Univerzitet u Beogradu

Matematički fakultet

Master teza

UPOREĐIVANJE STEPENA KOMPRESIJE KOD ALGORITAMA SA GUBICIMA I BEZ GUBITAKA

Ivona Brajević

Mentor
prof. dr Milan Tuba

Beograd, 2008. godine

SADRŽAJ

1. Uvod
2. Kompresija i teorija informacija
3. Algoritmi kompresije bez gubitaka
   - 3.1 Kôd, kodiranje i dekodiranje
   - 3.2 Izbor dobrog kôda
   - 3.3 Hofmanov algoritam
   - 3.4 LZW algoritam
   - 3.5 Kompresija zvuka bez gubitaka
   - 3.6 Kompresija slika bez gubitaka i formati grafičkog fajla
4. Multimedija
   - 4.1 Digitalizacija medija
   - 4.2 Proizvodnja i dostavljanje multimedija
   - 4.3 Razvoj multimedija
5. Algoritmi kompresije sa gubicima
   - 5.1 Kompresija zvuka algoritmima sa gubicima
   - 5.2 Kompresija slika algoritmima sa gubicima
     - 5.2.1 Kvalitet i veličina grafičkog fajla
     - 5.2.2 JPEG standard
     - 5.2.3 Sekvencijalni algoritam kodiranja
     - 5.2.4 Dekodiranje
   - 5.3 Kompresija video podataka
     - 5.3.1. Razvoj MPEG standarda
     - 5.3.2. Kompresija individualnih frejmova
     - 5.3.3. MPEG-1 algoritam za kompresiju video podataka
6. Stepen kompresije kod algoritama sa gubicima i bez gubitaka
7. Zaključak

SAŽETAK

Ovaj rad razmatra upoređivanje stepena kompresije kod algoritama sa gubicima i bez gubitaka. Razlog razvoja algoritama kompresije je efikasno skladištenje podataka na disk i njihov prenos kroz komunikacione kanale. Algoritmi kompresije bez gubitaka se koriste kod podataka koje bi gubici učinili neupotrebljivim, uglavnom kod tekstualnih podataka, a *ponekad za potrebe kompresije zvuka i slike, u svakom slučaju kada gubici nisu dozvoljeni. Stepen kompresije koji se postiže algoritmima bez gubitaka najčešće nije dovoljan za smeštanje i prenos digitalnog zvuka, slike i filma, jer je veličina ovih podataka ogromna. U tim slučajevima uobičajeno se koriste algoritmi sa gubicima. Može se pokazati da dozvoljavanjem jako malih gubitaka u fajlu koji se kompresuje, mogu da se postignu ogromani stepeni kompresije, a da po dekompresiji gubici ne budu ni primetni za naša čula. Razvoj algoritama sa gubicima je od izuzetnog značaja, jer predstavlja pokretač multimedijalne revolucije, koja je imala presudan uticaj na povećanje obima i načina korišćenja kako računara, tako i samih informacija.*

U ovom radu prikazani su rezultati eksperimenta u kome su upoređeni stepeni kompresije digitalnih slika kod algoritama bez gubitaka i sa gubicima. Za potrebe ovog eksperimenta korišćen je softverski alat GIMP 2.4.6.

1. Uvod

Kompresije podataka počele su ozbiljnije da se razvijaju osamdestih godina prošlog veka, jer su tada memorija i disk bili strahovito skupi i mnogo se skupocenog prostora trošilo na skladištenje podataka u razvijenom, tj. nekompresovanom obliku. Ideja kompresije podataka je na prvi pogled neprirodna. Međutim, koristi se činjenica da podaci koje ljudi koriste sadrže mnogo nepotrebnog viška, a po teoriji informacija može tačno da se izračuna koliko iznosi taj višak i odbaci ono što je redundantno. Algoritmi kompresije zasnivaju se upravo na toj činjenici. Fajlovi se u komprimovanom obliku čuvaju na disku i šalju kroz komunikacione kanale. Da bi se mogli koristiti, moraju se prethodno dekompresovati, tj. vratiti u prvobitan oblik. Claude E. Shannon je u svom radu iz 1948. godine "Matematička teorija komunikacije" formulisao teoriju kompresije podataka. Sama teorija ne pokazuje kako dizajnirati ili implementirati ove algoritme. Ipak, ova teorija pruža osnovna uputstva kako doći do optimalnih performansa.

Algoritmi kompresije se mogu podeliti u dve kategorije: algoritmi kompresije bez gubitaka i algoritmi kompresije sa gubicima. Kod kompresije bez gubitaka dobija se fajl koji će posle dekompresije biti identičan originalu. Postoji jako mnogo algoritama kompresije bez gubitaka, a dva koja se najčešće upotrebljavaju i koja su u ovom radu detaljnije opisana su Hofmanov kôd i LZW (Lempel Ziv Welch) algoritam. Oba algoritma rade na principu da karaktere ili niz karaktera koji se češće pojavljuju, kodiraju kraćim kodnim rečima. Najpoznatiji program za kompresiju podataka bez gubitaka je ZIP, a koristi se najčešće za kompresovanje tekstualnih podataka.

Stepen kompresije koji se može postići algoritmima bez gubitaka nije dovoljan za smeštanje na disk i prenos digitalnog zvuka, slike i filma kroz komunikacione kanale.

Osnovni problem je veličina ovih fajlova. Iz ovih razloga počeli su da se razvijaju algoritmi sa gubicima, koji pri kompresiji dozvoljavaju gubitke u fajlu, tako da dekompresovani fajl neće biti jednak originalu. Kompresija sa gubicima je pogodna za podatke koji potiču od digitalizovanih slika ili zvuka, jer je digitalna prezentacija već i sama samo aproksimacija. Može se pokazati da dozvoljavanjem malih gubitaka može da se postige visok stepen kompresije, a da gubici neće biti primetni, ili neće biti značajni za potrebe naših čula. Ipak, ako se ovakvi algoritmi za kompresiju sa gubicima i naknadnu dekompresiju vrše uzastopno na istim podacima, može doći do značajnog pada kvaliteta podataka. U svakom slučaju, razvoj ovih algoritama je revolucionaran i zaslužan za eksponencionalni razvoj multimedija.

Kod kompresije zvuka pojava MP3 formata dovela je do multimedijalne revolucije. Zahvaljujući saradnji dveju internacionalnih organizacija za standarde ISO i CCIT, digitalne slike su postale prirodni tip podataka, jer je stvoren JPEG standard za kompresiju digitalnih slika. JPEG se često koristi zato što zahtevi za multimedijom i kompresijom slika rastu eksponencionalno. Međutim, JPEG algoritam nije jedinstven standard, već skup pravila i preporuka sa mnogo podesivih parametara. Baziran je na diskretnoj kosinusnoj transformaciji, koja je najjednostavnija za hardversku implementaciju i primenljiv je na širok spektar digitalnih slika, tj. nije ograničen rezolucijom slike, sistemom prezentacije boja, kompleksnošću scene, itd. JPEG se koristi samo za kompresiju statične slike (jedna slika), ali je stvoren sličan standard MPEG koji se bavi kompresijom pokretnih slika (niza statičnih slika). Od algoritama kompresije sa gubicima u ovom radu detaljnije se opisuju JPEG i MPEG algoritmi.

Eksperiment upoređivanja stepena kompresije primenom algoritama sa gubicima i bez gubitaka izvršen je na slikama, iz razloga što je rezultate najjednostavnije prikazati kompresijom ovih podataka[1].

[1] U ovom radu reči kodiranje, kompresovanje i komprimovanje, kao i dekodiranje, dekompresovanje i dekomprimovanje se koriste kao sinonimi.

2. Kompresija i teorija informacija

Kompresija podataka predstavlja način da se ista informacija zapiše sa manje zauzetog prostora na disku. Pod stepenom kompresije podrazumeva se odnos veličina kompresovanog i originalnog fajla. Kompresija je oblast koja ne potiče direktno iz računarstva, već više iz matematike.

Teorijska osnova kompresija je teorija informacija. Teorija informacija je deo matematičke statistike koja se bavi pojmom informacije i njemu suprotnog pojma entropije. Ljudi uopšte imaju neku intuitivnu predstavu šta je to informacija, međutim u teoriji informacija taj pojam je prebačen u matematičku disciplinu, što znači da može da se kvantifikuje, tj. precizno računa i dokazuje. Polazi se od praktičnih situacija koje se dešavaju u svetu i koje u teoriji informacija treba matematički da se obrade. Ta teorija se polako, aksiomatski izgrađuje. Pojam opita se uzima za osnovni pojam i primer najelementarnijeg opita na kome matematički jasno možemo razumeti pojam informacije bi bio bacanje novčića. Imamo dva jednakoverovatna ishoda opita, pašće ili pismo ili glava. Pre bacanja novčića mi smo u neznanju, a nakon bacanja tačno možemo videti šta je palo, znači dobijemo neko znanje. Razlika između znanja posle eksperimenta i neznanja pre eksperimenta je informacija koju smo dobili kroz taj eksperiment. Znači, znanje koje smo dobili kroz opit predstavlja pojam informacije, a neznanje pre tog opita predstavlja pojam entropije, tj. informacija je entropija sa negativnim predznakom. Zatim se definiše bit kao osnovna jedinica za merenje informacije. Jedan bit je jedna binarna cifra, znači može imati vrednost nula ili jedan. U opitu bacanja novčića pašće ili pismo ili glava, a dobijenu informaciju zapisujemo npr. na sledeći način: nula ako je palo pismo, jedinica ako je pala glava. Pri komplikovanijim eksperimentima dobijene informacije zapisujemo na sličan način pomoću više binarnih cifara.

Dalje, sasvim prirodno, posmatrajući kako to izgleda u svetu, postavljaju se normalni zahtevi za informaciju i razmatra se kako to matematički može da se iskaže. Npr. može se od informacije očekivati da pri izvođenju nezavisnih opita zbir informacija bude jednak informaciji zbira. Pri bacanju dva novčića mi dobijamo dva bita informacije, a istu količinu informacije bismo dobili da smo jedan novčić bacali dva puta. Zatim, normalno je zahtevati da opit koji ima više mogućih ishoda ima veću entropiju. Ako opit ima dva ishoda naše neznanje pre eksperimenta nije veliko, znamo da će se dogoditi ili jedan ili drugi, a ako opit ima mnogo ishoda neznanje pre eksperimenta je veće i samim tim informacija koju dobijamo nakon njegovog izvršenja je veća. Pri bacanju kockice imamo šest mogućih ishoda. Samim tim naše neznanje pre bacanja je veće nego u slučaju novčića. A ako npr. za eksperiment uzmemo lutriju, broj mogućih ishoda je ogroman i naše neznanje pre eksperimenta je ogromno. Takođe, je sasvim prirodno očekivati da će entropija biti najveća ako su ishodi opita jednakoverovatni, jer je tada početno neznanje najveće. U slučaju nejednakoverovatnih ishoda nekog opita mi sa većom verovatnoćom možemo pretpostaviti šta će se dogoditi, znači naše neznanje pre eksperimenta bi u tom slučaju trebalo da bude svakako manje.

U teoriji informacija to su samo neki od uslova koji se prirodno zahtevaju od informacije, a samim tim i od entropije eksperimenta, a zatim se traži matematička funkcija koja te uslove zadovoljava. Interesantno je da se matematički može dokazati da postoji tačno

jedna funkcija koja zadovoljava sve tražene uslove. Ako imamo opit sa mogućim ishodima $x_1,x_2,...,x_n$ i redom njihovim verovatnoćama $p_1,p_2,...,p_n$, može se pokazati da je funkcija

$$-\sum_{i=1}^{n} p_i \log_2 p_i$$

jedinstvena funkcija za izračunavanje količine informacije koja zadovoljava sve te zahteve. Količinu entropije računamo pomoću iste funkcije, samo bez znaka minus. Na taj način rešen je problem merenja količine informacije, što će se na razne načine dalje koristiti i predstavljati teorijsku osnovu kompresije podataka.

Kao što je već pomenuto, kompresija podataka predstavlja način da se ista informacija zapiše sa manje zauzetog prostora na disku, i sasvim prirodno se postavlja pitanje kako je to uopšte moguće. Pored teorijske osnove kompresije koristi se činjenica da su svi podaci redundantni. Računarski programi, pisani tekstovi, snimljen zvuk, fotografija ili film, jednom rečju sve što bi mogli da uzmemo kao podatak, u sebi sadrži ogromne količine beskorisne informacije, odnosno ponavljanja ili redundantnosti. Npr. stari Latini su koristili metod pisanja reči sa izostavljanjem samoglasnika, iako oni predstavljaju polovinu ukupnog teksta. Iako na prvi pogled deluje da je bitna informacija izgubljena, pokaže se da sve ostane čitljivo, a to je upravo moguće jer je naš jezik napravljen tako da u sebi ima viška. Pored jezika, takav je i vizuelni svet, odnosno sve što je nastalo interakcijom čoveka i prirode, tj. na prirodan način. Jasno je da su u interakciji čoveka i prirode stvari nužno po sebi nastale redundantne, zbog toga što ljudi u prirodi ne bi mogli da prežive kao matematički aparati. Npr. upravo zbog toga što u sebi sadrže viška, ljudske reči ne liče jedna na drugu i na taj način nam omogućavaju da se razumemo u različitim situacijama. Kada bi ljudski jezik bio matematički efikasan ljudi ne bi mogli da se sporazumeju i izumrli bi.

Sa jedne strane imamo podatke koji su nastali u prirodi, na prirodan način, i koje treba da prebacimo na računar, koji je elektronski, matematički uredjaj. Sa druge strane, ono što bi nastalo u računarskom svetu bilo bi podložno matematičkim zakonima i bilo bi drugačije i efikasnije organizovano. Tu dolazi do sukoba. Memorija i disk su nekada bili jako skupi i mnogo se prostora trošilo zato što se previđala činjenica da se podaci upisuju u računar na neprirodan način. Svi podaci su se upisivali u onoj formi u kojoj su oni nastali u prirodi, a taj zapis je za računar neprirodan. Čak da su memorija i disk bili znatno jeftiniji i razvijeniji, kao što su danas, prenos podataka u razvijenom obliku bi predstavljao veliki problem pri komunikacijama, jer bi trošio mnogo vremena.

Upravo iz tih razloga, osamdesetih godina prošlog veka, počelo se ozbiljnije razmišljati o kompresijama, bez obzira što je sama ideja kompresije na prvi pogled neprirodna. Koristi se činjenica da su podaci koje ljudi koriste redundantni, da forma koja čoveku odgovara sadrži mnogo nepotrebnog viška, i da po teoriji informacija može da se izračuna tačno koliko iznosi taj višak. Pokazuje se da ono što smo ranije zapisivali na disk u hiljadu karaktera može da se sabije u sto karaktera kada bi destilisali čistu informaciju, a bacili ono što je redundantno.

Algoritmi kompresije se upravo na tome i zasnivaju. S obzirom na sadržaj podataka i namene fajla, danas postoji veliki broj algoritama kompresije. Svima njima je zajedničko da se podaci u fajlu organizuju na način komplikovaniji od načina na koji su organizovani ako kompresije nema, ali se postižu uštede u memorijskom prostoru koje ni u kom slučaju

nisu zanemarljive. Fajlovi se u komprimovanom obliku čuvaju na disku. Da bi se mogli koristiti, moraju se prethodno dekompresovati, tj. vratiti u prvobitan oblik. Zato, u računaru na kome će se ti fajlovi koristiti, mora da postoji program koji je u stanju da dekompresiju izvrši. Pored toga, kompresija i dekompresija podataka troše određeno vreme, pa se pri odlučivanju da li će se, i kakva će se kompresija podataka vršiti, mora i o tome voditi računa.

3. Algoritmi kompresije bez gubitaka

Algoritmi kompresije se mogu podeliti u dve kategorije: algoritmi kompresije bez gubitaka i algoritmi kompresije sa gubicima. Kod kompresije bez gubitaka dobija se fajl koji će posle dekompresije biti identičan originalu. Važno je pomenuti značenje formata fajla jer je u vezi sa algoritmima kompresije. Pod formatom fajla podrazumeva se način na koji se podaci organizuju u fajlu. Kada neke podatke snimimo u određenom formatu fajla najčešće se izvrši kompresija podataka algoritmom kompresije koji taj format podržava. Kompresija podataka se ne mora izvršiti ukoliko format podržava snimanje fajla u nekomprimovanom obliku. Jedan format često podržava više algoritama kompresije.

Razvojem algoritama kompresije, osamdestih godina prošlog veka, pojavio se program ZIP koji je vršio sabijanje podataka na disku. ZIP program je i danas jako popularan i koristi se za kompresovanje podataka tipa pisanih dokumenata, gde gubici nisu dozvoljeni. Npr. kod finansijskih dokumenata ili programa promena od samo jednog bita učinila bi te dokumente neupotrebljivim. Pored ZIP-a poznati formati kompresije bez gubitaka su BZIP2 i RAR. Većina programa za kompresovanje podataka podržava više algoritama kompresije s različitim brzinama i stepenima kompresije. ZIP i RAR datoteke obično su kompresovane Deflate algoritmom koji je izveden iz LZW (Lempel Ziv Welch) algoritma.

Danas postoji jako mnogo algoritama kompresije bez gubitaka. U ovom tekstu detaljnije će biti opisana dva koja su u široj upotrebi. Jedan je stariji i teoretski više korišćen, zove se Hofmanov kôd. U praksi se više koristi drugi – LZW (Lempel Ziv Welch) algoritam.

3.1 Kôd, kodiranje i dekodiranje

Da bi se objasnio rad algoritama kompresije, potrebno je razumeti pojam kôda, kodiranja i dekodiranja.

Pojam azbuke definišemo kao konačan, neprazan skup znakova. Znak je element azbuke, njena nedeljiva jedinica. Dopisivanjem znakova neke azbuke, jednog za drugim, dobijamo reč te azbuke. Broj znakova u reči zove se dužina reči. Neka je dat skup B od *m* objekata, čija je priroda bez značaja, tako da je

$$B = \{b_1, b_2, \ldots, b_m\},\ m \geq 1$$

Neka je data azbuka od *n* znakova, tako da je

$$A = \{a_1, a_2, \ldots, a_n\},\ n \geq 1$$

Ako svakom objektu iz skupa B pridružujemo po jednu reč azbuke A, onda se tako dobijen skup reči K zove kôd objekta B u azbuci A. Reč iz skupa K, koji čini kôd, zove se kodna reč. Proces pridruzivanja reči u azbuci A elementima skupa B zove se kodiranje. Obrnut proces, raspoznavanje elemenata skupa B, na osnovu zadate reči iz azbuke A, zove se dekodiranje. Broj znakova azbuke A, tj. broj *n* zove se osnova koda.

Ako je $n = 2$, azbuka $A = \{0,1\}$ se zove binarna, a kôd te azbuke binarni kôd. Svi podaci u današnjim elektronskim računarima se kodiraju znakovima binarne azbuke. Algoritmi kompresije vrše kodiranje ulaznih podataka znakovima binarne azbuke, pri čemu se

odbacuje višak informacija.

Ukoliko su sve kodne reči iste dužine, kôd je ravnomeran, u suprotnim je neravnomeran. Primer ravnomernog binarnog koda je ASCII kod. Primer neravnomernog koda je Morzeov kôd, u kome se slova i cifre zamenjuju rečima iz azbuke *{.,-}*. Na primer, slova *E, A, I, O, S, X, Y* zamenjuju se kodnim rečima redom - , . - , .. , - - - , ... , . - - . , - . - - . Ako se koristi neravnomeran kod, dužina kodiranog teksta (teksta koji se dobija od polaznog zamenom slova iz skupa B kodnim rečima) zavisi od izbora koda. Dakle, pojavljuje se mogućnost uštede simbola, tj. da se za istu poruku utroši manje simbola iz skupa *B*. Na izbor dobrog koda utiču učestanosti f_i (ili verovatnoće p_i) pojave simbola b_i u tekstu koji se kodira, $i = 1,2,...,m$. Ako su sve verovatnoće jednake, tj. $p_1 = p_2 = ... = p_m$, onda su i ravnomerni kodovi dobri. U suprotnom je intutivno jasno da ređe simbole treba kodirati dužim kodnim rečima.

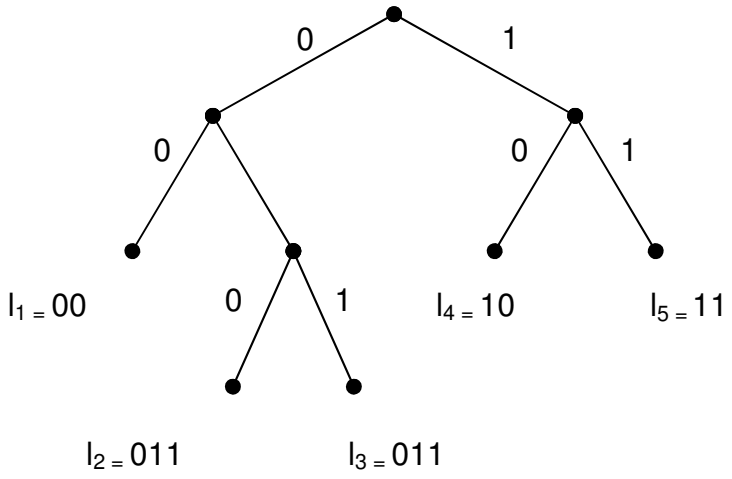
Kao što je već pomenuto, da bi se kompresovani fajlovi mogli koristiti, moraju se prethodno dekompresovati, tj. vratiti u formu koja odgovara čoveku. Da bi se to ostvarilo, potrebno je da kôd bude jednoznačan. Kôd je jednoznačan ukoliko nema kodne reči koja bi se mogla dekodirati u dva različita izvorna simbola. Jedna od klasa jednoznačnih kodova su trenutni kodovi — kodovi pri čijem se dekodiranju kodne reči prepoznaju onog trenutka kad se završe. Kôd je prefiksan, ukoliko zadovoljava uslov da ni jedna kodna reč nije prefiks bilo koje druge kodne reči. Lako se može zaključiti da je prefiksni kôd jednoznačan i trenutan kôd. Prefiksnom kodu može se pridružiti binarno korensko stablo za dekodiranje, u kome svakoj kodnoj reči odgovara jedinstven put od korena stabla do nekog od listova, pri čemu su listovi, čvorovi stabla bez sinova. U stablu svaka grana ka levom sinu ima oznaku 0, a ka desnom sinu oznaku 1. Niz oznaka grana na putu od korena do lista, koji odgovara nekoj kodnoj reči, upravo čini tu kodnu reč.

Primer prefiksnog kôda

Posmatrajmo kôd sa pet znakova i kodnim rečima

$$l_1 = 00,\ l_2 = 010,\ l_3 = 011,\ l_4 = 10,\ l_5 = 11$$

Želimo da dekodiramo poruku *0101101100*. Polazimo iz korena ulevo (prvi znak je 0), pa desno (znak 1), pa levo (znak 0). Kada se dođe do lista uočavamo da je prva kodna reč je $s_2 = 010$ ona koja odgovara tom listu. Ostatak poruke se dekodira na isti način, posle vraćanja u koren. Rezultat *je 010|11|011|00* = $l_2 l_5 l_3 l_1$. Stablo za dekodiranje je prikazano sledećom slikom:



Primer binarnog korenskog stabla za dekodiranje

3.2 Izbor dobrog kôda

Algoritmi kompresije vrše kodiranje ulaznih podataka, pri čemu to kodiranje mora biti jednoznačno i pri čemu se prirodno nameće pitanje dobrog kôda. Kada se traži dobar prefiksni kôd, cilj je da dužine kodnih reči $l_1, l_{2,\ldots,} l_n$ budu što manje. Tačan kriterijum da li neki skup brojeva $l_1, l_{2,\ldots,} l_n$ može da predstavlja dužine kodnih reči prefiksnog koda daje nejednakost Krafta. Ta teorema pokazuje da trenutni kôd sa dužinama kodnih reči $l_1, l_{2,\ldots,} l_n$ postoji akko važi nejednakost

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1$$

Neka su p_i verovatnoće znakova koji se kodiraju, a l_i dužine kodnih reči. Za prefiksni kôd koji minimizira vrednost $\sum_{i=1}^{n} p_i l_i$ kaže se da je optimalni kôd. Može se pokazati da za prefiksni optimalni kôd važe sledeća svojstva:

- Znaci c_j *i* c_k sa većim verovatnoćama pojavljivanja imaju kraće kodne reči, $p_j > p_k \rightarrow l_j \leq l_k$
- U kodnom stablu optimalnog prefiksnog kôda svaki unutrašnji čvor mora da ima oba sina. U suprotnom bi se kôd mogao pojednostaviti skraćivanjem kodnih reči u listovima podstabla sa samo jednim sinom. Iz te činjenice sledi da postoji optimalni prefiksni kôd u kome su kodne reči za dva znaka sa najmanjim verovatnoćama listovi sa istim ocem, na najvećem mogućem

rastojanju od korena (tj. te kodne reči se razlikuju samo po poslednjem bitu).

Označimo sa $H(p_1, p_2, ..., p_n)$ funkciju $H(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$. Može se pokazati da je ova funkcija donja granica za prosečnu dužinu kodne reči $\sum_{i=1}^{n} p_i l_i$, a to znači da uvek važi nejednakost $\sum_{i=1}^{n} p_i l_i \geq H(p_1, p_2, ..., p_n)$. Claude E. Shannon je pokazao da se kodiranjem dužih blokova teksta prosečna dužina kodirane poruke proizvoljno može približiti ovoj donjoj granici. Shannon je u svom radu iz 1948.godine "Matematička teorija komunikacije" formulisao teoriju kompresije podataka. Ta teorija postavlja fundamentalne granice na performanse svih algoritama za kompresiju podataka. Teorija ne pokazuje kako dizajnirati ili implementirati ove algoritme, ali pruža osnovna uputstva kako doći do optimalnih performansa.

3.3 Hofmanov algoritam

Hofmanov algoritam je nastao kao rezultat seminarskog rada diplomca Masačusetskog instituta tehnologije (MIT) Davida Huffmana. Hofmanov algoritam je algoritam za kompresiju bez gubitaka, ali se često koristi i pri algoritmu kompresije sa gubicima, kao finalni korak posle kvantizacije signala. Pri kompresiji signala, deo za kvantizaciju omogućava da se proizvede niz potpuno nezavisnih simbola. Hofmanov kôd ima više varijanti. Statički Hofmanov kôd je osnovna varijanta koja se može lakše objasniti i razumeti.

Hofmanov kôd je optimalan prefiksni kôd. Algoritam kodiranja, tj. nalaženja optimalnog kôda se zasniva na svođenju problema sa *n* znakova na problem sa *n-1* znakom. Matematičkom indukcijom dokazujemo da postoji optimalni kôd za skup od *n* znakova. Slučaj skupa od dva znaka je trivijalan. Induktivna hipoteza je da se za skup od *n-1* znakova može odrediti optimalni prefiksni kôd. Slučaj skupa sa *n* znakova se može svesti na slučaj manjeg skupa na sledeći način. Neka su c_i i c_j dva proizvoljna znaka sa najmanjim verovatnoćama javljanja u skupu koji se kodira. Prema pomenutom svojstvu optimalnog prefiksnog koda, postoji optimalan kôd u kome znacima c_i i c_j odgovaraju listovi na maksimalnom rastojanju od korena. Dva znaka c_i i c_j mogu se zameniti novim znakom, koji možemo da označimo sa $c_i c_j$, sa verovatnoćom $p_i + p_j$. Skup sada ima *n-1* znak, pri čemu zbir verovatnoća znakova nije promenjen, pa se za njega prema induktivnoj hipotezi može odrediti optimalni kôd. Optimalni kôd za polazni skup dobija se tako što se listu $c_i c_j$ u kôdu za *n-1* znak dodaju dva sina, lista koji odgovaraju znacima c_i i c_j.

Hofmanovom kodu možemo dodeliti binarno korensko stablo za dekodiranje, jer je prefiksan kod. Stablo izgrađujemo počevši od lišća. Da bismo ga izgradili, moramo unapred da znamo verovatnoće pojavljivanja svakog znaka skupa koji se kodira. U prvom koraku uočimo dva znaka najmanje verovatnoće javljanja i spojimo ta dva znaka u jedan.

Zatim, ponovimo postupak iz prethodnog koraka za novodobijeni skup znakova, koji sada ima jedan član manje. Postupak se ponavlja dok ne dobijemo skup sa dva znaka. Kodiranje sada vršimo tako što tim preostalim znakovima dodelimo kodove 0 i 1. U sledećem koracima razdvajamo spojene znakove, i svakom od njih dodajemo po 0 ili 1 na postojeće kodne reči. Postupak se završava kada razdvojimo sve spojene znakove.

Primer statičkog Hofmanovog kôda

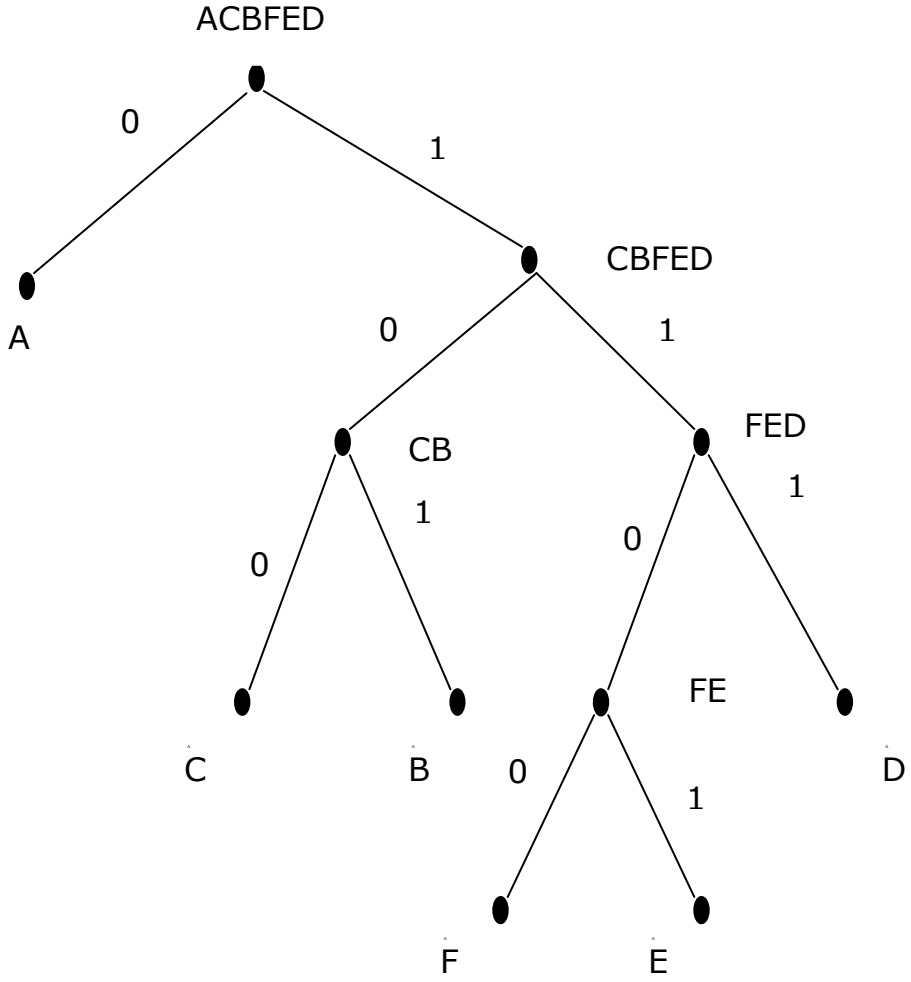
Neka je dat tekstualni fajl u kome se pojavljuju znaci A, B, C, D, E i F sa frekvencijama redom 45, 13, 12, 16, 9, 5. Znaci, skup ulaznih znakova je S = {A, B, C, D, E, F}. Opisaćemo statički Hofmanov algoritam kodiranja ovih znakova.

znak	**A**	**B**	**C**	**D**	**E**	**F**
frekvencija	**45**	**13**	**12**	**16**	**9**	**5**

Ulazna skup S ima 6 znakova za koje znamo frekvencije pojavljivanja. Primenom ranije opisanog algoritma dobijamo :

znakovi	**1**	**2**	**3**	**4**	**5**	**6**	**novi znak**
1. korak	**A** **45**	**B** **13**	**C** **12**	**D** **16**	**E** **9**	**F** **5**	**FE** **14**
2. korak	**A** **45**	**B** **13**	**C** **12**	**D** **16**	**FE** **14**		**CB** **25**
3. korak	**A** **45**	**D** **16**	**CB** **25**	**FE** **14**			**FED** **30**
4. korak	**A** **45**	**CB** **25**	**FED** **30**				**CBFED** **55**
5. korak	**A** **45**	**CBFED** **55**					**ACBFED** **100**

U slučaju Hofmanovog kodiranja ukupna dužina bita upotrebljenih za kodiranje ulazne azbuke je L = 45 * 1 + 13 * 3 + 12 * 3 + 16 * 3 + 5 * 4 + 9 * 4 = 224. U slučaju ravnomernog kôda, svaki znak se kodira sa tri bita i tada je L = 3*(45+13+12+16+5+9) = 300 bitova, što znači da se Hofmanovim kodiranjem uštedelo oko 25% prostora za skladištenje podataka. Kodno stablo dobijenog Hofmanovog koda je:



U većini slučajeva potrebno je Hofmanove tabele kodiranja pridružiti kompresovanom fajlu kao informaciju. U praksi tih tablica nema, tako da se u praksi statički Hofmanov kôd ne koristi. Obično se koristi da se dokaže da ne može bolje da se kodira, ali je u praksi neupotrebljiv. Zato se koristi dinamički Hofmanov kôd. On poštuje iste principe kao statički, ali sa tim da verovatnoće izgrađuje u toku rada. Počinje se ni sa čim. U početku, za prvih pedesetak karaktera ne radi dobro, ali se posle toga vidi koji se karakteri češće javljaju. Pokušaj kompresije prekratkih programa neće dati dobre rezultate. Hofmanov algoritam je najbolji mogući algoritam za kodiranje kada su verovatnoće poznate i nezavisne.

3.4 LZW algoritam

LZW algoritam dobio je naziv prema početnim slovima prezimena ljudi koji su ga razvili. Publikacija Abraham Lempel-a i Jacob Ziv-a je 1977. godine postavila osnove sledećeg velikog koraka u kompresiji podataka. Dok je Hofmanov koder postizao dobre rezultate, bio je ograničen na kodiranje jednog po jednog karaktera u jednoj vremenskoj jedinici. Lempel i Ziv su predložili šemu koja bi kodirala nizove podataka. Ova tehnika je uzela maha i prednost, pogotovu kod sekvenci znakova koji se često pojavljuju, kao što je na

primer tačka propraćena jednim slobodnim mestom u tekstualnim fajlovima. IEEE Computer je objavio tezu Terry Welch-a 1984. godine koja je prdstavila sam LZW (Lempel Ziv Welch) algoritam. Ova teza je donosila poboljšanja u odnosu na original, predlažući da bi kodna tabela mogla biti kreirana na isti način za kompresor i dekompresor, a da pritom nije bilo potrebno uključiti ovu informaciju u kompresovane podatke.

Može se očekivati da LZW koder kompresuje tekst, izvršni kod, i slične tipove podataka. LZW algoritam postiže izuzetno dobre rezultate kompresije pri kodiranju izuzetno redundantnih fajlova, kao što su brojevi u tabeli, kompjuterski izvorni kod.Takođe, LZW algoritam kompresije se koristi u GIF grafičkim formatima, i ponuđena je kao opcija u TIFF i PostScript grafičkim formatima. Iako same implementacije LZW-a ponekad mogu biti komplikovane, sam algoritam je veoma jednostavan. On vrši pretraživanje kako bi zamenio nizove karaktera jednostavnim kodiranim rečima koje se čuvaju u tabeli kodova. Najčešća implementacija LZW-a koristi 12-bitne kodirane reči kako bi predstavile 8-bitne unešene karaktere. Kodna tabela ima 4096 lokacija, s obzirom da je to broj unikatnih adresibilnih lokacija jednog 12-bitnog indeksa. Prvih 256 lokacija su inicijalizovane za same karaktere. Kako su nove kombinacije karaktera parsirane u toku unosa, ovi nizovi se dodaju tabeli nizova, i čuvaju se u lokacijama od 256 do 4095. Za razliku od Hofmanovog algoritma, tabela kodova se ne čuva u komprimovanom fajlu.

Primer tabele kodova

Kodiranje se vrši tako što identifikujemo sekvencu bajtova u originalnom fajlu, koja postoji u kodnoj tabeli. Neka je ulazni fajl niz brojeva: 123 145 201 4 119 89 243 245 59 11 206 145 201 4 243 245, i neka je data sledeca kodna tabela :

kod	*prevod*
0000	0
0001	1
· :	: ·
0254	254
0255	255
0256	145 201 4
0257	243 245
· :	: ·
4095	xxx xxx xxx

Prvih 256 lokacija u tabeli odgovara vrednostima jednog bajta, od 0 do 255, dok ostatak od 256 do 4095 lokacija odgovara nizu bajtova. Prikazana kodna tabela nije stvarno generisana LZW algoritmom, ona samo predstavlja pojednostavljeni primer.

Kodiranje originalnih podataka se može prikazati sledećom tabelom:

originalni podaci	123	145	201	4	119	89	243	245	59	11	206	145	201	4	243	245
kodirani podaci	123	256	119	89	257	59	11	206	256	257						

Iz tabele zaključujemo da npr. kôdu 256 odgovara niz od tri bajta 145 201 i 4. Stepen kompresije je veći što je duži niz bajtova koji odgovara jednom kodu. Iz ovog jednostavnog primera bismo mogli da zaključimo da postoje dva osnovna problema koje bi LZW algoritam trebalo da prevaziđe:

- Kako da odredi koje će se sekvence bajtova nalaziti u kodnoj tabeli
- Kako da obezbedi program za dekompresiju koji koristi istu kodnu tabelu, a koja se ne nalazi u kompresovanom fajlu.

Kada LZW algoritam počinje da kodira fajl, kodna tabela sadrži samo prvih 256 unosa, a ostatak tabele je prazan. To znači da se prvi bajt iz ulaznog fajla kodira pomocu 12 bitova. Kako kodiranje protiče, LZW algoritam identifikuje ponovljene sekvence podataka i dodaje ih u kodnu tabelu. Kompresija počinje kada se ista sekvenca bajtova drugi put pojavi u ulaznom fajlu. Ključni momenat je da sekvenca iz ulaznog fajla nije dodata u kodnu tabelu sve dok nije predstavljena u komprimovanom fajlu kao pojedinačni karakter (kôd od 0 do 255). To je važno, zato što omogućava programu za dekompresiju da rekonstruiše kodnu tabelu direktno iz kompresovanih podataka, bez prenošenja kodne tabele.

Šema LZW algoritma za kodiranje i primer LZW kodiranja

Neka su date promenljive STRING tipa String (promenljiva koja može sadržati vrednost od više bajtova) i CHAR tipa char (promenljiva koja može sadržati vrednost od jednog bajta). Iz ulaznog fajla čitaju se pojedinačni bajtovi i upisuju u kompresovan fajl pomoću 12 bita.

Kompresovaćemo sintagmu: ***kolo/kolo/naokolo/vilovitio/plahovito*** i prikazujemo tabelu koja korak po korak objašnjava algoritam kodiranja 36 bajtova ulaznog fajla koji je predstavljen prethodno navedenom sintagmom.

Šema LZW algoritma za kodiranje

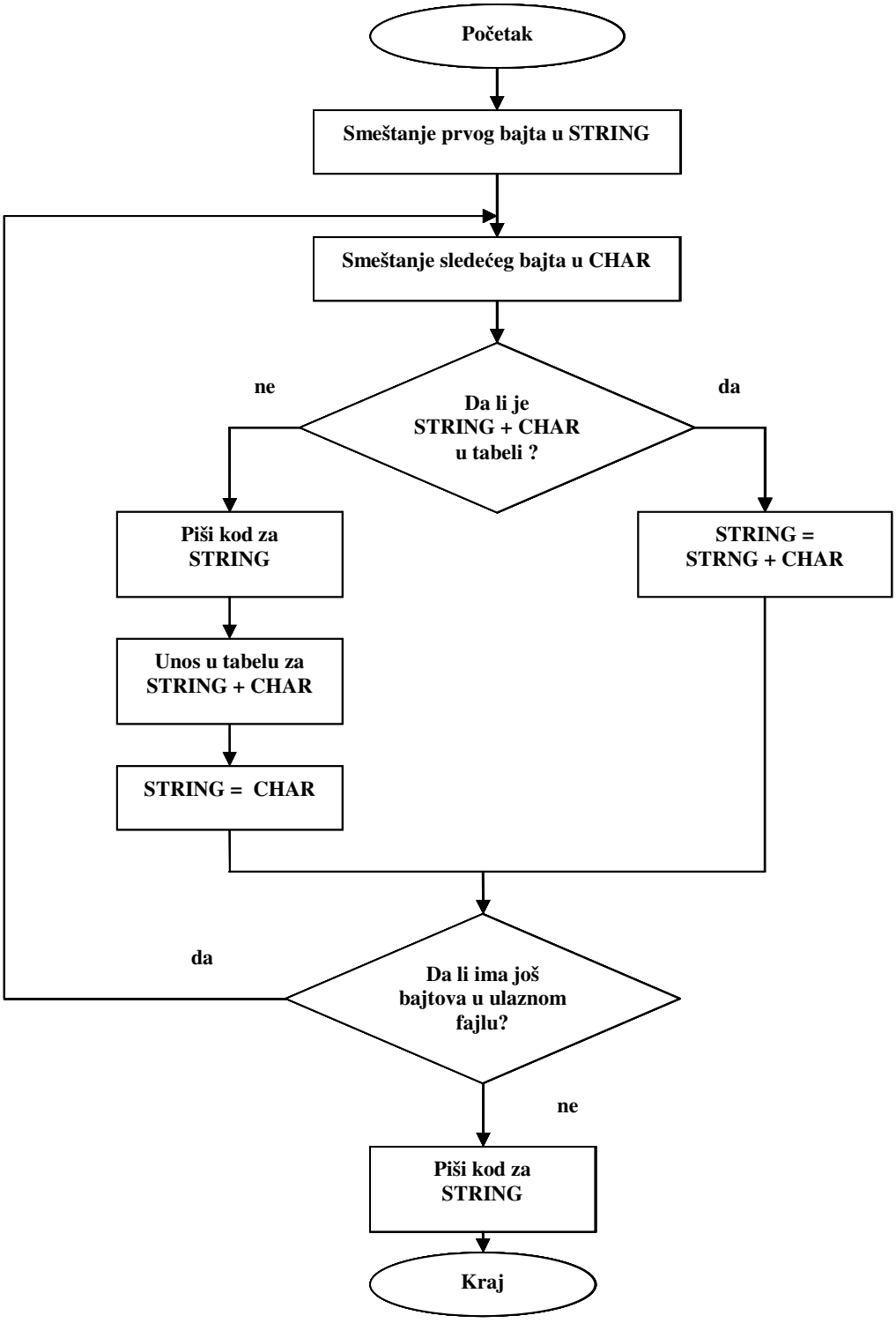


Tabela predstavlja primer LZW kodiranja sintagme: *kolo/kolo/naokolo/vilovito/plahovito*

Broj koraka	CHAR	STRING + CHAR	U tabeli?	Izlaz	Dodavanje u tabelu	Novi STRING
1	**k**	k				k
2	**o**	ko	ne	k	256 = ko	o
3	**l**	ol	ne	o	257 = ol	l
4	**o**	lo	ne	l	258 = lo	o
5	**/**	o/	ne	o	259 = o/	/
6	**k**	/k	ne	/	260 = /k	k
7	**o**	ko	da			ko
8	**l**	kol	ne	256	261 = kol	l
9	**o**	lo	da			lo
10	**/**	lo/	ne	258	262 = lo/	/
11	**n**	/n	ne	/	263 = /n	n
12	**a**	na	ne	n	264 = na	a
13	**o**	ao	ne	a	265 = ao	o
14	**k**	ok	ne	o	266 = ok	k
15	**o**	ko	da			ko
16	**l**	kol	da			kol
17	**o**	kolo	ne	261	267 = kolo	o
18	**/**	o/	da			o/
19	**v**	o/v	ne	259	268 = o/v	v
20	**i**	vi	ne	v	269 = vi	i
21	**l**	il	ne	i	270 = il	l
22	**o**	lo	da			lo
23	**v**	lov	ne	258	271 = lov	v
24	**i**	vi	da			vi
25	**t**	vit	ne	269	272 = vit	t
26	**o**	to	ne	t	273 = to	o
27	**/**	o/	da			o/
28	**p**	o/p	ne	259	274 = o/p	p
29	**l**	pl	ne	p	275 = pl	l
30	**a**	la	ne	l	276 = la	a
31	**h**	ah	ne	a	277 = ah	h
32	**o**	ho	ne	h	278 = ho	o
33	**v**	ov	ne	o	279 = ov	v
34	**i**	vi	da			vi
35	**t**	vit	da			vit
36	**o**	vito	ne	272	280 = vito	o
37	EOF			o		

Kada kažemo da LZW algoritam čita neki karakter, npr. "a", iz ulaznog fajla, mislimo da čita vrednost : 01100001 (97 izraženo pomoću 8 bitova), pri čemu je 97 ASCII kôd za

karakter "a". Kada kažemo da piše karakter "a" u kompresovani fajl, mislimo da piše 000001100001 (97 izraženo pomoću 12 bitova). Algoritam počinje smeštanjem prvog bajta ulaznog fajla u promenljivu tipa String. Ova akcija je u tabeli predstavljena linijom 1. Zatim algoritam ulazi u petlju koja se završava kada se pročita poslednji bajt iz ulaznog fajla. U svakom prolazu kroz petlju se vrše sledeće akcije :

1. Čitanje bajta iz ulaznog fajla
2. Proverava se da li vrednost koja se dobija spajanjem bajtova promenljive tipa String i bajta promenljive tipa char postoji u tabeli
3. U slučaju da dobijena vrednost koja je takođe tipa String postoji u kodnoj tabeli, promenljivoj STRING tipa String se dodeljuje dobijena vrednost.
4. U slucaju da dobijena vrednost ne pripada tabeli kodova izvršavaju se sledeća tri koraka :
    - Vrednost promenljive STRING se piše u komprimovani fajl.
    - Dobijena vrednost spajanjem bajtova promenljive STRING i CHAR se unosi u tabelu kodova.
    - Promenljiva STRING dobija vrednost promenljive CHAR.

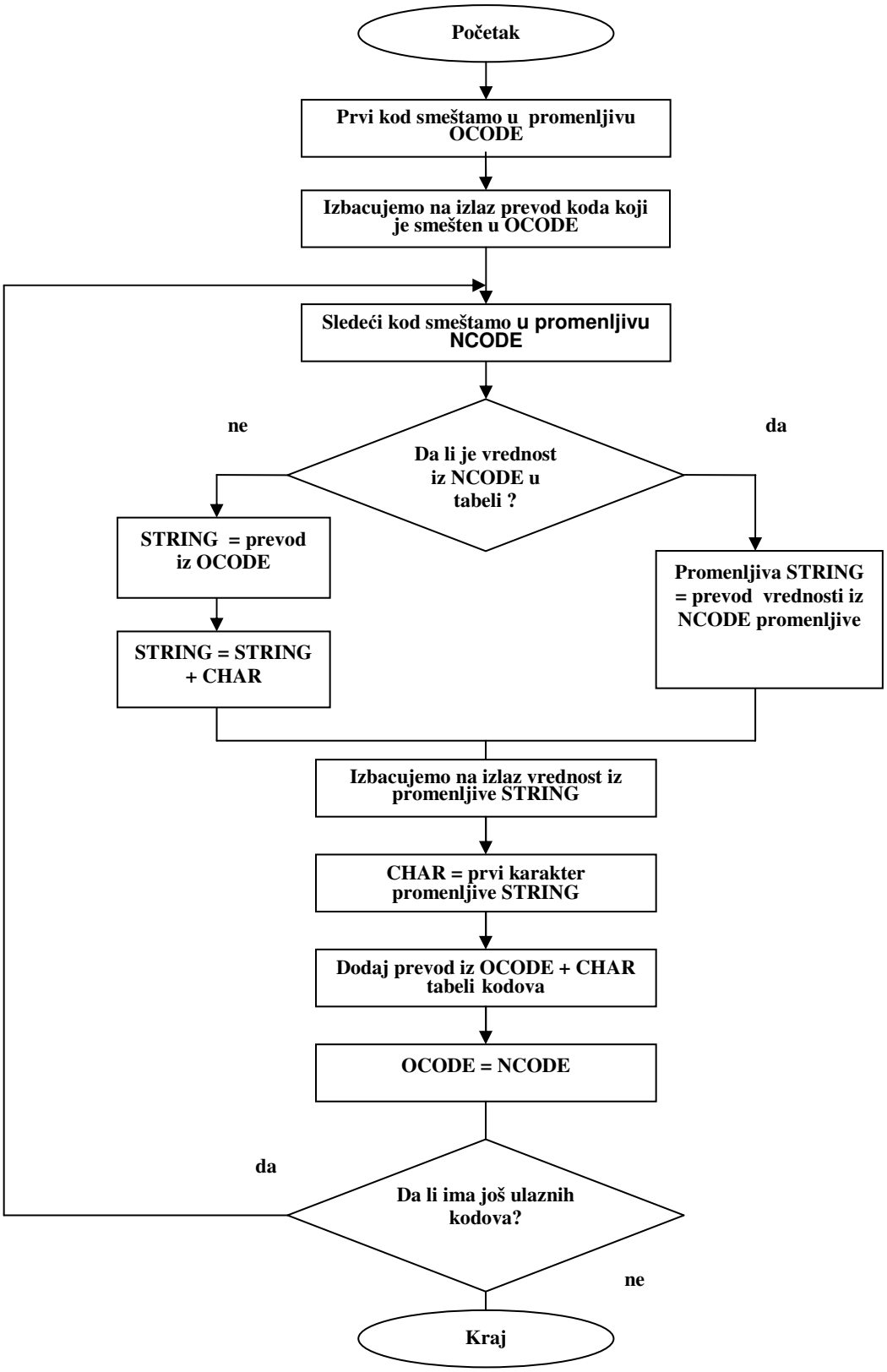
Nakon izlaza iz petlje vrednost promenljive STRING se piše u komprimovani fajl. Te akcije su predstavljene koracima 2 do 36 (pri čemu svaki korak, izuzev prvog, predstavlja jedan prolaz kroz petlju). Za prethodni primer izlazni tok je:

<k> <o> <l> <o> </> <256> <258> </> <n> <a> <o> <261> <259> <v> <i> <258> <269> <o> <259> <p> <l> <a> <h> <o> <272> <o>

LZW dekompresor kreira istu tabelu kôdova za vreme dekompresije. Počinje sa prvim 256-im unosom iz tabele inicijalizovane na pojedinačne karaktere. Tabela kôdova je ažurirana za svaki karakter pri unosu, osim prvog. Nakon što je karakter proširen u njemu odgovarajući niz kroz tabelu kôdova, poslednji karakter niza je dodat prethodnom nizu. Ovaj novi niz se dodaje tabeli u istoj lokaciji kao i u tabeli kodova kompresije. Samo je nekoliko linija koda potrebno za najelementarniju implementaciju LZW programa

Sledećom slikom prikazana je šema LZW algoritma za dekodiranje :

Šema LZW algoritma za dekodiranje



Promenljive OCODE i NCODE sadrže 12 bita koda iz kompresovanog fajla, promenljiva CHAR sadži vrednost od jedanog bajta, a promenljiva STRING sadrži vrednost niza bajtova. Kompresovani podaci su :

<107> <111> <108> <111> <47> <256> <258> <47> <110> <97> <111> <261> <259>
<118> <105> <258> <269> <111> <259> <112> <108> <97> <104> <111> <272>
<111>

Algoritam dekodiranja opisujemo na sledeći način:

Prvo unesemo prvi karakter (107), u promenljivu OCODE i izbacimo na izlaz prevod (k). Učitavamo (111) u NCODE. Pošto je NCODE u tabeli niza, postavljamo STRING = (o), (o) je onda izlaz. Promenljivu CHAR postavljamo na (o) i (ko) je prvi unos u tabelu niza. OCODE se postavlja na (111) i skače na početak petlje. Proces se nastavlja sve dok se ne isprocesiraju svi kompresovani podaci. Proces dekompresije kreira izlaz i kreira tabelu kodova koja je ista kao i pri procesu kompresije.

LZW algoritam kompresuje sekvence podataka koje se ponavljaju. S obzirom na to da su kodirane reči 12-bitne, bilo koji pojedinačno kodirani karakter proširiće veličinu podataka umesto što će je umanjiti. Ovo se uvek može videti u ranim stadijumima kompresovanja LZW algoritmom. Sintagma koju smo kodirali u prethodnom primeru je prekratka da bi se postigla kompresija. Nakon što je napravljena prihvatljiva tabela kodova, kompresija se značajno unapređuje. Postavlja se pitanje šta se dogodi u toku kompresije ukoliko smo iskoristili svih 4096 lokacija u našoj tabeli nizova. Postoji nekoliko rešenja. Jedno od rešenja je da jednostavno zaboravimo na dodavanje više ulaza i da koristimo tabelu kao takvu, a drugo bi bilo da ispraznimo sve ulaze od 256-4095 i započnemo izgradnju drveta ponovo. Neke pametnije šeme prazne ove ulaze i ponovo grade tabelu nizova od poslednjih *N* ulaznih karaktera. Broj *N* bi mogao na primer da bude 1024.

Jedna od prednosti LZW-a nad Hofmanovim algoritmom je u tome što je on u mogućnosti da kompresuje ulazni tok u jednom prolazu i pri tom ne zahteva nikakve prethodne informacije o unešenom toku podataka. Tabela nizova se pravi u toku kompresije i dekompresije. Još jedna od od prednosti je njegova jednostavnost koja dozvoljava brzo izvršavanje.

Kao što je već spomenuto, GIF slike koriste jednu od varijanti LZW-a. Kompresija je bolja od kompresije algoritma koji smo opisali, zato što koristi varijabilne kodirane reči. Sa obzirom da je tabela inicijalizovana za prvih 256 pojedinačnih karaktera, samo jedan dodatni je potreban da se kreira nova tabela nizova. Kodirane reči su 9-bitne sve dok ulaz 511 nije kreiran u tabeli nizova. U ovom momentu, dužina kodiranih reči se produžava na 10 bitova. Dužina može da se uveća do 12 bitova. Ovo uvećava kompresiju, ali dodaje kompleksnost GIF koderima i dekoderima.

3.5 Kompresija zvuka algoritmima bez gubitaka

Kompresija zvuka bez gubitaka, kao što joj samo ime kaže, kompresuje zvučni (audio) signal na takav način da se prilikom njegove dekompresije dobija signal koji je identičan polaznom signalu. Iako ima svojih prednosti, ova vrsta kompresije nije postigla veću popularnost u audio digitalnoj kompresiji, ponajviše zbog malog stepena kompresije. Pored toga, ovu kompresiju koriste audio inženjeri koji žele potpun kvalitet svojih audio fajlova. Neke od tehnika audio kompresije bez gubitaka su : FLAC (Free Lossless Audio Codec), TTA, Shorten. One se uglavnom razlikuju po brzini audio kompresije i dekompresije, dok kvalitet kompresovanog podatka nema nikakvu ulogu, jer kao što smo već istakli, polazni audio signal je potpuno identičan kompresovanom

3.6 Kompresija slika algoritmima bez gubitaka i formati grafičkog fajla

Kompresije slike bez gubitaka omogućuje da se prilikom njene dekompresije dobije slika identična slici pre kompresije. Kvalitet slike je u ovom slučaju u potpunosti sačuvan. Ovi algoritmi kompresije se obično koriste kod slika koje sadrže linije oštrih ivica, kao što su tehnički crteži, mape, tekst, fajlovi... U slučaju velikih gubitaka informacija o slici, oštre linije bi mogle postati nejasne. Zatim, kod slika čiji je sadržaj od velikog značaja, kao što su npr. slike napravljene u medicinske svrhe.

Algoritmi koji se često koriste kod kompresija slika bez gubitaka su *run-length* kodiranje (RLE) i Hofmanovo kodiranje. RLE je metod kompresije bez gubitaka, koji konvertuje uzastopne identične karaktere u kôd koji se sastoji iz karaktera i broja koji označava dužinu niza (run). Što je niz duži, kompresija je veća. Metod RLE zato najbolje rezultate daje u kompresiji crno-bele grafike i jednostavnih crteža. Prednost kompresije slika algoritmima bez gubitaka je što su kompresija i dekompresija jednostavne i brze, ali se njima ne postiže puni nivo kompresije, koji je inače moguć. Na primer, *run-length* kodiranje koristi samo sličnost piksela po horizontali (horizontalna koherencija), a ne i po vertikali.

Slika se u računaru predstavlja matricom kvadratića zvanih pikseli. Svaki piksel ima svoju boju. Boja piksela je predstavljena u računaru određenim brojem bitova. Broj bitova za opis boje jednak je za sve piksele na slici, i naziva se dubina piksela. Za ovakav prikaz slika koriste se termini rasterska ili bitmapirana grafika.

Pod formatom grafičkog fajla podrazumevamo način na koji se informacija o slici organizuje u fajlu. Korisnik se opredeljuje za format u zavisnosti od namene i sadržaja fajla, operativnog sistema, softvera sa kojim raspolažemo,... Danas postoji veliki broj formata za bitmapiranu grafiku. Namene ovih formata su raznovrsne. Neki su pogodni za obradu slika, a neki za njihovo arhiviranje i prikaz na Webu. Neki su prisutni na raznim platformama, dok su neki usko specijalizovani za posebne namene, i postoje samo na određenim operativnim sistemima. Najčešće, grafički format u svojoj specifikaciji ima naveden algoritam ili nekoliko algoritama kompresije koje podržava, tako da se može reći da format fajla i algoritam kompresije obično čine neraskidivu celinu.

Grafički formati **PCX, TIFF i BMP** su široko zastupljeni u obradi slika, uključujući skeniranje, prenos među platformama i njihovo korišćenje u stonom izdavaštvu. Ova tri

formata sadrže podatke koji su ili nekomprimovani, ili se komprimuju bez gubitaka, što ih čini dobrim pri editovanju, ali ih diskvalifikuje za korišćenje na Webu.

PCX je jedan od najstarijih grafičkih formata. Pojavio se ranih osamdesetih godina prošlog veka. I danas je jedan od najviše korišćenih formata na PC računarima. Prepoznaju ga praktično svi ikada napisani grafički programi. PCX fajlovi mogu čuvati podatke o slikama sa dubinom piksela 1, 4, 8 i 24 bita. Podaci su uvek komprimovani. Algoritam kompresije je RLE (Run Length Encoding). Ekstenzija ovog formata je PCX ili PCC.

TIFF je jedan od najšire podržanih grafičkih formata za čuvanje bitmapiranih slika na personalnim računarima. Smatra se pouzdanijim formatom od PCX-a, i ima mogućnost korišćenja boljih metoda kompresije od njega, pa su fajlovi nešto manji. Neke od metoda kompresije koje ovaj format podržava su PackBits, RLE, LZW, ZIP, JPEG, a može biti i bez kompresije. Ekstenzija ovog formata je tif ili TIFF. Činjenica da TIFF format podržava praktično sve dubine piksela i veliki broj algoritama kompresije, sa druge strane, predstavlja i manu ovog formata: različiti programi koji rade sa TIFF fajlovima su od ovih njegovih širokih mogućnosti prihvatili samo neke. Tako, TIFF fajl napravljen u jednom grafičkom programu često ne može biti prepoznat u drugom. BMP je standardni format za bitmapiranu grafiku korišćen u Windowsu. Podržava dubine piksela 1, 4, 8 i 24 bita. Podržava RLE algoritam kompresije podataka za slike sa 4 ili 8 bita po pikselu, a ekstenzija ovog formata je bmp.

Značajno je pomenuti i formate **GIF i PNG** koji su namenjeni korišćenju na Webu, jer, zahvaljujući optimizvanim algoritmima kompresije koji se u njima koriste, troše manje prostora za podatke o slikama, pa se lakše šalju preko mreže. Ovi formati imaju mogućnost progresivnog prikaza, tj. stvaranja iluzije bržeg prikazivanja slike.

GIF je star format, njegova ekstenzija je gif, i on je i danas popularan za prikaz jednostavnih slika na Webu. GIF slike su dubine piksela 8 bita i uvek su komprimovane. Metod kompresije je LZW. Ovo jeste algoritam kompresije bez gubitaka, ali odlično komprimuje jednostavne slike sa velikim oblastima obojenim istom bojom. GIF format je dobar izbor za crteže, crno-bele slike i za sitan tekst. Zbog male dubine piksela, a i zbog prirode LZW algoritma (koji ne komprimuje dobro slike sa neprekidnim tonovima) nije dobar za prikaz fotografija.

PNG je novi format za bitmapiranu grafiku, ima ekstenziju png i nastao je u pokušaju da se prevaziđu zakonski problemi vezani za korišćenje GIF-a (odnosno, LZW algoritma). Format PNG je potpuno patentno i licencno neopterećen. Svako može besplatno kreirati softver koji radi sa PNG slikama. Ovo je grafički standard predviđen da se koristi na Webu. Mada PNG omogućuje prikaz slika sa različitom dubinom piksela (do 48 bita po pikselu). PNG je, osim za prikaz slika na Web-u, dobro rešenje i za obradu slika, jer se njegovi podaci komprimuju bez gubitaka. PNG kompresija je među najboljima koje postoje bez gubitaka informacije. PNG koristi Deflate metod kompresije, metod koji se koristi i kod *pkzip*-a. Deflate je poboljšana verzija LZW algoritma kompresije. Radi slično LZW algoritmu, tj. prati ponavljanje horizontalnih uzoraka u svakoj liniji. Poboljšanje u odnosu na kompresiju prisutnu kod GIF-a je u istovremenoj kontroli vertikalnih uzoraka. Na taj način se postiže nešto veći stepen kompresije.

4 . Multimedija

Pod multimedijom podrazumevamo zvuk, slike i film (pokretne slike). To su vidovi predstavljanja informacija koji se koriste u izradi multimedijalnih sadržaja, npr. web strana, CD ili DVD prezentacija i sl. Ona je poslednjih godina najveća i najšire primenjivana aplikacija u računarstvu.

4.1 Digitalizacija medija

Pojave u realnom svetu (slike, zvuk, pokret) su kontinualne (analogne). Računar je diskretna mašina, i zato informacije ne može da čuva u kontinualnom obliku. Da bi se omogućilo kombinovanje, obrada, prikazivanje i arhiviranje multimedija, potrebno je pre svega izvršiti digitalizaciju medija. Proces digitalizacije se zasniva na matematičkim principima koji se moraju poštovati da bi se pri tom postupku prelaska sa analogne na digitalnu formu učinila što manja greška.

Digitalizacija je prestavljanje analognog signala nizom brojeva, koji predstavljaju izmerene vrednosti tog signala u sukcesivnim, najčešće ekvidistantnim trenucima. Proces digitalizacije analognog signala se sastoji iz dve faze. Prva faza se sastoji u diskretizaciji signala u vremenu ili prostoru, tj. procesu odabiranja vrednosti signala u određenim vremenskim ili prostornim intervalima. Druga faza se naziva kvantizacija i u njoj se vrednosti parametara zaokružuju na vrednosti određene diskretnim i ograničenim vrednostima nivoa kvantizacije. Da bi se digitalizovao zvuk, uređaj koji vrši uzorkovanje meri amplitudu zvučnog talasa više puta u sekundi. Da bi se digitalizovala slika na ekranu, vrši se uzorkovanje boja slike na malim rastojanjima. Da bi se digitalizovao pokret, slike se uzorkuju i po vremenskoj komponenti, pa se na ekranu prikazuje više slika u sekundi.

Važan faktor digitalizacije je veličina intervala odabiranja, odnosno frekvencija kojom se vrši odabiranje. Ako je frekvencija odabiranja mala u odnosu na dinamiku signala, bilo kakva rekonstrukcija signala ne može pomoći, jer se vrednosti signala između odabiraka ne mogu dobro proceniti. Postavlja se pitanje sa kojom najmanjom frekvencijom se mora uzorkovati analogni signal, da bi se posle digitalizacije mogao rekonstruisati sa dovoljnom preciznošću. Odgovor zavisi od dinamike analognog signala. Teorema o odabiranju precizno definiše potrebnu minimalnu frekvenciju odabiranja signala, da bi se on mogao rekonstruisati. Ta teorema kaže da ako je najviša frekvencija u spektru signala f_x , signal se može vrlo verno rekonstruisati ako se odabira sa frekvencijom većom od $2f_x$. Ta frekvencija se zove Najkvistova frekvencija. Ako se signal odabira sa nižom frekvencijom od Najkvistove, onda dolazi do tzv. preklapanja u spektru rekonstruisanog signala. Ove se manifestuje različito u zavisnosti kakve je prirode analogni signal. Npr., ako je u pitanju zvuk čuju se distorzije.

Greška koja se unosi zaokruživanjem uzorka zavisi od broja nivoa kvantizacije. Ta greška se naziva greška kvantizacije. Što je broj nivoa kvantizacije veći, to je greška kvantizacije manja i bolja je aproksimacija originalnog analognog signala. Sa druge strane, veći broj nivoa kvantizacije utiče i na povećavanje količine potrebne memorije da bi se signal prikazao i arhivirao u digitalnom obliku. Npr., kod zvuka gruba kvantizacija dovodi do pojave tzv. kvantizacionog šuma, koji se posle rekonstrukcije signala čuje u pozadini

originalnog zvučnog signala. Da bi se npr. dobio kvalitet zvuka propisan za zapis na CD potrebno je koristiti 16 bitova po odabirku, što iznosi 65536 nivoa kvantizacije. Signal bi mogao da se registruje i jednobajtno, ali se zbog dinamike, odnosno odnosa najvišeg i najtišeg zvuka ide na preciznije registrovanje.

Ono što povezuje sve medije je to što se svi oni digitalzuju i prikazuju kao struktuirana kolekcija bitova koji se mogu obrađivati pomoću softverskih programa, arhivirati u računarskim memorijama, prenositi preko računarskih mreža i prikazivati na monitoru. Dakle, zajednička digitalna reprezentacija medija omogućava njihovo kombinovanje i integraciju u jedinstvenu celinu koja se može opšte nazvati multimedijalni sadržaj.

4.2 Proizvodnja i dostavljanje multimedija

Proizvodnja multimedijalnih sadržaja ne podrazumeva samo pripremu pojedinih komponenata, već i njihovu integraciju u celoviti proizvod, jer upravo srž multimedija čini kombinovanje više medijskih komponenti u jednu celinu. Za potrebe proizvodnje multimedijalnih sadržaja razvijeno je mnogo softverskih alata za rad sa grafikom ili zvukom, kao i alati za integraciju komponenti koji sintetišu komponente i sinhronišu ih u prostoru i vremenu.

Nakon proizvodnje multimedijalnog sadržaja, neophodno je omogućiti njegov prenos do korisnika. Postoje dva načina dostavljanja multimedijalnih sadržaja do korisnika.

Prvi način dostave se primenjuje preko globalnih i lokalnih računarskih mreža. Problem kod ovog načina prenosa je propusni opseg mreže i stanje saobraćaja na mreži, koje može biti ograničavajuće za prikazivanje dinamičkih multimedijalnih sadržaja. Ograničenje može predstavljati i opterećenje servera na kome je smešten odgovarajući multimedijalni sadržaj, tako da prikaz sadržaja može biti usporen u odnosu na predviđenu brzinu.

U drugom načinu dostave multimedijalnih sadržaja koriste se CD-ROM-ovi i DVD-jevi. Memorijski kapacitet CD-ROM-oba je 650 MB, a za DVD-ove i do 17 GB. Druga važna karakteristika ovih uređaja je brzina transfera podataka. Sada je ona sasvim dovoljna za prikazivanje filmova u realnom vremenu.

4.3 Razvoj multimedija

Zvuk je postao sastavni deo računara još od pojave mikroračunara, krajem 70-tih godina, ali u vrlo primitivnoj formi. Računari su tada imali oscilator koji je na zvučnik slao signale određene frekvencije. Njegova prvobitna namena je bila da zameni zvono, tj. neka vrsta signalizacije. Od tog trenutka zvuk počinje malo po malo, linearno da se razvija. Kompleksnije melodije su nastajale tako što se otkrilo da dužina i visina tona mogu da se programiraju. Prvi računar koji je u sebi imao ugrađen muzički čip bio je IBM-ov PC Junior. Zatim se 80-tih godina pojavljuju muzičke karte koje ubrzo postaju obavezan sastavni deo svakog računara, tj. obavezno se uključuju u minimalan spisak računarskih komponenti. U to vreme muzičke karte su bile dosta dobrog kvaliteta, ali njihova iskorišćenost nije bila velika bez obzira što ih je svaki korisnik imao. Korišćenje digitalizovanog zvuka na računarima je na taj način polako napredovalo, sve dok jednog trenutka nije doživelo eksponencijalnu ekspanziju. Interesantno je da pokretač ove revolucije zvuka nije bila potreba za novim hardverom jer su muzičke karte bile dobrog kvaliteta, a CD i digitalizovan zvuk su već više od decenije postojali.

Razvitak slike i pokretne slike na računarima tekao je slično, ali značajno sporije od zvuka, jer su zahtevi za resursima bili mnogo jači. Otprilike sredinom osmdesetih godina su se pojavile neke pokretne karikature, ali pravog filma nije bilo. Međutim, i u ovom slučaju u jednom momentu dolazi do eksponencijalnog razvitka.

Osnovni problem sa digitalnim slikama ili zvukom jeste njihova veličina, tako da je upotreba digitalnih podataka ograničena velikom cenom smeštanja i prenosa, iako su uređaji za snimanje i prikazivanje danas mnogo jeftiniji. Smeštanje i prenos zvuka, slika i pokretnih slika na disk zahteva mnogo veće stepene kompresije od onih koji se mogu dobiti kod kompresije bez gubitaka. Rešenje ovog problema i razlog za izuzetan razvoj multimedija su algoritmi kompresije podataka sa gubicima. Kompresija sa gubicima dozvoljava gubitke u fajlu, tako da se po dekompresiji dobija fajl koji nije jednak originalu. Pokazuje se da sa jako malim dozvoljavanjem gubitaka mogu da se postignu ogromne uštede memorijskog prostora, dok sa druge strane ljudsko oko ili uho ove gubitke neće ni primetiti. S obzirom na tu činjenicu, može se reći da algoritmi kompresije sa gubicima predstavljaju pokretač revolucije digitalnog zvuka, slike i filma.

Multimedijalna revolucija, promenila je ne samo način na koje koristimo PC-je, već i samu našu upotrebu informacija. Tamo gde je nekada informacija bila definisana u vidu kolona brojeva ili stranica teksta, danas razmenjujemo informacije sa personalnim računarima koristeći naš glas, naše uši i oči ne samo za čitanje, već i za gledanje slika i filmova. Multimedijalni sadržaj je postao integralni deo Interneta. Teško je naći neku Web stranicu koja nema bar jednu animiranu sličicu, dok se muzika i video sve više koriste sa porastom brzine prenosa podataka na Internetu.

Pored CD-ROM uređaja, multimedijalni PC mora imati integrisanu zvučnu karticu, zvučnike, kao i hardver i softver koji je potreban za prikazivanje video snimaka i animacija. Postoji industrijski standard za tip hardvera koji zvanično predstavlja multimedijalni PC. Standard obezbeđuje kolikom brzinom CD-ROM mora prenositi podatke do CPU-a, koliko detaljno zvučni podsistem mora snimati i reprodukovati zvuk, kao i potrebnu procesorsku snagu radi obrade zvuka i video snimaka. Ali danas, taj standard se treba smatrati samo krajnjim minimumom za multimediju. Postoje brži CD-ROM uređaji, brže video kartice koje proizvode sve veće video slike i trodimenziona okruženja visoke rezolucije, bolji zvučni podsistemi snimaju i reprodukuju bogatiji, realističniji zvuk. Danas se u multimediji odigrava revolucija koja se može pokazati značajnom, kao što je bila i sama multimedija. Ona se naziva virtuelna stvarnost. To je pokušaj da se izazovu nadražaji pomoću čula zvuka, vida i dodira, koji bi bili toliko precizni da izgledaju stvarni.

5. Algoritmi kompresija sa gubicima

Algoritmi kompresija sa gubicima žrtvuju neke podatke u fajlu, tako da se po dekompresiji dobija fajl koji nije jednak originalu. Već je pomenuto da smeštanje na disk i prenos kroz komunikacione kanale zvuka, slika i pokretnih slika zahteva mnogo veće stepene kompresije od onih koji se mogu dobiti kod kompresije bez gubitaka, i da se upravo za te potrebe koriste kompresije sa gubicima. Ta ideja je naizgled loša, ali se pokazuje da nije tako, i dolazi se do interesantnih zaključaka.

Ako se radi o muzici ili slici, onda promena od jednog malog dela procenta neće da bude ni primetna, a ako i bude primetna neće da bude značajna za naša čula. Muzika i slika, a samim tim i film, jer on predstavlja niz statičnih slika, su informacije koje ne moraju da budu 100% verno reprodukovane. Sa druge strane, to su informacije koje teku u vremenu, tako da je jako teško primetiti šta se dogodilo. Trebalo je samo uočiti da kod ovih informacija postoje velike rezerve i da gubitke možemo dozvoliti. Još jedna srećna okolnost je da se pokazuje da se mogu postići veliki stepeni kompresije ako dozvolimo jako malo gubitaka. Kao što smo već istakli, značaj ovih algoritama je ogroman, jer su one pokrenule eksponencionalni razvitak digitalnog zvuka, slike i filma.

Predstavnik revolucije digitalnog zvuka je MP3 format, koji je već sada poprilično zastareo, nije ni najsavršeniji, ali je on tu revoluciju napravio. Pomoću MP3 formata na jedan CD staje oko petnaest albuma i oni mogu da se razmenjuju kroz komunikacione kanale. Kod kompresije slika zahtevi za sabijanjem podataka su još izraženiji nego kod zvuka. Ovde je izmišljen format JPEG za kompresiju digitalnih slika. Kod kompresije pokretnih slika napravljen je format MPEG, koji predstavlja neku varijantu JPEG-a. Podvarijante MPEG-a su MPEG2 i MPEG4.

5.1 Kompresija zvuka algoritmima sa gubicima

Kompresija zvuka sa gubitkom kompresuje zvučni (audio) signal tako da se prilikom njegove dekompresije dobija signal koji je nije identičan polaznom signalu. Zvuk koji se smatra "manje važnim" kodira se sa smanjenom preciznošću ili se upošte ne kodira. Da bi se odredilo koje su informacije u zvučnom signalu "manje važne", većina algoritma kompresije sa gubitkom koriste transformacije kao što je modifikovana diskretna kosinusna transformacija (MDCT) da kovertuje vremenski domen semplovanog zvuka u domen transformacije, koji je obično domen frekvencije. Komponentama frekvencija mogu se alocirati bitovi na osnovu njihove zvučnosti. Zvučnost frekvencijske komponente se definiše tako što se prvo izračunava prag maskiranja ispod koje se pretpostavlja da je zvuk izvan granica ljudske percepcije (psiho-akustični model). Takođe neki algoritmi kompresije sa gubitkom koriste LPC (Liner Perceptive Coding) da konvertuju vremenski domen semplovanog zvuka.

Pošto kod kompresije sa gubicima dolazi do opadanja kvaliteta zvuka, ova kompresija se smatra neodgovarajućom kod profesionalnih audioinženjerskih aplikacija kao što je editovanje zvuka. Medjutim ova kompresija je veoma pogodna za prenos i skladištenje audio podataka.

Najpoznatiji algoritam audio kompresije sa gubicima i metog za skladištenje i prenos

audio sadržaja je MPEG (Layer III) ili MP3 algoritam. Moving Picture Experts Group (MPEG) radna grupa je osnovana 1988. godine i definisala je standarde za video i audio kompresiju. MPEG-1 standard objavljen je 1993. godine od strane internacionalne organizacije za standarde (ISO). On uključuje specifikacije za 1-2 Mbps video kompresiju i tri sloja za audio kompresiju.

Termin MP3 je obično korišćen kao referenca za MPEG-1 Layer-3 specifikaciju za audio kompresiju. MP3 standard definiše procese dekodiranja, format niza bita i algoritam kodiranja audio sadržaja. Samo jezgro algoritma i teoriju su prvobitno razvili na Fraunhofer Institutu, koji ima nekoliko patenata vezanih za ovaj metod kodiranja.

MPEG audio istraživači su izvršili obiman subjektivan slušalački test kroz razvoj ovog standarda. Test je pokazao da čak pri kompresiji 1: 6 i pri optimalnim uslovima za slušanje, slušalački eksperti nisu bili u stanju da razlikuju kodirani i originalni audio klip.

Empirijski rezultati su takođe pokazali da ljudsko uho ima ograničenu frekvencijsku selektivnost koja varira u oštrini od nešto manje od 100Hz za najniže čujne frekvencije do nešto više od 4kHz za najviše. Prema tome čujni opseg može biti podeljen na kritične opsege koji predstavljaju moć razlučivanja ljudskog uha kao funkcija zavisna od frekvencije. Zbog toga što ljudsko uho ima ograničenu moć da razlikuje frekvencije, ta kritična tačka za maskiranje šuma bilo koje frekvencije jedino zavisi od aktivnosti signala u kritičnom opsegu te frekvencije. Za audio kompresiju, ova osobina, odnosno to pravilo se može koristiti u transformisanju audio signala u domen frekvencija, zatim podeliti rezultajući opseg u podopsege koji približno odgovaraju kritičnim opsezima, i na kraju kvantifikovati svaki podopseg na osnovu čujnosti kvantifikovanog šuma koji se nalazi unutar tog opsega. Za optimalnu kompresiju, svaki opseg bi trebalo da bude kvantifikovan sa ne većim brojem nivoa od potrebnog, da bi kvantifikovan šum bio nečujan.

MPEG standard za audio kompresiju prihvata audio sadržaj snimljen na 32 kHz, 44.1 kHz i 48 kHz. Standardni digitalni Compact Disk (CD) sadrži dva kanala nekompresovanih 16-bitnih PCM (Pulse Code Modulation) podataka izmerenih na 44.1 kHz. U ovom formatu, svaki uzorak predstavlja određeni napon u jednom trenutku vremena. Rezultujući niz bitova zahteva brzinu podataka od 1.411 Mbps da bi se obavio prenos ovog audio sadržaja. Dobro kodiran MP3 sadržaj može da dostigne približan CD kvalitet audio reprodukcije pri brzini podataka od 128 Kbps.

MP3 je izuzetno popularan digitalni audio format, koji bez obzira na gubitke, zadržava prilično dobar kvalitet. Mada je MP3 trenutno najpopularniji audio format, postoje bolji formati od njega, što znači da ima audio formata koji postižu veći stepen kompresije i imaju sličan kvalitet (npr. VQF, MP4). Međutim, MP3 je toliko rasprostranjen, a novi formati ne nude toliko velika poboljšanja da bi korisnici prešli na njihovo masovno korišćenje.

5.2 Kompresija slika algoritmima sa gubicima

5.2.1 Kvalitet i veličina grafičkog fajla

Pri čuvanju slike u računaru stalno se susrećemo sa problemom njenog optimalnog zapisa. Pod optimalnim zapisom smatramo što verodostojniji prikaz slike, sa što manje zauzetog

prostora na disku. Dva osnovna elementa koji utiču na kvalitet i veličinu grafičkog fajla su broj piksela slike i dubina piksela.

Odluku o broju piksela slike donosi korisnik u zavisnosti od namene slike – odnosno uređaja na kome će se prikazivati. Informacija o boji svakog piksela slike čuva se u nizu bitova fiksne dužine. Slika se na ekranu (odnosno štampaču) prikazuje u svojoj normalnoj veličini tako što se piksel slike predstavlja pikselom ekrana (odnosno tačkom štampe). Rezolucija štampača je nekoliko puta veća od rezolucije ekrana. Zato je pravilo, da se za dimenzije slike (broj vrsta i kolona mreže - matrice piksela) opredeljujemo u zavisnosti od namene slike. Ako sliku hoćemo da štampamo, opredelićemo se za finiji kvalitet nego ako hoćemo samo da je prikazujemo na ekranu.

Broj bitova upotrebljenih za jedan piksel naziva se dubina piksela (dubina boje, bit rezolucija). Što je dubina piksela veća, na slici je moguće prikazati više različitih boja. Odluku o dubini piksela, a time i o bogatstvu boja grafike, donosimo u zavisnosti od toga kakvi se podaci na slici nalaze. Pri tome se mora voditi računa da što je veća dubina piksela, to je više memorijskog prostora potrebno za čuvanje slike.

Ako slika sadrži samo crno-bele elemente, za opis piksela na slici dovoljne su dve boje - crna i bela. Ove dve boje mogu se definisati korišćenjem samo jednog bita po pikselu. U RGB modelu boja dubina piksela je 24 bita, a to se realizuje tako što se sa po 8 bita predstavljaju komponente crvene, zelene i plave boje, koje se kombinuju da bi se prikazala boja piksela. Na ovaj način, na ekranu se može predstaviti približno 16,7 miliona različitih boja, a to je više nego dovoljno za ljudsko oko.

Osnovni problem sa digitalnim slikama jeste njihova veličina. Čak pri malim rezolucijama npr. 640×480 piksela, sa 24 bita po pikselu koja su neophodna za fotorealistican kvalitet, rezultat je 900 KB za jednu sliku.

Kao što je već pomenuto, upotreba digitalnih slika često je ograničena velikom cenom smeštanja na disk i prenosa kroz komunikacione kanale. Svrha kompresije slika je njihovo efikasno skladištenje i prenos kroz komunikacione kanale.

Digitalne slike smeštene u matrici piksela su zapravo struktura pogodna za kompresiju, pošto su susedni pikseli često identični ili se razlikuju za male vrednosti. Pomenuto je da većina standardnih formata za rastersku grafiku (npr. GIF, PNG) podržava metode kompresije bez gubitaka, ali da se njima ne postiže puni stepen kompresije, koji je inače moguć. Algoritmi kompresije sa gubicima daleko bolje komprimuju podatke.

5.2.2 JPEG standard

Značaj digitalnih slika kao kompjuterskih podataka prepoznat je od strane internacionalnih organizacija za standarde ISO I CCITT. Saradnjom ove dve organizacije stvoren je JPEG (Joint Photographics Experts Group) standard za kompresiju digitalnih slika. JPEG se koristi samo za kompresiju statične slike (jedna slika), ali postoji sličan standard MPEG koji se bavi kompresijom pokretnih slika (niza statičnih slika). JPEG standard uključuje četiri osnovne metode za kompresiju.

Prve tri se temelje na diskretnoj kosinusnoj transformaciji (DCT) i spadaju u kompresiju sa gubicima - “lossy”. To znači da dekompresovana slika nije identična originalu – slici koju smo kompresovali. Pomoću ovih metoda može se dostići veoma visok nivo

kompresije, a algoritam je osmišljen tako da iskoristi poznata ograničenja ljudskog oka da slabije uočava razlike u nijansama boje, nego u intenzitetu svetlosti. JPEG standard specificira da se odnos između nivoa kompresije i verodostojnosti prikaza kompresovane slike može podešavati pomoću odgovarajućih parametara algoritma.

Četvrta metoda za kompresiju koju JPEG standard uključuje se temelji na prediktivnom kodiranju i spada u kompresiju bez gubitaka - "lossless". To znači da je dekompresovana slika identična originalu. Kod ove metode vrednost sledećeg piksela se određuje kao aritmetička sredina susednih piksela, pri čemu se pamti samo razlika između trenutne i proračunate vrednosti piksela.

Ne postoji jedinstven standard za JPEG algoritam, već skup pravila i preporuka sa mnogo podesivih parametara. JPEG je razvijen na osnovu sledećih zahteva:

1. Mora da zadovoljava široki raspon kvaliteta slike i mora da obezbedi mogućnost određivanja željenog odnosa kompresija/kvalitet.

2. Mora da bude primenljiv na praktično sve tipove digitalnih slika tj. ne sme da bude ograničen rezolucijom slike, sistemom prezentacije boja, kompleksnošću scene, itd.

3. Ne sme imati preveliku kompleksnost izračunavanja, kako bi mogao da se uz prihvatljive perfomanse implementira na što veći broj procesora. Nakon iscrpnog testiranja, metode bazirane na diskretnoj kosinusnoj transformaciji (DCT) pokazale su se kao najbolje.

4. Mora da obezbedi sledeće načine rada :

    - sekvencijalno kodiranje – kodiranje prelaskom po slici red po red;

    - progresivno kodiranje – tako kodirana slika se kod dekodiranja gradi višestrukim grubo-na-fino prelazima tako da se kvalitetet slike poboljšava;

    - kodiranje bez gubitka informacija (lossless), bez obzira na stepen kompresije;

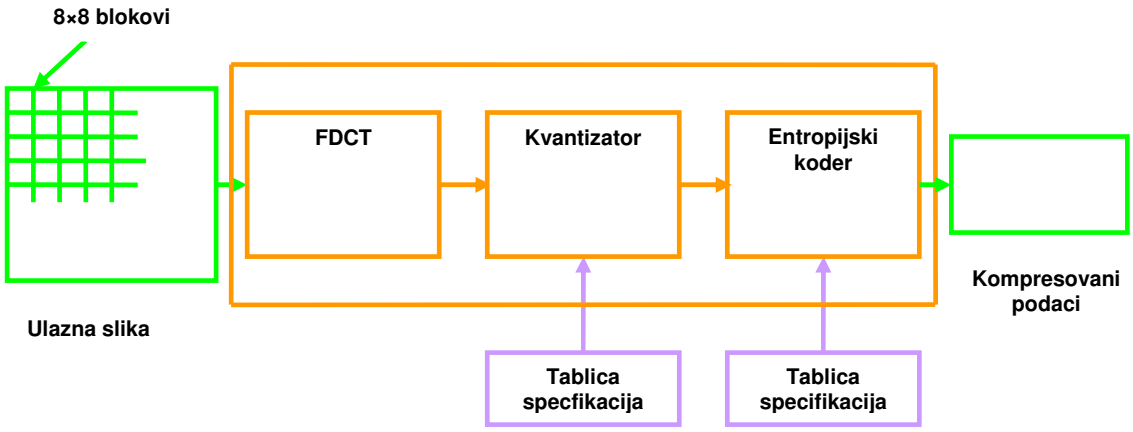
    - hijerarhijsko kodiranje – slika je kodirana na više rezolucija tako da je moguće dekodirati sliku u nižoj rezoluciji, bez da je prethodno dekodiramo u punoj rezoluciji.

5.2.3 Sekvencijalni algoritam kodiranja

Kao što je već pomenuto, JPEG grupa se odlučila za metod koji je zasnovan na direktnoj diskretnoj kosinusnoj transformaciji (FDCT). Ona je veoma slična diskretnoj Furijeovoj transformaciji (DFT), ali je jednostavnija za implementaciju na računaru.

Na sledećoj slici je prikazan algoritam sekvencijalnog kodiranja. Slika predstavlja algoritam kompresije za crno-belu sliku, odnosno slike s jednom komponentom (u slučaju crno-bele slike ta komponenta je svetlost). Kompresiju slike u boji možemo posmatrati

kao višestruku kompresiju crno-belih slika, tako što orginalnu sliku podelimo na više slika koje sadrže pripadne komponente.



JPEG koder

Osnovni koraci JPEG sekvencijalnog algoritma kodiranja su:

1. **Direktna diskretna kosinusna transformacija (FDCT)**
2. **Kvantizacija**
3. **Kodiranje entropije - koju razmatramo u dve etape, formiranje međusekvence i Hofmanovo kodiranje**

Kao što je već pomenuto, kod većine digitalnih slika, pri posmatranju velikih blokova piksela, može se ustanoviti da se intezitet piksela obično sporo menja. Postepene promene digitalnih signala slike znače da su komponente niskih frekvencija dominantne, dok je većina visoko frekventnih komponenata jednaka nuli. Ključna ideja JPEG algoritma je bazirana upravo na toj činjenici. Slika će biti konvertovana u frekventni domen, gde će samo mali broj komponenata biti značajan, pri čemu će biti dostignut visok nivo kompresije. U daljem tekstu detaljno opisujemo osnovne korake sekvencijalnog kodiranja.

1. Direktna diskretna kosinusna transformacija (FDCT)

JPEG algoritam tretira slike kao niz čiji su članovi blokovi piksela dimenzija 8 x 8. Intenzitet svakog piksela može biti posmatran kao prostorna funkcija dve promenljive: x i y. Ako se P bita koristi za reprezentaciju piksela, onda intenzitet piksela pripada intervalu $[0, 2^P-1]$.

Na ulazu u koder uzorci ulazne slike su grupisani u blokove od 8 x 8 piksela i dovedeni na ulaz bloka za FDCT. Pored direktne diskretne kosinusne transformacije postoji i njoj inverzna funkcija, inverzna diskretna kosinusna transformacija (IDCT). Jednačine za obradu 8 x 8 blokova piksela FDCT i IDCT su:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \text{ - 8 x 8 DCT}$$

$$f(u,v) = \frac{1}{4}\left[\sum_{x=0}^{7}\sum_{y=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\left(\frac{(2y+1)v\pi}{16}\right)\right] \text{ - 8 x 8 IDCT}$$

gde je : $C(u) = C(v) = \frac{1}{\sqrt{2}}$ za $u = 0, v = 0$, a $C(u) = C(v) = 1$ inače.

Svaki 8x8 blok koji sadrži uzorke slike možemo posmatrati kao diskretni signal sa 64 elementa koji su funkcija prostornih dimenzija x i y. Takav signal dovodimo na ulaz FDCT bloka na čijem izlazu dobijemo 64 FDCT koeficijenta frekvencije. Svaki koeficijent predstavlja jednu od 64 jedinstvenih prostornih "frekvencija" i predstavljaju "spektar" ulaznog signala. Prvi od tih koeficijenata odgovara srednjem intenzitetu u tom 8x8 bloku i zove se DC koeficijent, dok se ostalih 63 koeficijenata zovu AC koeficijenti. Pošto se intenzitet uzoraka obično sporo menja po bloku, većina AC koeficijenata su bliski 0. FDCT kompresija zasniva se upravo na toj činjenici, jer FDCT od takvih vrednosti koncentriše koeficijente u niže prostorne frekvencije. Prema tome većina koeficijenata ima vrednost nula ili blizu nule, te ih nije potrebno kodirati.

Kako je FDCT inverzna transformacija, vrednosti intenziteta piksela mogu se tačno rekonstruisati od vrednosti FDCT koeficijenta pomocu IDCT. Ovo znaci da FDCT ne pravi gubitke prilikom kompresije, ukoliko zanemarimo male greške zaokruživanja.

Implementacija FDCT i IDCT nije standardizovana JPEG standardom, jer ne postoji optimalan algoritam za sve tipove procesora.

2. Kvantizacija

Nakon obavljene FDCT transformacije pristupa se kvantizaciji koeficijenata. Svaki od 64 frekvencijskih koeficijenta se uniformno kvantizuje. Pošto ljudsko oko ne može da razlikuje male varijacije ovih koeficijenata, svaki koeficijent se deli odgovarajućom vrednošću iz kvantizacione tabele i predstavlja se celobrojnom vrednošću od 0 do 255. Na ovaj način se dobija niz koeficijenata u kome figurišu mali celi brojevi i u kome je veliki broj koeficijenata jednak nuli. To je povoljno za treći korak kompresije - kodiranje entropije, zbog toga što je osnovna ideja JPEG algoritma da se slika predstavi u frekventnom domenu gde će imati malo koeficijenata različitih od nula i samim tim će se bolje kompresovati (kodiranje je efikasnije ako treba kompresovati nizove sa uzastopnim identičnim elementima).

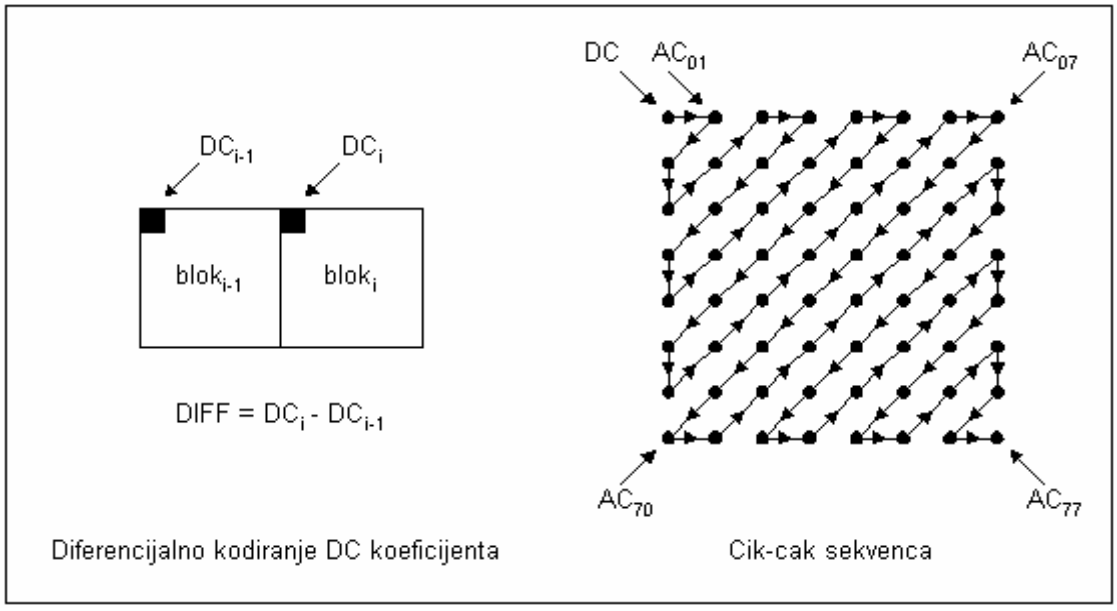
Kvantizacionu tabelu korisnik mora unapred zadati. Jedan skup kvantizacionih koraka (elemenata kvantizacione tabele) nalazi se u JPEG standardu. Tokom razvoja standarda izvršen je veliki broj testova za svaki od DCT koeficijenata. Testovi su se odvijali tako što se na jednoj slici vrednost datog koeficijenta inkrementirala u koracima po 1, a polazna i

novodobijena slika su stalno vizuelno upoređivane. Onog trenutka kada bi razlika postala uočljiva, vrednost koraka kvantizacije za dati koeficijent bi se definisala kao prethodni inkrement, dakle kao najveći inkrement za koji vizuelna razlika još uvek nije uočljiva. Prilikom testiranja su korišćene slike raznolikog sadržaja, koje su, opet, posmatrane sa različitih distanci. Prilikom implementacije JPEG algoritma se mogu koristiti ove vrednosti, ili se, ako je u pitanju neka specifična aplikacija, mogu sprovesti sopstveni testovi i koristiti njima dobijene vrednosti koraka kvantizacije.

Kvantizacija je osnovni uzrok gubitaka u JPEG algoritmu kompresije i kvantizacioni korak je glavni parametar koji odredjuje kompromis između stepena kompresije i verodostojnosti prikaza slike. Ako uzmemo veće korake kvantizacije, koeficijenti će biti predstavljeni nižim vrednostima, a takođe će biti veći broj koeficijenata jednakih nuli. Ovo daje veći stepen kompresije pri kodiranju entropije, ali će zato razlike između originalne i dekomprimovane slike biti uočljivije.

3. Kodiranje entropije

Treći i poslednji deo sekvencijalnog algoritma kompresije je kodiranje entropije i pri ovom postupku DC koeficijenti se tretiraju odvojeno od AC koeficijenata. DC koeficijent predstavlja srednju vrednost intenziteta u polju od 8 x 8 semplova i prirodno je da se DC koeficijenti susednih blokova ne razlikuju mnogo. Da bi se i ova činjenica iskoristila za što bolju kompresiju, DC koeficijenti se ne beleže po apsolutnoj vrednosti, već kao razlika u odnosu na DC koeficijent prethodnog 8 x 8 bloka. AC koeficijenti se potom preuređuju u niz po rastućim prostornim frekvencijama. Ovaj niz, prema tome, ima oblik F(0,1), F(1,0), F(2,0), F(1,1), F(0,2), i tako dalje, i često se naziva "cik-cak" sekvencom. Znači, kvantizacioni koeficijenti poređaju se u cik-cak sekvencu kako je to prikazano na sledećoj slici. Takav redosled pomaže da se olakša kodiranje entropije postavljanjem nisko-frekventnih koeficijenata (koji su vrlo verovatno različiti od nule) ispred visoko-frekventnih.



Priprema kvantizacionih koeficijenata za kodiranje entropije

U prvoj fazi "cik-cak" sekvenca koeficijenata se prevodi u jednu međusekvencu u kojoj se nulti AC koeficijenti kodiraju dužinom niza uzastopnih nultih koeficijenata. Međusekvenca za DC koeficijente se formira na sličan način.

JPEG standard nudi izbor izmedju dve metode entropijskog kodiranja ovako pripremljene međusekvence: Hofmanovog kodiranja i aritmetickog kodiranja. Iako aritmeticko kodiranje nudi malo bolju kompresiju, Hofmanovo kodiranje se češće primenjuje, uglavnom zato što je jednostavnije za implementaciju. Entropijsko kodiranje može se posmatrati kao proces iz dve etape : **transformacija niza koeficijenata u međusekvencu i Hofmanovo kodiranje.**

- **Međusekvenca**

Prvi korak kodiranja entropije je transformacija "cik-cak" niza koeficijenata u međuniz simbola.

U ovom nizu, uzastopni AC koeficijenti čija je vrednost nula kodiraju se samo brojem pojavljivanja. Zato se samo dva simbola mogu pojaviti u međunizu.

Prvi simbol opisuje broj pojavljivanja AC koeficijenata čija je vrednost nula i koji prethode AC koeficijentu koji ima vrednost različitu od nule. Ovaj simbol ima formu : **(RUNLENGHT, SIZE)**.

RUNLENGTH polje predstavlja broj prethodnih AC koeficijenata u cik-cak sekvenci jednakih nuli, dok SIZE polje predstavlja broj bitova potrebnih za reprezentaciju amplitude prvog AC koeficijenta koji ima vrednost različitu od nule.

Drugi simbol je amplituda tog ne-nula koeficijenta i ovaj simbol se obično označava sa **AMPLITUDE** i predstavlja vrednost amplitude koeficijenta.

RUNLENGTH i SIZE polja su duga četiri bita, tj mogu imati vrednosti od 0 do 15. Ako je broj uzastopnih nula-koeficijenata veći od 15, onda polje SIZE treba postaviti na 0, i ovo je zapravo dodatni simbol koji označava 16 uzastopnih koeficijenata čija je vrednost 0, tj. (15,0) se interpretira kao RUNLENGHT = 16. Odmah nakon ovog simbola treba da se pojavi sledeća dužina nula-koeficijenata. Dozvoljeno je do 3 simbola (15,0) zaredom pre nego što dođe specijalni simbol (0,0) koji predstavlja kraj bloka.

Bitovi koji predstavljaju vrednost simbola AMPLITUDE ne mogu se koristiti kao binarne vrednosti koeficijenata. Na primer, kada polje SIZE ima vrednost 1, AMPLITUDE simbol ima binarnu vrednost 0 ili 1. Medjutim, ako je AC koeficijent jednak 0, bio bi predstavljen prvim simbolom, stoga ako je SIZE = 1 vrednosti koeficijenta mogu biti -1 i 1. Dalje, kada polje SIZE ima vrednost 2, postoje četiri moguće vrednosti simbola AMPLITUDE i to su: -3, -2, 2 i 3. Struktura polja SIZE i simbola AMPLITUDE prikazana je sledećom tabelom:

SIZE	AMPLITUDE
1	-1,1
2	-3,-2,2,3
3	-7…-4,4…7
4	-15…-8,8…15
5	-31…-16,16…31
6	-63…-32,32…63
7	-127…-64,64…127
8	-255…-128,128…255
9	-511…-256,256…511
10	-1023…-512,512…1023

Struktura polja SIZE i simbola AMPLITUDE

Reprezentacija DC koeficijenata odnosno njihove razlike strukturirana je na sličan način.

Pošto u svakom bloku postoji samo jedan DC koeficijent. DC koeficijenti su predstavljeni **(SIZE, AMPLITUDE)** simbolom, gde je SIZE broj bitova za reprezentaciju vrednosti amplitude koeficijenta.

- **Hofmanovo kodiranje**

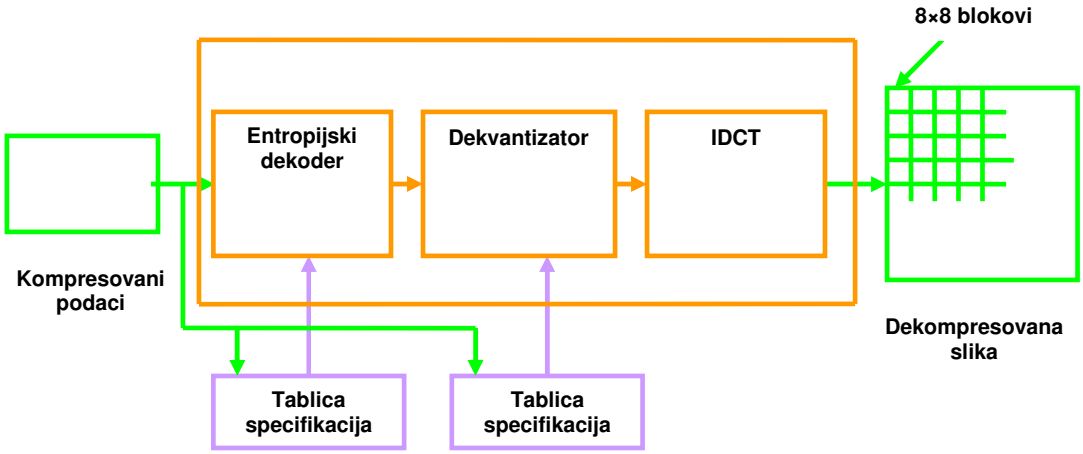
Nakon transformacije FDCT koeficijenata u međuniz kako je opisano, primenjuje se Hofmanovo kodiranje. Ovo kodiranje je zasnovano na statistici niza, gde je broj pojavljivanja svakog simbola izračunat u koraku preprocesiranja i zatim su najkraći kodovi dodeljeni simbolima koji se najčešće pojavljuju. U JPEG algoritmu, Hofmanovo kodiranje se primanjuje samo na simbole prvog tipa iz medjuniza, dok se simboli drugog tipa ne diraju.

Dakle, svaka JPEG kompresovana slika mora sadržati Hofmanovu tabelu kodova. Ova tabela može biti izračunata za svaku sliku posebno, ili jedna tabela može biti korišćena za sve slike, pri čemu se ova tabela dobija statističkom analizom velikog broja karakteristicnih slika za datu namenu.

Treba istaći razliku između kvantizacionih tabela i Hofmanovih tabela. Iako se u oba slučaja mogu koristiti predefinisane table, ovo izgleda manje podesno za Hofmanove tabele. Naime, kvantizacione tabele su konstruisane nakon detaljnog testiranja više korisnika i praktično su invarijantne, dok Hofmanove tabele zavise posebno od svake slike. Sa druge strane, kvantizacione tabele utiču na kvalitet slike, dok Hofmanove tabele utiču isključivo na stepen kompresije. Stepen kompresije može značajno opasti kada se koristi opšta Hofmanova tabela i kada određena slika ima statističke osobine koje odstupaju od proseka.

5.2.4 Dekompresija

Algoritam za dekodiranje prikazan je sledećom slikom:



JPEG dekoder

Prvi korak dekompresije je konvertovanje Hofmanovih kodova u međusekvencu. U ovom koraku koriste se Hofmanove tabele koje su sadržane u kompresovanoj slici. Zatim se simboli međusekvence proširuju do niza od 64 elementa FDCT koeficijenata, za svaki od 8x8 blokova piksela. Sledeci korak je dekvanitzacija, gde se svaki od FDCT koeficijent pomnoži odgovarajućim kvantizacionim korakom. Koraci kvantizacije (elementi matrice kvantizacije) se čitaju iz tabele kvantizacije, koja je kao i tabela Hofmanovih kodova, deo kompresovane slike. Dekvanitzacioni FDCT koeficijenti se onda preuređuju iz "cik-cak" sekvence u originalnu formu. Na kraju se primenjuje IDCT (inverzna diskretna kosinusna transformacija), koja rekonstruiše originalne vrednosti intenziteta piksela.

Zbog značajne primene JPEG kompresije, neprestano se radi na usavršavanju FDCT algoritma. Kao što je već pomenuto, JPEG grupa nije specificirala jedinstvenu DCT implementaciju imajući na umu da algoritam optimalan za jednu vrstu mikroprocesora ne mora da bude optimalan za drugu. Umesto toga, definisan je skup testova koji svaka implementacija DCT algoritma mora da ispuni. FDCT je najzahtevnijji deo JPEG algoritma, zato optimizacije JPEG algoritma treba usredsrediti najpre na ovaj korak. Prva optimizaciona tehnika koja se može preporučiti je poznat trik sa korišćenjem celih brojeva umesto brojeva u pokretnom zarezu. Druga poznata tehnika je da se sve neophodne vrednosti kosinusa izračunaju u tabeli, jer je mnogo brže čitati određenu vrednost kosinusa iz tabele, nego je računati iznova.

5.3 Kompresija video podataka

Smeštanje digitalnog filma na disk i njegov prenos kroz komunikacione kanale zahteva jako visoke stepene kompresije, značajno više nego u slučaju zvuka i statičnih slika. Pre razvoja algoritama sa gubicima postojale su samo pokretne karikature na računaru, ali pravog digitalnog filma nije bilo. Za potrebe digitalnog filma stvoren je format MPEG

koji vrši kompresiju video podataka algoritmom sa gubicima, koji je vrlo sličan JPEG algoritmu, ali još kompleksniji. Razvijeni su MPEG standardi za kompresiju podataka, koji su uglavnom vezani za kompresiju pokretnih slika. MPEG je široko usvojen standard za video kompresiju. On postiže odlične stepene kompresije. Slično kao kod JPEG standarda, MPEG-1 standard u mnogim delovima ne specificira fiksan algoritam, već skup preporuka s obzirom na odnos između stepena kompresije i verodostojnosti prikaza kompresovanih video podataka. MPEG-1 algoritam za video kompresiju je veoma kompleksan, sa mnogo kompromisa izmedju različitih prametara koje treba podesiti u zavisnosti od konkretne primene.

Digitalni sistemi, u poređenju sa postojećom analognom opremom, ispoljavaju značajne prednosti u pogledu prenosa, smeštanja, obrade i prikaza podataka. Da bismo pripremili filmove za digitalno procesiranje, svaki frejm filma je rastavljen u mrežu pravougaonih elemenata – piksela. S obzirom da je površina jednog piksela veoma mala, svaki piksel je predstavljen jednim intezitetom ili bojom. Broj mogućih inteziteta ili boja piksela je ograničen. Eksperimenti pokazuju da je 8 bitova dovoljno za kodiranje svake osnovne komponente boje, što znači da je 24 bita neophodno za kodiranje jednog piksela (3 komponente boje). Da bismo kreirali iluziju pokreta pomoću statičnih slika, neophodno je prikazati minimalno 25 frejmova u sekundi. S obzirom na to i na činjenicu da je za prikaz svakog frejma neophodna minimalna rezolucija 640x480 piksela (to je otprilike televizijska rezolucija), dobijamo da je za svaku sekundu digitalnog filma približno potrebno 25*640*480*24/8=22MB. Šta više, ako kompjuter pri isporučivanju digitalnih video podataka radi kao video server, ovaj broj (22 MB/s) mora biti pomnožen brojem korisnika koji istovremeno puštaju filmove sa servera.

Na osnovu ove analize, očigledno je da korišćenje digitalnog filma ili videa zahteva složene algoritme kompresije koji imaju mogućnost značajnog smanjenja količine podataka.

5.3.1 Razvoj MPEG standarda

Pored postojanja moćnih algoritama kompresije, standardizacija je drugo važno pitanje za široko prihvatanje digitalnog videa. Standardizacija je od posebnog značaja da bi se izbeglo pojavljivanje velikog broja nekompatibilnih formata, što se upravo dogodilo sa analognim videom. Standardizacija formata se odvija pod okriljem internacionalne organizacije za standarde (ISO). Rad na standardizaciji je proizveo nekoliko standarda. Standardi su poznati pod imenom MPEG, prema početnim slovima ISO radne grupe za standardizaciju digitalnog videa (Moving Picture Experts Group). U vreme kada je MPEG grupa u okviru ISO-a započela standardizaciju digitalnog videa, jednobrzinski CD-ROM-ovi sa protokom od 150 KB/s (1.2Mb/s) su dominirali tržištem i MPEG grupa je objavila zahtev za postizanje stepena kompresije 150:1 standardnim algoritmom. Ovaj stepen je neophodan da bi se proizvelo 25 frejmova u sekundi digitalnog filma u realnom vremenu za brzinu CD ROM-a (175.76Mbit/s:1.2 Mbit/s ≈ 150:1). Takođe, taj stepen kompresije je prilagođen propusnom opsegu Eternet lokalne mreže (10Mbit/s). Npr. moguće je imati 6-8 korisnika koji istovremeno pristupaju digitalnom video serveru, dok je ostavljen neki propusni opseg za uobičajene funkcije mreže.

Danas je MPEG-1 standard široko usvojen i primenjen. Tekst standarda je podeljen na pet delova, pri čemu svaki posebno objašnjava kodiranje video signala, kodiranje audio

signala (ovo kodiranje je dobro poznato kao MP3 audio kodiranje), kombinovanje video i audio signala sa odgovarajućim mehanizmima sinhronizacije, pravila za testiranje i verifikaciju implementacionih algoritama, kao i softverske implementacije kodera i dekodera. MPEG-2 standard predstavlja produžetak MPEG-1 standarda kroz prethodno nabrojane teme.

MPEG-4 standard je počeo da se razvija kao standard za kompresiju podataka za prenos kroz uskopropusne kanale. Međutim, značajno je odstupio od namenjenog pravca razvoja i široko prevazišao polje koje MPEG grupa pokušava da pokrije. MPEG-4 je više standardizovano okuženje za mnoge forme medija (npr. tekst, slike, animacije, 2D i 3D objekte) u kome se mogu koristiti različiti algoritmi kompresije, nego što sam pruža algoritam kompresije.

5.3.2 Kompresija individualnih frejmova

Da bi se postigli visoki stepeni kompresije sa minimalnim gubicima u kvalitetu prikaza, prostorna i vremenska povezanost video signala mora biti potpuno iskorišćena. Prostorna povezanost se predstavlja u individualnim frejmovima pomoću sličnih vrednosti boja susednih piksela. Za kodiranje blokova piksela individualnih frejmova, MPEG algoritam koristi JPEG algoritam za kompresiju statičnih slika. JPEG algoritam kompresije već je detaljno opisan u prethodnom delu ovog rada, tako da nema potrebe za njegovim ponovnim opisom.

5.3.3 MPEG-1 algoritam za kompresiju filma

Postoji nekoliko metoda kompresije koje koriste vremensku povezanost između uzastopnih frejmova. Najjednostavnije rešenje je da se ignoriše ova vrsta povezanosti i da se svaki frejm predstavi nezavisno od drugih. Neke rane implementacije algoritama za kodiranje digitalnog filma su se zasnivale upravo na ovom principu – svaki frejm je posebno kodiran pomoću JPEG algoritma kompresije. Ipak, ovaj pristup nije bio delotvoran kada su u pitanju vremenski povezani susedni frejmovi, zbog očigledne činjenice da u tipičnoj video-sekvenci nema mnogo naglih akcija.

Vremenska povezanost frejmova neke video-sekvence može biti iskorišćena ako se frejmovi kodiraju u odnosu na prethodne frejmove iz te sekvence. Ovo, takozvano "unazad-povezano" (backward-reference) kodiranje, je ekvivalent prediktivnom kodiranju, koje je dobro istraženo u teoriji digitalnih signala. Međutim, MPEG algoritam ide korak dalje, i uvodi "unapred-povezano" (forward-reference) kodiranje, tj. za kodiranje proizvoljnog frejma video-sekvence koristi se njegova povezanost sa predstojećim frejmovima te sekvence.

Osim postizanja visokih stepena kompresije, "unapred-povezano" (forward-reference) kodiranje je dobro prilagođeno da zadovolji zahteve koje digitalni video mora da ponudi korisnicima (najmanje isti nivo interaktivnosti koji nudi analogni video – pauziranje sekvenci, brzo premotavanje unapred i unazad). Sve ove vrste interakcija su manje ili više bazirane na mogućnosti slučajnog pristupa specifičnom frejmu sekvence. Iz ovih razloga, algoritmi kompresije moraju da obezbede postojanje samostalnih sekvenci, koje su kodirane bez povezanosti sa frejmovima iz drugih delova signala (ove sekvence omogućavaju brz slučajni pristup). Unutar samostalne sekvence, mora postojati najmanje jedan glavni frejm koji je kodiran bez ikakve povezanosti sa drugim frejmovima. Taj

glavni frejm je u potpunosti kodiran kao statična slika i predstavlja tačku slučajnog pristupa celoj samostalnoj sekvenci. U isto vreme, ovaj kadar je smernica za kodiranje narednih, kao i prethodnih frejmova sekvence.

Pre opisivanja algoritma, treba pomenuti da se svaki piksel predstavlja pomoću jedne ili više komponenti boja. Ako je piksel digitalnog video signala predstavljen pomoću više od jedne komponente, onda se podrazumeva da je svaki frejm predstavljen pomoću nekoliko slika, pri čemu za svaku komponentu boje postoji po jedna slika. U narednom izlaganju podrazumevaćemo da je svaki piksel video signala predstavljen pomoću jedne komponente boje.

U MPEG video sekvenci mogu se uočiti tri vrste frejmova: intra-frejmovi (intra-frames), predvidivi frejmovi (predicted frames) i interpolirani frejmovi (interpolated frames). U daljem tekstu pomenute vrste frejmova ćemo kraće obeležavati respektivno sa I-frejm, P-frejm i B-frejm. Svaki slučajan pristup sekvenci počinje sa I-frejmom koji je kodiran bez povezanosti sa drugim kadrovima. Metod kodiranja **I-frejmova** je u potpunosti ekvivalentan JPEG algoritmu za kodiranje statičnih slika.

P-frejmovi su kodirani u odnosu na odgovarajuće I-frejmove. P-frejm je podeljen na 16x16 makro-blokove piksela. Za svaki makro-blok P-frejma, MPEG algoritam traži najsličniji makro-blok u I-frejmu. Ova procedura se zove procena pokreta. Mera sličnosti makro-bloka je suma kvadrata razlika odgovarajućih inteziteta piksela :

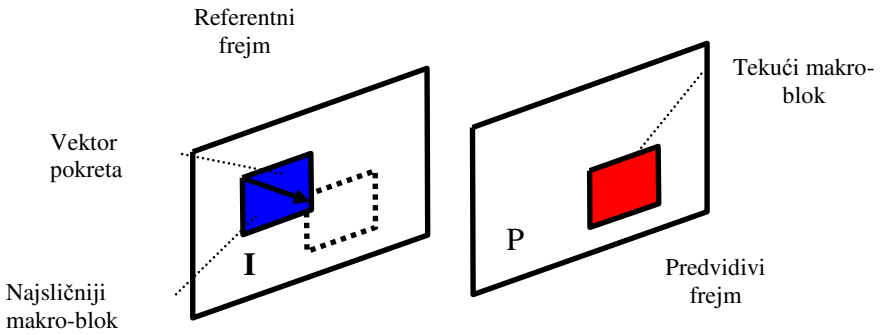
$$delta = \sum_{i=0}^{15} \sum_{j=0}^{15} \left[IntezitetP_{i,j} - IntezitetI_{i,j} \right]^2$$

gde su *i,j* indeksi vrste i kolone koji određuju piksel u makro-bloku, a $IntezitetP_{i,j}$, $IntezitetI_{i,j}$ su inteziteti piksela P-frejma i I- frejma.

Ovo izračunavanje zahteva 256 operacija množenja i isto toliko operacija sabiranja. Za najbolju kompresiju, ispitivani makro-blok treba uporediti sa svim mogućim makro-blokovima referentnog frejma. Pri rezoluciji 640x480, broj operacija potrebnih za upoređivanje svakog makro-bloka P-frejma iznosi (640-16)*(480-16)*256 = 7.5 miliona operacija.

Ovaj račun jasno pokazuje da je procena pokreta najzahtevniji deo MPEG algoritma. Postoji nekoliko heuristika za uprošćavanje ove pretrage. Jedno rešenje je da imamo korak u pretprocesiranju gde se makro-blokovi na referisanom frejmu biraju tako da samo pokriju ovaj frejmu. Detaljna pretraga je onda sprovedena samo za važne površine. Drugo rešenje je da se pretraga ograniči samo na specificne delove referisanog frejma. Pretraga obično počinje od okoline poslednjeg pronađenog makro-bloka. Zbog velike verovatnoće da oba makro-bloka pripadaju istom objektu, ova heuristika obično daje dobre rezultate.

Posle određivanja najsličnijeg makro-bloka u referentnoj slici, vektor pokreta se računa kao razlika koordinata levog gornjeg ugla predvidivog i referentnog makro-bloka, kao što je prikazano na sledećoj slici :

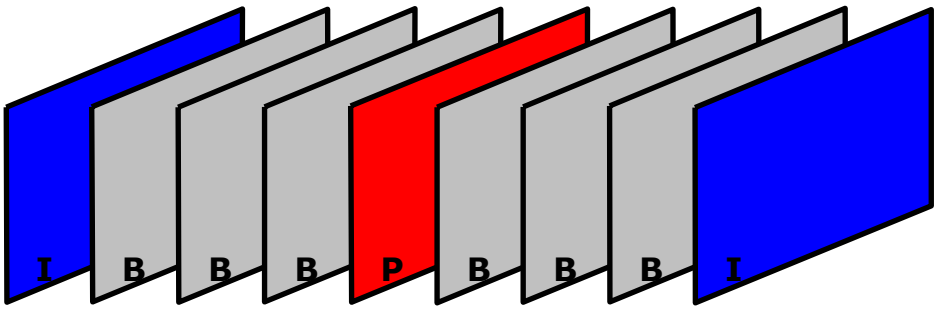


Izračunavanje vektora pokreta

Vektor pokreta je smešten u zaglavlju predvdivog makro-bloka. Da bi dalje poboljšali kompresiju, informacije vektora pokreta su kodirane kao razlike u odnosu na informacije vektora pokreta prethodnog susednog bloka.

S obzirom na piksele predvidivog makro-boka, mogu nastati dva slučaja. Ako je *delta* ispod neke predefinisane vrednosti $delta_{max}$, podrazumevamo da je isti makro-blok ponađen na referentnoj slici i ne čuvamo dodatne informacije. U drugom slučaju, pikseli makro-bloka su prediktivno kodirani, što znači da se samo razlike inteziteta u odnosu na referentni makro-blok beleže. Ove vrednosti onda podležu diskretnoj kosinusnoj transformaciji (DCT), kvantizaciji i kodiranju entropije. Ipak, treba reći da su ove razlike mnogo manje od inteziteta koji su određeni u I-frejmu, tako da su podložni većoj kompresiji.

Parametar $delta_{max}$ je drugi parametar, pored koraka kvantizacije, koji se može koristiti za fino podešavanje kompromisa između stepena kompresije i verodostojnosti prikaza. Pri većim vrednostima $delta_{max}$, više makro-blokova će biti kodirano samo vektorom pokreta, ali će odstupanja od referentnog frejma biti veća. B-frejmovi su slični P-frejmovi, ali oni koriste dvosmerno predviđanje. U MPEG sekvenci, oni su pozicionorani između I i P-frejmova, koji su referentni kadrovi za B-frejmove. Tipična MPEG sekvenca je prikazana na sledećoj slici :

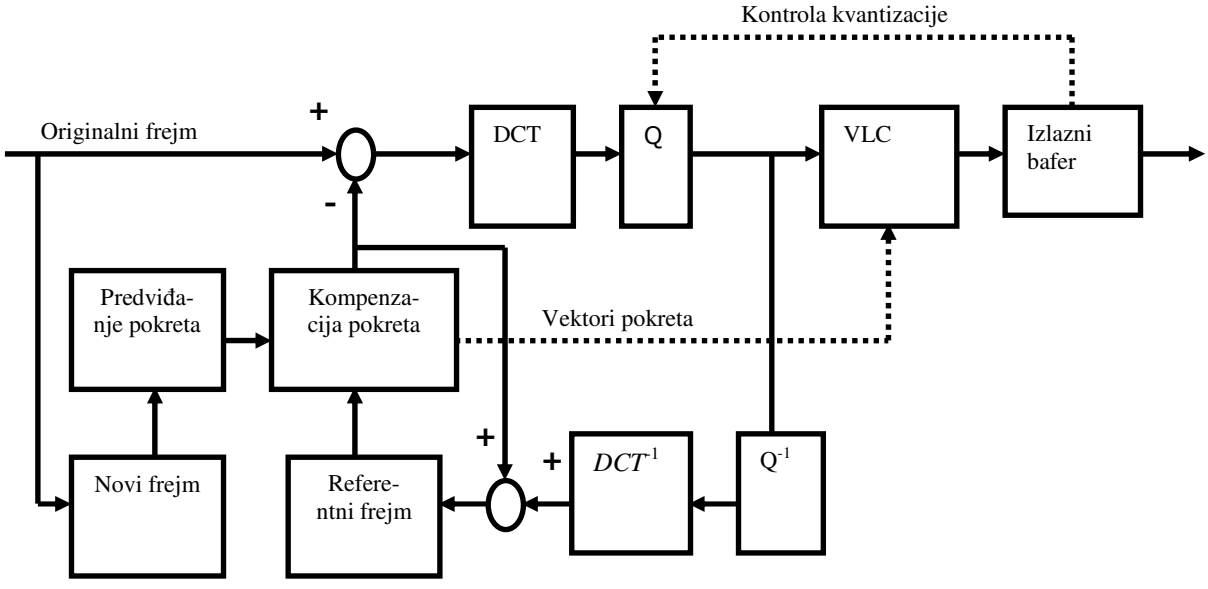


Tipična sekvenca intra-, interpoliranih i predvidivih kadrova

Svaki B-frejm ima dva referentna frejma, prethodni i sledeći u sekvenci. Za svaki makro-

blok u B-frejmu, izračunavanje sličnosti pokreta je sprovedeno za oba referentna frejma posebno. Takođe se računa razlika makro-bloka u odnosu na srednju vrednost dva referentna makro-bloka. Kao i za P-frejm, informacija o vektoru pokreta je smeštena u zaglavlju makro-bloka, sa razlikom da su u ovom slučaju smeštena dva vektora pokreta. Minimalna od tri *delta* vrednosti određuje kako će makro-blok biti kodiran. Na ovaj način, makro-blok može biti predstavljen pomoću razlika u odnosu na sledeći referentni kadar, baš kao makro-blokovi P-frejma, ili prethodni referentni frejm ili kao razlika u odnosu na srednju vrednost referentnih frejmova. Informacija o načinu kodiranja koji će se koristiti je takođe smeštena u zaglavlju makro-bloka. Kao i kod P-frejma, ako je minimalna *delta* vrednost manja od $delta_{max}$ dodatne informacije u makro-bloku se ne beleže. Kod B-frejmova ako je *delta* vrednost jako velika, onda se ceo makro-blok ne kodira pomoću razlika u odnosu na referentni makro-blok, već pravim vrednostima inteziteta, kao i makro-blokovi I-frejma.

Tačna proporcija I, P i B-frejmova nije fiksno određena MPEG standardom, izbor zavisi od konkretne aplikacije. Uobičajena proporcija je 1:3:5, što znači da sekvenca slučajnog pristupa počinje I-frejmom, zatim sledi 5 P-frejmova, a zatim 3 grupe koje se sastoje od jednog P-frejma i 5 B-frejmova. Predviđanje za svaki P-frejm je sprovedeno kao što je opisano. Početni I-frejm predstavlja referentni frejm za prvi P-frejm, a za ostale P-frejmove referentni frejm je prethodni P-frejm u sekvenci. Zbog gubitaka u kompresiji, referentni frejm ne može biti tačno rekonstruisan pri dekodiranju. Iz ovih razloga, računanje razlike prilikom kodiranja ne treba izvoditi nad originalnim referentnim frejmom, već nad kodiranim, a zatim dekodiranim referentnim frejmom. Na ovaj način, greška kvantizacije je nadoknađena. Ista procedura se primenjuje i na B-frejmove. Znači, svaki MPEG koder mora imati ugrađeni dekoder. Konfiguracija MPEG kodera je prikazana na sledecoj slici :

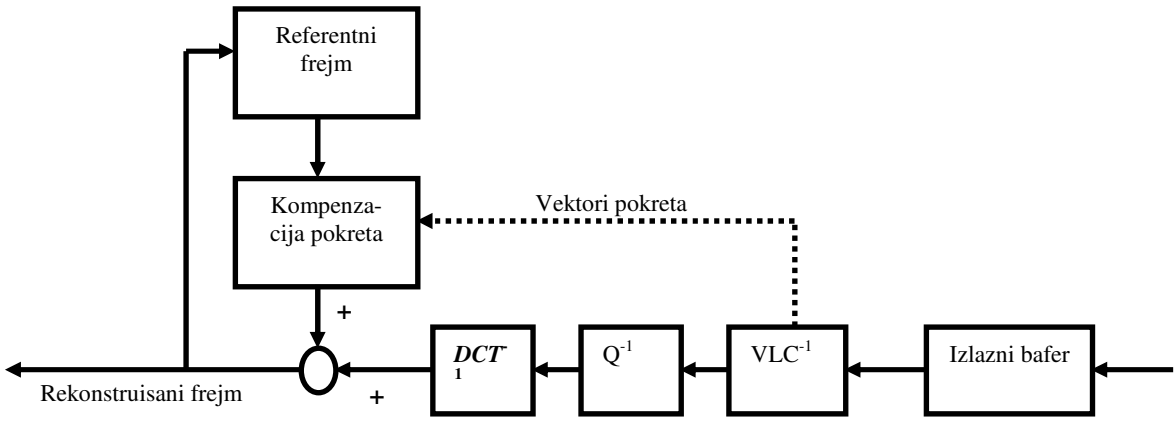


MPEG koder

Prethodna razmatranja pokazuju da nije moguće predvideti dužinu MPEG sekvence. Mnogi faktori utiču na dužinu sekvence, od kojih je kompleksnost scene verovatno dominantan. Ceo niz intrepoliranih frejmova mora biti dostupan zajedno sa dva granična referentna frejma, da bi ih mogli dekodirati. Ako bafer nije u mogućnosti da prihvati sve te kadrove, dekompresija u tom slučaju nije moguća. U najgorem slučaju, veličina bafera mora biti dovoljno velika da primi sekvencu u kojoj je svaki makro-blok interpoliranih frejmova intra-tipa, što je u većini realnih situacija bespotrebno trošenje resursa.

Drugi, ozbiljniji problem sa promenljivom dužinom sekvence je slučajan pristup, koji je nemoguće ostvariti bez fiksne dužine sekvence. Rešenje tih problema je adaptivna kvantizacija. Bafer koji je uključen u koder je iste veličine kao i bafer u dekoderu. Zasićenost ovog bafera se stalno prati prilikom kodiranja i upoređuje sa srednjim vrednostima dobijenim merenjima sprovedenim na velikom broju tipičnih test-sekvenci. Ako zasićenost bafera poraste, vrednosti koraka kvantizacije iz kvantizacione tabele se uniformno povećavaju, što rezultuje povećanim stepenom kompresije, ali takođe i nižim kvalitetom kompresovanog signala. Obrnuto, ako zasićenost bafera opadne, vrednosti koraka kvantizacije se umanjuju, dajući veći kvalitet kompresovanog signala i niži stepen kompresije. Dobijeni efekat je da se zasićenost bafera održava na konstantnom nivou i da se kreiraju sekvence fiksne dužine. Provera zasićenosti se vrši na nivou grupa makro-blokova, poznatih kao "ploča" (slice). Faktor skaliranja za korake kvantizacije se zapisuje u zaglavlju svake ploče.

Struktura MPEG dekodera je prikazana na sledećoj slici :



MPEG dekoder

6. Stepen kompresije kod algoritama sa gubicima i bez gubitaka

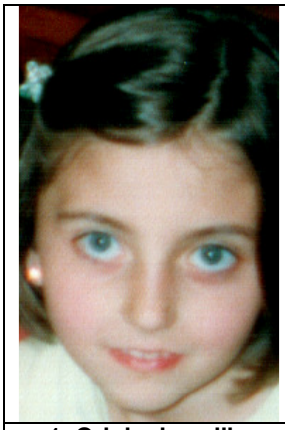
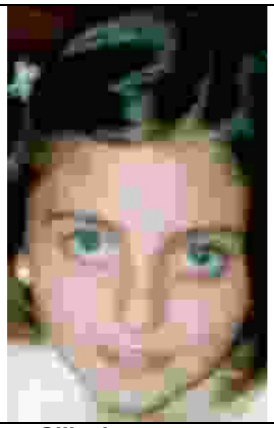
U ovom delu odredićemo i uporediti stepene kompresije koji se postižu kod algoritama sa gubicima i bez gubitaka. Eksperiment je izvršen na slikama. Kompresovane su tri slike različite vrste (fotografija, slika teksta i slika geografske karte), snimljene u RGB formatu boja i skenirane u rezoluciji 300 tačaka po inču. U sledećoj tabeli su predstavljeni uzorci slika na kojima će se upoređivati stepen kompresije koji dobijamo primenom različitih algoritama kompresije.

Tip slike	**Originalna veličina fajla**
fotografija	162 KB
slika geografske karte	319 KB
slika teksta	330 KB

Od softverskih alata za snimanje slika u različitim formatima korišćen je Gimp 2.4.6. Svaka slika je snimljena u TIFF grafičkom formatu koji podržava ZIP algoritam kompresije i JPEG grafčkom formatu sa različitim vrednostima parametra koji zovemo nivo kvaliteta. U Gimp-u, prilikom snimanja slike u JPEG formatu, postoji mogućnost podešavanja ovog parametra. Označavamo ga slovom q i on može imati vrednost od 0 do 100. U ovom eksperimentu izabrani su $q = 5$, $q = 45$ i $q = 85$ za snimanje slika sa niskim, srednjim i visokim vrednostima nivoa kvaliteta. Ovaj parametar predstavlja objedinjenje podesivih parametara koji utiču na kompromis između stepena kompresije i verodostojnosti prikaza slike. Npr., ako izaberemo $q = 5$, tj. nizak kvalitet slike, dobićemo manju veličinu fajla i slabiju verodostojnost prikaza slike.

Nakon kompresije upoređivaćemo veličine grfičkih fajlova, i odrediti postignute stepene kompresije. Pod stepenom kompresije podrazumevamo odnos veličina kompresovanog i originalnog fajla. Npr., ako je veličina originalnog fajla 100KB, a kompresovanog 50 KB, dobijeni stepen kompresije biće 0.5 (drugim rečima veličina kompresovanog fajla je smanjena za 50%).

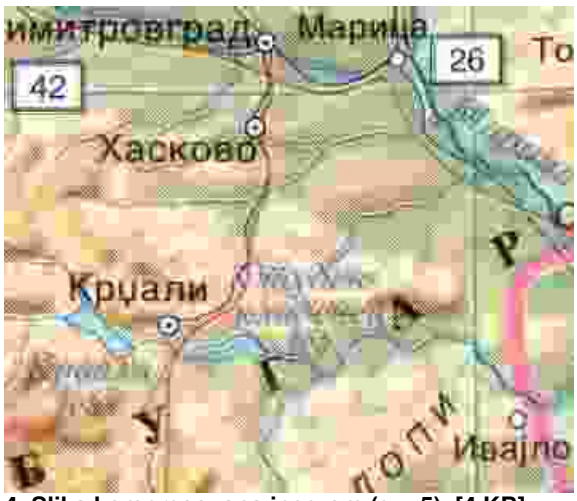
Rezultati dobijeni primenom ZIP algoritma i JPEG algoritama kompresije sa različitim vrednostima nivoa kvaliteta, za fotografiju, sliku geografske karte i teksta, respektivno su:

			
1. Originalna slika [139 KB]	**2. Slika kompresovana jpeg-om (q = 85), [11 KB]**	**3. Slika kompresovana jpeg-om (q = 45), [5 KB]**	**4. Slika kompresovana jpeg-om (q = 5), [2 KB]**

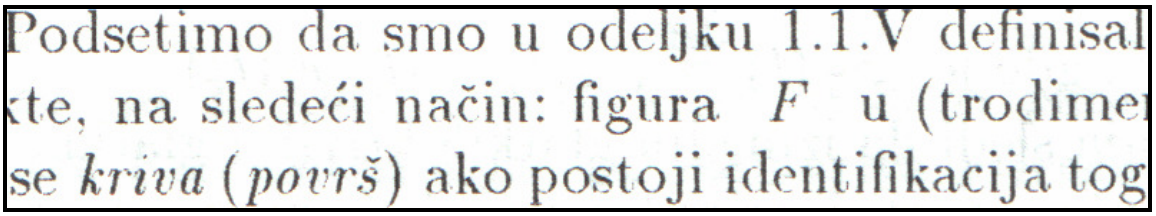
1. Originalna slika, [319 KB

2. Slika kompresovana jpeg-om (q = 85), [55 KB]

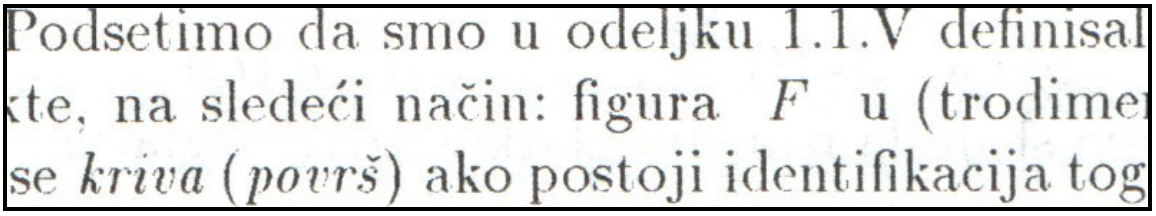
3. Slika kompresovana jpeg-om (q = 45), [25 KB]



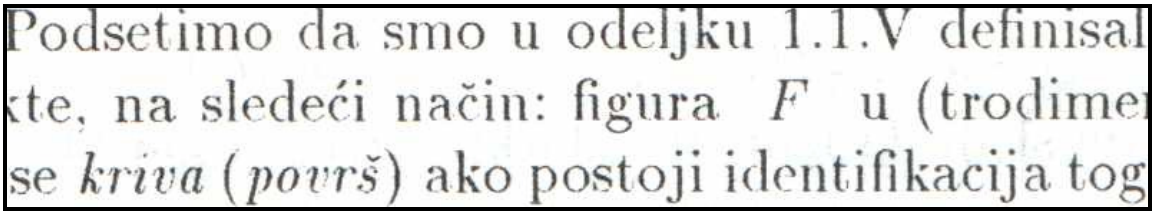
4. Slika kompresovana jpeg-om (q = 5), [4 KB]



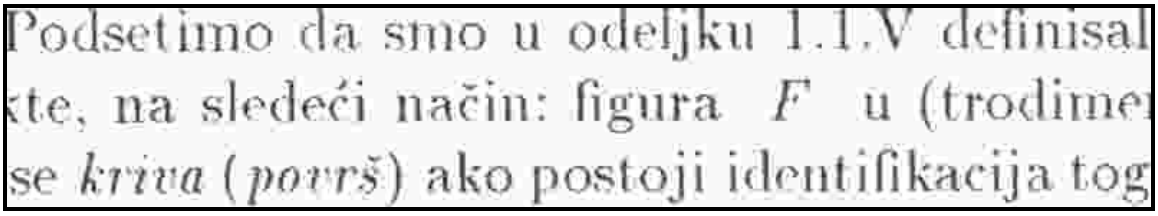
1. Originalna slika, [121 KB]



2. Slika kompresovana jpeg-om (q = 85), [29 KB]



3. Slika kompresovana jpeg-om (q = 45), [14 KB]



4. Slika kompresovana jpeg-om (q = 5), [4 KB]

Tabelarni prikaz rezultata kompresije fotografije :

Algoritam kompresije	**Nivo kvaliteta**	**Originalna veličina fajla**	**Veličina kompresovanog fajla**	**Stepen kompresije**
ZIP	/	162 KB	139 KB	0.858
JPEG	85	162 KB	11 KB	0.068
JPEG	45	162 KB	5 KB	0.031
JPEG	5	162 KB	2 KB	0.012

Tabelarni prikaz rezultata kompresije slike geografske karte :

Algoritam kompresije	**Nivo kvaliteta**	**Originalna veličina fajla**	**Veličina kompresovanog fajla**	**Stepen kompresije**
ZIP	/	319 KB	319 KB	1.000
JPEG	85	319 KB	55 KB	0.172
JPEG	45	319 KB	25 KB	0.078
JPEG	5	319 KB	4 KB	0.013

Tabelarni prikaz rezultata kompresije teksta :

Algoritam kompresije	**Nivo kvaliteta**	**Originalna veličina fajla**	**Veličina kompresovanog fajla**	**Stepen kompresije**
ZIP	/	330 KB	121 KB	0.367
JPEG	85	330 KB	29 KB	0.088
JPEG	45	330 KB	14 KB	0.042
JPEG	5	330 KB	4 KB	0.012

Primenom ZIP algoritma dobijena je najveća ušteda u slučaju zipovanja teksta, veličina fajla smanjena je za oko 63% (drugim rečima fajl je smanjen nešto manje od 3 puta). Ovaj rezultat je mnogo bolji u odnosu na zipovanje fotografije gde je veličina fajla smanjena za oko 14%, i u slučaju kompresije slike geografske karte, gde uštede nema. Razlog što u slučaju slike geografke karte nije došlo do smanjenja fajla je verovatno nedovoljna veličina uzorka, kao i struktura slike (slika sa neprekidnim tonovima sa relativno čestim naglim promenama inteziteta boje) koja je nedgovarajuća za ovaj algoritam kompresije.

Za sve tri vrste slika koje su kompresovane JPEG algoritmom dobijamo izuzetno visok stepen kompresije. Smanjenje kompresovanog fajla kreće se u opsegu 91% - 99% za sve vrednosti nivoa kvaliteta (drugim rečima fajlovi su smanjeni od 11 do 82 puta). Izuzetak je slika geografske karte, gde je za $q = 85$ fajl smanjen za oko 83% (tj. nešto manje od 6 puta). U svakom od slučajeva kompresije JPEG algoritmom dobijen je daleko veći stepen kompresije u odnosu na rezultate dobijene ZIP algoritmom. U slučaju kompresije sa gubicima, razlike u verodostojnosti prikaza (koja je subjektivan parametar) su za ljudsko oko neuočljive kada posmatramo slike koje su kompresovane sa visokom i srednjom vrednošću nivoa kvaliteta. Za nizak nivo kvaliteta ($q = 5$) uočavamo gubitke (slike postaju nejasne - boja se razgrađuje, a oštre ivice su zamrljane).

Već je pomenut PNG grafički format i istaknuto je da on vrši kompresiju bez gubitaka koja koristi Deflate algoritam (poboljšana verzija LZW algoritma) kompresije, koji se smatra jednim od najboljih za kompresiju bez gubitaka. Snimanjem uzoraka slika fotografije, geografske karte i teksta u PNG formatu dobijene su tri slike - fotografija.png, karta.png i tekst.png. Sledeća tabela prikazuje postignute stepene kompresije:

Naziv slike	Originalna veličina fajla	Veličina kompresovanog fajla	Stepen kompresije
fotografija.png	162 KB	84 KB	0.518
karta.png	319 KB	284 KB	0.890
tekst.png	330 KB	107 KB	0.324

Postignuti stepeni kompresije snimanjem slika u PNG formatu su za sve tri slike bolji, u odnosu na stepen kompresije koji je postignut snimanjem slika u TIFF formatu koji koristi ZIP algoritam. Međutim, jedino je u slučaju fotografije ova razlika značajna (veličina fajla smanjena je duplo).

Generalno, može se zaključiti da se kompresijom slika algoritmima bez gubitaka veličina fajla maksimalno može smanjiti 2 do 3 puta, dok se algoritmima sa gubicima fajl može smanjiti više od 20 puta, a da ljudsko oko to ne primeti. Znači, da algoritmi sa gubicima daleko bolje kompresuju slike u odnosu na algoritme bez gubitaka. Rezultati upoređivanja kompresije zvuka algoritmima sa gubicima i bez gubitaka bi dale slične rezultate, ali je prikaz rezultata eksperimenta jednostavnije izvršiti na slikama.

7. Zaključak

Eksperimentalni rezultati dobijeni upoređivanjem veličina fajlova dobijenih kompresijom slika JPEG algoritmom i ZIP algoritmom, kao predstavnicima svoje vrste, pokazali su da se daleko manji fajlovi mogu dobiti u slučaju kompresije sa gubicima. Može se zaključiti da se izborom visoke i srednje vrednosti nivoa kvaliteta, koji garantuju verodostojanost prikaza dekompresovane slike, dobijaju velike uštede u veličini fajla, nezavisno od tipa slike koja se kompresuje. Generalno, algoritmi kompresije sa gubicima se koriste kod digitalnog zvuka, slike i filma, jer ovi podaci dozvoljavaju gubitke, a postižu visoke stepene kompresije. Kod kompresije slika eksperimentom dobijeni stepeni kompresije se kreću uglavnom u opsegu 1:10 do preko 1:80. Za potrebe kompresije zvuka, statičnih slika i video podataka izvršena je standardizacija JPEG i MPEG formata, čime je izbegnuto pojavljivanje nekompatibilnih formata. Ovi standardi ne specificiraju fiksne algoritme. Ti algoritmi su veoma kompleksni i predstavljaju skup pravila i preporuka sa mnogo podesivih parametara. Oni utiču na kompromis između stepena kompresije i verodostojnosti prikaza dekompresovanih podataka. Kod kompresije slika bez gubitaka eksperimentom dobijeni stepeni kompresije su maksimalno 1:3. Ovaj stepen kompresije je nedovoljan za digitalan zvuk, a posebno za statične i pokretne slike gde su zahtevi za kompresijom još veći. Stoga se ovi algoritmi najčešće koriste za kompresiju tekstualnih podataka, gde gubici nisu dozvoljeni. Iako su razvijeni neki formati fajlova za smeštanje zvuka i statičnih slika, koji koriste optimizovane metode algoritama bez gubitaka i dostižu nešto veću uštedu memorijskog prostora, to je i dalje zanemarljivo u odnosu na rezultate dobijene algoritmima sa gubicima. Sa druge strane, ovi formati su značajni jer nema obrade zvuka ili slika bez gubitaka. Ako se npr. radi o digitalnim slikama, bilo kakve geometrijske transformacije bitmapiranih slika (skaliranje, translacija, refleksija, rotacija), kao i ponovljena kompresija i dekompresija, izazivaju opadanje kvaliteta grafičkog fajla. Znači, zgodno je vršiti transformacije nad originalom, a kada fajl želimo da pošeljemo preko mreže ili da napravimo veliku arhivu pesama i slika, napravimo male fajlove kao MP3 i JPEG kopije originala.

Kompresija i multimedija su u neposrednoj vezi. Multimedija praktično nije postojala do pojave kompresije. Pojava CD-a bila je dovoljna za zapis digitalnog zvuka, čija je revolucija završena, ali nije bila dovoljna za zapis digitalnih slika i filmova. Pre razvoja algoritama kompresije sa gubicima, postojanje statičnih i pokretnih slika na računaru predstavljalo je samo eksperimentalnu fazu njihovog razvoja. Sam razvoj hardvera ne bi mogao da zadovolji potrebe multimedije. Smanjenje potreba pomoću tehnika kompresije sa gubicima, omogućilo je revoluciju multimedija koja je povećala obim i promenila način korišćenja računara, kao i samih informacija. Danas razmenjujemo informacije sa personalnim računarima koristeći naš glas. Oči koristimo ne samo za čitanje, već u velikoj meri za gledanje slika i filmova.

Bitno je naglasiti da je mnogo decenija do sada, hardver bio znatno jeftiniji i da se razvijao brže nego komunikacije. Međutim, sa novim tehnologijama koje koriste nanometarski proces proizvodnje i sa taktom manjim od nanosekunde (sa fiksiranom brzinom svetlosti), približavamo se kraju eksponencijalnog razvoja kompjuterskog hardvera. Sa druge strane, optička vlakna pomeraju komunikacioni protok i kapacitet sa kilobajta u sekundi u gigabajte i terabajte u sekundi, tako da će u bliskoj budućnosti situacija biti obrnuta: biće

jeftino transferovati velike fajlove, ali će biti skupa njihova kompresija i dekompresija. U svakom slučaju, čak i sa komunikacijama koje koriste optičke kanale, mobilni deo komunikacija se mora obavljati bežično, i taj deo će biti ograničenje i nastaviće da bude bar još neko vreme izazov za razvoj i optimizaciju algoritama kompresije.

Literatura

[1] M. Tuba, predavanja iz predmeta Metodika nastave računarstva, Matematički fakultet, Beograd, 2005.

[2] M. Tuba, A. Samardžić, D. Tosić: JPEG Algorithm Parameter Adjustment, Bulletins for Applied Mathematics (BAM), CI, 2012/2002(C), pp. 303-312, Budapest, 2002.

[3] M. Tuba, A. Samardžić, D. Starčević: MPEG Video Compression Standard Parameters Selection, Bulletins for Applied Mathematics (BAM), CIV, 2076B/2003(C), Nr. 2151, pp. 575-582, Budapest, 2003.

[4] M. Živković: Algoritmi, Matematički fakultet, Beograd

[5] V. A. Obuljen: Shannon-ove teorije neodređenosti i informacije, Beograd, 2006.

[6] S. Smith: The Scientist and Engineer's Guide to Digital Signal Processing, *http://www.dspguide.com/*