

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

**Izračun prosječnih osnovnih frekvencija zvučnih glasova
koristeći zvučne zapise iz VEPRAD baze podataka**

Rijeka, srpanj 2015

Dominik Beževan
0069060241

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

**Izračun prosječnih osnovnih frekvencija zvučnih glasova
koristeći zvučne zapise iz VEPRAD baze podataka**

Mentor: dr. sc. Ivo Ipšić

Rijeka, srpanj 2015

Dominik Beževan
0069060241

IZJAVA

Izjavljujem da sam ovaj Završni rad na temu „Osnovna frekvencija zvučnih glasova“ izradio samostalno korištenjem stečenih znanja, vještina i potrebne literature.

Dominik Beževan

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Povjerenstvo za završne ispite preddiplomskog sveučilišnog studija računarstva

Preddiplomski sveučilišni studij računarstva

Klasa: 602-04/15-11/39

Ur. br.: 2170-15-14-15-1

Rijeka, 12.3.2015.

Z A D A T A K

za završni rad

Pristupnik: **Dominik Beževan**

JMBAG: 0069060241

Lok. mat. br.: 12300043

Naslov zadatka: **OSNOVNA FREKVENCIJA ZVUČNIH GLASOVA**

Thesis title: **FUNDAMENTAL FREQUENCY OF VOICED SOUNDS**

Sadržaj zadatka: Pomoću funkcija knjižnice SPTK realizirajte postupak određivanja osnovne frekvencije zvučnih glasova u signalima govora. Prikažite prosječne vrijednosti osnovne frekvencije zvučnih glasova za signale iz baze VEPRAD.

Zadano: 12.3.2015.

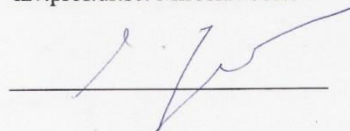
Mentor

prof.dr.sc. Ivo Ipšić

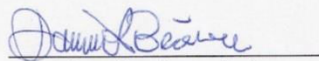


Predsjednik Povjerenstva

izv.prof.dr.sc. Miroslav Joler



Zadatak preuzeo dana: 16.3.2015.


(potpis pristupnika)

Dostaviti:

- Pristupnik
- Studentska služba
- Mentor
- Djelovođa Povjerenstva
- Predsjednik Povjerenstva

Sadržaj

1. Uvod	3
2. Ljudski govor u teoriji.....	4
2.1. Glasnice	5
2.2. Vokalni trakt.....	6
3. Osnovna frekvencija u teoriji	7
4. Metode izračuna osnovnih frekvencija	9
4.1. <i>Robust Algorithm for Pitch Tracking</i>	9
4.1.1. Način rada	9
4.2. <i>Sawtooth Waveform Inspired Pitch Estimator</i>	11
4.3. Način rada.....	11
5. Softver	15
5.1. Okruženje.....	15
5.2. Programski alati	16
5.2.1. Linux	16
5.2.2. <i>Windows</i>	18
6. SPTK	19
6.1. Instalacija.....	19
6.2. Knjižnice.....	20
6.3. <i>SPTK</i> problemi	22
6.4. Uloga basha	23
6.5. <i>SPTK</i> alternativa	23
6.5.1. <i>The Hidden Markov Model Toolkit</i>	23
6.5.2. <i>Python</i>	24
7. Izračun prosječne fundamentalne frekvencije	25
7.1. Pretraživanje <i>VEPRAD</i> baze.....	25
7.1.1. Opis skripte <i>find.sh</i>	26
7.2. Izrezivanje datoteka	28
7.2.1. Opis skripte <i>cut.sh</i>	29
7.3. Izračun fundamentalne frekvencije za pojedinačni uzorak	31
7.3.1. Opis skripte <i>calc.sh</i>	32
7.4. Izračun ukupne prosječne fundamentalne frekvencije	33
7.4.1. Opis skripte <i>avg.sh</i>	34

8. Rezultati izračuna	35
8.1. Izračun <i>RAPT</i> metodom.....	36
8.2. Izračun <i>SWIPE</i> metodom.....	39
8.3. Usporedba.....	42
9. Zaključak.....	45
Literatura.....	46

1. Uvod

Moderna tehnološka rješenja i napredak znanosti omogućuju dubinsku analizu svojstava zvuka te primjenu opažanja i saznanja za razvoj novih tehnoloških rješenja koja čovjeku pomažu u svakodnevicu.

Završni rad pobliže obrađuje i istražuje pojam osnovne frekvencije kao temeljnog svojstva koje proizlazi iz fizičke predispozicije ljudskih glasnica.

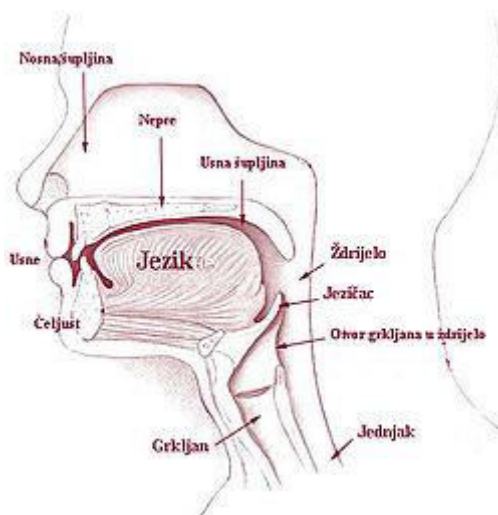
Naredne stranice će obuhvatiti teorijsku pozadinu ljudskog govornog trakta. Potom će se definirati i matematički potkrijepiti pojam osnovne frekvencije, te predstaviti algoritamske knjižnice korištene kod izračuna.

Glavni dio rada biti će predstavljanje napisanog programskog koda u paketu skripti za pripremu odabranih uzoraka i ekstrakciju bročane vrijednosti osnovnih frekvencija iz istih, koristeći programski paket *SPTK*. Navedeni će programski paket, programsko okruženje, kao i cjelokupan popis korištenih programa, biti navedeni i ukratko prezentirani u 4. poglavlju. Rezultati izračuna dobiveni korištenjem programskog koda prikazat će se, analizirati, te će, sukladno tome, biti doneseni zaključci.

2. Ljudski govor u teoriji

Prilikom komunikacije primjenom govora, poruka se u apstraktnom obliku pojavljuje u mozgu govornika te se zatim pretvara u skup neuronskih signala koji upravljaju postupkom artikulacije. Mehanizam nastajanja zvuka se može podijeliti na 3 sekcije: pluća, vokalne nabore (glasnice) i artikulatore.

Prvi korak je potisak zraka iz pluća govornika koja djelovanjem mišića prsnog koša stišću i potiskuju zrak prema glasnicama koje potom vibriraju. Takvim periodičkim titranjem, glasnice formiraju periodičku struju zraka, koja zatim prolazi kroz ždrijelo, odnosno usnu šupljinu koja ima ulogu svojevrsnog filtra [2]. To znači da usna šupljina djeluje na proizvedeni zvučni signal poput rezonantne kutije (transfer funkcija), dajući tom zvuku dodatne karakteristike (boja i kvaliteta glasa). Ova kombinacija rezultira stvaranjem oblikovanog spektruma sa širokopojasnim energijskim maksimumima [3].



Slika 2.1. Prikaz ljudskog vokalnog trakta

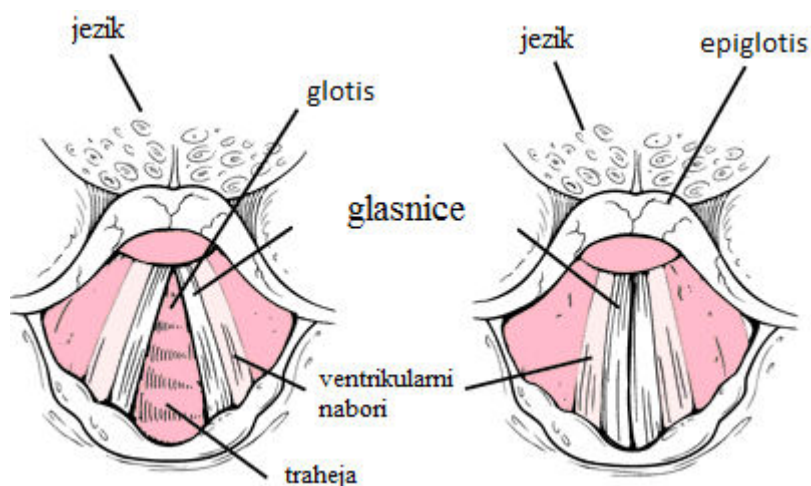
U fizikalnom procesu nastajanja govora sudjeluju pluća govornika koja se pod djelovanjem mišića prsnog koša stišću i potiskuju zrak kroz vokalni trakt.

Kao sličan primjer rezonantne kutije može se navesti udaralački (engl. percussion) instrument – *cajón*. Instrument najčešće ima oblik kutije (po tome je i dobio naziv), te ima jedan otvor sa stražnje strane [5]. Osnovni se pobudni signal generira udaranjem u prednju drvenu stijenku instrumenta, zatim se *obogaćuje* formantima, na mjestima pojačavanja prolaskom kroz rezonantnu kutiju (harmonici), te konačno izlazi van kroz otvor [4].

Primjer tog instrumenta ekvivalentan je primjeru ljudskog govornog organa koji također ima jedan otvor – usta, a pobudni signal, kako je već rečeno, nastaje pritiskom zraka, odnosno „udarom“ zraka.

2.1. Glasnice

Glasnice su vrlo značajan organ u procesu formiranja govora. One se ponašaju kao mehanički oscilator (pisak), koji prelazi u stanje relaksacijskih oscilacija uslijed struje zraka iz pluća koja kroz njih prolazi. Glasnice se odvajaju prilikom udisaja, a skupljaju prilikom gutanja ili govora. Muškarci imaju dulje glasnice koje proizvode niže frekvencije uz manji broj titraja u vremenskom intervalu - što znači da je izlazni produkt prosječno dublji, muškiglas. Žene, s druge strane, imaju kraće glasnice, te time proizvode više frekvencije (veći broj titraja u određenom vremenskom intervalu) što u konačnici znači i generiranje viših tonova.



Slika 2.2. Usporedba položaja glasnica. Nalijevo se može vidjeti odvajanje glasnica za vrijeme govora, dok je položaj glasnica u pasivnom stanju prikazan nadesno.

Na frekvenciju njihovog titranja, odnosno visinu tona utječu slijedeći parametri: pritisak zraka iz pluća na ulazu u glasnice, debljina i dužina glasnica te napetost samih glasnica.

Frekvencija titranja glasnica time je veća što glasnice brže titraju i obrnuto. Najmanja frekvencija pri kojoj glasnice mogu vibrirati se zove osnovna (F_0) frekvencija.

U slučaju da su glasnice potpuno opuštene, stisnute jedna do druge, neće doći do oscilacija i struja zraka iz pluća će neometano prolaziti kroz vokalni trakt.

2.2. Vokalni trakt

Tipična duljina vokalnog trakta je 17-18 cm, a djeluje kao rezonator oblika zatvorene cijevi. Vokalni trakt je temeljen na jednoj cijevi, čiji se oblik mijenja zavisno o položaju artikulatora. Trakt djeluje kao svojevrsan filter, kojemu je uloga da spektralno oboji pobudni signal. Slično kao što se geometrijom cijevi kod orgulja određuje ton (visina i spektralni sastav) signala koji se formira, tako će i geometrijski oblik vokalnog trakta određivati koje se spektralne komponente signala pojačavaju, a koje prigušuju [1].

Vokalni signal je kompleksno-periodičan val koji se sastoji od niza jednostavnih periodičnih valova. Ti se jednostavni periodični valovi zovu harmonici. Drugim riječima, može se reći da je pojačanje trakta uzorkovano sa umnošcima fundamentalne frekvencije – harmonicima, obzirom da je izvor harmoničan.

Harmonici su umnošci osnovne frekvencije i n -tog pozitivnog cijelog broja, odnosno:

$$H(k) = n \cdot F_0 \quad (2 - 1)$$

Energija je prisutna u svim harmonicima fundamentalne frekvencije glotalnog izvora, ali su amplitude pojedinih harmonika određene sa amplitudom izvora i sa funkcijom filtriranja. Ljudski vokalni trakt proizvodi više harmonika početne frekvencije, u odrasle osobe sve do 5000 Hz [12]. Porastom frekvencije svakog sljedećeg harmonika smanjuje se njegova amplituda.

Kako bi vokalni mehanizam bio u stanju proizvesti različite zvukove, kao npr. samoglasnike, on mora kontrolirati rezonancije vokalnog trakta koje proizvode karakteristične formante tj. formantne frekvencije (za samoglasnike su bitne F_1 , F_2 i djelomično F_3). To se ostvaruje kontrolom pozicije jezika, oblikovanjem usana pri izgovoru, kao i svakom promjenom koja utječe na oblik usne šupljine. Rezonator naglašava, pojačava pojedine harmonike (vidljivo tako što su oni prikazani kao maksimumi energije) tj. dodjeljuje formante tim harmonicima.

Promjenom formanta mijenjaju se odabrani harmonici, a time se utječe na boju i kvalitetu glasa. Nužno je spomenuti i tzv. zvučni spektar (engl. *soundspectrum*). To je reprezentacija isječka nekog zvuka – količina vibracije pri svakoj, individualnoj frekvenciji [13]. Uobičajeno se prikazuje grafom jačine zvuka ili tlačnog pritiska (dB) kao funkcije frekvencije (Hz).

Pojam fundamentalne frekvencije, koja je i predmet proučavanja ovog rada, bit će detaljnije opisan i matematički potkrijepljena u narednom poglavlju.

3. Osnovna frekvencija u teoriji

Osnovna(fundamentalna) frekvencija najniža je frekvencija svakog vibrirajućeg objekta. To je najniža frekvencija koju glasnice vibracijom mogu proizvesti. Također je poznata i kao prvi harmonik(F_0). Prosječne visine laringealnog glasa odnosno fundamentalne frekvencije čovjeka prikazane su u tablici 3.1. [14].

Tablica 3.1. Prikaz prosječnih fundamentalnih frekvencija ljudskog glasa

	Muškarci	Žene	Djeca
Pros. F_0	120 – 150 Hz	180 – 220 Hz	300 Hz

Ako je fundamentalna frekvencija ženskog glasa 200 Hz, to znači da zrak koji biva potisnut od strane glasnice titra 200 puta u sekundi, ili, drugačije rečeno, jednom u $\frac{1}{200}$ s. Vrijeme potrebno da se vibracija ponovi zove se period(T). U ovom je slučaju $T = \frac{1}{200}$, odnosno $T = \frac{1}{f}$. Drugi harmonik imat će vrijednost $2 \cdot F_0$, tj. imat će vremena točno za dvije vibracije (treći za tri, itd...).

Teorija harmonijskih gibanja kaže da se to gibanje onda može zapisati kao suma beskonačno mnogo sinusoidalnih funkcija, tako da je frekvencija pojedine sinusoide višekratnik frekvencije F_0 . Matematički to znači da je harmonik sinusoidalni doprinos određene frekvencije ukupnom periodičnom gibanju. Pri tom je frekvencija pojedinog harmonika višekratnik frekvencije F_0 koju onda zbog toga nazivamo još i fundamentalna frekvencija periodičkog gibanja.

Uzmimo slučaj da ženski glas ima fundamentalnu frekvenciju jednaku 200 Hz. Harmonici će se tada pojaviti na 800 Hz, 1200 Hz, 1600 Hz i tim slijedom sve dok postoji energija frekvencije. Tako se može reći da govorimo o frekvenciji raspona 8-10 kHz koju čovjek čuje kada se zvuk pušta na početnom stupnju fundamentalne frekvencije od 200 Hz [6]. Visoke frekvencije daju zvukovima dubinu i boju. Ako se te frekvencije *odrežu*, uklanjaju se harmonici iz uzoraka. Zvuk postaje *prazan* i monoton.

Valja napraviti distinkciju između pojmova fundamentalne frekvencije i harmonika te frekvencija formantata. Promjenom fundamentalne frekvencije mijenjaju se vrijednosti harmonika, a jednako tome formantna se frekvencija mijenja promjenom već spomenutih parametara oblika vokalnog trakta. Fundamentalna frekvencija i frekvencije harmonika, kako je već prije rečeno, posljedica su titranja glasnica, dok su formantne frekvencije karakteristične frekvencije rezonatora odnosno prijenosne ili filter funkcije (uvjetovane pomicanjem artikulatora) [7].

Uz sve navedeno, osnovni i ključni parametar za određivanje spola govornika jest fundamentalna frekvencija. U sljedećem poglavlju opisati će se 2 metode za aproksimaciju fundamentalne frekvencije nekog uzorka.

4. Metode izračuna osnovnih frekvencija

4.1. Robust Algorithm for PitchTracking

Metoda za estimaciju dizajnirana da funkcioniра na bilo kojoj frekvenciji i *frame rate-u*. Pouzdan je i u izračunu širokog spektra F_0 , kod raznih vrsta govornika i u uvjetima buke.

U *RAPT* su implementirana rješenja koja doprinose efikasnosti odnosno značajno smanjuju vrijeme izračuna, održavajući pritom preciznost.

RAPT algoritam ima prisutna kašnjenja pri izračunu *vuče* i nekoliko sekunda kašnjenja, tako da je njegova primjena ograničena na slučajeve u kojima se tolerira kašnjenja.

Kod izračuna se koriste dvije verzije uzorkovanih podataka: uzorak sa originalnom frekvencijom uzorkovanja i uzorak sa znatno smanjenom frekvencijom uzorkovanja.

4.1.1. Način rada

Algoritam računa periodične *NCCF* (engl. *normalized cross-correlation function*) signala uzorkovanog na niskoj frekvenciji za sva kašnjenja unutar interesnog područja. Pritom se spremaju lokalni maksimumi kod prvog prolaska *NCCF-a*. Kroskorelacija je mjera sličnosti dvije serije kao funkcije vremenskog pomaka jedne u odnosu na drugu. U prvom prolazu ulazni signal se uzorkuje nižom frekvencijom uzorkovanja, F_{ds} , koju se dobiva iz sljedećeg izraza:

$$F_{ds} = \frac{F_s}{\text{round}(\frac{F_s}{4F_{0\max}})}, \quad (4 - 1)$$

pri čemu je F_s frekvencija uzorkovanja originalnog signala govora, a $F_{0\max}$ vrijednost osnovne frekvencije.

Nakon toga računa se *NCCF* signala uzorkovanog većom frekvencijom i to samo u blizini relevantnih vrhova (engl. *pitch*) pronađenih u prvom prolasku *NCCF*. Ponovo se traži lokalni maksimum u refiniranoj *NCCF* kako bi se dobila poboljšana lokacija vrhova i aproksimacije pripadajućih amplituda.

Priložena je normalizirana kroskorelacijska funkcija koja se koristi za generiranje kandidata:

$$F_{ds} = \frac{F_s}{\text{round}(\frac{F_s}{4F_{0\max}})}, \quad (4 — 2)$$

Pri čemu je $\theta_{i,k}$ kandidat u vremenskom pomaku u analizi i -tog okvira, e_m izraz za energiju normalizacije.

$$e_j = \sum_{i=j}^{j+n-1} s_i^2, \quad \theta_{i,k} = \frac{\sum_{j=m}^{m+n-1} s_j * s_{j+k}}{\sqrt{e_m * e_{m+k}}}, \quad (-1 \leq \theta_{i,k} \leq 1). \quad (4 — 3)$$

Svaki vrh sačuvan iz *NCCF* visoke rezolucije generira kandidata F_0 za taj okvir (engl. *frame*). Relevantni vrhovi u trenutno susjednim analizama okvira su obično locirani na usporednim vremenskim pomacima, s obzirom da je F_0 sporo varirajuća funkcija u vremenu [9].

Optimalna objektivna funkcija za i -ti okvir jest:

$$D_{i,j} = d_{i,j} + \min_{k \in I_{i-1}} \{D_{i-1,k}, \delta_{i,j,k}\}, \quad 1 \leq j \leq I_i, \quad (4 — 4)$$

S početnim uvjetima:

$$D_{0,j} = 0, 1 \leq j \leq I_0; I_0 = 2. \quad (4 — 5)$$

Kod svakog okvira relevantna je i hipoteza bezzvučnosti samog okvira s obzirom da se kratkotrajni spektar zvučnih i bezzvučnih okvira dosta razlikuje. Kada postoji više maksimuma i imaju vrijednosti blizu 1,0, maksimum koji odgovara najkraćem periodu je obično relevantan.

Za odabir skupa najboljih kandidata za F_0 se koristi dinamičko programiranje. Kandidati se pritom odabiru kombinacijom lokalnog i kontekstualnog proračuna.

Dva stupnja *NCCF*-a provode se kako bi se smanjila kompleksnost sveukupnog izračuna. Interpolacija vrhova na signalu originalne frekvencije uzorkovanja koristi se kako bi se poboljšala preciznost.

Okvirna estimacija fundamentalne frekvencija dobiva se preko izraza:

$$F_{0i} = \frac{F_s}{L_{i,j}}, \quad (4 — 6)$$

gdje su vrijednosti j izraza one koje rezultiraju u vrijednosti globalnog minimuma za D .

4.2. *SawtoothWaveformInspiredPitchEstimator*

Metoda funkcionira na način da se vrh određuje kao fundamentalna frekvencija zupčanog vala čiji spektar najbolje odgovara spektru ulaznog signala. Kosinusov *kernel* nadograđen je algoritmom (baziranim na frekvenciji) sa sitastom estimacijom, čime se dobiva glatke vrhove sa amplitudama, za korelaciju s harmonicima [8].

Poboljšanje algoritma je postignuto tako što se koriste samo prvi harmonik i osnovni harmonici, što značajno smanjuje pogreške koje se često pojavljuju u drugim algoritmima. Srž algoritma je pronalazak frekvencije koja maksimizira prosječne udaljenosti između brijega i dola valova harmonika te frekvencije. Umjesto uporabe logaritma spektruma, kod harmonika se primjenjuje faktor monotonične težinske vrijednosti zastarijevanja, promatra se spektar u blizini harmonika i u primjeni su tzv. funkcije za težinsko ugađivanje (engl. *smooth weighting*). Uporaba logaritma u integralnoj transformaciji spektruma je nepogodna jer se mogu pojaviti regije spektruma bez energije, što bi spriječilo evaluaciju integrala, s obzirom da je:

$$\log(0) = -\infty. \quad (4 — 7)$$

Generalno govoreći, algoritam se može opisati kao izračun sličnosti između kvadratnog korijena spektruma signala i kvadratnog korijena spektruma zubastog vala, koristeći pritom optimalne veličine prozora

4.3. Način rada

Početni je korak izračun prosječne udaljenosti između brijega i dola vala.

Kao što je već rečeno, ako je signal periodičan i sa fundamentalnom frekvencijom f , njegov spektar mora sadržavati vrhove na umnošcima te frekvencije i dolovima između.

Kako je svaki vrh okružen sa dva dola vala, prosječna udaljenost između brijega i dola za k -ti vrh jest:

$$d_k(f) = [|X(kf)| - \frac{1}{2} [|X((k - \frac{1}{2})f)| + |X((k + \frac{1}{2})f)|]]. \quad (4 — 8)$$

Gledajući prosjek prvih n vrhova, globalna udaljenost vrha i dola jest:

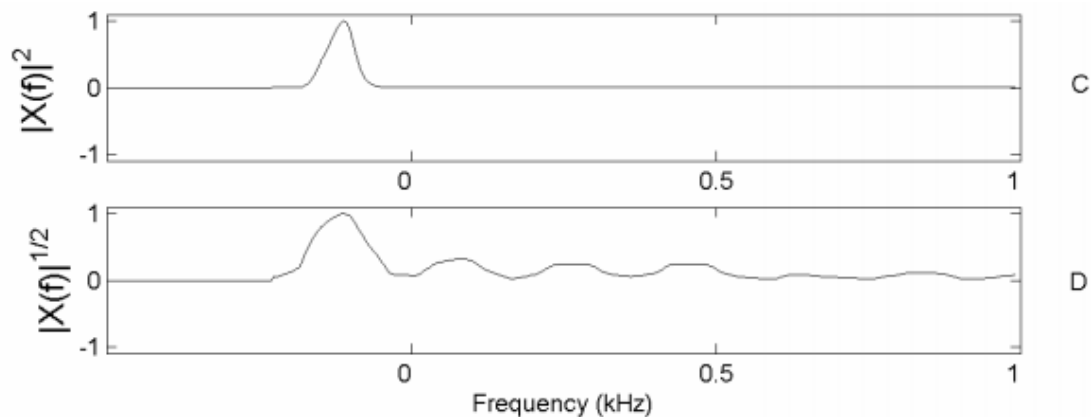
$$D_n(f) = \frac{1}{n} \sum_{k=1}^n d_k(f), \quad (4 — 9)$$

Gdje se u umjesto izraza $d_k(f)$ uvrštava izraz iz gornje relacije.

Prethodno metoda mjerenja udaljenosti vrha i dola funkcionira ako je signal harmoničan, ali ne i ako je disharmoničan. Kako bi se omogućio valjani izračun u potonjem slučaju, prvi korak jest *zamućivanje* lokacije harmonika zamjenom svakog pulsa trokutastom funkcijom sa bazom $f/2$,

$$\Lambda_f(f') = \begin{cases} \frac{f-f'}{4}, & \text{ako je } |f'| < \frac{f}{4} \\ 0 & (\text{inače}). \end{cases} \quad (4 — 10)$$

RAPT, kao ni *SWIPE*, ne koristi logaritam spektruma, već deformaciju drugog korijena. Potonji bolje aproksimira vrijednosti vrhova, omogućuje da težinskim vrijednostima harmonika budu proporcionalne njihovim amplitudama i daje bolji odziv auditornog sustava prema amplitudi, što je vidljivo na slici 4.1.



Slika 4.1. Deformacija spektruma koristeći $\sqrt[2]{X}$.

Kako bi se izbjegao problem pogreške nastao obuhvaćanjem subharmonika, spomenut u prethodnom potpoglavlju, faktor težinske vrijednosti zastarijevanja se aplicira na harmonike. Važna je parametar broj harmonika koji se koriste za analiziranje vrha. Oni su ograničeni sa 3,3 kHz za govorne signale radi smanjenja troškova izračuna, jer harmonici iznad te vrijednosti nisu značajnije doprinijeli povećanju točnosti.

Za najčešće vrste prozora (engl. *window*) koji se koriste u procesiranju signala, širina glavnog dijela iznosi $2k/T$, gdje parametar k ovisi o vrsti prozora. Optimalna veličina za analizu signala se može dobiti izjednačavanjem $2k/T$ sa širinom $f/2$, a konačni izraz je:

$$T^* = T = 4 \frac{k}{f}. \quad (4 — 11)$$

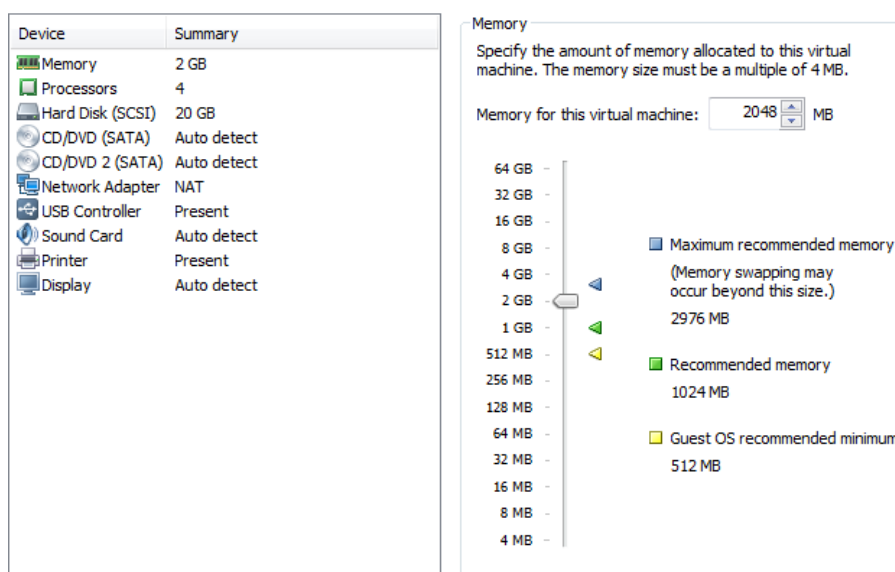
Zaokruživši sve prethodno navedene korake, *SWIPE* aproksimacija vrha u vremenu T može se formulirati kao:

$$p(t) = \operatorname{argmax} = \frac{\int_0^{ERBs(f_{max})} \frac{1}{\eta(\varepsilon)^2} K(f, \eta(\varepsilon)) |X(t, f, \eta(\varepsilon))|^{1/2} d\varepsilon}{(\int_0^{ERBs(f_{max})} \frac{1}{\eta(\varepsilon)} [K^+(f, \eta)]^2 d\varepsilon)^{1/2} (\int_0^{ERBs(f_{max})} |X(t, f, \eta(\varepsilon))| d\varepsilon)^{1/2}}. \quad (4 — 12)$$

5. Softver

5.1. Okruženje

U radu sa knjižnicama, algoritmima i rezultatima izračuna koristi se virtualna mašina sa instaliranom *gostomUbuntu 14.04.1 LTS*(virtualizacija *Windows 7 Ultimate / Ubuntu 14.04.1 LTS*). Odabrana je *Ubuntu* platforma jer pruža sigurnije i stabilnije okruženje za rad, a osim toga, vrijeme i koraci instalacije *SPTK* alata puno su kraći. U postavkama je *gostu* dano *2GB RAM* memorije i *20GB* prostora na tvrdom disku. Postavke se mogu vidjeti na slici 5.1.



Slika 5.1. Osnovne postavke virtualnog uređaja Ubuntu. Za optimalan rad u nj. ključna je RAM memorija i tip procesora.

Iako je glavnina rada na projektu izvedena na *Linuxu*, koriste se i neki *Windows 7* alati poput manje poznatog *Tf32*, te poznatijeg *Wavesurfera* za usporedbu točnosti dobivenih fundamentalnih frekvencija. Više o korištenim programskim alatima će biti rečeno u nastavku.

5.2. Programski alati

5.2.1. Linux

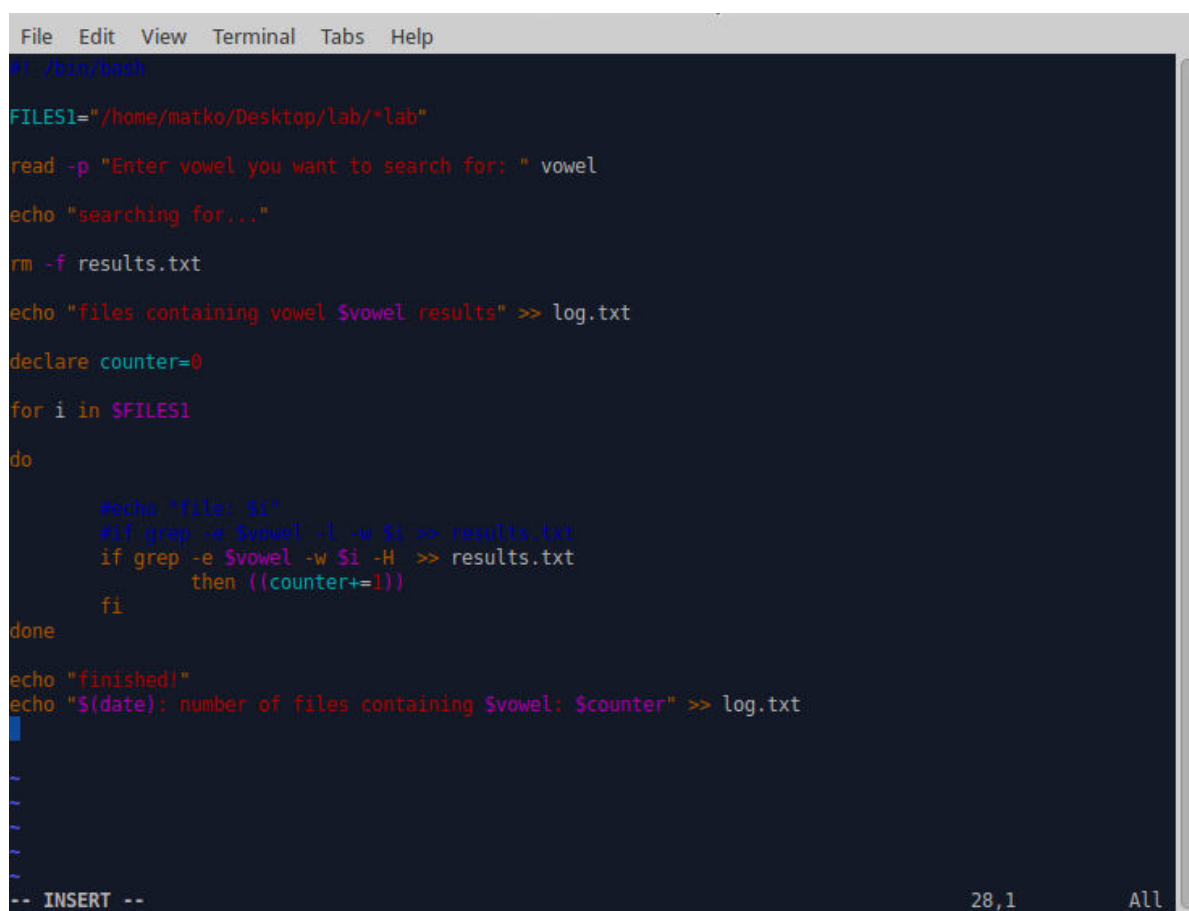
Bash

Bash je programski jezik, odnosno najčešće korištena ljuska za *UNIX* operacijske sustave. U projektu se koristio u prvom redu za komunikaciju sa svim relevantnim korištenim *linux* alatima (*SoX*, *SPTK*, ...), pisanje skripti, testiranje, te za pohranu rezultata.

Vim

Vim je tekstualni editor, klon *Bill Joy*evog uređivača teksta *vi*, čije ime dolazi od *Vi IMproved*.

Konkretno u ovom slučaju, koristio se putem *command line-a*, kao što je prikazano na slici 5.2., radi efikasnijeg rada u *cmd-u*.



```
File Edit View Terminal Tabs Help
#!/bin/bash

FILES1="/home/matko/Desktop/lab/*lab"

read -p "Enter vowel you want to search for: " vowel

echo "searching for..."

rm -f results.txt

echo "files containing vowel $vowel results" >> log.txt

declare counter=0

for i in $FILES1
do
    #echo "file: $i"
    #if grep -e $vowel -l -w $i >> results.txt
    if grep -e $vowel -w $i -H >> results.txt
    then ((counter+=1))
    fi
done

echo "finished!"
echo "$(date): number of files containing $vowel: $counter" >> log.txt

-- INSERT --
```

Slika 5.2. Prikaz *Vim* uređivača teksta kroz *command line* sučelje. Prikazan je programski kod skripte koja ima ulogu nalaženja relevantnih audio datoteka.

Speech Signal Processing Toolkit

SPTK je skup knjižnica za procesiranje *zvučnih* signala napisanih u programskom jeziku C za rad u *UNIX* okruženju. Paket je nastao od strane više autora koji su sudjelovali u istraživanju, a najoriginalniji *source* kodovi paketa su pripisani Takao Kobayashiju. *SPTK* sadrži par desetaka knjižnica koje se koriste za rezanje datoteka, konvertiranje *WAV* i *RAW* formata, konverzije frekvencije uzorkovanja, prikaz grafova, provedbu *LPC* i *PARCOR* analize, LSP analize spektra signala, LSP sinteze filtera, i dr. U konkretnom se slučaju koriste slijedeće knjižnice: *bcut*, *x2x*, *pitch*, *psgr*, *fdrw*.

Sound eXchange

SoX je platforma za procesiranje širokog spektra formata zvučnih signala (slici 5.2.). Alat nudi mogućnosti pretvaranja *audio* formata iz jednog u drugi, izrezivanje zvučnih datoteka, kao i dodavanje efekata na signal [10]. U projektu se koristio za izrezivanje traženih komadića pojedine *WAV* datoteke, kao segment veće cjeline, jedne od korištenih napisanih *bash* skripti.

```
sox:      SoX v14.4.2

Usage summary: [gopts] [[fopts] infile]... [fopts] outfile [effect [effect]]...

SPECIAL FILENAMES (infile, outfile):
-      Pipe/redirect input/output (stdin/stdout); may need -t
-d, --default-device      Use the default audio device (where available)
-n, --null                Use the 'null' file handler; e.g. with synth effect
-p, --sox-pipe            Alias for '-t sox -'

SPECIAL FILENAMES (infile only):
"[program [options] ..." Pipe input from external program (where supported)
http://server/file        Use the given URL as input file (where supported)

GLOBAL OPTIONS (gopts) (can be specified at any point before the first effect):
--buffer BYTES            Set the size of all processing buffers (default 8192)
--clobber                 Don't prompt to overwrite output file (default)
--combine concatenate     Concatenate all input files (default for sox, rec)
--combine sequence        Sequence all input files (default for play)
-D, --no-dither           Don't dither automatically
--dft-min NUM             Minimum size (log2) for DFT processing (default 10)
--effects-file FILENAME   File containing effects and options
-G, --guard               Use temporary files to guard against clipping
-h, --help                Display version number and usage information
--help-effect NAME        Show usage of effect NAME, or NAME=all for all
--help-format NAME        Show info on format NAME, or NAME=all for all
-i, --info                Behave as soxi(1)
--input-buffer BYTES      Override the input buffer size (default: as --buffer)
--no-clobber              Prompt to overwrite output file
-m, --combine mix         Mix multiple input files (instead of concatenating)
--combine mix-power       Mix to equal power (instead of concatenating)
-M, --combine merge       Merge multiple input files (instead of concatenating)
--multi-threaded          Enable parallel effects channels processing
--norm                    Guard (see --guard) & normalise
--play-rate-arg ARG       Default 'rate' argument for auto-resample with 'play'
--plot gnuplot|octave     Generate script to plot response of filter effect
-q, --no-show-progress    Run in quiet mode; opposite of -S
--replay-gain track|album|off Default: off (sox, rec), track (play)
-R                        Use default random numbers (same on each run of SoX)
-S, --show-progress       Display progress while processing audio data
```

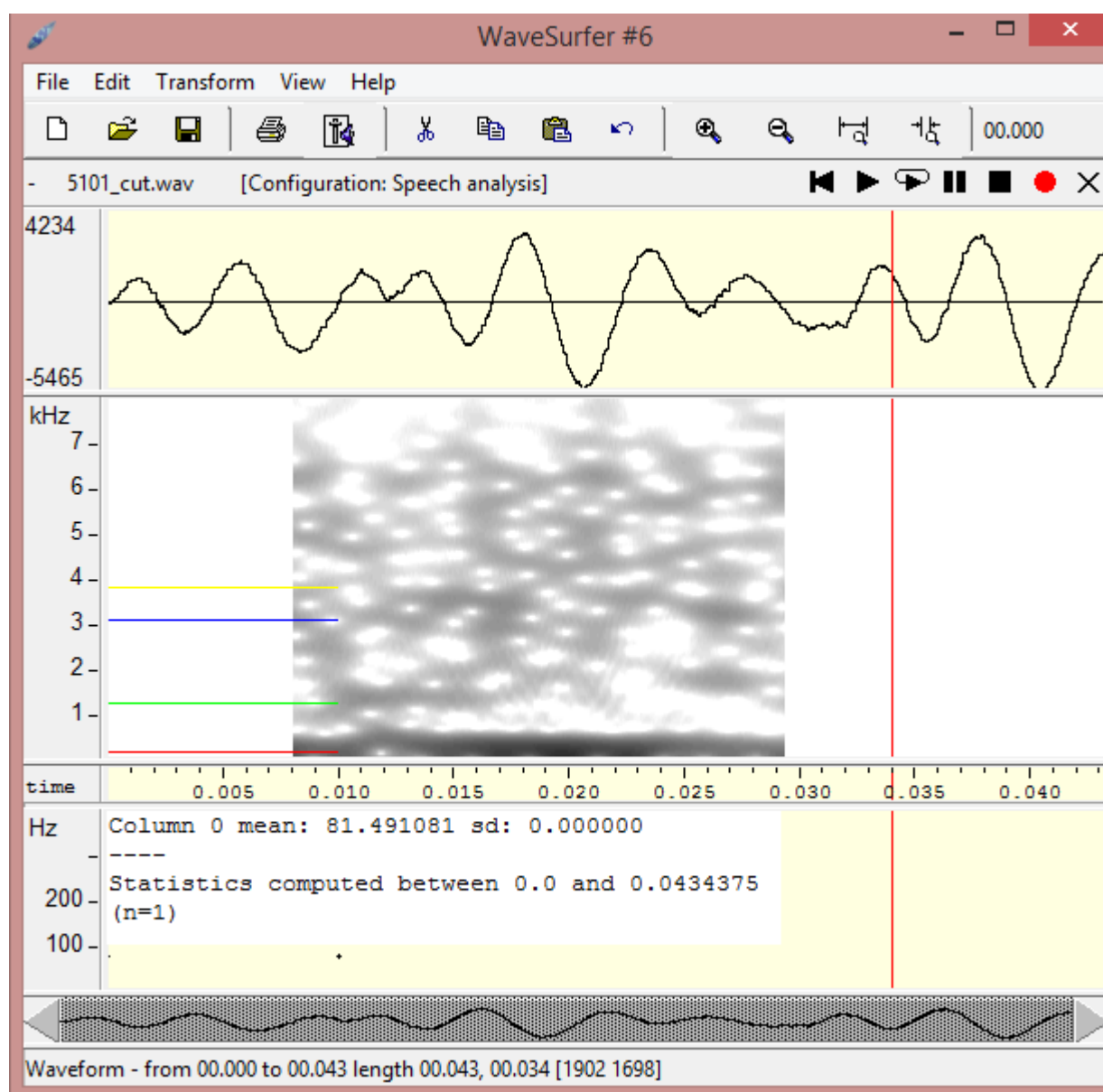
Slika 5.3. Prikaz SoX - -help knjižnice u cmd-u.

¹Takao Kobayashi (rođen 1961.), japanski astronom

5.2.2. Windows

Wavesurfer

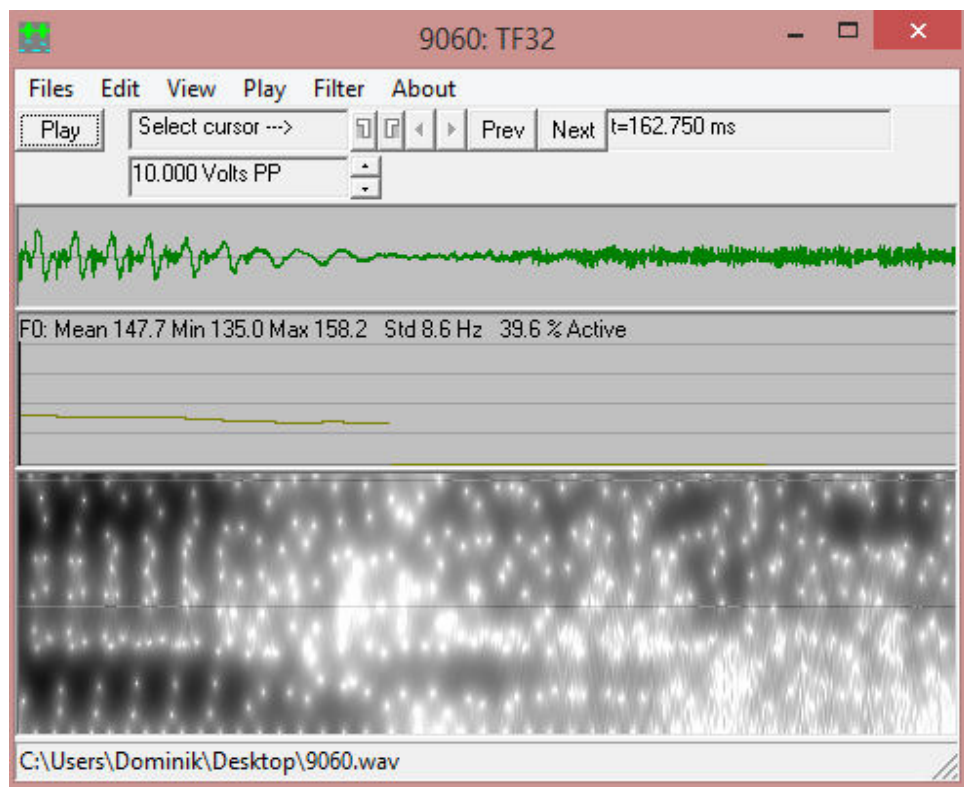
Wavesurfer je *open-source* alat za vizualizaciju i manipulaciju audio datotekama. Prikaz grafičkog sučelja se može vidjeti na slici 5.4. Wavesurfer posjeduje jednostavno i logički posloženo sučelje. Može se koristiti kao alat za samostalan rad u istraživanju zvuka. Upravo iz tog razloga, kao pouzdano rješenje, koristi se kao relevantan izvor aproksimiranih fundamentalnih frekvencija na pojedinim testnim primjercima.



Slika 5.4. Wavesurfer grafičko sučelje. Prikazana je WAVdatoteka u Speech Analysis konfiguraciji prozora. Prikazan je spektrogram te izračun prosječne frekvencije.

TF32

TF32(slika 5.5.)je također jedno od dostupnih *open-source* projekata za vremensko-frekvencijsku analizu čiji se rezultati, kao i oni dobiveni koristeći *Wavesurfer*, mogu usporediti sa rezultatima dobivenim pomoću izrađenih skripti.



Slika 5.5. Grafičko sučelje programa TF32. Prikaz WAV datoteke kroz spektrogram, te izračun minimalne, maksimalne te prosječne fundamentalne frekvencije

Notepad++

Uređivač teksta sa grafičkim sučeljem koji je korišten za finalno uređivanje skripti.

MS Word 2013

Dio paketa *MS Office*. Korišten za izradu dokumentacije ovog projekta odnosno završnog rada.

6. SPTK

6.1. Instalacija

Prije nego se krene za instalacijom paketa, potrebno je preuzeti *SPTK.tar.gz* datoteku. U njoj se nalazi tekstualna datoteka *INSTALL* sa uputama za instalaciju. Postupak je uobičajen na *linux* platformama. Nakon ekstrakcije *tar.gz* datoteke, putem *cmd-a* se naredbom *cd* pozicioniramo u direktorij ekstrakcije. Potom se naredbom *./configure* provjerava kompatibilnost operativnog sustava i *hardware-a* sa paketom te se potom stvara *makefile* koji se koristi u sljedećem koraku. Naredbom *make* i *makeinstall* instalira se *library* i *binary*.

Ukratko, *./configure* provjerava ako je sve spremno za *build* aplikacije, koji će se izvršiti u koracima *make* i *make install*. Naredba *make* prevodi (engl. *compile*) izvorni kod, te se potom stvaraju izvršne (engl. *executable*) datoteke. *Make install*, nakon što *make* stvori izvršne datoteke, sve relevantne datoteke aplikacije svrstava u odgovarajuće systemske direktorije [11].

6.2. Knjižnice

BCUT

Bcut knjižnica je kratica od *binary file cut*. Kao što samo ime nalaže, knjižnica se koristi za izrezivanje dijelova signala. Drugim riječima, selektirani se dio ulazne datoteke kopira u novu, izlaznu datoteku. Kao nužni parametri se za podešavanje koriste *-s* kao vrijednost kod koje izrezivanje počinje, dok *-e* označava kraj izrezivanja.

Primjer funkcionalne naredbe jest *bcut +f -s 3 -e 5 data.f > data.cut*.

Bez obzira na ispravnost rada, navedena se knjižnica nije koristila za izrezivanje signala koji su analizirani u ovom projektu. Razlog tome jest što su mjesta pojedinih glasova u snimkama *VEPRAD* baze podatakabilježena u nanosekundama, a *bcut* signal ne gleda po vremenskoj domeni, već ga dijeli na određen broj uzoraka koje potom numerira ($i=0; i++; i < \text{krajUzorka}$).

Iz tog razloga, kako bi se pojednostavio problem, koristi se alternativni, već spomenuti alat – *SoX*.

GWAVE

Gwave jest naredba koja služi za crtanje valnog oblika, signala vala. Naredba pretvara valni oblik govora ulazne datoteke u *FP5301* format crtanja, proslijeđujući rezultat na standardni *output* (ukoliko nema *pipelining* prema *psgr* ili sl.). *Gwave* je implementiran kao skriptna ljuska (engl. *shell script*) koja koristi *fig* i *fdrw* naredbe. Sadrži parametre poput *-s* i *-e*, odnosno

početne i krajnje točke prikazivanja valnog oblika, -ykojim se određuje maksimalna amplituda koja će biti obuhvaćena, -y2 kojim se određuje minimalna obuhvaćena amplituda itd.

PSGR

PSGR ima ulogu konverzije *FP5301* komandi crtanja iz ulazne datoteke u izlaznu *EPS* datoteku. Jednostavnije rečeno, u ovom radu je korišten za spremanje podataka u grafičkom obliku, kao alternativa *xgr-u* (više u prethodnom poglavlju), u izlaznu *EPS* datoteku.

Primjer naredbe je *gwave +f data.f | psgr >data.eps*.

Knjižnica nudi niz parametara koji se mogu koristiti, kao npr. parametri *-T*, *-B*, *-L*, *-R* za postavljanje margina, *-c* za postavljanje broja izlaznih kopija, *-r* za podešavanje rezolucije (*dpi*), *-x* za poravnanje u smjeru *x-osi* i *-y* za poravnanje u smjeru *y*.

PITCH

Pitch je naredba u kojoj leži ključ izračuna fundamentalne frekvencije zvučnih signala.

Točnije, koristi se za ekstrakciju vrhova zvučnog signala, iz kojih se potom računa fundamentalna frekvencija. *Pitch* naredba računa vrijednosti ulazne datoteku, šaljući numerirane izračune u izlaznu datoteku dogovorene ekstenzije *pitch*. Kod korištenja *Pitch* naredbe može se odabrati koji će se algoritam koristiti: *RAPT* ili *SWIPE*. To se može odabrati preko parametra *-a* (*A=0* za *RAPT*, odnosno *A=1* za *SWIPE*).

Primjer naredbe jest *pitch -a 1 -s 8 -p 80 -L 50 -H 400 -o 1 data.f > data.pitch*.

Mora se također postaviti i parametar *-s* odnosno frekvencija uzorkovanja, koja ovisi o vrsti ulaznog signala. Naposljetku, format izlaznih vrijednosti određuje se preko *-o*, a od mogućih *O=0* za *pitch*, odnosno *O=2* za $\log(F_0)$, za ovaj slučaj odabire se *O=1* za F_0 . Više o ovoj naredbi biti će rečeno u narednom poglavlju. Odabir filtera *-L* donju granicu traženja, odnosno *-H* za gornju granicu traženja fundamentalnih frekvencija nije nužno potrebno uključiti, ali se podešavanjem ipak dobivaju nešto preciznije vrijednosti (na taj način da se eliminiraju vrijednosti izvan okvira vjerovatnih fundamentalnih frekvencija kodanalize odabranog vokala).

6.3. SPTK problemi

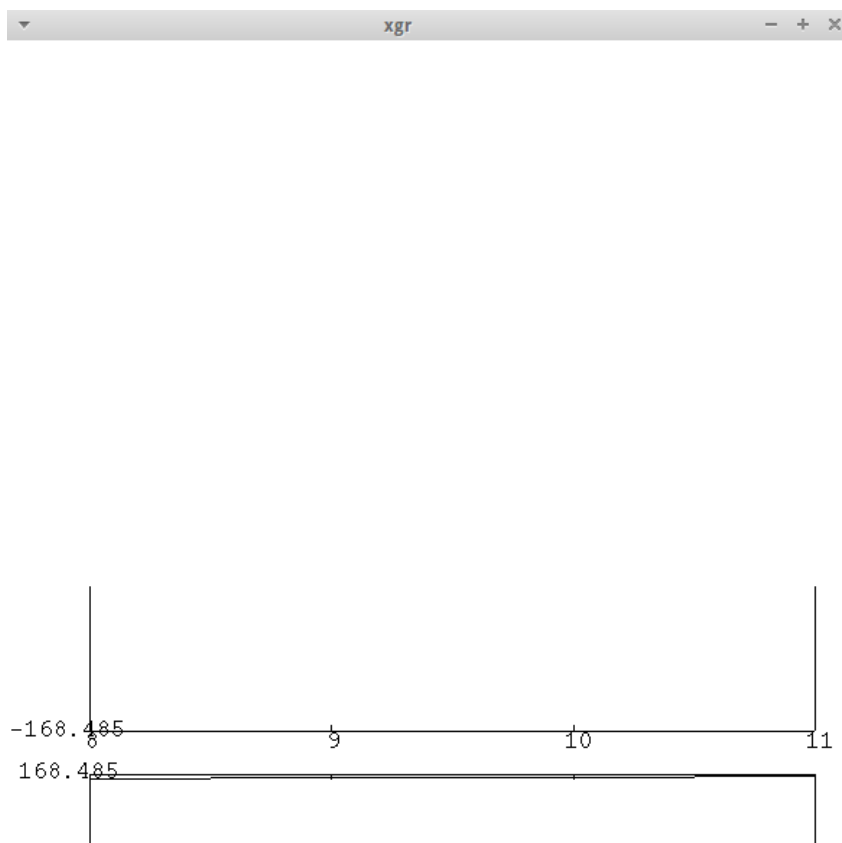
Pri testiranju *SPTK* paketa, unosom naredbe `gwave +s data.short | xgr`, izlazni je *output* bio identičan onomu prikazanom na slici 6.1.

```
X Error of failed request: BadName (named color or font does not exist)
Major opcode of failed request: 45 (X_OpenFont)
Serial number of failed request: 58
Current serial number in output stream: 59
```

Slika 6.1. Na slici se nalazi error poruka koja se javlja pri pokušaju prikaza izlaznih rezultata naredbom *xgr*

Problem se javljao kod svakog poziva *xgr* naredbe koja služi kao simulator *XY* dijagrama. *Xgr* naredba neizostavna je ukoliko se direktno kroz *pipelining* automatskim otvaranjem prozora želi dobiti *output*. Istraživanjem sličnih problema na *internet* forumima, ispostavilo se da je bilo potrebno instalirati *xorg* fonts *100dpi* and *75dpi*. Iako je time riješen gore navedeni problem, *xgr* kao grafički output daje poprilično nepregledan ispis, koji često ne stane na ekran, vidljive su bijele mrlje ili izrezani, nevidljivi dijelovi grafa. Primjer problema se vidjeti na slici 6.2.

U konačnici se može zaključiti kako je *xgr* nepouzdan, osim za *manje* (kraće) uzorke, kod kojih funkcionira nešto bolje.



Slika 6.2. Problemi prikaza rezultata metodom *xgr*

Kao alternativa je korištena naredba `psgr odnosnogwave +s data.short | psgr > data.eps`, čime se cjelokupan *output* spremaju *eps* (engl. *Encapsulate PostScript*) datoteku koja se potom može otvoriti alatom *Document Viewer* (ili sličnim) te pomicati (engl. *scroll*). U prozoru otvorenom sa *xgr* pomicanje stranice (engl. *scroll page*) nije dopušteno.

Unosom naredbe `x2x +sf data.short | pitch -a 1 -s 16 -p 80 -L 80 -H 165 > data.pitch` za navedenu *data.short* datoteku ekstrahiraju vrhovi (engl. *pitch*).

Kao output analiziranih datoteka na određeni broj dobivenih izreznih zvučnih glasova uzoraka (npr. izrezani glas *a* u svim *WAV* datotekama baze *VEPRAD*) *SPTK* kao izlazne vrijednosti daje 0, što u konačnici smanjuje ukupnu količinu vrijednosti na kojima se može računati prosječna fundamentalna frekvencija.

Jedan od mogućih razloga jest i to što se u ovom konkretnom slučaju koristi *SWIPE* algoritam za detekciju vrhova (*RAPT* za usporedbu izbacuje puno manje nultih vrijednosti). *SWIPE* i *RAPT* algoritmi teorijski su obrađeni u prošlom poglavlju, a u narednom poglavlju će biti prezentirana njihova primjena.

6.4. Uloga basha

Bash kao programski jezik se koristi sa pisanje skripti pomoću kojih se vrši selekcija datoteka, njihovo izrezivanje, aproksimacija i izračun prosjeka vrijednosti. Točnije, *bash* se koristi kao programski jezik koji obuhvaća kompletnu logičku stranu koda, dok se oni ključni komadići koda popunjavaju pozivima na pojedine *SPTK* knjižnice odnosno pozivom na specifičnu naredbu alata *SoX*. Više o samom kodu i pozivima unutar skripti biti će rečeno u narednom poglavlju.

6.5. SPTK alternativa

6.5.1. TheHidden Markov Model Toolkit

HTK se javlja kao alternativa *SPTK* knjižnicama, a koristi se sa stvaranje i manipulaciju Markovljevih modela. U odnosu na *SPTK*, *HTK* je primjenjiv na širem području istraživanja. Primarno se koristi u istraživanjima glasovnog prepoznavanja, iako se koristi i u brojnim drugim područjima, poput sekvenciranja *DNK*, pa tako i sinteze govora. Sadrži module i alate napisane u programskom jeziku *C* koji se mogu koristiti u analizi audio signala, testiranju i analizi dobivenih rezultata.

HTK se može podignuti na *Linux* i na *Windows-u*. Za preuzimanje *tar.gz* datoteke sa izvornim kodom potrebno jest registrirati se i prihvatiti uvjete licence.

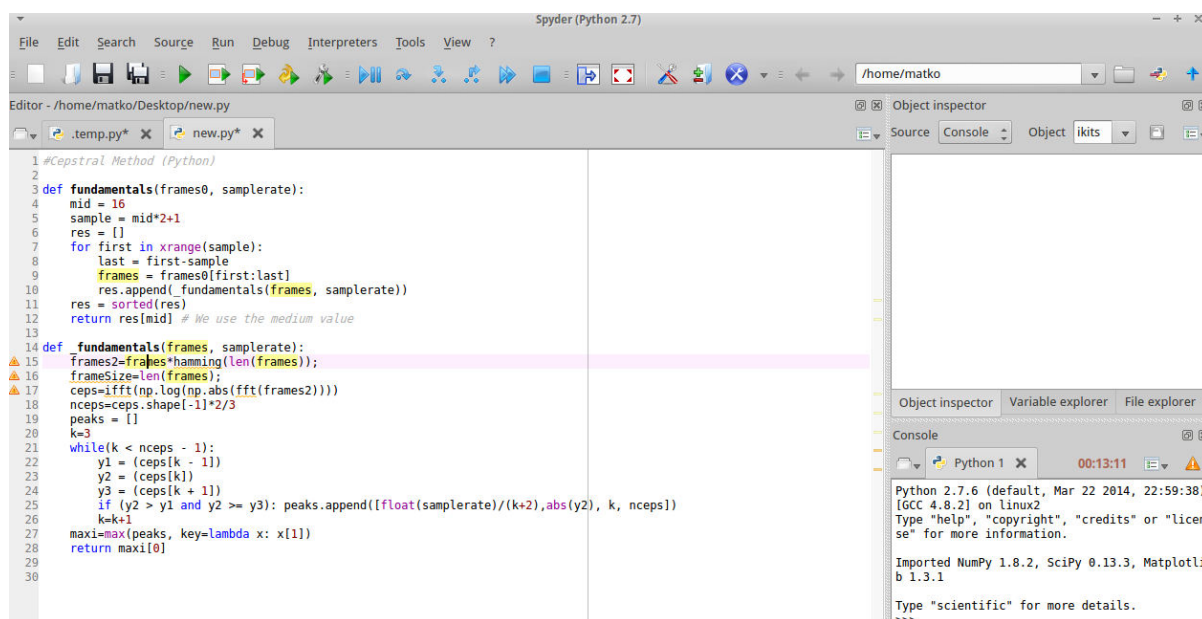
6.5.2. Python

Programski se jezik *python*, osim u *cmd-u*, može prevoditi (engl. *compile*), i s njime raditi, i u interaktivnom okruženju *Spyder*.

Okruženje je kros-platforma, vrlo se lako instalira, a u sebi, osim *compiler-a*, sadrži *python interpreter*, *command prompt* i *object inspector*, odnosno priručnik za uporabu brojnih knjižnica koje *python* sadrži, kao i brojne druge opcije. Grafičko sučelje prikazano je na slici 6.3.

Spyder, osim u „punoj“ verziji dolazi u nešto „lakšoj“, *Spyder (light)* verziji koji sadrži samo *interpreter* odnosno *command prompt*. Uz paket knjižnica koji *python* u osnovnom setu nudi, postoji i niz dodatnih knjižnica koje se preko interneta mogu preuzeti.

Za procesuiranje i obrađivanje audio datoteka, potrebno jest preuzeti dodatan set knjižnica *scikits.talkbox*. Prednost u odnosu na *SPTK* knjižnice definitivno jesu puno jednostavnije, ali i potpunije opcije grafičkog prikaza izlaznih vrijednosti. Primjenom *python* knjižnica za obradu signala mogli bi se dobiti rezultati, u najmanju ruku, usporedivi sa onima dobivenim s pomoću *SPTK*.



Slika 6.3. *Spyder* okruženje. Primjer python programskog koda.

7. Izračun prosječne fundamentalne frekvencije

Za izračun prosječne fundamentalne frekvencije vokala su korištene 4 manje skripte.

U ovom poglavlju svaka od njih će se opisati redom kojim se one koriste: *find.sh* → *cut.sh* → *calc.sh* → *avg.sh*, te će se zatim detaljnije, pojedinačno objasniti dijelove koda. Sve su skripte, kako je već ranije navedeno, napisane u *bash-u* uz, naravno, pozive na knjižnice *SPTK* i *SoX*.

7.1. Pretraživanje *VEPRAD* baze

Prvi korak izračun prosjeka osnovne frekvencije po uzorcima bio jest pretraživanje *VEPRAD* baze a segment koda je prikazan na prikazu skripte 7.1. Početni fond datoteka sastojao se od oko 2300 audio zapisa različitih formi teksta na hrvatskom jeziku izgovorenih od muškog govornika *SM04*. Audio zapisi su u *WAV* formatu, otipkani frekvencijom 16 kHz.

Napisana skripta funkcionira na način da najprije od korisnika zatraži unos vokala, a potom se taj vokal pretražuje preko prijepisa pojedinih uzoraka koji sadrže vremena svih vokala koji se pojavljuju unutar govornih zapisa tj. zvučnih signala. Nazivi datoteka kod kojih se traženi vokali pronadu, te vrijeme početka i završetka tih vokala, pohranjuju se u tekstualnu datoteku. Na taj način podaci su pripremljeni za idući korak izrezivanja.

```

#!/bin/bash

#bilježi se putanja direktorija iz kojeg će se povlačiti podatci o zapisima
PATH_1="/home/bezevan/Desktop/lab/*lab"

#skripta od korisnika traži unos vokala
read -p "Enter vowel you want to search for: "vowel
echo "searching for..."

rm -f -r statistics
mkdir statistics

echo "files containing vowel $vowel results" >> statistics/match_num.txt

#deklarirana je varijabla counter koja će se koristiti u petlji
declare counter=0

#traži se uneseni vokal u lab datotekama
for i in $PATH_1
do
#rezultati se pohranjuju u datoteku statistics/match_list.txt
    if grep -e $vowel -w $i -H >> statistics/match_list.txt
    then ((counter+=1))
    fi
done

echo "finished!"

#pohranjuju se statistički podatci
echo "$(date): number of files containing $vowel: $counter" >>
statistics/found_vowel_num.txt

```

Skripta 7.1. Prikazana je prva od 4 skripte: *find.sh*.

7.1.1. Opis skripte *find.sh*

Naredbaread od korisnika traži unos vokala odnosno samoglasnika za analizu.

Potrebna je varijabla koja služi za pohranu putanje prema direktoriju transkripcija - varijabla.

```
PATH_1="/home/bezevan/Desktop/
```

PATH_1 varijabla je nužan dio *for* petlje koja pomoću nje prebrojava sve datoteke u direktoriju *lab*, odnosno odredišnom direktoriju, i zatim ih pretražuje.

Grep naredba omogućuje da za svaku pronađenu *LAB* datotekuprovjeri sadrži li traženi vokal. Parametar *-e* određuje uzorak koji se traži, a *-H* modificira ispis na način da se za svaki pronađeni rezultat ispisuje samo ime datoteke. Svaki pronađeni uzorak, kao i njegovi parametri i ime datoteke pohranjuju se u *statistics/match_list.txt*. Brojač će po završetku skripte dati ukupan broj pronađenih uzoraka.

```

if grep -e $vowel -w $i -H >>
statistics/match_list.txt
    then ((counter+=1))
fi

```

7.2. Izrezivanje datoteka

Drugi je korak izrezivanje prethodno pronađenih vokala, što je realizirano segmentom koda skripte 7.2. Pritom je izvor potrebnih parametara za izrezivanje upravo tekstualna datoteka nastala u prethodnom koraku. Izrezani odsjecci pohranjuju se u novostvoreni direktorij *wav_cut* kao WAV datoteke. Nakon što se postupak završi, može se krenuti sa izračunom, odnosno sa slijedećim korakom.

```
#!/bin/bash

#mijenjanje putanja pojedinih zapisa u datoteci results.txt
sed -i s/://' '/' statistics/match_list.txt
sed -i 's/\\lab\\/wav/' statistics/match_list.txt
sed -i s/\\.lab\\.wav/ statistics/match_list.txt
sed -i -r 's/\\S+//4' statistics/match_list.txt

#prva 3 stupca pohranjuju se u polje collection za daljnju obradu
collection=( $(cut -d ' ' -f-3 results.txt) )

line=$(wc -w results.txt)

#deklariraju se varijable
Typeset -i i END
let END=$line cut_counter=1 count_fname=0 count_startt=1 count_endt=2

rm -f -r wav_cut

mkdir wav_cut

#u petlji se za svaku datoteku izrezuju trajanja vokala i spremaju u
wav_cut direktorij
while((count_startt<=END))
do
#povučene vrijednosti se pretvaraju u sekunde (zapis u ns) za potrebe SoX
alata
    read startt <<<$(echo "${collection[count_startt]}"*10^-09 | bc -l)
    read endt <<<$(echo "${collection[count_endt]}"*10^-09 | bc -l)

#string koji će biti ime iduće izrezane datoteke
fpath=${collection[count_fname]}
wav=".wav"

sox $fpath wav_cut/${cut_counter$wav} trim $startt $endt

#namještanje vrijednosti inkrementa za pretraživanje polja; pokazivači na
vrijednosti u polju
    ((count_startt+=3))
    ((count_endt+=3))
    ((count_fname+=3))
    ((cut_counter+=1))

done
```

Skripta 7.2. Prikazana je druga od 4 skripte: cut.sh.

7.2.1. Opis skripte *cut.sh*

Sed je alat koji se koristi za manipulaciju tekstem, a poznat je po filtriranju teksta metodom *pipelining-a*. *Sed* vrši samo jedan prijelaz (engl. *passover*) preko ulazne datoteke. Parametar *-i* se koristi za čuvanje originalne datoteke radom na kopiji. Bez njega bi se u slučaju pogreške pobrisao cijeli sadržaj. Parametar *-r* uključuje opciju *extended regular expressions*.

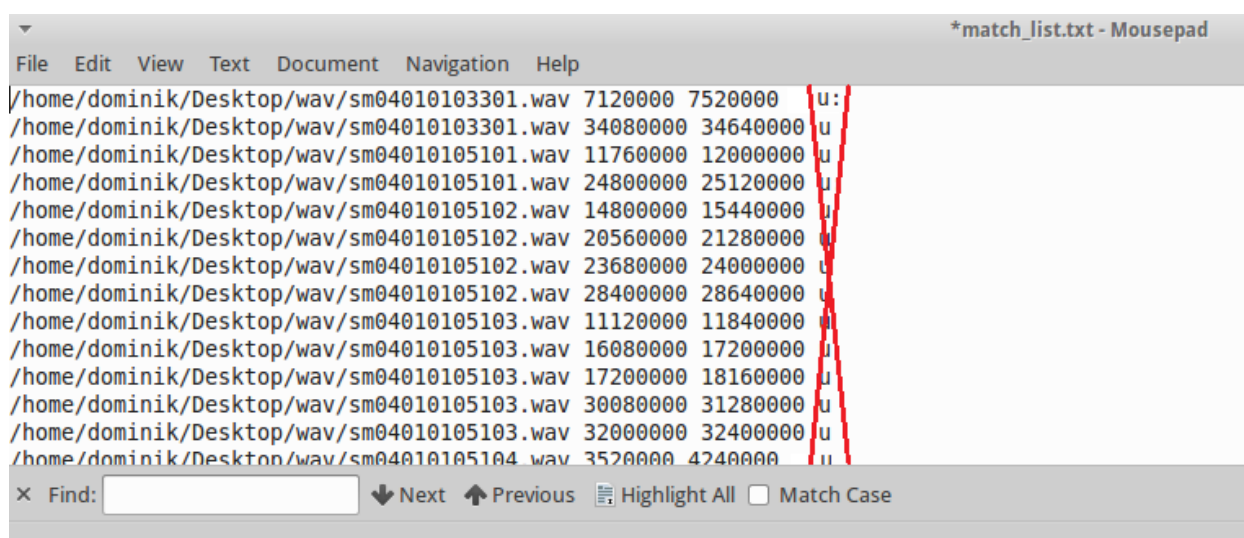
```
sed-i      s/://'      '/'
results.txt

sed-i      's/\\lab\\/\\wav/'
results.txt

sed-i      s/\\.lab\\.wav/
results.txt
```

U ovom se slučaju pomoću njega modificira zapis *LAB* datoteka na način da se:

- 1.) briše znak *'* koji se pojavljuje spremanjem *grep* izlazne vrijednosti.
- 2.) transformira ime pretraživanog direktorija u *WAV* (koji sadrži originalne *WAV* datoteke baze *VEPRAD*), jer se u sljedećem koraku obrađuju *WAV* datoteke za koje je prethodno preko *LAB* transkripcija utvrđeno da sadrže traženi vokal,
- 3.) mijenja ekstenzija *.lab* → *.wav*,
- 4.) briše 4. stupac podataka u *statistics/match_list.txt*, kako je prikazano na slici 7.1.



Slika 7.1. Efekt naredbe *sed-i -r 's/AS+//4'* *results.txt*. Uklanja se posljednji stupac.

U slijedećoj naredbi u varijablu *collection* pohranjujese 3 stupca iz datoteke *statistics/match_list.txt*: ime WAVuzorka, vrijeme početka trajanja traženog vokala u toj datoteci i vrijeme završetka trajanja istog.

```
collection=$(cut -d ' ' -f-3 results.txt)
```

Wc-w naredba se koristi prebrojavanje odvojenih riječi unutar polja, odnosno *stringova* koji nisu odvojeni razmakom.

Dobivena vrijednost označava kraj datoteke koja se pretražuje i se pohranjuje u varijablu *END*.

Za pretvorbu *nanosekunda* u *sekunde* potrebno je decimalne vrijednosti trajanja vokala pomnožiti sa 10^{-9} . S obzirom da *bashu* svojim osnovama ne podržava računanje sa decimalnim brojevima, potrebno je koristiti unutar *cmd-a* naredbu *bc-l*.

```
read startt <<<$(echo "${collection[count_startt]}"*10^-09 | bc -l)
```

Kratice *abc* (engl. *arbitrary precision calculator language*) zapravo označava programski jezik koji podržava između ostalog i decimalne brojeve, a se u *bashu* može koristiti i jednostavnim pozivom, kao u gornjem primjeru. Uključivanjem *-l* parametra poziva se knjižnica za matematičke izračune (engl. *mathlib*).

```
sox $fpathwav_cut/$cut_counter$wav trim $startt $endt
```

SoX alat se ovom naredbom koristi za „izrezivanje“ dijelova WAVdatoteka. Varijabla *fpath* označava put do originalne WAVdatoteke, a idući parametar označava odredište izlazne datoteke. Varijable *startt* i *endt* definiraju dio koji će se izrezati.

7.3. Izračun fundamentalne frekvencije za pojedinačni uzorak

Slijedeće što dolazi na red jest dobivanje niza vrijednosti aproksimiranih fundamentalnih vrijednosti, za svaki uzorak pojedinačno, kako je vidljivo na prikazu skripte 7.3. Dobivene vrijednosti za svaki pojedini uzorak spremaju se u istoimene datoteke sa *pitch* ekstenzijom u zaseban direktorij *pitch*.

```
#!/bin/bash

#postavlja se putanja puta do direktorija izrezanih wav datoteka
PATH_2="/home/bezevan/Desktop/lab/wav_cut/*wav"

pitch=".pitch"
txt=".txt"

rm -f -r pitch
rm -f -r pitch/txt
mkdir pitch
mkdir pitch/txt
declare j=0

#za svaku pronadjenu datoteku aproksimira se niz vrijednosti f0, koje se
spremaju za svaki uzorak zasebno
for i in $PATH_2
do
#fname putanja se modificira na način da se uklanja absolutni put s ciljem
da ostane samo ime datoteke
  fname=$(sed 's|/home/bezevan/Desktop/lab/wav_cut/||' <<< $fname)
  fname=$(sed 's/.wav/.pitch/' <<< $fname)
  fname=$i
  txtname=$fname
  txtname=$(sed 's/.pitch/.txt/' <<< $txtname)

#pitch naredba koja izvršava izračun
  x2x +sf $i | pitch -a 0 -s 16 -p 80 -L 50-H 400 -o 1 >pitch/$fname

#izlazne vrijednosti pohranjene u .pitch datoteku spremaju se u tekstualnu
  dmp +f pitch/$fname >pitch/txt/$txtname
done
```

Skripta 7.3. Prikazana je treća od 4 skripte: calc.sh.

7.3.1. Opis skripte *calc.sh*

Naredne dvije linije koda mijenjaju brišu absolutnu putanju datoteke, ostavljajući samo njen naziv, te mijenjaju ekstenziju *.wav* u *.pitch*.

```
fname=$(sed 's|/home/bezevan/Desktop/lab/wav_cut/||' <<< $fname)
fname=$(sed 's/.wav/.pitch/' <<< $fname)
```

Iz istog se razloga rade i nužni koraci u kojima se imenuje izlazna datoteka naredbe *pitch*, koristeći varijablu *i* (inkrement *for* petlje koji sadrži ime datoteke) koja se modificira i sprema u *txtname*.

```
txtname=$fname
txtname=$(sed 's/.pitch/.txt/' <<< $txtname)
fname=$i
```

S obzirom da *pitch* naredba ne radi sa WAV datotekama, potrebno je *x2x* naredbom iste pretvoriti u decimalni format (engl. *float*).

Pitch naredba poziva *SPTK* knjižnicu za izračun „vrhova“ analiziranog signala, tražeći namještanje nužnih parametara: *-a* → odabir algoritma za izračun, *-s* → frekvencija uzorkovanja, za ljudski glas 16kHz, *-p* → veličina okvira uzorka, *-o* → način ispisa.

Izračun spremljen u *pitch* datoteku potom se *dmp +f* naredbom pretvara u zapis prikladan za spremanje u tekstualnu datoteku, radi lakše daljnje manipulacije.

```
x2x +sf $i | pitch -a 0 -s 16 -p 80 -L 50-H 400 -o 1
>pitch/$fname
dmp +f pitch/$fname >pitch/txt/$txtname
```

7.4. Izračun ukupne prosječne fundamentalne frekvencije

Završni dio obrade uzoraka jest pregledavanje dobivenih aproksimacija pohranjenih unutar *pitch* direktorija, što je prikazano u segmentu koda skripte 7.4.

Pojedina pohranjena vrijednosti se uzima kao relevantna ukoliko je dobivena vrijednost veća od 0. U tom slučaju vrijednost se pribraja u sumu, pri čemu se brojač povećava za 1.

Nakon što se pregledaju svi zapisi, ukupna se suma dijeli za brojem pribrojanih vrijednosti te se, kao izlazna vrijednost, dobiva prosječna fundamentalna frekvencija traženog vokala.

```
#!/bin/bash

#putanja na pohranjene izlazne vrijednosti naredbe pitch
PATH_3="/home/bezevan/Desktop/lab/pitch/txt/*.txt"

typeset -i END k
let j=1 avg=0 sum=0 counter=0 value=0

#svaka .txt datoteka se čita, pritom se svaka vrijednost >0
#zbraja u sumu, i povećava se brojač
for i in $PATH_3
do

    line=$(wc -w $i)
    let END=$line
    collection=$(cut -d ' ' -f 2 $i)
    ((j+=1))
    let k=1

    while ((k<=END))
    do
        value=${collection[k]}
        read value <<<$(echo "$value" | bc -l)

        if [ $(echo "$value > 0" | bc) -eq 1 ]; then

            #povećavanje sume za trenutnu vrijednost >0
            read sum <<<$(echo "$sum"+"$value" | bc -l)
            ((counter+=1))

        fi

        #skakanje na sljedeću vrijednost
        ((k+=2))
    done

done

#izračun prosjeka
if [ "$counter" -ne "0" ]; then
    read avg <<<$(echo "$sum"/"$counter" | bc -l)
fi

echo $avg > statistics/average_f0_freq.txt
```

Skripta 7.4. Prikazana je četvrta od 4 skripte: avg.sh.

7.4.1. Opis skripte *avg.sh*

U varijablu *collection* se pohranjuje drugi stupac varijable *i*, a pritom je uključen *-d*, odnosno delimiter *'*'.

IF petlja provjerava ako je F_0 vrijednost veća od nule, te se samo u tom slučaju ona pribraja sumi, pri čemu se povećava brojač.

```
if [ $(echo "$value > 0" | bc) -eq 1 ]; then
    read sum <<<$(echo "$sum"+"$value" | bc -l)
    ((counter+=1))
fi
```

Druga *IF* petlja je osigurač koji provjerava ukoliko je brojač različit od 0, tse u tom slučaju računa prosjek.

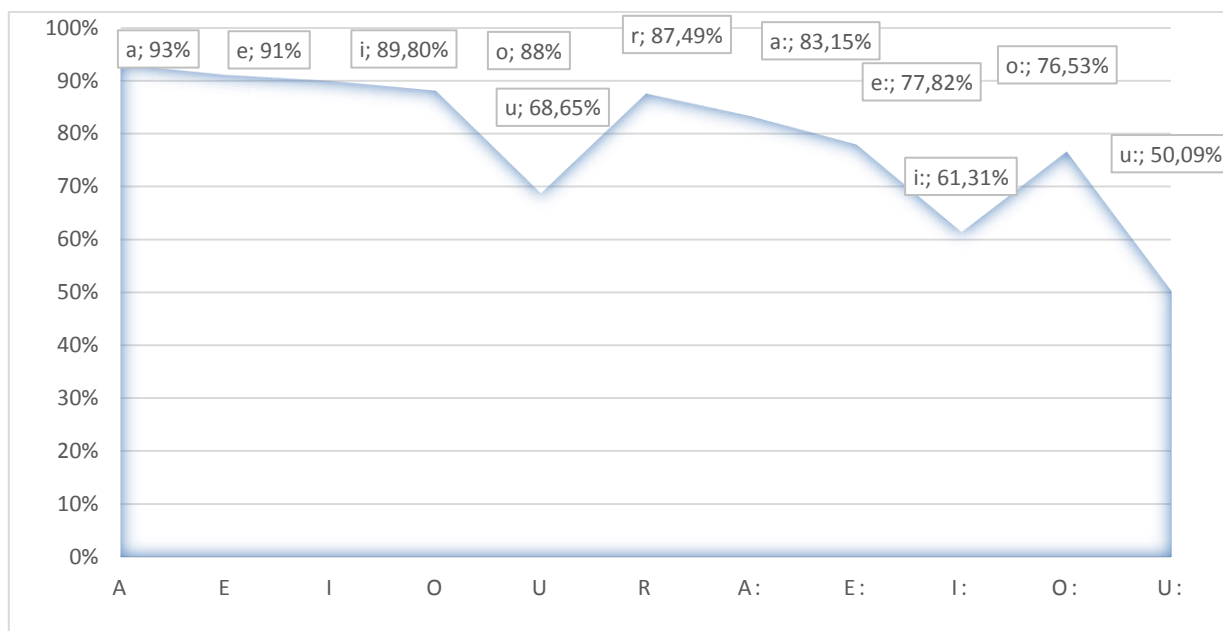
```
if ["$counter" -ne "0" ]; then
    read avg <<<$(echo "$sum"/"$counter" | bc -l)
fi
```

U jednoj i drugoj petlji koristi se *bc* –koji je jedna od nužnih opcija za manipulaciju decimalnim brojevima.

8. Rezultati izračuna

Vrijednosti fundamentalne frekvencije aproksimirati će se na zvučnim signalim baze *VEPRAD*. Vrijednosti će biti izračunate za za sve kratke samoglasnike (*a, e, i, o, u*), duge samoglasnike (*a:, e:, i:, o:, u:*), te za slogotvorni samoglasnik *r*, na način da će se iz svake *WAV*datoteke izrezati točno onaj dio zvučnog signala koji sadrži određeni vokal. U izračunu se koriste dva prethodno obrađena algoritma: *RAPT* i *SWIPE*.

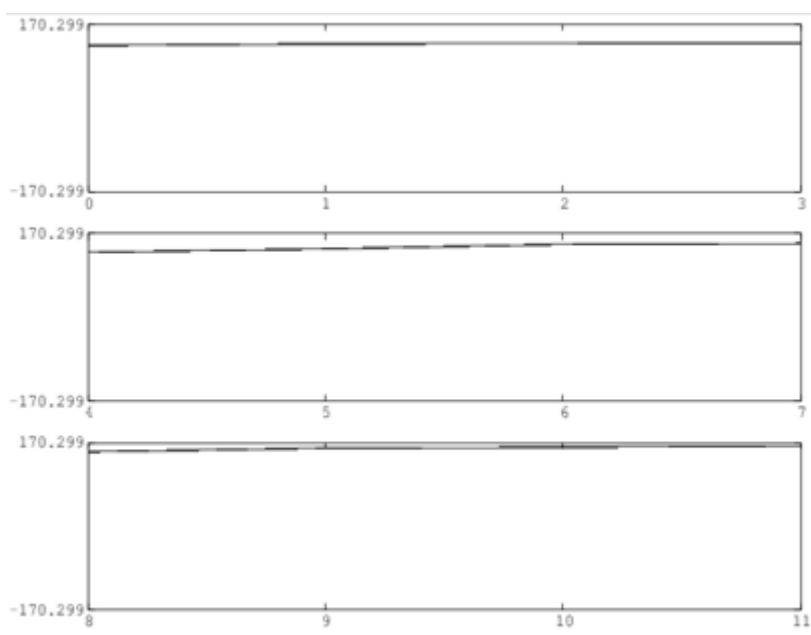
Idući dijagram, koji se nalazi na slici 8.1., predstavlja postotak uzoraka pojedinih vokala na ukupan broj analiziranih zvučnih zapisa baze *VEPRAD*. U nešto više od 3000 zvučnih zapisa najčešće se koristi samoglasnik *a*, a slijede ga redom *e, i, o*. Najrjeđe se koristi vokal *u*.



Slika 8.1. Učestalost pojedinih samoglasnika unutar analiziranih audio zapisa baze *VEPRAD*.

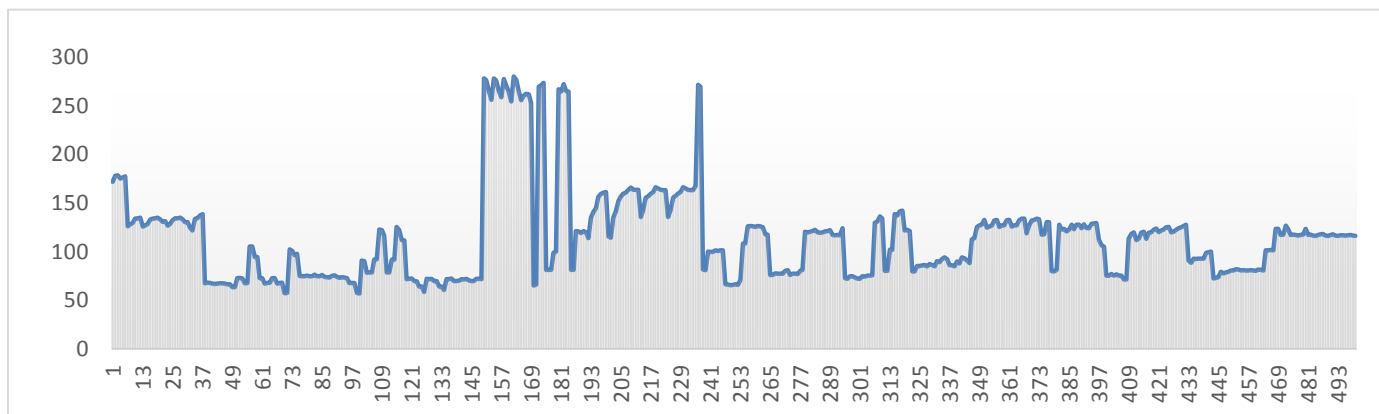
8.1. Izračun *RAPT* metodom

Izračun je prikazan dijagramima, tablicama i njihovim opisima. Pri izračunu vrijednosti odnosno potencijalnih *pitch* vrijednosti, *pitch* naredba ih treba pohraniti u izlaznu datoteku *primjer.pitch*. Za grafički prikaz iste može se upotrijebiti naredba *gwave* u kombinaciji sa *xgr* ili *psgr*, kao što je prikazano na slici 8.2.



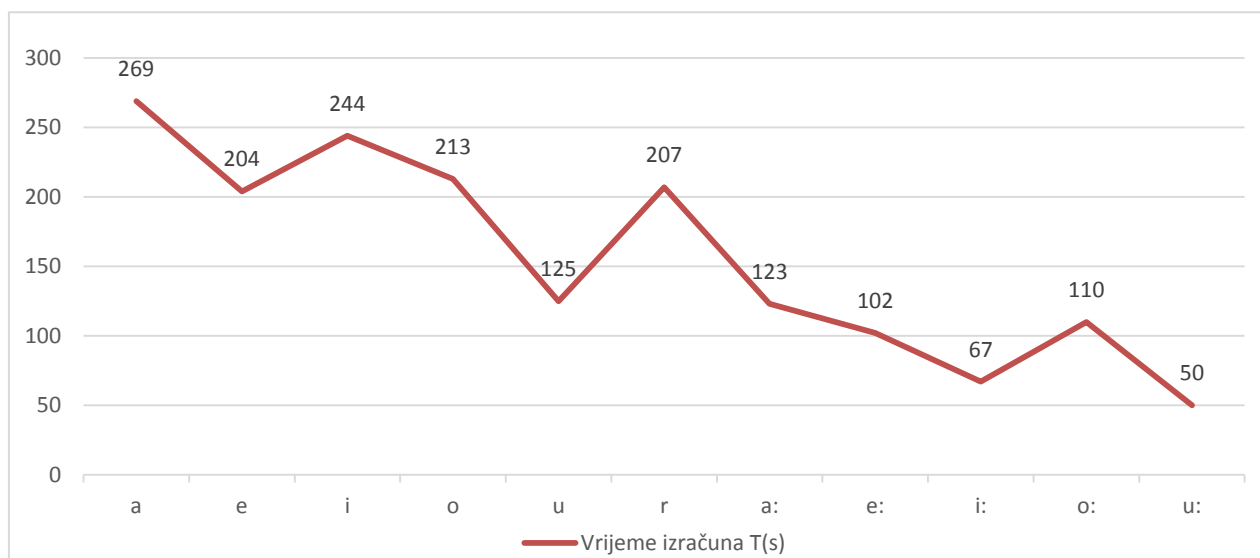
Slika 8.2. Grafički prikaz izlaznih rezultata jedne od datoteka koristeći *gwave* i *psgr* naredbu.

Na slici 8.3.prikazan je niz od prvih 500 izračunatih vrijednosti koristeći *RAPT* metodu. Prosječna F_0 odrasle osobe jest do 300 Hz, što znači da se vrijednosti veće od navedene mogu odbaciti.



Slika 8.3. Linijski prikaz prosječnih RAPT izračuna F_0 frekvencije za glas *a*. Prikazan je uzorak od prvih 500 rezultata zbog bolje preglednosti.

Izračuni vremenski ovise o tipu korištenog algoritma te o broju uzoraka koje analiziraju. Na slici 8.4. se nalazi dijagram trajanja izračuna F_0 pojedinih samoglasnika. Vremena izračuna podrazumijevaju vremensko trajanje petlje u kojoj se *pitch* naredbom obrađuju svi uzorci traženog samoglasnika i pohranjuju dobiveni rezultati. Najdulje je trajao izračun za samoglasnik *a*, sukladno najvećem broju uzoraka.



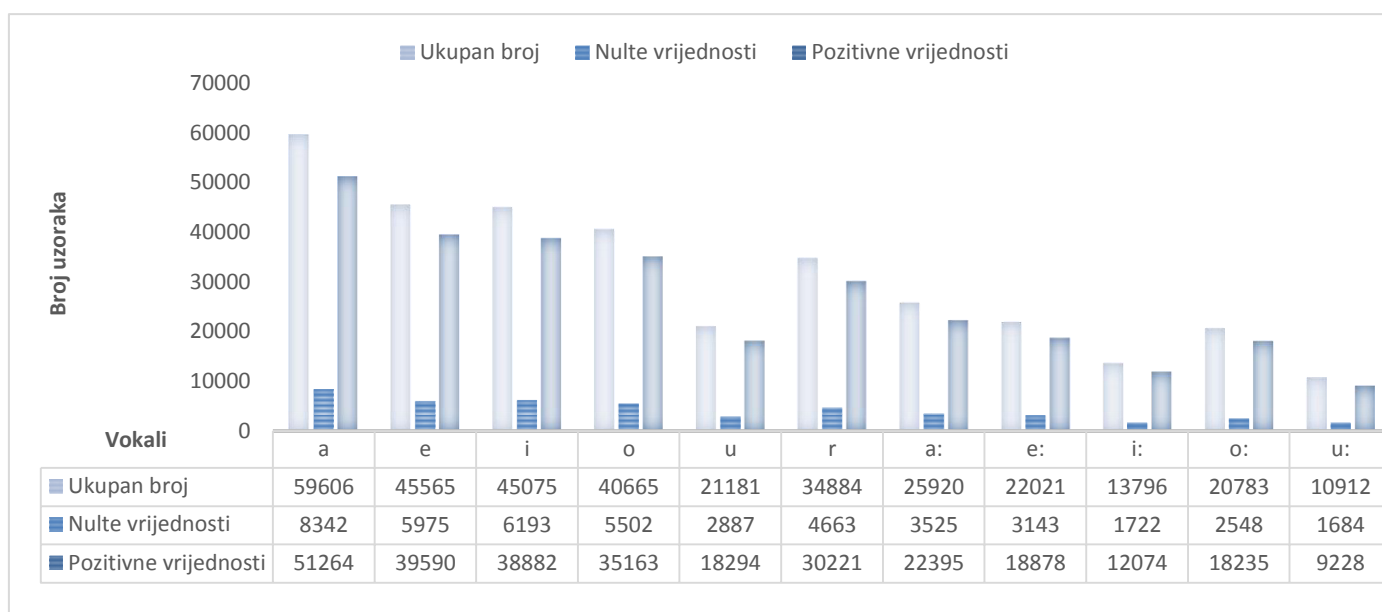
Slika 8.4. Vremena izračuna RAPT algoritma za pojedini glas.

U tablici 8.1. nalaze se prosječne izračunate fundamentalne frekvencije samoglasnika dobivene izračunom. Svaka od dobivenih vrijednosti za pojedini vokal rezultat je prosjeka svih dobivenih vrijednosti većih od 0. To znači da se u prosjek ubrajaju vrijednosti vokala na različitim dijelovima izgovorene riječi tj. različitim naglasaka, i od strane različitih govornika.

Tablica 8.1. Prikaz prosječne F_0 koristeći koristeći algoritam RAPT.

	<i>a</i>	<i>e</i>	<i>i</i>	<i>o</i>	<i>u</i>	<i>r</i>	<i>a:</i>	<i>e:</i>	<i>i:</i>	<i>o:</i>	<i>u:</i>
F_0 /Hz	104,4128	104,3845	104,8693	105,5924	105,4474	104,5304	107,5692	102,7312	104,3702	105,7649	101,4240

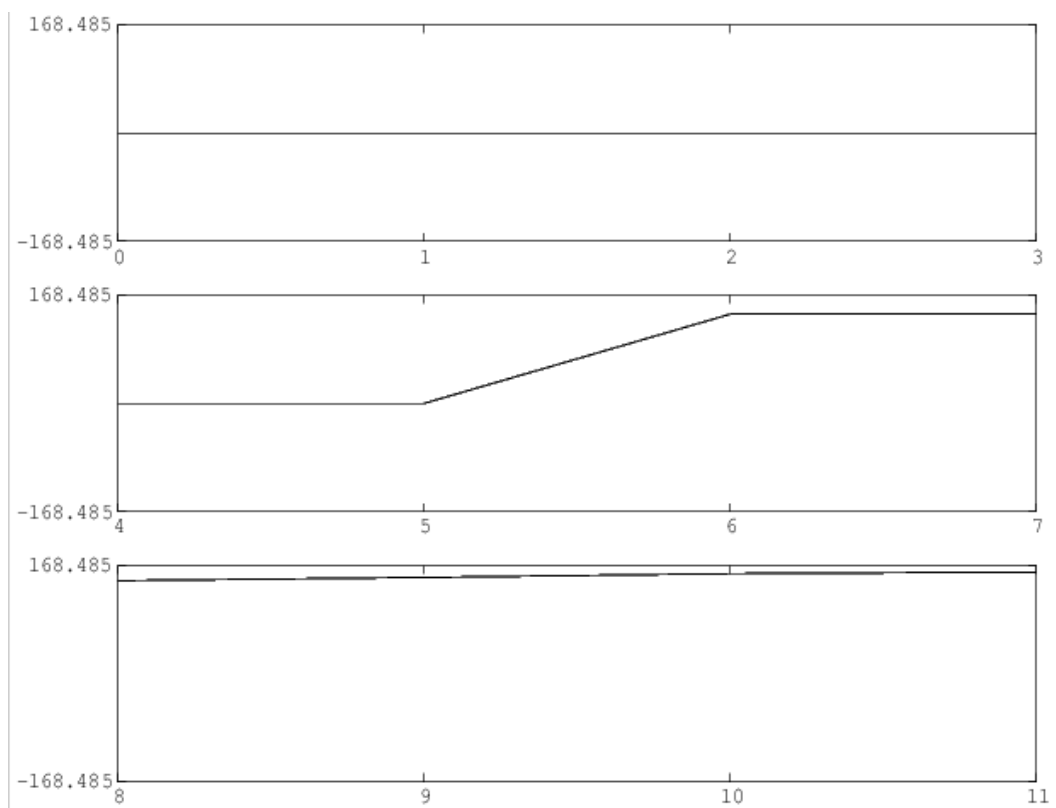
Na slici 8.5. prikazani su redom:ukupan broj izračunom dobivenih vrijednosti, broj pojedinačnih izračuna većih od 0 i broj pojedinačnih izračuna jednakih nuli.Kao relevantni podatci u izračunu koriste se jedino vrijednosti veće od nule. Nulte vrijednosti se ne uzimaju u obzir pri izračunu.



Slika 8.5. Grafički prikaz podataka analiziranih uzoraka dobivenih algoritmom RAPT i tablica vrijednosti. Prikazan je ukupan broj izračuna za pojedini glas, broj nultih vrijednosti te pozitivnih vrijednosti.

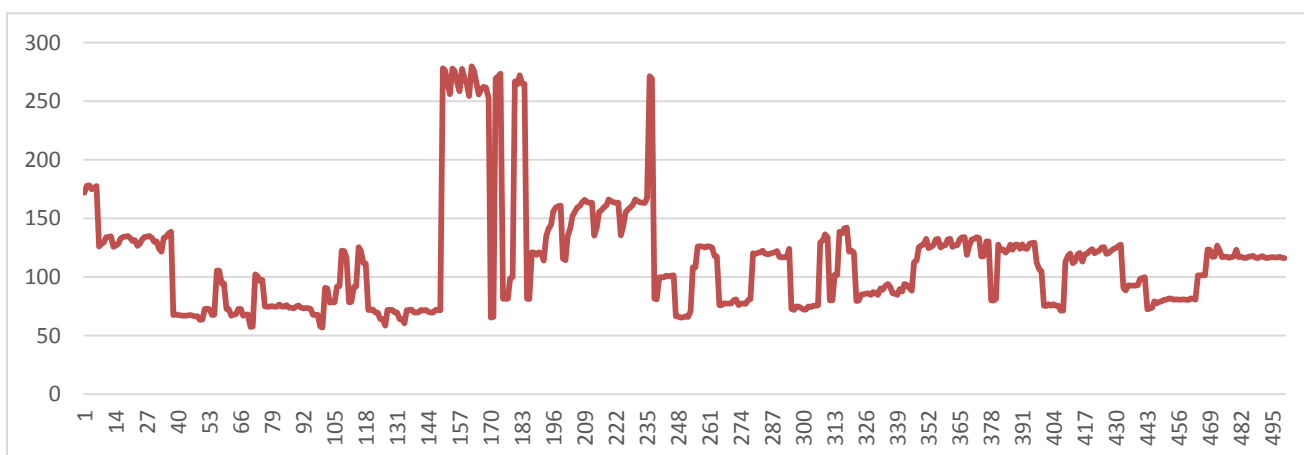
8.2. Izračun *SWIPE* metodom

Na slici 8.6. vidi se prikaz izlaznih rezultata naredbe *pitch* za datoteku na isti način prikazanu u prethodnom potpoglavlju.



Slika 8.6. Grafički prikaz izlaznih rezultata jedne od datoteka koristeći *gwave* i *psgr* naredbu.

Na grafu prikazanom na slici 8.7. nalazi se linijski prikaz rezultata za vokal *a* dobivenih *SWIPE* analizom, kao i u prethodnom potpoglavlju za *RAPT*.



Slika 8.7. Linijski prikaz prosječnih *SWIPE* izračuna F_0 frekvencije za glas *a*. Prikazan je uzorak od prvih 500 rezultata zbog bolje preglednosti.

Razlika u kodu koji poziva *RAPTi SWIPE* se očituje jedino u parametru $-a$, koji je za *RAPT* jednak 0, odnosno za *SWIPE* jednak 1.

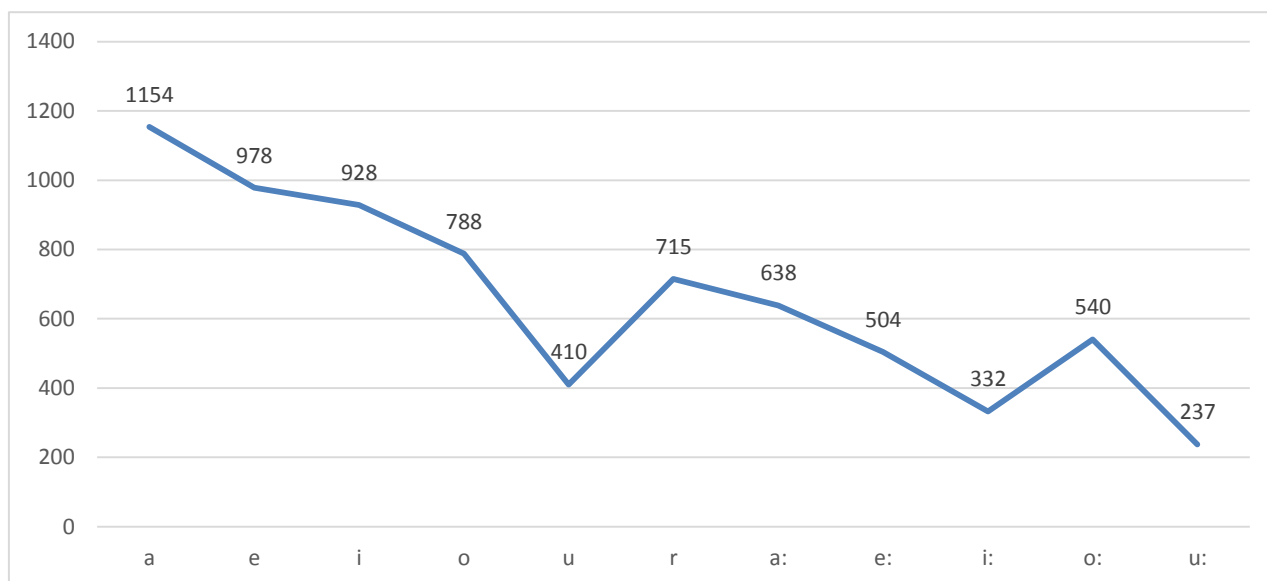
x2x +sf **Si**pitch -a [A=1 || A=0] -s 16 -p 80 -L 50-H 400 -o 1>pitch/**Sfnam**

Prosječne fundamentalne frekvencije dobivene *SWIPE* metodom nalaze se u tablici 8.2. Razlike u odnosu na *RAPT* rezultate bit će komentirane u idućem potpoglavlju.

Tablica 8.2. Prikaz prosječnih F_0 samoglasnika dobivenih algoritmom *SWIPE*.

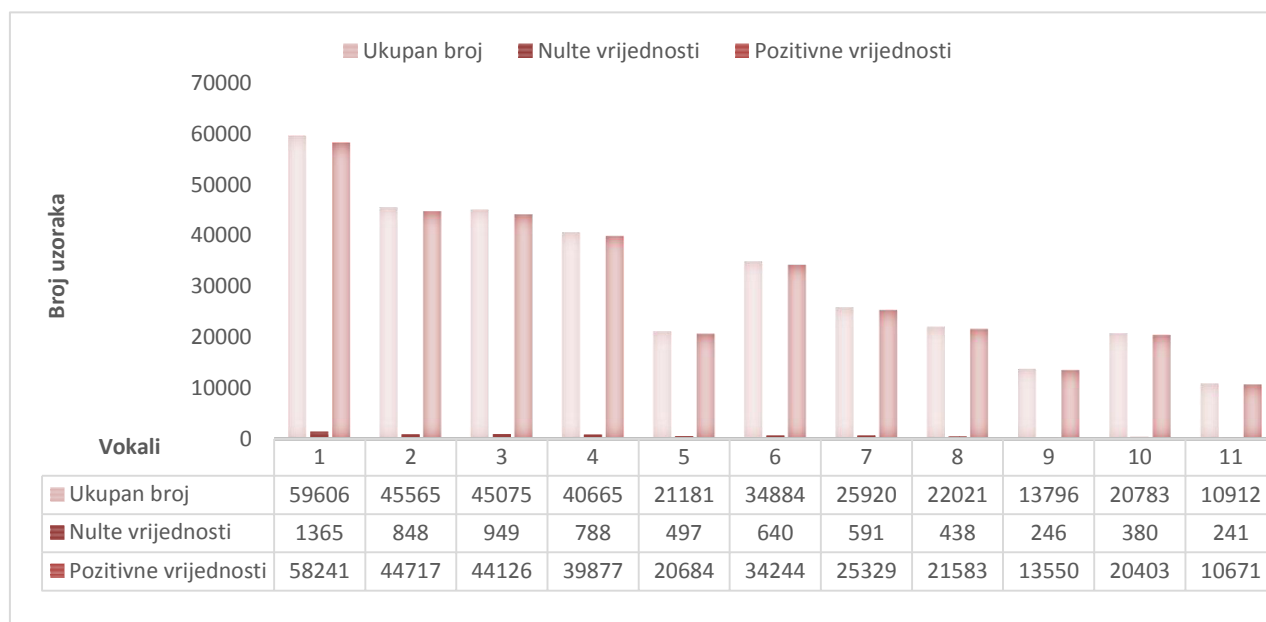
	<i>a</i>	<i>e</i>	<i>i</i>	<i>o</i>	<i>u</i>	<i>r</i>	<i>a:</i>	<i>e:</i>	<i>i:</i>	<i>o:</i>	<i>u:</i>
F_0/Hz	128,2186	124,7274	124,9357	126,4599	123,2329	125,4452	126,0400	124,0184	126,5109	131,1598	124,4689

Vremena izračuna algoritma prikazana su u donjem grafu, na slici 8.8. Izračuniraju dulje u odnosu na vremena prethodnog algoritma.



Slika 8.8. Vremena izračuna *SWIPE* algoritma za pojedini glas

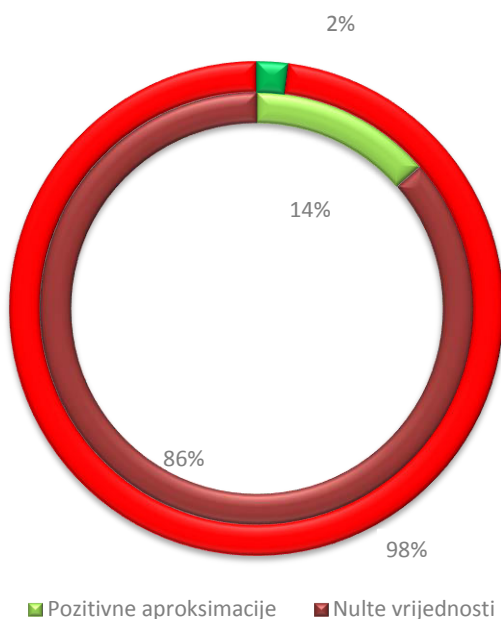
U sljedećem prikazu koji se nalazi na slici 8.9. vidi se kako je broj relevantnih, pozitivnih vrijednosti, manji u odnosu na ukupan broj dobivenih *RAPT* algoritmom. Većinske su neuporabljive, nulte vrijednosti.



Slika 8.9. Prikaz podataka analiziranih uzoraka dobivenih algoritmom *SWIPE*. Prikazan je ukupan broj izračuna za pojedini glas, broj nultih vrijednosti te pozitivnih vrijednosti.

8.3. Usporedba

Na slici 8.10. se mogu vidjeti omjeri efikasnosti dvaju algoritama dobiveni na temelju postotka pozitivnih vrijednosti u odnosu na ukupan broj izračunom dobivenih vrijednosti. *SWIPE* algoritam sa samo 2% pozitivnih vrijednosti dosta zaostaje za 14% vrijednosti dobivenih *RAPT* analizom. Većinske nulte vrijednosti mogu se zanemariti.



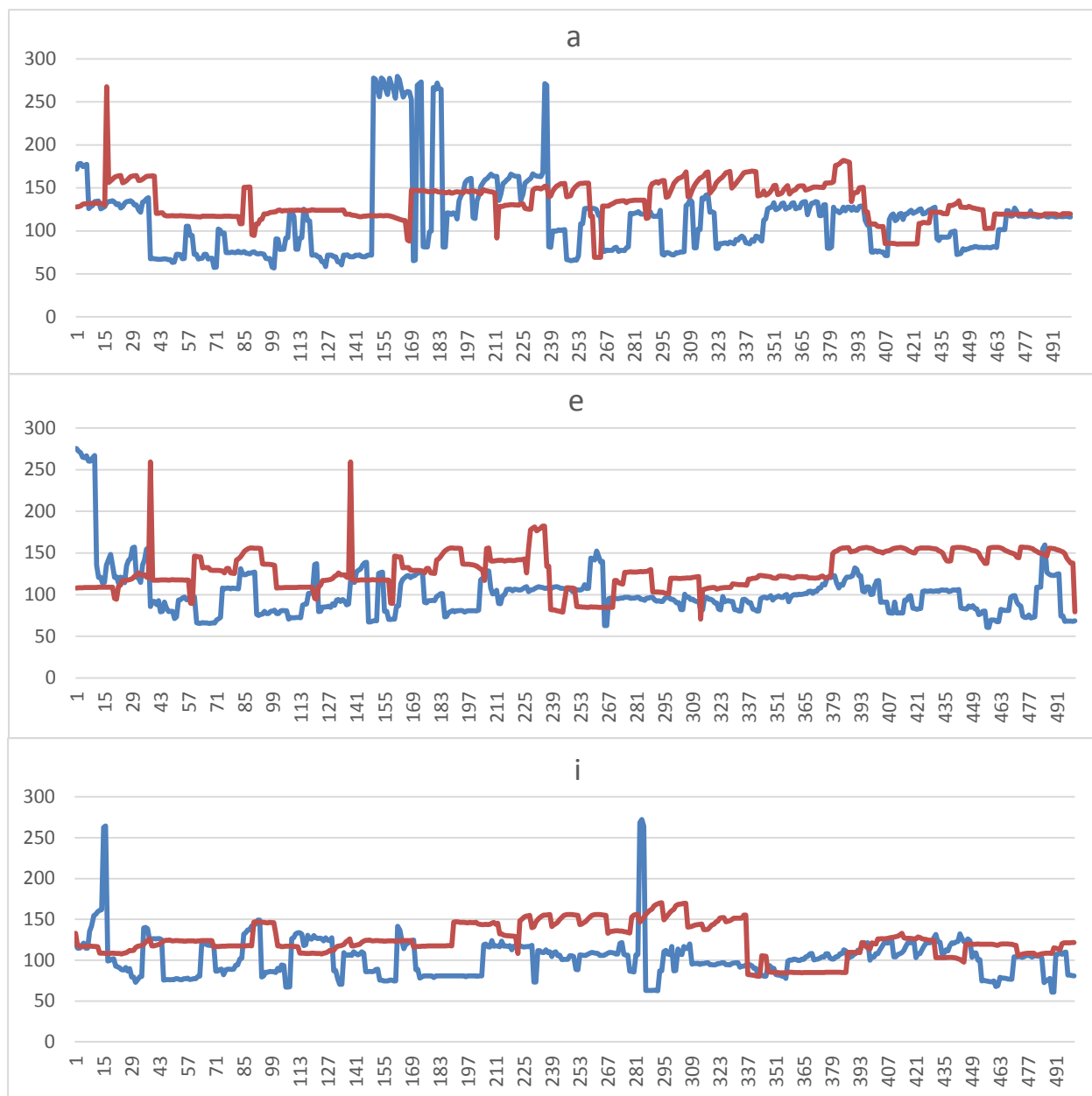
Slika 8.10. Usporedni prikaz efikasnosti *RAPT* i *SWIPE* algoritma. Vanjski krug odnosi se na *SWIPE*, a unutarnji se odnosi na *RAPT*.

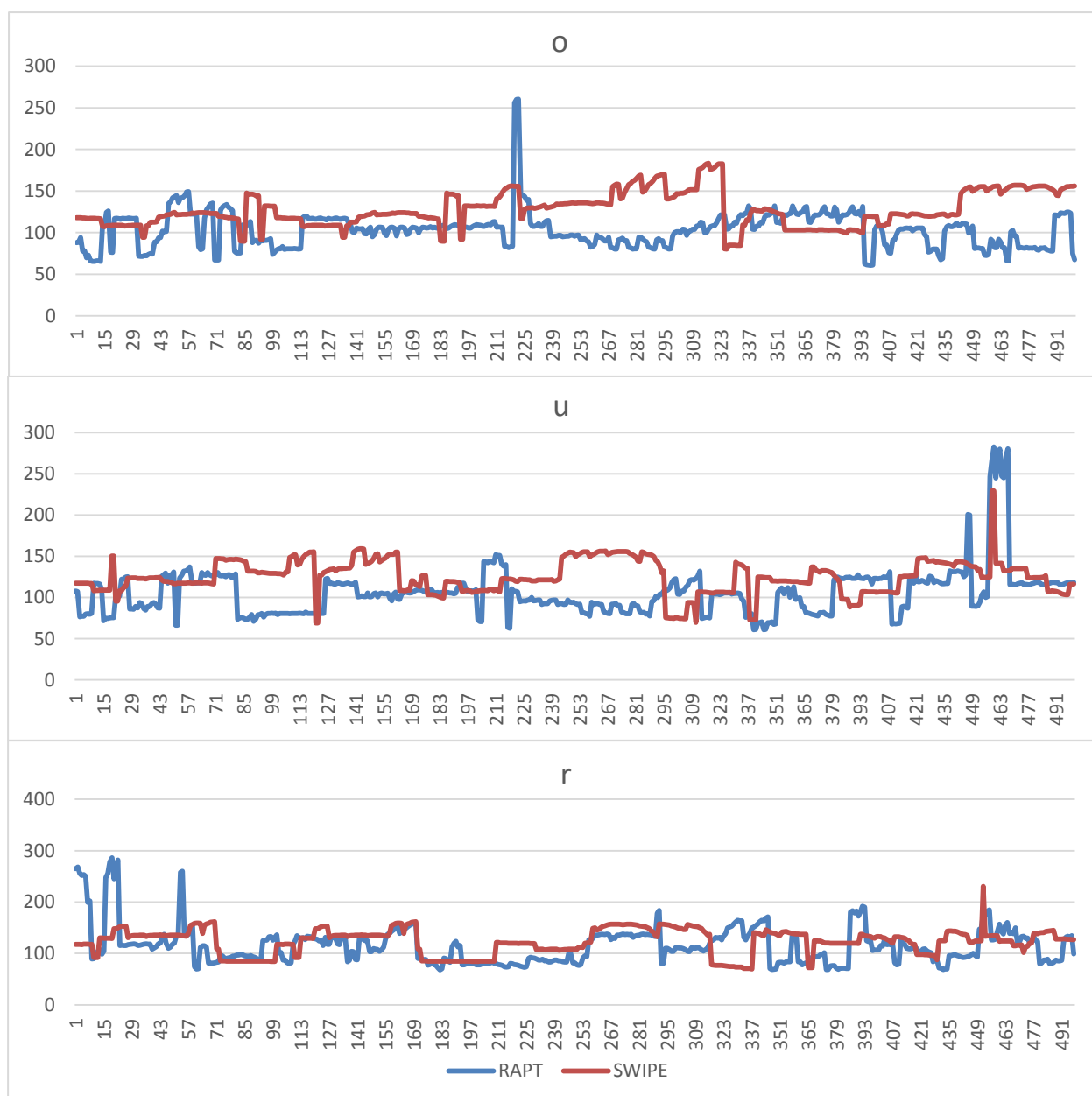
Različiti omjeri efikasnosti, odnosno drugačiji princip rada algoritama, jedan su od razloga zbog kojih su i različite konačne prosječne aproksimacije F_0 . U tablici 8.3. prikazane su vrijednosti F_0 dobivene *RAPT* i *SWIPE* izračunom.

Tablica 8.3. Usporedba dobivenih F_0 dvaju algoritama.

Samoglasnici	<i>a</i>	<i>e</i>	<i>i</i>	<i>o</i>	<i>u</i>	<i>r</i>	<i>a:</i>	<i>e:</i>	<i>i:</i>	<i>o:</i>	<i>u:</i>
<i>RAPT</i>	104,41	104,38	104,87	105,59	105,45	104,53	107,57	102,73	104,37	105,76	101,42
<i>SWIPE</i>	128,22	124,73	124,94	126,46	123,23	125,45	126,04	124,02	126,51	131,16	124,47

U narednom spojenom prikazu, na grafovima prikazanima na slici 8.11, nalaze se grafički prikazi sa linijskim grafovima *RAPT* (označeno crvenom bojom) i *SWIPE* (označen plavom bojom) algoritama za pojedine kratke samoglasnike, uključujući i slogotvorni samoglasnik r.

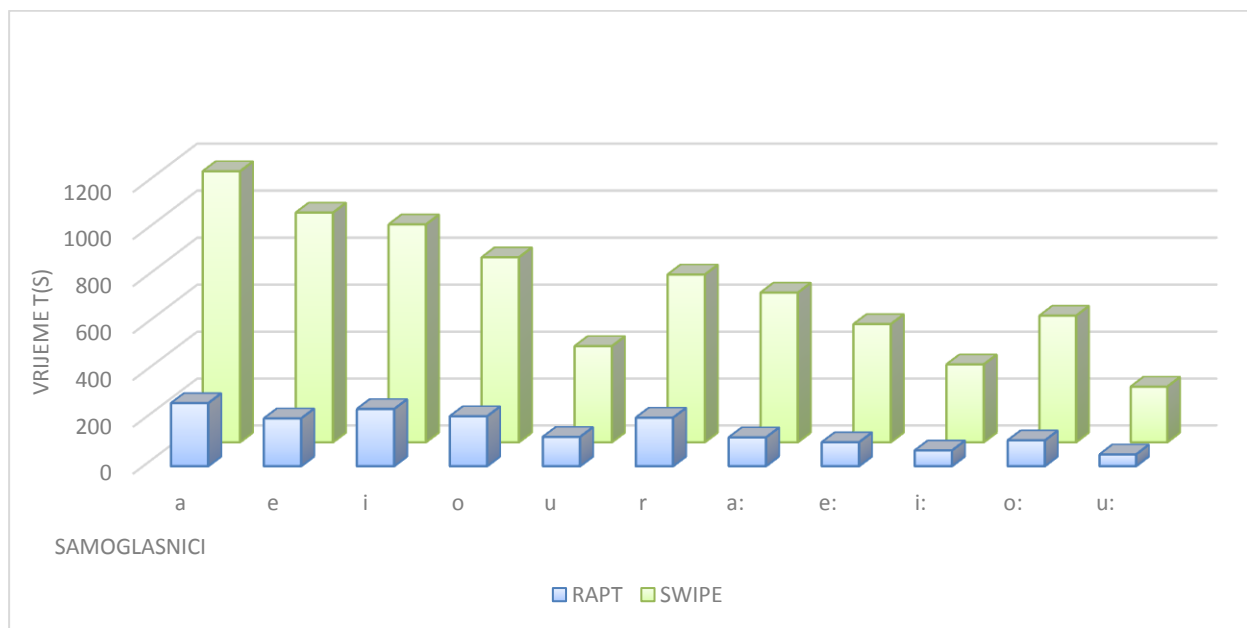




Slika 8.11. Linijski prikazi prosječnih RAPT i SWIPE izračuna F_0 frekvencije redom za glasove a, e, i, o u. Prikazan je uzorak od prvih 500 rezultata zbog bolje preglednosti.

Grafovi uspoređuju rezultate koje postižu algoritmi kroz pojedine aproksimacije. Međuodnosi tih dvaju algoritama kod pojedinih izračuna dugih samoglasnika daju isti zaključak, a to je da u prosjeku *SWIPE* daje stabilnije, ali pritom veće vrijednosti u odnosu na *RAPT* koji prosječno aproksimira niže vrijednosti, ali uz nešto veći broj odstupanja.

Uspoređujući vremena izračuna prikazana na slici 8.12, jasno se vidi da je *RAPT* vremenski višestruko efikasniji u odnosu na izrazito spori *SWIPE* algoritam.



Slika 8.12. Usporedni prikaz vremena trajanja izračuna *RAPT* i *SWIPE* algoritma.

RAPT izvodi klasifikaciju binarnih zvučnih signala ovisno o prisutnosti ili nedostatku zvučnosti unutar signala.

9. Zaključak

Tema rada bila je pripremno upoznavanje pojma fundamentalne frekvencije u području ljudskog glasa, te zatim rješavanje problema izračuna fundamentalne frekvencije.

Prethodno izračunima odnosno predstavljanju programskog koda, u prvim se nekoliko poglavlja predstavlja konstitucija ljudskih glasnica, vokalnog trakta te bit fundamentalne frekvencije u govoru.

Analiza i izračun prosječne vrijednosti fundamentalne frekvencije svakog od samoglasnika na ciljanim uzorcima uspješno je provedeno, pri čemu se sam izračun izvršio koristeći dva odvojena algoritma – *RAPT* i *SWIPE*. Pokazalo se kako je *RAPT* vremenski puno učinkovitiji, ali *SWIPE* u konačnici daje nešto stabilnije i točnije rezultate.

U narednom su se poglavlju dobiveni rezultati usporedili te je dan adekvatan zaključak.

Treba naglasiti važnost istraživanja ljudskog govora, kao i mogućnosti prepoznavanja govora kao širokog i kompleksnog područja, pri čemu je pojam fundamentalne frekvencije tek – *fundamentalan*, ali je pritom, bitan kao jedan od temelja. Otkrivanje pojma fundamentalne frekvencije polazna je točka za daljnje istraživanje i produbljivanje znanja na ovom području.

Literatura

- [1] Zavod za elektroničke sustave i obradbu informacija, FER [Online]:
<http://dog.zesoi.fer.hr/predavanja/HTML/Osnove%20procesa%20nastajanja%20govora.htm>
- [2] American Academy of Otolaryngology [Online]: <http://www.entnet.org/content/how-voice-works>
- [3] The University New South Wales [Online]: <http://newt.phys.unsw.edu.au/jw/voice.html>
- [4] Massachusetts Institute of Technology [Online]: <http://ocw.mit.edu/high-school/engineering/guitar-building/physics-of-the-guitar/how-strings-make-sound/>
- [5] The Drum Works [Online]: <http://www.thedrumworks.com/world-drums-1/cajon-drums/>
- [6] Behind the Mixer [Online]: <http://www.behindthemixer.com/why-harmonics-are-%20important-%20understand/>
- [7] Researchgate.net[Online]:
http://www.researchgate.net/post/What_is_the_difference_between_FFT_harmonics_and_LPC_formants11
- [8] Department of Computer & Information Science & Engineering, UF [Online]:
<http://www.cise.ufl.edu/~acamacho/publications/dissertation.pdf>
- [9] Columbia University [Online]: <http://www.ee.columbia.edu/~dpwe/papers/Talkin95-rapt.pdf>
- [10] SoX - Sound eXchange , homepage [Online]: <http://sox.sourceforge.net/Main/HomePage>
- [11] askubuntu.com: <http://askubuntu.com/questions/173088/what-does-configure-make-make-install-do>
- [12] The Undergraduate linguistics club at the Ohio State University [Online]:
<https://underlingsosu.wordpress.com/2013/03/08/phonetics-phriday-fundamental-frequency-harmonics-and-formant-frequencies/>
- [13] The University New South Wales
[Online]:<http://newt.phys.unsw.edu.au/jw/sound.spectrum.html>
- [14] Studentski seminar, FER [Online]:
<https://www.fer.unizg.hr/download/repository/dokumentacija%5B1%5D.pdf>