KOMPRESIJA SLIKE

Postoji više razloga zašto se mora vršiti kompresija slike prije njenog memorisanja ili prenosa. Tri osnovna su:

- veliki zahtjevi za memorijskim prostorom
- relativno spori uređaji koji nisu u mogućnosti da prikazuju nekopresovanu sliku u realnom vremenu
- propusni opseg mreže koji ne dozvoljava prenos slike u realnom vremenu

Kao ilustraciju velikih memorijskih zahtjeva, posmatrajmo tipičnu multimedijalnu aplikaciju, kao što je enciklopedija:

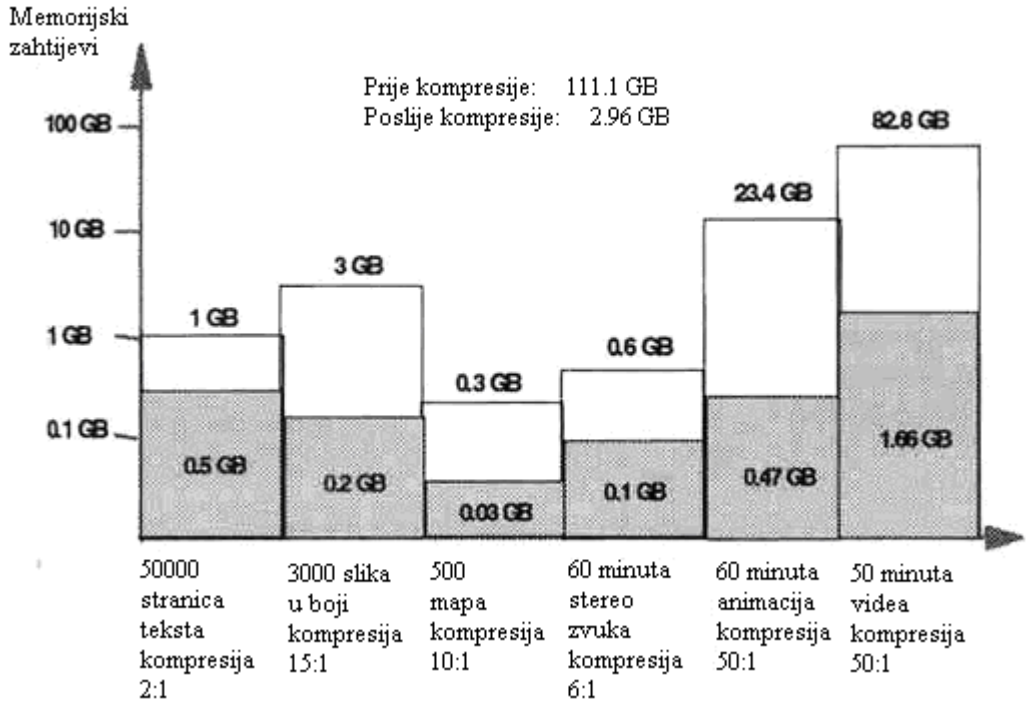
- 500,000 stranica teksta (2 KB po stranici) – ukupno 1 GB;
- 3,000 kolor slika (u prosjeku 640x480x24 bita = 1 MB po slici) – ukupno 3 GB;
- 500 mapa (u prosjeku 640x480x16 bita = 1 MB po mapi) – ukupno 0.3 GB;
- 60 min stereo zvuka (176 KB po sekundi) – ukupno 0.6 GB;
- 30 animacija, u prosjeku po 2 minute (640x320x16 bita x 16 slika/sec = 6.5 MB/sec) – 23.4 GB;
- 50 digitalizovanih filmskih zapisa (video sekvenci) u prosjeku od 1 minuta (640x480x24 bita x 30 slika/sec = 27.6 MB/sec) – 82.8 GB.

Dakle, za enciklopediju je potrebno 111.1 GB memorije. Pretpostavimo da je izvršena sljedeća kompresija:

- tekst 2:1;
- kolor slike 15:1;
- mape 10:1;
- stereo zvuk 6:1;
- animacije 50:1;
- video 50:1.

Na Slici 106 prikazani su zahtjevi za memorijskim prostorom prije i poslije kompresije. Kompresijom se zahtijevi za memorijom smanjeni sa 111.1 GB na svega 2.96 GB, čto predstavlja količinu podataka sa kojom je znatno lakše ¨rukovati¨.

Za ilustraciju sporosti uređaja za prikaz slike, razmatrajmo prikazivanje jednosatnog filma sa nekog memorijskog medija, recimo CD-ROM-a. Ako pretpostavimo kolor video sekvencu slika sa rezolucijom svake slike od 620x560 piksela i 24 bita/pikselu, trebaće nam oko 1MB memorije po slici. Za prikaz 30 slika u sekundi treba očitati 30 MB podataka sa diska u jednoj sekundi, odnosno, 108 GB u toku jednog sata. Iako postoje uređaji sa velikim memorijskim kapacitetom, nemoguće je, zbog brzine prenosa postojećih uređaja, u realnom vremenu prikazati toliku količinu podataka. Brzina prenosa bi trebala biti 30 MB/sekundi, dok postojeći uređaji, kao, npr. CD-ROM imaju brzinu prenosa Kx150 KB/sekundi, što sa vrijednošću za K od 52 iznosi 7.8 MB/sekundi. Dakle, jedino rješenje se nalazi u kompresiji slike.



Slika 106. [16] Memorijski zahtjevi za enciklopediju prije i poslije kompresije

Treći ralog za kompresiju je ograničen propusni opseg komunikacionih mreža. Propusni opseg tradicionalnih mreža je desetak Mb/sekundi, što je isuviše sporo i za prenos samo jedne slike u nekompresovanoj formi.

Moderne tehnike za kompresiju slike nude rješenje za ove probleme tako da je moguće izvršiti kompresiju mirne slike čak i do 50 puta bez znatnog uticaja na kvalitet reprodukovane slike. U slučaju sekvence slika stepen kompresije može biti i veći.

Klasifikacija tehnika za kompresiju slike

Kompresija digitalnih podataka, pa prema tome i slike, se zasniva na različitim algoritmima koji koriste prostornu redundatnot informacija koje sadrži digitalna slika. Redundantnost informacija u slici postoji zbog znatne prostorne korelacije piksela. Statistička redundantnost je povezana sa raspodelom intenziteta piksela slike i može se eliminisati korišćenjem entropijskih tehnika. Eliminacijom statističke redundantnosti postiže se kompresija bez gubitaka. U slučaju video sekvenci susjedne slike u sekvenci se malo razlikuju (osim u slučaju vrlo brzih pokreta) pa kažemo da pored prostorne redundantnosti postoji i vremenska redundantnost. Vremenska redundantnost se otklanja metodama kompenzacijom pokreta, čime se postiže još veći stepen kompresije video sekvenci nego mirnih slika.

Tehnike kompresije se mogu podijeliti u dvije kategorije:

- kompresija bez gubitaka;
- kompresija sa gubicima.

Tehnike bez gubitaka omogućavaju perfektnu rekonstrukciju originalne slike, dok je kod tehnika sa gubicima rekonstruisana slika slična originalu, ali ne i u potpuosti jednaka.

Tehnike sa gubicima omogućavaju veći stepen kompresije i zbog toga se mnogo više primjenjuju. Kompresija bez gubitaka se koristi u onim slučajevima kada je neophodno sačuvati originalnu sliku, da li zbog nemogućnosti njenog ponovnog snimanja ili važnih informacija koje bi, eventualno, mogle biti izgubljene u postupku kompresije (na primer, medicinski snimci). Ako je lako ponoviti proces snimanja ili se može tolerisati izvestan stepen gubitka informacija (video telefonija, televizija, multimedijalni sistemi, itd.) koriste se postupci kompresije sa gubicima.

KOMPRESIJA BEZ GUBITAKA

Posmatrajmo sivu sliku od $M \times N$ piksela sa mogućih 2^B nivoa svjetline. Ako se za zapis svakog nivoa svjetline koristi B bita, za memorisanje slike u ovoj formi je neophodno MNB bita. Ovaj način zapisa digitalnih slika je poznat kao impulsna kodna modulacija (Pulse Code Modulation - PCM).

Hafmanovo kodovanje

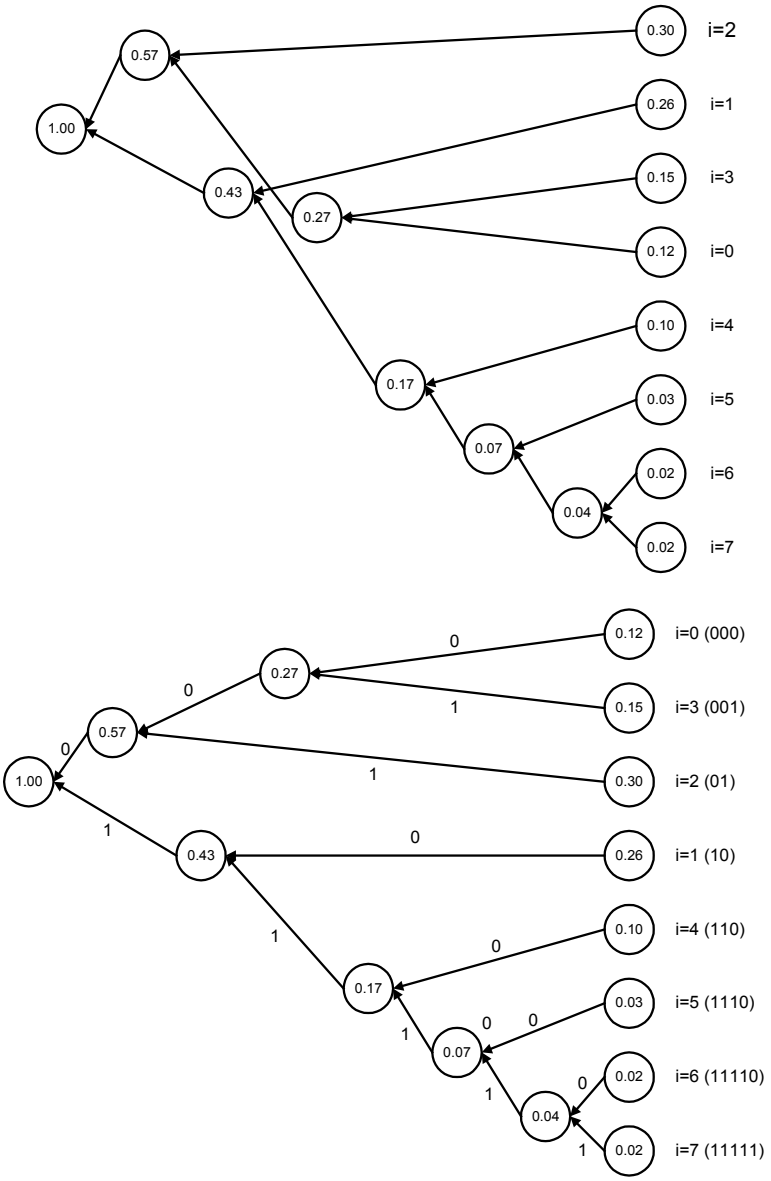
Suština Hafmanovog (Huffman) kodovanja slike leži u tome da se nivoima svjetline koji se češće pojavljuju pridružuju kodne riječi sa manje bita, dok se nivoima svjetline koji se ređe pojavljuju pridružuju kodne riječi sa više bita. Na taj način se postiže da prosječni broj bita po pikselu bude manji od B. Tabela kodnih riječi koja povezuje nivoe svjetline sa kodnim riječima se formira tako da se minimizira prosječna dužina kodne reči. Pri tome su dužine kodnih riječi različite.

Prilikom zapisa slike, sve kodne riječi se spajaju u neprekidan niz bitova, iz koga je na kraju neophodno dešifrovati pojedinačne kodne riječi. Zbog toga nijedna kraća kodna reč ne smije predstavljati prefiks (početak) duže kodne reči.

Primjer:

Pretpostavimo jednostavan slučaj slike sa osam nivoa intenziteta ($B = 3$). Vjerovatnoće pojavljivanja svakog od nivoa svjetline $p(i)$ se mogu procijeniti na osnovu histograma slike. Ove vjerovatnoće se poredaju opadajućim redoslijedom i pridruže se krajnjim desnim čvorovima grafa. Zatim se dva čvora sa najmanjim vjerovatnoćama spajaju i novom čvoru se dodjeljuje zbir njihovih vjerovatnoća. Postupak se nastavlja sve dok zbir vjerovatnoća čvorova koji se spajaju ne postane jednak jedinici, Slika 107. Tako se formira kodno stablo. Ono se zatim preuredi tako da se eliminišu presjeci grana. Granama stabla koje se stiču u korijen pridružuje se vrijednosti 0 (gornja grana) i 1 (donja grana). Ista procedura se nastavlja do krajnjih desnih čvorova. Kodna riječ koja odgovara nekoj

svjetlini se dobiva očitavanjem nula i jedinica koje se nalaze na putu od korijena do čvora koji odgovara toj svjetlini.



Slika 107. Konstrukcija Hafmanovog koda

Hafmanova kodna knjiga predstavlja tabelu preslikavanja svih mogućih nivoa svjetline u odgovarajuće kodne riječi. Hafmanova kodna knjiga za prethodi primjer je data sljedećom tabelom.

Hafmanova kodna knjiga za navedeni primjer ima oblik:

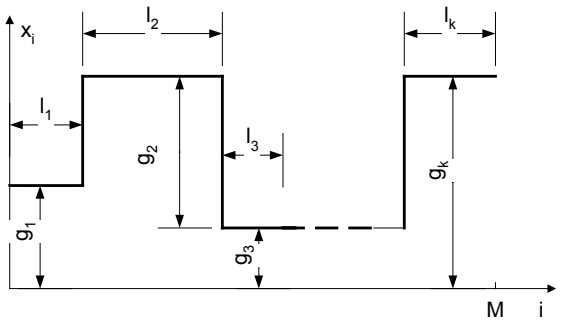
i	**Hafmanov kod**
0	000
1	10
2	01
3	001
4	110
5	1110
6	11110
7	11111

U navedenom primjeru prosječna dužina kodne riječi je $L = 2.55$, pa je realizovani stepen kompresije 3/2.55= 1.176.

Primjenom Hafmanovog koda se ne može postići veliki stepen kompresije kod sivih slika sa velikim brojem nivoa sivog jer je histogram video signala u dužem vremenskom periodu prilično uniforman, te se ovaj kod koristi u kombinaciji sa algoritmima za prediktivno ili transformaciono kodovanje.

Kodovanje dužina nizova

U slučajevima kada imamo mali broj nivoa svjetlina, dešava se da određen broj susjednih piksela u jednoj liniji ima istu vrijednost. Takvu grupu piksela dužine l_i sa nivoom sivog g_i, nazivamo niz. Ako se u jednoj liniji nalazi *k* takvih segmenata, sadržaj linije slike se umjesto pojedinačnim vrijednostima piksela može predstaviti uređenim parovima (g_i, l_i), $1 \le i \le k$, i umjesto kodovanja pojedinačnih piksela može se kodovati cijela grupa. Oaj metod kodovanja se naziva metod kodovanja dužina nizova (run-length coding - RLC).



Slika 108. Grafička prikaz sadržaja jedne linije slike

Svaki par (g_i, l_i) naziva se niz sivog (gray-level run). Veliki stepen kompresije se postiže ako su dužine nizova sivog velike, a posebno u slučajevima binarnih slika, kada postoje samo dvije vrste piksela, vrijednosti 0 i 1. Kod binarnih slika je često veća vjerovatnoća pojave jednog od nizova, bijelog ili crnog. Ako se pretpostavi da linija počinje nizom sa većom vjerovatnoćom, onda se koduju samo dužine nizova, a ne i njihovi nivoi za koje se zna da se smenjuju. Kraj svake linije se označava posebnim kodnim simbolom (end of line – EOL). Kraj slike se označava ponavljanjem EOL simbola, najčešće 6 puta.

RLC kodovanje je standardizovano od strane Međunarodne telekomunikacione unije ITU-T za prenos binarnih slika po javnoj telefonskoj mreži. Osnovna primjena je prenos faksimila.

LZW postupak kompresije

Za kompresiju slike se mogu iskoristiti i postupci namijenjeni kompresiji binarnih datoteka, koji su široko rasprostranjeni u računarstvu. Jedan od takvih postupaka je poznati LZW algoritam, nazvan po inicijalima njegovih autora (Lempel, Ziv, Welch).

Slčno prethodno opisanim algoritmima LZW algoritam koduje nizove simbola, ali za razliku od njih generiše kodnu tabelu u toku postupka kodovanja.

Konstrukcija kodne tabele se vrši i prilikom kodovanja i prilikom dekodovanja slike. Digitalna slika se posmatra kao dugačak jednodimenzionalni niz, sastavljen od podnizova čija dužina može biti različita, zavisno od primjene algoritma (npr. jedan bajt).

Kodovanje

Na početku izvršavanja algoritma se vrši inicijalizacija tokom koje se kodna tabela popuni svim pojedinačnim podnizovima koji se mogu pojaviti na ulazu (npr., ako su dužine podnizova jedan bajt, kodna tabela u početku sadrži borjeve od 0 do 255). Takođe se vrči inicijalizacija promjenljive koja sadrži prefiks niz *P* tako što se ona napuni nizom dužine 0. Zatim se u svakom koraku učitava po jedan podniz slike, smiješta u promjenljivu *Z*, i u tekućoj promjenljivoj *S* se vrši spajanje prefiks niza i sadržaja promenljive *Z*, $S = P + Z$. Ako se sadržaj tekuće promjenljive *S* poklapa sa nekim od postojećih sadržaja u kodnoj tabeli, prefiks niz se ažurira dodavanjem tekućeg niza ($P = S$) i postupak se ponavlja da bi se pronašla najduža moguća sekvenca podnizova iz ulaznog niza koja se već nalazi u kodnoj tabeli. U trenutku kad se desi da nakon dodavanja novog podniza *Z* iz ulaznog niza u tekuću promjenljivu njen sadržaj ne postoji u kodnoj tabeli, (što znači da se samo prefiks sadrži u kodnoj tabeli), kodna riječ prefiksa se šalje u izlazni niz, kodna tabela se dopunjava dodavanjem tekuće promenljive $S = P + Z$ na kraj kodne tabele, a prefiks se postavlja na vrijednost posljednjeg pročitanog niza ($P = Z$). Kada se dođe do kraja slike, u izlazni niz se ubaci kodna reč za kraj datoteke EOI.

Algoritam za kodovanje

1. Na početku kodna tabela sadrži sve pojedinačne podnizove koji se mogu pojaviti na ulazu, a prefiks P je prazan;
2. Z := sljedeći podniz iz ulaznog niza;
3. Postoji li P+Z u kodnoj tabeli?
   - ako postoji, P := P+Z (P proširi Z-om);
   - ako ne postoji,
     I. u izlazni niz pošalji kodnu riječ koja odgovara P-u (pozicija na kojoj je u kodnoj tabeli upisan P);
     II. u kodnu tabelu dodaj P+Z;
     III. isprazni P i u njega upiši samo Z (P := Z);
4. ima li još znakova u ulaznom nizu?
   - ako ima, vrati se na korak 2;
   - ako nema:
     I. u izlazni niz pošalji kodnu riječ koja odgovara P-u;
     II. KRAJ.

Primjer kodovanja:

Sadržaj kodne tabele na početku kodovanja/dekodovanja:

(1) A
(2) B
(3) C

Ulazni niz za kodovanje:

Pozicija	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Podniz	A	B	B	A	B	A	B	A	C

Postupak kodovanja

Korak	**Poz**	**Kodna tabela**	**Izlaz**
1.	1	(4) A B	(1)
2.	2	(5) B B	(2)
3.	3	(6) B A	(2)
4.	4	(7) A B A	(4)
5.	6	(8) A B A C	(7)
6.	--	--	(3)

Dekodovanje

I prilikom dekodovanja LZW algoritmom kodna tabela se progresivno rekonstruiše tako da nije potrebno pamtiti niti prenositi kodnu tabelu. Prije početka dekodovanja kodna tabela izgleda isto kao i prije kodovanja - sadrži sve pojedinačne znakove koji se mogu pojaviti na ulazu.

Radi objašnjenja uzmimo neki trenutak u toku dekodovanja, kad kodna tabela već sadrži i znakove veće dužine. Algoritam zapamti prethodnu kodnu riječ (pK), a zatim učita trenutnu (tK), pronađe u kodnoj tebeli niz.tK (sadržaj kodne tabele koji odgovara kodnoj riječi tK) i pošalje ga u izlazni niz. U kodnu tabelu se zatim doda niz.pK proširen prvim znakom niza.tK. Budući da moramo učitati sljedeću kodnu riječ da bismo znali kojim znakom da proširimo trenutnu, nizove dodajemo u kodnu tabelu sa zakašnjenjem od jednog koraka u odnosu na kodovanje.

Poseban je slučaj ako tK odgovara prvom slobodnom mjestu u kodnoj tabeli (kojem još nije pridružen niz). To se može dogoditi upravo zbog navedenog kašnjenja s dodavanjem nizova u kodnu tabelu. Događa se kad prilikom kodovanja naiđemo ponovo na niz koji smo upravo dodali u kodnu tabelu u prethodnom koraku. Prilikom dekodovanja taj niz još ne postoji u kodnoj tabeli. Budući da se prilikom kodovanja, odmah nakon upisa u kodnu tabelu, vrši izjendačavanje $P = Z$ sljedeći podniz koji kodujemo uvijek počinje sa zadnjim znakom prethodnog. Prema tome, uslov da podniz na ulazu kodera može biti jednak upravo dodatom nizu u kodnu tabelu je da su mu prvi i zadnji znak jednaki. Iz toga izvodimo sljedeće pravilo za dekodovanje: niz.pK proširujemo prvim znakom njega samog te taj niz pošaljemo na izlaz dekodera i dodamo u kodnu tabelu.

Algoritam za dekodovanje

1. Na početku kodna tabela sadrži sve moguće znakove koji se mogu pojaviti na ulazu;
2. tK := prva kodna riječ u ulaznom nizu (odgovara jednom od postijećih nizova u kodnoj tabeli);
3. U izlazni niz pošalji niz.tK (niz iz kodne tabele koji odgovara kodnoj riječi tK);
4. pK := tK;
5. tK := sljedeća kodna riječ u ulaznom nizu;
6. Postoji li tK u kodnoj tabeli?
   - ako postoji,
     1. u izlazni niz pošaljemo niz.tK;
     2. P := niz.pK;
     3. Z := prvi znak niz.tK;
     4. na prvo slobodno mjesto u rječniku dodamo niz P+Z;
   - ako ne postoji,
     1. P := niz.pK;
     2. Z := prvi znak niz.pK;
     3. niz P+Z pošaljemo u izlazni niz i dodamo ga u kodnu tabelu;
7. Ima li još kodova u ulaznom nizu?
   - ako ima, vrati se na korak 4;
   - ako nema, KRAJ.

Primjer dekodovanja:

Postupak dekodovanja

Korak	Kod	Izlaz	Kodna tabela
1.	(1)	A	--
2.	(2)	B	(4) A B
3.	(2)	B	(5) B B
4.	(4)	A B	(6) B A
5.	(7)	A B A	(7) A B A
6.	(3)	C	(8) A B A C

Analizirajmo korak **4.** Prethodnu kodnu riječ (2) smo spremili u pK, a tK je (4). Na izlaz šaljemo niz.tK, odnosno "A B". Nizu.pK ("B") dodajemo prvi znak niza.tK ("A") i to dodajemo kao novi string u kodnoj tabeli ("B A").

Sad slijedi korak **5.** U pK spremamo (4) i pročitamo novi tK = (7). To mjesto u kodnoj tabeli je još prazno. Stoga na niz.pK ("A B") dodajemo prvi znak njega samog, dakle "A" i taj niz ("A B A") dodajemo u kodnu tabelu pod brojem (7), dakle upravo onim koji smo pročitali. To znači da taj isti niz moramo i poslati na izlaz.

Faktor kompresije LZW algoritma primijenjenog na sive slike zavisi od broja nivoa svjetline u originalnoj slici i kreće se od 1.5 do 3. Najbolji faktori kompresije se postižu kod slika sa malim brojem nivoa svjetline, posebno kod binarnih slika, kod kojih faktor kompresije može dostići 5.

Ova metoda je vrlo popularna u praksi. Daljnje razrade dodaju promjenljivu dužinu kodne riječi niza (ovisno o trenutnoj veličini kodne tabele), brisanje starih nizova iz kodne tabele i sl. LZW algoritam i njegove brojne modifikacije osnova su mnogih praktičnih realizacija programa za kompresiju i dekompresiju binarnih datoteka, kao što su ZIP, ARJ i drugi. LZW algoritam je takođe inkorporiran i u mnoge formate za zapis slike sa kompresijom, kao što su TIFF i GIF format. Razlozi za ovako široku primenu LZW algoritma i njegovih modifikacija su što je kompresija brza, bez gubitaka, ima zadovoljavajući stepen kompresije, a može se primijeniti na slike sa ma kakvim brojem bita po pikselu.

Metoda LZW je patentirana, vlasnik patenta je firma Unisys, no ona dozvoljava slobodnu upotrebu metode, osim za proizvođače komercijalnih programa.

Prediktivno kodovanje bez gubitaka

Kako postoji visok stepen korelacije susjednih piksela u slici, vrijednost nekog piksela je moguće uspješno procijeniti na osnovu vrijednosti susjednih piksela. Histogram razlike između originalne i procijenjene slike imaće vrlo mali broj popunjenih nivoa i biće koncentrisan u oblasti oko nule. Za kodovanje ovih razlika umjesto originalne moguće je koristiti kratke kodne riječi, a zbog vrlo neravnomjernog histograma može se efikasnije primijeniti i Hafmanovo kodovanje.

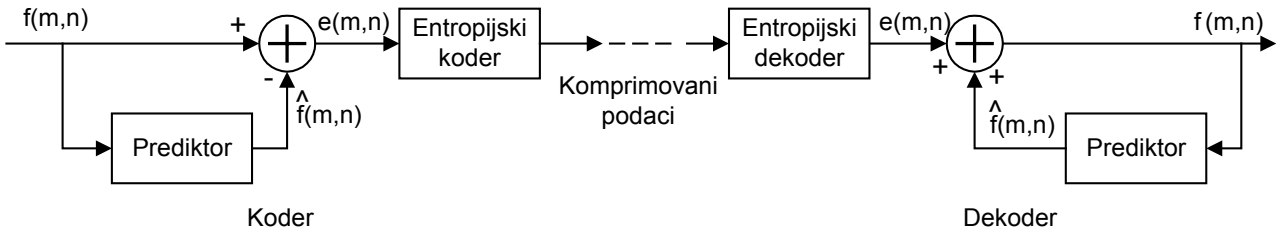
Prediktivne tehnike se najčešće primjenjuju za kompresiju slike sa gubicima, ali se mogu primijeniti i za kompresiju slike bez gubitaka, Slika 109.

Blok za predikciju vrši procjenu vrijednosti tekućeg piksela na osnovu vrijednosti susjednih piksela, koji se nalaze unutar nekog prozora W:

$$\hat{f}(m,n) = P\{f(m-i,n-j)\},\ i,j \in W$$

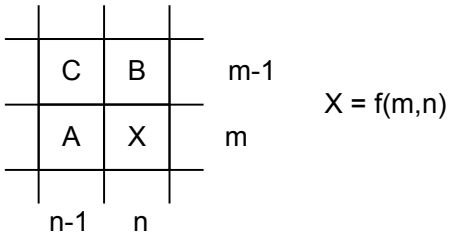
Zatim se izračunava greška predikcije:

$$e(m,n) = f(m,n) - \hat{f}(m,n)$$



Slika 109. Prediktivno kodovanja slike bez gubitaka

Raspored piksela u dvodimenzionalnom prozoru W koji se koristi za predikciju najčešće izgleda ovako:



Za predikciju se koriste vrijednosti piksela iz iste ili prethodne linije, tako da su moguće sljedeće kauzalne predikcione formule:

$$\hat{f}(m,n) = A$$

$$\hat{f}(m,n) = B$$

$$\hat{f}(m,n) = C$$

$$\hat{f}(m,n) = A + B - C$$

$$\hat{f}(m,n) = A + \text{Int}\left[\frac{B-C}{2}\right]$$

$$\hat{f}(m,n) = B + \text{Int}\left[\frac{A-C}{2}\right]$$

$$\hat{f}(m,n) = \text{Int}\left[\frac{A+B}{2}\right]$$

Vidimo da prediktor zapravo predstavlja 1D ili 2D FIR filtar sa malim brojem koeficijenata.

Histogram greške predikcije koja se dobije oduzimanjem procijenjene vrijednosti svakog piksela od odgovarajuće stvarne vrijednosti ima vrlo mali broj nivoa i vrlo izražen maksimum oko nulte vrijednosti, te se može kodovati Hafmanovim koderom, čime se ostvaruje kompresija podataka.

Rekonstruisana slika se dobije inverznim operacijama. Dekodovanjem se dobija rekonstruisana slika greške predikcije. Iz greške predikcije i već rekonstruisanih podataka o pikselima u istoj i prethodnoj liniji, primjenom istovjetnog prediktora se rekonstruiše cijela slika.

Ovom metodom se postižu kompresije sa faktorima vrijednosti oko 2. Zbog svoje jednostavnosti, ova metoda prediktivnog kodovanja bez gubitaka je postala dio JPEG standarda za kompresiju slika.