Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Lambe Čočorovski

Sistem za sintezo govora iz besedila

diplomsko delo
na univerzitetnem študiju

Mentor: prof. dr. Dušan Kodek

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01890/2013
Datum: 07.01.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **LAMBE ČOČOROVSKI**

Naslov: **SISTEM ZA SINTEZO GOVORA IZ BESEDILA**
SYSTEM FOR TEXT TO SPEECH SYNTHESIS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Pretvorba zapisanega besedila v govor spada med probleme, s katerimi se znanost ukvarja že zelo dolgo. Problem je zapleten in je poleg tega odvisen od jezika. Na osnovi pregleda obstoječih metod za sintezo govora iz besedila izdelajte sistem, ki deluje za slovenski jezik. Uporabite znana glasoslovna pravila slovenskega jezika in s pomočjo prosto dostopnih rešitev izdelajte vso potrebno programsko opremo. Delovanje sistema preizkusite na več besedilih in s pomočjo množice poslušalcev ovrednotite kvaliteto sintetiziranega govora.

Mentor:
prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Lambe Čočorovski, z vpisno številko **63060038**, sem avtor diplomskega dela z naslovom:

Sistem za sintezo govora iz besedila

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Dušana Kodeka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 27. junija 2013 Podpis avtorja:

Rad bi se zahvalil mentorju prof. dr. Dušanu Kodeku za mentorstvo in pomoč pri izdelavi diplomskega dela. Zahvaljujem se tudi družini za strpnost tekom študija in vsem ostalim, ki so kakor koli pripomogli k nastanku tega dela in mi pomagali pri študiju.

Kazalo

Slike

SLIKE

Povzetek

Področje sinteze govora se z vse zmogljivejšimi računalniki vse bolj razvija. V diplomski nalogi sta za pregled področja predstavljeni metoda sinteze na osnovi baze difonov in sinteza z unit selection metodo. Opisana je osnovna zgradba sintetizatorjev, analiza besedil in uporaba algoritmov za digitalno obdelavo signalov. Uporabljen je DTW algoritem, kot pripomoček za označevanje fonemov zvočnega signala s pomočjo MFCC koeficientov in LPC kodiranje za samo sintezo govora.

Za izvedbo sintetizatorjev je uporabljeno odprtokodno okolje Festival Speech Synthesis System z Edinburgh Speech Tools in napotki za gradnjo novih jezikov s pomočjo projekta Festvox. Predstavljene so tudi nekatere značilnosti slovenskega knjižnega jezika in njihova uporaba v Festivalu. Sintetizatorja sta bila uspešno zgrajena in rezultati so ovrednoteni z MOS lestvico.

Ključne besede: sinteza govora, Festival, Festvox, sinteza z difoni, unit selection

Abstract

Speech synthesis is increasingly developing as the computers are getting more powerful. In this work, the area of speech synthesis is introduced through a diphone and unit selection based synthesis methods. The basic architecture of the synthesizers, text analysis and the use of algorithms for digital signal processing are described. DTW algorithm was used as a tool for labelling phonemes from an audio signal using MFCC coefficients and LPC coding of the speech signal.

Festival Speech Synthesis System with Edinburgh Speech Tools and Festvox documentation are used for building new languages. The characteristics of the Slovenian language are described and their usage in the Festival. We have successfully built two speech synthesizers and the results are evaluated using a MOS scale.

Key words: speech synthesis, Festival, Festvox, diphone synthesis, unit selection

Poglavje 1

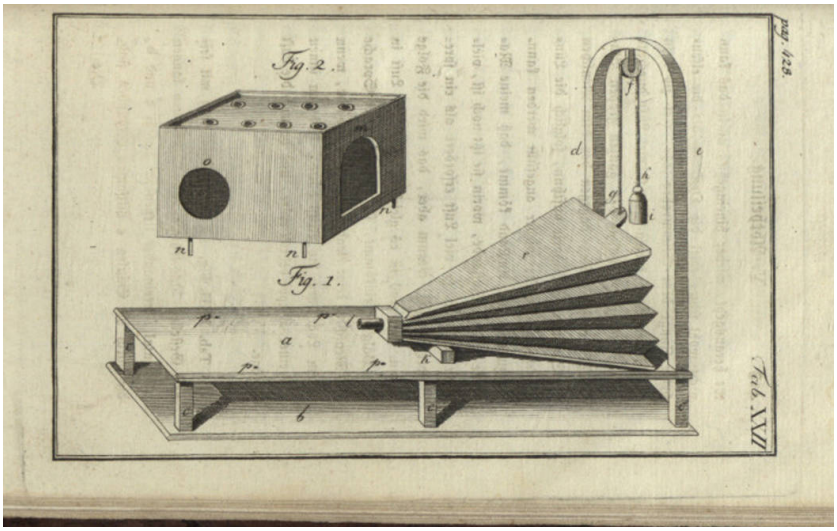
Uvod

Text-to-Speech (TTS) ali "*besedilo v govor*"je sistem, ki na računalniku zapisano besedilo, pretvori v govor, ki ga je mogoče slišati preko zvočnikov. S pomočjo OCR (*Optical Character Recognition*) je za natisnjena besedila možno dodati optično razpoznavo in nato prenos na računalnik, v primeru, da bi želeli razširiti sam sistem.

1.1 Zgodovina TTS sistemov

Prvi zapiski o napravah, ki bi bile zmožne proizvajati zvok, segajo v leto 1791, ko je Wolfgang von Kempelen izdelal pnevmatični sintetizator (slika 1.1) [1]. Operator je stiskal z zrakom napolnjen meh, ta je potoval skozi piščal in simulirano človeško govorno cev, ki jo je bilo mogoče ročno spreminjati. Piščali je bilo mogče zamenjati za izgovorjavo pripornikov in simulacijo nosnic. Von Kempelen je v svojem delu podrobno preučil vse dejavnike, ki vplivajo na izgovorjavo glasov in se je pri izdelavi svoje naprave tega tudi držal. Tako je na sliki 1.2 vidno, kako je želel izdelati umetna usta in nos. Za izgovorjavo nosnikov je operator moral pustiti "nosni"luknji odkriti, sicer pa jih je moral z dvema prstoma (sredinec in prstanec) desne roke zapreti. S kazalcem in mezincem pa je pritiskal na dve ročici, ki sta bili povezani s piščalmi, skozi katere je ob pritisku na njiju šel skozi zrak in je tako ustvaril

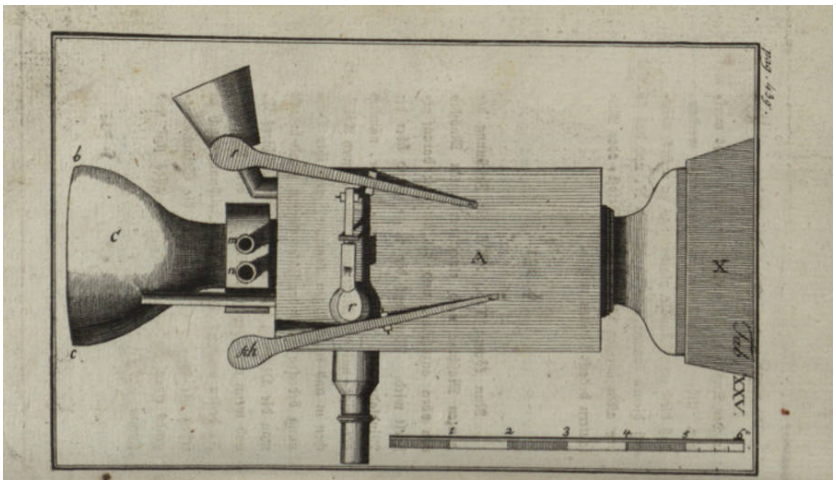
glas fonemov s, š, z in ž. Za uspešno delovanje naprave je operator moral imeti veliko prakse, če je želel ustvariti zaporedje glasov, ki bi imelo pomen.



Slika 1.1: Prikaz Wolfgang Von Kempelenove naprave iz njegove knjige *Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine*. Operator je z desno roko stiskal meh, protiutež pa je služila za njegovo napihovanje. Meh simulira človeška pljuča.

Potrebno je bilo počakati do 20. stoletja in na razvoj elektronike, da je prišlo do ponovnega razvoja sistemov za sintezo govora. To področje je zajelo veliko zanimanje zaradi raziskav o kompresiji signala za namen telefonske komunikacije, ki se je razvijala. Tako je Homer Dudley izdelal napravo imenovano *Vocoder ("voice coder")* , ki velja za enega prvega resnejših poskusov elektronskega sintetizatorja in je osnova večine današnjih sintetizatorjev. Kasneje je za popularizacijo in demonstracijo svojega dela izdelal *Voder*, ki ga je moral upravljati izurjen demonstrator. S pedalom je kontroliral ton, s prsti obeh rok pa amplitudo pasovno prepustnih filtrov čez spekter frekvenčnih področij, ki so pomembna za človeški govor.

Kasneje so računalniki postajali vse zmogljivejši (postali so zmožni obedelovati veliko podatkov v krajšem času in shranjevanje podatkov ni bila več ovira). Tako so znanstveniki prišli do ideje o sintezi na podlagi lepljenja. Naj-



Slika 1.2: Skica modela govorne poti iz pljuč do ust.

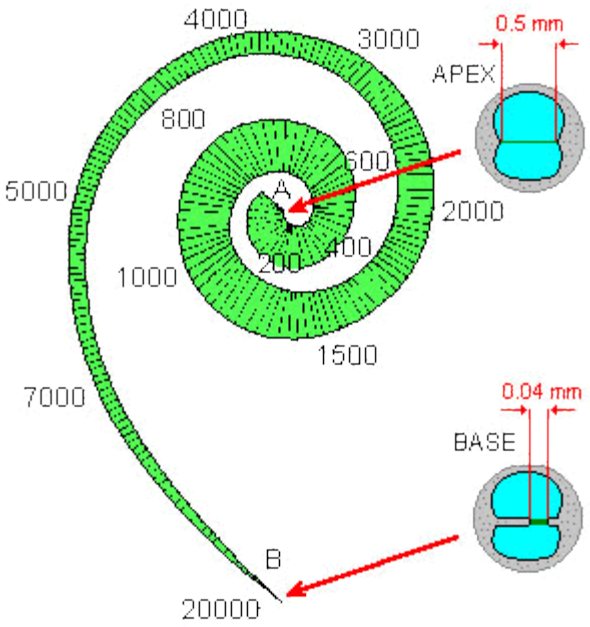
prej je v 70-ih letih nastala ideja o lepljenju difonov z informacijo o besedilu iz katerega so vzeti (naglaševanje, lokacija difona, itd.). V začetku 90-ih je Yoshinori Sagisaka iz laboratorija Advanced Telecommunications Research iz Japonske nadgradil sistem z možnostjo večih različic posameznih difonov.

Naslednja nadgradnja sistema so bili *Unit-selection* sistemi. Ideja je, da z uporabo posplošene baze zvočnih zapisov, pokrijemo večji del možnih izgovorjav besed (dolžina izgovorjave, naglaševanje, ...) kot z bazo difonov [12].

Področje je danes dostopno vsem, ki imajo osebni računalnik in tako omogoča še hitrejši razvoj. S sintezo govora se danes razvija tudi zaznava govora in prevajanje besedil, saj pri je v ozadju veliko skupnih algoritmov procesiranja singnalov. Tudi mobilna telefonija in razvoj računalniških iger prispevata k razvoju. Danes poznamo veliko računalniških odzivnikov s katerimi lahko komuniciramo in nadomeščajo ljudi, ki so prej bili zaposleni za komunikacijo s telefonskim operaterjem za reševanje problemov naročnikov ali pa na primer za napovedovanje prihodov letal na letališčih. Področje se torej razvija tudi iz ekonomskih razlogov.

1.2 Komunikacija in zaznavanje govora

Človek je skozi evolucijo začel uporabljati govor za lažjo komunikacijo z drugimi ljudmi. Videl je, da lahko ustvarja glasove, ki so med seboj razlikujejo in katerih pravilna kombinacija tvori pomenljive besede, s katerimi se da sporazumevati. Kadar želimo nekaj izgovoriti, nam možgani željo kodirajo in pošljejo živčne signale do mišic v govorni cevi, pljuč, nosne in ustne votline in ustnic. Vse je potrebno uskladiti, da izgovorimo želene glasove. Na drugi strani prejemnik prejme sporočilo, ki je potovalo po zraku v obliki zvočnega valovanja in ga dekodira v možganih. Na sliki 1.3 je prikazana membrana v ušesu, ki zvočni signal loči po frekvencah. Pri nižjih frekvencah uho lahko z večjo resolucijo loči tone. Kadar so frekvence višje pa posamezna frekvenčna območja združuje in jih ni mogoče slušno razlikovati.



Slika 1.3: Membrana v ušesu in frekvence, ki jih zaznava na posameznih delih [2].

Poglavje 2

Glasoslovje

Ko se ljudje učimo govora, se naučimo izgovorjave vseh fonemov potrebnih za izgovorjavo besed. Da so besede smiselne morajo biti sestavljene iz manjših delov, katerih zamenjava pomeni spremembo pomena besede. Tem glasovnim enotam pravimo fonemi. Za računalniško sintezo govora je potrebno vse te foneme definirati in ločiti, če želimo izdelati dober govorni sistem. Algoritem v ozadju je v veliki meri podoben tistemu, ki poteka pri našem nezavednem učenju izgovorjave besed. Potrebno je znati izgovarjati vse možne glasove, ki nastopajo v poljubnem jeziku, če bomo želeli izgovarjati besede. Jeziki se med seboj razlikujejo po glasovih, ki v njih nastopajo in ker je namen diplomske naloge preučiti sestavo slovenskega govora, sledi opis glasov slovenskega knjižnega jezika.

2.1 Glasovi slovenskega knjižnega jezika

Slovenski knjižni jezik ima 29 fonemov. Delimo jih na samoglasnike in soglasnike. Samoglasniki so (*i*, ozki in široki *e*, ozki in široki *o*, *u* in polglasnik. Soglasniki so zvočniki (*m, n, r, l, j, v*) in nezvočniki *p, b, f, t, d, s, z, c, š, ž, č, dž, k, g, h*. Nezvočnike delimo glede način izgovora na zapornike (*p, b, t, d, k, g*), pripornike (*f, s, z, š, ž, h*) in zlitnike (*c, č, dž*).

2.1.1 Posebne glasovne zveze in pravila za izgovorjavo

Pri pregledu glasov SKJ nas zanima njihov fonetični opis. Ta je pomemben za razumevanje proizvodnje govora. Delitev je uporabna, zato ker ena črka ne pomeni en glas. Obstajajo posebne glasovne zveze, ki jih je mogoče zapisati kot ″pravila″ za izgovorjavo besed (glej LTS)[3].

Črko *l* beremo kot *u* kadar:
1.) *-ol-* nastopi med soglasnikoma iste besede: *jabolko, solza, žolč*
2.) *-lc-, -lk-, -lsk-, -lstv-* nastopijo v izglagolskih izpeljankah in se nanašajo na živega vršilca dejanja: *bralca, bralka, bralstvo*

Kadar na koncu ene besede in začetku druge nastopita dva enaka soglasnika, ju izgovorimo kot enega daljšega. Lahko tudi kot enega navadnega, v kolikor to ne povzroča dvoumnosti. Če sta zapornika ali zlitnika se lahko izgovarjata posebej:
gledal v vzorec, jaz z zetom, jaz s sinom.

V zvezi *t/d + c/č/dž* (zobni zapornik + zlitnik), izgovorimo oba glasova ali samo dolgi zlitnik: *pod cekar* ali *pocekar.*

V zvezi *e/i/e + u*, prvih ne izgovarjamo: *nesel* beremo kot *nesu.* Napačno bi bilo izgovarjati *e.*

V zvezi *i* + samoglasnik, zlasti v besedah z grško-romanskih poreklom, ju izgovarjamo kot *i + j* + samoglasnik: *pacient* bermo kot *pacijent.*

Ta pravila večinoma veljajo in so dober približek. V primeru besede *pacient* se vidi problem zapisa pravil SKJ. V besedi nastopi glas *j*, ni pa zapisan v besedi. Včasih pa glas *j* pišemo s črko *i* (npr. *celuloid*). V drugih jezikih, kot je angleščina, so zapisi besed dosti bolj različni od izgovorjave in je težje zapisati takšna pravila.

Poglavje 3

Sinteza govora

Sinteza ali sestava govora (ang. *speech synthesis*) je proces umetne tvorbe človeškega govora. Kot je opisano v poglavju 1.1 so bili sistemi za sintezo najprej mehanski, kasneje so potrebovali operaterja za upravljanje z elektronskimi komponentami. Danes, s široko uporabo osebnih računalnikov, telefonov, dlančnikov, ipd. pa so to TTS sistemi, kjer uporabnik definira vhodno (poljubno) besedilo za sintezo.

3.1 Analiza besedila

Ker je vhod v sistem besedilo (zaporedje znakov - črke, številke, posebni znaki, itd.), je potrebno dobiti besede, ki jih bo sistem znal prebrati. Kadar sami beremo besedila znamo zaradi dolgoletnih izkušenj izluščiti potrebne besede, znamo jih pravilno dolgo izgovoriti in vstavljati potrebne premore med besedami. Pri uporabi trenutnih sistemov za sintezo govora, lahko hitro opazimo, da delajo veliko napak - njihova analiza besedil je navadno slabša od same izgovorjave.

Primer: za zapis *"20.05 je deževalo."* vemo, da gre za zapis datuma, ki ga preberemo kot *"dvajestega petega"* ali pa *"dvajsetega maja"*. Izkušen bralec s tem ne bi imel problemov, marsikateri sistem za analizo besedil pa

ne bi vrnil pravilnih besed. V slovenščini, kakor v drugih jezikih, obstajajo posebna pravila za branje datumov. Zato je za pravilno sintezo primera najprej potrebno ugotoviti, da gre za zapis datuma. Nato zapis damo v sistem za obdelavo datumov, ki vrne pravilno zaporedje besed, za katere najdemo izgovorjavo v leksikonu ali po ročno definiranih pravilih (LTS).

Tako v zgornjem primeru, kot tudi za ostala besedila, je potrebna uporaba analize besedil, ki jo delimo na tri dele [7]:
1.) klasifikacija: besedilo klasificiramo v znane razrede (naravni jezik, okrajšave, datum, ura, e-poštni naslovi itd.)
2.) dekodiranje: iz razredov ugotovimo pomen zapisov (v zgornjem primeru sistem ugotovi, da 20 pomeni dan v mesecu in 5 je mesec v letu)
3.) verbalizacija: zapis pretvorimo v besede, ki jih bo sistem znal izgovoriti (*"20."* pretvori v *"dvajsetega"* in *5.* v *"maja"*). Uspešen sistem bo pri verbalizaciji meseca znal izbrati primernejšo formo (*petega* ali *maja*).

3.2 Označevanje

Ker je poterebno vedeti kdaj v posnetkih nastopajo posamezni fonemi, je potrebno na nek način označiti čas trajanja fonemov. Le na takšen način je mogoče iz posnetkov vzeti manjše kose, ki niso nujno le posamezni fonemi (lahko so tudi difoni ali cele besede) in jih nato zlepiti v večje enote. Najenostavnejši način je poslušanje posnetkov in ročno označevanje začetka in konca, vendar je v takem primeru potreben človek z dobrimi slušnimi posobnostmi in je ob večjem številu posnetkov takšen način tudi časovno zelo zahteven. Drugi način pa omogoča avtomatično označevanje. Da je to mogoče, je potrebno uporabiti algoritem za avtomatično razpoznago govora (ASR) in posnetke z natančno označenimi glasovi, ki v njih nastopajo. V tem poglavju sledi opis metode, ki je najbolj uporabljena pri določanju akustičnih lastnosti govorca. Računa MFCC koeficiente, ki jih lahko uporabimo tudi za razpoznavo govorca, a to ni pomembno v tem primeru. Z DTW algorit-

mom nato poiščemo podobnost med dvema posnetkoma in tako avtomatično označujemo glasove.

7

3.2.1 Določanje in označevanje periode tonov

Osnovna frekvenca F_0 je najnižja frekvenca periodičnega signala in frekvenca s katero oscilirajo glasilke (le pri izgovorjavi zvočnikov) [18]. Perioda osnovne frekvence je

$$T_0 = \frac{1}{F_0}. \tag{3.1}$$

Število vzorcev signala v periodi je

$$L = T_0 F_s, \tag{3.2}$$

kjer je F_s frekvenca vzorčenja. F_0 je pri moških med 50 in 120 Hz. Pri ženskah pa med 140 in 400Hz.

Pri določanju periode tona iz zvočnega signala je potrebno uporabiti PDA (*Pitch Detection Algorithm*) algoritem. To je potrebno, ker veliko algoritmov pri sintezi govora potrebuje ravno informacijo o periodi tonov (npr. PSOLA algoritem za spreminjanje trajanja zvočnega signala in tona ali pa tudi določanje spodaj opisanih MFCC koeficientov).

3.2.2 MFCC

Zaradi posebne zgradbe človeškega ušesa, lahko zaznavamo le določene frekvence (slika 1.3) - poslušalec ne loči razlike med toni s frekvencami v posameznih frekvenčnih območjih. Te so na območju od 0 do 1 kHz skoraj linearno razporejene, nad 1 kHz pa logaritemsko. To lahko modeliramo z zaporedjem filtrov (slika 3.1).

MFCC (*Mel Frequency Cepstral Coeficients*) olajša (pohitri) računanje s signalom, saj ga predstavimo z nekaj koeficienti. Koeficienti ne predstavljajo

kompresiranega signala, ker ni mogoča rekonstrukcija signala iz samih koeficientov (izguba informacije). V splošnem je enačba za računanje kepstra

$$c(n) = IDFT(log(|DFT(x(n) * w(n))|)), \tag{3.3}$$

kjer je $x(n)$ vhodi signal, $w(n)$ pa uporabljena okenska funkcija. Pri računanju MFCC pa nastopi nekaj razlik.

Poudarjanje višjih frekvenc

V prvem koraku pri računanju MFCC posneti govorni signal pošljemo skozi visoko prepustni filter, katerega naloga je poudariti višje frekvence govora. Filter je opisan z enačbo (3.4) in njenim z-transformom (3.5), kjer je a navadno med 0.9 in 1.

$$y(n) = x(n) - a * x(n-1) \tag{3.4}$$

$$H(z) = 1 - a * z^{-1} \tag{3.5}$$

Delitev na okvirje

V tem koraku posnetke razbijemo na okvire dolžine nekaj deset milisekund. Na tem intervalu je odvisno od frekvence vzorčenja določeno število vzorcev. Za kasnejše poenostavljeno računanje DFT je bolje izbrati število vzorcev dolžine potence števila 2.

Uporaba okenske funkcije

Signal moramo pomnožiti z okensko funkcijo (3.8), sicer pride do Gibbsovega pojava. Najbolj pogosta izbira okenske funkcije pri računanju MFCC je Hammingovo okno (3.6).

$$w_h(n) = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{N-1}, & 0 \leq n \leq M \\ 0, & \text{sicer} \end{cases} \tag{3.6}$$

Boljša izbira je Kaiserjevo okno (3.7) [6] .

$$w_k(n) = \begin{cases} \frac{I_o(\alpha\sqrt{1-(\frac{n}{M/2})^2}}{I_o(\alpha)}, & -\frac{M-1}{2} \leq n \leq \frac{M-1}{2} \\ 0, & \text{sicer} \end{cases} \tag{3.7}$$

$$y(n) = x(n) * w(n) \tag{3.8}$$

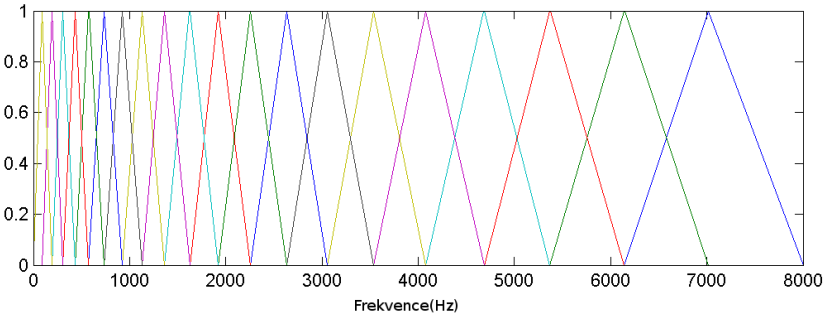
Izračun DFT

S FFT izračunamo DFT in dobimo frekvenčno predstavitev posameznih intervalov. Pri tem nas faza ne zanima, saj človek ne zaznava faznih zamikov.

$$Y(\omega) = FFT(h(t) * x(t)) = H(\omega) * X(\omega) \tag{3.9}$$

Uporaba zaporedja filtrov

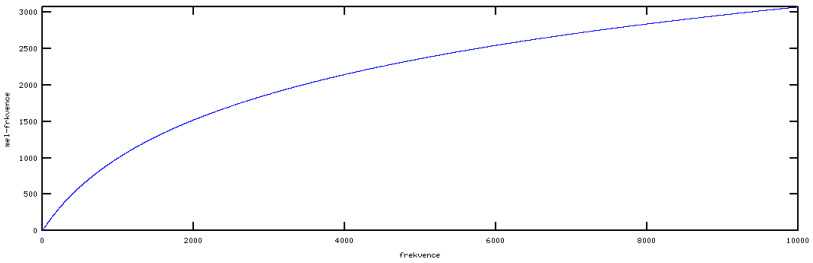
Za vsako frekvenčno območje izberemo trikotni filter, ki se začne na sredini prejšnjega in konča na sredini naslednjega in so enakomerno razporejena po mel frekvencah (slika 3.1). Po filtriranju izračunamo uteženo vsoto spektral-



Slika 3.1: Zaporedje trikotnih filtrov.

nih komponent m_i (3.11), ki predstavlja pripadajoča spektralna območja. Zaradi narave človeškega slušnega sistema je to potrebno še pretvoriti na mel lestvico (3.10).

$$f_{mel} = 1127 * ln\left(1 + \frac{f_{lin}}{700}\right) \tag{3.10}$$



Slika 3.2: Mel frekvece v odvisnosti od linearne po (3.10).

$$m(i) = \sum_{k=0}^{N-1} |Y(k)|^2 * F_i(k) \tag{3.11}$$

Izračun DKT

Zadnji korak predstavlja računanje DKT (3.12) (*diskretna kosinusna transformacija*) s katerim frekvenčno predstavitev signala pretvorimo nazaj v časovno. Pred tem je potrebno logaritmirati rezultat iz prejšnjega koraka $log(m_i)$. Logaritmiramo zaradi narave logaritma, ki produkt pretvori v vsoto. Rezultat tega so MFCC koeficienti

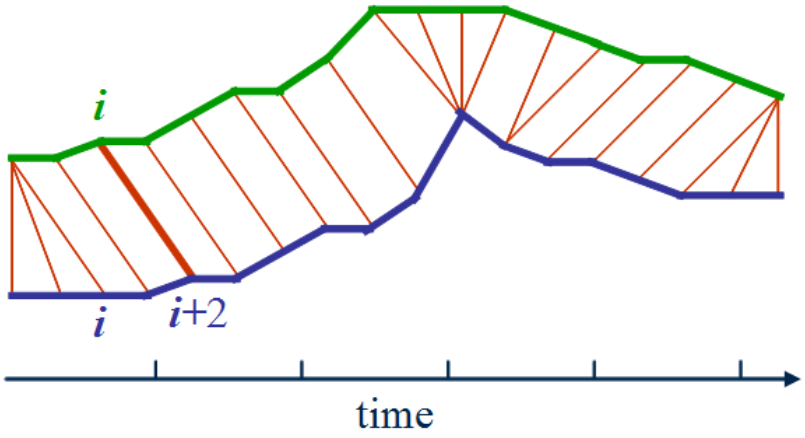
$$C(k) = \alpha(k) \sum_{n=0}^{N-1} log(m(n)) cos[\frac{\pi}{N}(n + 1/2)k], \qquad 0 \leq k \leq M \tag{3.12}$$

kjer je

$$\alpha(k) = \begin{cases} \sqrt[2]{\frac{1}{N}}, & \text{za} \quad k = 0, \\ \sqrt[2]{\frac{2}{N}}, & \text{za} \quad 1 \leq k \leq M. \end{cases} \tag{3.13}$$

3.2.3 Dynamic Time Wraping

DTW je algoritem na podlagi dinamičnega programiranja s katerim poiščemo podobnost dveh različno dolgih zaporedij (v našem primeru so to zvočni posnetki) [13]. Ker ni mogoče dvakrat enako hitro izgovoriti določenih glasov (fonemov, difonov, ipd.), bodo večkrat posneti isti glasovi različni v hitrosti izgovorjave ali pa v dolžini datoteke. Kot je prikazano na sliki 3.4 lahko



Slika 3.3: DTW najde podobnost med dvema različno dolgima zaporedjema.

besedo "primer"izgovorimo z dalšim glasom "e", torej "primeer". Tudi od govorca je odvisno kako hitro govori. Rezultat DTW je število, ki meri podobnost med dvemi zaporedji. Manjše število pomeni večjo podobnost. Kadar imamo posnetek ene besede je treba upoštevati:

1. na začetku posnetka mora biti tišina,
2. določiti je potrebno mejo, ki predstavlja prag tišine. Kadar jo presežemo pomeni začetek besede,
3. konec besede se določi enako kot v koraku 2.

Kadar je v posnetku več besed pa je postopek enak, a je potrebno najti meje med besedami.

DTW je uporaben tudi pri ASR. V tem primeru mora algoritem poiskati najboljše prileganje z znano besedo iz slovarja besed, ki jih znamo zaznati.

Delovanje

Z A (3.14) označimo zaporedje vektorjev (MFCC) dolžine n, ki predstavlja posnetek, ki ga želimo primerjati z zaporedjem vektorjev B (3.15) dolžine m iz znanega posnetka.

$$A = a_1, a_2, ..., a_n \tag{3.14}$$

$$B = b_1, b_2, ..., b_m \tag{3.15}$$

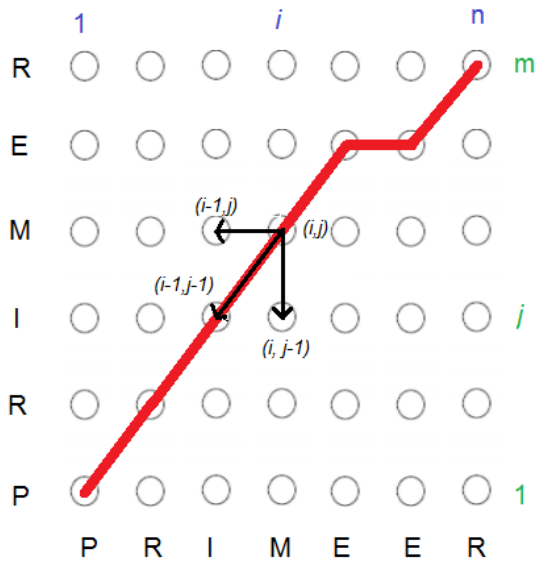
Potrebno je poiskati optimalno pot $P = p_1, p_2, ..., p_k$, pri čemer je par $p_s = (i_s, j_s)$ po parih vektorjev a_i in b_j. Ker sta zaporedji različno dolgi, se nekateri elementi zaporedja A (oz. B) večkrat "preslikajo"v elemente zaporedja B (oz. A) (sliki 3.3 in 3.4).

$$\gamma(i,j) = d(a_i, b_j) + min \begin{cases} \gamma(i, j-1) \\ \gamma(i-1, j) \\ \gamma(i-1, j-1 \end{cases} \tag{3.16}$$

Pri tem upoštevamo robna pogoja: $P(1) = (a_1, b_1)$ in $P(k) = (a_n, b_m)$. To storimo zato, da ne preiskujemo vseh možnih poti. Dodatna omejitev, ki jo lahko postavimo je onemogočenje velikih časovnih preskokov (indeksa i in j se na poti ne zmanjšujeta - kvečjemu ostaneta enaka). Zato omejimo iskanje le na sosednje vektorje (3.16). To omogoča uporaba dinamičnega programiranja in Bellmanov princip optimalnost. Ta pravi, da je na optimalni poti med začetno točko P(1) in končno točko P(k) optimalna tudi pot med katerima koli točkama na tej poti. Algoritem računa akumulirano vsoto (razdaljo) (3.16), pri čemer je $d(a_i, b_j)$ evklidska razdalja med vektorjema. Na vsakem koraku torej velja, da je akumulirana razdalja $\gamma(i, j)$ od točke $(0, 0)$ do točke (i, j) najmanjša (optimalna).

3.3 Naglaševanje in izgovorjava

Med govorom izgovarjamo glasove besed različno dolgo. Kadar želimo nekaj poudariti lahko besedo drugače naglasimo, ali pa kakšno črko izgovorimo



Slika 3.4: Delovanje DTW.

z daljšim glasom (npr. besedo *kaj* lahko izgovorimo z daljšim glasom *a*, ko želimo izraziti začudenje). Zato moramo imeti način s katerim lahko izgovorjavo glasov nastavljamo različno dolgo.

3.3.1 CART

CART (*Classification And Regression Tree*) algoritem je učilni algoritem, ki se uporablja pri strojnem učenju, modeliranju podatkov, statistiki in rudarjenju podatkov. CART drevo je mešanica CT (*Classification Tree*) in RT (*Regression Tree*) drevesa. Prvo uporabljamo za napovedovanje razreda h kateremu pripadajo podatki, drugo pa za numerično napovedovanje. Pri CART drevesih pa lahko uporabimo oboje skupaj.

V odločitvenem drevesu se odločamo glede na vrednost spremenljivke od korena drevesa do listov (sinov). Kadar dosežemo pogoj za zaustavitev (ne odločimo se več za nobeno opcijo ali pridemo do lista), je postopek

končan. V vsakem koraku se odločimo ali gremo na ″levo″ (levo poddrevo) ali ″desno″ (desno poddrevo). Zaporedje odločitev predstavlja skupek pogojev, ki morajo biti izpolnjeni, da pridemo do želenega rezultata.

Ob znanih vhodih $X_1, X_2, ...X_n$ želimo napovedati izhod Y. Enačba za povprečno medsebojno informacijo (3.17), nam pove koliko izvemo o izhodu Y iz vhoda X.

$$I(Y;A) = H(Y) - H(Y|X) \tag{3.17}$$

Drugi del enačbe nam pove koliko se v povprečju zmanjša negotovost o izhodu pri znanem X. Če enačbo formuliramo drugače

$$I(Y;X = x) = H(Y) - H(Y|X = x), \tag{3.18}$$

lahko izračunamo izgubo negotovosti za vsako vrednost x_i. Algoritem je rekurziven in na vsakem nivoju drevesa deluje rekurzivno. Pri tem išče vprašanje Q, ki maksimizira

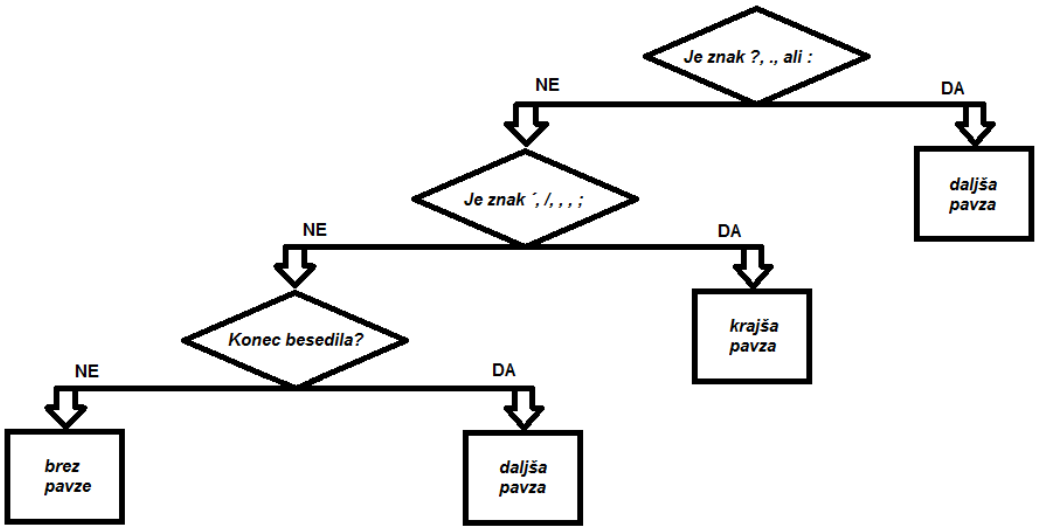
$$I(Y;Q|X = x) = H(Y|X = x) - H(Y|X = x, Q). \tag{3.19}$$

V našem primeru so CART drevesa uporabna pri določanju relacije glas/črka (glej poglavje 2), pri določanju premora ob nastopu ločil in pri določanju dolžine izgovorjave glasov.

3.3.2 Izgovorjava

Pavza pri ločilih

S CART drevesi lahko tudi napovedujemo ali je potrebno narediti pavzo med besedami ki jih izgovarjamo (npr. kadar nastopijo ločila) in njeno dolžino. Primer CART drevo za ta namen je na sliki 3.5 Po koncu povedi ali ob dvopičju (npr. pred naštevanjem ali pri razlagi pojma) naredimo pavzo preden začnemo brati naprej. Ta pavza je navadno daljša od pavze ob vejicah ali podpičju.



Slika 3.5: Primer drevesa za odločitev ali bo ali ne bo pavze.

Dolžina izgovorjave

Glasovi se izgovarjajo različno dolgo. Poznamo tri različne metode:

- izgovorjava vseh glasov je enako dolga,
- dolžino določimo ročno za vsak glas posebej, a je še zmeraj fiksna. Pri tem uporabimo že znane podatke iz sorodnih jezikov.
- dolžino določimo z uporabo učilnih algoritmov.

Prvo metodo izberemo navadno na prvem koraku, ko sistem še preizkušamo ali sploh deluje. Odločitev o izbiri druge ali tretje metode je odvisna od časa, ki ga imamo na razpolago. Če imamo veliko znanja je tretja metoda dobrodošla, a je druga primernejša v kolikor se prvič srečujemo s področjem sinteze govora.

3.4 Leksikon in branje nenaravnih zapisov

Pri branju besedila se mora sistem odločiti po kateri metodi bo sestavil želeno izgovorjavo [11]. Na voljo so tri možnosti:

- izgovorjavo najde v leksikonu,
- izgovorjavo izdela po naučenih pravilih,
- izgovorjavo izdela po ročno definiranih pravilih (LTS).

Nenaravni zapisi so lahko različnih vrst. To so lahko matematični izrazi (npr. $a^2 + b^2 = c^2$), kemijske formule (H_2O), števila, datumi, okrajšave, naslovi elektronske pošte itd. Za vsakega izmed teh sklopov je potrebno narediti sistem, ki jih bo znal dekodirati v pripadajočo formo in nato z uporabo pravil spremeniti v ustrezen naravni jezik v obliki besed (verbalizacija) [7]. Ta proces je del analize besedila.

Od uspešnega sistema za sintezo pričakujemo, da bo znal prebrati tudi števila. Števila lahko pomenijo letnice, datum, vsoto denarja ali pa kaj drugega. Tako ima lahko, odvisno od konteksta, enako število več različnih pomenov in tudi izgovorjavo. Pri branju števil je torej potrebno imeti funkcijo, ki bo ob zaznanem številu vrnilo tekst s pravilno izgovorjavo.

3.5 LPC

Pri vsakem signalu lahko predpostavimo model, v katerem vzorce signala napovemo iz prejšnjih vzorcev [4, 5]. Pri napovedi nastopi napaka e_i, ki je razlika med vhodnim signalom s_i in njegovo linearno napovedjo $\hat{s}_i$

$$e_i = s_i - \hat{s}_i, \tag{3.20}$$

kjer je

$$\hat{s}_i = \sum_{k=1}^{p} a_k s_{i-k}. \tag{3.21}$$

a_k so LPC koeficienti, p pa red napovedi. Želene koeficiente a_k dobimo z iskanjem minimuma kvadratne napake

$$E_i = \frac{1}{M}\sum_{j=1}^{M} e_j^2 = \frac{1}{M}\sum_{j=1}^{M}(s_j - \hat{s}_j)^2 = \frac{1}{M}\sum_{j=1}^{M}(s_j - \sum_{k=1}^{N} a_k s_{j-k}). \tag{3.22}$$

M pomeni število vzorcev signala, N pa število prejšnjih vzorcev. Z računanjem parcialnih diferencialnih enačb

$$\frac{\partial E}{\partial a_k} = 0\Big|_{a_{k=optimalen}}. \tag{3.23}$$

Dobimo

$$R(i) = \sum_{k=1}^{p} \alpha_k R(|i-k|), \tag{3.24}$$

kjer je

$$R(k) = \sum_{j=0}^{N-1-k} s_j s_{j+k} \tag{3.25}$$

avtokorelacija signala *s*. Zdaj lahko enačbo poenostavljeno zapišemo v matrični obliki:

$$\begin{bmatrix} R(0) & R(1) & ... & R(N-1) \\ R(1) & R(0) & ... & R(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & ... & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(N) \end{bmatrix} \tag{3.26}$$

Ali skupaj drugače zapisano

$$R = \alpha * r \tag{3.27}$$

Izkaže se, da je R Toeplitzova matrika, kar poenostavi računanje rešitve

$$\alpha = R^{-1} * r \tag{3.28}$$

po Levinson-Durbinovem algoritmu, ki pohitri klasično računanje inverzne matrike

$$E^{(0)} = R(0) \tag{3.29}$$

$$k_i = \frac{R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)}{E^{(i-1)}}, \quad 1 \leq i \leq p \tag{3.30}$$

$$a_i^{(i)} = k_i \tag{3.31}$$

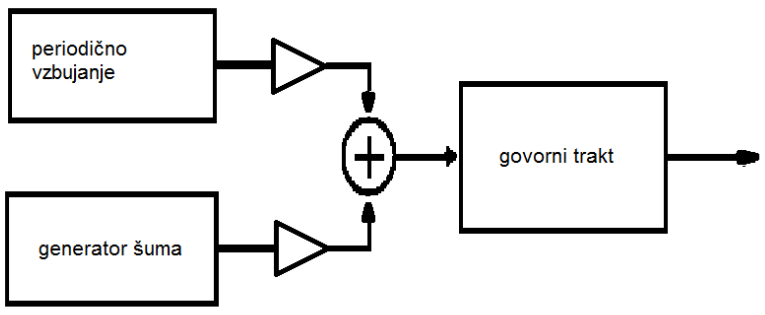
$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}, \quad 1 \leq k \leq i-1 \tag{3.32}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{3.33}$$

Na ta način dobimo optimalne koeficiente predikcije a_k. Z uporabo okenske funkcije (npr. Hammingovo okno) lahko zmanjšamo napako na začetnih in končnih vzorcih v okvirju, saj je tam napaka predikcije zaradi sosednjih ničel največja. LPC koeficiente nato uporabimo za samo sintezo govora obratnim procesom kodiranja [20].

3.6 Vrste sinteze

V tem poglavju so predstavljene nekatere vrste računalniške sinteze govora. Ker je pri izdelavi metod za sintezo potrebno upoštevati fonetične lastnosti glasov določenega jezika in način njihove tvorbe, lahko pridemo do modela na sliki 3.6. Kadar izgovarjamo zvočnike nam glasilke vibrirajo, pri nezvočnikih



Slika 3.6: Model tvorbe govora.

pa ne. Vibracije glasilk so periodične z osnovno frekvenco F_0. Za zvočnike je torej potrebno periodično vzbujanje. Pri nezvočnikih ni osnovne frekvence

in to modeliramo z generatorjem naključnega šuma. Oba generatorja vodita v model govornega trakta, ki ga predstavimo s časovno variantim digitalnim filtrom. Na vhodu v filter je stikalo, ki po potrebi (odvisno od tega ali gre za zvočnik ali nezvočnik) preklaplja na generator periodičnega vzbujanja oz. na generator šuma. Z vsem skupaj modeliramo lastnosti govornega trakta. Model lahko zapišemo s prenosno funkcijo:

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.34}$$

G je ojačanje amplitude signala, kar modelira pretok zraka. Koeficiente a_k filtra lahko izračunamo z LPC, saj ju v obeh primerih opisuje ista prenosna funkcija filtra.

3.6.1 Akustični model

Gre za metodo, pri kateri računalniško simuliramo človeško govorno cev (*vocal tract*), jezik, ustnice, itd. Ta sistem lahko ustvari zelo naravno zveneči govor, porabi pa veliko računanja pri sintezi in je v praksi neuporaben [9].

3.6.2 Metode na podlagi lepljenja

Uporaba lepljenja za sintezo govora (*concatenation synthesis*) je način, ki se zdi na prvi pogled najbolj racionalna odločitev, kadar se odločamo o izbiri metode za sintezo govora. Tukaj združujemo posnete dele človekovega govora in se nam zdi, da bi se z dobrim lepljenjem dalo sestaviti zelo naravno zveneči sistem. Sistemi za sintezo govora, ki tvorijo izhodni govorni signal s pomočjo lepljenja (oz. konkatenacije), "lepijo"posamezne manjše posnetke med sabo v skupni končni rezultat. To je lahko na nivoju fonemov, difonov, trifonov, itd. Lahko so tudi cele besede ali celo stavki. Osnovna zamisel takega sistema je, neodvisno od velikosti delov, lepljenje manjših delov v večje. To je potrebno zato, ker v bazi ni mogoče imeti vseh možnih besed določenega jezika. V

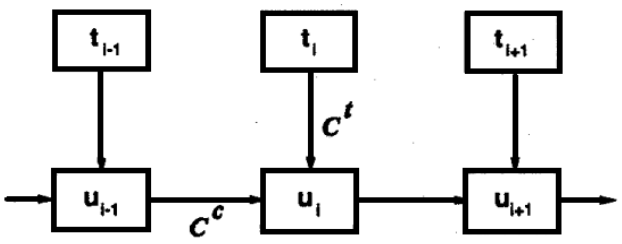
poglavju je podrobneje opisana uporaba različnih tehnik lepljenja, ki jih je dobro poznati, kadar se odločamo za sestavo novega sintetičnega glasu.

Unit selection

Unit selection ali ″izbira enot″ je metoda za izgradnjo naravno zvenečega sintetizatorja z lepljenjem enot (fonemov, difonov, itd.), tako da tvorijo želene besede. Enote je potrebno hraniti v bazi posnetkov, ki pokriva vse (ali čim več) enot.

Z večjo bazo lahko pokrijemo več karakteristik govorca, ki vplivajo na kvaliteto glasu. Ker je pri unit selection metodi ideja, da je lahko enota lepljenja poljubno velika (tudi cela poved), se lahko sintetični posnetki slišijo enako kot tisti, ki so bili posneti. V tem primeru ne bo šlo za pravo sintezo, saj bo algoritem izbral kar celo poved in jo predvajal brez dodatnega procesiranja signala. Izbira besedila, ki ga bomo izbrali za snemanje in vnos v bazo, lahko močno vpliva na kvaliteto govora. Če bodo v bazi posnetki besedil bralca pravljic, bo sintetiziran zvok povzemal njegove lastnosti govora. Tako tudi, če je napovedovalec vremena, poročil, ipd. Zmeraj pa mora na posnetkih v bazi biti glas enega samega govorca.

S $C^t(t_i, u_i)$ označimo ciljno ceno med enoto v bazi u_i in opisnim vektorjem t_i, s $C^c(u_{i-1}, u_i)$ pa ceno lepljenja dveh zaporednih enot (slika 3.7) [12]. Pri analizi besedila dobimo fonetični opis besed, ki jih želimo sintetizirati



Slika 3.7: Princip unit-selection algoritma [12].

in še druge lastnosti izgovorjave (dolžina, osnovna frekvenca, naglaševanje

itd.). Vse skupaj damo kot vhod v algoritem in označimo z $t_1^n = (t_1, ..., t_n)$. Algoritem mora v bazi poiskati zaporedje enot $u_1^n = (u_1, ..., u_n)$ z najboljšim prileganjem. To ustreza enačbi:

$$u = \min_{u \in U^n} (\sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i)) \tag{3.35}$$

Iskanje najboljših enot lahko izvedemo z Viterbijevim algoritmom [12, 15].

Sinteza z zbirko difonov

Ideja te sinteze je združevanje difonov v želene besede, tako da drugo polovico prvega difona zlepimo s prvo polovico naslednjega. Pri tem si dva sosednja difona delita skupni fonem.

Zbirka difonov določenega jeziak zajema vse možne difone, ki se lahko v tem jeziku pojavijo. Nekateri difoni se v slovenščini (in tudi v drugih jezikih) ne pojavljajo, zato jih ni potrebno dodati v zbirko. Do zbirke vseh difonov lahko pridemo z analizo daljših posnetkov, kjer si zapisujemo vse difone, ki nastopijo. Druga možnost je po pravopisnih pravilih iskanje dovoljenih zaporedij glasov in iz tega zapisovanje v zbirko (npr. dovoljena zaporedja samoglasnik/samoglasnik, samoglasnik/soglasnik, itd.). V prvem primeru lahko difone iz posnetkov ″odrežemo″, pri drugem pa jih je potrebno še posneti. Pri tem je potrebno zagotoviti dobro kvaliteto posnetkov, saj v bazi posnetkov difonov vsak nastopi le enkrat.

Če posnetke difonov dobimo izdaljših govornih posnetkov, boti ti imeli veliko kontekstno odvisnih lastnosti govora (npr. isti difon se na koncu bese lahko drugače izgovori, kot tisti na sredini). Tudi samo iskanje je zahtevno saj zahteva dobrega poslušalca in pravilna orodja, kar je časovno zahtevno. Poleg tega rezultat niso kontekstno neodvisno zveneči difoni. Zato je boljša izbira vmestitev difonov v nesmiselne besede, ki zmanjšajo vpliv sosednjih glasov.

3.6.3 Sinteza z omejeno domeno govora

Kadar imamo omejeno domeno besed (ang. *limited domain synthesis*), ki jih sistem potrebuje za svoje delovanje, je najlažji način za sintezo novega način posneti vse besede, ki se lahko pojavijo in jih nato skupaj povezati v želeno poved. Primer tega bi lahko bil ženski glas na železniških postajah, ki napoveduje odhod in prihod vlakov na posameznih peronih: "Vlak Ljubljana-Zagreb prihaja na peron številka pet."Iz primera se vidi, da je domena možnih kombinacij relativno majhna. V tem primeru je potrebno posneti številke vseh peronov in kraje iz katerih vlaki odhajajo in prihajajo. Ostali deli stavka se ne spreminjajo. Tak sistem je enostavno in hitro zgraditi, ker ni dodatne obdelave posnetkov, kvaliteta govora pa je taka, da poslušalec ne more ločiti ali gre za sintetični ali pravi govor.

Poglavje 4

Izvedba

Za preverjanje postopka izgradnje sistema za sintezo govora, sem implementiral dva sintetizatorja za slovenski jezik. Prvi je osnovan na bazi difonov, drugi pa uporablja unit selection metodo. Pri delu sem si pomagal z *Festival Speech Synthesis System* in *Festvox*, ki sta spodaj tudi opisana.

4.1 Festival Speech Synthesis System

Festival je odprtokodno okolje za izgradnjo sintetičnih jezikov. Avtor je Univerza v Edinburghu [8]. Festvox je projekt univerze Carnegie Mellon University, ki olajša izdelavo sistemov za sintezo govora [21, 14]. Vsebuje primere nekaterih že grajenih jezikov, dokumentacijo in napotke za gradnjo novih sistemov. Namen Festvoxa je olajšanje uporabe Festivala in Edinburgh Speech Tools Library, ki vsebuje implementacije nekaterih algoritmov v jeziku C++ in so uporabni pri sintezi govora. Za gradnjo jezikov uporabimo jezik Scheme z interpreterjem, ki je vgrajen v Festival. Namen orodij je gradnja jezikov na dogovorjen način (uporaba enakih datotek, dogovorjena imena datotek, ipd.), kar omogoča pregled dela drugih in lažje odpravljanje napak.

Kot je opisano v [9], je za izdelavo jezika v Festivalu potrebno:

- zgraditi osnovne datoteke (.scm),

- zgraditi shemo difonov,
- posneti govorca,
- označiti posnetke,
- določiti osnovno periodo in LPC,
- test št. 1,
- vstaviti besede v leksikon in napisati LTS pravila,
- analizirati besedilo,
- nastaviti prosodične lastnosti (trajanje, intonacija),
- test št. 2,
- ustvarjen govor shraniti v obliko za distribucijo.

Postopki so jasno opisani, vendar prihaja do večjih razlik v odvisnosti od jezika, s katerim se ukvarjamo, zato je tukaj na mestu znanje iz glasoslovja.

4.1.1 Scheme programski jezik

V poglavju je za lažje razumevanje na kratko opisan programski jezik Scheme, ki ga je za uporabo Festivala potrebno znati na vseh korakih sinteze govora. Je splošno-namenski funkcijski jezik, dialekt jezika Lisp [10, 16]. Nastal je leta 1975 v laboratoriju za umetno inteligenco univerze MIT v ZDA in je avtorjem služil kot orodje za njihove raziskave in za učenje na univerzi. Kot skriptni jezik Festivala so avtorji izbrali Scheme ker :

- računalniku je enostaven za intepretertiranje,
- omgoča samodejno sproščanje pomnilnika (ang. *garbage collection*),
- ima vse potrebne osnovne strukture,
- enostaven je za uporabo v vgrajenih sistemih (GNU Project, Gimp, App Inventor for Android in še mnogi ga uporabljajo kot skriptni jezik),
- avtorjem je bil znan.

Z izborom jezika, kjer potrebujemo le interpreter, se pokaže uporabnost sistema. To omogoča spreminjanje nastavitev, če ugotovimo, da smo se kje zmotili in pri tem ni potrebno celotnega programa ponovno prevajati. Tudi nadgradnje sistema so enostavne. Festival vsebuje svoj interpreter za Scheme, tako da ni potrebna uporaba zunanjih interpreterjev (Dr. Scheme, Chicken,

Kawa, ...) in po namestitvi omogoča takojšnjo uporabo [9].

4.2 Izdelava sintetizatorjev

V poglavju sta opisana sintetizatorja z bazo difonov in unit selection metodo, s katerima sem zgradil sintetična govorca. Obe imata veliko skupnega, zato je nekaj poglavij ločenih. V stvareh, ki se razlikujejo pa je to opisano. Leksikon, pravila za branje števil, LTS pravila so pri obeh izvedeni na enak način.

4.2.1 Leksikon in branje posebnih zapisov

Ob izbiri jezika v festivalu se naloži tudi pripadajoči leksikon. Ta je v datoteki *festvox/fri_si_lambe_lexicon.scm*. Dodajanje izgovorjav za besede in druge znake je enostavno, potrebno je poznati le fonetični zapis besed v slovarju.

Leksikon

Primer slovarja z nekaj vnosi črk abecede, števil, znakov in besedo *megla* je podan v kodi 4.1. Dodajanje izvršimo z ukazom *lex.add.entry ([zapis] [tip] [izgovorjava])*. Na ta način je možno dodati poljubno mnogo vnosov in pokriti (vse) besede, ki želimo, da bi jih sistem znal prebrati. Nujno potrebno pa je pri posebnih simbolih kot je npr. "/"(deljeno), kajti pomen je vsem znan, ni pa samoumeven. Izgovorjavo besed zapišemo tudi po zlogih in naglasnih točkah. Festival pri branju besedila zmeraj pogleda ali pozna izgovorjavo besede v leksikonu. Sicer se odloči za LTS. V leksikon sem dodal izgovorjavo črk abecede, števil od 0 do 9, izgovorjavo ločil in nekaterih besed (le za preverjanje delovanja). Ostale besede sistem bere po LTS, razen pri unit selection metodi, kjer lahko izgovorjavo najde v posnetkih.

```
1 (lex.add.entry '("a" nn (((a) 0))))
2 (lex.add.entry '("b" nn (((b ) 0))))
3 (lex.add.entry '("c" nn (((c) 0))))
4 (lex.add.entry '("1" nn (((e n a) 0))))
```

```
(lex.add.entry '("2" nn (((d v a) 0))))
(lex.add.entry '("3" nn (((t r i) 0))))
(lex.add.entry  '("/" n ( ((d e l) 0) ((j e) 0) ((n o) 0) )))
(lex.add.entry  '("megla" n ( ((m e@ g) 0) (( l a) 1) )))
(lex.add.entry '("." nn (((p i) 1) ((k a) 0))))
```

Koda 4.1: Del leksikona v katerem je fonetična izgovorjava besed ločena po zlogih.

LTS

Spodaj so zapisana pravila za izgovorjavo besed LTS (*letter-to-sound-rules*). Ta pravila se uporabijo, ko sistem ne najde naučenih pravil za izgovorjavo ali pa zapisa v leksikonu. Pravila so vzeta iz slovenskega pravopisa, kot je opisano v poglavju 2.1.1.

```
(lts.ruleset
  fri_si
  (
   (VOCAL a e i o u)
   (ALL b c x d f g h j k l m n p r s y t v z q))
  (
   ( [ a ] = a )
   ( [ e ] = e )
   ( [ i ] = i )
   ( [ o ] = o )
   ( [ u ] = u )
   ( [ b ] = b )
   ( [ c ] = th )
   ( [ x ] = ch )
   ( [ d ] = d )
   ( [ f ] = f )
   ( [ g ] = g )
   ( [ h ] = h )
   ( [ j ] = j )
   ( [ k ] = k )
```

```
21     ( # [ l ] = l )
22     ( [ m ] = m )
23     ( [ n ] = n )
24     ( [ p ] = p )
25     ( [ r ] = r )
26     ( [ s ] = s )
27     ( [ y ] = sh )
28     ( [ t ] = t )
29     ( [ v ] = v )
30     ( [ z ] = z )
31     ( [ q ] = zh )
32     ( VOCAL [ l ] # = u )
33     ( VOCAL [ l ] = l )
34     ( ALL [ l ] = l )
35 ))
```

Koda 4.2: LTS pravila.

Branje števil

Za izgovorjavo števil poskrbi koda v *festvox/fri_si_lambe_tokenizer.scm*. Ker je kode veliko, je spodaj prikazana le koda za branje števil dolžine 3. Pri teh nastopata dve izjemi - pri številih, ki se začnejo z 1 in 2. Zato sta ta dva primera posebej obravnavana. Pri ostalih številih je rezultat sestavljen iz besede, ki predstavlja prvo števko skupaj z besedo *"sto"*, za tem sledi rekurzivni klic funkcije, ki vrne besedo za izgovorjavo števil med 1 in 99. Na analogen način zna sistem pravilno prebrati števila od 1 do 999.999.999.

```
1  ((equal? len 3);; dolzina 3
2    (cond
3      ((string-equal (car digits) "1");; 1xx
4       (cons "sto" (dig_to_num (cdr digits))))
5      ((string-equal (car digits) "2");; 2xx
6       (cons "dvesto" (dig_to_num (cdr digits))))
7      (t;; 300+
8       (append (dig_to_num (list (car digits)))
9       (list "sto") (dig_to_num (cdr digits)))
10 )))
```

Koda 4.3: Koda za branje števil dolžine 3.

Branje časa

Za zapise časa v obliki *HH:MM:SS* poskrbi koda:

```
;;; Voice/si token_to_word rules
(define (fri_si_lambe::token_to_words token name)
  "(fri_si_lambe::token_to_words token name)
Specific token to word rules for the voice fri_si_lambe.  Returns a list
of words that expand given token with name."
  (cond
   ((string-matches name "[1-9][0-9]+")
    (slo_number name))     ;branje stevil

((string-matches name "[0-9][0-9]:[0-9][0-9]:[0-9][0-9]")  ;; branje casa
   (append (slo_number (remove_leading_zeros (string-before name ":"))) ;HH
   (list "ur")
   (slo_number (remove_leading_zeros (string-before (string-after name ":")
       ":")))  ;MM
   (list "minut" "in" )
   (slo_number (remove_leading_zeros (string-after (string-after name ":") "
       :"))) ;SS
   (list "sekund")

))
   (t ;; when no specific rules apply do the general ones
    (list name))))
```

Koda 4.4: Koda za branje časa.

Branje e-naslovov

V besedilih pogosto najdemo e-poštne naslove, zato jih je dobro znati pravilno prebrati. Spodnja koda zna prebrati klasične e-naslove kot je npr.: *ime.priimek@ustanova.domena.si*. Uporabljeni so regularni izrazi in v Festivalu vgrajene funkcije (*string-matches, string-before, string-after*), ki olajšajo delo z nizi ([8]).

```
(define
   (email name)
      (cond
         ((string-matches (string-before name "@") ".*[.].*") ;; pred @ in
             piko
            (append (list (string-before name "."))
               (list "pika")
               (email (string-after name "."))))

         ((string-matches name ".*@.*")   ;; pred @ in @
            (append (list (string-before name  "@"))
               (list "afna")
               (email (string-after name "@"))))

         ((string-matches name ".*[.].*") ;; po @
            (append (list (string-before name "."))
               (list "pika")
               (email (string-after name "."))))
         (t
         (list name)
         )
))
```

Koda 4.5: Koda za branje e-poštnih naslovov

Branje okrajšav letnic

Spodnja koda pretvori zapise kot so npr.*v 80-ih letih* v *v osemdesetih letih* in jih nato prebere.

```
((string-matches name "[2-9][0]ih")
   (append (slo_number (string-before name "i"))
      (list "ih")))

((string-matches name "[2-9][0]-ih")
   (append (slo_number (string-before name "-"))
      (list "ih")))
```

Koda 4.6: Koda za branje posebnih zapisov letnic

4.2.2 Označevanje glasov

Pri označevanju glasov sem uporabil ročno označevanje s pomočjo EMU orodja in avtomatično označevanje s pomočjo DTW algoritma. Rezultat označevanja so *.lab* datoteke v mapi *lab*. Z avtomatičnim označevanjem dobimo dobre rezultate, a včasih pride do napak, zato je dobro rezultate ročno pregledati in po potrebi ročno popraviti. Primer napake pri avtomatičnem označevanju lahko nastopi, kadar nismo utegnili posneti govora v času, ki smo ga imeli na voljo za snemanje določene besede. Ta je, kot že omenjeno, pogojen z dolžino pomožne datoteke iz drugega jezika, ki jo uporablja DTW za primerjanje.

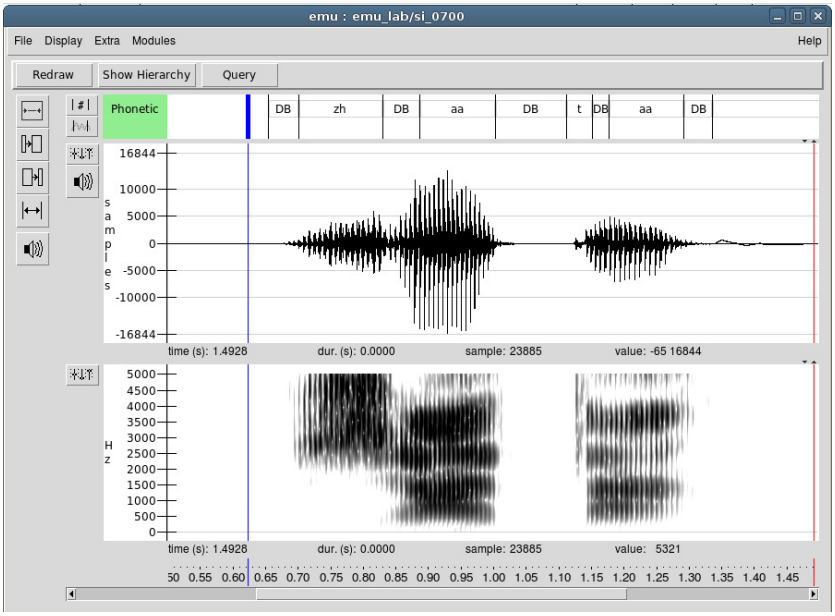
```
1  separator ;
2  nfields 1
3  #
4      0.72000      26 #
5      0.78500      26 DB
6      0.88000      26 zh
7      0.98500      26 DB
8      1.06500      26 aa
9      1.11000      26 DB
10     1.17500      26 t
11     1.21000      26 DB
12     1.36500      26 aa
13     1.41500      26 DB
14     2.00000      26 #
```

Koda 4.7: Primer zapisa označevalne datoteke (si_0624.lab) s časom začetka in konca pojava fonemov.

Ročno označevanje s pomočjo EMU

EMU Labeller je orodje za označevanje fonemov v zvočnih datotekah. Označujemo začetek in konec trajanja izgovorjave fonema in tudi tišino. Oznako DB pa vmestimo tam, kjer želimo eksplicitne meje med fonemi [9]. Pri tem si pomagamo s spektrom posnetka (slika 4.1). Pri tihih delih je na

spektrogramu belo, sicer pa črno. Na vrhu so meje fonemov, ki jih dobimo z avtomatičnim označevanjem, po potrebi pa jih premaknemo. Pomagamo si tudi s poslušanjem označenega fonema in lahko slišimo, če so meje pravilno postavljene. Velikokrat se zgodi, da meja zajema še nepotrebni tihi del na koncu besede ali pa naslednji fonem.



Slika 4.1: Označevanje fonemov nesmiselne besede ″žata″ s programom EMU Labeller.

Avtomatično označevanje

Pri avtomatičnem označevanju je uporabljen DTW algoritem predstavljen v poglavju 3.2.3. Uporabi se tudi klasifikacija fonemov, saj pri zapornikih je na začetku glas zelo nestabilen in je zato boljše označiti fonem eno tretjino od začetka, kar zajame še tihi del zapornika. Kadar po fonemu nastopi tišina (konec povedi, besede ali daljši premor), je dobro označiti fonem na četrtini njegovega trajanja. Fonemi pred tišino imajo manj energije kakor fonemi znotraj besed. Če bi stabilna točka bila izbrana kasneje, bi prišlo do večje razlike v energiji kadar bi združili dva difona in bi to tudi slišali. Za vse

ostale difone je stabilna točka na sredini fonema[9]. Teh pravil sem se držal tudi pri ročnem označevanju.

4.2.3 Snemanje

Pri snemanju je bilo potrebno zagotoviti čim boljše snemalne pogoje. Zaželeno je, da je čim manj šuma v prostoru kjer snemamo. Zaradi tega sem snemal z mikrofonom (Logitech h150) z naglavnimi slušalkami in prenosnim računalnikom, ker so navadno tišji od stacionarnih. To je pomembno pri snemanju, mikrofon pa dobro izniči šum. Naglavne slušalke z mikrofonom so pri snemanju zelo na mestu, saj je mikrofon zmeraj na enaki oddaljenosti od ust, kar je potrebno zagotoviti, da je amplituda posnetkov karseda enaka. Pri snemanju zvoka v prostoru, kjer je stacionarni računalnik, se kaj hitro posname tudi šum, ki izhaja iz ventilatorjev in okolice in poslabša kvaliteto posnetkov. Dobri posnetki pa so ključnega pomena za vse nadaljne korake in se je vredno potruditi pri tem koraku. Za snemanje sem uporabil program *na_record* iz zbirke Speech Tools. Vse posnetke posnamemo enako dolgo zaradi ujemanja z že znanimi posnetki, ki jih uporabljamo pri označevanju fonemov z DTW - boljši rezultati. V ta namen programu *na_record* nastavimo dolžino datoteke, ki jo bomo posneli. Te so dolge okoli 2 sekundi za bazo difonov.

4.2.4 Klasifikacija fonemov SKJ

V seznamu fonemov (*/festvox/fri_si_lambe_phoneset.scm*) sem definiral uporabljene foneme SKJ po klasifikaciji v poglavju 2.1.

```
52 (defPhoneSet
53   fri_si
54   ;;;  Phone Features
55   (
56    (vc + -) ;; vowel / consonant
57    (vlng s l d a 0);;vowel length:short long dipthong schwa
58    (vheight 1 2 3 0 -);;vowel height:high mid low
59    (vfront 1 2 3 0 -);;vowel frontness: front mid back
60    (vrnd + - 0);;lip rounding
```

```
61    (ctype s f a n l 0);;stop fricative affricative nasal liquid
62    (cplace l a p b d v 0);;labial alveolar palatal labio-dental  dental
          velar
63    (cvox + - 0);;voiced/unvoiced
64    )
65   (
66    (#  - 0 - - - 0 0 -)
67    (a  + l 3 1 - 0 0 -)
68    (e  + l 2 1 - 0 0 -)
69    (i  + l 1 1 - 0 0 -)
70    (o  + l 3 3 - 0 0 -)
71    (u  + l 1 3 + 0 0 -)
72    (ee + l 1 2 + 0 0 -)
73    (@@ + s 2 2 + 0 0 -);; polglasnik
74    (oo + l 1 2 + 0 0 -)
75    (p  - 0 - - - s l -);; zaporniki stop
76    (b  - 0 - - - s l +)
77    (t  - 0 - - - s d -)
78    (d  - 0 - - - s d +)
79    (k  - 0 - - - s v -)
80    (g  - 0 - - - s v +)
81    (f  - 0 - - + f b -);; priporniki fricative
82    (s  - 0 - - + f d -)
83    (z  - 0 - - + f d +)
84    (sh - 0 - - + f p -)
85    (zh - 0 - - + f p +)
86    (h  - 0 - - + f v -)
87    (v  - 0 - - + f b +) ;; zvocnik frikativ labiodental
88    (th  - 0 - - + a d -) ;; fonem "c", zlitniki africative
89    (ch - 0 - - + a p -)
90    (dz - 0 - - + a p +)
91    (m  - 0 - - + n l +);; zvocniki nosniki nasal
92    (n  - 0 - - + n d +)
93    (r  - 0 - - + l d +);; jezicniki
94    (l  - 0 - - + l d +)
95    (y  - 0 - - + l p +);; fonem "j" drnsniki
96   )
97 )
98 (PhoneSet.silences '(#))
```

Koda 4.8: Klasifikacija fonemov.

4.2.5 Sintetizator z bazo difonov

Gradnja seznama difonov

Obstajajo difoni, ki v SKJ ne obstajajo in jih zato ni potrebno posneti, saj nikoli ne nastopijo pri izgovorjavi besed, vendar sem za poenostavitev problema v seznam difonov dodal vse kombinacije fonemov brez fonemov *đ*, širokega e in o in z glasom za tišino. Teh je 651. Skripta, ki poskrbi za grandjo seznama je *fri_si_lambe_diphone/festvox/si_schema.scm*. Seznam je v datoteki *fri_si_lambe_diphone/etc/sidiph.list*.

```
1 ( si_0001 "# t aa b a b aa #" ("b-a" "a-b") )
2 ( si_0002 "# t aa th a th aa #" ("th-a" "a-th") )
3 ( si_0003 "# t aa ch a ch aa #" ("ch-a" "a-ch") )
4 ( si_0599 "# a t aa #" ("#-a") )
5 ( si_0645 "# t aa t a #" ("a-#") )
```

Koda 4.9: Primer vnosa difona ″ba″ in ″ab″ v seznam.

Fonemi v besedah imajo, v odvisnosti od pozicije kjer se nahajajo, različno jakost izgovorjave (tiste na koncu besed navadno izgovorimo z nižjo inteziteto kot tiste na začetku). Zato je pri snemanju smiselno difone vstaviti v nesmiselne besede, kot je prikazano zgoraj. Tako zmanjšamo napake pri naglaševanju in dolžini izgovorjave. To je uporabno tudi pri avtomatičnem označevanju z DTW algoritmom, saj vemo kateri glasovi nastopijo pri izgovorjavi in točno kdaj. Beseda z oznako *si_0599* je primer, kadar glas *a* nastopi za tišino (#, torej na začetku besede. Z oznako *si_0645* pa kadar glas *a* nastopi pred tišino, torej na koncu besede.

Izgradnja pomožnih datotek

Pomožne datoteke sem generiral s pomočjo že obstoječega angleškega (ameriškega) jezika *rab_diphone*. Pri tem sem moral narediti preslikavo fonemov (v datoteki si_schema.scm), saj so oznake nekaterih fonemov v tem jeziku drugačne (v angleškem jeziku npr. ni našega fonema *c*).

```
(set! fri2mrpa_map
   '((c th) ;; c
   (j y))) ;; j
```

Koda 4.10: Preslikovalna tabela fonemov iz *mrpa* seznama fonemov v Festivalu.

Snemanje

Difone v bazi (v ″nesmiselnih″ besedah - datoteka */etc/sidipih.list*) je bilo potrebno posneti, da dobimo po 1 vzorec za vsak difon.

Označevanje

Posnetke sem označeval z že opisanim postopkom za avtomatično označevanje z DTW. Pri tem so bile potrebne pomožne datoteke, ki sem jih generiral z že obstoječim ameriškim glasom *rab_diphone*, seznam difonov in lastni posnetki difonov v nesmiselnih besedah. Rezultat so označevalne datoteke v mapi */lab*.

4.2.6 Sintetizator na osnovi unit selection

Enako kot pri sintezi z bazo difonov je tukaj treba generirati pomožne datoteke iz zapisov v bazi (datoteka *etc/lambe.data*). V bazo sem dodal 200 zapisov. Vnosi v bazi so vzeti iz časopisov, raznih knjih in nekaj je tudi izmišljenih povedi. Že na tem koraku se je potrebno potruditi, saj je na ta način pokritih več kontekstno odvisnih izgovorjav besed, kar izboljša rezultate. Primer zapisov v *etc/lambe.data*:

```
( lambe_0001 "nikola tesla je razsvetlil in elektrificiral svet." )
( lambe_0003 "genij, xudak, xarovnik in xlovekoljub." )
( lambe_0005 "danes je lep dan." )
```

Izgradnja pomožnih datotek

Pomožne datoteke sem generial iz jezika *fri_si_lambe_diphone* Pomožne datoteke so:
.lab v *fri_si_lambe_clunits/prompt-lab* z oznakami fonemov
.wav v *fri_si_lambe_clunits/prompt-wav* s pomožnimi posnetki za označevanje
.utt v *fri_si_lambe_clunits/prompt-utt* z informacijo o naglaševanju in trajanju posnetkov
.cep v *fri_si_lambe_clunits/prompt-cep* z MFCC koeficienti

Snemanje

Zapise v bazi *etc/lambe.data* sem posnel enako kot pri bazi difonov. Snemanje se tu razlikuje v tem, da so v bazi smiselne povedi in ne posamezne besede.

Označevanje

Označevanje posnetkov sem, kakor pri bazi difonov, opravil najprej z uporabo DTW algoritma, kasneje pa posnetke ročno pregledal z EMU labeler in po potrebi popravil.

Določanje periode tonov

Ker se F_0 razlikuje v odvisnosti od govorca, sem moral pravilno upoštevati meje pasovno zapornih filtrov. Ker je F_0 mojega govora nižja od govora ženske ali otroka, sem filtriral le frekvence med 80 in 140 Hz. To pomeni, da je najdaljša dovoljena perioda tona T_0 ((3.1)) 0,0125 s, najkrajša pa 0,0071 s. Pri določanju periode tonov je potrebno upoštevati, da nezvočniki nimajo osnovne periode, zato je v primeru nezvočnikov potrebno ustaviti privzeto vrednost periode. To storimo, ker potrebujemo oznake kdaj računati MFCC. Če bi MFCC računali na enakih časovnih intervalih, to ne bi bilo potrebno. Informacija o periodi tonov se nahaja v mapi */pm*. Primer datoteke *lambe_0001.pm* z oznako periode:

```
1 EST_File Track
2 DataType ascii
3 NumFrames 510
4 NumChannels 0
5 NumAuxChannels 0
6 EqualSpace 0
7 BreaksPresent true
8 EST_Header_End
9 0.010125 1
10 0.020250 1
11 0.030375 1
12 0.040500 1
13 0.050625 1
14 0.060750 1
```

Tudi te je možno ročno preveriti in popraviti s pomočjo programa EMU Labeller, a je to časovno potratno in je bolje imeti dober avtomatični označevalnik.

Izračun MFCC

MFCC koeficiente lahko izračunamo na enakih časovnih intervalih, boljša rešitev pa je računanje ob nastopu tonske periode [9]. Izbrani parametri za izračun MFCC:

- Faktor poudarjanja višjih frekvenc (enačba 3.5): 0.97,
- število trikotnih filtrov po mel frekvencah(slika 3.1): 24,
- število MFCC: 12,
- uporabljena okenska funkcija: Hammingovo okno (enačba 3.6).

Okvirji ne bodo zmeraj enako dolgi. Pri nezvočnikih so dolgi 0.01 sekunde. Za boljšo frekvenčno predstavitev (signal ni periodičen) lahko dolžino okvirja povečamo, tako da se prekriva s sosednjimi [19]. Dolžino okvirjev sem pomnožil z 2.5 in pri nezvočnikih z 2.0.

Izgranja sintetizatorja

Za izgradnjo sintetizatorja poskrbi Festival in je avtomatična. Po tem koraku lahko preizkusimo zvok. Lahko tudi pripravimo verzijo za distribucijo (ustvarimo *.group* datoteko), vendar tega nisem storil, saj ni potrebno.

4.3 Preizkus delovanja

4.3.1 Festival

Za preizkus delovanja jezikov, se da uporabiti Festival (*festival/src/main/festival.exe* na Windows OS, oz. *festival/src/main/festival* na Linux OS). Odpre se ukazna vrstica, kjer pišemo ukaze v jeziku Scheme (slika 4.3.1). Ob zagonu se naloži privzeti jezik, ki ga definiramo v datoteki *festival/lib/voices.scm*. To je jezik *kal_diphone*, lahko pa nastavimo poljubni privzeti jezik (npr. *fri_si_lambe_diphone*). Jezik lahko izberemo z ukazom *(<IME_JEZIKA>)*.

Slika 4.2: Ukazna vrstica orodja Festival

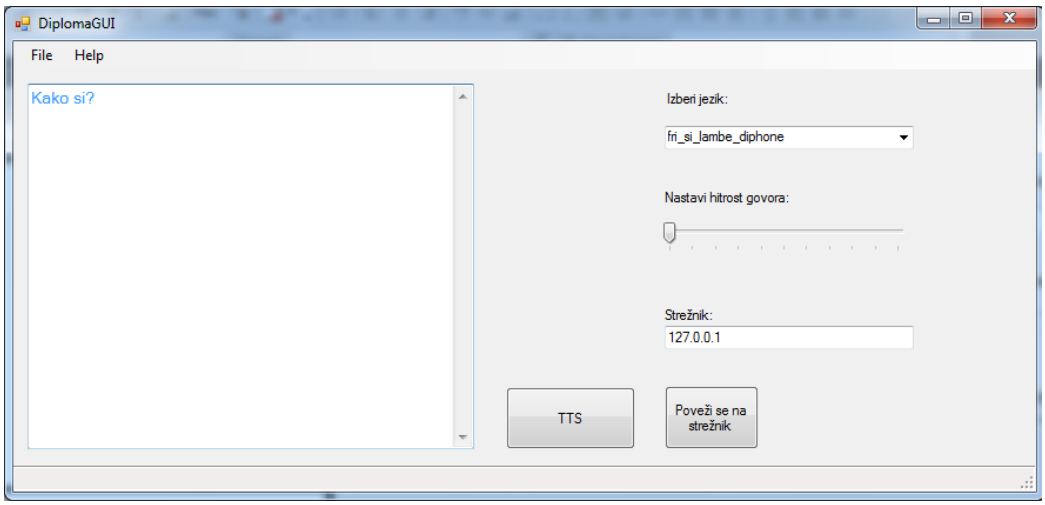
4.3.2 GUI

Za lažjo uporabo sintetizatorja izdelanega v okolju Festival, sem izdelal grafični programski vmesnik (slika 4.3) v jeziku C++ (deluje v Windows OS), ki

poenostavi izbiro jezikov prisotnih v Festivalu in klicanje osnovnih funkcij kot so *SayText* za izgovorjavo povedi [8].

Program je uporaben tudi takrat kadar želimo večkrat uporabiti Festival, kajti omogoča *"odjemalec-strežnik"* komunikacijo. Ta je potrebna zato, da ne zaganjamo Festivala zmeraj kadar bi želeli prebrati besedilo, ki ga je mogoče dati kot argument pri zaganjanju programa v obliki datoteke ipd. Program poskrbi za zagon odjemalca v ozadju. Potrebno je zagnati tudi strežnik (to storimo z ukazom *festival.exe –server [IP]*, s katerim bo komuniciral odjemalec. Strežnik je lahko tudi na oddaljenem računalniku (privzeto na 127.0.0.1) in omogoča povezavo preko LAN omrežja oz. interneta. Po uspešni povezavi je pošiljanje ukazov Festivalu hitro in ni potrebno čakati na njegov zagon, ki traja približno 2-5 sekund in pri krajših povedih slišimo rezultat v manj kot sekundi.

Preko GUI lahko izberemo vse jezike, ki so na voljo v okolju Festival in nastavimo hitrost govora (slika 4.3). Spodaj (koda 4.11) je prikazana koda



Slika 4.3: GUI s katerim je enostavneje preveriti delovanje novega jezika.

izvedbe povezave med GUI, strežnikom in odjemalcem.

```
1  namespace diploma {
2     FILE *pPipe; // pipe to stdin of festival_client.exe
3     public ref class Form1 : public System::Windows::Forms::Form{
4     public:  Form1(void){
5        InitializeComponent();
6        pPipe =_popen("c:\\festival\\festival\\src\\main\\festival_client.exe"
             , "wt");
7     }
8     //TTS button - Text to Speech
9     private: System::Void button1_Click(System::Object^ sender, System::
          EventArgs^ e){
10       String^ odg = gcnew String(textBox2->Text);
11       fprintf(pPipe, "%s", "(SayText \""+odg+"\")");
12       fflush(pPipe);
13    }
14    //Trackbar change - duration change
15    private: System::Void trackBar1_Scroll(System::Object^ sender, System::
          EventArgs^ e){
16       fprintf(pPipe, "%s %f %s", "(Parameter.set 'Duration_Stretch ",(
             trackBar1->Value/10.0),")");
17       fflush(pPipe);
18    }
19    //Combobox change - language change
20    private: Syste::Void comboBox1_SelectedIndexChanged(System::Object^
          sender, System::EventArgs^ e){
21       fprintf(pPipe, "%s", "("+comboBox1->Text")");
22       fflush(pPipe);
23    }
24 }}
```

Koda 4.11: *Koda izvedbe GUI - pisanje na standardni vhod odjemalca. Odjemalec se poveže na strežnik (privzeto 127.0.0.1) in mu pošlje ukaz. Prikazana je tudi koda za spreminjanje hitrosti izgovorjave in jezikov.*

Poglavje 5

Rezultati

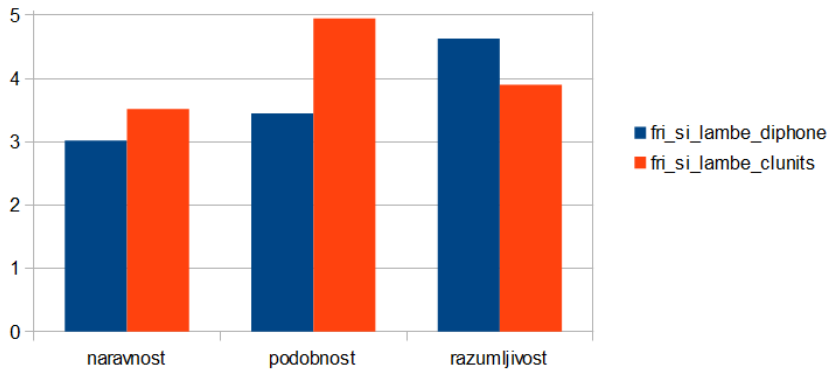
Rezultat diplomske naloge sta dva jezika - *fri_si_lambe_diphone* in *fri_si_lambe_clunits*.

5.1 Primerjava implementacij

Najboljšo oceno kvalitete govora lahko dobimo od poslušalcev. Damo jim poslušati daljše besedilo. Poslušalec pri tem lahko ocenjuje število besed, ki jih je znal prepoznati in kako naravno zveni govor. Lahko se zgodi, da se sintetični govor sliši povsem ″robotsko″, a vseeno prepoznamo vse ali večino besed. Lahko pa se zgodi, da je govor naravno zveneč, ne moremo pa prepoznati vseh besed zaradi raznih napak sintetizatorja (naglaševanje, hitrost izgovorjave ipd.). Spodaj je opisan izvedeni MOS test in njegovi rezultati.

5.2 MOS test

MOS (en. *Mean Opinion Score*) test, je test s katerim poslušalci lahko ocenijo kvaliteto govora sintetizatorja [17]. Poslušalci lahko dajo oceno od 1 do 5, kjer je 5 najboljša ocena. V našem primeru 5 pomeni, da ni poslušalec ne more ugotoviti, da gre za sintetični govor. Z oceno 1 je govor zelo moteč in ne



Slika 5.1: Rezultati MOS testov.

razumljiv (napake pri izgovorjavi besed, napake pri naglaševanju, izpuščeni glasovi, ipd.). Ocenjujemo lahko tudi podobnost med posnetki, ki so v bazi in so bili uporabljeni za izdelavo sintetičnih govor in med sintetičnimi govori istih besed. Poslušalec s 5 oceni govor, ki se mu zdi povsem enak originalnemu, z 1 pa tistega, kjer ni mogoče prepoznati govorca (sliši se, kot da bi drug govoril).

Pri vseh treh testih sem dal desetim poslušalcem, ki jim je slovenščina materni jezik, poslušati isto besedilo. Ocenjevali so naravnost, razumljivost in podobnost z orignialnim zvokom. Vse povedi so bile naključno izbrane in niso vsebovane v bazi (*etc/lambe.data*).

Test naravnosti

Pri testu naravnosti sem dal poslušati sintetične povedi obeh sintetizatorjev. Izkaže se da jezik *fri_si_lambe_clunits* zveni bolj naravno (ocena 3,52). To je bilo tudi pričakovati, saj je manj obdelave signalov kot pri sintezi z difoni (ocena 3,02).

Test podobnosti

Test podobnosti meri podobnost z originalnimi posnetki iz baze. *Unit-selection* sinteza je pričakovano dobila oceno 4,95, saj razlik skoraj ni čutiti.

Pri sintezi z difoni je podobnost precej manjša (ocena 3,45).

Test razumljivosti

Jezik *fri_si_lambe_diphone* je kljub slabši podobnosti z mojim originalnim glasom in manjši naravnosti boljše razumljiv (4,63 proti 3,9).

Uporabljena MOS lestvica

Rezultati na sliki 5.1 upoštevajo spodnjo MOS lestvico.

Ocena	Opis
5	Odličen zvok, zelo naraven, visoka podobnost
4	Skoraj naraven, majhno popačenje
3	Delno naraven, opazno popačenje, ne moteče
2	Nizka stopnja naravnosti, moteč govor
1	Nenaraven, visoka stopnja popačenja, neprepoznavnost govorca

Tabela 5.1: MOS lestvica.

Poglavje 6

Zaključek

Izdelati sistem za sintezo govora je enostavna stvar. Podrobnosti pa so tisto, kar loči enostaven sistem, ki se ne zna soočati z vsemi možnimi vhodi - to je branje bolj kompleksnih tekstov, pravilno naglaševanje, pravilna dolžina izgovorjenih besed in ostale značilnosti naravnozvenečih govorov, od kompleksnejšega.

Nadgradnja sistema

Izdelana sistema se da nadgraditi na nivoju interpretiranja ukazov za boljše izrabljanje že izdelanega sistema in na nivoju procesiranja signalov. V prvem primeru je to npr.:

- branje števil dolžine 10 ali več,
- branje decimalnih števil,
- branje kompleksnih matematičnih izrazov,
- branje e-pošte,
- branje rimskih števil,
- povečanje slovarja,
- itd.

Na nivoju obdelave signalov in snemanja je to lahko:

- bolj učinkotivo označevanje - uporaba obstoječih sistemov za razpoznavanje govora (npr. Sphinx),

- uporaba HMM modelov,
- uporaba modelov za napovedovanje naglaševanja,
- uporaba boljšega mikrofona in boljšega bralca,
- uporaba EGG (en. *electroglottograph*) za določanje F_0.

Literatura

[1] W. V. Kempelen, *Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine*, Dunaj, 1791, Skenirana knjiga dostopna na:

`http://drl.ohsu.edu/cdm/compoundobject/hom/id/3031`

[2] The Cochlea, *"Basilar Membrane"*, dostopno na:

`http://147.162.36.50/cochlea/cochleapages/theory/bm/bm.html`

[3] ZRC SAZU, ISJ Frana Ramovša, *"Slovenski pravopis"*, Ljubljana 2010, Slovenska akademija znanosti in umetnosti

[4] L. R. Rabiner, R. W. Schafer, *"Digital Processing of Speech Signals"*, September 1978, Prentice Hall, ISBN-10 0132136031

[5] J. L. Flanagan, J. B. Allen, M. A. Hasegawa-Johnson, *"Speech Analysis Synthsis and Perception"*, 2008

[6] A. V. Bhalla, S. Khaparkar, *"Performance Improvement of Speaker Recognition System"*, Marec 2012, Gyan Ganga College of Technology, Indija

[7] P. Taylor, *"Text-to-Speech Synthesis"*, Cambridge University, 2007

[8] A. W. Black, P. Taylor, R. Caley, *"The Festival Speech Synthesis System - System Documentation, Edition 1.4"*

[9] A. W Black, K. A. Lenzo, *"Building Synthetic Voices"*, Language Technologies Institute, Carnegie Mellon University, Dostopno na:

`http://festvox.org/bsv`

[10] G. L. Steele Jr., G. J. Sussman, *"The Revised Report on SCHEME: A Dialect of LISP"*, Januar 1978, Massachusetts Institute of Technology, Artificial Intelligence Laboratory

[11] F. Malfrere, T. Dutoit, *"High-Quality Speech Synthesis For Phonetic Speech Segmentation"*, Circuits Theory and Signal Processing Lab, Faculte Polytechnique de Mons

[12] A. J. Hunt, A. W. Black, *"Unit selection in a concatenative speech synthesis system using a large speech database"*, 1996, ATR Interpreting Telecommunications Research Labs, Japan

[13] VIB Departmend of Plant Systems Biology, *"DTW algorithm"*, Ghent University, Belgium, dostopno na:

`http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm`

[14] A. W. Black, *"Speech Synthesis in Festival, A practical course on making computers talk, Edition 2.0"*, May 2000, University of Edinburgh, Carnegie Mellon University

[15] A. W. Black, N. Campbell *"Optimising selection of units from speech databases for concatenative synthesis"*, ATR Interpreting Telecommunications Reserach Labs, Japan

[16] R. Kent Dybvig, *"The Scheme Programming Language, Fourth Edition"*, The MIT Press, ISBN 978-0-262-51298-5

[17] Wikipedia, *"Mean opinion score"*, dostopno na:

`http://en.wikipedia.org/wiki/Mean_opinion_score`

[18] Wikipadia, *"Fundamental frequency"*, dostopno na:

`http://en.wikipedia.org/wiki/Fundamendal_frequency`

[19] N. Sen, T. Basu, S. Chakroborty, *"Comparison of Features Extracted Using Time-Frequency and Frequency-Time Analysis Approach for Text-Independent Speaker Identification"* Signal Processing Research Group, CET IIT Kharagpur, India; Samsung Bangalore, India

[20] Wikipedia, *"Linear-predictive coding"*, dostopno na:

`http://en.wikipedia.org/wiki/Linear_predictive_coding`

[21] CMU Speech Group, *"Festvox"*, Carnegie Mellon University, dostopno na:

`http://festvox.org`