SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 877

Određivanje vektora značajki govornog signala pomoću procesora za digitalnu obradbu signala

Josip Budulica

Zagreb, lipanj 2013

Zahvala

Želim se zahvaliti svome mentoru prof.dr.sc. Davoru Petrinoviću na pomoći i savjetima tijekom izrade ovog rada, te na dobroj volji i angažmanu tijekom njegovog mentorstva.

1 Sadržaj

2 Popis slika

3 Popis tablica

1 Uvod

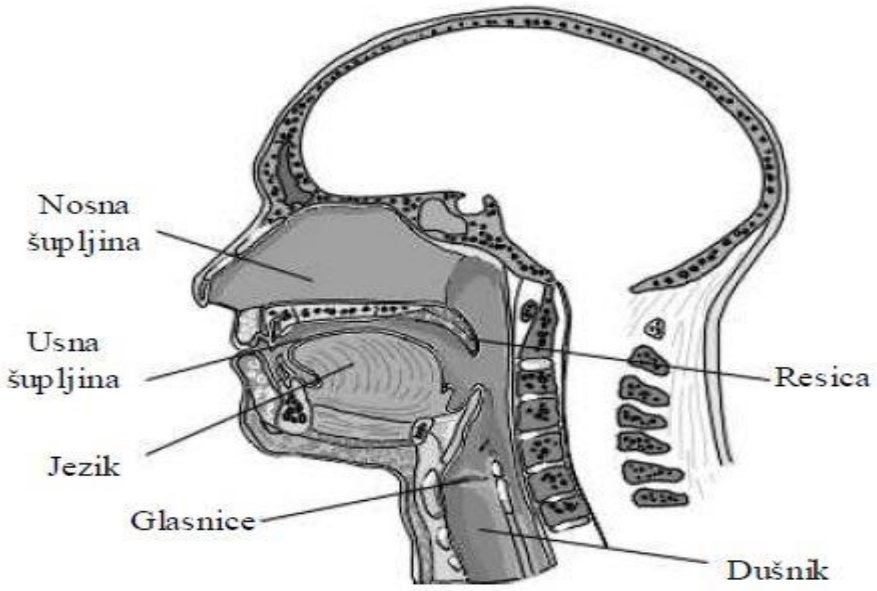
Govor je signal koji nosi informaciju (akustički valni oblik), a osnovna mu je namjena komunikacija. Gledajući govorni signal u vremenskoj domeni vrlo malo se može reći o njegovim karakteristikama. Istraživanja u područjima matematike, akustike i govorne tehnologije pridonijela su u stvaranju mnogih metoda za valjanu reprezentaciju govornog signala. Takve metode imaju mogućnost sažimanja velikog broja podataka sadržanih u govornom signalu u mali skup podataka koji tada sadrži karakteristike danog govornog signala. U narednim poglavljima objašnjen je proces stvaranja govora i neke tehnike reprezentacije govornog signala u svrhu automatskog prepoznavanja govora ili govornika. Također, za potrebe ovog rada, implementiran je i sustav za prepoznavanje samoglasnika. Implementacija je izvedena u programskoj okolini Matlab, programskom jeziku C (u frakcionalnoj aritmetici) te na razvojnom sustavu za digitalnu obradbu signala (TMS320VC5505 eZdsp usb stick).

2 Proces nastajanja govora

2.1 Anatomija govornog sustava

Formiranje govora, kao i bilo koja svjesna radnja čovjeka, počinje od mozga. U njemu se pojavljuje poruka koja se zatim pretvara u skup neuronskih signala koji uptavljaju postupkom artikulacije. Artikulatorima nazovamo organe koji sudjeluju u formiranju zvučnog signala koji sadrži informaciju poruke koja se želi prenijeti (jezik, usnica, glasnice, itd.). Važnu ulogu u procesu nastajanja govora imaju i pluća govornika koja se pod djelovanjem mišića prsnog koša stišću i potiskuju zrak kroz vokalni trakt. Vokalni trakt se u širem smislu sastoji od slijedećih osnovnih dijelova:

• prostor između glasnica, glottis,
• pharynx ili ždrijelo (veza usta i jednjaka),
• usna šupljina,
• jezik,
• stražnje (meko) nepce,
• srednje nepce,
• prednje (tvrdo) nepce,
• nadzubno meso,
• zubi,
• usne,
• velum ili resica koja zatvara usnu šupljinu prema nosnoj i
• nosna šupljina koja završava s nosnicama.



Slika 1 Presjek i osnovni dijelovi vokalnog trakta koji sudjeluju u produkciji govornog signala

2.2 Glasnice

Značajniju ulogu u procesu formiranja govora imaju glasnice. Smještene su na vrhu dušnika (engl. *trachea*) te djeluju u kombinaciji sa strujanjem zraka. Prilikom strujanja zraka, koji dolazi iz pluća, glasnice osciliraju. Ovakvo ponašanje je slično ponašanju piska (engl. *reed*) kod zviždaljke ili kod puhačkih instrumenata s piskom. Na frekvenciju titranja glasnica najviše utječu pritisak zraka iz pluća te napetost samih glasnica koju je moguće kontrolirati. Kada su napete, svojim periodičkim titranjem glasnice formiraju periodičku struju zraka koja zatim prolazi kroz ostatak vokalnog trakta i tako formira zvučne glasove.

2.3 Vokalni trakt

Vokalni trakt se ponaša kao svojevrstan filtar, koji će oblikovati spekar pobudnog signala. Koje se spektralne komponente pojačavaju a koje prigušuju, odlučivat će geometrijski oblik vokalnog trakta. Geometrijski oblik vokalnog trakta je naravno promjenjiv, a određuje ga položaj artikulatora govornika. Dakle, pored karakteristika pobudnog signala vokalog trakta, na formiranje glasa utječu:

- položaj jezika,
- položaj usana,
- položaj čeljusti i
- položaj resice.

Mnogi su istraživači u prošlosti bili zainteresirani te su pokušavali otkriti na koji točno način vokalni trakt utječe pri proizvodnji glasa. Mađarski znanstvenik Johann Wolfgang Ritter von Kempelen de Pazmand, koji je živio od 1734. Do 1804. je još prije dva stoljeća gradio mehaničke modele vokalnog trakta koji su bili sposobni proizvesti zvuk sličan govoru. Već i takva rana istraživanja su potvrdila da je upravo oblik vokalnog trakta, tj. oblik zračnog prostora kroz koji se zvučni val širi odgovoran za karakteristike zvuka koji nastaje. Kao što je već rečeno, na taj oblik je moguće utjecati artikulatorima. Tako npr. dizanje ili spuštanje donje čeljusti otvara ili zatvara zračni prostor. Obzirom da se u tom prostoru nalazi i jezik, svjesna promjena njegovog oblika može značajno izmijeniti oblik volumena tog slobodnog zračnog prostora. Usnice su također vrlo pokretljive, pa se izmjenom njihovog oblika i otvora može utjecati na oblik volumena vokalnog trakta na samom njegovom izlazu.

3 Automatsko prepoznavanje govora i govornika

3.1 Automatsko prepoznavanje govora

Zadaća automatskog prepoznavanja govora jest pretvaranje govorne poruke u tekstualni oblik. Takve aplikacije nalaze široku primjenu. Kao primjer, korisnik može izgovoriti određenu naredbu koju računalo mora prepoznati te sukladno njoj izvršiti određenu radnju. Takve naredbe su načešće jednostavne izolirane riječi iz relativno malog skupa mogućih naredbi (npr. 100 različitih riječi). Postoje primjene automatskog prepoznavanja govora u kojima računalo mora prepoznati i pohraniti riječi izgovorene od strane korisnika. Računalo tada ima ulogu tajnice. Složenost ovog sustava je nešto veća od prošlog primjera jer je za ovakvu aplikaciju potreban mnogo veći skup riječi. Najsloženiji problem prepoznavanja govora predstavlja prepoznavanja spojenog (prirodnog) govora s riječima iz praktički neograničenih rječnika. Jedan od velikih problema sustava za prepoznavanje jest i zavisnost o govorniku. U nekim aplikacijama od sustava se traži da jednako dobro rade za sve govornike dok se u nekim aplikacijama zahtijeva rad sustava za samo jednog ili ograničenog skupa govornika. Svi takvi sustavi su direktno vezani uz jezik koji se koristi, jer univerzalni sustavi koji bi radili za bilo koji jezik nisu niti približno tako dobri kao oni koji su projektirani za svaki jezik nezavisno. To je zato što se bolja učinkovitost prepoznavanja postiže ugradnjom fonetičkih i lingvističkih pravila u sustav prepoznavanja, a koja su naravno različita za svaki jezik. I pored svih navedenih problema, današnji sustavi za engleski jezik trenirani za dotičnog korisnika mogu postići točnost prepoznavanja od 95% za prirodni (vezani) izgovor s riječima iz vrlo velikih rječnika. Kako se radi o vrlo složenim algoritmima, mogućnosti sustava za prepoznavanje su ovisne o procesnim mogućnostima sklopovske platforme.

3.2 Prepoznavanje govornika

Osim prepoznavanja govora, važnu primjenu nalazi i sustav za prepoznavanje govornika. U takvim sustavima čuvaju se podaci o pojedinim govornicima te se na temelju govorne aktivnosti korisnika određuje o kojem se točno korisniku radi. Za svakog potencijalnog kandidata iz tog konačnog skupa govornika izračunava se vjerojatnost da je snimljena govorna sekvenca izgovorena upravo od strane tog govornika. Sortiranjem takve liste vjerojatnosti dobivaju se najvjerojatniji kandidati za prepoznavanje. Smatra se da sustav radi dobro ako je vjerojatnost pravog govornika mnogo veća od vjerojatnosti bilo kojeg drugog krivog govornika iz te baze govornika. Ponekad je zbog visoke sličnosti dvaju glasova nemoguće sa visokom vjerojatnošću izdvojiti jednog govornika od drugog.

U takvim su slučajevima moguće zamjene, jer su objektivno karakteri ta dva glasa vrlo slični. Zbog toga, prilikom projektiranja ovakvog sustava važno je uzimati informacije u glasu koje su uvjetovane fizikalnim značajkama govornika kao što su oblik i duljina vokalnog trakta.

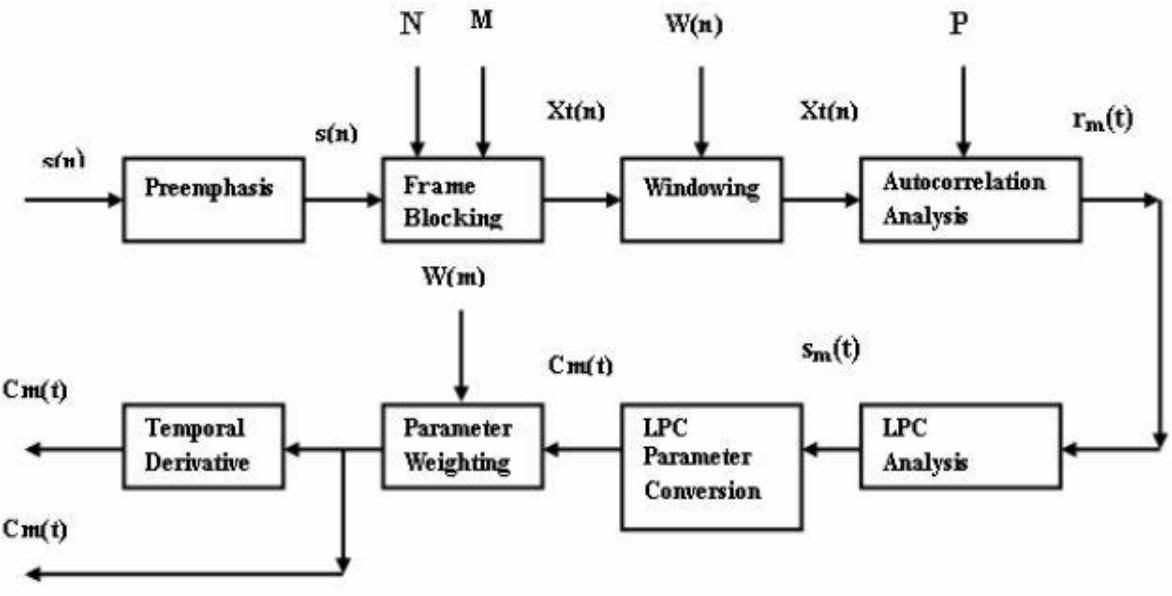
4 Postupci reprezentacije govornog signala u svrhu prepoznavanja govora ili govornika

Gledajući govorni signal općenito mogu se uočiti njegove velike varijacije tokom vremena. Međutim ako se govorni signal promatra u dovoljno kratkom vremenskom periodu (5-100 ms), njegove su karakteristike prilično stacionarne. Računanjem amplitudnog spektra u kratkim vremenskim intervalima, dobiva se osnova u daljnjem računanju značajki govornog signala. Razlog zbog kojeg se računa spektar u kratkom vremenskom periodu je saznanje da pužnica uha također provodi sličnu analizu. Analiza u pužnici se izvodi na nelinearnoj frekvencijskoj skali (skale poznate pod nazivom Bark skala ili Mel-frekvencijska skala). Skala je linearna do otprilike 1000 Hz, a nakon toga je otprilike logaritamska. Postupci reprezentacije govornog signala u svrhu automatskog prepoznavanja govora igraju ključnu ulogu u ukupnim performansama sustava. Mnoge tehnike računanja značajki govornog signala su dostupne za korištenje, a neke od njih su:

- LPC analiza (engl. *Linear Prediction Coding*)
- LPCC (engl. *Linear Predictive Cepstral Coefficients*)
- PLP analiza (engl. *Perceptual Linear Predictive*)
- Estimacija snage (engl. *Power Estimation*)
- Estimacija osnovne frekvencije (engl. *Pitch estimation*)
- MFCC analiza (engl. *Mel-Frequency Cepstral Coefficients*)
- RASTA (engl. *Relative spectra filtering of log domain coefficients*)
- DELTA (engl. *First order derivative*)

4.1 LPC analiza (Linear Prediction Coding)

LPC je jedna od najmoćnijih tehnika analize govora i vrlo je korisna metoda za kvalitetno kodiranje govora pri niskom *bit rate-u*. Glavna ideja kod LPC-a je aproksimacija trenutnog uzorka govora linearnom kombinacijom prošlih uzoraka govora.



Slika 2 Blok dijagram LPC analize

Svi iznosi prošlih uzoraka ne doprinose jednako iznosu trenutnog uzorka, pa se moraju prije sumacije pomnožiti odgovarajućim težinskim koeficijentima. Ti težinski koeficijenti su zapravo parametri ili koeficijenti linearnog prediktora. Cilj LPC nalize je minimizirati sumu kvadrata razlika između originalnog signala govora i estimiranog signala govora, tj. naći parametre prediktora kojima se ostvaruje najbolje predviđanje. Parametri prediktora ne mogu biti konstantni u duljem vremenskom intervalu, pa se oni zato računaju na kratkim vremenskim okvirima (engl. *frames*) na kojima se mogu smatrati stalnima. Okviri su obično duljine 20-30 ms. Koeficijenti linearnog prediktora su ujedno i koeficijenti IIR filtra koji modelira spektralnu ovojnicu govornog signala, odnosnu frekvencijsku karakteristiku vokalnog trakta. Takav filtar ima samo polove (all-pole IIR filtar), jednako kao i vremenski diskretni akustički model vokalnog trakta sa spojenim cijevima istih dužina V(z). Amplitudno-frekvencijska karakteristika IIR filtra dobivena LPC analizom aproksimira ovojnicu vremenski kratkotrajnog spektra signala za određeni okvir analize. Prijenosna funkcija linearnog prediktora reda p dana je izrazom:

$$P(z) = \sum_{k=1}^{p} a_k * z^{-k}$$

Parametri linearnog prediktora se mogu odrediti metodom autokorelacije, kovarijance ili metodom ljestvičaste strukture.

4.2 PLP analiza (Perceptual Linear Prediction)

PLP model je razvio Hermansky 1990. godine. Cilj originalnog PLP modela bio je točnije opisati psihofiziku ljudskog sluha prilikom procesa računanja značajki govornog signala. PLP analiza je slična LPC analizi. Temelji se na spektru kratkog odsječka govornog signala. Razlika između tih dviju analiza je u tome što PLP analiza modificira spektar kratkih odsječaka govora pomoću nekoliko transformacija zasnovanim na nekim psihofizičkim značajkama ljudskog sluha. Pri računanju PLP parametara koriste filtri razmješteni prema Bark skali. Postupak dobivanja PLP koeficijenata se sastoji od nekoliko koraka:

- nad otipkanim govornim signalom se izvodi DFT
- *critical-band* spektar snage (naziv zbog Bark skale) se računa kroz diskretnu konvoluciju snage spektra sa po dijelovima aproksimiranom *critical-band* krivuljom.
- primjena *pre-emphasis* filtra u svrhu ujednačavanja glasnoće
- kompresija temeljena na saznanju o nelinearnom odzivu uha pri različitim intenzitetima zvuka
- na dobiveni rezultat primjenjuje se inverzna DFT kako bi se dobila ekvivalentna autokorelacijska finkcija
- napokon, PLP koeficijenti se dobivaju nakon autoregresivnog modeliranja i pretvorbe autoregresivnih koeficijenata u kepstralne koeficijente.

4.3 Estimacija snage

Koristiti nekakvu vrstu mjerenja snage je vrlo uobičajeno danas. Snaga je relativno jednostavna za izračunati. Računa se na pojedinim vremenskim isječcima prema slijedećoj formuli:

$$P(n) = \frac{1}{N}\sum_{m=0}^{N-1}\left(w(m)s\left(n-\frac{N}{2}+m\right)\right)$$

Gdje je *N* broj uzoraka u kojima se računa snaga, *s(n)* signal, a *w(m)* funkcija vremenskog otvora. Trajanje vremenskog prozora kontrolira količinu usrednjavanja i glađenja u računanju snage. Veliko preklapanje rezultira smanjenjem šuma. Ipak, pretjerano glađenje može zasjeniti stvarne varijacije u govornom signalu. U praksi se umjesto obične snage

računa logaritam snage pomnožen sa 10 (snaga u decibelima) želeći se imitirati logaritamski odziv ljudskog auditornog sistema. Glavna značajka računanja snage je ta da predstavlja osnovu u izdvajanju zvučnih segmenata govora od bezvučnih. Vrijednosti snage bezvučnih segmenata govora su značajno manji nego kod zvučnih segmenata govora. Podatak o snazi se može iskoristiti za određivanje prelaska govornog signala iz zvučnog u bezvučni dio i obrnuto.

4.4 Estimacija osnovne frekvencije

Osnovna frekvencija ili *pitch* je definirana kao frekvencija oko koje glasnice osciliraju tijekom zvučnog govornog signala. Osnovna frekvencija je dugo bio parametar kojega je teško pouzdano ocijeniti. Postoji mnogo algoritama za procjenu *pitch-a*, a ovdje će biti spomenuta dva široko-korištena algoritma.

4.4.1 Gold- Rabiner algoritam

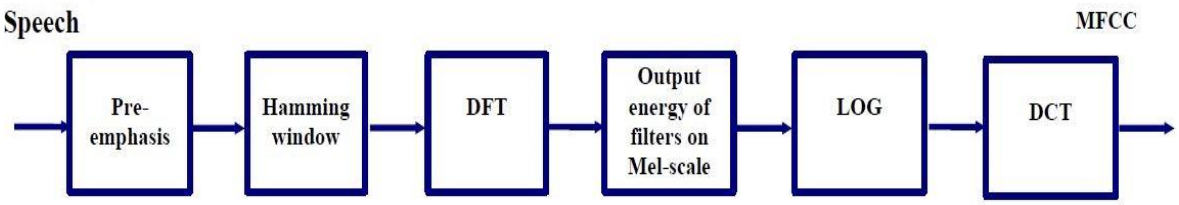
Jedan od najranijih i najjednostavnijih algoritama za estimaciju *pitch-a*. Radi na principu računanja više pomaknutih impulsa koji nastoje pratiti periodičnost govornog signala i tada koriste spomenute impulse u svrhu određivanja perioda *pitch-a*.

4.4.2 Određivanje *pitch-a* na temelju kepstra

U kepstru zvučnog govornog segmenta na indeksu koji odgovara osnovnom periodu titranja glasnica može se uočiti šiljak, dok u kepstru bezvučnog govornog segmenta taj šiljak ne postoji. Upravo ovo svojstvo kepstra može se koristiti kao osnova za određivanje zvučnosti govornog segmenta kai i za određivanje osnovnog perioda zvučnih govornih segmenata.

4.5 MFCC analiza

Blok dijagram provođenja MFCC analize dan je slikom 3.



Slika 3 Blok dijagram provođenja MFCC analize

U sljedećem tekstu biti će opisano zašto se koriste baš ovi blokovi tj. koja je logika iza ovako poredanih operacija.

Prvi blok u blokovskom dijagramu (engl. *pre-emphasis*) služi za pojačavanje komponenti visokih frekvencija tj. pojačavanje energije dijelova signala koji su viših frekvencija. Blok se može opisati sljedećom prijenosnom funkcijom:

$$P(z) = 1 - 0.97 * z^{-1}$$

Uzorci zvuka se formiraju u okvire uobičajenog trajanja 20-25 ms. Okviri se množe vremenskim otvorom. Obično se odabire Hammingov vremenski otvor prikazan sljedećom formulom:

$$w(n) = 0.54 - 0.46\cos(2\pi \frac{n}{N-1})$$

Kako bi se lakše razumjeli sljedeći postupci važno je razumjeti da se proizvodnja govora opisuje preko modela izvor-filtar. Izvor se povezuje sa zrakom koji se ispušta kroz pluća. Ako je zvuk bezvučan, kao kod glasova „s“ i „f“, glasnice ne osciliraju. Ako je zvuk zvučan, kao kod glasova „a“ i „e“ na primjer, glasnice vibriraju a frekvencija vibracija je oko *pitch-a*. Filtar daje oblik spektru signala proizvodeći tako različite zvukove. Vokalni trakt se opisuje spomenutim filtrom. Ugrubo govoreći, dobre analize reprezentacija govornog signala pokušavaju ukloniti utjecaj izvora (sustav mora dati isti odziv za visoki *pitch* ženskog glasa i za niski *pitch* muškog glasa), a ostvariti dobru karakterizaciju filtra. Problem je u tome što su izvor *e[n]* i impulsni odziv filtra *h[n]* međusobno konvoluirani. Potrebno je izvršiti dekonvoluciju u sustavima prepoznavanja govora.

U vremenskoj domeni:

$$e[n] * h[n] = x[n]$$

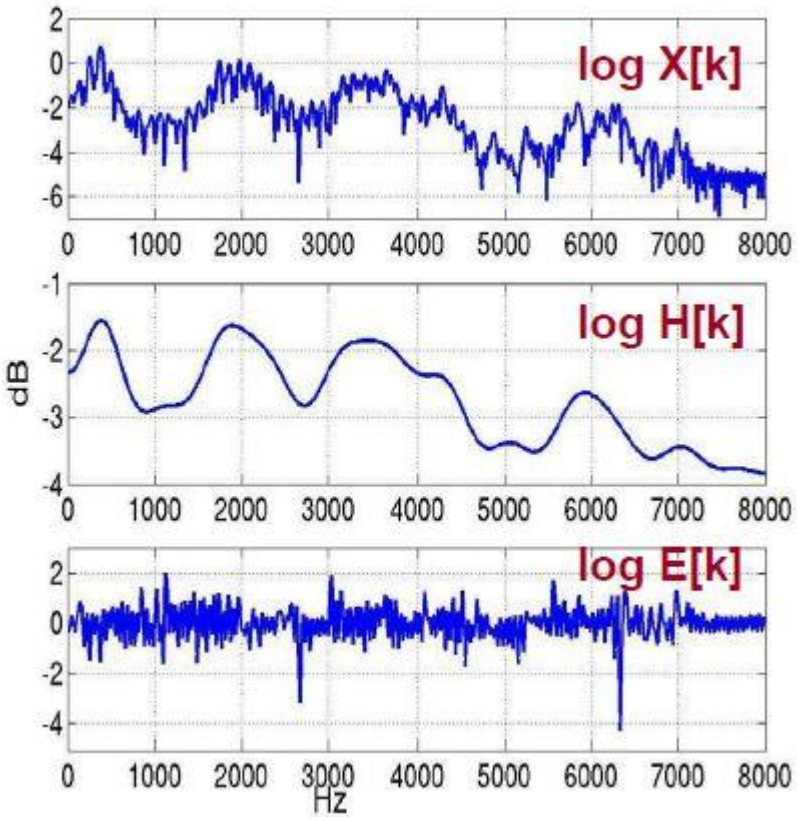
Gdje operator „*“ označava konvoluciju.

U frekvencijskoj domeni:

$$E[z]H[z] = X[z]$$

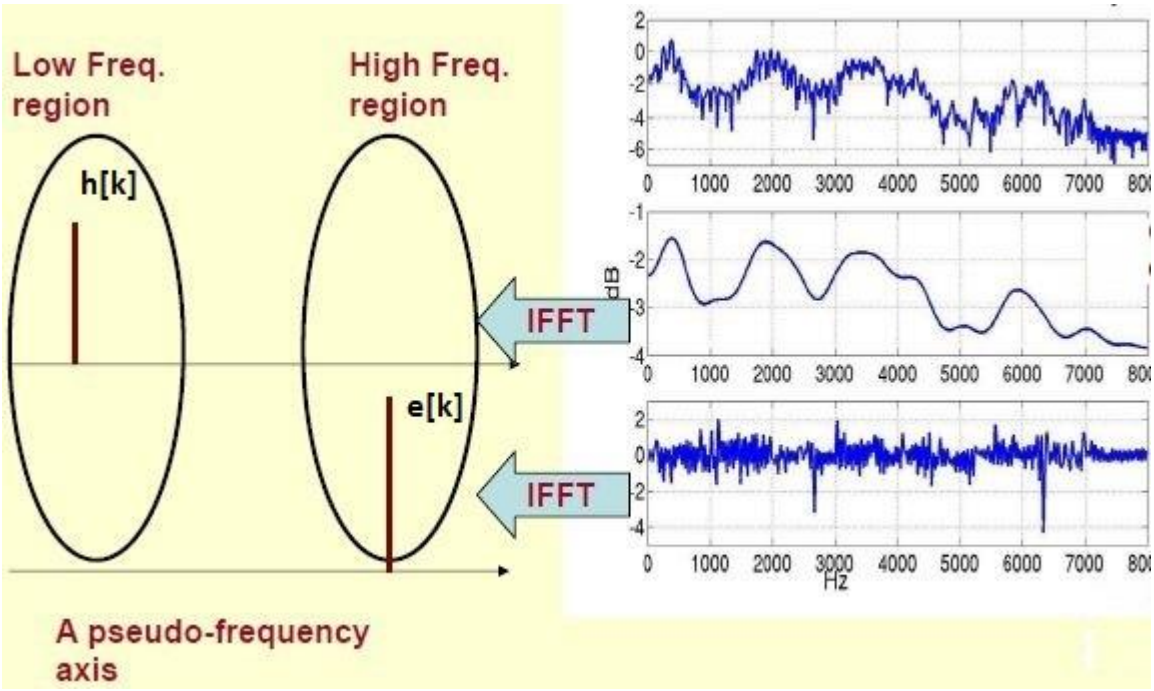
Javlja se problem. Kako izvesti dekonvoluciju? Valja se poslužiti matematičkim trikovima. Potrebno je u logaritmirati umnožak u frekvencijskoj domeni kako bi se on mogao rastaviti u zbroj.

$$\log X[k] = \log H[k] + \log E[k]$$



Slika 4 Izdvajanje ovojnice u logaritmu spektra

Nakon toga, potrebno je izvesti IFFT. Rezultat će biti prikazan na pseudo-frekvencijskoj osi (quefrency). Na slici 5 prikazana je pojednostavljena IFFT. Radi jednostavnosti, ovojnica spektra u logaritamskoj domeni može biti promatrana kao sinusni signal sa npr. 4 ciklusa u sekundi. Nakon što se nad tako promatranim signalom primijeni IFFT, dobiti će se šiljak na 4 Hz u pseudo-frekvencijskoj domeni. Razlika logaritma spektra i njegove ovojnice može se promatrati kao sinusni signal od npr. 100 ciklusa u sekundi koji se nakon primjene IFFT-a preslikava kao šiljak u 100 Hz u pseudo-frekvencijskoj domeni.



Slika 5 Postupak transformacije logaritma spektra u pseudo-frekvencijsku domenu

Kako je u praksi poznat samo spektar signala, a ne i njegova ovojnica posebno, ovakvom transformacijom dobiti će se zbroj transformacije ovojnice i transformacije ostalog dijela spektra.

$$x[k] = h[k] + e[k]$$

Kako je prikazano na prethodnoj slici, ta dva dijela (*h[k]* i *e[k])* su odmaknuta zbog razlike u pseudo-frekvencijama te se tražena ovojnica može izdvojiti nisko-propusnim filtrom. Oznaka *x[k]* predstavlja Kepstar signala (engl. *Cepstrum*).

$$X[k] = H[k] * E[k]$$

$$\|X[k]\| = \|H[k]\| * \|E[k]\|$$

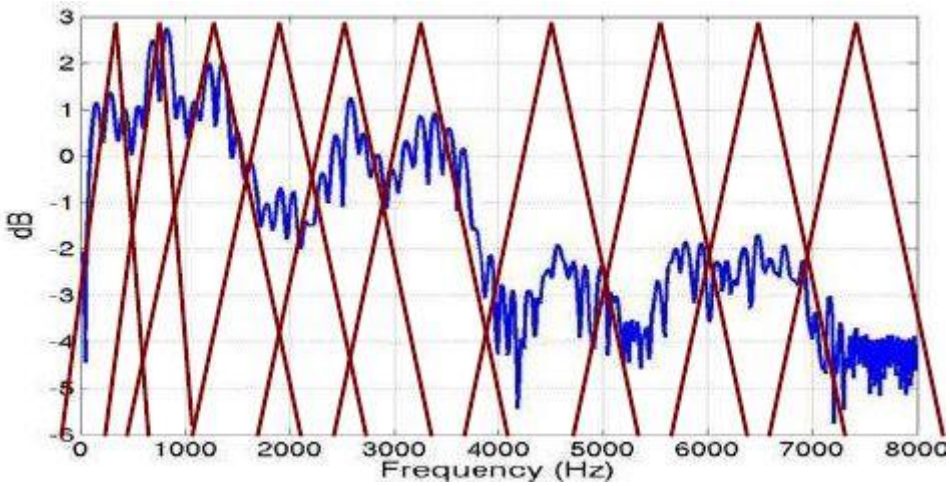
$$\log\|X[k]\| = \log\|H[k]\| + \log\|E[k]\|$$

IFFT..

$$x[k] = h[k] + e[k]$$

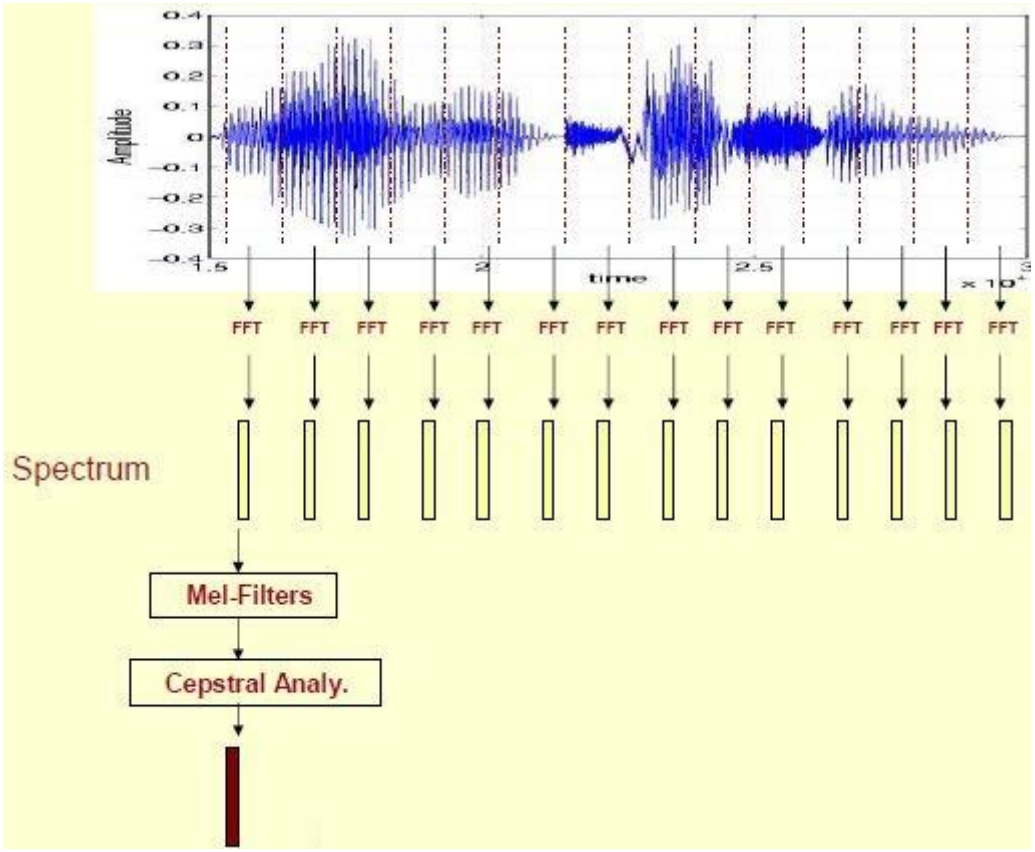
Opisani postupak naziva se kepstralna analiza signala.

Perceptualni eksperimenti su pokazali kako se ljudsko uho koncentrira samo na određene regije pri spektralnoj analizi zvuka pa nije potrebno poznavati cijelu spektralnu ovojnicu. Mel-Frequency analiza govora je zasnovana na perceptualnim experimentima ljudskog uha. Uočeno je da se ljudsko uho ponaša kao neka vrsta filtra tj. da se koncentrira samo na određene frekvencijske komponente. Takvi filtri su neuniformno rapoređeni po frekvencijskoj osi. Više filtara ima na nižim frekvencijama, a manje na višim frekvencijama. Na slici 6 prikazan je raspored takvih filtara.



Slika 6 Skup filtara raspoređenih prema Mel skali

Spektar koji je prošao kroz ovakve filtre naziva se Mel-Spektar. Ako se na ovakvom spektru provede opisana Kepstralna analiza, dobiveni kepstralni koeficijenti (*h[k]*) nazivaju se Mel-Frequency kepstralni koeficijenti (MFCC – Mel-Frequency Cepstral Coefficients). MFCC analiza predstavlja „state-of-art“ u sustavima automatskog prepoznavanja govora. Na slici 7 ukratko je prikazan postupak dobivanja MF kepstralnih koeficijenata.



Slika 7 Postupak dobivanja kepstralnih koeficijenata

U praksi se umjesto FFT-a nad logaritmiranim spektrom snage koristi DCT (Descrete Cosine Transform) zbog toga što DCT ima više energije smještene u manji broj koeficijenata u usporedbi sa DFT-om te se tako dobiva na kompresiji podataka. O MFCC analizi će se još govoriti u narednim poglavljima te će eventualne nejasnoće biti razjašnjene.

5 Implementacija prepoznavanja samoglasnika

U sklopu ovoga rada realizirane su implementacije prepoznavanja samoglasnika u programskom paketu Matlab, C programskom jeziku, a izvedeno je i prepoznavanje samoglasnika u realnom vremenu korištenjem DSP-a.

5.1 Implementacija u Matlabu

Sustav prepoznavanja samoglasnika u Matlabu uzima prethodno snimljene uzorke govora (samoglasnika) spremljenog u „wav“ formatu koje, nakon obrade, uspoređuje sa spremljenim modelima svih postojećih samoglasnika te na temelju Euklidove udaljenosti određuje o kojem se samoglasniku radi. Samoglasnici su snimljeni u programu Audacity frekvencijom otipkavanja 16 kHz u 16 bitnoj preciznosti. Prilikom pokretanja skripte „pokreni.m“ otvara se grafičko korisničko čija je namjena olakšati korištenje ovog sustava. U direktoriju gdje je smješten sam program, nalaze se snimke samoglasnika u „wav“ formatu. Snimke pod nazivima „a“, „e“, „i“, „o“ i „u“ služe kao referentne snimke tj. sa informacijama dobivenim iz tih snimaka će se uspoređivati informacije dobivene iz testnih snimaka. Testne snimke se nalaze u istom direktoriju pod nazivima „1“, „2“, „3“, „4“ i „5“ (u „wav“ formatu). Prilikom pokretanja programa, pozivima funkcije „wavread“, učitavaju se snimke referentnih samoglasnika. Svaka snimka je duljine jedne sekunde no u matlab okolinu se sprema samo pola sekunde (8000 uzoraka) za daljnju obradu. Za svaku referentnu snimku samoglasnika poziva se „mfcc_funkcija“ funkcija. Prototip funkcije izgleda ovako:

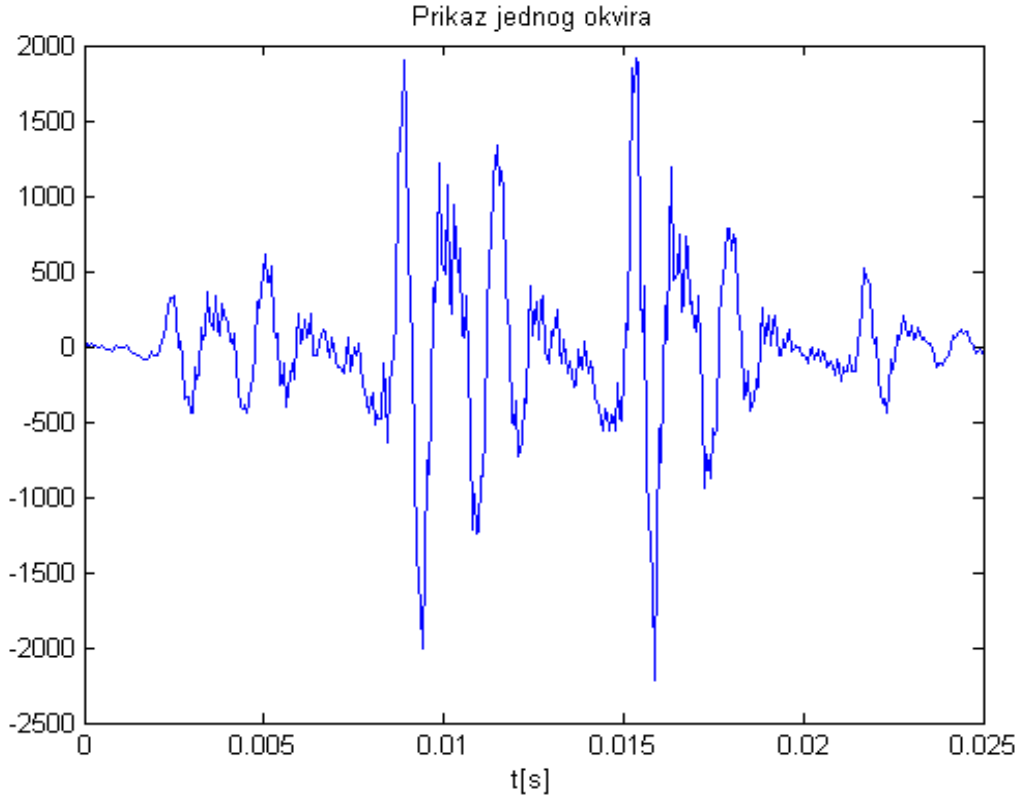
```
[ CC, FBE, frames ] = mfcc_funkcija( speech, fs, Tw, Ts, alpha, window, M, N, L ).
```

Gdje su:

- speech – ulazni signal (govor)
- fs – frekvencija uzorkovanja
- Tw – duljina trajanja vremenskog okvira (u sekundama)
- Ts – duljina vremenskog odmaka okvira (u sekundama)

- alpha – koeficijent filtra prvog reda (pre-emphasis filtar)
- window – handle za vrstu korištenog vremenskog otvora
- M – broj Mel filtara
- N – broj kepstralnih koeficijenata
- L – parametar za lifter (filtar)
- CC – kepstralni koeficijenti
- FBE – spektar snage pomnožen Mel filtrima
- frames – govorni signal spremljen po okvirima.

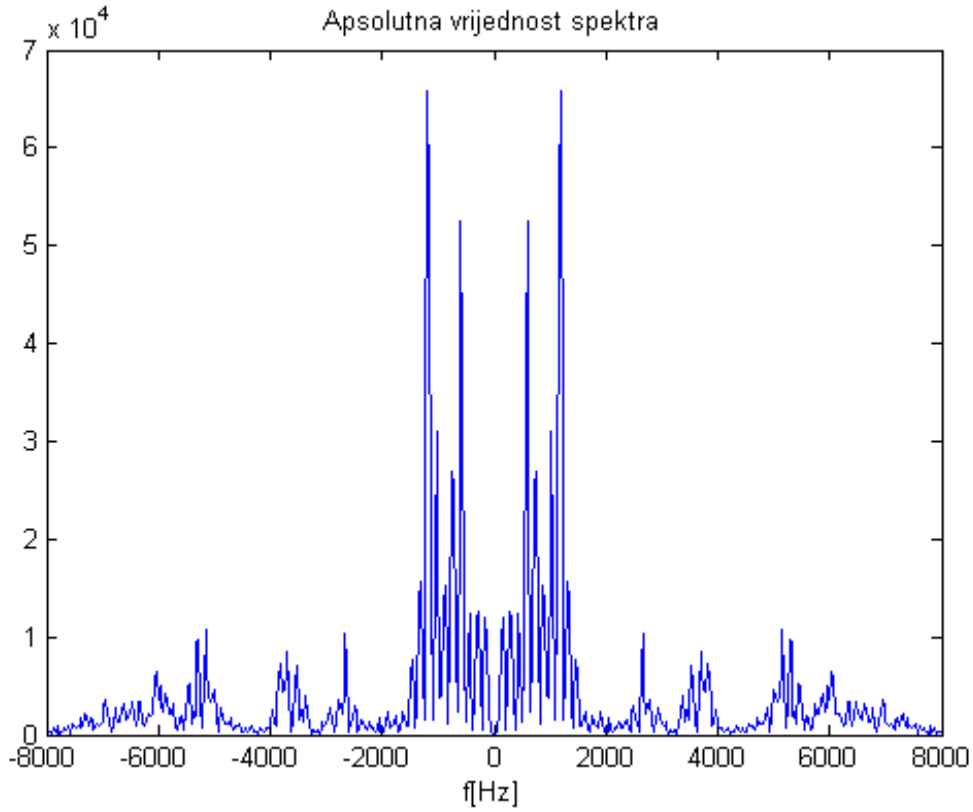
Prilikom ulaska u funkciju zvuk se provlači kroz filtar prvog reda koji pojačava energiju komponenata viših frekvencija. Uzorci govora se zatim formiraju u okvire. Odabrana duljina okvira je 25 ms što znači da će u jednom okviru biti 400 uzoraka. Odmak svakog okvira od početnog indeksa prošlog okvira je 10 ms (160 uzoraka). Tako se dobiju 48 okvira od po 400 uzoraka govora. Svaki okvir množi se s Hammingovim vremenskim otvorom. Na slici 8 prikazan je jedan okvir (samoglasnik „a“) nakon prolaska kroz Hammingov vremenski otvor.



Slika 8 Jedan okvir govornog signala (samoglasnik "a")

Nad svakim okvirom se izvodi FFT u 512 točaka (uzorci od 401 do 512 se nadomještaju nulama). Nakon što se izračuna apsolutna vrijednost spektra (slika 9), potrebno je

izračunati snagu spektra. To se radi tako da se svaki uzorak apsolutne vrijednosti spektra pomnoži sa samim sobom. Kako bi ubrzali algoritam rješavamo se nepotrebnih informacija uzimajući samo jedinstven dio spektra tj. 257 uzoraka.



Slika 9 Apsolutna vrijednost spektra jednog okvira

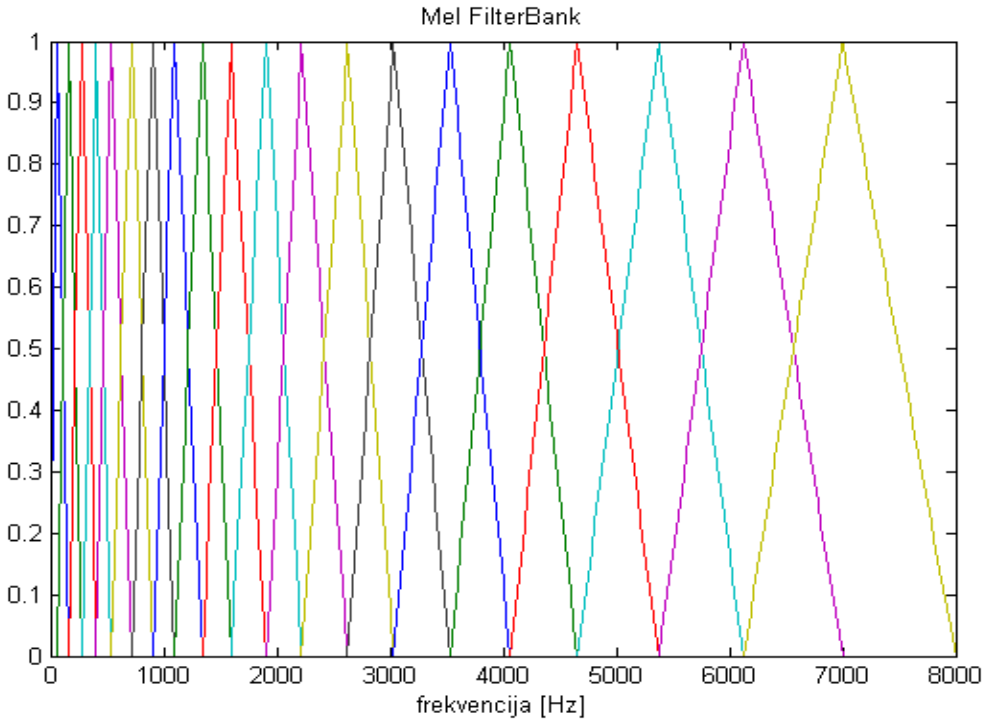
Sada je još potrebno prilagoditi snagu spektra ljudskom uhu. Kao što je već rečeno, ljudsko uho se koncentrira na određene dijelove u spektru, a ne na cijeli spektar. To će se izvršiti propuštanjem snage spektra kroz niz filtara trokutastog oblika koji su raspoređeni prema Mel-frekvencijskoj skali. Kako su filtri trokutasti, svaki filtar ima tri karakteristične točke: početna točka, točka maksimuma i krajnja točka. Početna točka (frekvencijska os) svakog filtra (ne uključujući prvi) je ujedno i točka maksimuma prethodnog filtra, a krajnja točka svakog filtra (ne uključujući posljednji) je ujedno i točka maksimuma sljedećeg filtra. Relacije između frekvencijske skale i Mel skale dane su sljedećim izrazima:

$$F_{mel} = 2595 \log_{10}\left(1 + \frac{F_{Hz}}{700}\right)$$

$$F_{Hz} = 700\left(10^{\frac{F_{mel}}{2595}} - 1\right).$$

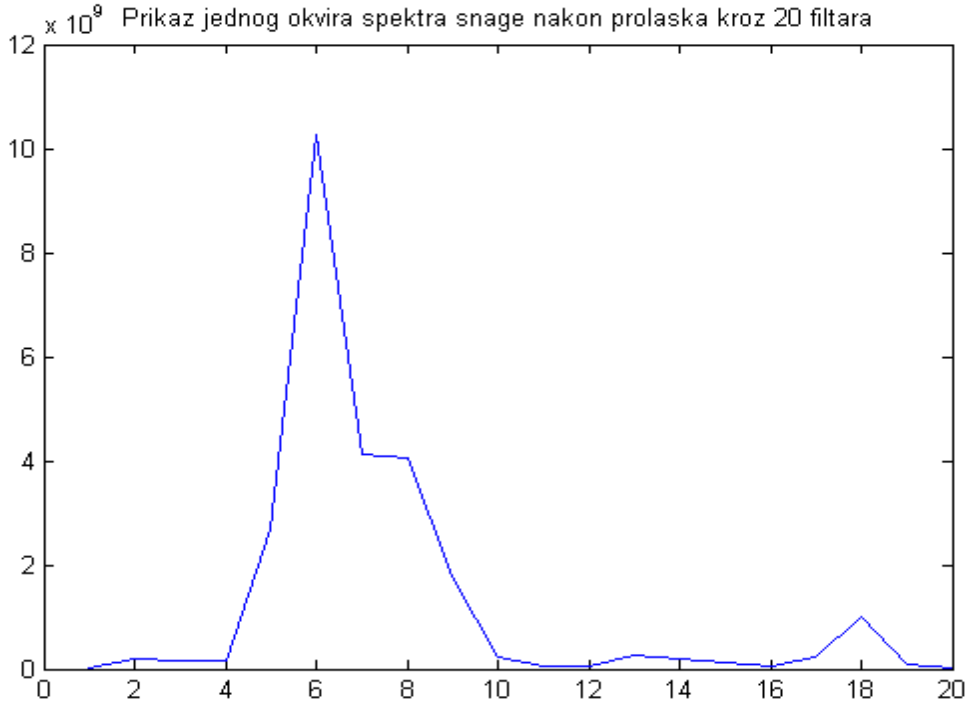
Pozivom funkcije „MelFrequencyFilterBank.m“ formira se skup filtara koji su linearno razmješteni na Mel-frekvencijskoj skali, a logaritamski na frekvencijskoj skali (više ih ima na nižim frekvencijama, a manje na nižim). Kao ulazne parametre, funkcija prima

frekvenciju uzorkovanja, broj točaka FFT-a i željeni broj filtara. Generirani filtri prikazani na slici 10.



Slika 10 Generirani skup filtara prema Mel skali

Jedan okvir, nakon prolaska kroz filtre, prikazan je na slici 11.



Slika 11 Okvir spektra snage nakon prolaska kroz 20 filtara

Ono što je sada potrebno napraviti jest izračunati logaritam prainačenog spektra snage (Mel spektar snage) te izračunati diskretnu kosinusnu transformaciju nad svakim od 48 okvira. DCT izražava konačnu sekvencu podataka preko sume kosinusnih funkcija koje osciliraju na različitim frekvencijama. DCT je zapravo slična DFT-u ali koristi samo realne brojeve. Postoji 8 inačica standardnog DCT-a. Najčešće korištena inačica je diskretna kosinusna transformacija tipa II. Često se jednostavnije naziva DCT i dana je sljedećim izrazom:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] k = 0, \dots, N-1.$$

Za potrebe ovog algoritma korištena je treća inačica diskretne kosinusne transformacije koja se često naziva i inverzna DCT (IDCT). DCT-III dana je sljedećim izrazom:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right] k = 0, \dots, N-1.$$

Često se x_0 još dodatno skalira faktorom $\sqrt{\frac{2}{N}}$.

Kao što je rečeno, potrebno je izračunati IDCT logaritma Mel spektra snage signala za svaki okvir (sada su okviri po 257 uzoraka). Ovim postupkom će se dobiti željena informacija sadržana u ovojnici spektra, a odbaciti će se komponente više frekvencije koje ne nose značajnu informaciju. Time se provela dekonvolucija izvora i filtra koje je objašnjena ranije. Računajući IDCT zadržati će se samo 13 (12+1) koeficijenata. Prvi koeficijent (nulti) dobiven iz diskretne kosinusne transformacije je zapravo suma svih logaritama Mel spektra snage dobivenih u prethodnom koraku. Iz tog razloga je prethodno u zagradi napisano „12+1“. Uzimajući u obzir da govorni signal nije konstantan moguće je izvući i više od navedenih značajki MFCC analizom. Estimiranjem delta značajki dobiva se dodatnih 13 koeficijenata koji govore o promjenjivosti MFC koeficijenata između okvira.

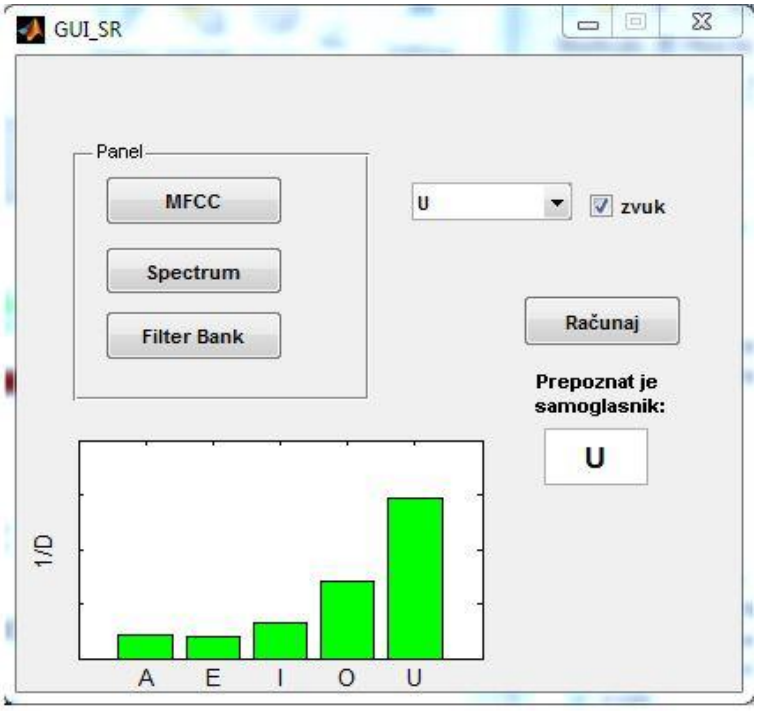
$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

Daljnjom analizom nad dobivenim delta koeficijentima moguće je dobiti informaciju o promjenjivosti delta koeficijenata. Tako se dobiva dodatnih 13 koeficijenata koji se još nazivaju i delta-delta koeficijenti (engl. *double delta coefficients*). Eksperimentalnim postupcima je pokazano kako nema potrebe za računanjem većeg broja koeficijenata jer se učinkovitost značajno ne povećava računanjem istih, a računanje većeg broja koeficijenata značajno utječe na brzinu algoritma što je vrlo važno u sustavima za rad u realnom

vremenu. Kada bi se koristila IFFT umjesto IDCT-a nebi se postigla ista učinkovitost reprezentacije značajki signala sadržana u tako malom broju koeficijenata.

Time je izvršen postupak reprezentacije govornog signala (samoglasnika) MFCC analizom. Krajnji rezultat MFCC analize prikazan je s 13 podataka (brojeva, koeficijenata) za svaki od 48 okvira govornog signala.

Cijela ova analiza će se prilikom pokretanja sustava izvršiti 5 puta, tj. za svaki referentni signal samoglasnika. Korisnik, preko grafičkog korisničkog sučelja, može odabrati testni govorni signal koji bi sustav trebao prepoznati. Grafičko korisničko sučelje prikazano je na slici 12.

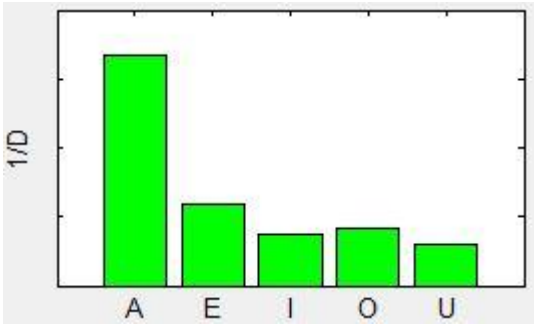


Slika 12 Grafičko korisničko sučelje

Prilikom odabira testnog uzorka u padajućem izborniku, ponovno se poziva funkcija za MFCC analizu. Ovog puta, signal za MFCC analizu jest testni signal, a po završetku izvođenja funkcije „mfcc_funkcija“ stvara se matrica značajki testnog signala. Pritiskom na tipku „Računaj“, pokreće se programski odsječak odgovoran za prepoznavanje samoglasnika. Ovaj sustav koristi najjednostavniju i najintuitivniju metodu prepoznavanja govora, metodu najbližih susjeda. Metoda najbližih susjeda je zapravo Euklidova udaljenost dana izrazom:

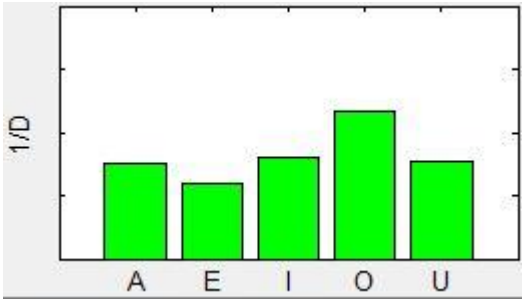
$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

Euklidova udaljenost računa se za testni signal sa svim referentnim signalima. Kao rezultat dobiva se vektor od 5 elemenata koji su raspoređeni po indeksima koji odgovaraju poretku samoglasnika (npr. samoglasnik „a“ je na indeksu 1, samoglasnik „e“ na indeksu 2 itd.). Indeks na kojem se nalazi najmanji broj dobiven Euklidovom udaljenosti označavat će najvjerojatniji samoglasnik kojeg testni signal predstavlja. Kao nekakva mjera kvalitete sustava prepoznavanja samoglasnika na zaslonu će se grafičkim prikazom dati informacija o izračunatim Euklidovim udaljenostima. Na slici 13 prikazan je primjer kada je prepoznat samoglasnik „a“ (napomena: Euklidove udaljenosti prikazane u recipročnom obliku).



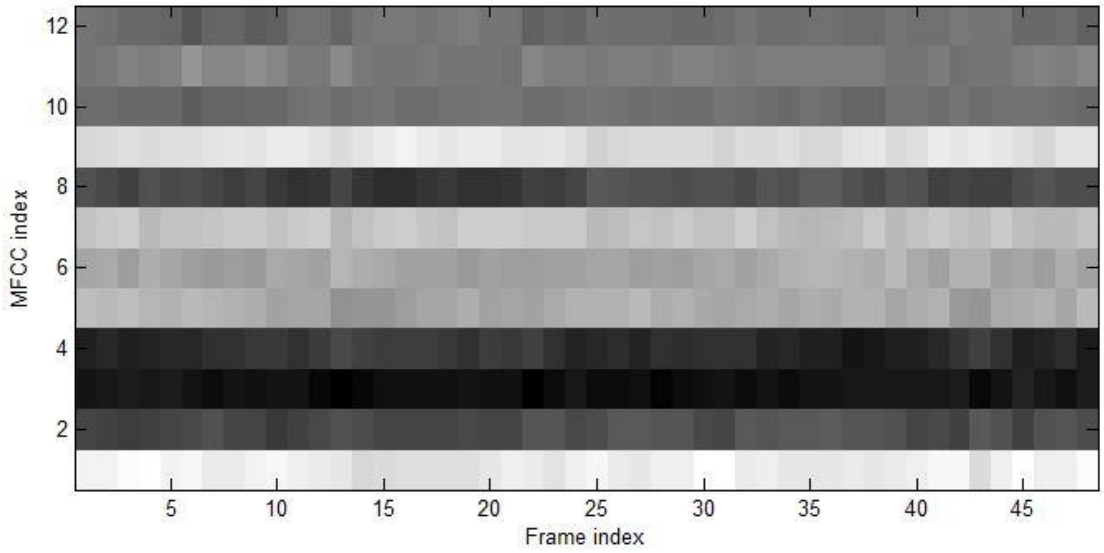
Slika 13 Prepoznat samoglasnik "a"

Najgori rezultati dobiveni su kod prepoznavanja samoglasnika „o“. Primjer je prikazan na slici 14.



Slika 14 Prepoznat samoglasnik "o"

Valja napomenuti kako su prikazani rezultati dobiveni na temelju testnih i referentnih podataka istog govornika. Pritiskom na tipku „MFCC“ iscrtavaju se MFC koeficijenti po okvirima. Primjer za samoglasnik „a“ dan je na slici 15.



Slika 15 MFC koeficijenti

Na x-osi nalaze se indeksi okvira što se može povezati sa vremenom. Na y-osi nalaze se indeksi MFC koeficijenata. Nijansama sive boje prikazane su vrijednosti MFC koeficijenata. Tamnija nijansa odgovara manjoj vrijednosti (negativnije) a svjetlija većoj (pozitivnije). Za potrebe implementacije u C-u generiraju se datoteke testnih i referentnih signala koje će biti spomenute u sljedećem poglavlju.

5.2 Implementacija u C-u

5.2.1 Frakcionalna aritmetika

U C programskom jeziku implementirana je funkcionalnost sustava izvedenog u Matlabu. Način implementacije u C-u biti će osnova implementacije na DSP-u koja će biti objašnjena u sljedećem poglavlju. Korišteni procesor je 16-bitni i za rad s podacima koristi cjelobrojnu aritmetiku. Za zapis podataka koristi se 16 bitova: bit predznaka i 15 bitova podataka (mogu se prikazati cijeli brojevi iz intervala $[-2^{15}, 2^{15}-1]$) ili 16 bitova podataka (prikazuje cijele brojeve iz intervala $[0, 2^{16}-1]$).

Ako je potrebno prikazati frakcionalne podatke, zbog nepostojanja jedinice za operacije s pomičkim zarezom (FPU), rješenje je zapis podataka korištenjem Q formata. Q format koristi 16 bitova kao i cjelobrojni tip podataka, ali je razlika u njihovoj interpretaciji. Jedan bit se koristi za predznak, dok je preostalih 15 bitova podijeljeno na cijeli i frakcionalni dio. Npr., ako je broj spremljen u Q2.13 formatu, dva se bita koriste za cijeli, a 13 za frakcionalni dio (jedan bit za predznak). To omogućuje prikaz brojeva iz intervala $[-4, 4-2^{-13}]$ uz preciznost 2^{-13}. Za potrebe implementacije u C-u i na DSP-u koristi se format Q15 (svih 15 bitova koristi se za frakcionalni dio) što efektivno skalira

podatke na interval [-1,1>. Operacije zbrajanja, oduzimanja i posmaka izvode se na isti način kao i kod rada s cjelobrojnim tipovima podataka, ali to nije slučaj kod množenja. Množenjem dva broja spremljena u formatu Q15 dobiva se rezultat u Q30 formatu (preostala dva bita su bitovi predznaka). Taj rezultat se očito ne može spremiti u 16 bitova. Rezultat takvog množenja potrebno je privremeno spremiti kao tip podatka većeg kapaciteta, te ga zatim posmaknuti za 15 bitova udesno i spremiti u tip podataka manjeg kapaciteta dostatan za čuvanje podataka u Q15 formatu. Kod operacija s podacima u Q15 formatu treba paziti da ne dođe do preljeva (engl. *overflow*) ili podljeva (engl. *underflow*). Kako su podaci skalirani na interval [-1,1>, rezultat množenja će uvijek biti u tom istom intervalu što znači da do preljeva ne može doći pri operacijama množenja. Problem nastaje kod operacija zbrajanja i oduzimanja jer rezultat nastao zbrajanjem dva broja iz intervala [-1,1> može biti izvan tog istog intervala. Kako bi se doskočilo problemu preljeva, moguće je rezultat postaviti u zasićenje (ovisno da li se radi o preljevu ili podljevu). Problem kod ovakvog načina suočavanja sa preljevima je mogućnost akomuliranja značajne pogreške u preciznosti rezultata ukoliko se preljevi događaju učestalo. Druga mogućnost je skaliranje podataka prije obrade. Faktor kojim će se podaci skalirati određuje se eksperimentalno. Jedan od kritičnih dijelova koda kod kojeg bi moglo doći do preljeva jest algoritam za računanje FFT-a. Tijekom računanja FFT-a potrebno je izvršiti velik broj zbrajanja što gotovo sigurno rezultira preljevom. U trenutku kada nastupi preljev za točan će rezultat biti potreban dodatni bit koji jednostavno nije dostupan za ovakav format zapisa u memoriju. Ako se svaki uzorak signala nad kojim se računa FFT umanji za pola, do preljeva neće doći. Kao rezultat poslije računanja FFT-a dobivaju se podaci skalirani sa 1/N, gdje N označava broj točaka u kojima se računa FFT. U ovom primjeru računa se FFT u 512 točaka tako da će podaci biti skalirani sa 2^9 u odnosu na rezultate dobivene FFT-om u Matlabu. Tijekom izvođenja koda pokušava se iskoristiti maksimalni raspon zapisa u 16 bitova tako da se podaci skaliraju prema potrebi. Ako bi se krajnji rezultati implementacije u C-u vratili u stvarne vrijednosti, dijeljenjem sa 2^{15}, dobile bi se vrijednosti koje su 2^4 puta manje od krajnih rezultata u Matlabu.

5.2.2 Implementacija u C-u

Implementacija u C-u rađena je po uzoru na implementaciju u Matlabu. Program napisan u C-u koristi tekstualne datoteke testnig i referentnih podataka generirane kodom u Matlabu. Testni signali su spremljeni u tekstualne datoteke pod nazivima „1“, „2“, „3“, „4“ i „5“. Podaci u datoteci „1.txt“ odgovaraju uzorcima testnog signala samoglasnika „A“, u

datoteci „2.txt“ odgovaraju uzorcima testnog signala samoglasnika „E“ itd. Datoteke sa testnim signalima sadrže 8000 uzoraka što odgovara duljini govornog signala u trajanju od pola sekunde. U datotekama „cepA_sr.txt“, „cepE_sr.txt“, „cepI_sr.txt“, „cepO_sr.txt“, „cepU_sr.txt“ nalaze se referentni MFC koeficijenti nastali usrednjavanjem skupa vektora MFC koeficijenata iz Matlaba. 8000 uzoraka testnog signala skaliraju se na interval od [-1,1> te se zatim množe sa 2^{15} kako bi se dobio željeni Q15 zapis. Signal se tada provodi kroz visoko-propusni filtar implementiran danim kodnim odsječkom.

```
tmp_old=0;
  for(i=0; i<8000;i++){
        tmp_new=fxd_novi_desni[i];
        tmp2=(long int)(tmp_old*fxd_alpha);
        tmp1=(long int)tmp_new<<15;
        fxd_novi_desni[i]=(short)((tmp1-tmp2)>>15);
        tmp_old=tmp_new;
        }
```

Polje „fxd_novi_desni“ je tipa *short* i u njemu se nalaze uzorci testnog signala. Prema formuli koja opisuje filtar, uzima se prethodni uzorak signala pomnožen sa faktorom „alpha“ koji u ovom primjeru iznosi 0.97. Taj se oduzima od trenutnog uzorka govornog signala. Ovaj postupak pojačava dijelove viših frekvencija u govornom signalu koji su tiši od komponenata niske frekvencije. U kodnom odsječku se vidi kako se umnožak dva broja tipa *short* privremeno sprema u tip podatka veće duljine *long int* te se zatim rezultat ponovno vraća u polje podataka tipa *short* izvodeći adekvatan posmak bitova. Nakon toga signal se raspoređuje u 48 okvira kako je to opisano kod implementacije u Matlabu. Okviri se množe Hammingovim filtrom kako je to prikazano u sljedećem kodnom odsječku.

```
//Hammingov prozor
for (i=0 ; i < wlen ; i++){
                    win=ham1-ham2*cos((double)(2*pi*(i)/(double)(wlen-1)));
                    fxd_win[i]=(short)(win*(1<<15));
                   }
//Množenje s Hammingovim prozorom
//S=47
for (i = 0 ; i <=S ; i++){
                   for (j = 0 ; j < wlen ; j++){
                                       tmp1=(long int)(fxd_za_obraditi_desni[i][j]*fxd_win[j]);
                                       fxd_za_obraditi_desni[i][j]=(short)(tmp1>>15);
                                       }
                   }
```

Izrada Hammingovog prozora radi se prije obrade nad signalom ali ovdje je radi jasnijeg prikaza dano u istom odsječku. To za ovu implementaciju i nije bitno ali kod

implementacije u realnom vremenu (na DSP-u) vrlo je važno da se prije obrade signala izračuna sve što se može koristiti kao fiksni podatak tijekom obrade. Slijedi izračunanti spektar za svaki od 48 odsječaka govornog signala. Koristi se funkcija za izvođenje FFT algoritma u frakcionalnoj aritmetici (Q15 format). Prototip funkcije koja računa *fixed point* FFT izgleda ovako:

```
int fix_fft(short fr[], short fi[], int m, int inverse)
```

Funkcija prima pokazivač na prvi element realnog i imaginarnog dijela ulaznog signala te parametre *m* i *inverse*. Parametar *m* je povezan sa brojem točaka u kojima se računa FFT. Ovdje se želi izračunati FFT u 512 točaka, a kako je 2^9=512, *m* će označavati potenciju broja 2 tj. 9. Parametar *inverse* će reći funkciji želi li se izračunati FFT ili inverzna FFT (0 za FFT i 1 za IFFT). Kako je ranije objašnjeno, radi spriječavanja pojave preljeva, ulazni uzorci se dijele sa 2. Rezultat toga je skalirana vrijednost izlaza (tj. spektra) sa faktorom 1/N. Nakon izračuna spektra signala, uzima se samo jedinstveni dio spektra (prvih 257 realnih i imaginarnih uzoraka) te se izračunava spekar snage. Spektar snage za svaki prozor se sada provlači kroz skup od 20 trokutastih filtara raspoređenih prema Mel-frekvencijskoj skali. Funkcija za izračunavanje skupa filtara dana je u slijedećem kodnom odsječku.

```
for(i=0;i<=(nfilt+1);i++){edgemelfs[i]=maxmelf*i/(nfilt+1); //kar.  točke filtara
                          edgefrqs[i]=700.*(pow(10.,(edgemelfs[i]/ 2595.0)) - 1.);
                          edgeDFTbins[i]=(int)(round(edgefrqs[i]/maxf*(nfft/2+1))); }
edgeDFTbins[0] = 1;
for(i=0;i<nfilt;i++){l = edgeDFTbins[i];          // indeks početne točke filtra
                     c = edgeDFTbins[i+1];       // indeks središnje točka filtra
                     h = edgeDFTbins[i+2];       // indeks završne točke filtra
                     NbinsUpSlope = c-l;         // broj točaka prvog dijela filtra
                     NbinsDownSlope = h-c;       // broj točaka drugog dijela filtra
                     j=l;
                     for(z=0;z<=NbinsUpSlope;z++){
                          if(NbinsUpSlope!=0){ pom=z/(double)(NbinsUpSlope);
                                               fxd_MelFBank[i][j-1] =(short)( pom*32768);
                                               if(pom>=1)
                                                        fxd_MelFBank[i][j-1]=32767;
                                               }
                                               j++;}
                      j=c;
                      for(z=NbinsDownSlope;z>=0;z--){pom=z/(double)(NbinsDownSlope);
                                                fxd_MelFBank[i][j-1] =(short)(pom*32768);
                                                if(pom>=1)
                                                       fxd_MelFBank[i][j-1]=32767;
                                                j++; }
                     }
```

Varijabla *maxmelf* predstavlja najveću frekvenciju na Mel-frekvencijskoj skali (Fs/2 na frekvencijskoj skali). Prvo se računaju karakteristične točke filtara. Pod karakterističnim točkama filtra smatraju se početna točka, središnja točka i krajnja točka. Karakteristične točke se raspoređuju linearno po Mel-frekvencijskoj skali te se nakon toga vraćaju u frekvencijsku skalu. Nakon toga preostaje rasporediti točke na rastućem i padajućem nagibu za svaki filtar na frekvencijskoj skali. Skup filtara je naravno izračunat prije obrade a kako izgleda može se vidjeti na slici 10. Kao rezultat prolaska spektra snage kroz izračunati skup filtara dobiveno je 20 brojeva za svaki vremenski odsječak signala tj. matrica 20x48. Slijedi izračunati logaritam Mel spektra snage. Kako se koristi frakcionalna aritmetika potrebno je napraviti funkciju za računanje logaritma koja će raditi sa *fixed point* podacima. Gotova funkcija za računanje logaritma iz biblioteke radi sa floating point brojevima i takav način računanja bio bi dosta sporiji od korištenog načina (bitno kod izvođenja u stvarnom vremenu kod DSP-a). Funkcija *fxlog* aproksimira logaritam ulaznog podatka bez korištenja množenja. Zapravo se koristi posmak bitova što tehnički odgovara množenju/dijeljenju sa 2 no DSP takve operacije obavlja značajno brže od običnog množenja/dijeljenja. Princip rada funkcije *fxlog* može se opisati na primjeru. Recimo da se želi izračunati $\ln(x)$ gdje je *x* npr. jednak 54. Izraz se može zapisati i u obliku:

$$\ln\frac{54}{x}$$

koji bi bio istinit kada bi *x* težio u 1. Ako je x pomnoži sa nekim brojem *k*, da bi izraz bio istinit, nova vrijednost *y*, *y'* bi trebala zadovoljiti slijedeće:

$$y' = \ln\left(\frac{54}{kx}\right)$$
$$= \ln\left(\frac{54}{x}\right) + \ln\left(\frac{1}{k}\right)$$
$$= y - \ln k$$

Drugim riječima, ako se *x* pomnoži sa *k*, treba se $\ln k$ oduzeti od *y*. Za vrijednost *k* potrebno je uzeti neki „lijepi“ broj kako bi se lako množilo s njim (3/2,2,4,16 itd.). Kako su svi „lijepi“ brojevi za *k* veći od 1, potrebno je početi sa *x*-om koji je manji od 1 (želi se *kx* dovesti do 1). Npr. ako se *x* pomnoži s 1/256, potrebno je od y oduzeti $\ln\left(\frac{1}{256}\right) = 5.5452$. Ovo je tek početni korak od kojega je potrebno iterativnim putem dovesti *kx* što bliže 1. Da bude jasnije, trenutno stanje prikazano je u tablici.

x	y
54	0
54/256=0.2109	5.5452

Tabela 1 Računanje logaritma (početno stanje)

U prvom koraku gleda se najveći *k* kojim se može pomnožiti *x*, a da bi rezultat množenja ostao manji od 1. To je broj 4, a od y je potrebno oduzeti $\ln(4) = 1.3863$. Postupak se ponavlja a već nakon 4 koraka dobiveno stanje dano je u tablici.

x	Y
0.2109	5.5452
0.8436	4.1589
0.9491	4.0411
0.9788	4.0103
0.9788*65/64=0.9941	4.0103-0.0155=3.9948

Tabela 2 Računanje logaritma (nakon 4. koraka)

Stvarna vrijednost prirodnog logaritma od 54 jest 3.9890, a ovdje je već nakon 4 koraka dobivena aproksimirana vrijednost jednaka 3.9948. Apsolutna pogreška je $\ln 0.9941$. Za mali x vrijedi da je $\ln(1-x) \approx -x$ tako da je apsolutna pogreška u ovom slučaju približno 1-0.9941=0.0059. Ako se ova vrijednost oduzme od aproksimirane vrijednosti dobije se 3.9889 što je prilično dobra aproksimacija. Sada kada je pobliže objašnjen princip rada funkcije *fxlog*, ista je dana u slijedećem kodnom odsječku.

```
int fxlog(int x) {
 int t,y;
 y=0xa65af;
 if(x<0x00008000) x<<=16,          y-=0xb1721;
 if(x<0x00800000) x<<= 8,          y-=0x58b91;
 if(x<0x08000000) x<<= 4,          y-=0x2c5c8;
 if(x<0x20000000) x<<= 2,          y-=0x162e4;
 if(x<0x40000000) x<<= 1,          y-=0x0b172;
 t=x+(x>>1); if((t&0x80000000)==0) x=t,y-=0x067cd;
 t=x+(x>>2); if((t&0x80000000)==0) x=t,y-=0x03920;
 t=x+(x>>3); if((t&0x80000000)==0) x=t,y-=0x01e27;
 t=x+(x>>4); if((t&0x80000000)==0) x=t,y-=0x00f85;
 t=x+(x>>5); if((t&0x80000000)==0) x=t,y-=0x007e1;
 t=x+(x>>6); if((t&0x80000000)==0) x=t,y-=0x003f8;
 t=x+(x>>7); if((t&0x80000000)==0) x=t,y-=0x001fe;
 x=0x80000000-x;
 y-=x>>15;
 return y;
 }
```

Bitno je napomenuti da ova funkcija koristi Q16 format tako da je ulazne uzorke potrebno prilagoditi tom formatu kako bi se dobili ispravni rezultati. Kao što je već spomenuto kod implementacije u Matlabu, sljedeći je korak provesti inverznu diskretnu kosinusnu transformaciju. Matričnim množenjem matrice logaritama Mel spektra snage i matrice formirane kodom danim u nastavku, dobivaju se traženi MFC koeficijenti.

```
for(k=1;k<=nfilt;k++){
                    for(i=0;i<ncep;i++){
                                temp[i][k-1]=sqrt(2.0/20.0)*cos(i*pi*(k-0.5)/20.0);
                                fxd_temp[i][k-1]=(short)(temp[i][k-1]*(1<<15));
                                printf("%d ",fxd_temp[i][k-1]);
                                }
                    }
```

Kao rezultat spomenutog matričnog množenje dobiva se matrica 13x48 tj. dobiveno je 13 koeficijenata za svaki od 48 vremenskih odsječaka govornog signala. Kako bi se koeficijenti mogli usporediti sa referentnim MFCC vektorom, računa se aritmetička sredina po dobivenim koeficijentima između svih vremenskih odsječaka. Kako pritom nebi došlo do preljeva, prvo se uzima nulti koeficijent iz prvog prozora i nulti koeficijent iz drugog prozora koji se zasebno podijele sa 2 nakon čega se zbroje. Rezultat tih operacija se ponovno dijeli sa 2 i zbraja se sa uzorkom trećeg prozora koji se također podijeli sa 2 itd. Dobiveni vektor ima 13 elemenata koji se uspoređuju sad 5 referentnih vektora iste duljine. Usporedba se vrši računanjem Euklidove udaljenosti, a najmanji dobiveni broj znači najvjerojatniji samoglasnik.

5.2.3 Pokretanje C programa

Pokretanjem programa u Matlabu generiraju se sve potrebne datoteke koje se koriste prilikom izvođenja programa napisanog u programskom jeziku C. U datotekama su spremljeni testni uzorci govornih signala (samoglasnici) i referentni MFCC koeficijenti koji služe za usporedbu.

Prilikom pokretanja programa u C-u, ispisuje se poruka koja upućuje korisnika kako je potrebno unijeti željeni testni signal tj. naziv tekstualne datoteke u kojoj su spremljeni uzorci testnog signala. Nakon nekoliko statusnih poruka, ispisuju se i podaci o usrednjenim MFC koeficijentima. Usporedbom tog vektora sa referentnim vektorima dobiva se podatak o najvjerojatnijoj interpretaciji testnog signala, što se ispisuje odgovarajućom porukom na ekranu. Pokretanje C programa je moguće izvršiti i iz komandne linije. Potrebno je samo pozicionirati se u direktorij u kojem se program nalazi,

napisati ime izvršne datoteke te nakon nje naziv testnog signala koji se želi učitati (npr. upisivanjem „fix_PrepoznavanjeSamoglasnika.exe 3“).

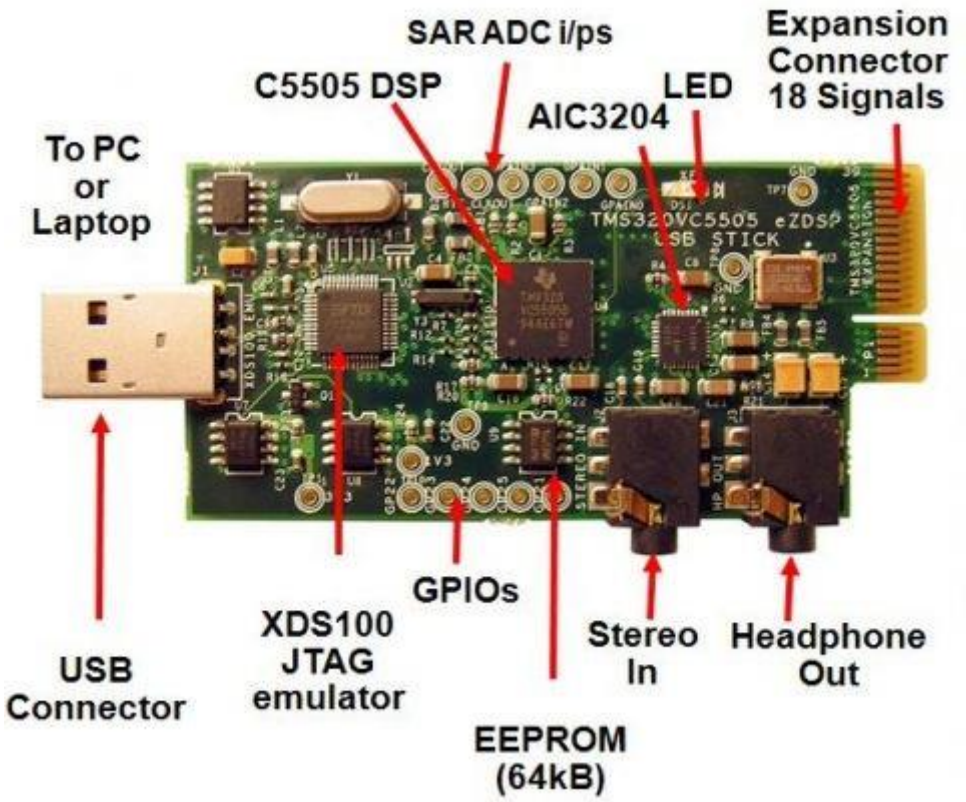
5.3 Implementacija na DSP-u

5.3.1 Razvojni sustav TMS320VC5505 eZDSP USB Stick

Korišten je TMS320VC5505 eZDSP USB Stick razvojni sustav proizvođača Texas Instruments. On sadrži sljedeće:

- TMS320VC5505 procesor (sa periferijama)
- TLV320AIC3204 codec
- Code Composer Studio IDE programski alat

TMS320VC5505 je *fixed-point Digital Signal Processor* (DSP) iz porodice TMS320C5000 Texas Instruments proizvođača. Dizajniran je za aplikacije niske potrošnje. TMS320VC5505 fixed-point DSP je zasnovan na TMS320C55x DSP generaciji CPU procesorske jezgre. C55x DSP arhitektura ostvaruje visoke performanse povećanim paralelizmom radnji fokusirajući se pritom na nisku potrošnju. CPU ima internu sabirničku strukturu koja se sastoji od jedne programske sabirnice, 3 podatkovne sabirnice za čitanje (jedna 32-bitna i dvije 16-bitne), dvije 16-bitne podatkovne sabirnice za pisanje i dodatne sabirnice dodijeljene periferijama i DMA-u. Ove sabirnice donose mogućnost izvođenja do četiri 16-bitnih podatkovnih čitanja i dva 16-bitna podatkovna pisanja u jednom ciklusu. C55x CPU ima dvije MAC jedinice (*multiply and accumulate*), a svaka je sposobna za 17x17 bitno množenje i 32-bitno zbrajanje u jednom ciklusu. Središnja 40-bitna aritmetičko-logička jedinica (ALU), potpomognuta dodatnom 16-bitnom aritmetičko-logičkom jedinicom, omogućuje optimizaciju paralelnog izvršavanja instrukcija i potrošnje energije. Periferni sklopovi uključuju tri brojila opće namjene (od kojih se jedno može koristiti kao watchdog brojilo), generator takta temeljen na fazno vezanoj petlji (PLL) i sklop za ubrzavanje brze Fourierove transformacije (FFT).



Slika 16 TMS320VC5505 eZdsp USB Stick

Važna komponenta razvojnog sustava je TLV320AIC3204 codec (poznat i pod nazivom AIC3204). Njegove su osnovne značajke niska potrošnja, programabilni ulazi i izlazi, fazno vezana petlja i ugrađeni prilagodljivi blokovi za obradbu signala. Ulazni analogno – digitalni pretvornik (ADC) i izlazni digitalno-analogni pretvornik (DAC) mogu raditi na taktovima od 8 kHz do 192 kHz. To su niskošumni pretvornici s omjerom signal-šum (SNR) 93 dB za ulazni, odnosno 100 dB za izlazni pretvornik. Ostali elementi razvojnog sustava su SPI EEPROM memorija veličine 512 Kb, korisnički upravljiva LE dioda, ugrađeni USB XDS100 JTAG emulator, sučelje za proširenje, priključci za testiranje i USB sučelje za spajanje s računalom.

5.3.2 Implementacija na DSP-u

Kao što je rečeno, temelj implementacije sustava za prepoznavanje samoglasnika na DSP-u jest implementacija u C programskom jeziku. Realizacija u C programskoj jeziku je za razliku od realizacije u Matlabu izvedena u frakcionalnoj aritmetici. Takav način implementacije pogoduje korištenom DSP-u (*fixed point*). *Fixed point* DSP-ovi svoju prednost nad *floating point* DSP-ovima imaju u cijeni i brzini. *Floating point* procesori su pak bolji u svemu ostalom. Precizniji su i imaju veći dinamički raspon. Još jedna vrlo bitna stvar po kojoj *floating point* procesori imaju prednost je jednostavnost programiranja. Rad

sa *fixed point* procesorom može značajno otežati posao programeru. Programer treba voditi računa o preljevima. Da bi se preljevi spriječili, programeru moraju biti poznati dinamički rasponi u kojima se pojavljuju podaci u pojedinim dijelovima koda. To može značajno produžiti vrijeme programiranja. Preljevi koji bi nastali zbrajanjem se spriječavaju adekvatnim skaliranjem. Sprječavanje preljeva skaliranjem objašnjeno je u poglavlju *5.2.1*. Važno je skalirati tek onoliko koliko je dovoljno da ne dođe do preljeva čime se postiže veći dinamički raspon podataka i očuvanje točnosti. Skalirati „tek onoliko koliko je potrebno“ je relativna stvar jer za različite ulaze podaci mogu varirati. Voditi brigu o svemu ovome se prije može nazvati umjetnošću nego znanošću.

DSP je programiran pomoću programskog alata Code Composer Studio (CCSv4). Za lakšu inicijalizaciju i konfiguriranje sklopovlja korištena je biblioteka pod nazivom BSL (engl. *Board Support Library*). Nakon manjih preinaka koda napisanog za potrebe implementacije u C-u (npr. drugačijih interpretacija kompajlera), isti je testiran i na DSP-u. Da bi se provjerila ispravnost koda, u oba sustava su dovedeni isti ulazni podaci. Rezultati dobiveni implementacijama u C-u i na DSP-u su se pokazali identičnima, a od rezultata dobivenih u Matlabu razlikuju se u prosjeku u drugoj ili trećoj decimali.

5.3.3 Poboljšanja i optimizacije

Ako se želi realizirati ovakav sustav za prepoznavanje govora u realnom vremenu, bitno je imati brz algoritam. Kako je ranije pokazano, za svaki vremenski odsječak trajanja 20-25 ms izračunat je vektor značajki. Za pomak prozora od 10 ms znači da će se cijeli algoritam morati izvršiti u vremenskom odsječku od 10 ms (15 ms). Kako se vremenski odsječci preklapaju potrebno je imati dva spremnika uzoraka (engl. *buffer*) govornog signala koji će se neprekidno puniti. Kada se jedan napuni počinje se sa obradom, a za to se vrijeme još uvijek puni drugi spremnik. Za ovakav način prikupljanja uzoraka najbolje je koristiti DMA (Direct Memory Access) modul koji će bez pomoći procesora prikupljati uzorke audio codeca te će procesoru dati znak kada se njegov spremnik napuni. Lošija opcija bi bila korištenje prekidne rutine. Lošija je zato što bi tada procesor morao prekidati izvođenje glavnog programa samo kako bi prikupio jedan uzorak, a on itekako ima važnijeg posla. Brza Fourierova transformacija (FFT) je učinkoviti algoritam za računanje diskretne Fourierove transformacije (DFT). FFT je jedan od najčešće korištenih algoritama u digitalnoj obradi signala. DSP-ovi su idealni za takve aplikacije. Korišteni DSP ima hardverski akcelerator za FFT (HWAFFT) koji je usko vezan sa centralnim procesorom. Hardverski akcelerator omogućuje nisku potrošnju energije i visoku brzinu računanja FFT-

a komunicirajući sa procesorom preko koprocesorskih instrukcija. Fizički je smješten izvan DSP jezgre ali ima pristup punom rasponu memorije jezgre te tako može pristupiti jezgrinim internim registrima, akomulatorima i generatorima adresa (AGU). U svrhu optimizacije algoritma sve što nije potrebno mijenjati u obradi izračunati će se prije nje. Sa time se misli na izračune Hammingovog filtra, skupa filtara na Mel frekvencijskoj skali, *twiddle* faktore za FFT i ostalo. Korištena je i funkcija za računanje logaritma bez množenja, a koja je objašnjena u poglavlju 5.2.2. Kako je nulti MFC koeficijent zapravo suma logaritama spektra snage njega je efikasnije izračunati na taj način nego kroz postupak diskretne kosinusne transformacije. To su sve postupci koji optimiziraju rad algoritma tj. smanjuju vrijeme njegovog izvođenja.

6 Sažetak

U okviru ovog rada istraženi su postupci reprezentacije govornog signala koji se koriste u postupcima automatskog prepoznavanja govora ili govornika. Poseban naglasak stavljen je na reprezentacije temeljene na kepstralnom vektoru značajki uz Mel-frekvencijsku skalu. Postupak određivanja vektora značajki implementiran je u programskoj okolini Matlab i testiran je na unaprijed snimljenim govornim zapisima. Algoritam je realiziran i u programskom jeziku C u frakcionalnoj aritmetici. U konačnici je taj algoritam testiran na razvojnom sustavu za digitalnu obradbu signala TMS320VC5505 eZDSP proizvođača Texas Instruments. U svrhu demonstracije, realiziran je jednostavni klasifikator samoglasnika temeljen na euklidskoj udaljenosti estimiranih vektora značajki u odnosu na pohranjene modele.

7 Summary

In this paper various techiques of speech signal representations which are widely used in aplications for automatic speech and speaker recognition have been researched. Special emphasis has been put on representations based on cepstral vector with Mel-frequency scale. Feature extraction procedure has been implemented in program environment Matlab and tested on pre recorded speech samples. Algorithm has been implemented in C programming language in fractional arithmetic. Ultimately, the algorithm has been tested on development tool for digital signal processing TMS320VC5505 eZDSP produced by Texas Instruments. In purpose of demonstration, simple vowel classifier based on euclidean distance between estimated vector of features and reference models has been made.

8 Literatura

- Chip design of MFCC extraction for speech recognition, Jia-Ching Wang, Jhing-Fa Wang, Yu-Sheng Weng
- Techniques for feature extraction in speech recognition system: A comparitive study, Urmila Shrawankar
- Speech Recognition Using Euclidean Distance, Akanksha Singh Thakur, Namrata Sahayam, March 2013
- Feature extraction for speech recognition, Manish P. Kesarkar
- Uvod u digitalnu obradbu govora korištenjem Matlaba, Davor Petrinović

9 Zaključak

Određivanje vektora značajki je vrlo važno u aplikacijama automatskog prepoznavanja govora ili govornika. MFCC analiza predstavlja *state-of-art* u takvim aplikacijama. U ovom radu je pokazano kako se sa vrlo malim brojem podataka (vektor koeficijenata) može dobro opisati govorni signal. Kako se u ovom radu koristi najjednostavnija metoda klasifikacije samoglasnika određivanjem euklidske udaljenosti, dobiveni rezultati su očekivani i zadovoljavajući. Za postizanje boljih rezultata u sustavima automatskog prepoznavanja govora ili govornika mogu se koristiti vektori značajki dobiveni MFCC analizom uz primjenu složenijih klasifikatora temeljenim na skrivenim Markovljevim modelima (engl. *Hidden Markov Models*) ili neuronskim mrežama.