Algoritmi kompresije multimedijalnih sadržaja

Guljaš, Vilko

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Vilko Guljaš

Algoritmi kompresije multimedijskih podataka

Završni rad

Pula, 2019.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Algoritmi kompresije multimedijskih podataka

Završni rad

JMBAG: 0303063244, redovan student

Studijski smjer: Informatika

Kolegij: Multimedijalni sustavi

Mentor: doc. dr. sc. Darko Etinger

Pula, kolovoz 2019.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Vilko Guljaš, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, 2019. godine

IZJAVA

o korištenju autorskog djela

Ja, Vilko Guljaš dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Algoritmi kompresije multimedijskih podataka koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Sadržaj

Uvod 1
1. Kompresija podataka 2
1.1. Lossless kompresija 2
1.2. Lossy kompresija 3
2. Kompresija tekstualnih datoteka 4
2.1. Run-length encoding algoritam 4
2.2. Lempel-Ziv-Welch algoritam 5
2.3. Huffmanov algoritam 7
3. Kompresija slike 10
3.1. Run-length Encoding kompresija slika 11
3.2. JPEG 14
4. Kompresija audio zapisa 20
4.1. MP3 21
5. Kompresija video zapisa 24
6. Arhivske datoteke 26
7. Zaključak 27
Literatura 28
Popis slika 30
Sažetak: 31
Abstract: 32

Uvod

Kompresijom podataka smanjuje se njihova veličina čime se oslobađa prostor za pohranu te se ubrzava prijenos tih podataka što je vrlo korisno u brojnim slučajevima, no ovisno o vrsti kompresije, cijena je često gubitak kvalitete. Kompresiju postižemo pomoću raznih algoritama, svaki od njih ima svoje prednosti i mane, te se koristi u određenim slučajevima.

Kompresija podataka ima dugu povijest izvan informatike. Morseov kod, koji je izmišljen 1838. godine je najraniji oblik kompresije podataka gdje se slova abecede sažimaju u kraće kodove. U informatici je kompresija podataka počela igrati važnu ulogu 1970-ih godina, kada su bile razvijene prve verzije Lempel-Ziv algoritama. Razvojem informatike stvarala se sve veća potreba za uštedu memorijskog prostora, što je pokrenulo razvoj raznih tehnika kompresije multimedijskih sadržaja koji su u sirovom obliku zauzimali preveliki prostor.

Danas postoji velik broj algoritama odnosno tehnika kompresije raznih tipova podataka gdje svaka od njih ima svoje specifičnosti, te odabir tehnike kompresije ovisi upravo o tim specifičnostima i cilju kompresije; Pojedini formati multimedijskih sadržaja kompatibilni su samo sa određenim softverom za prikaz/reprodukciju ili određenim operativnim sustavima.

U ovome radu obrađeno je nekoliko osnovnih algoritama kompresije, te procesi kompresije najzastupljenijih formata multimedijskih sadržaja, koje često koriste osnovne algoritme kao potprocese.

1. Kompresija podataka

Kompresija podataka je proces kojim se smanjuje veličina datoteke određenim metodama odnosno algoritmima te se koristi svugdje u informatici u svrhu očuvanja više kapaciteta za pohranu podataka, ubrzanjem prijenosa podataka, smanjenja troška hardvera za pohranu i manjeg tereta na mrežu. Ovi razlozi su jednaki i za korporacije i za obične korisnike. U oba slučaja kompresija omogućuje financijsku uštedu – manja potrošnja na hardver za pohranu, te vremensku uštedu – veće brzine transfera podataka.

Kompresirati se može gotovo svaki tip podataka na više načina, te je bitno odabrati optimalni način kompresije – algoritam – za određeni tip podataka i krajnji cilj kompresije.

Jednostavan primjer kompresije podataka s kojim je većina svakodnevnih korisnika računala upoznata su .rar i .zip datoteke odnosno arhive. Koriste različite algoritme kompresije ali krajnji rezultat im je vrlo sličan – pretvoriti skup datoteka bez obzira na njihovu vrstu, u jednu kompresiranu datoteku (s nastavkom .zip ili .rar). To je takozvana „lossless“ kompresija gdje u ovom slučaju datoteke koje smo komprimirali možemo pregledati, no ne možemo ih koristiti dok nad njima ne izvršimo dekompresiju. Ovakav tip kompresije je vrlo popularan jer omogućava lakšu i bržu razmjenu podataka raznih veličina preko interneta.

1.1. Lossless kompresija

U lossless kompresiji, kao što ime nalaže, ne može doći do gubitka informacija. Podatke koji su na ovaj način kompresirani se moraju dekompresirati u originalne podatke, jednake kao i prije kompresije.

Najbolji primjer primjene lossless kompresije je kompresija tekstualnih sadržaja, gdje je vrlo bitno da tekst ostane potpuno jednak jer i najmanje odstupanje može potpuno promijeniti smisao teksta odnosno prikazati netočne podatke.

Postoji mnogo situacija u kojima sami podaci imaju prioritet nad što većom uštedom vremena i prostora. Kao što je navedeno ranije, bilo kakvi tekstualni podaci –

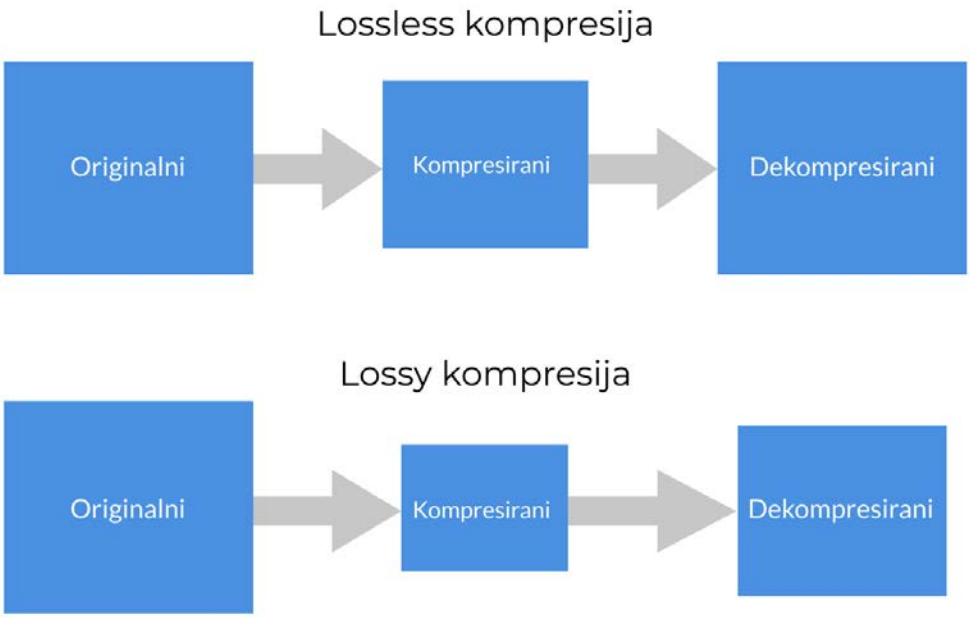
dokumenti, računi i slično, ne smiju biti ništa drugo osim originala. Primjere možemo naći i u drugim tipovima podataka – slike, video i audio sadržaji koji će se eventualno koristiti u svrhu određenih istraživanja moraju sadržavati i najsitnije detalje kao i u originalu kako ništa ne bi bilo previđeno.

No postoji daleko više situacija u kojima nije potrebno dekompresirati datoteke u njihov originalni oblik, već je prioritet što veći omjer kompresije, u takvim slučajevima koristimo lossy kompresiju.

1.2. Lossy kompresija

Metode lossy kompresije omogućuju veći omjer kompresije, no podaci u procesu gube neke informacije pa se tako ne dekompresiraju u točno onakve kakvi su bili originalno, prije kompresije. Laički rečeno, podaci će izgubiti na kvaliteti. Ovisno o raznim faktorima, u nekim slučajevima će taj gubitak biti neprimjetan, no u drugim slučajevima će se itekako primijetiti.

Količina izgubljenih informacija će ovisiti o algoritmu koji koristimo i omjeru kompresije, a odabir algoritma i omjera će ovisiti o tome koliko izgubljenih informacija smijemo dopustiti u određenom slučaju.



Slika 1. lossy i lossless kompresija, izvor: vlastiti izvor

2. Kompresija tekstualnih datoteka

Tekstualna kompresija najčešće spada pod lossless kompresiju, jer u bilo kakvoj tekstualnoj datoteci nije poželjno izgubiti podatke, gdje samo jedan krivi ili nepostojeći znak (character) može imati iznimno loš utjecaj na datoteku.

Većina algoritama za tekstualnu kompresiju se temelje na statističkoj metodi ili metodi rječnika.

2.1. Run-length encoding algoritam

Run-length encoding (RLE) je najjednostavniji algoritam kompresije teksta, on je bez gubitaka, i baziran je na nizovima jednakih znakova (byteova).

Osnovni princip je pronalaženje nizova jednakih znakova u tekstu, te zamjena svih znakova u nizu sa brojem koji predstavlja duljinu niza i znakom koji stvara niz; Ako se znak *x* ponavlja *n* puta, zamijeni cijeli niz sa jednim parom *nx*.

bee free -> b2e fr2e

(8 znakova) -> (8 znakova)

U ovom slučaju brojka 2 ukazuje na početak niza za kompresiju, no ako dvojke ili bilo koji drugi broj već postoje u originalnom tekstu, mora postojati distinkcija između njih inače se ne zna koja dvojka služi čemu te bi to predstavljalo problem pri dekompresiji. Tada se dodaje dodatni znak koji se ne pojavljuje u tekstu, kao indikator početka niza, najčešće znak iz ASCII tablice.

Dakle ako uzmemo prethodni primjer i dodamo dvojku,

2 bee free -> 2 b⊥2e fr⊥2e

(10 znakova) -> (12 znakova)

Ovdje znak ⊥ predstavlja početak niza. U oba slučaja problem je što uopće nismo postigli kompresiju, naprotiv u drugom primjeru imamo više znakova nego u originalnom tekstu. Niti u jednom jeziku na svijetu nećemo pronaći konzistentne nizove

od tri ili više jednakih znakova pa iz tog razloga ovaj algoritam ima dosta limitiranu primjenu – samo u datotekama gdje znamo da će se pojavljivati duži nizovi jednakih znakova odnosno byteova.

AAAAA5HHHHHHHHHJJJJJSSSSSSSS7ASSSS -> ⊥5A5⊥8H⊥5J⊥8S7A⊥4S

(33 znakova) -> *(18 znakova)*

2.2. Lempel-Ziv-Welch algoritam

Lempel-Ziv-Welch (LZW) je algoritam baziran na metodi rječnika. Osnovni princip je stvaranje novih stringova odnosno fraza koje se pojavljuju u tekstu te njihovo dodavanje u rječnik.

LZW algoritam kreće tako da se inicijalizira rječnik sa svim simbolima abecede – svih 256 simbola ASCII tablice. Nakon inicijalizacije rječnika početni simbol već postoji u rječniku pa encoder prelazi na drugi simbol, stvara string koji se sastoji od prethodnog i trenutnog simbola, i pretražuje postoji li taj string u rječniku. U slučaju prvog spajanja simbola u string, on neće postojati u rječniku pa će ga encoder dodati, no kasnije u tekstu ako je taj string već u rječniku, encoder prelazi na sljedeći simbol u tekstu i spaja ga s prethodnim stringom.

Uzmimo za primjer rečenicu „ranjenici i ranoranioci“.

Pseudokod:

1. Inicijalizacija rječnika simbolima iz ASCII tablice – ulazi od 0 do 255
2. Unosi se prvi simbol *r* koji je pronađen u rječniku (114 prema ASCII kodu)
3. Unosi se drugi simbol *a* i spaja se s prethodnim simbolom što rezultira stringom *ra* koji nije pronađen u rječniku. Encoder ispisuje 114 (*r*), a string *ra* se dodaje u rječnik na prvo sljedeće slobodno mjesto - 256.
4. Unosi se sljedeći simbol *n* i spaja s prethodnim što rezultira stringom *an* koji nije pronađen u rječniku. Encoder ispisuje 97 (*a*), a string *an* se dodaje u rječnik na prvo sljedeće slobodno mjesto - 257.
5. Ponavljaj korake 3. i 4. do end of file-a.

	CHAR	*STRING + CHAR*	U rječniku?	Ispis	Dodaj u rječnik	Komentar
1	r					
2	a	ra	ne	r	256 = ra	
3	n	an	ne	a	257 = an	
4	j	nj	ne	n	258 = nj	
5	e	je	ne	j	259 = je	
6	n	en	ne	e	260 = en	
7	i	ni	ne	n	261 = ni	
8	c	ic	ne	i	262 = ic	
9	i	ci	ne	c	263 = ci	
10	/	i/	ne	i	264 = i/	Razmak prikazujemo znakom /
11	i	/i	ne	/	265 = /i	
12	/	i/	da (264)			Prvo nailaženje na string koji je dodan u rječnik.
13	r	i/r	ne	264	266 = i/r	Trenutni char se spaja s prethodno dodanim stringom. U ispisu, 264 predstavlja string dodan u rječnik - */i*
14	a	ra	da (256)			
15	n	ran	ne	256	267 = ran	
16	o	no	ne	n	268 = no	
17	r	or	ne	o	269 = or	
18	a	ra	da (256)			
19	n	ran	da (267)			
20	i	rani	ne	267	270 = rani	
21	o	io	ne	i	271 = io	
22	c	oc	ne	o	272 = oc	
23	i	ci	da (263)			
24	EOF	/		263		End of file.

2.3. Huffmanov algoritam

Ovaj algoritam razvio je David Huffman 1952. godine. Prvenstveno je služio samo za kompresiju tekstualnih datoteka no osim toga primjenjuje se i kod kompresije drugih tipova podataka. Osnovna ideja Huffmanovog algoritma je zadati (što kraće) kodne riječi simbolima koji imaju najveću frekvenciju pojavljivanja. Iz tog razloga potrebno je znati vjerojatnost pojave svakog simbola. Na temelju vjerojatnosti pojave simbola kreira se Huffmanovo stablo.

Dakle, kompresija Huffmanovim algoritmom funkcionira na principu binarnih stabala elemenata. U početku su svi elementi listovi stabla, sadrže svoju vrijednost (simbol), frekvenciju pojavljivanja, te poveznicu s elementom roditeljem. Unutarnji elementi (čvorovi) također sadrže svoju vrijednost, frekvenciju, poveznice s elementima djecom i poveznicu s elementom roditeljem.

Ovaj algoritam neće postići najbolju moguću kompresiju u svim primjenama, no u određenim slučajevima dokazuje da je vrlo dobra opcija budući da se koristi u procesima kompresije nekih od najzastupljenijih formata digitalnih slika (JPEG) i audio zapisa (MP3).

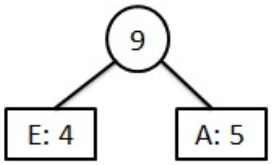
Postupak izrade Huffmanovog binarnog stabla:

1. Stvori elemente listove za svaki jedinstveni simbol i kreiraj min heap svih elemenata listova. Glavni faktor koji se uspoređuje među elementima je frekvencija pojavljivanja. U početku je korijen element koji ima najmanju frekvenciju.

Uzmimo kao primjer zamišljeni niz znakova u kojemu vrijedi sljedeće:

znak	E	A	C	F	D	B
frekvencija	4	5	7	12	15	25

2. Iz hrpe (min heap) izvlačimo dva elementa s najmanjim frekvencijama i dodajemo unutarnji čvor s njihovim zbrojem. Dobiveni element dodajemo natrag u min heap.

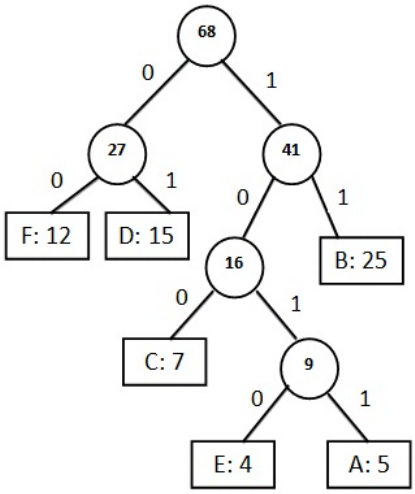


Slika 2. Huffmanov algoritam, drugi korak, izvor: https://4.bp.blogspot.com/-A2fQkvB33c4/VCBT-e4A9OI/AAAAAAAAC38/Go9pNBr6sew/s1600/1.jpg

Min heap sada izgleda ovako:

znak	C	EA	F	D	B
frekvencija	7	9	12	15	25

3. Ponavljamo prethodni korak sve kod nam u heapu ne ostane samo jedan element.



Slika 3. Huffmanov algoritam, dovršeno stablo, izvor: https://2.bp.blogspot.com/-sqc-xXvawIM/VCBUA1fTatI/AAAAAAAAC4g/dF3MyPX7twg/s1600/5.jpg

znak	FDCEAB
frekvencija	68

Izračunati Huffmanov kod ovoga primjera je sljedeći:

znak	vjerojatnost znaka	Huffmanov kod
A	0.074	1000
B	0.368	01
C	0.103	111
D	0.221	00
E	0.059	1001
F	0.176	110

„BBDDFBEADB“ = 0101000011001100110000001

3. Kompresija slike

Digitalna slika je pravokutno polje pixela raspoređenih u retke *r* i stupce *s*, gdje izraz *r* X *s* predstavlja rezoluciju slike. Svaki pixel zauzima malu pravokutnu regiju ekrana i može biti jedan bit ako je crne ili bijele boje, ili više bitova ako je neke druge boje. Cilj kompresije slike je manipulirati slikama na tehničkoj razini s obzirom na osjetljivosti ljudskog oka – smanjiti memorijsku veličinu slike a da ona prividno ne izgubi na kvaliteti iako se upravo to dešava

Na sljedećoj slici je prikazane su originalna slika te kompresirana verzija iste slike, gdje je stupanj kompresije izričito velik te su razlike vrlo očite.

Slika 4. kompresija slike, izvor:
https://commons.wikimedia.org/wiki/File:JPEG_example_JPG_RIP_001.jpg

Neki od najpoznatijih formata digitalnih slika su JPEG (Joint Photographic Experts Group) koji je vrlo pogodan za kompresiju visokodetaljnih slika, PNG (Portable Network Graphics) čija je prednost uređivanje postojećih slika bez gubitka na kvaliteti te dijelovi slike mogu biti prozirni (transparency), GIF (Graphics Interchange Format) koji se uglavnom koristi za sličice manje rezolucije te se njime mogu izraditi animacije.

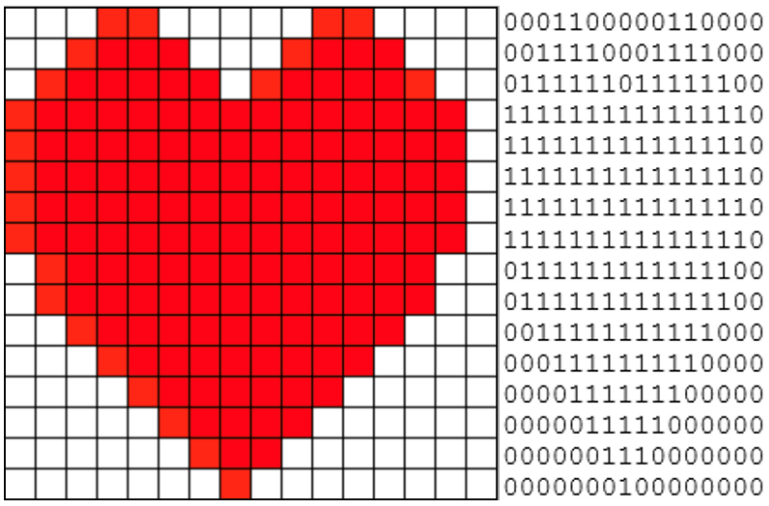
3.1. Run-length Encoding kompresija slika

Ideja RLE kompresije nad slikama je da ako nasumično odaberemo jedan pixel u slici, postoji dobra šansa da i njegovi susjedni pixeli imaju istu boju.

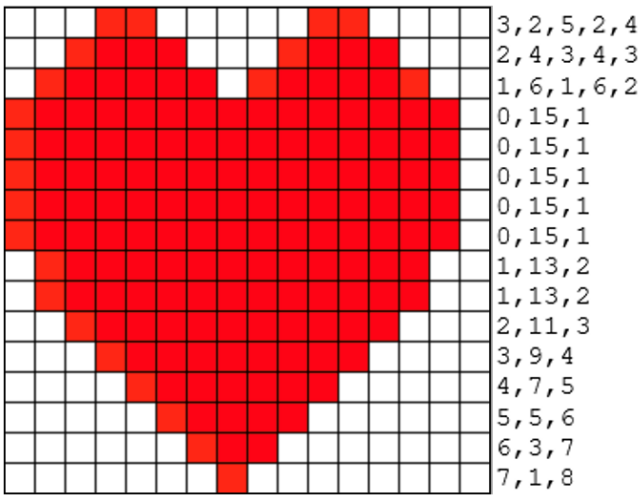
Kako bi primijenili RLE kompresiju nad slikama, moramo pretpostaviti da su pixeli slike spremljeni u nizu (array) zvanome *bitmap* u memoriji. Pixeli su u bitmapi uglavnom raspoređeni u linijama skeniranja (scan lines) tako da je prvi pixel u bitmap onaj koji se nalazi u gornjem lijevom kutu slike, a zadnji onaj koji se nalazi u donjem desnom kutu slike.

Princip RLE kompresije je isti kao i pri kompresiji teksta, no u ovom slučaju postoje dodatni detalji te se umjesto tekstualnih simbola. Recimo da pokušavamo kompresirati sliku koja se sastoji samo od crnih i bijelih pixela. Pri kompresiji se dakle bitmap skenira red po red tražeći nizove pixela iste boje. Ako bitmap krene s nizom od 20 bijelih pixela, zatim 7 crnih pixela, zatim 65 bijelih. Budući da kompresor zna da baratamo sa samo dvije boje, kao output je potrebno samo ispisati brojke 20,7,65. U ovakvim slučajevima kompresor uvijek pretpostavi da je prvi pixel bijele boje, tako da ako bitmap kreće sa crnim pixelom, output stream kreće s nulom (0 bijelih pixela na početku bitmap).

Uzmimo za primjer sljedeću sliku; umjesto crne koristimo crvenu boju koja će dakle poprimiti binarnu vrijednost 1.



Slika 5. Prikaz slike prije RLE kompresije, izvor: https://www.khanacademy.org/computing/ap-computer-science-principles/computers-101/file-compression/a/simple-image-compression



Slika 6. Prikaz slike nakon RLE kompresije, izvor: https://www.khanacademy.org/computing/ap-computer-science-principles/computers-101/file-compression/a/simple-image-compression

RLE se također može koristiti i za grayscale odnosno slike s više nijansi boje. Svaki niz pixela istog intenziteta (iste nijanse) je kodiran kao par (duljina niza, vrijednost pixela). Duljina niza obično zauzima jedan byte što dozvoljava nizove duge do 255 pixela dok vrijednost pixela iznosi više bitova, ovisno o nijansi boje.

Uzmimo za primjer sljedeći niz:

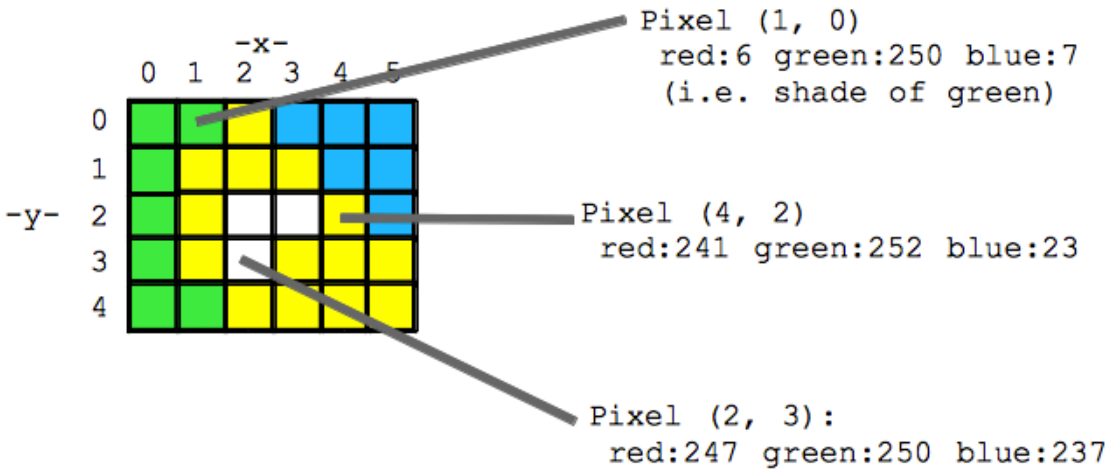
14, 25, 25, 25, 25, 25, 25, 56, 77, 78, 43, 43, 43, 43, 43, 43, 43

Budući da kompresija slika s više nijansi ne sadrži binarne vrijednosti, kompleksnost se povećava te je bitno razlikovati koje brojke predstavljaju vrijednost pixela a koje duljinu niza. Ako je broj nijansi 256, taj broj možemo smanjiti na 255 te jednu vrijednost iskoristiti kao „zastavicu“ koja će biti indikativna početku niza. Ukoliko postavimo vrijednost zastavice na 255, prethodni niz izgledat će ovako:

14, 255, 6, 25, 56, 77, 78, 255, 7, 43

Dakle, prvi puta kada naiđemo na zastavicu znamo da sljedeća dvije brojke predstavljaju duljinu niza i vrijednost pixela od kojih se taj niz sastoji.

U slikama u boji uobičajeno je da svaki pixel teži tri byte-a, po jedan za svaku nijansu crvene, zelene i plave boje (RGB) tako da jedan pixel možemo prikazati vrijednostima (45, 76, 123).



Slika 7. RGB vrijednosti pixela, izvor: https://web.stanford.edu/class/cs101/image-1-introduction.html (

Ukoliko želimo izvršiti kompresiju nad takvim slikama potrebno je efektivno stvoriti tri slike iz jedne – po jednu za nijanse crvene, zelene i plave boje koje će se kodirati posebno, na primjer, niz sljedećih pixela:

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

će postati tri različite sekvence:

R = (45, 45, 46, 46)

G = (76, 78, 79, 82)

B = (123, 123, 123, 123)

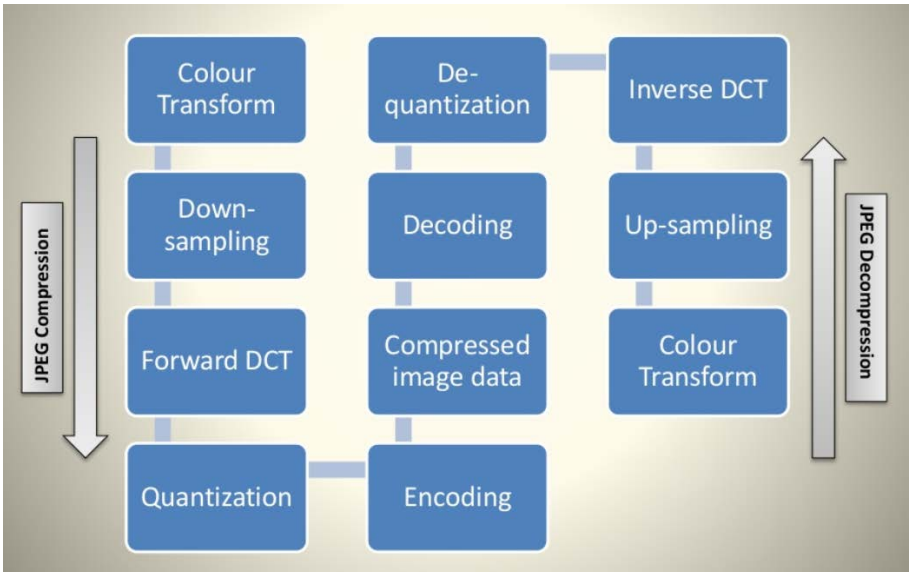
Prednost RLE algoritma je to što je vrlo jednostavan za implementaciju, no nedostatak je činjenica da je algoritam sve manje i manje efikasan što je slika kompleksnija. Prema tome, ovaj algoritam je koristan za jednostavne slike u kojima se pojavljuju samo crna i bijela boja ili općenito slike koje se sastoje od velikih jednobojnih prostora (line art, arhitekturski nacrti…).

3.2. JPEG

JPEG – Joint Photographic Experts Group je najzastupljeniji grafički format na webu. JPEG kompresija je lossy, no moguće je odabrati stupanj kompresije, gdje veći stupanj rezultira manjom veličinom datoteke no i lošijom kvalitetom slike. Prema tome, idealni stupanj kompresije onaj pri kojemu je pad razine kvalitete slike neprimjetan dok je veličina datoteke smanjena. Za razliku od RLE kompresije, JPEG kompresija ja najkorisnija nad slikama u boji i grayscale slikama – fotografijama, kompleksnim digitalnim slikama i slično. U takvim primjerima moguće je ostvariti omjer kompresije od čak 10:1 prije nego li dođe do primjetnog gubitka kvalitete slike.

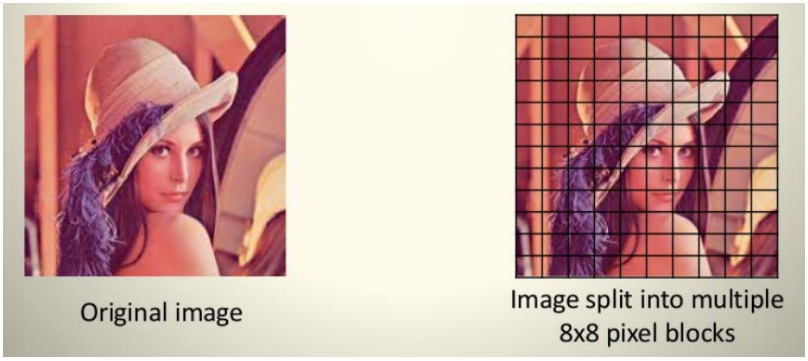
Postupak JPEG kompresije sastoji se od pet osnovnih koraka:

1. Splitting – podjela slike
2. Konverzija RGB prostora boja u YCbCr prostor boja
3. DCT (Discrete Cosine Transformation) transformacija
4. Kvantizacija
5. Entropy coding



Slika 8. Shema procesa JPEG kompresije, izvor: https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348

Splitting - Slika se dijeli u blokove pixela dimenzije 8 x 8



Slika 9. Podjela pixela u makroblokove, izvor: https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348

Konverzija RGB u YCbCr - JPEG pri kompresiji koristi YCbCr prostor boja. Ako znamo da je RGB također prostor boja koji se sastoji od različitih nijansi crvene, zelene i plave boje, istu logiku možemo primijeniti i za YCbCR prostor boja, ali sa drugih tri različitih komponenti. Y predstavlja „luminance“ odnosno svjetlinu bilo koje boje, ljudsko oko je osjetljivo na ovu komponentu. Cb predstavlja plave komponente relativno zelenim komponentama (color blueness), te Cr predstavlja crvene komponente relativno zelenim komponentama (color redness), ljudsko oko je manje osjetljivo na ove komponente; JPEG kompresija prema tome iskorištava razinu naše osjetljivosti kako bi eliminirala detalje slika koje su ljudskom oku nepotrebne.

Slika 10. RGB komponente, izvor http://www.bk.isy.liu.se/en/courses/tsbk06/material/

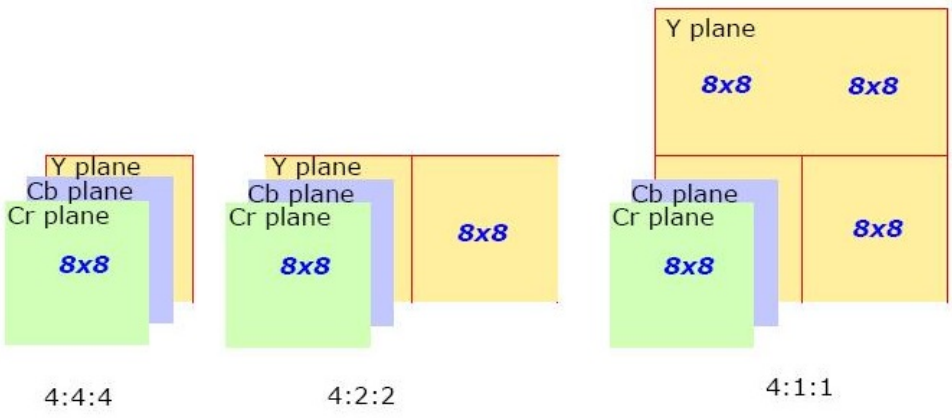
Slika 11. YCbCr komponente, izvor: http://www.bk.isy.liu.se/en/courses/tsbk06/material/

Konverzija koristi RGB ulazne vrijednosni za svaku komponentu opsega [0.0, 255.0] i transformira ju u YCbCr gdje komponente imaju opsege: Y [0.0, 255.0], Cb [-128.0, 127.0], Cr [-128.0, 127.0]. Jednadžba matrice za pretvorbu je prikazana na sljedećoj slici (Slika 12).

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

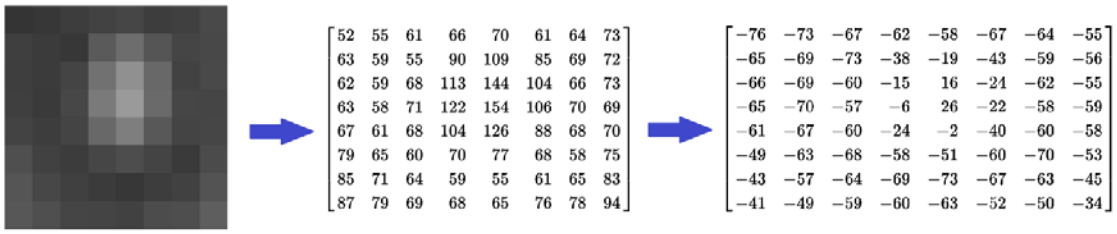
Slika 12. RGB-YCbCr konverzijska matrica, izvor: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdprfx/b550d1b5-f7d9-4a0c-9141-b3dca9d7f525

Ova konverzija omogućava nam da efektivno izvršavamo proces downsampling-a – smanjenje dubine boje slika. Pomoću downsampling-a iskorištavamo prije napomenutu osjetljivost odnosno manjak osjetljivosti ljudskog oka na plave boje tako da se slikama konvertiranim u YCbCr prostor boja smanjuje dubina boje Cr i Cb komponenata. Dakle svaki pixel zadržava vrijednost svoje svjetline (Y), dok se ostali pixeli (CbCr) grupiraju i u skladu s određenim pravila grupe poprimaju jednaku boju. Ukratko, Y vrijednost se uzima za svaki pixel, a Cb i Cr vrijednosti se uzimaju za blok pixela čija veličina ovisi o MCU formatu. MCU – Minimal Coded Unit predstavlja najmanju jedinicu koja se može kodirati.



Slika 13. struktura MCU za JPEG sampling formate, izvor: https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch6/ch6_image_downsampling.htm

Discrete Cosine Transformation – pretvara informacije spremljene u bloku (8 x 8) pixela iz prostorne domene u frekvencijsku domenu. Budući da ljudsko oko nije osjetljivo na komponente visokih frekvencija, takve komponente možemo tretirati kao da su redundantni, nepotrebni. Kako bi te komponente uklonili potrebno je sliku pretvoriti u frekvencijsku domenu što je glavni cilj DCT-a. Kako bi uopće mogli započeti proces, prvo je potrebno centrirati vrijednosti oko nule kako bi korespondirali funkciji kosinusa, taj proces se naziva „Zero-shift“. Kao primjer na slici prikazan je 8 x 8 blok pixela iz Y komponente čiji je inicijalni opseg [0, 255] na kojemu činimo Zero-shift; novi opseg je [-128, 127] te sve vrijednosti umanjimo za 128.



Slika 14. Zero-shift, izvor: https://abyx.be/post.php?id=12

Budući da je matrica slike dvodimenzionalna, potrebno je koristiti standardiziranu 2-D DCT formulu za JPEG kompresiju. Ako je ulazna matrica P(x,y) a transformirana matrica G(u,v), DCT formula za blokove od 8x8 pixela glasi:

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

Slika 15. 2D DCT formula, izvor: https://abyx.be/post.php?id=12

Transformirana matrica prikazana je na slici x. Sve vrijednosti transformirane matrice sada predstavljaju „utege“ (weights) koji određuju koja je u konačnom rezultatu uloga određene kosinusne funkcije.

$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Slika 16. Transformirana matrica, izvor: https://abyx.be/post.php?id=12

Kvantizacija – je proces gdje se skup vrijednosti mapira u manji skup vrijednosti kako bi se štedio prostor. To je u principu vrlo jednostavan proces, a postiže se tako da matrice koje su dobivene DCT transformacijom podijelimo sa standardiziranim kvantizacijskim JPEG matricama koje definiraju koji uteg treba biti podijeljen s kojim brojem. Te matrice su dizajnirane da podijele utege viših frekvencija s većim brojevima, jer više frekvencija imaju manji utjecaj na konačni rezultat.

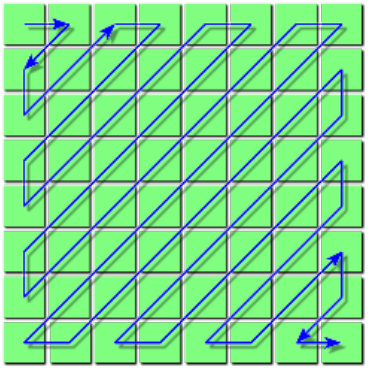
Kao primjer je prikazan proces kvantizacije matrice dobivene u prethodnom primjeru DCT transformacije, te se dijeli s JPEG kvantizacijskom matricom s postavkom kvalitete od 50%, te se rezultati zaokružuju na najbliži integer. Na primjer, element transformirane matrice [-61.20] dijelimo s odgovarajućim elementom kvantizacijske matrice (10) te kao rezultat dobivamo odgovarajući element konačne, kvantizirane matrice. [-61.20] / 10 = -6.12 -> -6. Isto vrijedi za sve ostale elemente matrice.

$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} / \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Slika 17. Kvantizacija, izvor: https://abyx.be/post.php?id=12

Svaka postavka kvalitete ima različitu kvantizacijsku tablicu, što je bolja kvaliteta, to su manji brojevi u kvantizacijskoj tablici s kojima vršimo dijeljenje.

Entropy coding – za razliku od ostalih koraka, entropy coding je lossless. Krećemo tako da našu 8 x 8 matricu u linearnu sekvencu integera (bitstream), no u ovom slučaju nije idealno da pratimo scan linije, već se koristi dijagonalni „zig-zag“ uzorak krećući od prvog elementa matrice, koji prolazi odnosno uvrštava sve elemente matrice završavajući s elementom u donjem desnom kutu. Kvantizirane vrijednosti poredane u bitstreamu se zatim kodiraju Huffmanovim algoritmom, istim principom kao i kod kompresije teksta.



Slika 18. Zig-zag uzorak za stvaranje bitstreama, izvor: https://abyx.be/post.php?id=12

4. Kompresija audio zapisa

Kao što se pri kompresiji slika uklanja redundancija s obzirom na ljudsko osjetilo vida, isto se postiže i kompresijom audio zapisa, uklanjaju se frekvencije ili zvukovi koje ljudski sluh ne prepoznaje ili su razlike dovoljno male da je razlika u kvaliteti neprimjetna. Algoritmi kompresije audio zapisa su softverski implementirani kao audio kodeci (audio codecs) te kao i kod algoritama za druge tipove podataka, mogu biti lossy ili lossless.

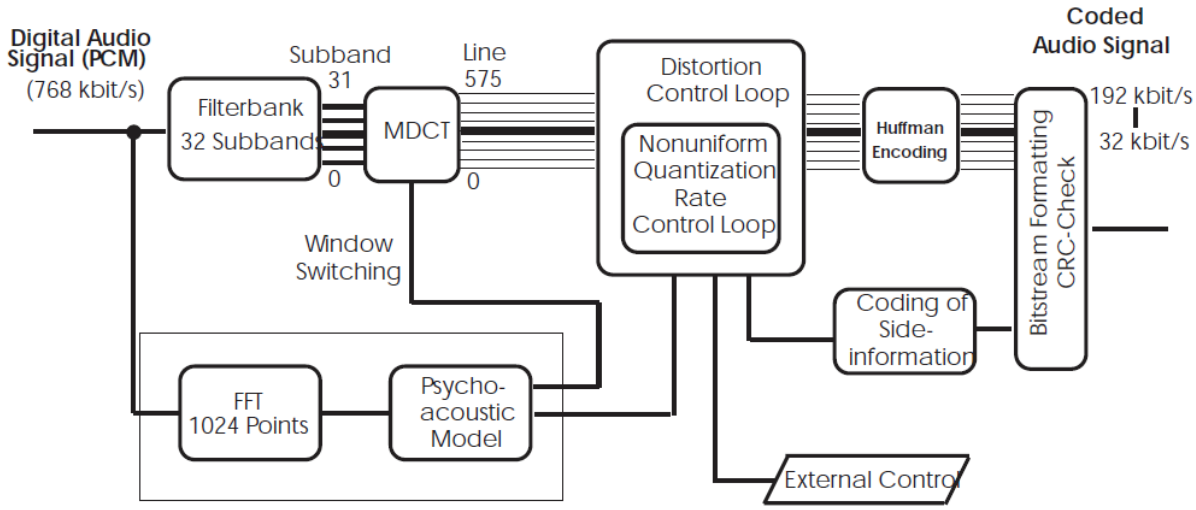
Neki od najzastupljenijih izvornih odnosno ne-kompresiranih formata su WAV (Waveform Audio File Format) i AIFF (Audio Interchange File Format). Microsoft i IBM su razvili WAV format te se on koristi na Microsoft Windows sustavima za sirove, ne-kompresirane audio zapise, dok je AIFF razvijen od strane Apple-a te se koristi za sirove podatke u Apple Macintosh sustavima. Oba formata esencijalno postižu jednake rezultate što se tiče same audio reprodukcije, no AIFF ima bolju podršku za meta-podatke (naslovnice albuma, podaci o pjesmi i izvođaču).

Najzastupljeniji lossless kodeci su FLAC (Free Lossless Audio Codec) i ALAC (Apple Lossless Audio Codec). Kao i u primjeru izvornih formata, ovi kodeci imaju gotovo jednake rezultate – kompresiranjem izvornih formata smanjuju većinu datoteke za približno 50%, no budući da FLAC nije podržan na Apple-ovim sustavima, razvili su svoju alternativu – ALAC.

Najpopularniji lossy kodeci su MP3 (MPEG-1 Audio Layer 3) i AAC (Advanced Audio Coding). U ovom slučaju AAC je dizajniran kako bi bio nasljednik MP3 kodeka, te iako je standardiziran u određenom broju uređaja te daje bolje rezultate, MP3 je i dalje daleko zastupljeniji.

4.1. MP3

MP3 je vrlo popularan i zastupljen lossy audio kodek, upravo zato što je kompresija vrlo blizu optimalne u odnosu na potrebe svakodnevnih korisnika. Omjer kompresije je iznimno velik (približno 10:1), dok je kvaliteta zvuka nedvojbeno lošija od sirovih ili lossless kodeka no za većinu ljudi ta razlika nije toliko primjetna niti bitna. Zamjena veće kvalitete za memorijski prostor se većini korisnika isplati.



Slika 19. Blok dijagram MP3 encodera, izvor: BRANDENBURG K. *MP3 and AAC explained*

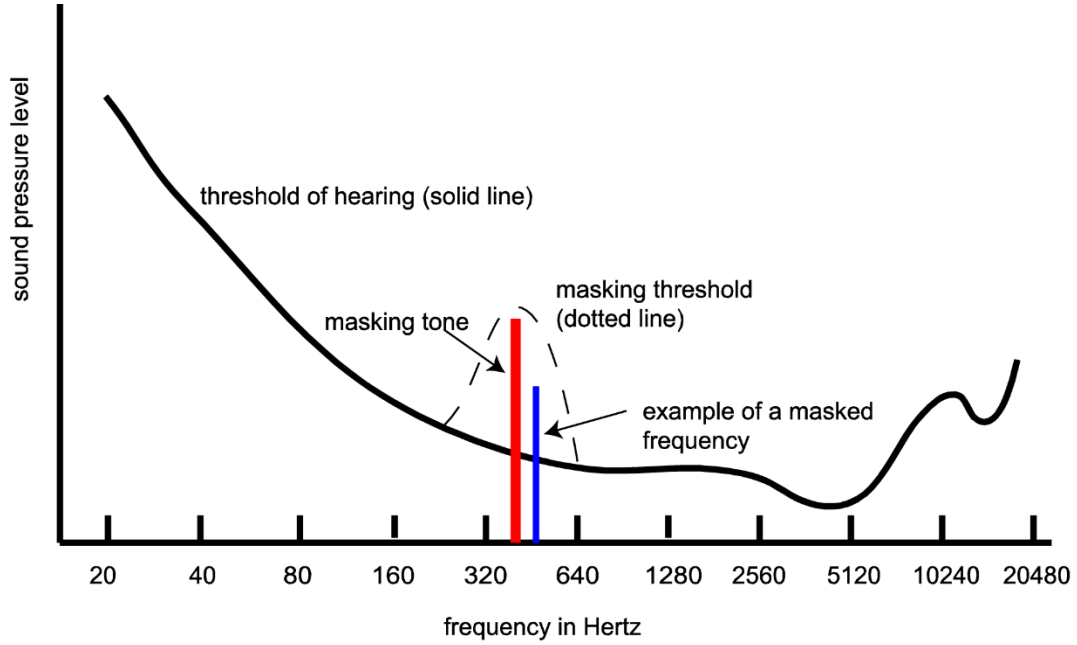
1. Filter banka

Filter banka se koristi za dekompoziciju ulaznog signala u podskupove spektralnih komponenti u domeni vrijeme/frekvencija. Zajedno sa odgovarajućom filter bankom u dekoderu stvara se sustav analize/sinteze.

Filter banka koja se koristi u MP3 kompresiji smatra se hibridnom filter bankom. Stvorena je kaskadno od dvije različite filter banke. Prva je polifazna - podjela svakog polifaznog frekvencijskog pojasa (polyphase frequency band) u 18 kvalitetnijih potpojasa (subbands) povećava potencijal uklanjanja redundancije. Druga filter banka je modificirana verzija DCT transformacije, MDCT (Modified Discrete Cosine Transform).

2. Perceptivni Model

Perceptivni (psihoakustički) model se koristi kako bi se procijenila granica maskiranja (masking threshold) tonova iznad granice ljudskog sluha, te kako bi se ti tonovi ublažili – maskirali bez percepcijske razlike ljudskom sluhu između originalnog i maskiranog tona.



Slika 20. Granica maskiranja, izvor: http://digitalsoundandmusic.com/5-3-8-algorithms-for-audio-companding-and-compression/

3. Kvantizacija i kodiranje

Spektralne komponente se kvantiziraju i kodiraju s ciljem održavanja zvuka ispod granice maskiranja. Ovaj korak se može izvesti na vrlo različite načine, od jednostavnog sažimanja blokova do sustava analize/sinteze s dodatnom kompresijom.

U MP3 kompresiji se koristi sustav dvaju ugniježđenih petlji za iteraciju. Kvantizacija se izvršava pomoću power-law kvantizera (power-law quantizer). Na taj način, veće vrijednosti se automatski kodiraju s manjom preciznošću i oblikovanje zvuka je dio procesa kvantizacije

Nakon kvantizacije, kvantizirane se vrijednosti kodiraju Huffmanovim algoritmom. Da se proces kodiranja prilagodi raznim statistikama specifične glazbe koja se kodira, odabire se optimalna Huffmanova tablica. Kako bi se dobila još bolja prilagodba, različite Huffman tablice se mogu koristiti za različite dijelove spektra.

Budući da se oblikovanje zvuka mora izvršiti kako bi se on održao ispod granice maskiranja, prije koraka kvantizacije se primjenjuju globalna vrijednost gain-a, koja određuje veličinu koraka kvantizacije, i faktori razmjera (scalefactors), koji određuju faktore oblikovanja zvuka za svaki scalefactor pojas. Proces pronalaska optimalnog gain-a i faktora razmjera se izvršava pomoću dvije ugniježđene petlje na „analysis-by-synthesis“ način.

- Unutarnja petlja (rate loop)

  Huffmanov kod zadaje kratka kodna imena kvantiziranim vrijednostima. Ako broj bitova koji su nastali procesom kodiranja premašuje broj dostupnih bitova za zadani blok podataka, to se može prepraviti podešavanjem globalnog gain-a što rezultira većim korakom kvantizacije odnosno manjim kvantiziranim vrijednostima. Ovaj proces se ponavlja s različitim veličinama koraka kvantizacije sve dok Huffmanovo kodiranje ne rezultira brojem bitova koji je manji od broju dostupnih bitova.

- Vanjska petlja (noise control loop)

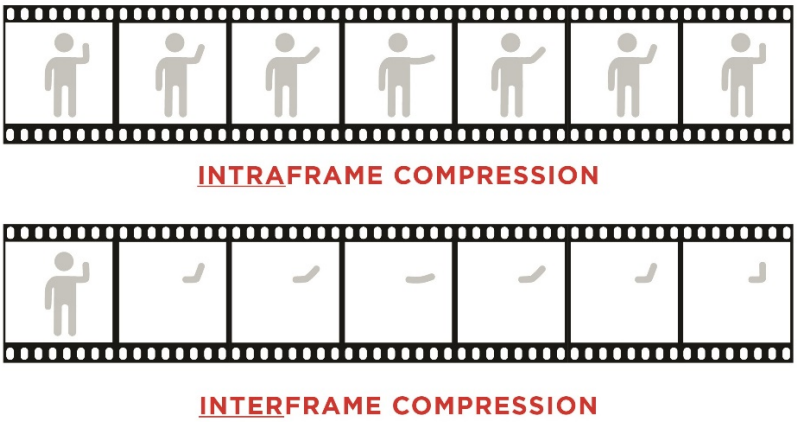
  Kako bi se kvantizacijski zvuk oblikovao prema granici maskiranja, faktori razmjera se primjenjuju svakom scalefactor pojasu. Sustav kreće sa zadanim faktorom 1.0 za svaki pojas. Ako kvantizacijski zvuk u određenom pojasu premašuje granicu maskiranja, faktor razmjera za taj pojas se podešava da se kvantizacijski zvuk smanji. Budući da postizanje slabijeg kvantizacijskog zvuka zahtijeva veći broj kvantizacijskih koraka, rate petlja se ponavlja svaki put kada se koriste novi, podešeni faktori razmjera – rate petlja je ugniježđena u noise control petlji. Noise control petlja se izvršava sve dok stvarni zvuk nije ispod granice maskiranja za svaki scalefactor pojas.

5. Kompresija video zapisa

Video zapisi su vrsta multimedijskih sadržaja koji kombiniraju sekvencu slika kako bi stvorili efekt pokretne slike. Osim vizualnog dijela, video zapisi često sadrže i audio komponente koje odgovaraju prikazanim slikama.

Prostorno (intra-frame) kodiranje je pristup kojime se vrši kompresija nad svakim frameom videa pojedinačno. JPEG format slika je primjer prostornog kodiranja. Frameovi koji se u video zapisima kodiraju na ovaj način nazivaju se I-frameovi i kodiraju se pojedinačno, bez ikakve reference na prethodni ili sljedeći frame. To su uglavnom frameovi na početku novih scena u video zapisima.

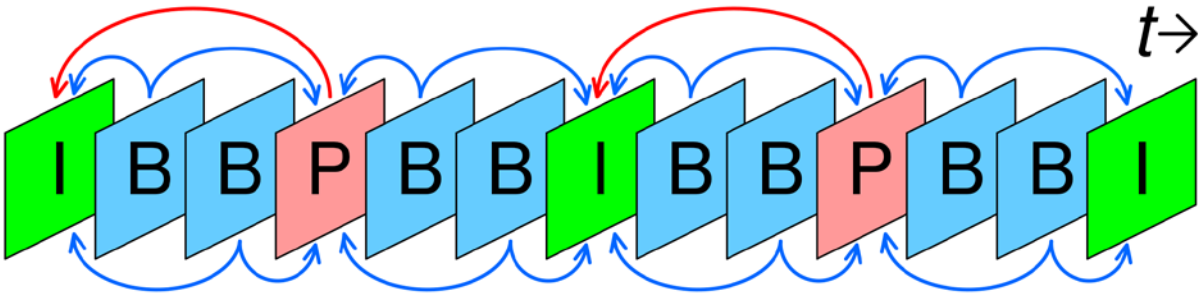
Vremensko (inter-frame) kodiranje je pak pristup kojime se vrši kompresija s obzirom na odnos između frameova. Video sadržaji uglavnom sadrže sekvence frameova gdje se neke sekcije tijekom videa uopće ne mijenjaju, ili se mijenjaju suptilno (pozadine).



Slika 21. Intraframe i interframe kodiranje, izvor: https://support.biamp.com/General/Video/Video_Basics

U takvim slučajevima, cilj kompresije je opet ukloniti redundanciju ponavljajućih podataka. Kodeci video kompresije sadrže instrukcije koje manipuliraju blokove pixela iz framea u frame – mogu ostati netaknuti, pomaknuti se na drugi dio framea, rotirati, promijeniti boju, promijeniti cijeli blok i slično. Frameovi koji izvršavaju te instrukcije nazivaju se P-frameovi i sadrže otprilike upola manje podataka od I-frameova. Neki kodeci koriste i treći tip framea, B- frame. Dok P-frameovi stvaraju predikcije bazirane na prethodnom frameu, B-frame koristi metodu predikcije baziranu i na prethodnom i

na sljedećem frameu. B-frameovi sadržavaju otprilike četvrtinu podataka u usporedbi sa I-frameovima.



Slika 22. Group Of Pictures struktura, izvor: https://en.wikipedia.org/wiki/Inter_frame#/media/File:IBBPBB_inter_frame_group_of_pictures.svg

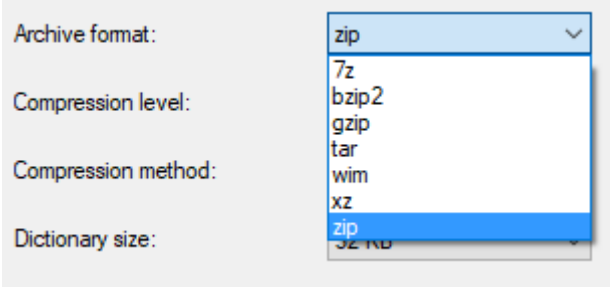
Najpopularniji i najzastupljeniji kodek video kompresije je H.264 koji je ujedno i jedan od najefikasnijih kodeka danas te se smatra industrijskim standardom, moguća je lossy i lossless kompresija te nudi vrlo dobar omjer kompresije zadržavajući visoku kvalitetu reprodukcije. 2013. godine stvoren je njegov nasljednik, H.265 koji je nadogradnja na H.264 u svakom smislu, postiže veći omjer kompresije te je moguće kodirati video sadržaje većih rezolucija, do 8K (8192 x 4320). Iako H.265 postiže bolje rezultate, industrija ga usvaja sporijim tempom zbog problema vezanih za licenciranje no izgledno je da će s vremenom preuzeti titulu industrijskog standarda.

6. Arhivske datoteke

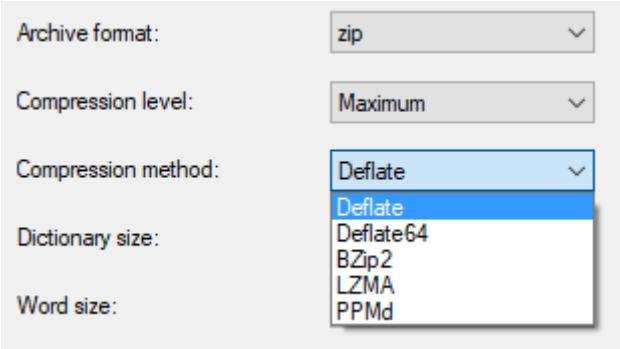
Alati za kompresiju datoteka (7Zip, WinZip, WinRAR…) pružaju mogućnost kompresije jedne ili više datoteka ili mapa u jednu datoteku - arhivu, s jednim file extensionom, ovisno o željenom formatu. Arhive koriste lossless kompresiju, te se datoteke unutar arhive mogu pojedinačno dekomprimirati i „vaditi“ iz arhive.

Ukoliko korisnici žele pregledati datoteku unutar arhive, primjerice neki video zapis, to je moguće izvršiti putem prozora odabranog arhivskog softvera, no u tom slučaju je datoteku potrebno dekompresirati i fizički pohraniti iako korisnici to nisu specificirali. Na Windows operativnim sustavima se u takvim slučajevima datoteke spremaju u zadani Windows temp folder koji sadrži privremene datoteke. Nakon zatvaranja datoteke ona se briše iz temp foldera.

Različiti arhivski softveri podržavaju kompresiju više formata, a svaki format može koristiti određene algoritme ili metode kompresije.



Slika 23. 7Zip - podržani formati arhiva, izvor: vlastiti izvor



Slika 24. 7Zip - podržane metode kompresije zip formata, izvor: vlastiti izvor

7. Zaključak

Razvojem informatike veličina neobrađenih podataka postepeno postaje sve veća i veća. Iako se paralelno razvijaju sve bolji uređaji za pohranu podataka, sa sve većim kapacitetom, određeni tipovi podataka su i dalje preveliki kako bi se efikasno pohranjivali ili prenosili. Prema tome, razvijeni su algoritmi i tehnike kompresije podataka čiji je cilj kodiranje podataka na taj način da se postigne smanjenje veličine datoteka – rezultirajući efikasnijom pohranom i prijenosom.

Zbog prirode određenih vrsti datoteka, razvijeni su načini kompresije bez gubitka podataka, gdje se pri kodiranju statističke metode modeliranja za smanjenje ponavljajućih informacija u datoteci. No u slučajevima kada je dopušteno odbacivanje određenih podataka, koriste se tehnike lossy kompresije – većina multimedijskih sadržaja koristi lossy kompresiju gdje se esencijalno iskorištavaju ljudska osjetila kako bi se smanjila redundancija odnosno odbacili podaci koji su za nas neprimjetni. Lossy kompresija zvuka odbacuje zvukove frekvencija koji ljudski sluh slabo čuje ili ne čuje uopće, kompresija slike odbacuje boje na koje je ljudski vid manje osjetljiv, video kompresija često koristi kompresirane slike kao frameove, te na mjestima gdje je to moguće koristi sadržaj istog framea više puta.

Različite vrste sadržaja često koriste kompresijske tehnike koje su specifične toj vrsti sadržaja, no većina kao dio cjelokupnog procesa kompresije koristi barem jedan ili kombinaciju nekoliko osnovnih algoritama kompresije – u stadiju procesa kada su podaci uređeni na takav način da ti algoritmi njima mogu baratati. U ovome radu objašnjeni su neki od osnovnih algoritama kompresije, te postupci kompresije određenih formata raznih tipova multimedijskih sadržaja.

Literatura

Knjige:

1. Shi, Yun i Sun, Huifang (1999.) - Image and Video Compression for Multimedia Engineering. Fundamentals, Algorithms and Standards
2. Miano, John (1999.)- Compressed image file formats JPEG, PNG, GIF, XBM, BMP
3. Sayood, Khalid (2017.) - Introduction to Data Compression
4. Wootton, Cliff (2005.) - A practical guide to video and audio compression
5. Salomon, David (2006.) - Data compression - The Complete Reference – Fourth Edition
6. Richardson, Iain. (2010.) - The H.264 Advanced Video Compression Standard

Članci:

1. Brandenburg, Karlheinz (2001.) – MP3 and AAC explained. Dostupno na: https://www.semanticscholar.org/paper/MP3-and-AAC-Explained-Brandenburg/3898f5b3c74f1311bee5750b6c22650ae23b210a [pristupljeno 13. rujna 2019.]

Internet izvori:

1. Lecture 8-9 (colour, transform coding, subband coding). Dostupno na: http://www.bk.isy.liu.se/en/courses/tsbk06/material/ [pristupljeno 13. rujna 2019.]
2. Khan Academy – Simple Image Compression. Dostupno na: https://www.khanacademy.org/computing/ap-computer-science-principles/computers-101/file-compression/a/simple-image-compression [pristupljeno 13. rujna 2019.]
3. ABYX – JPEG compression. Dostupno na: https://abyx.be/post.php?id=12 [Pristupljeno 13. rujna 2019.]
4. The Crazy Programmer – Huffman Coding Algorithm With Example. Dostupno na: https://www.thecrazyprogrammer.com/2014/09/huffman-coding-algorithm-with-example.html [pristupljeno 13. rujna 2019.]

5. FileFormat.Info – Run-Length Envoding (RLE). Dostupno na: https://www.fileformat.info/mirror/egff/ch09_03.htm [pristupljeno 13. rujna 2019.]
6. SlideShare – K. M. Aishwarya – JPEG Image Compression. Dostupno na: https://www.slideshare.net/AishwaryaKM1/jpeg-image-compression-56894348 [pristupljeno 13. rujna 2019.]
7. Microsoft – Color Conversion (RGB to YCbCr). Dostupno na: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdprfx/b550d1b5-f7d9-4a0c-9141-b3dca9d7f525 [pristupljeno 13. rujna 2019.]
8. Image Downsampling. Dostupno na: https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch6/ch6_image_downsampling.htm [pristupljeno 13. rujna 2019.]
9. Ditigal Sound And Music – Algorithms for Audio Companding and Compression. Dostupno na: http://digitalsoundandmusic.com/5-3-8-algorithms-for-audio-companding-and-compression/ [pristupljeno 13. rujna 2019.]
10. Biamp – Video Basics. Dostupno na: https://support.biamp.com/General/Video/Video_Basics [pristupljeno 13. rujna 2019.]
11. Microsoft – How does JPEG actually work? Dostupno na: https://blogs.msdn.microsoft.com/devdev/2006/04/12/how-does-jpeg-actually-work/ [pristupljeno 13. rujna 2019.]
12. Codex – An Overview of H.264 Advanced Video Coding. Dostupno na: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/ [pristupljeno 13. rujna 2019.]
13. Youtube – Raul Fraile: How GZIP compression works | JSConf EU 2014. Dostupno na: https://www.youtube.com/watch?v=wLx5OGxOYUc [pristupljeno 13. rujna 2019.]
14. Vcodex – An Overview of H.264 Advanced Video Coding. Dostupno na: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/ [pristupljeno 13. rujna 2019.]

Popis slika

Slika 1. Lossy i lossless kompresija……3

Slika 2. Huffmanov algoritam, drugi korak……8

Slika 3. Huffmanov algoritam, dovršeno stablo……8

Slika 4. Kompresija slike……10

Slika 5. Prikaz slike prije RLE kompresije……11

Slika 6. Prikaz slike nakon RLE kompresije……12

Slika 7. RGB vrijednosti pixela……13

Slika 8. Shema procesa JPEG kompresije……14

Slika 9. Podjela pixela u makroblokove……15

Slika 10. RGB komponente……15

Slika 11. YCbCr komponente……15

Slika 12. RGB-YCbCr konverzijska matrica……16

Slika 13. Struktura MCU za JPEG sampling formate……16

Slika 14. Zero-shift……17

Slika 15. 2D DCT formula……17

Slika 16. Transformirana matrica……17

Slika 17. Kvantizacija……18

Slika 18. Zig-zag uzorak za stvaranje bitstreama……19

Slika 19. Blok dijagram MP3 encodera……21

Slika 20. Granica maskiranja……22

Slika 21. Intraframe i interframe kodiranje……24

Slika 22. Group Of Pictures struktura……25

Slika 23. 7Zip - podržani formati arhiva……26

Slika 24. 7Zip - podržane metode kompresije zip formata……26

Sažetak:

Kapacitet prostora za pohranu uređaja predstavlja jedan od ključnih faktora u razvoju informatike. S potrebom za većom količinom kapaciteta, dok se s jedne strane razvijao hardver sve većeg kapaciteta, s druge strane su nastale tehnike kompresije podataka te je počeo razvoj sve boljih i efikasnijih tehnika za razne tipova podataka.

U ovom radu obrađene su osnove kompresije podataka, te objašnjeni neki od algoritama kompresije podataka na primjerima kompresije teksta. Pojedinačno su objašnjeni određeni formati ostalih nekoliko vrsta multimedijskih sadržaja te su objašnjeni koncepti ili prikazani postupci koje koriste pri kompresiji.

Ključne riječi: Algoritmi kompresije podataka, kompresija teksta, kompresija slika, JPEG, kompresija audio zapisa, MP3, kompresija video zapisa, H.264

Abstract:

Device storage capacity represents one of the key factors in the development of information technology. With the necessity for more capacity, while one side was developing hardware with more storage capacity, the other side invented data compression techniques and started developing better and more efficient techniques for various types of data.

This paper deals with the basics od data compression and explains some of the data compression algorithms using text compression examples. Specific formats of the several other types of multimedia content are explained individually as well as the concepts or methods they use for compression.

Key words: Data compression algorithms, text compression, image compression, JPEG, audio compression, MP3, video compression, H.264