној трансформацији (DCT) и спадају у компресију са губицима - "*losse*". То значи да декомпресована слика није идентична оригиналу – слици коју смо компресовали. Помоћу ових метода може се достићи веома висок ниво компресије, а алгоритам је осмишљен тако да искористи позната ограничења људског ока да слабије уочава разлике у нијансама боје, него у интензитету светлости. JPEG стандард специфицира да се однос између нивоа компресије и веродостојности приказа компресоване слике може подешавати помоћу одговарајућих параметара алгоритма.

Четврта метода за компресију коју JPEG стандард укључује се темељи на предиктивном кодирању и спада у компресију без губитака - *"lossless"*. То значи да је декомпресована слика идентична оригиналу. Код ове методе вредност следећег пиксела се одређује као аритметичка средина суседних пиксела, при чему се памти само разлика између тренутне и прорачунате вредности пиксела. Не постоји јединствен стандард за JPEG алгоритам, већ скуп правила и препорука са много подесивих параметара.

JPEG је развијен на основу следећих захтева:

1. Мора да задовољава широки распон квалитета слике и мора да обезбеди могућност одређивања жељеног односа компресија/квалитет;

2. Мора да буде применљив на практично све типове дигиталних слика тј. не сме да буде ограничен резолуцијом слике, системом презентације боја, комплексношћу сцене, итд;

3. Не сме имати превелику комплексност израчунавања, како би могао да се уз прихватљиве перфомансе имплементира на што већи број процесора. Након исцрпног тестирања, методе базиране на дискретној косинусној трансформацији (DCT) показале су се као најбоље;

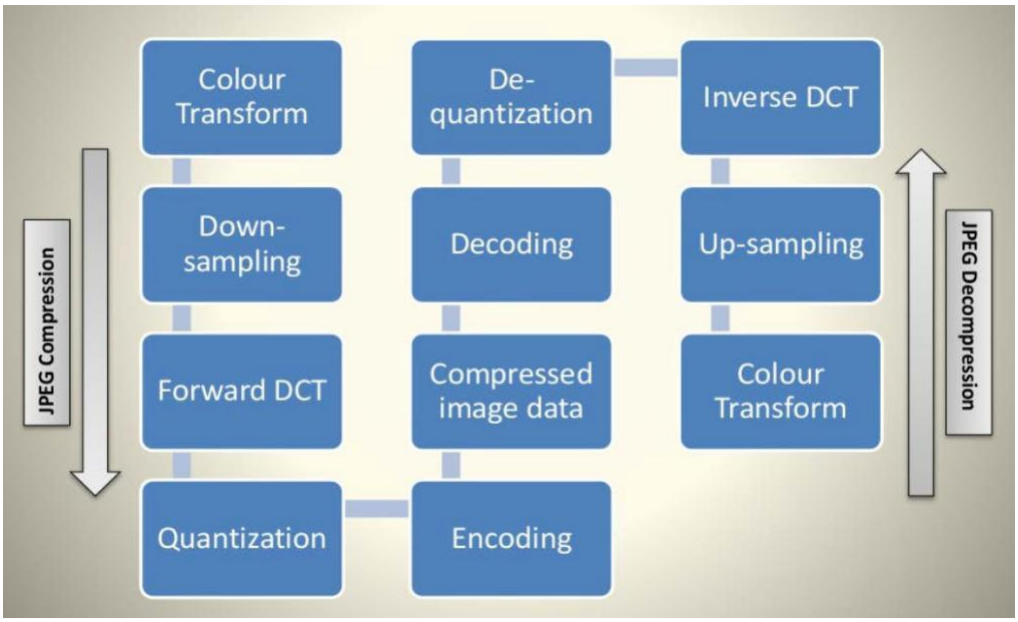
4. Мора да обезбеди следеће начине рада :

- секвенцијално кодирање – кодирање преласком по слици ред по ред;
- прогресивно кодирање – тако кодирана слика се код декодирања гради вишеструким грубо-на-фино прелазима тако да се квалитетет слике побољшава;
- кодирање без губитка информација, без обзира на степен компресије;
- хијерархијско кодирање – слика је кодирана на више резолуција тако да је могуће декодирати слику у нижој резолуцији, без да је претходно декодирамо у пуној резолуцији.

4.1.1 ЈПЕГ алгоритам

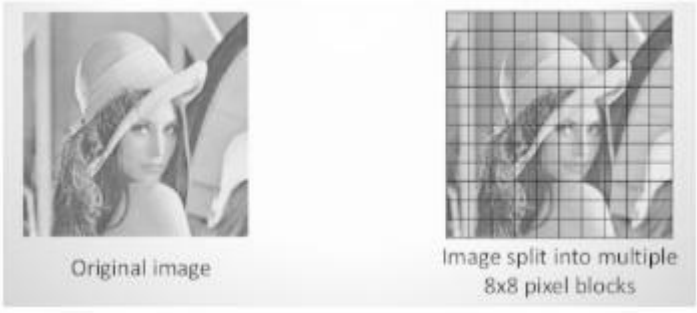
JPEG је као стандард објављен 1994. године. JPEG кодирање се састоји из следећих корака:

1. Splitting – дељење слике; Слика се дели на блокове димензије 8 · 8;
2. Оригинална слика у боји (RGB) се конвертује у алтернативни модел боја (YCbCr) и информација о бојама се дели на узорке;
3. DCT (*Discrete Cosine Transformation*) - Дискретна косинусна трансформација
4. Квантизација
5. Кодирање ентропије (*Entropy coding*)



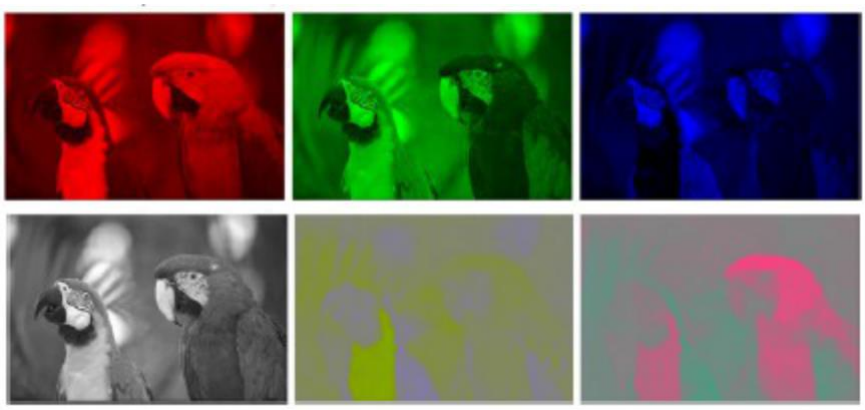
Слика 20: *Шема процеса JPEG компресије[7]*

Splitting – слика се дели у блокове пиксела димензије 8x8



Слика 21: *Дељење слике у блокове пиксела [8]*

Конверзија RGB у YCbCr - ЈПЕГ при компресији користи YcbCr [9] простор боја. Ако знамо да је РГБ [10] такође простор боја који се састоји од различитих нијанси црвене, зелене и плаве боје, исту логику можемо примиенити и за YCbCr простор боја, али помоћу друге три различите компоненте. **Y** представља осветљење (*brightness*) пиксела и назива се још и „*luminance“*. **Cb** представља разлику између компоненте плаве боје и луминансе, а **Cr** представља разлику црвене боје и луминансе. Због густине рецептора за осветљење и боју у оку, људско око је осетљивије на промену осветљења (Y компонента) него на промену боја (Cb и Cr компоненте).



Слика 22: *RGB компоненте (горе), YCbCr компоненте (доле) [11]*

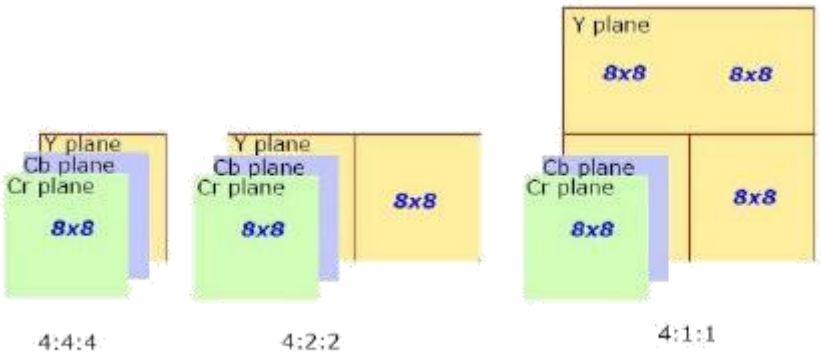
Конверзија користи РГБ улазне вредности за сваку компоненту опсега [0.0, 255.0] и трансформише у YCbCr где компоненте имају опсеге:
Y [0.0, 255.0], Cb [-128.0, 127.0], Cr[-128.0, 127.0].
Једначина матрице конверзије је приказана на следећој слици (слика 23).

$$[Y \quad Cb \quad Cr] = [R \quad G \quad B] \begin{bmatrix} 0.299 & -0.168935 & 0.499813 \\ 0.587 & -0.331665 & -0.418531 \\ 0.114 & 0.50059 & -0.081282 \end{bmatrix}$$

Слика 23: *RGB-YCbCr матрица конверзије [12]*

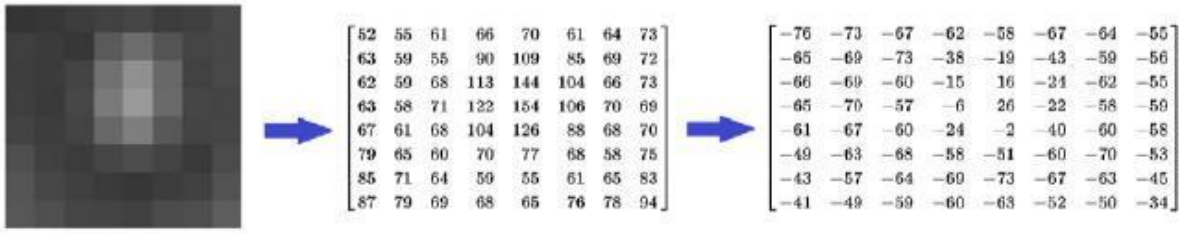
Ова конверзија омогућава нам да ефективно извршавамо процес *downsampling*-а – смањење дубине боје слика. Помоћу *downsampling*-а користимо претходно напоменуту осетљивост односно мањак осетљивости људског ока на плаве боје тако да се сликама конвертованим у YCbCr простор боја смањује дубина боје Cr и Cb компонената. Дакле сваки пиксел задржава вредност своје светлине (Y), док се остали пиксели (CbCr) групишу и у складу с одређеним правилима групе попримају једнаку боју. Укратко, Y вредност се узима за сваки пиксел, а Cb

и Cr вредности се узимају за блок пиксела чија величина зависи од MCU формата. MCU – *Minimal Coded Unit* представља најмању јединицу која се може кодирати.



Слика 24: *Структура МЦУ за ЈПЕГ формат [13]*

Дискретна косинусна трансформација (*Discrete Cosine Transformation)* – претвара информације спремљене у блоку (8 x 8) пиксела из просторног домена у фреквенцијски домен. Будући да људско око није осетљиво на компоненте високих фреквенција, такве компоненте се могу третирати као да су редундантне, непотребне. Како би се те компоненте уклониле потребно је слику претворити у фреквенцијски домен што је главни циљ ДЦТ-а. Како би се уопште могао започети процес, прво је потребно центрирати вредности око нуле како би се подударало са функцијом косинуса, тај процес се назива „*Zero-shift*“. Као пример на слици приказан је 8 x 8 блок пиксела из Y компоненте чији је иницијални опсег [0, 255] на којем се примењује „*Zero-shift*“; нови опсег је [-128, 127] те све вредности се умање за 128.



Слика 25: *Zero-shift [14]*

Будући да је матрица слике дводимензионална, потребно је користити стандардизовану 2-Д ДЦТ формулу за ЈПЕГ компресију. Ако је улазна матрица **P(x,y)** а трансформисана матрица **G(u,v)**, ДЦТ формула за блокове од 8x8 пиксела гласи:

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Трансформисана матрица приказана је на слици 26. Све вредности трансформисане матрице сада представљају „тежине“ (*weights*) које одређују која је у коначном резултату улога одређене косинусне функције.

$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.56 \\ -48.53 & 12.07 & 34.10 & -14.76 & -1.024 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Слика 26: *Трансформисана матрица [15]*

Квантизација – је процес где се скуп вредности мапира у мањи скуп вредности како би се уштедео простор. То је у принципу врло једноставан процес, а постиже се тако што се матрице које су добијене ДЦТ трансформацијом поделе са стандардизованим квантизацијским ЈПЕГ матрицама које дефинишу која тежина треба бити подељена са којим бројем. Те матрице су дизајниране да поделе тежине виших фреквенција с већим бројевима, јер више фреквенције имају мањи утицај на коначни резултат.

Као пример је приказан процес квантизације матрице добијене у претходном примеру ДЦТ трансформације, а затим се дели са ЈПЕГ квантизацијском матрицом са подешавањем квалитета од 50%, па се резултати заокругљују на најближи цео број. На пример, трансформисани елемент матрице [-61.20] се дели са одговарајућим елементом матрице квантизације (10) и као резултат се добије одговарајући коначни, квантизовани елемент матрице. [-61.20] / 10 = -6.12 -> -6. Исто важи и за све остале елементе матрице.

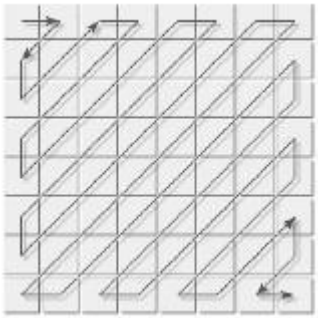
$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.56 \\ -48.53 & 12.07 & 34.10 & -14.76 & -1.024 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} /$$

$$\begin{bmatrix} 15 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 25 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 54 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Слика 27: *Квантизација [16]*

Свака поставка квалитета има различиту табелу квантизације, што је бољи квалитет, то су мањи бројеви у табели квантизације са којом делимо.

Кодирање ентропије - за разлику од осталих корака, кодирање ентропије је без губитака. Процес кодирања започиње тако што се коефицијенти матрице посматрају у цик-цак (*zig-zag)* редоследу, а потом се примењује РЛЕ (*Run Length Algorithm*) алгоритам. Овај алгоритам здружује блиске вредности коефицијената, коефицијенти различити од 0 се кодирају помоћу Хофмановог кода. Као алтернатива Хофмановом кодирању може да се користи аритметичко кодирање. Кодирање и декодирање аритметичким кодом је спорије у односу на Хофманов код, али кодиране слике су обично 5-7% мање у погледу меморије која је потребна за њихово чување.



Слика 28: *Цик-цак образац*

5. Компресија без губитака

Код компресије без губитака добија се фајл који ће после декомпресије бити идентичан оригиналу. Важно је поменути значење формата фајла јер је у вези са алгоритмима компресије. Под форматом фајла подразумева се начин на који се подаци организују у фајлу. Када неке податке снимимо у одређеном формату фајла најчешће се изврши компресија података алгоритмом компресије који тај формат подржава. Компресија података се не мора извршити уколико формат подржава снимање фајла у некомпресованом облику. Један формат често подржава више алгоритама компресије.

Развојем алгоритама компресије, осамдестих година прошлог века, појавио се програм ZIP који је вршио сабијање података на диску. ZIP програм је и данас јако популаран и користи се за компресовање података типа писаних докумената, где губици нису дозвољени. На пример, код финансијских докумената или програма, промена од само једног бита учинила би те документе неупотребљивим. Поред ZIP-а познати формати компресије без губитака су BZIP2 и RAR. Већина програма за компресовање података подржава више алгоритама компресије с различитим брзинама и степенима компресије. ZIP и RAR датотеке обично су компресоване Дефлате алгоритмом који је изведен из ЛЗW (*Lempel Ziv Welch*) алгоритма.

Данас постоји јако много алгоритама компресије без губитака. У овом тексту детаљније ће бити описан алгоритам који је у широј употреби. Он је старији и теоретски више коришћен, зове се Хофманов кôд.

5.1 Компресија слика алгоритмима без губитака и формати графичког фајла

Компресија слике без губитака омогућује да се приликом њене декомпресије добије слика идентична слици пре компресије. Квалитет слике је у овом случају у потпуности сачуван. Ови алгоритми компресије се обично користе код слика које садрже линије оштрих ивица, као што су технички цртежи, мапе, текст фајлови итд. У случају великих губитака информација о слици, оштре линије би могле постати нејасне. Затим, код слика чији је садржај од великог значаја, као што су нпр. слике направљене у медицинске сврхе.

Алгоритми који се често користе код компресија слика без губитака су *run_length* кодирање (RLE) и *Хофмановo* кодирање. РЛЕ је метод компресије без губитака, који конвертује узастопне идентичне карактере у кôд који се састоји из карактера и броја који означава дужину низа (run). Што је низ дужи, компресија је већа. Метод РЛЕ зато најбоље резултате даје у компресији црно-беле графике и једноставних цртежа. Предност компресије слика алгоритмима без губитака је што су компресија и декомпресија једноставне и брзе, али се њима не постиже пуни ниво компресије, који је иначе могућ. На пример, *run-length* кодирање користи само сличност пиксела по хоризонтали (хоризонтална кохеренција), а не и по вертикали.

Под форматом графичког фајла подразумевамо начин на који се информација о слици организује у фајлу. Корисник се опредељује за формат у зависности од намене и садржаја фајла, оперативног система, софтвера са којим располаже итд. Данас постоји велики број формата за битмапирану графику. Намене ових формата су разноврсне. Неки су погодни за обраду слика, а неки за њихово архивирање и приказ на Вебу. Неки су присутни на разним платформама, док су неки уско специјализовани за посебне намене, и постоје само на одређеним оперативним системима. Најчешће, графички формат у својој спецификацији има наведен алгоритам или неколико алгоритама компресије које подржава, тако да се може рећи да формат фајла и алгоритам компресије обично чине нераскидиву целину.

Графички формати *PCX, TIFF* и *BMP* су широко заступљени у обради слика, укључујући скенирање, пренос међу платформама и њихово коришћење у стоном издаваштву. Ова три формата садрже податке који су или некомпресовани, или се компресују без губитака, што их чини добрим при едитовању, али их дисквалификује за коришћење на Вебу.

ПЦХ (енгл. PCX**)** је један од најстаријих графичких формата. Појавио се раних осамдесетих година прошлог века. И данас је један од највише коришћених формата на PC рачунарима. Препознају га практично сви икада написани графички програми. PCX фајлови могу чувати податке о сликама са дубином пиксела 1, 4, 8 и 24 бита. Подаци су увек компресовани. Алгоритам компресије је РЛЕ (**енгл**. *Run Length Encoding*). Екстензија овог формата је PCX или PCC.

ТИФФ (**енгл.** TIFF) је један од најшире подржаних графичких формата за чување битмапираних слика на персоналним рачунарима. Сматра се поузданијим форматом од PCX-а, и има могућност коришћења бољих метода компресије од њега, па су фајлови нешто мањи. Неке од метода компресије које овај формат подржава су PackBits, RLE, LZW, ZIP, JPEG, а може бити и без компресије. Екстензија овог формата је tif или TIFF. Чињеница да TIFF формат подржава практично све дубине пиксела и велики број алгоритама компресије, са друге стране, представља иману овог формата: различити програми који раде са TIFF фајловима су од ових његових широких могућности прихватили само неке. Тако, TIFF фајл направљен у једном графичком програму често не може бити препознат у другом.

БМП (енгл. BMP**)** је стандардни формат за битмапирану графику коришћен у *Windows* -у. Подржава дубине пиксела 1, 4, 8 и 24 бита. Подржава РЛЕ алгоритам компресије података за слике са 4 или 8 бита по пикселу, а екстензија овог формата је bmp.

БМП је некомпресовани сликовни формат датотеке у коме је слика разложена на ситне квадратиће, тзв. пикселе. Приликом дигитализације слике, сваком пикселу се дефинише положај по хоризонтали и вертикали (x и y), нијанса боје и интензитет осветљености. Скуп ових података уз податке о маргинама, димензијама приказа и др. чини сликовну датотеку с наставком .бмп.

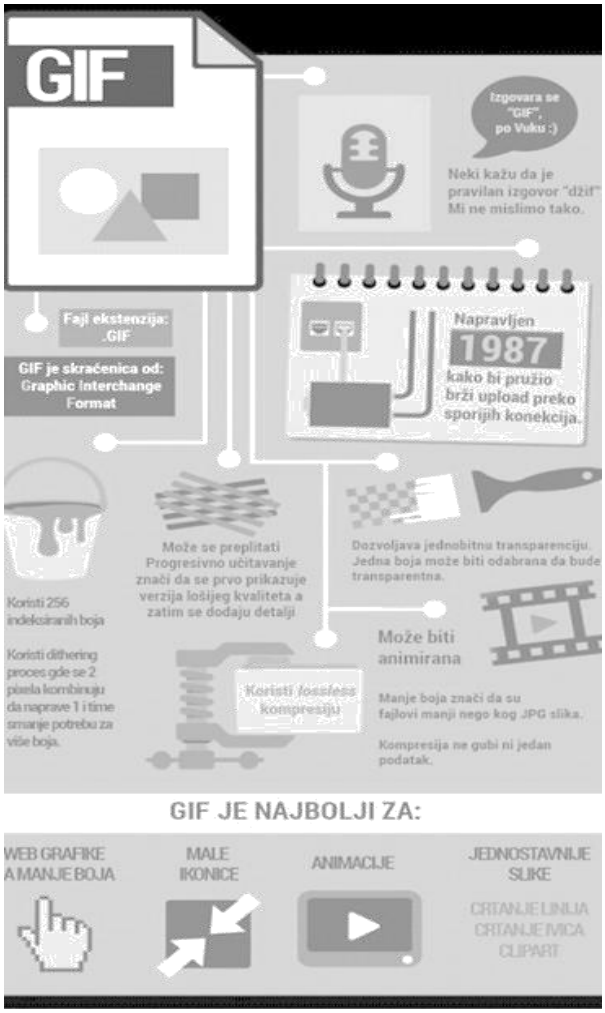
Услед чињенице да се за сваки пиксел чува неколико података, још једном:

- положај по хоризонтали и вертикали (x и y);
- нијанса боје и интензитет осветљености.

следи да слике у БМП сликовном формату испадају јако велике, јер се нпр. само за боју у систему 24 битне боје за сваки пиксел требају три октета/бајта, плус положај и интензитет осветљености.

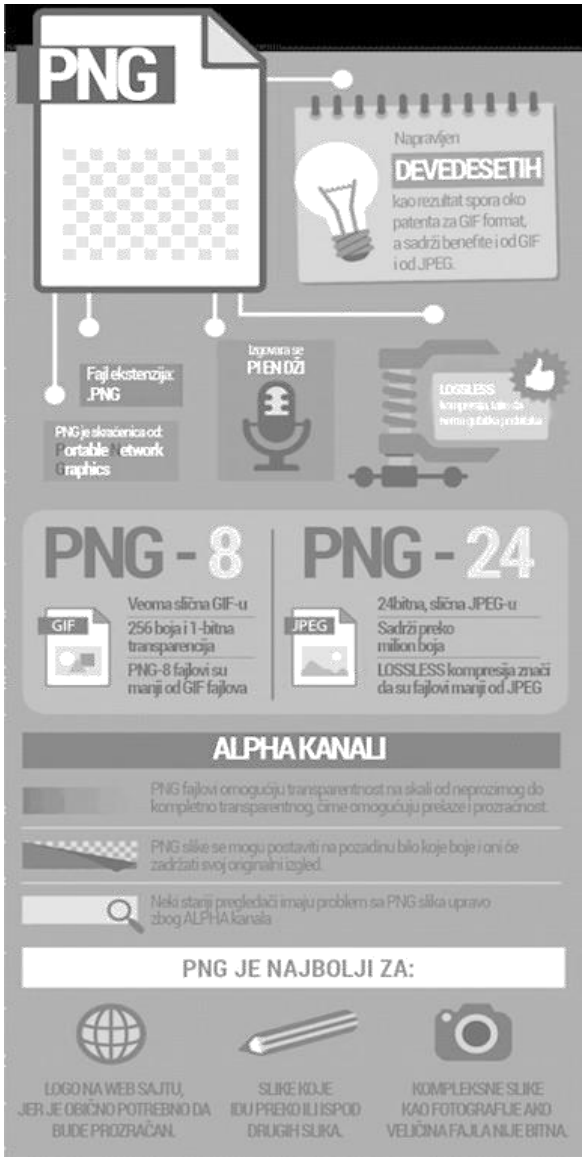
Значајно је поменути и формате **GIF** и **PNG** који су намењени коришћењу на Вебу, јер, захваљујући оптимизованим алгоритмима компресије који се у њима користе, троше мање простора за податке о сликама, па се лакше шаљу преко мреже. Ови формати имају могућност прогресивног приказа, тј. стварања илузије бржег приказивања слике.

ГИФ (**енгл.** GIF) је стар формат, његова екстензија је gif, и он је и данас популаран за приказ једноставних слика на Вебу. GIF слике су дубине пиксела 8 бита и увек су компресоване. Метод компресије је LZW. Ово јесте алгоритам компресије без губитака, али одлично компресује једноставне слике са великим областима обојеним истом бојом. ГИФ формат је добар избор за цртеже, црно-беле слике и за ситан текст. Због мале дубине пиксела, а и због природе LZW алгоритма (који не компресује добро слике са непрекидним тоновима) није добар за приказ фотографија.



Слика 29: *Инфографик ГИФ*

ПНГ (енгл. PNG) је нови формат за битмапирану графику, има екстензију png и настао је у покушају да се превазиђу законски проблеми везани за коришћење GIF-а (односно, LZW алгоритма). Формат PNG је потпуно патентно и лиценцно неоптерећен. Свако може бесплатно креирати софтвер који ради са PNG сликама. Ово је графички стандард предвиђен да се користи на Вебу. Мада PNG омогућује приказ слика са различитом дубином пиксела (до 48 бита по пикселу). PNG је, осим за приказ слика на Вебу, добро решење и за обраду слика, јер се његови подаци компресују без губитака.



Слика 30: *Инфографик ПНГ*

PNG компресија је међу најбољима које постоје без губитака информације. PNG користи Дефлате метод компресије, метод који се користи и код *pkzip-a* [17]. Дефлате је побољшана верзија LZW алгоритма компресије. Ради слично LZW алгоритму, тј. прати понављање хоризонталних узорака у свакој линији. Побољшање у односу на компресију присутну код GIF-а је у истовременој контроли вертикалних узорака. На тај начин се постиже нешто већи степен компресије.

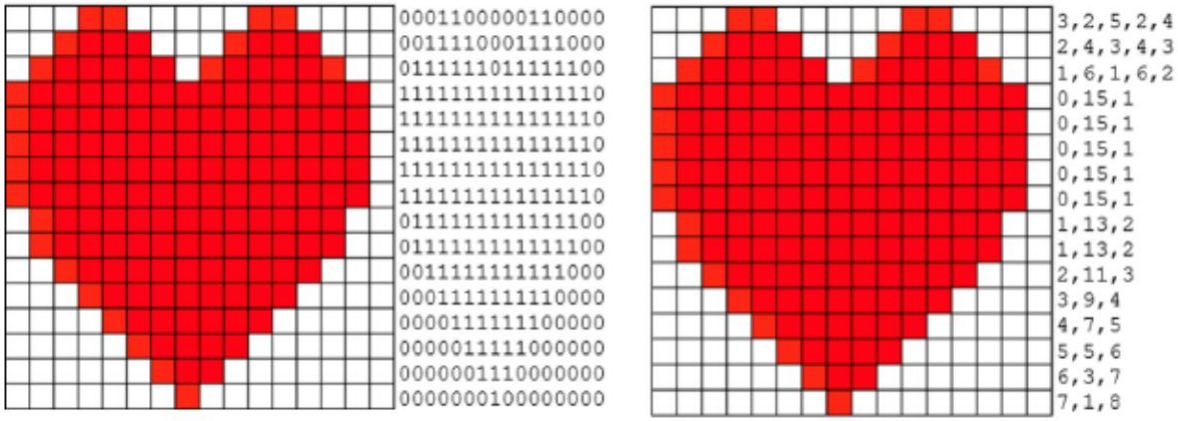
6. Run – length Encoding компресија слика

Идеја РЛЕ (**енгл.** *Run-length Encoding*) компресије над сликама је да ако се насумично одабере један пиксел на слици, постоји велика шанса да суседни пиксели имају исту боју.

Да би се применила РЛЕ компресија на слике, мора да се претпостави да су пиксели слике ускладиштени у низу који се зове битмапа у меморији. Пиксели у битмапи су углавном поређани у линијама скенирања тако да је први пиксел у битмапи онај у горњем левом углу слике, а последњи у доњем десном углу слике.

Принцип РЛЕ компресије је исти као и код компресије текста, али у овом случају постоје додатни детаљи. На пример, покушати компресију слике која се састоји само од црно-белих пиксела. При компресији се битмапа скенира ред по ред за низове пиксела исте боје. Ако битмапа почиње низом од 20 белих пиксела, затим 7 црних пиксела, па 65 белих, будући да компресор зна да има посла са само две боје, као излаз потребно је само исписати бројеве 20,7,65. У таквим случајевима компресор увек претпоставља да је први пиксел беле боје, па ако битмапа започне црним пикселом, излазни ток започиње нулом (0 белих пиксела на почетку битмапе).

Приказана је слика пре и након РЛЕ компресије. (слика 31); уместо црне користи се црвена боја која зато узима бинарну вредност 1.



Слика 31: *Приказ слике пре РЛЕ компресије (лево) и након РЛЕ компресије (десно)[18]*

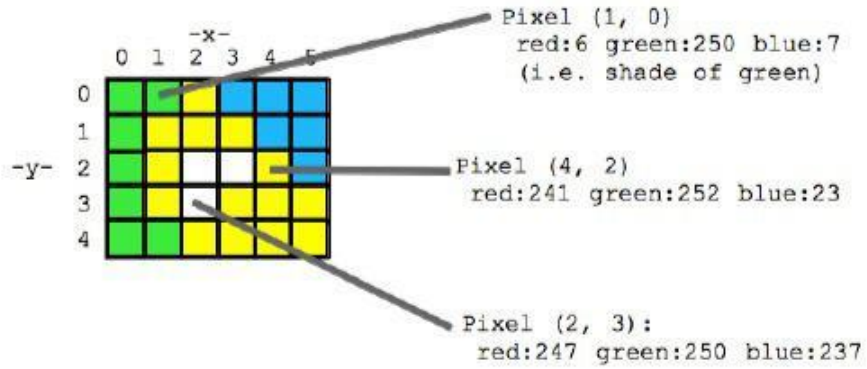
Сваки низ пиксела истог интензитета (исте нијансе) кодиран је као пар (дужина низа, вредност пиксела). Дужина низа обично заузима један бајт који омогућава низове дужине до 255 пиксела, док вредност пиксела заузима више битова, у зависности од нијансе боје.
На пример, ако се посматра следећа секвенца:

14, 25, 25, 25, 25, 25, 25, 56, 77, 78, 43, 43, 43, 43, 43, 43, 43

Будући да компресија слике у више нијанси не садржи бинарне вредности, сложеност се повећава и важно је разликовати који бројеви представљају вредност пиксела, а који дужину низа. Ако је број нијанси 256, тај број можемо смањити на 255 и једну вредност користити као „заставицу“ која ће бити показна за почетак низа. Ако вредност заставе поставимо на 255, претходни низ ће изгледати овако:
14, 255, 6, 25, 56, 77, 78, 255, 7, 43

Дакле, први пут када се наиђе на заставицу знамо да следеће две цифре представљају дужину низа и вредност пиксела од којих се тај низ састоји.

У сликама у боји уобичајено је да сваки пиксел тежи три бајта, по један за сваку нијансу црвене, зелене и плаве боје, тако да један пиксел може да се прикаже вредностима (45, 76, 123).



Слика 32: *РГБ вредности пиксела[19]*

Да би се извршила компресија на таквим сликама, потребно је да се ефикасно створе три слике од једне – по једна за нијансе црвене, зелене и плаве, које ће бити посебно кодиране. На пример, низ следећих пиксела:

(45, 76, 123), (45, 78, 123), (46, 79, 123), (46, 82, 123)

постаће три различите секвенце:

Р = (45, 45, 46, 46)
Г = (76, 78, 79, 82)
Б = (123, 123, 123, 123)

Предност РЛЕ алгоритма је у томе што је врло једноставан за имплементацију, али недостатак је чињеница да је алгоритам све мање ефикасан што је слика сложенија. Стога је овај алгоритам користан за једноставне слике на којима се појављују само црна и бела боја или обично слике које се састоје од великих монохроматских простора (линијске слике, архитектонски дизајн, итд.).

7. Хофманов алгоритам за компресију слика без губитака

Најпопуларнија техника за уклањање сувишности приликом кодирања је Хофманово кодирање. Хофманово кодирање је префиксно кодирање промењиве дужине низа без губитака названо по Д. А. Хофману који је осмислио алгоритам 1952. године. Појединачни знакови кодирају се кодним речима у зависности од вероватноће њиховог појављивања. На тај начин уклања се статистички вишак (редунданција) у низу знакова. До данас је алгоритам претрпео многе промене.

Хофманов алгоритам користи јединствену методу за одабир репрезентација сваког симбола из чега је изведен алгоритам без префикса, што значи да ниједна кодна реч није префикс друге. Због тога није потребан сепаратор између кодних речи. Заснован је на следећим двема чињеницама. Симболи који се најчешће појављују могу се доделити краћим кодним речима од симбола који се ређе појављују. Два симбола која се најређе појављују имаће кодне речи исте дужине, с разликом само у најмање значајном биту.

Хофманово кодирање слике заснива се на статистичким својствима слике. Дигитална мирна слика *f(i,j)* може да се прикаже матрицом A са елементима $a_{i,j}$. Елементи матрице су елементи слике. Нека је скуп S = {α_1, α_2,...,α_k} скуп могућих нивоа светлости који се могу појавити у слици. Вероватноћа да поједини елемент слике $a_{i,j}$ поприми дискретну вредност $\alpha_k \in$ S функција је P($a_{i,j}$). Функција густине вероватноће дискретне случајне промењиве $a_{i,j}$ је:

$$p_k = P(\alpha_{i,j} = \alpha_k) := \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_{i,j} \qquad (7.4)$$

где *m* означава висину слике, а *n* означава ширину слике. Вероватноћа појављивања дискретне вредности α_k изражава се као однос елемената слике које добијају вредност α_k и укупног броја елемената слике. Садржај информација је дефинише се изразом:

$$I(\alpha_k) := log_2 \frac{1}{P(\alpha_k)} = -log_2 P(\alpha_k) \qquad (7.5)$$

где је тежина или количина корисне информације у битовима, које носи одговарајући елемент слике. То значи да нису сви елементи слике подједнако важни. Будући да важе следеће релације

$$0 \leq P(\alpha_k) = p_k \leq 1 \qquad (7.6)$$
$$log_2 P(\alpha_k) \leq 0 \qquad (7.7)$$

следи

$$-log_2 P(\alpha_k) \geq 0 \qquad (7.8)$$

Својство логаритамске функције је да логаритам веће вероватноће даје мањи број у апсолутној вредности од логаритма мање вероватноће. Из тога следи да ће елементи слике који попримају дискретну вредност веће вероватноће садржати мање корисне информације. Такви елементи слике могу се грубо квантизовати. Кодирање ентропије делује на то. Просечна количина корисних информација коју носе сви елементи слике назива се ентропија и може се дефинисати као:

$$H(A) \coloneqq \sum_{k=1}^{S} P(\alpha_k) I(\alpha_k) = -\sum_{k=1}^{S} P(\alpha_k) log_2 P(\alpha_k) \qquad (7.9)$$

Када сви елементи слике имају једнаку вредност α_k , тада према (7.4) и (7.5) произилази:

$$P(\alpha_{i,j} = \alpha_k) = 1 \qquad (7.10)$$
$$I(\alpha_k) = 0 \qquad (7.11)$$

из тога, према (7.9) следи:

$$H_{max}(A) = log_2 S \qquad (7.12)$$

где је S=2n – 1 тј. максималан број употребљених нивоа квантизације. Код постојања максималне ентропије, сваки елемент слике има различиту дискретну вредност, односно потребно је најмање n битова да би се приказале све дискретне вредности. Статистичка редунданција се може изразити као:

$$R(A) \coloneqq log_2 S - H_{max}(A) \qquad (7.13)$$

Претходни израз наводи на закључак да већа ентропија значи мање редундантности.

Хофманов алгоритам је врло једноставан и најлакше га је описати као такозвани похлепни алгоритам, односно као алгоритам који на сваком кораку бира најбоље локално решење, у нади да ће пронаћи глобални оптимум. Хофманов алгоритам генерише код у структури бинарног стабла минималне просечне дужине путање са листе учестаности појављивања. Списак фреквенција појављивања састоји се од вероватноће појављивања нивоа осветљености на слици.

На врху бинарног стабла налази се основни чвор који се назива корен. Он се дели на максимално два чвора који се зову деца и сваки следећи чвор може имати највише двоје деце. Дубина стабла једнака је максималном нивоу неког чвора у стаблу. Основни ниво је 1.

Прво треба да се направи хистограм слике која треба да се компресује. Сваки ниво осветљености повезан је са учестаношћу појављивања те вредности на слици. Нивои осветљености који се најчешће појављују имају највише фреквенције. Нивои осветљености који се најређе јављају имају најниже фреквенције [20].

Почиње се са онолико стабала колико је нивоа осветљености које је потребно кодирати. Свако стабло садржи по један чвор са фреквенцијом нивоа осветљености листа. На тај начин се ствара почетна „шума“ стабала са једним чвором.

Следеће ће се понављати док не остане само једно стабло. Одаберу се два стабла са најмањом учестаношћу појављивања. Спојити их у једно стабло додавањем новог корена са вероватноћом појављивања једнаком збиру учестаности појављивања претходна два стабла, при чему ова два стабла постају чворови (деца) новог стабла. Није важно који је леви или десни чвор (дете). Често се корен ниже фреквенције ставља лево.

Вредност један и нула додељују се сваком од парова хоризонталног стабла. Да би се добио код за сваки ниво осветљености, следи се пут од корена до сваког листа, затим се чита низ нула и јединица. Овим поступком креира се табела кодираних секвенци битова или речник. На овај начин, свакој вредности нивоа осветљености додељује се кôд записан у кодној табели.

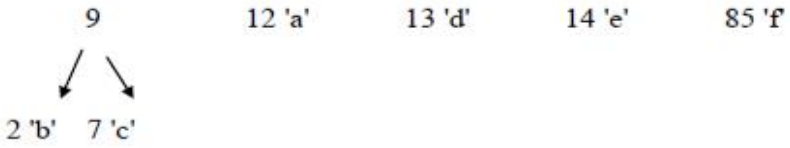
Декодер мора знати табелу кодова за правилно декодирање. Потребно је реконструисати бинарно стабло, а затим да се иде од корена стабла према вредностима битова из кодираног низа који се декодира док се не дође до неког знака. Овај поступак се понавља све док се не дође до краја кодираног низа.

За типичних 256 различитих нивоа осветљености, дужина кода сваког симбола не прелази 8 бита. Хофманово кодирање се може једноставно имплементирати, али ефикасност зависи од статистичке природе слике. Хофман пресликава симболе фиксне дужине у кодове променљиве дужине. Ово је оптимално само када су вероватноће симбола умношци броја 2. Хофманови кодови оптимални су за дати извор расподеле вероватноће. Међутим, њихова просечна дужина је дужа него код ентропијског кодирања, у оквиру вредности једног бита.Најлакши начин да се види како функционише Хофманов алгоритам је обрађивање примера.

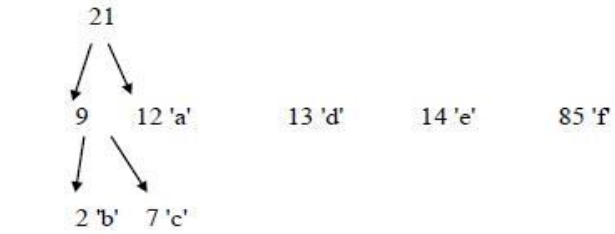
Претпоставити да се након скенирања датотеке добију следеће фреквенције знакова:

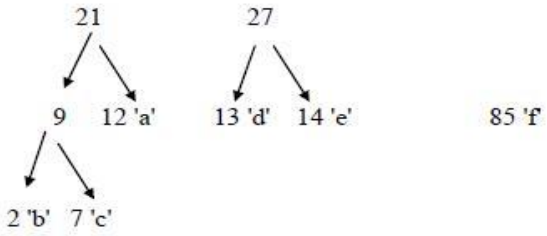
Карактер:	Фреквенција:
'a'	12
'b'	2
'c'	7
'd'	13
'e'	14
'f'	85

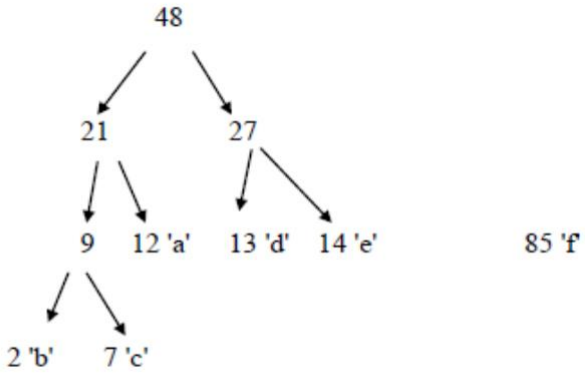
Сада је потребно да се креира бинарно стабло за сваки знак који такође чува фреквенцију са којом се јавља. Алгоритам је следећи: Наћи два бинарна стабла на листи која у својим чворовима чувају минималне фреквенције. Повезати ова два чвора на новоствореном заједничком чвору који неће сместити карактер, али ће сачувати збир фреквенција свих чворова повезаних испод њега. Дакле, то изгледа овако:

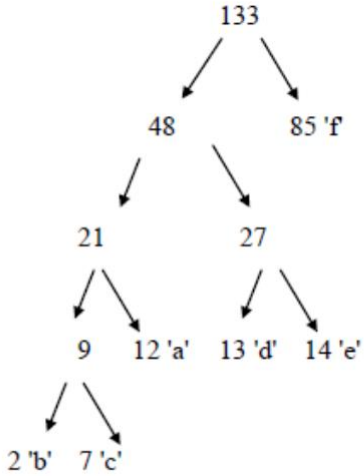


Понављати овај поступак док не остане само једно стабло.









Једном када је стабло изграђено, сваки чвор листа одговара слову са кодом. Да би се одредио код за одређени чвор, проћи стандардну путању за претрагу од корена до чвора листа.

За сваки корак лево додати 0, а за сваки корак десно додати 1. Тако да се за стабло претходно одрађено добију следећи кодови:

Слово:	Код:
'a'	001
'b'	0000
'c'	0001
'd'	010
'e'	011
'f'	1

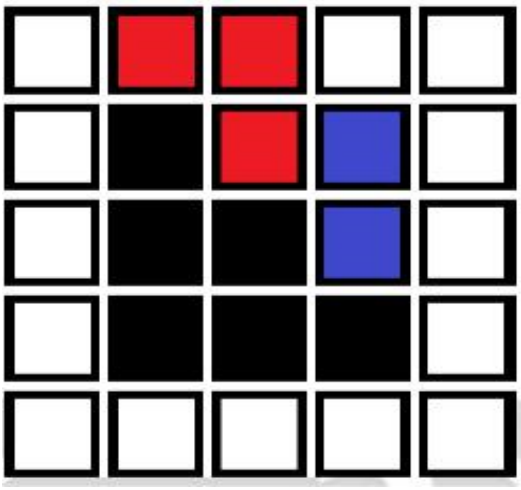
Да би се видело колика је уштеда приликом компресије потребно је да се израчуна колико се битова првобитно користи за чување података и од тога одузме колико битова се користи за чување података помоћу Хофмановог кода.

У датом примеру, пошто постоји шест знакова, претпоставити да је сваки ускладиштен са тробитним кодом. С обзиром да постоји 133 оваква знака, укупан број употребљених битова је 3 * 133 = 399.
Сада помоћу Хофманових фреквенција кодирања може да се израчуна нови укупни број искоришћених битова:

Letter	Code	Frequency	Total Bits
'a'	001	12	36
'b'	0000	2	8
'c'	0001	7	28
'd'	010	13	39
'e'	011	14	42
'f'	1	85	85
Total			238

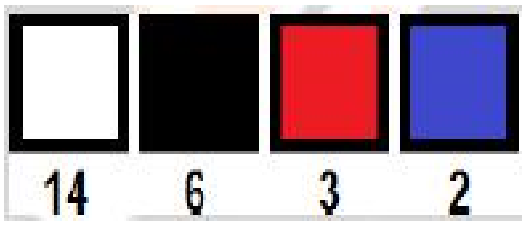
Тако се уштедео 399 - 238 = 161 бит, или скоро 40% простора за складиштење.

Хофманово кодирање се може сликовито приказати компресијом растерске слике.
Претпоставити да је 5 × 5 растерска слика са 8-битном бојом, односно 256 различитих боја.
Некомпресована слика ће узети 5 x 5 x 8 = 200 битова простора за складиштење.

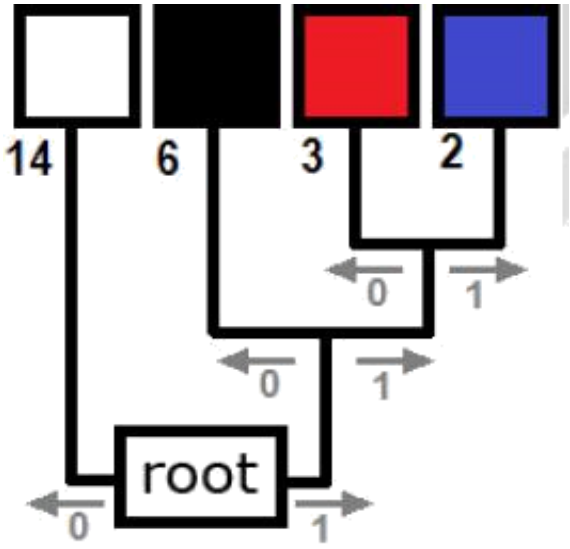


Слика 33: *Мрежа пиксела*

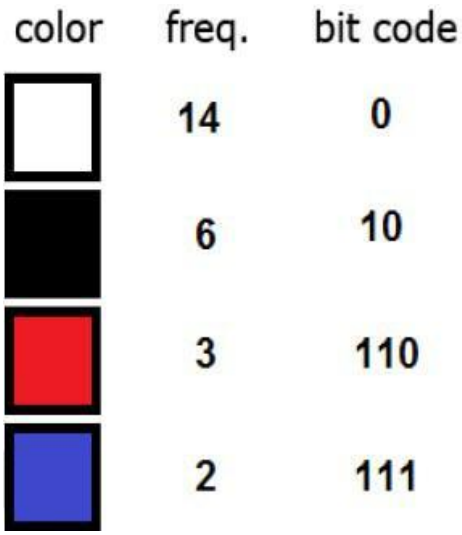
Прво се рачуна колико пута се свака боја појављује на слици. Затим се сортирају боје према опадајућој фреквенцији. Завршава се са редом који изгледа овако:



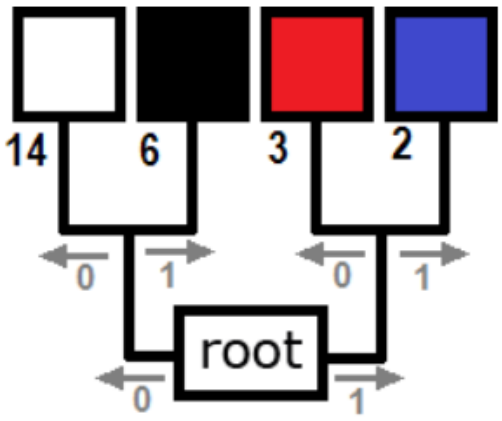
Сада саставити боје градећи стабло тако да су боје најудаљеније од корена најређе. Боје су спојене у паровима, а чвор чини везу. Чвор се може повезати или са другим чвором или са бојом. На пример, стабло може изгледати овако:



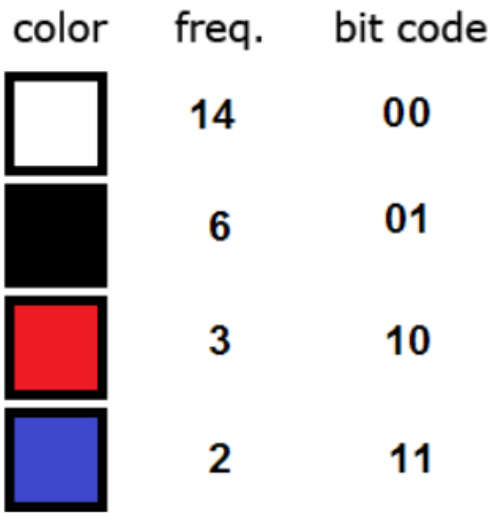
Резултат који је добијен је Хофманово стабло. Може се користити за кодирање и декодирање. Свака боја је кодирана на следећи начин. Кодови се креирају премештањем са корена стабла на сваку боју. Ако се на чвору скрене десно пише се 1, а ако се скрене лево - 0. Овај поступак даје Хофманову таблицу кодова у којој је сваком симболу додељен битни код тако да најчешћи симбол има најкраћи код, док најмањи уобичајени симбол добија најдужи код.



Хофманово стабло и кодна табела који су креирани нису једини могући, постоји још могућности.
Алтернативно Хофманово стабло може да изгледа и овако:



Одговарајућа табела кодова би онда била:



Коришћење прве варијанте је пожељније у овом примеру. То је зато што пружа бољу компресију за дату специфичну слику у односу на другу варијанту.
Будући да свака боја има јединствени битни кôд који није префикс било које друге, боје могу бити замењене својим битним кодовима у датотеци слике. Боја која се најчешће појављује, бела, биће представљена са само једним битом, а не са 8 битова. Црна ће узети два бита. Црвена и плава ће узети три бита. Након извршених ових замена, 200-битна слика биће компресована на 14 x 1 + 6 x 2 + 3 x 3 + 2 x 3 = 41 бита, што је око 5 бајтова у поређењу са 25 бајтова на оригиналној слици.

Наравно, за декодирање слике компресована датотека мора садржати табелу кодова, која заузима мало простора. Сваки битни код изведен из Хофмановог стабла недвосмислено идентификује боју, тако да компресија не губи информације [21].

7.1 Софтвер

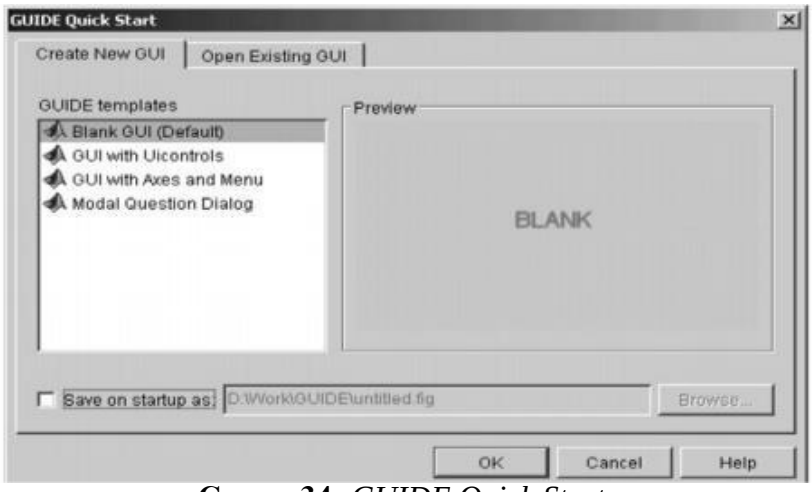
МАТЛАБ је окружење за нумеричке прорачуне и програмски језик четврте генерације који је развила фирма *MathWorks* [22]. МАТЛАБ омогућава лако манипулисање матрицама, приказивање функција и фитовање, имплементацију алгоритама, креирање графичког корисничког интерфејса, као и повезивање са програмима писаним у другим језицима, међу којима су Ц, Ц ++, Ц #, Јава, Фортран и Пајтон.

Матлаб је настао као скраћеница за *"MATrix LABoratory"* ("лабораторија за матрице"). Изумео га је Клив Молер (**енгл**. *Cleve Moler*) касних 1970их шеф катедре за информатику на Универзитету "Нови Мексико".

7.2Симулација Хофмановог алгоритма за компресију слика

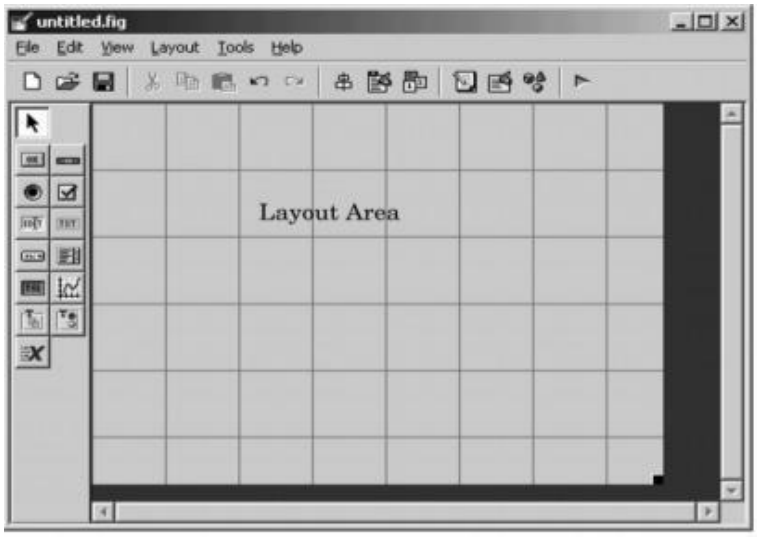
У МАТЛАБ-у постоји могућност да се креирају програми базирани на ГУИ (**енгл**. *GUI – Graphical User Interface*)[23].

Програм се стартује наредбом *guide* са командне линије, после чега се отвара *GUIDE Quick Start* прозор (слика 34), који нуди опције стартовања нових ГУИ-а или отварања постојећих. Други начин стартовања програма је опција **File → New→ GUI**.



Слика 34: *GUIDE Quick Start*

После притиска ОК дугмета, уколико је селектован *Blank GUI (Default)*, отвориће се Layout Editor (Слика 35). Са леве стране прозора се могу уочити све расположиве графичке УИ (*UI-User Interface*) контроле (објекти), које се уобичајено, без коришћења ГУИДЕ-а, добијају функцијом *uicontrol*. То су: **Push Button, Toggle Button, Radio Button, Checkbox, Edit Text, Static Text, Slider, Frame, Listbox, Popup Menu** и **Axes**.



Слика 35: *Layout Editor*

Затим се на радну површину превлаче потребни елементи који се налазе са леве стране прозора. Када се креира жељено графичко окружење, покрене се програм и МАТЛАБ тада креира два фајла. Један је класичан **.m** фајл у којем се налази програмски кôд, а други је **.fig** фајл у којем се налази ГУИ.

Приликом покретања апликације, отвара се прозор (слика 36) који омогућава кориснику да изврши компресију жељене слике тако што ће је селектовати из одређеног фолдера кликом на дугме '**Select image**', а то је реализовано на следећи начин. Могуће је селктовати различите формате слика .**bmp, .jpg, .png** и **'tiff'**.

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject          handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB % handles structure with handles and user
data (see GUIDATA) global file_name;

%guidata(hObject,handles)

%%%% Узима се улазна слика која треба да буде компресована од стране корисника%%%%
file_name=uigetfile({'*.bmp;*.jpg;*.png;*.tiff;';'*.*'}

%%%% Одредити величину у бајтовима оригиналне слике %%%%
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;

set(handles.text7,'string',Size);
imshow(file_name,'Parent', handles.axes1)
```

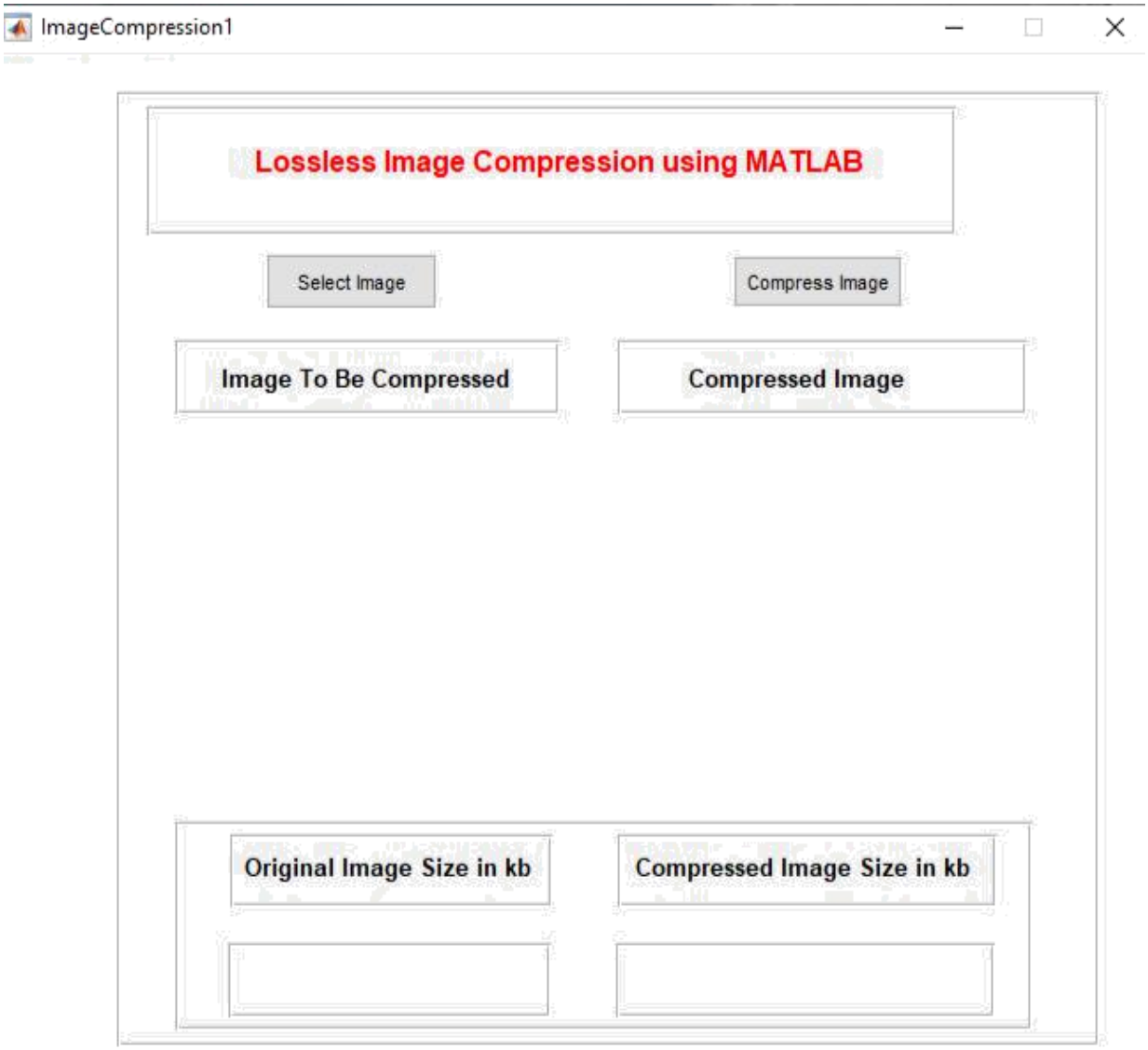
Затим, да би се добила компресована слику потребно је притиснути дугме **'Compress Image'**.

У алгоритму компресије слике, улазна слика је подељена на блокове 8x8, а дводимензионални ДЦТ се израчунава за сваки блок. ДЦТ коефицијенти се затим квантизују, кодирају и преносе. Читач датотека декодира квантизоване ДЦТ коефицијенте, рачуна инверзни дводимензионални ДЦТ сваког блока, а затим их враћа у једну слику. За типичне слике, многи ДЦТ коефицијенти имају вредности близу нуле; ови коефицијенти се могу одбацити без озбиљног утицаја на квалитет реконструисане слике.
Израчунава се дводимензионални ДЦТ од 8 до 8 блокова на улазној слици; одбацује (поставља на нулу) све осим 10 од 64 ДЦТ коефицијента у сваком блоку, а затим реконструише слику помоћу дводимензионалне инверзне дискретне косинусне трансформације сваког блока. Користи се метода израчунавања матрице трансформације.

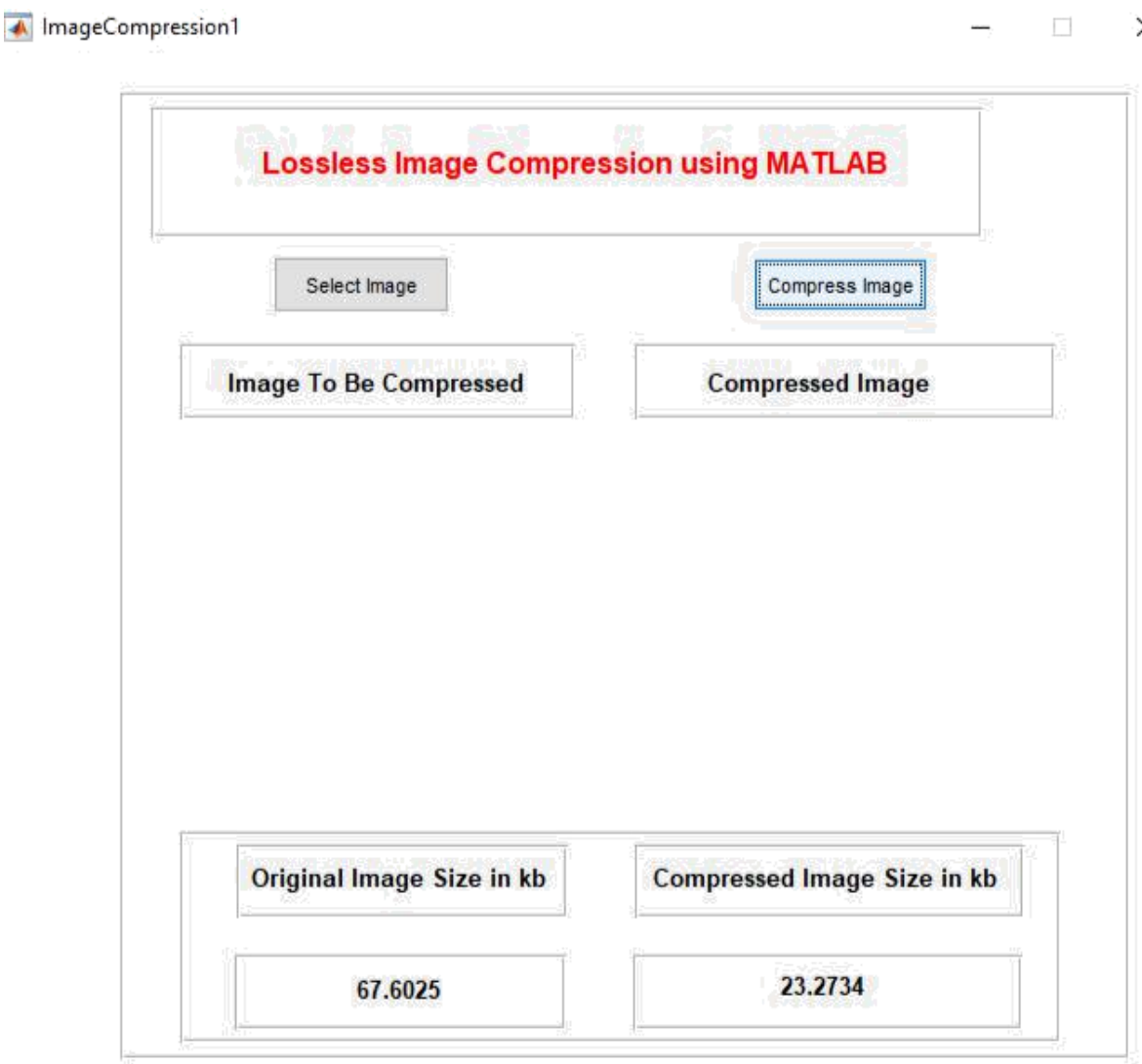
```
I1 = imread(file_name);

%%%% Компресија прве равни слике %%%%

I = I1(:,:,1);
%%%% Промена прецизности пиксела са unit8 на double %%%%

I = im2double(I);
T = dctmtx(8);

%%%% Помоћу команде за обраду блокова, улазна слика је подељена на блокове 8x8 %%%%
%%%% Израчунава се ДЦТ за сваки блок %%%%
%%%% Команда blkproc рачуна ДЦТ слике I %%%%
%%%% Трансформациона матрица T и транспонована матрица T' као параметри P1&P2 ДЦТа
%%%%

B = blkproc(I,[8 8],'P1*x*P2',T,T');
```

```
%%%% маска матрице %%%%

mask = [1     1     1     1     0     0     0  0
1     1     1     0     0     0     0     0
1     1     0     0     0     0     0     0
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     0
0     0     0     0     0     0     0     0];

%%%% Наношење  маске на матрицу B за уклањање високофреквентних вредности %%%%
B2 = blkproc(B,[8 8],'P1.*x',mask);
%%%% Инверзна ДЦТ за сваки блок %%%%
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
```

```
%%%% Компресија друге равни слике %%%%
I = I1(:,:,2);
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
     1   1   1   0   0   0   0   0
     1   1   0   0   0   0   0   0
     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   0
     0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I3 = blkproc(B2,[8 8],'P1*x*P2',T',T);

I = I1(:,:,3);
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
     1   1   1   0   0   0   0   0
     1   1   0   0   0   0   0   0
     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   0
     0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I4 = blkproc(B2,[8 8],'P1*x*P2',T',T);

%%%% Спајање све три равни %%%%
L(:,:,:)=cat(3,I2, I3, I4);
imwrite(L,'CompressedImage.jpg');
%%%% Поређење величине оригиналне слике и компресоване слике %%%%
fileinfo = dir('CompressedImage.jpg');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
```



Слика 36: *Почетни екран*

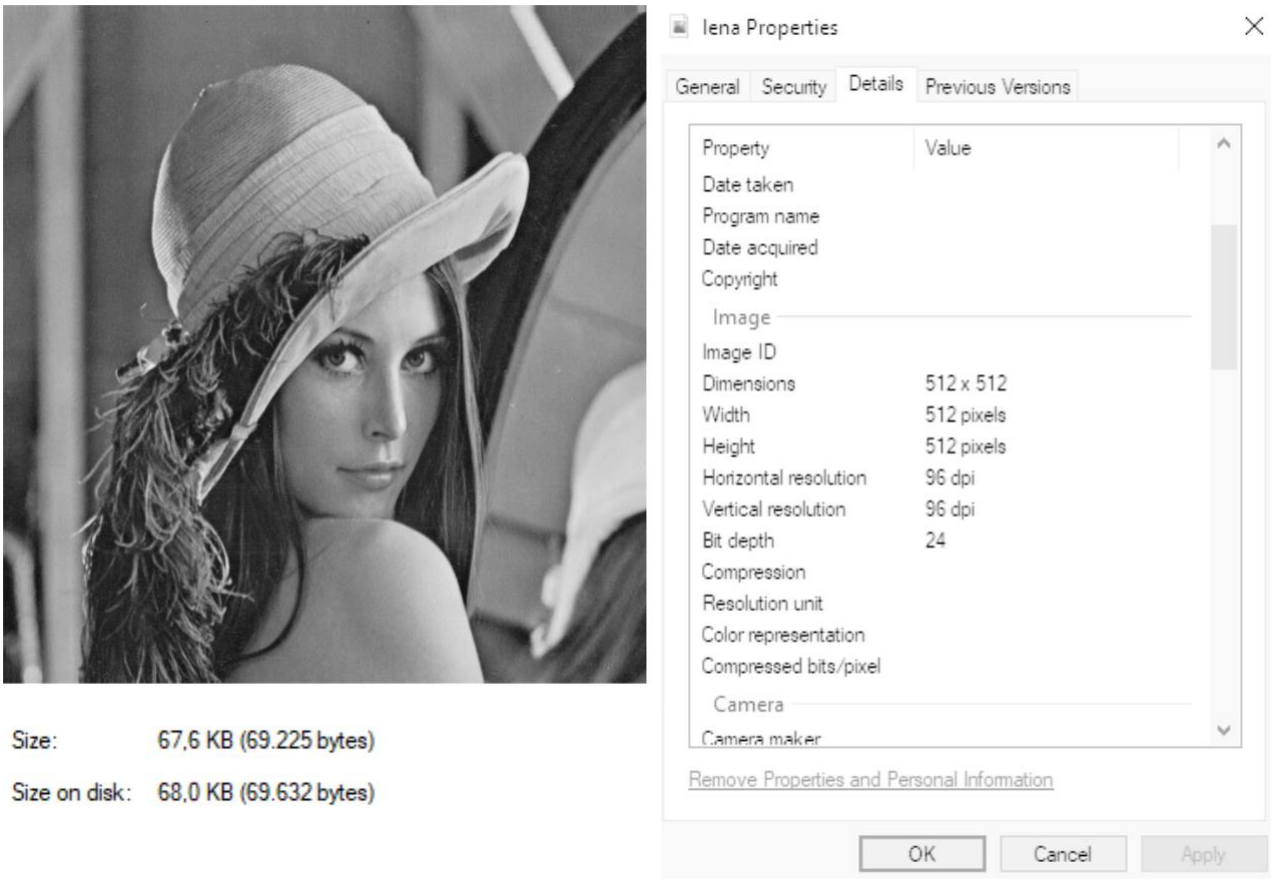


Слика 37: *Реултат након компресије*

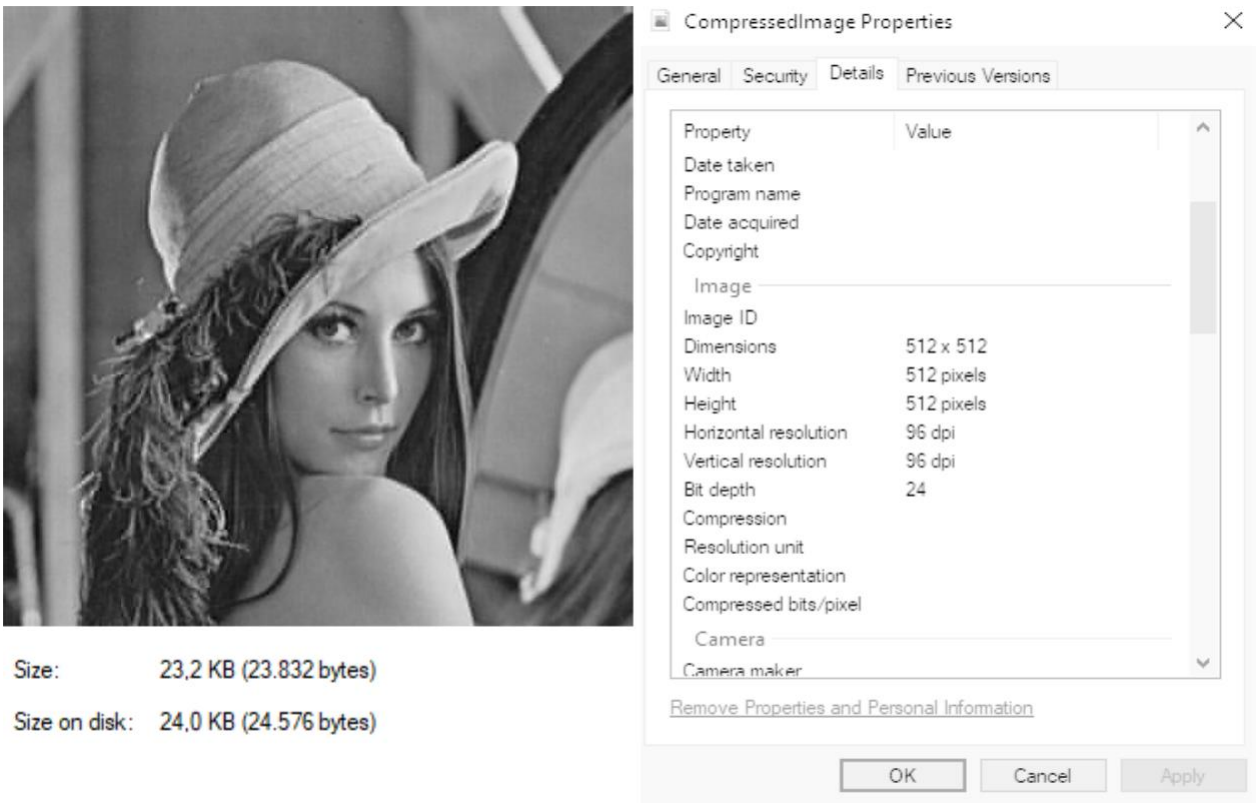
Компресована слика се аутоматски чува на рачунару у фолдеру у којем се налази апликација. Можемо упоредити детаље оригиналне слике и слике која је добијена након компресије.

Слика 38 приказује оригиналну слику, 'lena.jpg' у сивој скали која је коришћена за даљу компресију. Приказане су такође и информације оригиналне слике.

На слици 39 приказана је иста слика која је компресована помоћу Хофмановог алгоритма преко развојног окружења МАТЛАБ. Ова слика је пример компресије без губитака, јер нема губитка у резолуцији слике у поређењу са оригиналном сликом. Такође су приказане информације о слици након компресије.



Слика 38: *Оригинална слика*



Слика 39: *Компресована слика*

Упоређивањем две слике јасно је видљиво да је Хофманов алгоритам / кôд радио како се очекивало. Оригинална слика величине 67.6025кб се своди на 23.2734кб, што је приближно 30%. Пошто је резолуција и оригиналне слике и компресоване слике иста, тј 96dpi (тачака по инчу), доказује да је дошло до компресије слике без губитака.

Коришћење МАТЛАБ-а као развојног језика омогућава следеће предности: МАТЛАБ је математички софтвер опште намене који је лак за употребу и веома користан за анализу података, симулације, презентацију резултата и још много тога. Софтвер ради на УНИКС-у (**енгл**. *UNIX*) и ВИНДОВС-у (**енгл.** *WINDOWS*). МАТЛАБ је тумач, што значи да извршава сваку командну линију одмах након уноса. Дакле, могуће је или да се уносе команде ред по ред (режим упита) или да се прво припреми датотека скрипте, а затим покрене програм [13].

Компресија података је најважнија ствар у данашњем свету. Потребно је компресовати огромну количину података како би били пребачени са једног места на друго или у формату за складиштење. Због тога се подаци морају компресовати. Ова предложена техника компресије побољшала је ефикасност компресије датотека (попут .txt, .docx, .pptx) помоћу Хофмановог приступа.

8. Закључак

У рачунарству рад са великом количином података је неизбежан. Када се подаци компресују смањује се број битова којим су ти подаци представљени.

Главне предности компресије су смањење хардвера за складиштење, време преноса података и пропусни опсег комуникације. Ово резултује значајном уштедом трошкова. Компресоване датотеке захтевају знатно мањи капацитет складиштења од некомпресованих датотека, што значи значајно смањење трошкова за складиштење. Компресованој датотеци је, такође, потребно мање времена за пренос док троши мање пропусног опсега мреже.

Компресија података је тако постала веома битна у свету компјутера и нашироко се користи у многим апликацијама и уређајима.

Различите врсте садржаја често користе технике компресије специфичне за ту врсту садржаја, али већина користи бар један или комбинацију неколико основних алгоритама компресије као део целокупног процеса компресије - у фази процеса када су подаци распоређени тако да ови алгоритми могу да их обрађују. Овај рад је представио основне алгоритме компресије и поступак компресије. Компресијом слика Хофмановим алгоритмом није дошло до губитака, оригинална слика и компресована слика имају исту резолуцију, разлика је у величини слике.

9. Литература

[1] Математичка теорија комуникације. Преузето: Август 20, 2020, https://knowledgeinformationdata.wordpress.com/tag/claude-e-shannon/

[2] Примери слика - X зрачење. Преузето: Септембар 22, 2020, од https://faraday.emu.edu.tr/ee583/

[3] Примери слика - УВ зрачење. Преузето: Септембар 22, 2020, од https://faraday.emu.edu.tr/ee583/

[4] NTSC, PAL. Преузето: Август 27, 2020, од https://www.diffen.com/difference/NTSC_vs_PAL

[5] CODEC. Преузето: Август 27, 2020, од https://www.merriam-webster.com/dictionary/codec

[6] Gregory K. Wallace (1991). The JPEG still picture compression standard. *Communications of the ACM [online]*. Доступно на: http://ijg.org/files/Wallace.JPEG.pdf

[7] ЈПЕГ компресија шематски приказ: Преузето: Септембар 9, 2020, од https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348

[8] Дељење слике у блокове пиксела. Преузето: Септембар 15, 2020, од https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348

[9] R. Vijaya Kumar Reddy (2016). Grey level to RGB using YCbCr color space technique. *International Journal of Computer Applications [online]*. Доступно на: https://www.researchgate.net/publication/306127848_Grey_level_to_RGB_using_YCbCr_color_space_Technique

[10] R. Vijaya Kumar Reddy (2016). Grey level to RGB using YCbCr color space technique. *International Journal of Computer Applications [online]*. Доступно на: https://www.researchgate.net/publication/306127848_Grey_level_to_RGB_using_YCbCr_color_space_Technique

[11] Пример RGB и YCbCr слике. Преузето: Септембар 22, 2020, од http://www.bk.isy.liu.se/en/courses/tsbk06/material/ (lecture 8-9)

[12] Матрица конверзије. Преузето: Септембар 23, 2020, од https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdprfx/b550d1b5-f7d9-4a0c-9141-b3dca9d7f525

[13] Image and Video processing (2011). *Intel Integrated Performance Primitives for Intel Architecture Reference Manual[online]*. Доступно на: https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/index.htm#IPPI/ippi_ch6/ch6_image_downsampling.htm

[14] Zero-shift. Преузето: Септембар 24, 2020, од https://abyx.be/post.php?id=12

[15] Трансформисана матрица. Преузето: Септембар 24, 2020 од https://abyx.be/post.php?id=12

[16] Квантизација. Преузето: Септембар 24, 2020, од https://abyx.be/post.php?id=12

[17] Elie Biham (1995). *A known Plaintext Attack on the PKZIP Stream Cipher*

[18] Run Length Encoding. "Coding - Compression", CS Field Guide (CCBY-NC-SA 3.0). Оригиналан чланак доступан на: https://csfieldguide.org.nz/en/chapters/coding-compression/run-length-encoding/

[19] Ashley Taylor. CS101 – Introduction to computing principles, Image – Introduction RGB everywhere. Доступно на: https://web.stanford.edu/class/cs101/index.html

[20] Др Миодраг В. Поповић, Дигитална обрада слике, Академска мисао, Београд, 2006.

[21] Rafael C. Gonzales, Digital Image Processing, 2008.

[22] "MATLAB Documentation". MathWorks. Преузето: Септембар 25, 2020, од https://www.mathworks.com/help/matlab/index.html

[23] J.S. Park. MATLAB GUI (Graphical User Interface) Tutorial for Beginners, University of Incheon. Доступно на: http://www.ee.bgu.ac.il/~adcomplab/Tutorial%20MATLAB%20GUI.pdf