SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ZAVOD ZA ELEKTRONIČKE SUSTAVE I OBRADBU INFORMACIJA

PROJEKT IZ KOLEGIJA
"NAPREDNE METODE DIGITALNE OBRADE SIGNALA"

KOMPRESIJA SLIKE PRIMJENOM EZW ALGORITMA

DARIO ZRNO
DAVOR ŽIVKO

Siječanj 2005.

Sadržaj:

1 Uvod

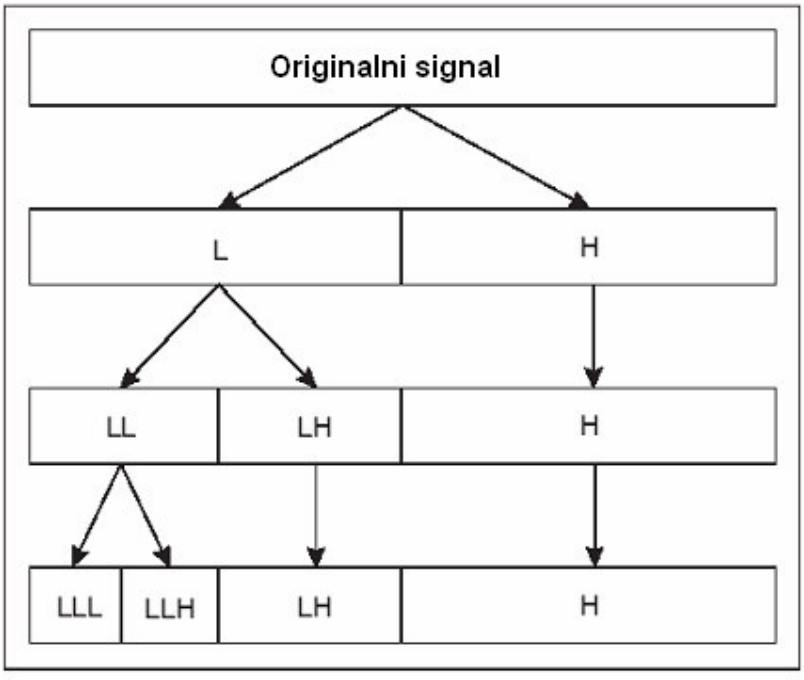
Tijekom proteklog desetljeća, s otkrićem teorije waveleta i višerezolucijske analize, tehnike za obradbu slike temeljene na wavelet transformacijama, bile su predmetom intenzivnog izučavanja. Naročito u području kompresije slike, postoje mnogi algoritmi. Među njima jedan od najznačajnijih je EZW (*Embedded Zerotree Wavelet*) algoritam, kojeg je 1993. opisao J. Shapiro.

EZW algoritam je originalno zamišljen da radi sa slikama tj. 2D signalima, ali ga je moguće primijeniti i na signale drugih dimenzija. Algoritam se temelji na progresivnom kodiranju, koje rezultira u nizu bitova (*bit stream*) rastuće preciznosti. To znači da će, ukoliko se dodaje više bitova u niz (*stream*), dekodirana slika sadržavati više detalja. Progresivno kodiranje (*progressive encoding*) poznato je i pod imenom *embedded encoding*, što objašnjava naziv algoritma. EZW algoritam omogućuje kodiranje sa gubitkom (*lossy*) i bez gubitaka (*loseless*). Kompresija s gubitkom znači da potpuna rekonstrukcija izvornih podataka nije moguća. Koristimo li kodiranje s gubitkom, uz korištenje nekih optimizacijskih postupaka, dolazimo do značajnih rezultata u kompresiji slike.

EZW algoritam koristi diskretnu wavelet transformaciju (DWT), pa će prije detaljnog objašnjenja samog algoritma biti dan i kratak osvrt na osnovna svojstva DWT transformacije. Na kraju, nakon iznošenja teoretske podloge EZW algoritma, bit će prikazani rezultati testiranja implementiranog modela tog algoritma u Matlabu.

2 Diskretna wavelet transformacija (DWT)

Diskretna wavelet transformacija je najčešće korištena wavelet transformacija. To je rekurzivni proces filtriranja ulaznog niza podataka niskopropusnim i visokopropusnim filtrima, kako je prikazano na Slici 1.



Slika 1. Jednodimenzionalna DWT

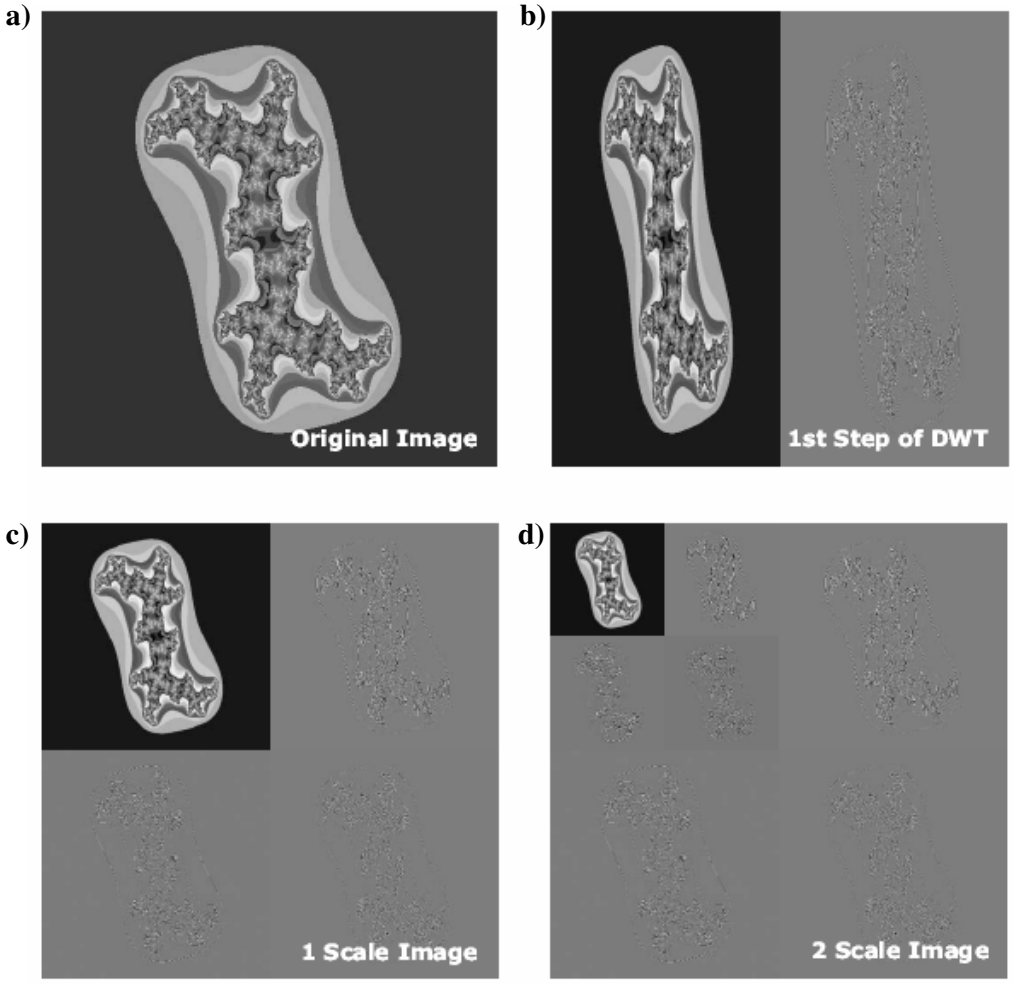
U prvoj iteraciji, cijeli ulazni signal filtrira se niskopropusnim i visokopropusnim filtrom, a rezultati filtriranja pohranjuju se zasebno. Ta dva vektora tvore prvu skalu DWT-a. Skala je parametar wavelet transformacije, koji specificira trajanje waveleta.

Oba filtrirana vektora imaju duljinu istu kao i početni signal. Kako ne bi došlo do ekspanzije podataka, ti se filtrirani signali moraju decimirati s faktorom 2. Koeficijenti u tako generiranom, visokim propustom filtriranom, signalu nazivaju se wavelet koeficijenti.

Signal filtriran visokopropusnim filtrom se pohranjuje, pa se slijedeće iteracije tiču samo vektora nastalog u prvoj iteraciji niskopropusnim filtriranjem. Svaka nova iteracija dodaje novu skalu transformiranoj slici, kao što je prikazano na Slici 1. Za implementaciju procesa filtracije često se koristi i konvolucijski algoritam.

Kako bi se mogla obaviti transformacija slike, potrebna je dvodimenzionalna transformacija. Kvadratna dvodimenzionalna DWT koristi redove jednodimenzionalnih DWT-a, prema algoritmu:

- 1) zamijeniti svaki redak slike njegovom 1D-DWT
- 2) zamijeniti svaki stupac slike njegovom 1D-DWT
- 3) ponoviti korake 1) i 2) na najnižem potpojasu, kako bi se kreirala slijedeća skala
- 4) ponavljati korak 3) dok se ne kreira željeni broj skala



Slika 2. Dvodimenzionalna DWT. **a)** Polazna, netransformirana slika **b)** nakon prvog koraka (svi reci su transformirani). **c)** Transformirana slika s jednom skalom i **d)** s dvije skale

Wavelet skala pokazuje koliko puta je transformacija primijenjena na polaznu sliku. Na Slici 2d) jasno vidimo da je za vrijeme druge iteracije jedino najniži potpojas bio transformiran.

3 EZW Algoritam

3.1 Uvod

EZW (*Embedded Zerotree Wavelet*) algoritam dizajnirao je 1993. godine J. M. Shapiro. Izvorno je zamišljen za kodiranje 2D signala (slika), ali može poslužiti i kod signala većih dimenzija.

EZW koder koristi progresivno kodiranje za kompresiju slike u niz bitova (*bit stream*). Kompresija progresivnim kodiranjem rezultira nizom bitova (*bit stream*) rastuće preciznosti. To znači da će, ukoliko se dodaje više bitova u niz, dekodirana slika sadržavati više detalja, slično kao kod JPEG kodera. Progresivno kodiranje u literaturi je poznato i pod nazivom *embedded coding* – odatle slovo E u kratici EZW.

Kodiranje slike EZW algoritmom, u kombinaciji s nekim optimizacijskim postupcima, rezultira zapanjujuće učinkovitim alatom za kompresiju slike, čiji komprimirani niz podataka može koristiti bilo koju brzinu prijenosa bitova. Naravno, da bi sve brzine bile moguće, negdje mora postojati gubitak informacije. Dakle, radi se o kompresiji s gubitkom. EZW koderom moguće je ostvariti i kompresiju bez gubitaka, ali sa puno lošijim rezultatima.

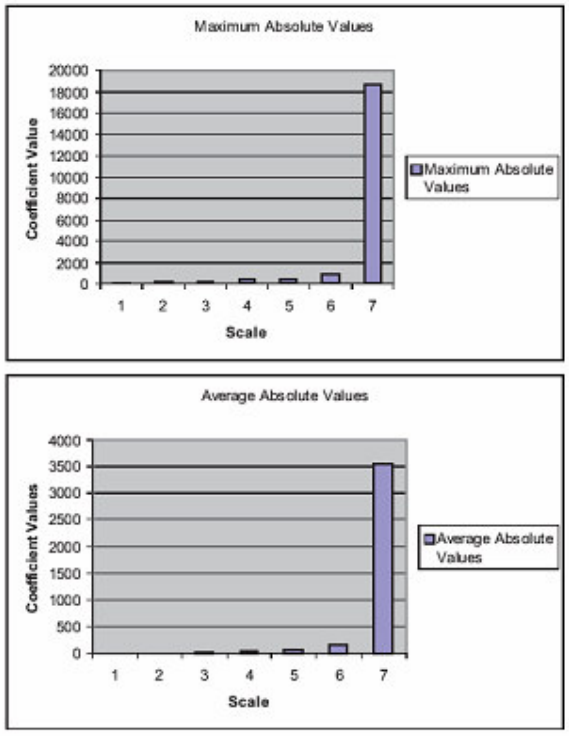
Razvoj EZW algoritma zasnovan je na dvjema činjenicama:

- što je wavelet koeficijent veći, sadržavat će veću količinu informacije, i samim time postaje važniji. Zbog toga, EZW algoritam prvo obrađuje veće wavelet koeficijente
- maksimalne i srednje apsolutne vrijednosti koeficijenata smanjuju se pomicanjem od nižih frekvencijskih potpojasa (najviših skala) ka onim višima, što je ilustrirano Slikom 3

Ove dvije činjenice koriste se u progresivnom kodiranju wavelet koeficijenata u padajućem redu, kroz više iteracija. Za svaki prolaz kroz sliku odabire se prag (*threshold*) u odnosu na kojega se mjere wavelet koeficijenti. Ako je wavelet koeficijent veći od praga, kodira se i uklanja iz slike. Ako je manji, ostavlja se za slijedeći prolaz. Nakon što se ispitaju svi wavelet koeficijenti, prag se smanjuje i slika se skenira ponovo, kako bi se dodalo više detalja kodiranoj slici. Ova procedura se ponavlja dok wavelet koeficijenti nisu u potpunosti kodirani ili je neki drugi kriterij zadovoljen.

Prirodne slike općenito imaju spektar niske frekvencije. Kada se na sliku primijeni wavelet transformacija, energija u potpojasevima će se smanjivati kako se skala smanjuje, tako da će prosječno wavelet koeficijenti u višim pojasevima imati manje iznose od koeficijenata u nižim pojasevima. Dakle, viši pojasevi samo dodaju detalje u sliku. Zato će wavelet koeficijenti u višim pojasevima biti u prosjeku manji od onih u nižim pojasevima. To pokazuje da je progresivno kodiranje najprirodniji odabir za kompresiju slika transformiranih waveletima. Na slici 3. prikazane su ovisnosti maksimalne i prosječne vrijednosti wavelet koeficijenata u ovisnosti o skali.

.



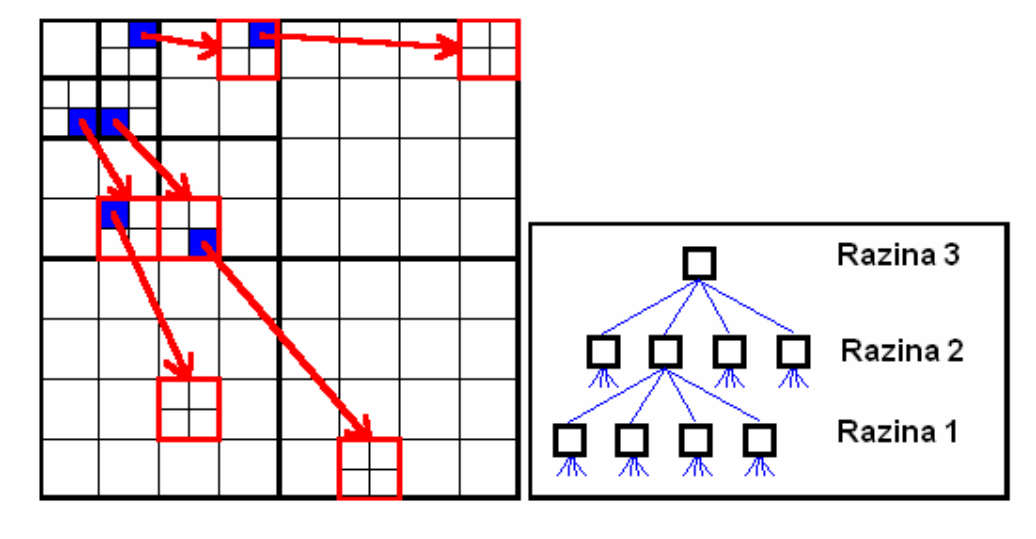
Slika 3. Vrijednosti wavelet koeficijenata kroz skale

Wavelet koeficijenti su dvodimenzionalni. To znači da, ukoliko želimo komprimirati transformirani signal, moramo kodirati ne samo vrijednosti koeficijenata, nego i njihove pozicije u vremenu. Kod 2D signala (slika), poziciju u vremenu lakše je izraziti kao poziciju u prostoru. Nakon transformacije waveletom, slika se može prikazati stablima, što je posljedica decimacije, sastavnog dijela wavelet transformacije. Tako svaki koeficijent u nižem pojasu možemo gledati kao roditelja za četiri potomka u slijedećem višem pojasu. Svaki od potomaka također ima četiri potomka u slijedećem višem pojasu. Svaki korijen, dakle, ima četiri lista, što nas dovodi do kvadratnog stabla, tj. *quad - tree* strukture

3.2 *Quad tree* struktura

Kvadratno stablo, u kojem objekt-roditelj ima četiri objekta-potomka, nazivamo *quad-tree* struktura. Svaki od potomaka roditelj je za nova četiri potomka, i tako dalje. Apsolutna vrijednost objekata u *quad - tree* strukturi smanjuje se od roditelja ka potomku. To je važno svojstvo EZW algoritma jer, u slučaju da koeficijent bude proglašen neznakovitim, svi njegovi potomci također postaju neznakoviti, pa cijela ta grana biva smatrana nositeljem beskorisne informacije.

Koeficijenti u nižim frekvencijskim pojasevima imaju četiri potomka u prvom slijedećem višem pojasu. Svaki od tih potomaka ima četiri potomka u slijedećem višem pojasu. To je prikazano na Slici 4. Važno je primijetiti položaje potomaka u odnosu na roditelje, zbog pohranjivanja informacije o položaju koeficijenata pri EZW kodiranju.



Slika 4. Razlaganje slike na kvadratna stabla

Da bismo u potpunosti opisali naziv algoritma, moramo još definirati pojam nul – stabla (*zerotree*). Nul–stablo je ono kvadratno stablo kojemu su svi čvorovi jednaki ili manji od korijena (*root*). Takvo stablo kodira se jednim simbolom i rekonstruira dekoderom kao kvadratno stablo koje je ispunjeno nulama. Korijen, također, mora biti manji od praga u ovisnosti o kojemu se wavelet koeficijenti mjere.

EZW koder koristi činjenicu da se wavelet koeficijenti smanjuju sa skalom. Pretpostavlja se da je velika vjerojatnost da će svi koeficijenti unutar kvadratnog stabla biti manji od određenog praga ako je korijen stabla manji od tog praga. U tom se slučaju cijelo stablo može kodirati jednim simbolom. Ako skeniramo sliku predefiniranim redoslijedom, krećući se od više skale prema nižoj, mnoge pozicije mogu biti kodirane korištenjem simbola kojim se opisuju nul–stabla u izlaznom nizu bitova.

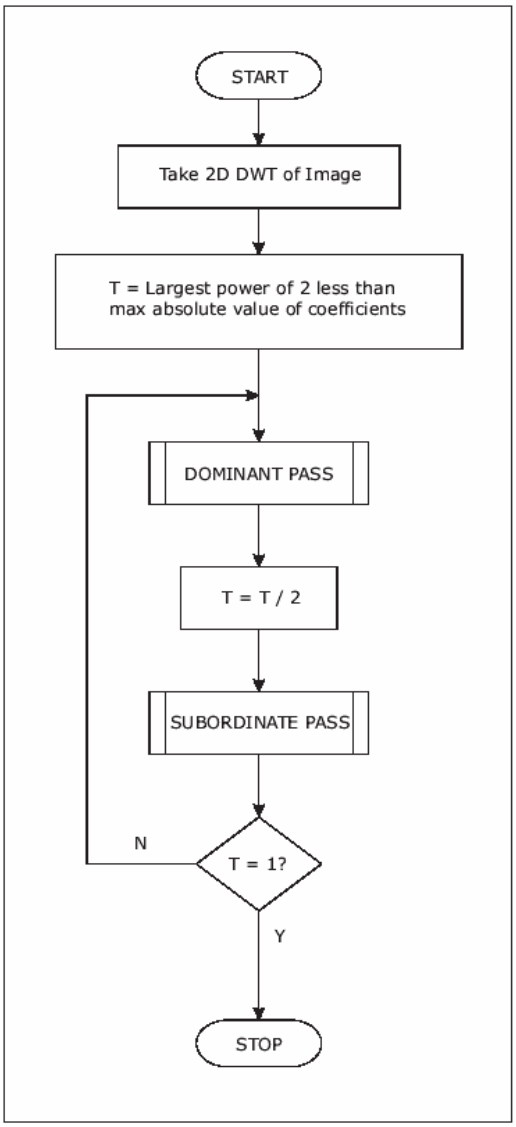
3.3 Algoritam

Prvi korak algoritma je određivanje početnog praga. Ako koristimo *bitplane coding*, početni prag se određuje pomoću formule:

$$T_{poč} = 2^{\lfloor \log_2 k \rfloor} \tag{1}$$

gdje je $T_{poč}$ početni prag, a k maksimalna apsolutna vrijednost koeficijenta unutar slike. Koristeći taj početni prag, tzv. *dominant-pass* procedura primijenjena na sve koeficijente, identificirat će sve koeficijente koji su znakoviti. Znakoviti koeficijenti su svi kojima je apsolutna vrijednost veća od početnog praga. *Dominant-pass* procedura bit će detaljnije objašnjena u dijelu 3.3.1.

Nakon što je *dominant-pass* procedura primijenjena na sve koeficijente, vrijednost praga se prepolavlja, a izvodi se *subordinate-pass* procedura, koja će biti objašnjena u dijelu 3.3.2. Ona progresivno kodira detaljiziranu informaciju o znakovitim koeficijentima. Ovaj se proces ponavlja sve dok se ne postigne željeno stanje. Na Slici 5 prikazan je blok dijagram cijelog tog procesa.



Slika 5. Blok dijagram kompresije slike primjenom EZW algoritma

Željeno stanje postiže se, ili kada prag dosegne minimalnu vrijednost (obično 1, ali može i manje), ili kada se zadovolji neka prije određena funkcija. Ta funkcija je najčešće brojač bitova upisanih u komprimirani niz podataka, a završava dostizanjem neke zadane vrijednosti. Kako je broj ulaznih bitova poznat, a broj izlaznih bitova u potpunosti kontroliran, ova funkcija može jamčiti točne omjere kompresije. Druga moguća funkcija je funkcija pogreške, koja prati maksimalnu apsolutnu pogrešku piksela između originalne i rekonstruirane slike, i završava kad pogreška padne ispod zadane granice.

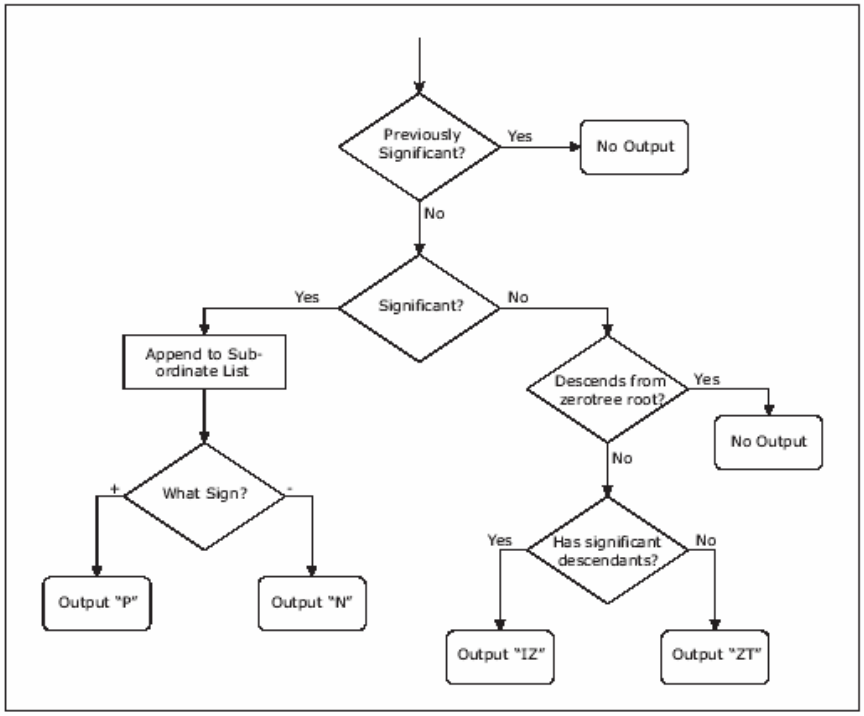
3.3.1 *Dominant pass* procedura

Dominant-pass procedura (dominantni prolaz) pregledava koeficijente na trenutnom pragu, s ciljem identifikacije znakovitih koeficijenata i *zerotree* struktura. Izlaz procedure je jedan od slijedeća četiri simbola, koji su prikazani u Tablici 1.

Vrijednost wavelet koeficijenta	Izlazni kodirani simbol
Wavelet koeficijent veći od praga	**P**
Wavelet koeficijent manji od praga	**N**
Wavelet koeficijent je korijen nul–stabla	**T**
Wavelet koeficijent je izolirana nula	**Z**

Tablica 1. Izlazni simboli u ovisnosti o vrijednosti wavelet koeficijenata

Slika 6 prikazuje kako *dominant-pass* procedura odlučuje koji simbol treba poslati na izlaz. Prvi korak je provjera da li je trenutni koeficijent već prije bio proglašen znakovitim. Ako je, on se jednostavno preskače, a na izlaz se ne šalje ništa. Ako je koeficijent tek u tekućem koraku proglašen znakovitim, njegova se apsolutna vrijednost dodaje na *subordinate* listu, a na izlaz se šalje **P**, odnosno **N**, ovisno o predznaku koeficijenta. *Subordinate* lista sadrži sve znakovite koeficijente.



Slika 6. Dijagram toka *dominant-pass* procedure

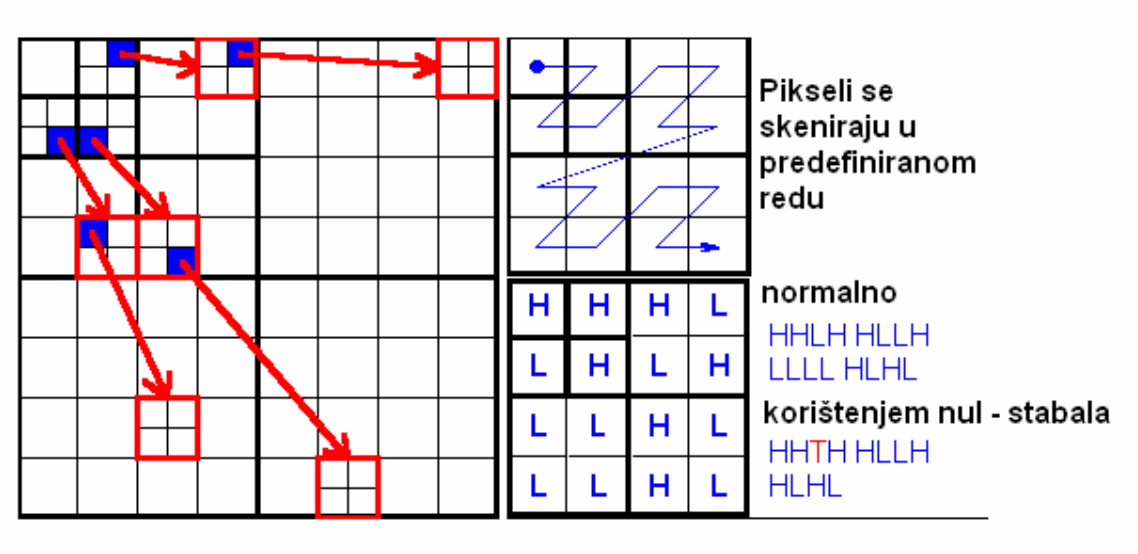
Ukoliko koeficijent nije znakovit, slijedeći je korak provjera da li je taj neznakoviti koeficijent možda potomak nekog već prije otkrivenog nul-stabla. Ako je, na izlaz se ne šalje ništa. Ako koeficijent nije dio već postojećeg *zerotree* stabla, onda on mora biti, ili korijen nove *zerotree* strukture (u tom slučaju, izlaz je **T**), ili pak izolirana nula (izlaz je **Z**).

Red kojim procedura skenira koeficijente vrlo je važan. Koeficijenti u nižim pojasevima moraju biti skenirani prije onih u višim pojasevima. To je potrebno kako bi se maksimizirala vrijednost pojačanja dobivenog korištenjem *zerotree* strukture za kodiranje neznakovitih koeficijenata. Ipak, poredak skeniranja koeficijenata može varirati. Postoje i drugi načini skeniranja, poput Mortonovog redoslijeda, kojeg koristi algoritam kojim ćemo demonstrirati rad EZW algoritma.

Korištenjem ovakvog načina kodiranja wavelet koeficijenata, moguće je postići znatnu uštedu u količini informacije koju je potrebno prenijeti. Naime, EZW koder koristi činjenicu

da se wavelet koeficijenti smanjuju sa skalom. Pretpostavlja se da je velika vjerojatnost da će svi koeficijenti unutar kvadratnog stabla biti manji od određenog praga ako je korijen stabla manji od tog praga. U tom slučaju cijelo se stablo može kodirati jednim simbolom. Ako skeniramo sliku predefiniranim redoslijedom, krećući se od više skale prema nižoj, mnoge pozicije mogu biti kodirane uporabom simbola kojim se opisuju nul–stabla u izlaznom nizu bitova.

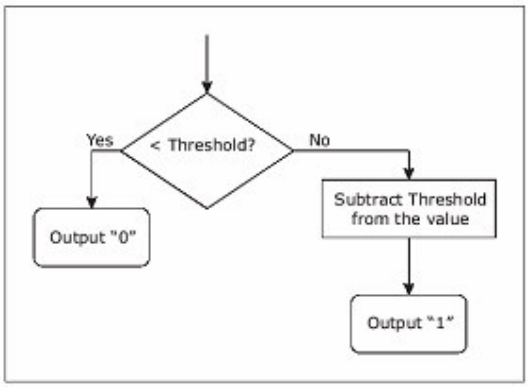
Kako bi se vizualno dočarala ideju algoritma, na Slici 7 je opisan postupak kodiranja vrijednosti koeficijenata u običnom načinu i korištenjem nul – stabala.



Slika 7. Skeniranje i kodiranje wavelet koeficijenata

3.3.2 *Subordinate pass* procedura

Subordinate-pass procedura (povratni prolaz) poznata je i pod nazivom *refinement-pass*, jer ona 'rafinira', pročišćava, vrijednosti znakovitih koeficijenata. Svaki koeficijent koji *dominant-pass* proglasi znakovitim šalje se na *subordinate* listu, gdje *subordinate-pass* procedura provjerava da li mu je trenutna vrijednost veća od trenutne vrijednosti praga. Ako jest, u entropijski koder šalje se 1, a vrijednost koeficijenta sa liste umanjuje se za trenutnu vrijednost praga. Ako je vrijednost koeficijenta manja od trenutne vrijednosti praga, u entropijski koder šalje se 0. Dijagram toka ove procedure prikazan je na Slici 8.



Slika 8. Dijagram toka *subordinate-pass* procedure

Na kraju se *subordinate* lista mora sortirati od najviših prema najnižim vrijednostima, kako bi dekoder mogao reproducirati rezultat. Tako se koeficijenti viših vrijednosti, dakle oni koji nose najviše informacije, prvi kodiraju. Za sortiranje se preporuča neki brzi algoritam, poput *quick sort* algoritma.

Subordinate-pass procedura može čak i biti izostavljena iz EZW algoritma, jer ona ionako samo 'uljepšava' koeficijente. Njenim se izostavljanjem znatno smanjuje veličina koda, vrijeme izvršavanja, te zahtjevi na memoriju, ali se i smanjuje kvaliteta rekonstruirane slike. Također, znatno se smanjuje i odnos signal-šum (SNR). Dakle, izostavljanje *subordinate-pass* procedure iz EZW algoritma je moguće, ali nije preporučljivo. S druge strane, izostavljanje *dominant-pass* procedure uopće nije moguće, jer ona predstavlja srce algoritma.

3.4 Dekodiranje

Izlazni niz bitova EZW kodera mora započeti s nekim informacijama koje će služiti za sinkronizaciju dekodera. Minimalna informacija koju dekoder zahtijeva je broj razina wavelet transformacija i početni prag koji je korišten. Dodatno, možemo poslati dimenzije slike i srednju vrijednost slike. Slanje srednje vrijednosti slike je korisno, jer ako ju uklonimo iz originalne slike prije kodiranja, nakon nepotpune rekonstrukcije u dekoderu, dekoder može zamijeniti rekonstruiranu srednju vrijednost sa originalnom. Time se poboljšava odnos signal/šum (PSNR).

Pri dekodiranju niza podataka, nastalog primjenom EZW algoritma na sliku, također se koriste *dominant-pass* i *subordinate-pass* procedura. *Dominant-pass* iz niza podataka učitava simbole (**P**, **N**, **T** ili **Z**), kako bi se dobila informacija o znakovitosti pojedinog koeficijenta. Inicijalno su sve vrijednosti postavljene na 0. Jako je važno da redoslijed skeniranja pri dekodiranju bude identičan redoslijedu kojim se skeniralo pri kodiranju. Kad procedura naiđe na simbol **P** ili **N**, pozicija trenutnog koeficijenta dodaje se na novu *subordinate* listu. *Dominant-pass* završava kad završi skeniranje svih koeficijenata u slici. *Subordinate-pass* učitava po jedan bit iz niza za svaki koeficijent pohranjen u listi. Taj bit množi se s vrijednošću trenutnog praga, a rezultat se dodaje (ako je koeficijent pozitivan), odnosno oduzima (ako je negativan) od vrijednosti koeficijenta.Dekodiranje završava kad se zadovolji predefinirana funkcija, ili kad se isprazni niz podataka.

Bolji rezultati postižu se malom izmjenom na dekoderu. Umjesto računanja s jednom vrijednošću za svaki koeficijent, dekoder će uzimati u obzir i gornju i donju granicu područja nesigurnosti. Na primjer, neka na vrijednosti praga 64 dekoder učita simbol pozitivnog znakovitog koeficijenta (**P**). Dekoder će vrijednost koeficijenta tada postaviti u 64. Budući je koeficijent znakovit pri pragu 64, on može poprimiti bilo koju vrijednost iz intervala [64,128), što bi ga učinilo neznakovitim za prag 128 (vrijednost mu je svakako manja od 128). Koristeći simbole za *subordinate-pass*, te se granice mogu 'rafinirati'. Naime, ako je prvi simbol 1, originalni koeficijent mora biti barem za 32 veći od praga znakovitosti 64, pa se granice pomiču na [96,128). Ako je, pak, prvi simbol jednak 0, vrijednost originalnog koeficijenta mora biti u intervalu [64,96). Na taj način će, kada dekodiranje završi, vrijednosti svih koeficijenata biti postavljene u središta tih intervala.

4 Uklanjanje šuma iz signala uz korištenje EZW algoritma

EZW kodiranje/dekodiranje s gubitkom (*lossy*) ima funkciju sličnu uklanjanju šuma. Uklanjanje šuma metodom *wavelet shrinkage* sastoji se od uklanjanja wavelet koeficijenata koji su manji od nekoga iznosa. Budući da je šum nekoreliran i maloga iznosa, i wavelet koeficijenti koji se odnose na šum bit će maloga iznosa. Dakako, ova metoda nije idealna,

budući postoje i neki spektri signala koji će biti maloga iznosa, i posljedično također otklonjeni, ali će se njihova odsutnost praktično ostati vizualno neregistrirana.

Postoje dvije popularne metode uklanjanja šuma ovom metodom, a to su: *hard thresholding* i *soft thresholding*. Kod prve metode jednostavno odbacujemo koeficijente koji su manji od neke granične razine. Kod druge metode, odbacujemo sve koeficijente koji su manji od neke razine, a za koeficijente koji su veći od te razine, radimo umanjivanje za iznos te razine.

Kod korištenja *lossy* EZW-a odbacuju se koeficijenti kako bi se zadovoljili neki vizualni kriteriji rekonstrukcije. Odbacuju se mali koeficijenti koji su kodirani na kraju EZW izlaznog niza i radi se modifikacija koeficijenata koji se ne odbacuju. Dakle, ovaj postupak kodiranja je ekvivalentan metodi *soft thresholding*.

Zbog činjenice da se mali koeficijenti na kraju izlaznog niza mogu odbaciti, moguće je u jednom postupku raditi i uklanjanje šuma i kompresiju izlaznih podataka.

5 Primjer

Ogledni primjer kodiranja slike korištenjem EZW algoritma prikazan je na Slici 9. Na njoj vidimo matricu u koju su upisani koeficijenti kodirani EZW algoritmom. Također, prikazan je i redoslijed skeniranja.

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Slika 9. Primjer kodiranja EZW algoritmom

Izlazni niz, podijeljen u dominantni (*dominant-pass*) i povratni prolaz (*subordinate-pass*) prikazan je na Slici 10. Vidimo da količina izlaznih podataka osigurava malu potrošnju memorije i prijenosnog pojasa (*bandwidth*) za spremanje i prijenos podataka.

```
D1:  pnztpttttzttttttttptt
S1:  1O1O
D2:  ztnpttttttt
S2:  1OO11O
D3:  zzzzzppnppnttnnptpttntttttttttptttptttttttttptttttttttttt
S3:  1OO111O1111O11O11OOO
D4:  zzzzzzztztznzzzzpttptpptpnptntttttptpnpppptttttptptttpnp
S4:  11O1111101100100000111011010001001010110O
D5:  zzzzztzzzzztpzzzttptttttnptpptttptttnppnttttpnnptttptpptttt
S5:  1O1111001101000101111101011011001000000001101101100110000111
D6:  zzzttzttttzttttttnnttt
```

Slika 10. Dominantni i povratni prolaz kodiranih podataka

5.1 Praktična primjena EZW Algoritma

Kako bismo dokazali učinkovitost algoritma na realnim slikama, proveli smo kodiranje nekolicine slika uz par modifikacija originalnog koda.

Kao izvor slika koristili smo USC-SIPI bazu slika. Originalna slika prikazana je na Slici 11.

originalna slika

Slika 11. Originalna slika

Originalnu sliku smo nekoliko puta kodirali EZW algoritmom uz promijenjene kontrolne parametre. U prvom slučaju kao granični prag, pri kojem se prekida kodiranje i dekodiranje wavelet koeficijenata, uzeli smo iznos praga 0.5. Kodiranje slike je obavljeno u sedam prolaza. Rekonstruirana slika prikazana je na Slici 12. Vidljivo je da ne postoji odstupanje između originalne i rekonstruirane slike. Ta činjenica je provjerena i računski, oduzimanjem slika i sumiranjem elemenata matrice u kojoj je bila pohranjena razlika.

Slika 12. Rekonstruirana slika

Kao izlazni niz slike, dobili smo niz kodova od ukupno 127344 'p', 'n', 'z' i 't' kodova. Budući da su dimenzije početne slike 256x256 piksela, te da je raspon vrijednosti piksela 256, što znači da je za njihov zapis potrebno 8 bitova po pikselu, očita je znatna ušteda u količini memorije potrebne za zapis slike.

Zavisno o konkretnim potrebama, moguće je korištenjem drugih iznosa kontrolnih pragova, poput iznosa minimalnog praga ili maksimalne vrijednosti izlaznog niza postići razne kvalitete rekonstruirane slike, a to će ovisiti o količini podataka koji će biti kodirani.

Kako bismo ilustrirali ovu tvrdnju, originalnu sliku sa Slike 11 kodirali smo EZW algoritmom, te ju zatim dekodirali nekoliko puta uz promijenjene kontrolne parametre. Budući da je pri kodiranju slike korišteno sedam razina pragova, uz početni iznos praga 64, a konačni 0.5, logičnim se pokazalo da sliku dekodiramo uz promijenjen broj razina dekodiranja.

Na Slici 13 prikazana je rekonstruirana slika nakon samo jednog prolaza EZW dekodera preko izlaznog niza podataka.

Slika 13. Rekonstruirana slika nakon jednog prolaza

Vidimo da je i nakon samo jednog prolaza, uz korištenje samo jednog praga iznosa 64, moguće dobiti rekonstruiranu sliku u kojoj će se poprilično jasno nazirati strukture slike i na kojoj će biti moguće identificirati što se zapravo na njoj nalazi.

Postupak smo ponovili i za dva prolaza dekodera. Rezultat je prikazan na Slici 14.

rekonstruirana slika

Slika 14. Rekonstruirana slika nakon dva prolaza

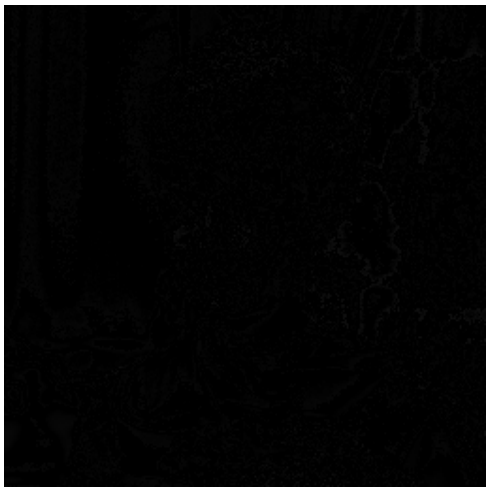
Nakon dva prolaza dekodera, moguće je već uočiti i nijanse unutar rekonstruirane slike. Povećanjem broja prolaza, slici se dodaje sve više detalja. Time će slika svakim prolazom sve više dobivati na dubini. Nakon tri prolaza, rekonstruirana će slika biti sasvim blizu krajnjem rezultatu. Prikazana je na Slici 15.

rekonstruirana slika

Slika 15. Rezultat nakon tri prolaza dekodera

Još uvijek se na slici mogu vidjeti šumovi i izobličenja, ali uz činjenicu da je korišteno samo tri od sedam prolaza, dobivamo uvid u prirodu samih slika i EZW kodera/dekodera. Pogledamo li razliku između originalne slike i rekonstruirane slike, koja je prikazana na Slici 16, vidimo da je odstupanje veoma malo.

razlika originalne i rekonstruirane slike



Slika 16. Odstupanje rekonstruirane slike od originalne za tri prolaza dekodera

Provjerimo li numerički koliko je odstupanje između tih slika, možemo vidjeti kako je maksimalno odstupanje intenziteta piksela unutar slike jednako 15. Suma svih apsolutnih odstupanja piksela cijele slike jednako je 283811. Kako su dimenzije slike jednake 256x256 piksela, dobivamo da je prosječno odstupanje piksela rekonstruirane slike od originalne približno jednako 4. Budući da je prosječna vrijednost intenziteta piksela u originalnoj slici približno jednaka 73, vidimo da je prosječno odstupanje vrijednosti piksela originalne slike od rekonstruirane nakon korištenja tri od sedam prolaza jednako nekoliko postotaka.

Na Slici 17 nalazi se rezultantna slika, koju dobivamo korištenjem četiriju od sedam prolaza dekodera.

rekonstruirana slika

Slika 17. Rezultat nakon četiri prolaza dekodera

Očito će daljnji prolazi dekodera imati mali utjecaj na vizualno poboljšanje kvalitete slike, jer je već nakon četiri prolaza slika vizualno gotovo u potpunosti rekonstruirana.

6 Zaključak

Iz navedenih razmatranja i pokazanih primjera, možemo zaključiti kako je *Embedded zerotree wavelet* algoritam veoma učinkovit alat za kodiranje i kompresiju slike. U slučajevima kada smo u realizaciji sustava ograničeni količinom memorije ili širinom prijenosnog pojasa, postupak podvrgavanja podataka EZW algoritmu prije spremanja ili prijenosa nameće se kao jedino logično rješenje.

7 Literatura

1. E. Kriegler: 'An Implementation of the EZW Algorithm', EZW_24Jan03.pdf

2. C. Valens: 'EZW Encoding', dostupno na URL: http://perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html

3. A. Doran, C. Wang, H. Zhang: 'Embedded zerotree wavelet (EZW) algorithm', dostupno na URL: http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj1/report/node8.html

4. 'An Implementation of EZW', dostupno na URL: http://pesona.mmu.edu.my/~msng/ezw_old.html

5. X. Liu: 'Comparison between wavelet image compression methods', dostupno na URL: http://www.math.sc.edu/~liu/Comparison.pdf

6. USC-SIPI Image database, dostupno na URL: http://sipi.usc.edu/database/

8 Prilog

U prilogu se nalaze MatLab kodovi, korišteni u analizi rada EZW algoritma

Glavna skripta

```
% ************************************************************************
%       GLAVNI PROGRAM ZA DEMONSTRACIJU RADA EZW KODERA I DEKODERA
%        Davor Zivko i Dario Zrno, NMDOS projekt, sijecanj 2005.
% ************************************************************************

% ucitavanje putanje slike
string = input('path ->','s');

% učitavanje slike
[img map]=imread(string);

% konverzija RGB slike u sivu
img2 = rgb2gray(img);

% prikaz originalne slike
figure,imshow(img2),title('originalna slika');

% SKALIRANJE! -> kako bi se mogli postici i p i n simboli
X = double(img2) - 128;
X0=X;

% maksimalni elementi po stupcima
Y0=max(X);
% maksimalni element unutar matrice
Y1=max(Y0);

for i=0:20;         % maksimalno 20 pragova (ako zadovoljavaju uvjete)
    if 2^i<=Y1 & 2^i>0.5*Y1;
        threshold=2^i;
        initialthreshold=threshold;
        % zapamti broj razina
        laststeplevel=i+1;
        break;
    end;
end;

sublist=[];
sub_list=[];

% dimenzije slike
[xx,yy]=size(X);

% mapping kreira Morton scale order matrix-> redoslijed skeniranja
A=mapping(xx);

% veličina Mortonove matrice
[m,n]=size(A);

%postavljanje vektora sa redoslijedom kodiranja kao globalne varijable
global N;

N=zeros(m*n,2);
```

```
% upisuje indekse skeniranja
for i=1:m,
    for j=1:n,
        N(A(i,j),1)=i;
        N(A(i,j),2)=j;
    end
end

% inicijalizacija reda skeniranja
order=1;

% dok je prag razlicit od 0.5, obavlja se dominantpass i subordinantpass
while threshold ~= 0.5,

    threshold

   % Dominant Pass
   [D,X,sublist,sub_list] = dominantpass(X,threshold,sublist,sub_list);
   DD{order}=D
   significantlist{order}=sub_list;

   % Subordinate pass - prepolavlja se prag
   threshold=threshold/2;
   if threshold ==0.5,
       break;
   end
    % zapisuje preko subordinatepass, u SS, preko S, je li neki koeficijent
    % jednak pragu ili nije
   S = subordinatepass(sublist,threshold);
   SS{order}=S
   order=order+1;
end

%*************************************************************************
%                            EZW dekoder
%*************************************************************************

global N;

% velicina inicijalne slike
[m,n]=size(N);

% inicijalizacija prostora i pocetnog praga
XX=zeros(sqrt(m));
threshold=initialthreshold;

% kreiranje matrice indeksa pozicija za sve p i n koeficijente
sublist=[];
for level=1:laststeplevel,
    RR=zeros(size(XX));
    [a,b]=size(DD{level});
    % dominant pass
    i=1; j=1;
    while j<=b,
        if RR(N(i,1),N(i,2))==0
            if DD{level}(j)=='p'
                if threshold==1
                    XX(N(i,1),N(i,2))=threshold;
                else
                    XX(N(i,1),N(i,2))=1.5*threshold;
```

```
                end
            end
            if DD{level}(j)=='n'
                if threshold==1
                    XX(N(i,1),N(i,2))=-threshold;
                else
                    XX(N(i,1),N(i,2))=-1.5*threshold;
                end
            end
            if DD{level}(j)=='t'& A(N(i,1),N(i,2))<=m/4
                % postavlja sve nasljednike nul-stabla na nulu
                RR=checkchildren(i,RR);
            end
            RR(N(i,1),N(i,2))=1;
            i=i+1;
            j=j+1;
        else i=i+1;
        end
    end

% subordinate pass
[xx,yy]=size(significantlist{level});
threshold=threshold/2;

for i=1:xx,
    if level==laststeplevel|threshold==0.5
        break
    end
    if SS{level}(i)==1
        if XX(sub_list(i,1),sub_list(i,2))>0;
            XX(sub_list(i,1),sub_list(i,2))=
fix(XX(sub_list(i,1),sub_list(i,2))+ threshold/2);
        else
            XX(sub_list(i,1),sub_list(i,2))=
fix(XX(sub_list(i,1),sub_list(i,2))-threshold/2);
        end
    end
    if SS{level}(i)==0
        if XX(sub_list(i,1),sub_list(i,2))>0;
            XX(sub_list(i,1),sub_list(i,2))=
fix(XX(sub_list(i,1),sub_list(i,2))-threshold/2);
        else
            XX(sub_list(i,1),sub_list(i,2))=
fix(XX(sub_list(i,1),sub_list(i,2))+threshold/2);
        end
    end
end
threshold;
level;
XX;
end

initialimage=X0;
reconstructedimage=XX;
razlika = XX - X0;

% prikaz rekonstruirane slike i razlike izmedju originala i rekonstrukcije
figure,imshow(uint8(reconstructedimage+128)),title('rekonstruirana slika');
figure,imshow(uint8(razlika)),title('razlika originalne i rekonstruirane
slike');
```

Funkcija za dominantni prolaz:

```
function [D,X,sublist,sub_list]=dominantpass(X,threshold,sublist,sub_list)
%
% Rekurzivna funkcija za racunanje dominantnog prolaza
%
% Autori  : Davor Zivko i Dario Zrno
% Projekt : Kompresija slike EZW algoritmom
% Kolegij : Napredne metode digitalne obrade signala, sk.god. 2004/05
%
% function [D,X,sublist,sub_list] =
% dominantpass(X,threshold,sublist,sub_list)
%
% Ulazi funkcije:
%         X................... matrica koeficijenata
%         threshold........... trenutna vrijednost praga
%         sublist............. matrica znacajnih koeficijenata
%         sub_list............ pozicije znacajnih koeficijenata
%
% Izlazi funkcije:
%         D................... matrica izlaznih simbola
%         X................... matrica koeficijenata
%         sublist............. matrica znacajnih koeficijenata
%         sub_list............ pozicije znacajnih koeficijenata
%

D=[];

global N;

% X je matrica koeficijenata, velicine m*n
[m,n]=size(X);

% referencijska matrica
R=zeros(m);

[a,b]=size(N);

% koeficijent X(1,1) je DC komponenta!
if abs(X(1,1))>=threshold
    sublist=[sublist, abs(X(1,1))];
    sub_list=[sub_list;N(1,1),N(1,2)];
    if X(1,1)>0;
        D=[D,'p'];
    else D=[D,'n'];
    end
    X(1,1)=0;
else D=[D,'z'];
end

% provjera potomaka prvog elementa
for k=2:4,
    if abs(X(N(k,1),N(k,2)))>=threshold,
        sublist=[sublist, abs(X(N(k,1),N(k,2)))];
        sub_list=[sub_list;N(k,1),N(k,2)];
        if X(N(k,1),N(k,2))>0
            D=[D,'p'];
        else D=[D,'n'];
        end
        X(N(k,1),N(k,2))=0;
```

```
        % Znacajni koeficijent se zamjenjuje sa nulom u originalnoj slici
    else
        % Provjera potomaka za 2,3,4
        result = checkdescendants1( k,X,threshold,0);
        if result==1
            D=[D,'z'];
        else
            D=[D,'t'];
            R(N(k,1),N(k,2))=1;
            R=checkchildren(k,R);
        end
    end
end

% provjera ostalih elemenata
for k=5:a,
    if abs(X(N(k,1),N(k,2)))>=threshold,
        % zapisi vrijednost znacajnog koeficijenta!
        sublist=[sublist, abs(X(N(k,1),N(k,2)))];
        % zapisi njegovu poziciju u matrici
        sub_list=[sub_list;N(k,1),N(k,2)];
        if X(N(k,1),N(k,2))>0,
            D=[D,'p'];
        else D=[D,'n'];
        end
        % na mjesto znacajnog koeficijenta upisi nulu
        X(N(k,1),N(k,2))=0;
        % ako se nije ni p ni n,a u R je zapisana 0 sto znaci da nije bilo
        % provjere tog koeficijenta u prethodnim koracima
    elseif R(N(k,1),N(k,2))==0
        % provjera potomaka
        result = checkdescendants1( k,X,threshold,0);

        if result==1,
            % radi se o izoliranoj nuli
            D=[D,'z'];
            % radi se o nul-stablu
        else D=[D,'t'];
            % zapisi 1 u referencijsku matricu za buduce provjere
            % kako bi znao da je vec provjeravan
            R(N(k,1),N(k,2))=1;
            R=checkchildren(k,R);

        end
    end
end
```

Subordinate prolaz:

```
function S = subordinatepass(sublist,threshold)
%
% Funkcija za racunanje povratnog prolaza
%
% Autori  : Davor Zivko i Dario Zrno
% Projekt : Kompresija slike EZW algoritmom
% Kolegij : Napredne metode digitalne obrade signala, sk.god. 2004/05
%
% function S = subordinatepass(sublist,threshold)
%
% Ulazi funkcije:
%         sublist.............. matrica znacajnih koeficijenata
%         threshold............ trenutna vrijednost praga
%
% Izlaz funkcije:
%         S.................... binarna matrica, koja ima 1 za koeficijente
%                               jednake pragu, a 0 za razlicite od praga
%

S=[];

% broj koeficijenata
[m,n]=size(sublist);

% ova petlja zapisuje S = 1 ako je koeficijent jednak pragu,a 0 ako je
% razlicit od praga
for i=1:n;
    if bitand(sublist(1,i),threshold)==threshold,
        % if sublist(1,i)>=threshold,
        S=[S,1];
        % sublist(1,i)=sublist(1,i)-threshold;
    else S=[S,0];
    end
end
```

Provjera potomaka:

```
function   result = checkdescendants1(j,X,threshold,result)
%
% Funkcija za provjeru potomaka. Ako je result=1, koeficijent ima
% najmanje jednog znacajnog potomka
%
% Autori  : Davor Zivko i Dario Zrno
% Projekt : Kompresija slike EZW algoritmom
% Kolegij : Napredne metode digitalne obrade signala, sk.god. 2004/05
%

global N

[m,n]=size(N);

% provjera potomaka
for i=(4*j-3):4*j;
    % ako je rezultat 1 ili ako smo presli dimenzije
    if result==1 | i > m
        break;
    end;
    % ako je vrijednost potomka veca od praga
    if abs(X(N(i,1),N(i,2)))>=threshold
        result=1;
        break;
    else
        % rekurzivno zvanje
        result=checkdescendants1(i,X,threshold,result);
    end;
end;
```

Provjera izravnih potomaka:

```
function   RR=checkchildren(j,RR)
%
% Funkcija za provjeru izravnih potomaka. Ako je naidje na simbol t, tada
% je potrebno u referencijskoj matrici postaviti sve potomke u 1 ->
% imamo nul-stablo
%
% Autori  : Davor Zivko i Dario Zrno
% Projekt : Kompresija slike EZW algoritmom
% Kolegij : Napredne metode digitalne obrade signala, sk.god. 2004/05
%

global N
[m,n]=size(N);

for i=(4*j-3):4*j;
    if  i<=m,
        RR(N(i,1),N(i,2))=1;
        RR=checkchildren(i,RR);
    end
end;
```

Kreiranje redoslijeda skeniranja:

```
function A = mapping(n)
%
% Funkcija za kreiranje redoslijeda skeniranja. Izracunava Mortonovu
% matricu
%
% Autori  : Davor Zivko i Dario Zrno
% Projekt : Kompresija slike EZW algoritmom
% Kolegij : Napredne metode digitalne obrade signala, sk.god. 2004/05
%

if n == 2
    A = [1 2; 3 4];
else
    B = mapping(n/2);
    A = [B B+(n/2)^2; B+(n/2)^2*2 B+(n/2)^2*3];
end
```