PROJEKTNI ZADATAK

Osnove kompresije audio signala

Teorijske osnove

Kompresija signala omogućava značajno smanjenje propusnog opsega potrebnog za prenos digitalnog signala, zadržavajući pri tome njegov percipirani kvalitet. Tehnike kompresije signala generalno mogu da se podele u 2 velike kategorije:

1. **tehnike kompresije bez gubitaka** (eng. lossless compression) i
2. **tehnike kompresije sa gubicima** (eng. lossy compression).

Kod kompresije bez gubitaka posle procesa kodovanja i dekodovanja dobija se signal koji je identičan originalnom ulaznom signalu. Ove tehnike uglavnom se zasnivaju na nekoj vrsti entropijskog kodovanja kod kog se umesto kodova fiksne dužine koriste kodovi kod kojih dužina svakog koda odgovara količini informacije koju odgovarajući element signala nosi, čime se iz signala eliminiše redundansa, odnosno elementi signala koji ne nose nikakvu korisnu informaciju. Dve metode entropijskog kodovanja koje se najčešće koriste u praksi su Hafmanov (Huffman) kod i aritmetičko kodovanje.

S druge strane, kod kompresije sa gubicima posle kodovanja i dekodovanja dobija se signal koji nije istovetan sa originalnim signalom, već ima određeno izobličenje. Osnovna ideja kod ovih tehnika kompresije je da se izvrši kvantizacija signala, odnosno da se kompresovani signal koduje sa manjim brojem bita nego originalni signal, ali tako da izobličenje signala bude što je moguće manje primetno. U kompresiji audio signala ovo se postiže korišćenjem različitih psihoakustičkih efekata, pre svega efekta maskiranja. Kada u zvučnom signalu postoji komponenta relativno velike snage na nekoj frekvenciji, prag čujnosti u okolini ove frekvencije se podiže, čime se povećava i snaga kvantizacionog šuma koju je moguće uneti u maskirani opseg frekvencija bez velikog uticaja na percipirani kvalitet. Pošto raspodela šuma kvantizacije direktno zavisi od amplitudskog spektra kompresovanog signala, kvantizacija se gotovo uvek vrši u frekvencijskom domenu. Za prelazak u spektralni domen koriste se transofrmacije ili filter banke.

Algoritam za kodovanje

U okviru ovog zadatka predstavljen je prost algoritam za kompresiju audio signala koji se sastoji iz sledećih koraka:

Slika 1 – Audio koder

Prvi korak u kodovanju predstavlja analiza signala upotrebom Brze Furijeove Transformacije (FFT). Kako bi se omogućila rekonstrukcija signala bez gubitaka, prelazak u spektralni domen u fazi kompresije, kao i prelazak iz spektralnog u vremenski domen u fazi dekompresije se obavljaju koristeći metodu „preklopi i saberi“ (pogledati vežbu 7). Obzirom da na ulazu postoje dva kanala (levi i desni), vrši se FFT nad svakim kanalom zasebno.

Drugi korak, odnosno kvantizacija se vrši tako što se dobijeni spektralni koeficijenti predstave sa zadatim brojem bita. U ovoj fazi koeficijenti ostaju predstavljeni sa 16-bitnim promenljivim, samo je potrebno izvršiti skaliranje vrednosti. Kvantizacija se može uraditi na prost način tako da su svi spektralni koeficijenti predstavljeni jednakim brojem bita. Drugi način jeste da se koristi psihoakustički model, koji definiše koji spektralni koeficijenti se koduju sa koliko bita.

Nakon ovog koraka vrši se učešljavanje dva kanala, tako što se u izlazni bafer upisuju naizmenično jedna 16-bitna vrednost levog kanala, potom jedna 16-bitna vrednost desnog kanala. Ovako upakovan izlazni bafer predstavlja ulaz u narednu fazu.

Hafmanovo kodovanje predstavlja algoritam za kompresiju signala bez gubitaka (moguća je savršena rekonstrukcija signala). Osnovni koncept je uvođenje bitske predstave simbola, takve da su simboli predstavljeni bitskim rečima različitih dužina. Na osnovu verovatnoće pojavljivanja simbola u signalu se određuje kojim elementima će se dodeliti kraće, a kojim duže reči.

Obzirom da se rešenje pokreće na simulatoru, kako bi se ubrzao proces kompresije/dekompresije, za potrebe ovog zadatka je realizovana zasebna pomoćna aplikacija koja vrši Hafmanovo kodovanje/dekodovanje. Prethodno je potrebno ulaz iz prethodne faze zapisati u izlaznu datoteku.

Prilikom zapisa datoteke u u izlaznu datoteku formira se zablavlje datoteke. Format datoteke kodovane sa realizovanim koderom prikazan je na slici ispod. Zaglavlje datoteke je označeno zelenom bojom, i sastoji se iz oznake „*DSP1*“ koja označava naš format, polja koje predstavlja broj bita korišćenih za kvantizaciju (validne vrednosti 1-16, 16 znači da je kompresija bez gubitaka) i polja koje predstavlja veličinu polja sa podacima.

‘D’	**’S’**	**’P’**	**’1’**	**quantB**	**size**	**data**
8 bit	8 bit	8 bit	8 bit	16 bit	32 bit	Size*8 bit

Alat za Huffmanovo kodovanje možete pozvati iz konzole sa:

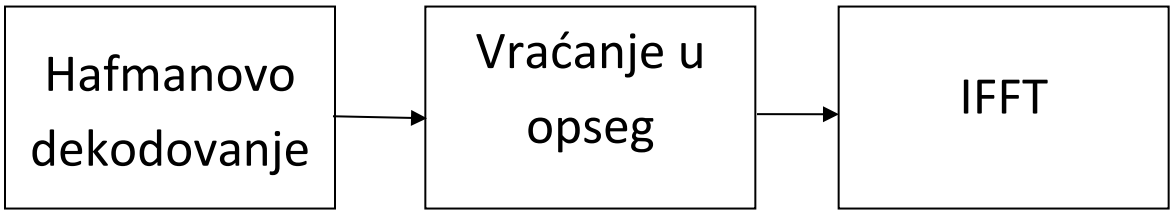
HuffmanEnc <input_file> <compressed_file> <dictionary>

Gde *input_file* predstavlja vašu zadatu .wav datoteku, *compressed_file* predstavlja izlaznu datoteku iz ovog koraka, a *dictionary* predstavlja takođe izlaznu datoteku, koja sadrži korišćeni rečnik simbola se koristi prilikom dekompresije.

Alat vrši samo kompresiju podataka, ne i zaglavlja. Zaglavlje ostaje u istom formatu.

Izlazna datoteka predstavlja komprimovani audio signal, i potrebno joj je dati ekstenziju *.dsp1* (oznaka datoteke u predloženom formatu).

Algoritam za **dekompresiju** signala predstavlja inverzan proces kompresiji:



Slika 2 – Audio dekoder

Dekodovanje datoteke kodovane Hafmanovim algoritmom možete izvršiti pozivom alata iz konzole:

HuffmanDec <compressed_file> <output_file> <dictionary>

Prilikom pokretanja alata potrebno je proslediti kompresovanu datoteku koja je bila izlaz iz prethodnog koraka, kao i rečnik koji je korišćen prilikom kompresije.

Potom u okviru aplikacije na DSP procesoru potrebno je izvršiti učitavanje dobijenog izlaza iz Hafman dekodovanja. Potrebno je pročitati zaglavlje datoteke kako bi se proverila dužina podataka i parameter quantB koji će biti korišćen u procesu dekodovanja. Potom je potrebno nad dobijenim podacima pokrenuti funkciju za dekodovanje.

Prvo se vrši učitavanje jednog okvira podataka za levi kanal, i vrši dekompresija istog. Potrebno je vrednosti vratiti u originalni opseg koristeći parametar quantB. Potom uraditi inverznu FFT koristeći algortam „preklopi i saberi“. Potom se vrši učitavanje sadržaja za desni kanal i dekompresija istog.

Za svaki kanal se posebno radi vraćanje u opseg. Vraćanje u opseg podrazumeva operaciju inverznu kvantizaciji. Odnosno sve koeficijente koji su u ulaznoj datoteci bili predstavljeni sa B bita, potrebno je skalirati nazad na 16-bitni opseg vrednosti.

Inverznu FFT je potrebno uraditi primenom metode “preklopi i saberi”.

Zadaci

Zadatak 1

Realizovati audio **(en)koder** predstavljen na slici 1.

1) Prvi korak je prmena brze Furijeove transformacije za prelazak u spektralni domen. Potrebno je koristiti metodu „preklopi i saberi“. Veličina ulaznog okvira je 128 odbiraka, a veličina bloka nad kojim se radi FFT je 256 (dva uzastopna bloka). Za prozoriranje signala koristiti prozorsku funkciju Vorbis.
2) Drugi korak predstavlja kvantizacija dobijenog spektra koristeći B bita.
3) Treći korak jeste primena Hafmanovog kodovanja koristeći alat HuffmanEnc

Između drugog i trećeg koraka potrebno je izvršiti zapisivanje komprimovanog signala u datoteku.

Prilikom zapisa datoteke u u izlaznu datoteku formira se zaglavlje datoteke. Format datoteke kodovane sa realizovanim koderom prikazan je na slici ispod. Zaglavlje datoteke je označeno zelenom bojom, i sastoji se iz sledećih polja:

- oznaka „DSP1“ koja označva naš format
- quantBL – broj bita korišćenih za kvantizaciju levog kanala (validne vrednosti 1-16, 16 znači da je kompresija bez gubitaka)
- quantBR – broj bita korišćenih za kvantizaciju desnog kanala (validne vrednosti 1-16, 16 znači da je kompresija bez gubitaka)
- MS – oznaka da li se koristi združeno kodovanje (biće korišćeno u narednom zadatku, za sada 0)
- size - veličina polja sa podacima (u bajtima)

Polje data sadrži komprimovane podatke. Naizmenično se menjuju jedan okvir (frame) podataka za levi kanal, potom jedan okvir za desni. Okviri su fiksne dužine, po 256 odbiraka.

‘D’	’S’	’P’	’1’	quantB	size	data
8 bit	8 bit	8 bit	8 bit	16 bit	32 bit	Size*8 bit

frameL 0	frameR 0	frameL 1	frameR 1	frameL 2	frameR 2	...	frameR N
256*16	256*16	256*16	256*16	256*16	256*16		256*16

Pozivati alat za Huffmanovo kodovanje i dekodovanje iz konzole.

Potom realizovati audio **dekoder** predstavljen na slici 2.

1) Prvi korak jeste primena Hafmanovog dekodovanja koristeći alat HuffmanDec
2) Drugi korak predstavlja vraćanje u opseg pozivanjem funkcije za rekonstruisanje.

3) Treći korak jeste sinteza, sa istim parametrima kao prilikom prvog koraka enkodera. Za prozoriranje signala koristiti prozorsku funkciju Vorbis

Isprobati enkodovanje i dekodovanje sa parametrima koji se nalaze u datoteci Zadatak1.txt i popuniti tabelu. Stepen kompresije predstavlja količnik veličine originalne i komprimovane datoteke. Prilikom računanja stepena kompresije ukupnu veličinu koprimovane datoteke izračunati kao zbir veličina komprimovane datoteke (.dsp1) i rečnika.

OČEKIVANI IZLAZ IZ ZADATKA:

- Datoteke sa izvornim kodom: EncoderMain.c, DecoderMain.c, encode.c, decode.c i (sve ostale datoteke koje ste menjali/dodavali).
- Komprimovanu datoteku Zadatak1-N.dsp1, rečnik Zadatak1-N.dict i izlaznu datoteku nakon dekodovanja out_Zadatak1-N.wav, za svaku od kolona iz Zadatak1.txt. N zameniti rednim brojem kolone u tabeli.
- Popunjena tabela Zadatak1.txt
- Slike Z1_ Enc_FFT_buffer1.png i Z1_Enc_FFT_buffer1_Spektar.png koja sadrži prikaz fft_buffer u vremenskom i frekventnom domenu u trenutku pre primene prozorske funkcije u okviru enkodera. Prikaz napraviti kada se na ulaz dovede signal1.wav, za bilo koji okvir ulaznih podataka, osim prvog.
- Slika Z1_Enc_FFT_buffer2.png koja sadrži prikaz sadržaja fft_buffer nakon prozoriranja za isti okvir kao u prethodnoj tački.
- Slika Z1_Enc_FFT_buffer3.png koja sadrži prikaz sadržaja fft_buffer nakon poziva rfft za isti okvir kao u prethodnoj tački;
- Slika Z1_Enc_FFT_buffer4_B.png koja sadrži prikaz fft_buffer nakon kvantizacije sa B bita (B u nazivu slike zameniti brojem bita sa kojim se kvantizuje). Sliku napraviti za isti okvir kao u prethodnoj tački
- Slika Z1_Dec_FFT_buffer1.png koja sadrži prikaz sadržaja fft_buffer nakon poziva rifft u okviru dekodera, za isti okvir kao u prethodnoj tački.
- Slika Z1_Dec_FFT_buffer2.png i Z1_Dec_FFT_buffer2_Spektar.png koje sadrže prikaz uspešno dekodovanog signala u vremenskom i frekventnom domenu.

*** Kontrolna tačka 1 ***

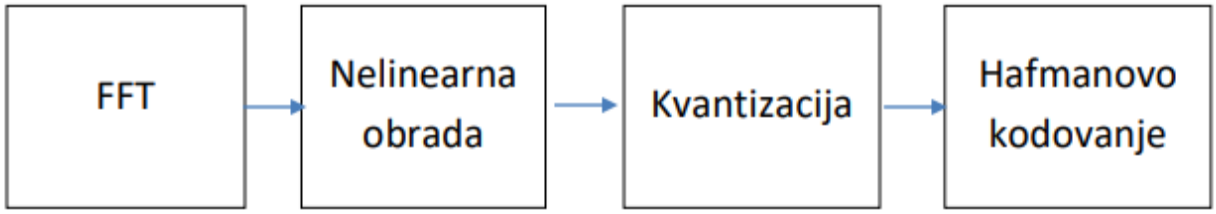
Zadatak 2

Proširiti koder iz prethodnog zadatka tako da se pre kvantizacije koeficijenata nad svakim brojem izvrši predprocesiranje koje čini nelinearna obrada po sledećoj formuli:

$$X = sign(X) * \sqrt{X}$$

Sign predstavlja funkciju koja vraća 1 ukoliko je X pozitivan broj, odnosno (-1) ukoliko je X negativan broj. Za računanje kvadratnog korena iskoristiti funkciju sqrt16 (DSPlib – PDF se nalazi u vežbi 3).

Prošireni sistem je dat na slici ispod:



Modifkovati dekoder iz prethodnog zadatka tako da vrši inverznu operaciju pomenutoj nelinearnoj obradi, odmah nakon vraćanja vrednosti u opseg (za kvadriranje broja pomnožiti ga samog sa sobom koristeći _smpy).

OČEKIVANI IZLAZ IZ ZADATKA:

- Datoteke sa izvornim kodom: encode.c, decode.c i (sve ostale datoteke koje ste menjali/dodavali).
- Komprimovanu datoteku Zadatak2-N.dsp1, rečnik Zadatak2-N.dict i izlaznu datoteku nakon dekodovanja out_Zadatak2-N.wav, za svaku od kolona iz Zadatak2.txt. N zameniti rednim brojem kolone u tabeli.
- Popunjena tabela Zadatak2.txt

Zadatak 3

Proširiti koder iz prethodnog zadatka tako da je moguće pre poziva funkcije za enkodovanje aktivirati opciju Združeno kodovanje (Joint Coding). Osnovna ideja je da se ne koduju kanali kao takvi (levi i desni), nego da se izračunaju vrednosti dva nova kanala M i S, gde M predstavlja srednju vrednost odbiraka levog i desnog kanala, a S polovinu razlike levog i desnog kanala.

$$M(i) = \frac{InputBufferL(i) + inputBufferR(i)}{2}$$

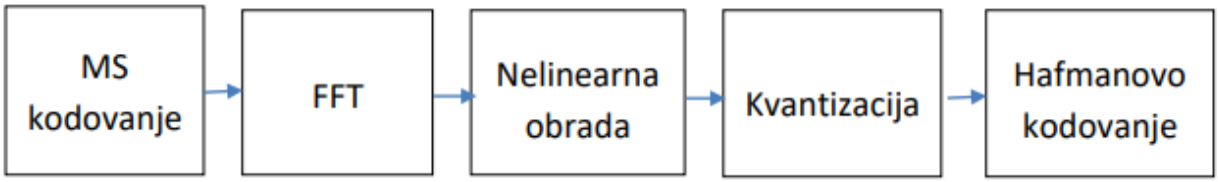
$$S(i) = \frac{InputBufferL(i) - inputBufferR(i)}{2}$$

Ovakav pristup dovodi do značajno bolje kompresije u slučaju da su sadržaji levog i desnog kanala podjednaki (S u većini slučajeva 0). Za slučaj da je ova opcija omogućena, potrebno je u zaglavlju postaviti polje MS na 1, u suprtonom na 0. Kanali M i S se prosleđuju funkciji za enkodovanje na isti način kao što je to rađeno sa L i R. Sa dekoderske strane rekonstrukcija iz MS u LR se vrši na sledeći način:

$$InputBufferL(i) = M(i) + S(i)$$

$$InputBufferR(i) = M(i) - S(i)$$

Novonapravljeni enkoder ima blok dijagram dat sa:



OČEKIVANI IZLAZ IZ ZADATKA:

- Datoteke sa izvornim kodom koja sadrži implementaciju MS kodovanja i dekodovanja.
- Komprimovanu datoteku Zadatak3-N.dsp1, rečnik Zadatak3-N.dict i izlaznu datoteku nakon dekodovanja out_Zadatak3-N.wav, za svaku od kolona iz Zadatak3.txt. N zameniti rednim brojem kolone u tabeli.
- Popunjena tabela Zadatak3.txt

Zadatak 4

Modifikovati enkoder iz prethodnog zadatka tako da se omogući poseban vid kvantizacije, koji će biti aktiviran kada se funkciji prosledi parametar **B=0**.

U tom slučaju kvantizaciju izvršiti na sledeći način:

- Opseg 0 – 100 Hz – anulirati
- Opseg 100Hz – 512Hz – kvantizovati sa 8 bita
- Opseg 512Hz – 4096Hz – kvantizovati sa 12 bita
- Opseg 4096Hz – 14336Hz – kvantizovati sa 6 bita
- Sve iznad 14336Hz anulirati

Omogućiti da dekoder na isti način vrši vraćanje vrednosti u opseg ukoliko iz zaglavlja pričita da je vrednost **B=0**.

OČEKIVANI IZLAZ IZ ZADATKA:

- Datoteke sa izvornim kodom: encode.c, decode.c i (sve ostale datoteke koje ste menjali/dodavali).
- Komprimovanu datoteku Zadatak4-N.dsp1, rečnik Zadatak4-N.dict i izlaznu datoteku nakon dekodovanja out_Zadatak2-N.wav, za svaku od kolona iz Zadatak2.txt. N zameniti rednim brojem kolone u tabeli.
- Popunjena tabela Zadatak4.txt • Slika Z4_Enc_FFT_buffer1.png koja sadrži prikaz sadržaja fft_buffer nakon poziva rfft za proizvoljan okvir podataka;
- Slika Z1_Enc_FFT_buffer2.png koja sadrži prikaz fft_buffer nakon kvantizacije za B=0. Sliku napraviti za isti okvir kao u prethodnoj tački

*** Kontrolna tačka 2 ***

Za odbranu projektnog zadatka 2 napisati kratak izveštaj po uzoru na projektni zadatak 1. Potrebno je na odbranu poneti svoj laptop računar ili u slučaju nemogućnosti, blagovremeno obavestiti svog asistenta.