UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

HLASOM OVLÁDANÝ KURZOR MYŠI

Bratislava, Máj 2011 Bc. Ján Kolek

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

HLASOM OVLÁDANÝ KURZOR MYŠI

Diplomová práca

Štúdijný program: Kognitívna veda

Štúdijný odbor: 9.2.11 Kognitívna veda

Školiace pracovisko: Katedra aplikovanej informatiky

Zodpovedný vedúci: RNDr. Marek Nagy, PhD.

Evidenčné číslo: 206c466a-61d0-48ca-bfd8-8a346c266595

Bratislava, Máj 2011 Bc. Ján Kolek

Čestné prehlásenie

Týmto prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod odborným vedením vedúceho diplomovej práce a použil som uvedenú literatúru.

Bratislava, Máj 2011 Ján Kolek

Abstrakt

Táto diplomová práca sa zameriava na menej tradičné využitie spracovania digitálneho signálu a to hlasom ovládaný kurzor myši. Kurzor je ovládaný pomocou nepretržitého vyslovovania samohlások s možnosťou okamžitej zmeny smeru (samohlásky) bez potreby odmlky.

Práca obsahuje spracovanie dôležitého teoretického pozadia k danej téme a vlastný návrh a implementáciu riešenia, ako ovládať kurzor počítačovej myši pomocou samohlások. Rozpoznávanie týchto samohlások je riešené dvoma prístupmi a to pomocou diskrétnej Furierovej transformácie a pomocou LPC a využitia Durbinovej metódy na výpočet koeficientov reči. Tieto dve metódy využívame na výpočet spektrálnej obálky, ktorej charakteristické črty umožňujú rozpoznávanie samohlások.

Oba tieto postupy sa nám podarilo implementovať. Práca obsahuje podrobný popis použitých algoritmov v jazyku C a analýzu dosiahnutých výsledkov.

Kľúčové slová: vzorkovacia frekvencia, digitálny signál, formant, frekvenčné spektrum, spektrálna obálka

Abstract

This paper presents a less traditional usage of digital signal processing, the voice controlled mouse cursor. The cursor is controlled with continuous pronouncing of vowels with possibility of instant change of direction (vowel) without requirement of silence.

This work contains important theoretical background to theme and scheme and implementation of solution, how to control the mouse cursor with vowels. We used two approaches to recognize vowels. Linear predictive coding with usage of Durbin method and Discrete Fourier transform. We used these two methods, to calculate a spectral envelope of digital signal. This spectral envelope provides characteristic lines, which allow us the recognition.

We implemented both approaches successfully. Paper provides detailed characterization of algorithms in C language and analysis of reached results.

Key words: sampling frequency, digital signal, formant, frequency domain, spectral envelope

Poďakovanie

Týmto by som chcel vyjadriť vďaku môjmu odbornému vedúcemu diplomovej práce RNDr. Marekovi Nagyovi, PhD. z Katedry aplikovanej informatiky, za odborné vedenie, cenné rady a pomoc pri spracovaní uvedenej témy. Tiež ďakujem všetkým priateľom a príbuzným, ktorí ma podporovali v štúdiu a bez ktorých by nebolo možné.

Obsah

Úvod

Cieľom tejto práce je navrhnúť a implementovať program, pomocou ktorého budú ľudia schopní ovládať počítačový kurzor myši pomocou kontinuálneho vyslovovania samohlások.

Toto netradičné využitie spracovania digitálneho signálu sme sa rozhodli implementovať z niekoľkých dôvodov. Medzi ne patrí motivácia vytvoriť použiteľný program, ktorý by v budúcnosti mohol nájsť uplatnenie predovšetkým v radoch ľudí, ktorým fyzická alebo iná nespôsobilosť zabraňuje ovládať počítač bežným spôsobom, alebo pomocou iných alternatívnych riešení. Dôvodom bol tiež samotný obor rozpoznávania zvuku pomocou spracovávania digitálneho signálu, ktorého potenciál ešte stále nie je dnešnými technológiami vyčerpaný.

Rozpoznávanie samohlások v tejto práci je opísané a implementované pre slovenské samohlásky. Toto rozpoznávanie vykonávame pomocou vyhľadávania formantov v spektrálnej obálke frekvenčného spektra samohlásky.

Zvolili sme dva prístupy, ktorými je možné vypočítať spektrálnu obálku. Tieto dva prístupy sú diskrétna Furierová transformácia (DFT) a lineárne prediktívne kódovanie (LPC). Pri LPC je tiež potrebné vypočítať koeficienty rečového signálu. Na tento výpočet sme použili Durbinovú metódu.

Metódu rozpoznávania samohlások pomocou vyhľadávania formantov sme zvolili kvôli jej univerzálnosti. Hranice frekvencií v ktorých sa dôležité formanty nachádzajú sú rovnaké pre každého a nie je teda potrebné úvodné trénovanie programu na konkrétneho človeka.

V teoretickej časti práce je detailne popísaný vznik reči u človeka ako aj teória prevodu analógového signálu na diskrétny digitálny signál a jeho spracovanie pomocou vyššie uvedených metód. Práca tiež obsahuje matematické definície pre DFT a LPC.

V druhej časti práce predstavujeme návrh implementácie a samotnú implementáciu. Je tu podrobne popísaný beh programu ako aj jednotlivé metódy spolu s ukážkami kódu.

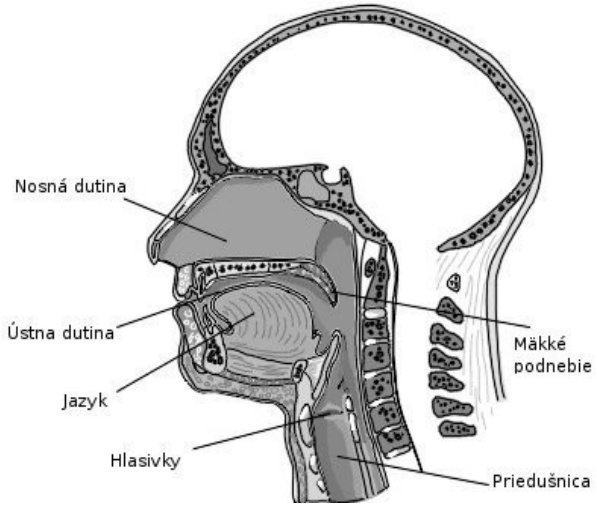
V závere predstavujeme výsledky ku ktorým sme sa dopracovali. Tie sú v podobe výstupov programu, ktoré predstavujú vypočítané spektrálne obálky pre jednotlivé samohlásky a experiment, ktorým sme testovali úspešnosť a použiteľnosť ovládania kurzoru pomocou nášho programu na skupine respondentov.

1. Proces vytvárania reči človekom

Nezávisle na tom akou rečou človek hovorí, používa vždy rovnakú anatómiu na vytváranie zvukov. Výsledok produkovaný ľudskou anatómiou má samozrejme svoje limity, ohraničené predovšetkým zákonmi fyziky.

„Fyzikálnou reprezentáciou reči u človeka sú rečové kmity. Ich zdrojom sú ľudské rečové orgány. Tie sa skladajú z hlasiviek, ústnej, hrdelnej a nosnej dutiny, mäkkého a tvrdého podnebia, zubov a jazyka. K týmto orgánom je potrebné zaradiť aj zdroj hlasovej energie a teda pľúca a s nimi späté svaly.“ [1]

Ak chceme zjednodušene definovať proces vytvárania zvukov reči u človeka, môžeme povedať, že … „Proces produkcie reči u ľudí je vzduch vytláčaný z pľúc, ktorý následne prechádza rečovým traktom (obr. č. 1) a ústami. Pri tomto type definície môžeme chápať pľúca, ako zdroj zvuku a rečový trakt, ako prostriedok alebo filter na vytváranie rôznych variácií zvukov, ktoré nám umožňujú vytvárať reč. “ [3]



Obr. č. 1: Ľudský rečový trakt. [3]

Pre lepšie pochopenie vytvárania zvukov, a následne reči, rečovým traktom, je potrebné zaviesť niekoľko definícii. Najmenšou jednotkou reči sú fonémy. Fonémy je možné od seba odlíšiť napríklad podľa miesta alebo podľa artikulujúceho orgánu pomocou ktorého daná fonéma vzniká, alebo podľa výsledného dojmu. „Fonémy sú definované ako množina samostatných zvukov. Poznáme dva typy foném a to znelé a neznelé zvuky.“ [3] Toto rozdelenie a vlastnosti znelých a neznelých foném má veľký dopad aj na spôsob, akým sa k ním pristupuje počas rozpoznávania jednotlivých slov, alebo ich častí, počítačom.

„Fonologické výskumy ukázali, že v existujúcich svetových jazykoch je aktívne využívaných iba asi dvanásť diferenciálnych príznakov. Tento fakt je možné objasniť fyziológiou rečového traktu, ktorý je schopný svojimi artikulačnými orgánmi vytvoriť okolo dvanásť rôznych polôh pri vytváraní rôznych hovorených výrazov. Počet foném v existujúcich svetových jazykoch sa pohybuje od 12 do 60. Napríklad v anglickom jazyku je 42 foném, v ruskom 40 a podobne.“ [1]

Znelé zvuky v reči sú väčšinou samohlásky, ktoré vznikajú vytláčaním vzduchu z pľuc cez hlasivky: „Zdrojom všetkých znelých zvukov sú kmitajúce hlasivky. Frekvencia kmitania hlasiviek je rozdielna u detí a dospelých ale tiež aj u mužov a žien. Táto frekvencia sa väčšinou pohybuje v rozmedzí 150 – 400 Hz a zároveň udáva základný tón ľudského hlasu.“ [1] Frekvencia, ktorou hlasivky kmitajú určuje aj výšku produkovaného zvuku. „Tento vzduchový stimul produkovaný kmitaním hlasiviek následne prejde aj zvyškom rečového traktu, v ktorom niektoré frekvencie rezonujú. Je všeobecne známe, že ženy a deti majú vyššie položené hlasy ako muži, čo je výsledkom rýchlejšej frekvencie kmitania hlasiviek počas produkcie znelých zvukov. Je teda dôležité zahrnúť výšku periódy pri analýze a syntéze reči ak očakávame, že výsledný produkt bude dostatočne presne reprezentovať vstupný signál“ [3]

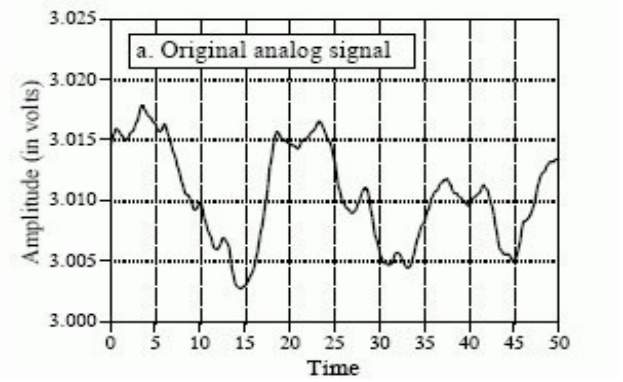
Neznelé zvuky sú obyčajne spoluhlásky. Oproti znelým zvukom, teda samohláskam, majú menšiu energiu (hlasitosť) a nachádzajú sa na vyšších frekvenciách frekvenčného spektra. „Vytváranie neznelých zvukov zahŕňa prúd vzduchu vytláčaný rečovým traktom vo vírivom prúde. Počas tohto procesu hlasivky nekmitajú, namiesto toho ostávajú otvorené pokým nie je zvuk vytvorený. Výška výsledného zvuku je v prípade neznelých zvukov nepodstatná, keďže nedochádza k vibráciám hlasiviek.“ [3]

Rozdelenie foném na znelé a neznelé má veľký význam z hľadiska analýzy a syntézy zvuku. Môžeme povedať, že vibrácie hlasiviek sú jedným z kľúčových artiklov pri vytváraní jednotlivých typov zvukov, pomocou ktorých ľudia tvoria reč. Výslednú podobu zvukov ovplyvňujú aj niektoré iné faktory, ako sú napríklad tvar rečového traktu, alebo množstvo vzduchu ktoré je Človek schopný vytlačiť z pľúc. „Vzduch prúdiaci z pľúc môžeme považovať za zdroj pre rečový trakt, teda filter, pomocou ktorého sa vytvára reč. [3]

2. Spracovanie signálu

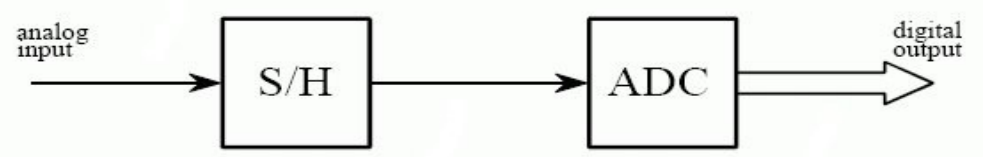
2.1. Prevod analógového signálu na digitálny

Prevod analógového signálu (zvuku) na digitálny signál sa realizuje jeho digitalizáciou. Táto digitalizácia sa deje prostredníctvom tzv. ADC (z anglického analog-to-digital converter), čiže analógovo digitálnych prevodníkov, ktoré sa nachádzajú na zvukových kartách počítačov. Môžeme povedať, že prevod analógového signálu na digitálny sprevádzajú dva základné javy a to vzorkovanie a kvantovanie.



obr. č. 2: Originálny analógový signál [2]

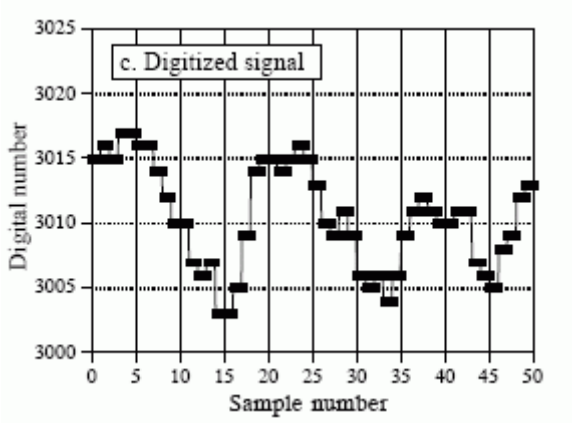
Na obrázku č. 2 je znázornený originálny analógový signál znázornený ako amplitúda napätia v čase. Na zjednodušenie vysvetlenia digitalizácie predpokladajme, že maximálna amplitúda tohto signálu je 4.095 voltov a použijeme 12 bitový digitizér. Takto môže jednotlivé amplitúdy tohto analógového signálu ľahko priradiť digitálnym hodnotám medzi 0 až 4095, ktoré poskytuje 12 bitový digitizér.



Obr. č. 3: Konverzia analógového signálu na digitálny [2]

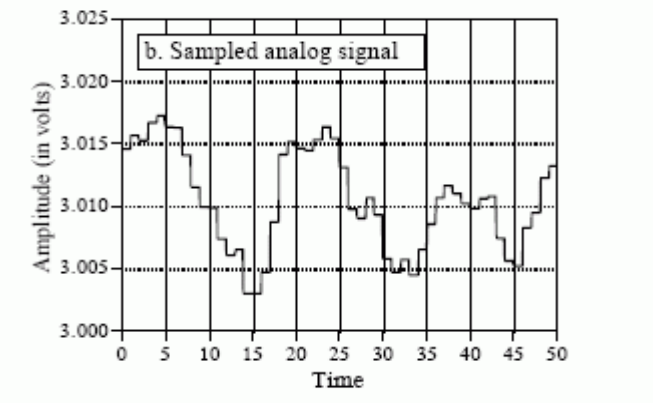
Na to aby sa mohla uskutočniť konverzia v ADC je potrebné aby bola amplitúda počas jej vykonávania konštantná. Vidíme, že na obrázku č. 3 je proces konverzie rozdelený na dve časti. Sekcia S/H (sample and hold) ma na za úlohu udržať napätie

signálu konštantné počas toho ako sa vykonáva konverzia. Je teda zrejmé, že hodnota signálu, ktorá vstupuje do ADC sa mení iba v určitých časových intervaloch, kedy sa zoberie presná hodnota signálu v danom čase. Rozdiel medzi originálnym analógovým signálom a vzorkovaným signálom je možné vidieť na obrázkoch č. 2 a 4. „Ostatné zmeny signálu, ktoré sa udiali medzi intervalmi vzorkovania sú kompletne ignorované. Takto Vzorkovanie prevádza v tomto prípade nezávislú premennú času zo spojitej na diskrétnu.“ [2] Z týchto faktov je rovnako zrejmé, že vernosť výsledného digitalizovaného signálu priamo úmerne rastie s veľkosťou použitej vzorkovacej frekvencie (počet vzoriek za jednotku času).



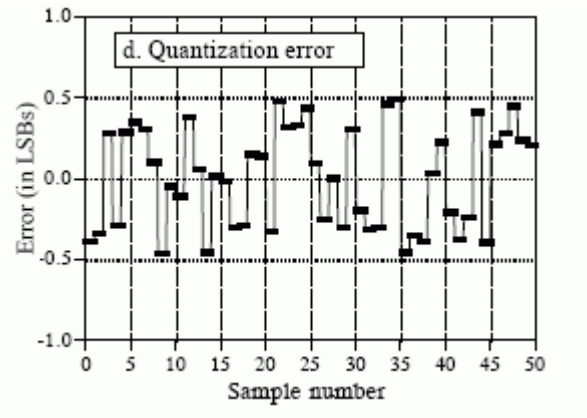
obr. č. 4: Vzorkovaný analógový signál [2]

Na obrázku č. 5 vidíme, že ADC priradil jednotlivým hodnotám analógového signálu z obr. č. 4 číslo medzi 0 a 4095. „Z tohto faktu vyplýva chyba, keďže hodnota analógového signálu môže byť hocijaké napätie medzi 0 a 4.095 voltami. Ak by teda boli hodnoty signálu napríklad 2.56000 a 2.56001 voltov, obe budu konvertérom prevedené do digitálneho čísla 2560. Inými slovami, kvantovanie prevádza premennú napätia zo spojitej na diskrétnu.“[2]



obr. č. 5: Digitalizovaný signál [2]

Je dôležité neporovnávať priamo analógový signál a digitalizovaný signál. Takéto porovnanie by mohlo viesť k spojeniu kvantovania a vzorkovania, pričom každá z týchto dvoch metód má iný vplyv na výsledný digitalizovaný signál. „Je dôležité pristupovať k nim oddelene, pretože obe degradujú signál rôznymi spôsobmi, rovnako v elektronike sú kontrolované rôznymi parametrami. Sú aj prípady kedy sa používa iba jedna z metód.“ [2]



obr. č. 6: Chyba kvantovania [2]

Kvantovanie spôsobuje chybu pri prevode Analógovej hodnoty signálu do digitálnej. Každá hodnota v digitalizovanom signále môže mať chybu maximálne ±½ LSB (z anglického: Least Significant Bit, čo je žargón pre susedné hodnoty kvantovania). V našom prípade sa teda hodnota LSB rovná 1, keďže sme digitalizovnému signálu priraďovali celé čísla od 0 do 4095. Chyba nášho digitalizovaného signálu znázornená na obrázku č. 6 vznikla odčítaním analógovej hodnoty na obrázku. 4 od digitálnej hodnoty obrázku 5. Inými slovami … „Digitálny výstup (na obr. č. 5) sa rovná sčítaniu vstupnej analógovej hodnoty (na obr. č. 4) a kvantovacej chyby (na obr. č. 6).“ [2]

Je dôležité si uvedomiť, že chyba ktorú kvantovanie spôsobuje je vlastne náhodný šum ktorý pridávame k reálnemu šumu ktorý sa nachádzal v signále. „Tento prídavný šum je rovnomerne rozdistribuovaný medzi ±1/2 LSB, s priemernou hodnotou 0 a štandardnou odchýlkou 1/√12 LSB (0.29 LSB). Napríklad ak analógový signál prejde 8-bitovým digitizérom, tento pridá náhodný šum o hodnote 0.29/256, alebo 1/900, z maximálnej amplitúdy signálu. Pre porovnanie 12-bitový digitizér pridá šum o hodnote 0.29/4096 teda 1/14 000 maximálnej amplitúdy signálu a 16-bitový digitizér o hodnote 0.29/65536 teda 1/227 000. Keďže chyba spôsobená kvantovaním je náhodný šum, počet bitov použitých digitizérom určuje presnosť výsledných dát.“ [2]

V mnohých prípadoch môže byť rozdiel medzi použitím napríklad 8-bitového alebo 12-bitového digitizéra veľmi veľký. Pri rozhodovaní sa aký počet bitov je potrebný pre digitalizáciu signálu by mali hrať kľúčovú úlohu odpovede na dve základné otázky a to: Koľko šumu sa nachádza v pôvodnom analógovom signále? Koľko šumu sme ochotní tolerovať vo výslednom digitalizovanom signále?

Pokiaľ je naším analógovým signálom ľudská reč, tak jej digitalizovaním pomocou ADC dostaneme digitálny rečový signál, ktorý je možné ďalej spracovať.

2.2 Vzorkovacia frekvencia

Ako bolo uvedené v kapitole 2.1 vzorkovacia frekvencia predstavuje počet hodnôt nameraných za jednotku času (najčastejšie sekundu) počas digitalizácie analógového signálu. Dôležitú úlohu pri nahrávaní, teda digitalizovaní zvuku, zohráva zvolenie správnej vzorkovacej frekvencie. To, že bola zvolená správna vzorkovacia frekvencia môžeme povedať, ak je možné jednoznačne zrekonštruovať analógový signál z digitálnych vzoriek tohto signálu. Spolu s dvíhaním hodnoty vzorkovacej frekvencia stúpa aj kvalita digitalizovaného záznamu zvuku ale aj objem dát ktoré takto nahraný zvuk zachycujú.

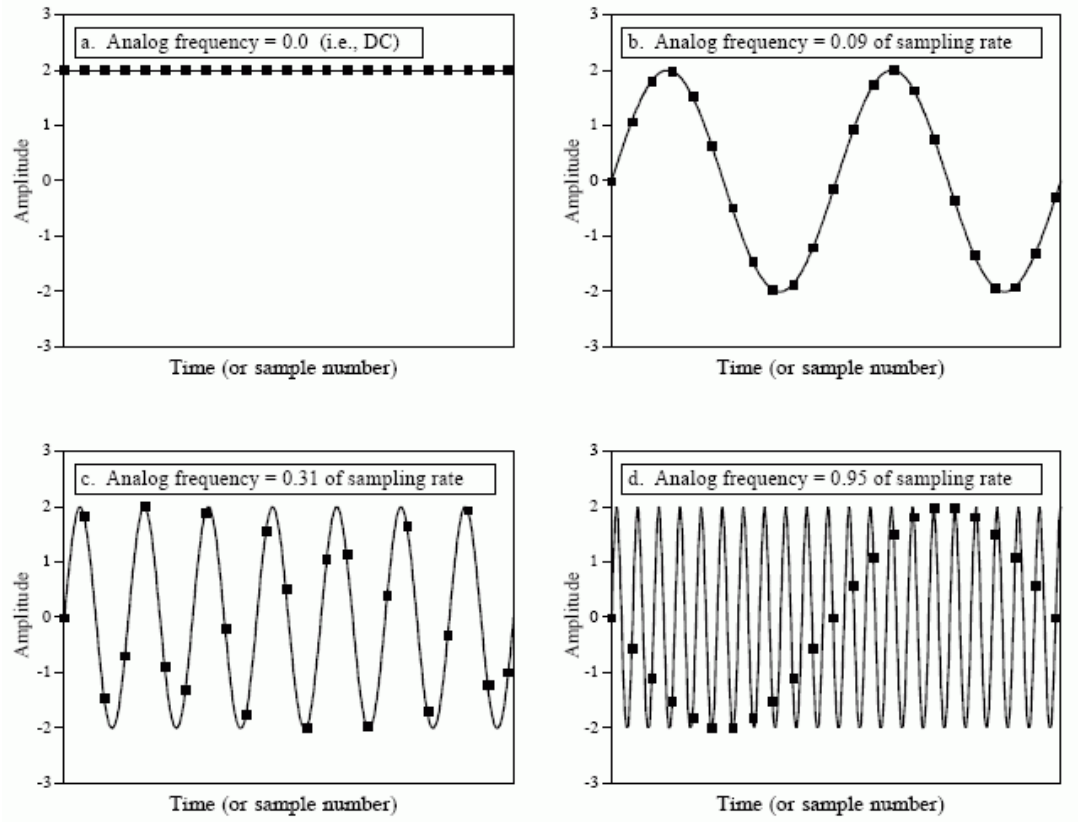
Na obrázku číslo 7 je znázornených niekoľko sinusoid pred a po digitalizovaní. Spojitá čiara predstavuje analógový signál a Čierne štvorce predstavujú hodnoty vzoriek získaných digitalizáciou daného analógového signálu.

Na obrázku 7a je znázornená kosínusová vlna nulovej frekvencie, teda konštantná

hodnota. Je zrejmé, že vzorky získané digitalizáciou obsahujú všetky informácie potrebné na rekonštrukciu analógového signálu. Je to teda správne zvolená vzorkovacia frekvencia.

Na obrázku 7b je sínusová vlna o frekvencii 0.09 vzorkovacej frekvencie. Pri vzorkovacej frekvencii 1000 vzoriek za sekundu bude teda táto sínusovka opakovať svoj cyklus 90 krát. Z to dostávame 1000 / 90 = 11.1 vzoriek na každý cyklus. Sú tieto vzorky dostačujúce na rekonštrukciu analógového signálu? „Odpoveď znie áno, pretože žiadna iná sinusoida ani kombinácia sinusoid nedokáže vyprodukovať takéto vzorky.“[2] Rovnaká odpoveď je aj pre obrázok 7c, kde je frekvencia sinusoidy 0.31 násobok vzorkovacej frekvencie. V našom prípade teda 3.2 vzorky na cyklus.

Obrázok 7d znázorňuje sinusoidu s ešte vyššou frekvenciu (0.95 násobok vzorkovacej frekvencie). V tomto prípade už získané vzorky nie sú schopné spoľahlivo reprezentovať analógový signál. „Tieto vzorky reprezentujú inú sínusovú vlnu ako bola obsiahnutá v pôvodnom analógovom signále. V skutočnosti je originálna sinusoida s frekvenciou 0.95 zamenená za sinusoidu s frekvenciou iba 0.05. Tento fenomén zmeny frekvencie sinusoidy počas vzorkovania sa nazýva **aliasing**.“ [2] môžeme teda povedať, že v tomto prípade išlo o zle zvolenú vzorkovaciu frekvenciu.



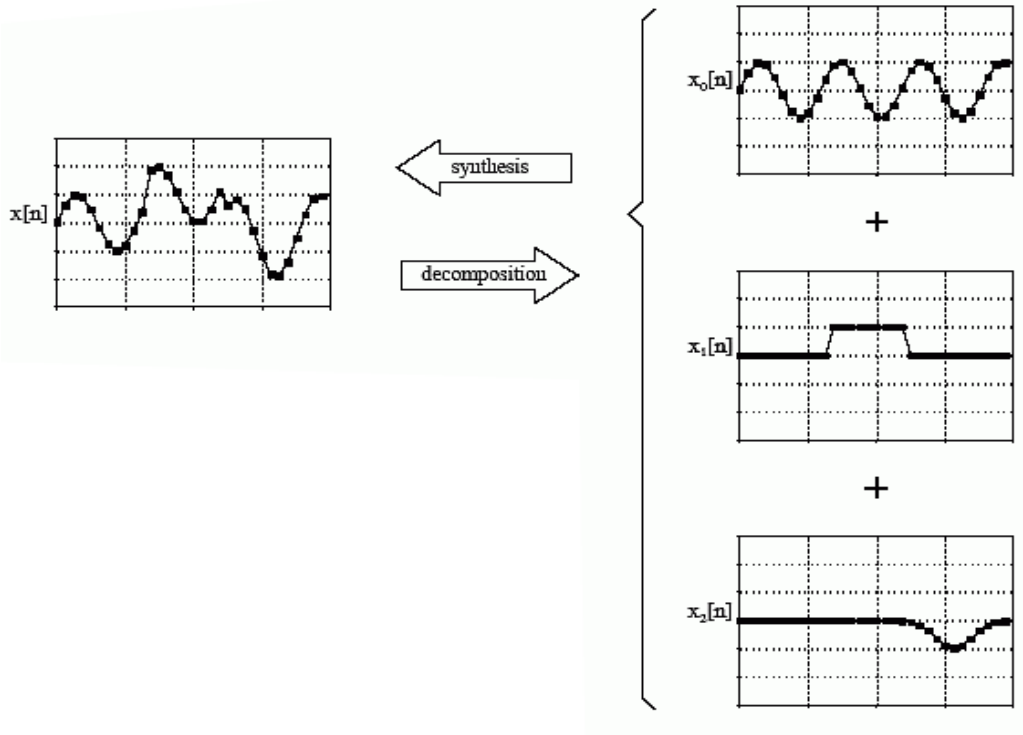
obr. č. 7: Ilustrácia správneho a nesprávneho vzorkovania signálu. Na obrázkoch a, b, a c je znázornená správne zvolená vzorkovacia frekvencia. Na obrázku d je znázornená nesprávne zvolená vzorkovacia frekvencia, keďže nie je možné jednoznačne zrekonštruovať pôvodný analógový signál. [2]

Správnu mieru vzorkovacej frekvencie udáva takzvaná vzorkovacia teoréma, ktorá je často nazývaná aj Shannonova teoréma alebo Nyquistova teoréma podľa autorov článku na túto tému z roku 1940. Teoréma hovorí, že spojitý signál môže byť správne vzorkovaný len vtedy, ak neobsahuje časti s vyššou frekvenciou ako je polovica vzorkovacej frekvencie

Napríklad vzorkovacou frekvenciou 4000 vzoriek za sekundu sme schopný správne vzorkovať analógový signál s frekvenciami 2000 a menšími. Ak sa v signále nachádzajú vyššie frekvencie, budú aliazované za frekvencie medzi 0 a 2000.

2.3 Dekompozícia

Dôležitým pojmom pri analýze signálu je dekompozícia. Dekompozícia je inverzná operácia k syntéze. Zatiaľčo syntézou (sčítavanie viacerých vstupných signálov) dostávame z viacerých signálov jeden výsledný, dekompozícia slúži na "rozbitie" jedného signálu na viacero jednoduchších aditívnych komponentov. Je to zložitejší proces ako syntéza, pretože existuje nekonečne množstvo dekompozícii pre jeden signál, zatiaľčo existuje vždy len jeden výsledok pre syntézu.



Obr. č. 8: Ilustrácia syntézy a dekompozície signálu [2]

Na obrázku č. 8 je znázornená syntéza a dekompozícia signálu. Sčítaním, teda syntézou, signálov $x_0[n]$, $x_1[n]$ a $x_2[n]$ dostaneme signál x[n]. Dekompozícia je opačný proces, teda rozbitie signálu x[n] na viacero signálov.

Z hľadiska analýzy digitálneho signálu je dekompozícia veľmi dôležitá. Umožňuje namiesto riešenia zložitých problémov (analýzy celého signálu), riešiť jednoduchšie a výpočtovo menej náročné problémy (analýzu jednotlivých komponentov dekompozície). Takéto riešenie pod-problémov je možné vďaka superpozícii. Predpokladajme, že máme vstupný signál x[n]. Výsledkom použitia lineárnej funkcie na signál x[n] dostaneme výstupný signál y[n]. Ďalej predpokladajme že vstupný signál x[n] rozdelíme

dekompozíciou na skupinu jednoduchších signálov, ktoré nazveme komponenty vstupného signálu. Použitím rovnakej lineárnej funkcie, ako v predošlom prípade, na tieto komponenty, dostaneme množinu komponentov výstupného signálu, ktorých syntézou dostaneme výstupný signál y[n]. Výstupný signál takejto syntézy je identický so signálom vyprodukovaným priamym použitím funkcie na pôvodný vstupný signál x[n]. "Namiesto snaženia sa porozumieť, ako systém mení komplikovaný signál, stačí ak pochopíme ako sa vplyvom systému mení veľmi jednoduchý signál. V žargóne spracovania signálu sú vstupný a výstupný signál považované za superpozíciu (sumu) jednoduchších signálov. Toto je základom pre takmer všetky techniky používané pri spracovaní signálu"[2]

2.3.1 Typy dekompozície

Je niekoľko možných spôsobov dekompozície signálu. Dva najznámejšie a najpoužívanejšie z nich sú Impulzná dekompozícia a Furierová dekompozícia. V tejto kapitole stručne objasníme impulznú dekompozíciu. Furierovou dekompozíicou sa budeme podrobnejšie zaoberať v tretej kapitole.

Impulzná dekompozícia rozdeľuje vstupný signál obsahujúci N vzoriek na N komponentov z ktorých každý obsahuje N vzoriek. Každý z komponentov obsahuje jeden bod (hodnotu signálu vzorky) z pôvodného vstupného signálu a zvyšné vzorky majú hodnotu 0. Takáto dekompozícia nám umožňuje skúmať a spracovávať signál doslova po jednej vzorke. Vzorka komponentu, ktorá je nenulová a obsahuje hodnotu pôvodného signálu, sa nazýva **impulz**.

"Systémy sú charakterizované podľa toho ako reagujú na impulzy. Ak vieme ako systém zareaguje na impulz, je možné vypočítať výstup systému pre ľubovoľný vstup. Tento prístup sa nazýva konvolúcia."[2]

3. Spracovanie signálu slovenských samohlások

V predošlých kapitolách sme sa zaoberali tvorbou zvuku, následne reči, v ľudskom rečovom trakte a základnými postupmi pri digitalizácii analógového signálu. Diskrétny digitálny signál získaný digitalizáciou však sám o sebe nestačí na identifikáciu znelých zvukov teda samohlások. Na identifikáciu samohlások je potrebné získať informácie z frekvenčného spektra signálu. Frekvenčné spektrum signálu je možné vypočítať pomocou diskrétnej Furierovej transformácie. Potrebné údaje na identifikáciu samohlások je však možné získať aj priami z rečového signálu pomocou metódy lineárneho prediktívneho kódovania (LPC). Rozpoznávať samohlásky zo signálu je teda možné viacerými postupmi. V tejto práci sa budeme zaoberať dvoma vyššie uvedenými, teda pomocou Diskrétnej Fuurierovej transformácie a pomocou LPC.

Predtým ako detailnejšie rozoberieme tieto metódy je potrebné zadefinovať čo je to frekvenčné spektrum a podľa ktorých informácii z neho dokážeme určiť o akú samohlásku ide.

Frekvenčné spektrum je reprezentácia signálu (v časovom priestore) vo frekvenčnom priestore. Frekvenčný priestor používa namiesto vyjadrenia hodnoty signálu v čase vyjadrenie amplitúdy signálu na danej frekvencii.

3.1 Rozpoznávanie samohlások vo frekvenčnom spektre

V tejto práci sa zameriame na rozpoznávanie znelých zvukov teda samohlások. Slovenské samohlásky ktorými sa budeme zaoberať sú a, e, i, o, u. „Pri tvorbe samohlások človekom sa okrem základného hlasivkového tónu objavujú vyššie zosilnené tóny, ktoré vznikajú rezonanciou v dutinách hlasového traktu.“ [1] Tieto zvýšené tóny, ktoré nazývame formanty, zohrávajú dôležitú úlohu pri rozpoznávaní samohlások počítačom.

Po pre-transformovaní rečového signálu do frekvenčného spektra je možné identifikovať o ktorú samohlásku ide vďaka prvým dvom formantom, ktoré sú špecifické

pre každú samohlásku. Slovenské samohlásky majú prvé dva formanty v nasledovných frekvenčných rozmedziach:

a:	F1 = 750 – 1100 Hz	F2 = 1100 – 1500 Hz
e:	F1 = 500 – 700 Hz	F2 = 1500 – 2000 Hz
i:	F1 = 300 – 500 Hz	F2 = 2000 – 3000 Hz
o:	F1 = 500 – 700 Hz	F2 = 900 – 1200 Hz
u:	F1 = 300 – 500 Hz	F2 = 600 – 1000 Hz

Samotné frekvenčné spektrum však nie je vhodné na vyhľadávanie formantov. Na to aby sme mohli určiť tieto formanty potrebujeme dostať tzv. obálku frekvenčného spektra. Termín spektrálna obálka predstavuje vyhladenú funkciu, ktorá prechádza cez vrcholy frekvenčného spektra. [5]

V tejto práci sa zameriame na dva spôsoby, pomocou ktorých je možné získať obálku frekvenčného spektra. Spôsobom na prevod rečového signálu do frekvenčného spektra je rýchla Furierová transformácia (FFT – fast Fourier transformation), čo je algoritmus na používaný pri optimalizácii diskrétnej Furierovej transformácie. Následným vyhladením frekvenčného spektra pomocou aritmetického filtra je možné získať obálku frekvenčného spektra z ktorej je možné určiť formanty. Druhým spôsobom, ktorý je veľmi efektívny pri analýze zvukového signálu, je lineárne prediktívne kódovanie (LPC – linear predictive coding) s použitím Durbinovho algoritmu. Tieto dva spôsoby sa navzájom líšia v prístupe, pomocou ktorého sa získava obálka frekvenčného spektra.

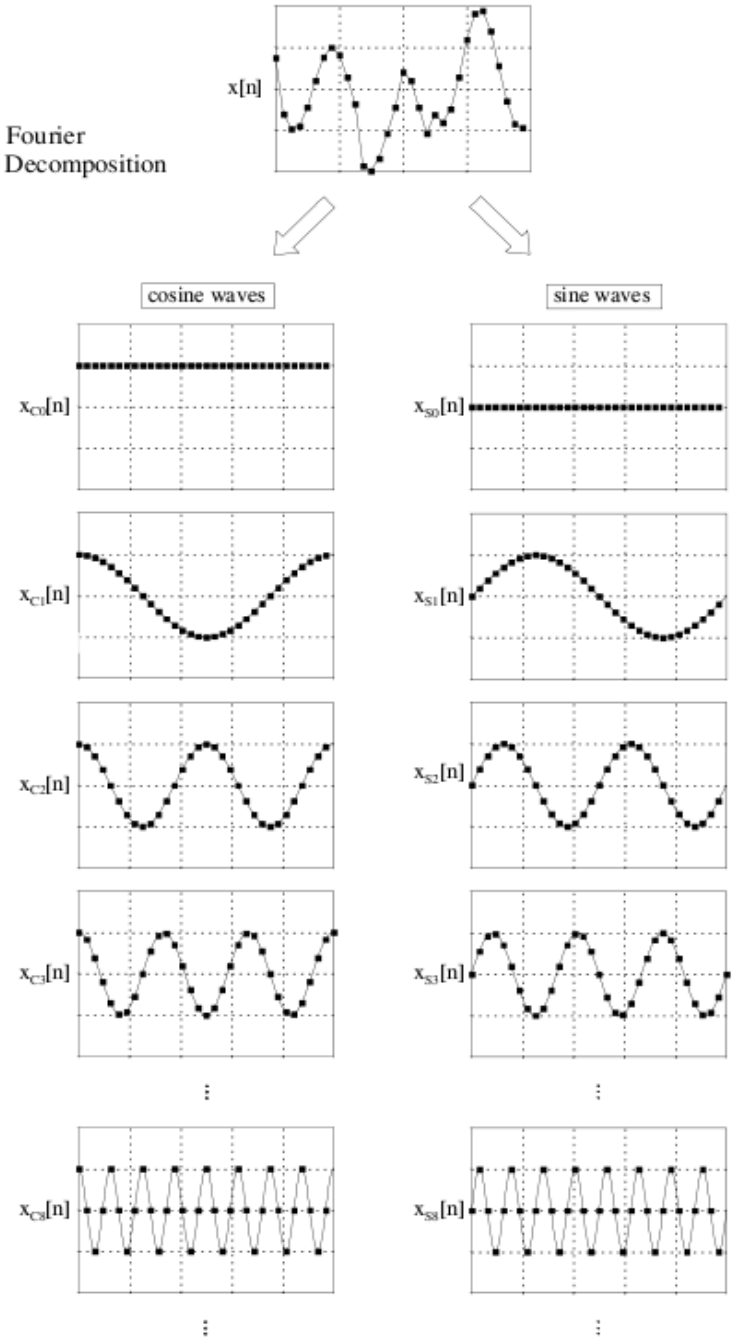
3.2 Furierová transformácia

3.2.1 Furierová dekompozícia

Diskrétna Furierová transformácia je matematická technika založená na Furierovej dekompozícii signálu, ktorá sa používa v prípade diskrétneho digitálneho signálu.

Furierová dekompozícia je založená na fakte, že každý signál o dĺžke N vzoriek je

možné rozložiť na N+2 signálov. Polovica z týchto signálov sú sínusové vlny a druhá polovica kosínusové vlny. Furierová dekompozícia je znázornená na obrázku č. 9. Sinusoidy postupne zvyšujú svoju frekvenciu. Najnižšia má frekvenciu 0. Nasledujúca má vždy frekvenciu o jedna vyššiu od predošlej. Frekvencia sínusoviek pre každý signál je teda fixná. Jediná vec ktorá sa líši pre rôzne signály, je amplitúda dekompozovaných sínusových a kosínusových vĺn.



obr. č. 9: Furierová dekompozícia

Furierová dekompozícia tvorí základ pre oblasť matematiky, ktorá sa nazýva Furierová analýza.

3.2.2 Diskrétna Furierová transformácia

Diskrétna Furierová transformácia (DFT) je matematická technika z rodiny Furierovej analýzy. DFT sa používa v prípade diskrétneho digitálneho signálu.

Furierová analýza je pomenovaná podľa francúzskeho matematika Jeana Baptistu Josepha Fouriera (1768 – 1830).

Metódy Furierovej analýzy ku ktorým patrí aj DFT sú založené na Furierovej dekompozícii signálu. Na základe typu signálu ktorý je transformáciou spracovávaný rozlišujeme štyri typy Furierových transformácií. Signál môže byť spojitý alebo diskrétny a zároveň periodický alebo aperiodický. Na základne kombinácií týchto dvoch kritérií rozdeľujeme Furierové transformácie na :

Aperiodická-spojitá (Aperiodic-Continuous)

“Zahŕňa napríklad transformáciu exponenciálnych alebo gausových kriviek. Tento signál môže obsahovať hodnoty od mínus nekonečna do nekonečna bez periodického opakovania. Furierová transformácia používaná pre takýto signál sa nazýva Furierová transformácia.”[2]

Periodická-spojitá (Periodic-Continuous)

Tu môžu byť signálmi napríklad sínusové vlny, alebo iný spojitý signál ktorý ma periodický charakter. Transformácia používaná v tomto prípade sa nazýva Furierové série (Fourier series)

Aperiodická-diskrétna (Aperiodic-Discrete)

Vstupným signálom je diskrétny signál bez periodického opakovania. Definované su teda diskrétne body medzi záporným a kladným nekonečnom. Transformácia takéhoto signálu sa nazýva diskrétna časová Furierová transformácia (discrete time Fourier transformation)

Periodická-diskrétna (Periodic Discrete)

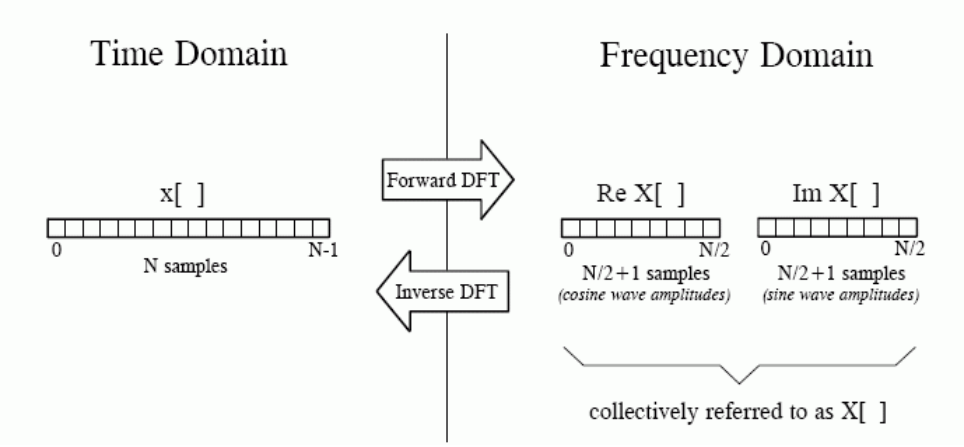
Transformovaný vstupný signál v tomto prípade je periodický a diskrétny. Hodnoty diskrétnych bodov sú taktiež medzi záporným a kladným nekonečnom. Transformácia takéhoto signálu sa nazýva diskrétna Furierová transformácia (discrete Fourier transform - DFT).

Pri spracovaní diskrétneho signálu počítačom, je vstupným signálom konečný počet vzoriek. Furierová transformácia pracuje so sínusovými a kosínusovými vlnami nekonečnej dĺžky. Riešením v tomto prípade je uvažovať náš konečný vstupný signál ako nekonečný. Tu môžu nastať dva prípady. V prvom je signál neperiodický a hodnoty mimo vstupných hodnôt vstupného signálu su rovné nule. V tomto prípade by sa použila diskrétna časová Furierová transformácia. Druhou možnosťou je uvažovať signál periodický. Ak mal teda vstupný signál dĺžku N tak perióda nekonečného periodického signál bude N. v tomto prípade by sme použili DFT.

“Ukázalo sa, že na vyjadrenie neperiodického nekonečného signálu sínuovkami je potrebný nekonečný počet týchto sínusoviek. Tento fakt robí nemožným vypočítať diskrétnu časovú Furierovú transformáciu počítačovým algoritmom. Jediným typom Furierovej transformácie, ktorý je možno použiť pri spracovaní digitálneho signálu, je teda DFT.”[2]

Každá z Furierových transformácií má reálnu a komplexnú verziu. Pre vstupný signál dĺžky N dostaneme po pre-transformovaní pomocou DFT dva výstupné signály dĺžky N/2+1. Tieto dva signály obsahujú ako hodnoty amplitúd kosínusových a sínusových vĺn. Pri reálnej Furierovej transformácii používame hodnoty amplitúde kosínusových vĺn ako reálnu zložku výstupného signálu.

Pri komplexnej transformácii používame aj imaginárnu zložku, čo sú hodnoty amplitúd sínusových vĺn. Prevod na reálnu a imaginárnu zložku sa realizuje pomocou tzv. Priamej DFT. Môžme teda povedať, že na prevod diskrétneho signálu z časového spektra (time domain) do frekvenčného spektra (frequency domain) používame Priamu DFT. Opačný proces, teda získanie pôvodného signálu z frekvenčného spektra sa používa tzv. inverzná DFT. Tieto dva procesy sú zobrazené na obrázku č. 10.



Obr č. 10: DFT

"Frekvenčné spektrum obsahuje rovnaké informácie ako časové spektrum. Sú iba v inej forme. Pokiaľ poznáme aspoň jedno z nich vieme vyrátať druhé. Ak máme časové spektrum, proces vyrátania frekvenčného spektra nazývame aj dekompozícia, analýza alebo priama DFT. Ak máme frekvenčné spektrum proces vyrátania časového spektra nazývame tiež syntéza alebo inverzná DFT." [2]

Sínusové a kosínusové vlny, ktoré dostaneme ako výsledok DFT, sú nazývame aj bázové funkcie. Tieto vlny sa líšia amplitúdou a frekvenciou opakovania sa cyklov. Na vypočítanie bázových funkcií DFT sa používajú nasledovné rovnice:

$c_k[i] = \cos(2 \pi ki/N)$

$s_k[i] = \sin(2 \pi ki/N)$

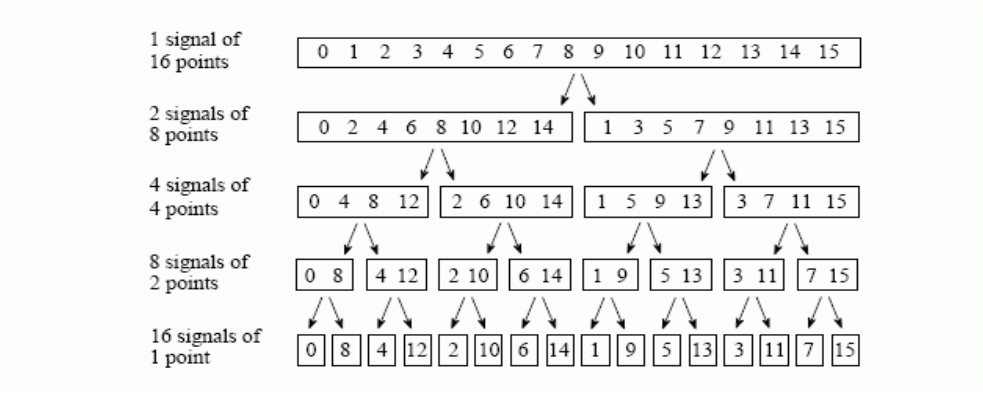
kde $c_k[]$ je kosínusová vlna pre amplitúdu ktorá je uložená v Re X[k] a $s_k[]$ je sínusová vlna s amplitúdou, ktorá je uložená v Im x[k]. N je počet bodov vstupného diskrétneho signálu a i teda ide od 0 po N-1.

Na výpočet DFT sa používajú tri rôzne spôsoby. Najrýchlejším z hľadiska z týchto troch, väčšinou stovky krát rýchlejšia, je tzv. rýchla Furierová transformácia (FFT z angl. Fast Furier transform). Tento typ výpočtu DFT použijeme aj v tejto práci. Často sa používa aj spôsob založený na korelácii a teda detekovanie známej formy vlnenia signálu v inom signále. Tento spôsob má však význam iba pri krátkych vstupných signáloch, väčšinou sa používa ak je vstupný signál kratší ako 32 vzoriek. V opačnom prípade sa používa FFT.

3.2.3 Rýchla Furierová transformácia

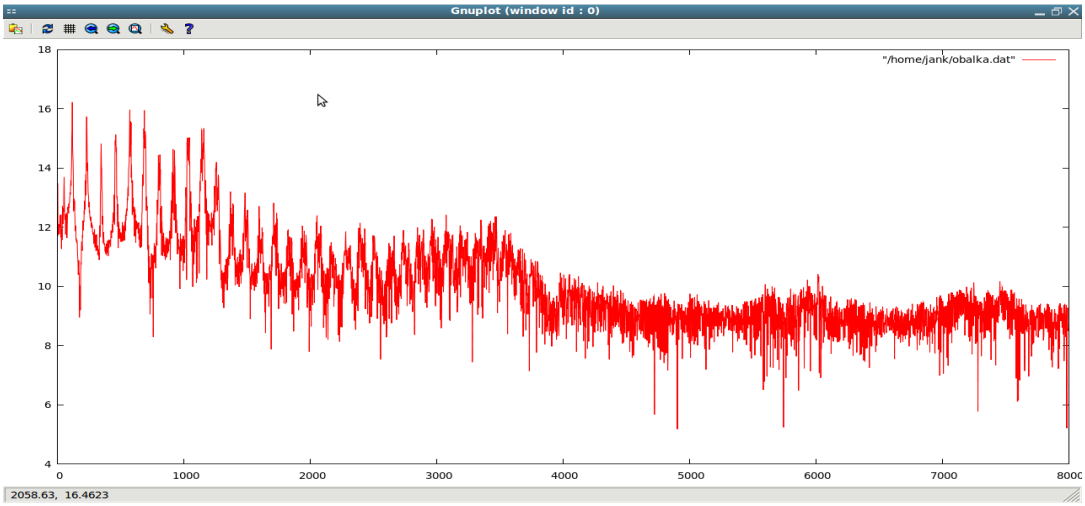
FFT je algoritmus ktorý umožňuje výpočet DFT v oveľa kratšom čase ako samotná DFT. FFT je zložená z niekoľkých fáz.

Najskôr sa signál o dĺžke N bodov pomocou Špeciálnej dekompozície rozdelí na N signálov dĺžky 1. Táto dekompozícia je znázornená na obrázku č. 11. Potom je vypočítané frekvenčné spektrum pre jednotlivé signály z tejto dekompozície. A následne sa signály skladajú aby sme získali frekvenčné spektrum pôvodného vstupného signálu dĺžky N.



Obr č. 11: dekompozícia signálu pri FFT[2]

FFT slúži pri získavaní obálky frekvenčného spektra práve na prevod pôvodného rečového signálu do frekvenčného spektra. Následne aritmetickým spriemerovaním hodnôt spektra získame obálku. Na obrázku číslo 12 môžeme vidieť frekvenčné spektrum pre samohlásku „a“ získané pomocou FFT.

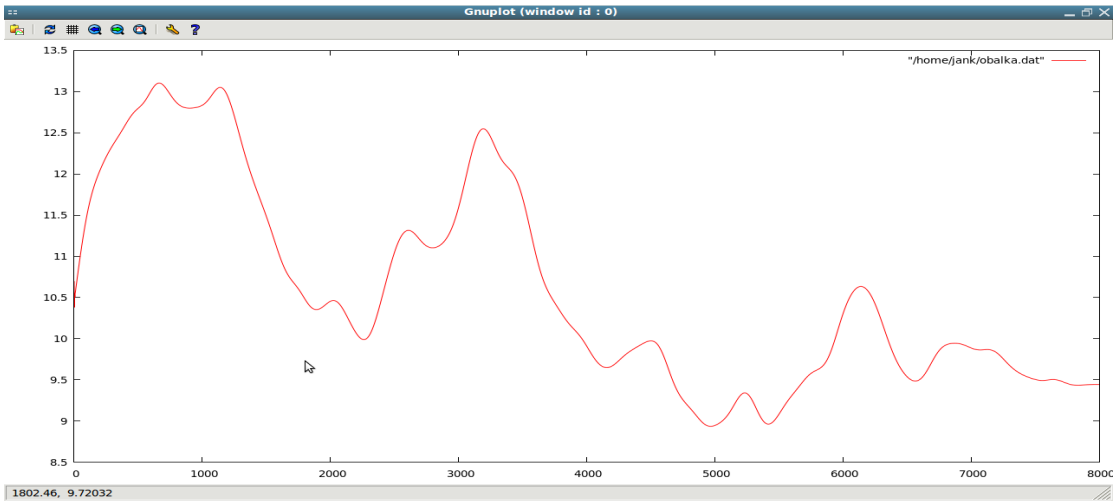


Obrázok č. 12: frekvenčné spektrum samohlásky „a“

Obrázok číslo 13 už znázorňuje to isté frekvenčne spektrum filtrované pomocou aritmetického priemeru (obálku).

Pre rozpoznanie konkrétnej samohlásky je potrebné v obálke identifikovať prvé dva formanty a podľa ich pozície určiť o akú samohlásku sa jedná. Pokiaľ sú získané dáta správne mali by platiť údaje nachádzajúce sa v tabuľke v kapitole 3.1.

V tejto práci sme použili FFT z knižnice GSL (GNU Scientific Library).



Obrázok č. 13: obálka frekvenčného spektra pre samohlásku „a“

3.2.3 Matematická definícia

Všetky transformácie z rodiny Furierových transformácií môžu byť rátané s reálnymi alebo v komplexnými číslami.

Reálna DFT vychádza z rovníc:

$$\Re X[k] = \frac{2}{N} \sum_{n=0}^{N-1} x[n] \cos(2\Pi kn / N) \quad \text{(v3.1)}$$

$$\Im X[k] = \frac{-2}{N} \sum_{n=0}^{N-1} x[n] \sin(2\Pi kn / N) \quad \text{(v3.2)}$$

FFT je efektívny algoritmus na výpočet diskrétnej Furierovej transformácie (DFT) založenej na komplexných Číslach:

$$X[k]=\frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2\Pi kn/N} \quad \text{(v3.3)}$$

„Výpočet N koeficientov pomocou DFT podľa vzťahu (v 3.3) vyžaduje značné množstvo výpočtových operácií – v priemere sa jedná o N^2 operácií sčítania komplexných čísel. FFT je postup, ktorý umožňuje zníženie množstva operácií pri výpočte koeficientov DFT .

3.3 Lineárne prediktívne kódovanie

Lineárne prediktívne kódovanie (LPC), z anglického linear predictive coding, je digitálna metóda na kódovanie analógového signálu. Táto metóda využíva na určenie konkrétnej hodnoty analógového signálu lineárnu funkciu jeho predošlých hodnôt. Môžeme teda povedať, že LPC je metóda, ktorá sa snaží odhadnúť nasledujúcu hodnotu analógového signálu na základe predošlých už známych hodnôt.

LPC je jednou z najefektívnejších metód analýzy akustického signálu. Táto metóda sa zameriava na odhad parametrov modelu vytvárania reči priamo z rečového signálu. Silnou stránkou tejto metódy je možnosť získať veľmi presné odhady uvedených parametrov pri malej výpočtovej záťaži.

Prvý krát bola táto metóda predstavená ako metóda pre kódovanie ľudskej reči Ministerstvom obrany Spojených štátov v roku 1984.

LPC je metóda, ktorá sa snaží modelovať produkciu ľudskej reči v rečovom trakte. Jedným z hlavných komponentov ovplyvňujúcim vytváranie ľudskej reči je vzduch vytláčaný z pľúc. Vzduch prúdiaci z pľúc môžeme považovať za zdroj pre rečový trak, ktorý sa správa ako filter a zo zdroja produkuje reč.

„Idea vzduchu prúdiaceho z pľúc ako zdroja a rečového traktu ako filtra sa nazýva aj source-filter model pre produkciu reči. Source-filter model je model, ktorý je tiež používaný pri lineárnom prediktívnom kódovaní, je založený na myšlienke oddelenia

zdroja od filtra pri produkcii zvuku. Tento model je použitý pri analýze aj syntéze zvuku pomocou LPC a je odvodený z matematickej aproximácie rečového traktu reprezentovaného ako trubica s meniacim sa priemerom.“[3]

Source-filter model použitý pri LPC je nazývaný model lineárneho prediktívneho kódovania (linear predictive coding model – LPC model). Má dva základné komponenty a to pre analýzu a syntézu signálu.

Pri rozpoznávaní samohlások je podstatná analytická časť LPC modelu, ktorá je jednou z najefektívnejších metód analýzy akustického signálu.

3.3.1 Matematická definícia LPC

Podľa Jozefa Psutku [1] sa model vytvárania reči skladá zo systému s časovo premenným prenosom a budiacich funkcií, ktoré pri vytváraní znelých zvukov budia systém postupnosťou impulzov a pri vytváraní neznelých zvukov náhodným šumom. Princíp metódy LPC je založený na predpoklade, že k-tá vzorka signálu s(k) sa dá popísať lineárnou kombináciou Q predchádzajúcich vzoriek a budenia u(k), tj.

$$s(k) = -\sum_{i=1}^{Q} a_i s(k-i) + Gu(k) \quad \text{(v1)}$$

kde G je koeficient zosilnenia a Q je rád modelu. Prenosovú funkciu modelu H(z) môžeme zapísať v tvare

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{A(z)} = \frac{G}{1 + \sum_{i=1}^{Q} a_i z^{-i}} \quad \text{(v2)}$$

Sledované parametre sú v tomto prípade koeficienty a_i číslicového filtra a koeficient zosilnenia G. Ak je splnený predpoklad približnej stacionarity signálu na sledovanom časovom intervale, je možné na určenie a_i a G využiť metódu najmenších štvorcov. Pri neznámom člene Gu(k) v rovnici (v1) vzniká medzi skutočnou hodnotou s(k) a odhadovanou hodnotou s'(k) chyba e(k). Funkcia krátkodobej chyby signálu

$$E = \sum e^2(k) = \sum [s(k) - s'(k)]^2 = \sum [s(k) + \sum_{i=1}^{Q} a_i s(k-i)]^2 \quad \text{(v3)}$$

má minimum v bode

$$\frac{\partial E}{\partial a_j} = 0, 1 \leqslant j \leqslant Q. \quad \text{(v4)}$$

Riešením (v4) dostaneme sústavu rovníc

$$\sum_{i=1}^{Q} a_i \sum_k s(k-j)s(k-i) = -\sum_k s(k-j)s(k), 1 \leqslant j \leqslant Q. \quad \text{(v5)}$$

Pokiaľ označíme

$$\varphi(j,i) = \sum_k s(k-j)s(k-i), \quad \text{(v6)}$$

potom je možné (v5) prepísať na

$$\sum_{i=1}^{Q} a_i \varphi(j,i) = -\varphi(j,0), 1 \leqslant j \leqslant Q. \quad \text{(v7)}$$

„Pri riešení tejto sústavy rovníc je možne použiť dva základné postupy z hľadiska voľby medzí sumácie pre k a to autokoerlačný a kovariančný. Pri spracovaní reči sa väčšinou vyžíva autokorelačný prístup, ktorý je založený na predpoklade, že signál mimo skúmaného krátkodobého intervalu alebo mikrosegmentu je rovný nule. Je teda možné ho zapísať vzťahom“ [1]

$$s_n(k) = s(n+k)w(k), 0 \leqslant k \leqslant N-1, \quad \text{(v8)}$$

kde w(k) je okienko konečnej dĺžky. Pri spracovaní reči sa najčastejšie používa Hammingovo okienko. N vo vzorci (v8) je dĺžka mikrosegmentu (určená počtom vzoriek ktoré daný mikrosegment obsahuje).

Pokiaľ sa $s_n(k)$ líši od nuly iba na intervale $0 \leq k \leq N-1$, tak potom sa chyba predikcie $e_n(k)$ pre prediktor Q-teho radu bude líšiť od nuly na intervale $0 \leq k \leq N-1+Q$,

takže $$E_n = \sum_{k=0}^{N-1+Q} e_n^2(k). \quad \text{(v9)}$$

Keďže sme povedali, že $s_n(k)$ sa líši od nuly iba na intervale $0 \leq k \leq N-1$ a mimo neho je rovné nule, je možné vzťah (v6), ktorého medze musia súhlasiť s medzami (v9)

$$\varphi_n(j,i) = \sum_{k=0}^{N-1+Q} s_n(k-j)\, s_n(k-i),\ 1 \leqslant j \leqslant Q,\ 0 \leqslant i \leqslant Q, \quad \text{(v10)}$$

vyjadriť ako

$$\varphi_n(j,i) = \sum_{k=0}^{N-1-(j-i)} s_n(k)\, s_n(k+j-i),\ 1 \leqslant j \leqslant Q,\ 0 \leqslant i \leqslant Q. \quad \text{(v11)}$$

(v11) je krátkodobá autokorelačná funkcia R_n, pričom platí, že

$$\varphi_n(j,i) = R_n(j-i), \quad \text{(v12)}$$

kde

$$R_n(i) = \sum_{k=0}^{N-1-i} s_n(k)\, s_n(k+i). \quad \text{(v13)}$$

Pokiaľ je $R_n(i)$ párnou funkciou, tak platí

$$\varphi_n(j,i) = R_n(|j-i|),\ 1 \leqslant j \leqslant Q,\ 0 \leqslant i \leqslant Q \quad \text{(v14)}$$

a rovnicu (v7) je možné prepísať do tvaru

$$\sum_{i=1}^{Q} a_i R_n(|j-i|) = -R_n(j),\ 1 \leqslant j \leqslant Q. \quad \text{(v15)}$$

alebo do maticového tvaru

$$\begin{bmatrix} R_n(0); R_n(1); R_n(2); \ldots; R_n(Q-1) \\ R_n(1); R_n(0); R_n(1); \ldots; R_n(Q-2) \\ \vdots \\ R_n(Q-1); R_n(Q-2); R_n(Q-3); \ldots; R_n(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_Q \end{bmatrix} = \begin{bmatrix} -R_n(1) \\ -R_n(2) \\ \vdots \\ -R_n(Q) \end{bmatrix} \quad \text{(v16)}$$

Podobne môžeme vyjadriť aj minimálny stredný kvadrát chyby predikcie. Ak zoberieme do úvahy v krátkodobom intervale vzťahy (v3), (v5) a (v13), platí, že

$$E_n = R_n(0) + \sum_{i=1}^{Q} a_i R_n(i). \quad \text{(v17)}$$

3.3.2 Durbinova metóda

Keďže matica (v16) je symetrická je možné na výpočet koeficientov ai využiť tzv. Durbinov algoritmus. Je to iteratívny algoritmus pôvodne navrhnutý Levinsonom a neskôr upravený Durbinom (tzv. Levinson-Durbinov algoritmus). Levison-Durbinov algoritmus využíva autokorelačnú metódu na dosiahnutie lineárnej predikcie parametrov náhodného signálu. V tejto metóde sa na riešenie koeficientov a_i v sústave rovníc využíva rekurzia. Koeficienty sú postupne vyjadrené v iteráciách pre i =1,2, … , Q.

$$\begin{aligned} E_n^{(0)} &= R_n(0), \\ k_i &= -[R_n(i)+\sum_{j=1}^{i-1} a_j^{(i-1)} R_n(i-j)]/E_n^{(i-1)}, \\ a_i^{(i)} &= k_i, \\ a_j^{(i)} &= a_j^{(i-1)}+k_i a_{i-j}^{(i-1)}, 1 \leqslant j \leqslant i-1, \\ E_n^{(i)} &= (1-k_i^2) E_n^{(i-1)}, \end{aligned} \qquad \text{(v18)}$$

kde $a_j^{(i)}$ je j-ty parameter prediktoru rádu i. Veľmi často sa v praxi možno stretnúť s tým, že sa namiesto koeficientov a_i používajú medzi-výsledky k_i. Tieto koeficienty sa nazývajú aj reflekčné koeficienty alebo PARCOR (partial correlation coefficients). „Tieto koeficienty sa využívajú pri riešení potencionálnych problémov pri prenose koeficientov filtra a_i. “[3]

Pri výpočte koeficientov a_i zároveň dostávame aj chybu predikcie $E_n^{(i)}$. Podľa toho ako sa vyvíja funkcia chyby predikcie je možné jednoducho optimalizovať rád prediktoru.

Koeficienty a_i umožňujú aj výpočet spektra signálu. Toto spektrum má tvar vyhladenej obálky skutočného spektra pôvodného diskrétneho digitálneho signálu. Podobnú obálku sme dostali napríklad aplikovaním FFT a následným vyhladením získaného signálového spektra pomocou aritmetického filtra.

Po substitúcii $z = e^{jw}$ platí pre frekvenčný prenos modelu vytvárania akustického signálu nasledovný vzťah

$$H(j\omega)=\frac{G}{1+\sum_{i=1}^{Q} a_i e^{-j\omega i}} \quad \text{(v19)}$$

Je tiež možné určiť vzťah pre výkonové spektrum P(ω), ktoré má po úprave tvar

$$P(\omega)=[\ H(j\omega)\ ]^{2}=\frac{G^{2}}{RA(0)+2\sum_{i=1}^{Q} RA(i)\cos(i\omega)} \quad \text{(v20)}$$

kde

$$RA(i)=\sum_{k=0}^{Q-1} a_k a_{k+i} \quad \text{(v21)}$$

a a0 = 1, 0≤ i≤ Q. ω - predstavuje kruhovú frekvenciu ktorú môžeme do vzťahov (v19) a (v20) dosadiť zo vzťahu $\omega=2\Pi f/F_v$, kde F_v[Hz] je vzorkovacia frekvencia signálu s(t) a f[Hz] je meniaca sa frekvencia. Platí, že $f\leq 0{,}5F_v$.[1]

4. Návrh riešenia

Cieľom tejto diplomovej práce je zanalyzovať, navrhnúť a implementovať program, pomocou ktorého je užívateľ schopný jednoducho ovládať hlasom kurzor myši svojho osobného počítača. Pre takýto program sme sa rozhodli z niekoľkých dôvodov. Rozpoznávanie reči je v súčasnej dobe stále ešte nedokonalou, ale rozmáhajúcou sa vetvou computer science a predovšetkým umelej inteligencie. Vybrať si v tejto oblasti problém, ktorého riešenie a implementácia zodpovedá rozsahu diplomovej práce, a ktorý nie je ešte riešený, alebo má len veľmi málo riešení, je veľmi motivujúce. Ovládanie kurzoru myši hlasom sme zvolili aj z toho dôvodu, že podobná implementácia môže byť užitočná a použiteľná aj v praxi. Príkladom môžu byť ľudia, ktorým rôzne postihnutia neumožňujú štandardné používanie myši a ovládanie počítača môže byť teda pre nich veľký problém.

4.1. Identifikácia problému

Pri návrhu riešenia kurzoru myši ovládaného hlasom je potrebné zodpovedať niekoľko základných otázok. Akými príkazmi budeme kurzor ovládať? Koľko funkcií bude náš program obsahovať? Koľkými smermi sa bude myš hýbať. Pre účel tejto diplomovej práce sme sa rozhodli použiť ovládanie pohybu myši samohláskami. Samohlásky sme zvolili kvôli veľmi dobrým vlastnostiam pri spracovaní a analýze digitálneho signálu a kvôli jednoduchosti ovládania. Ovládanie nepretržitým vyslovovaním samohlásky po dobu, počas ktorej chceme aby sa mys hýbala niektorým smerom, považujeme za jednoduchšie ako zadávanie slovných príkazov. Na to aby sme kurzor boli schopný takýmto ovládaním dostať na ľubovoľné miesto obrazovky sú potrebné minimálne štyri smery, ktorými sa bude kurzor pohybovať. Je potrebné teda jednoznačne rozpoznávať štyri samohlásky. Z ostatných funkcií je pre základné ovládanie moderných operačných systémov potrebné zakomponovať klikanie myši.

4.2. Pohyb kurzoru

V analýze riešenia sme v jednoduchosti načrtli spôsob, akým bude kurzor myši ovládaný počas svojho pohybu po obrazovke. V tejto kapitole navrhneme riešenie takéhoto problému.

V slovenskom jazyku je päť základných samohlások. Sú nimi a,e,i,o,u. Tieto samohlásky je možné pri analýze ich digitálneho signálu rozpoznať identifikovaním prvých dvoch formantov v spektrálnej obálke. Formanty a spektrálna obálka sú bližšie rozoberané v kapitolách 2 a 3. Každá samohláska má tieto formanty na rôznych intervaloch frekvenčného spektra. Tieto Intervaly sú rozpísané v tabuľke v kapitole 3.2. Keďže potrebujeme ovládať kurzor v štyroch smeroch, môžme si dovoliť jednu z týchto samohlások nepoužiť. Na základe intervalov formantov sme sa rozhodli nepoužiť samohlásku o, ktorej druhý formant sa čiastočne prekrýva s inými samohláskami. Jej vylúčením sme teda dosiahli diskrétnosť druhých formantov samohlások a,e,i a u.

V teoretickej časti tejto práce sme uviedli dva spôsoby, ktorými je možné dopracovať sa k spektrálnej obálke a následne identifikovať prvé dva formanty potrebné na určenie o akú samohlásku ide. V návrhu riešenia a následnej implementácii sme sa pre účel tejto práce rozhodli použiť oba prístupy, aby sme mali možnosť porovnať ich použiteľnosť, rýchlosť a spoľahlivosť pri rozpoznávaní samohlások. Tieto dva prístupy popísané v kapitole 3 sú prevod do frekvenčného spektra pomocou DFT a analýza rečového signálu pomocou LPC a Durbinovho algoritmu. Oba tieto postupy by v konečnej fáze mali viesť k spektrálnej obálke na ktorej je možné určiť formanty.

Program je možné rozdeliť na troch základné časti.

Ako prvé je potrebné zaznamenať signál. V prípade ovládania kurzoru myši hlasom ide teda o nahrávanie zvuku pomocou mikrofóna. Toto nahrávanie by malo prebiehať neustále počas celého behu programu aby bolo možné ovládať myš okamžite a bez potrebnej inicializácie nahrávania. Takýto spôsob je vhodný z hľadiska možného využitia programu ľuďmi s hendikepom, ktorým by inicializácia nahrávania zvuku potrebného na

aktiváciu pohybu kurzoru mohla pôsobiť problémy ak by to bolo napríklad stlačenie tlačidla a podobne. Z tohoto dôvodu navrhujeme aby bol zvukový záznam vykonávaný neustále počas celého behu programu. Pri nahrávaní zvuku je potrebné zvoliť správnu hodnotu pre vzorkovaciu frekvenciu aby sme boli schopný zaznamenať signál dostatočne presne.

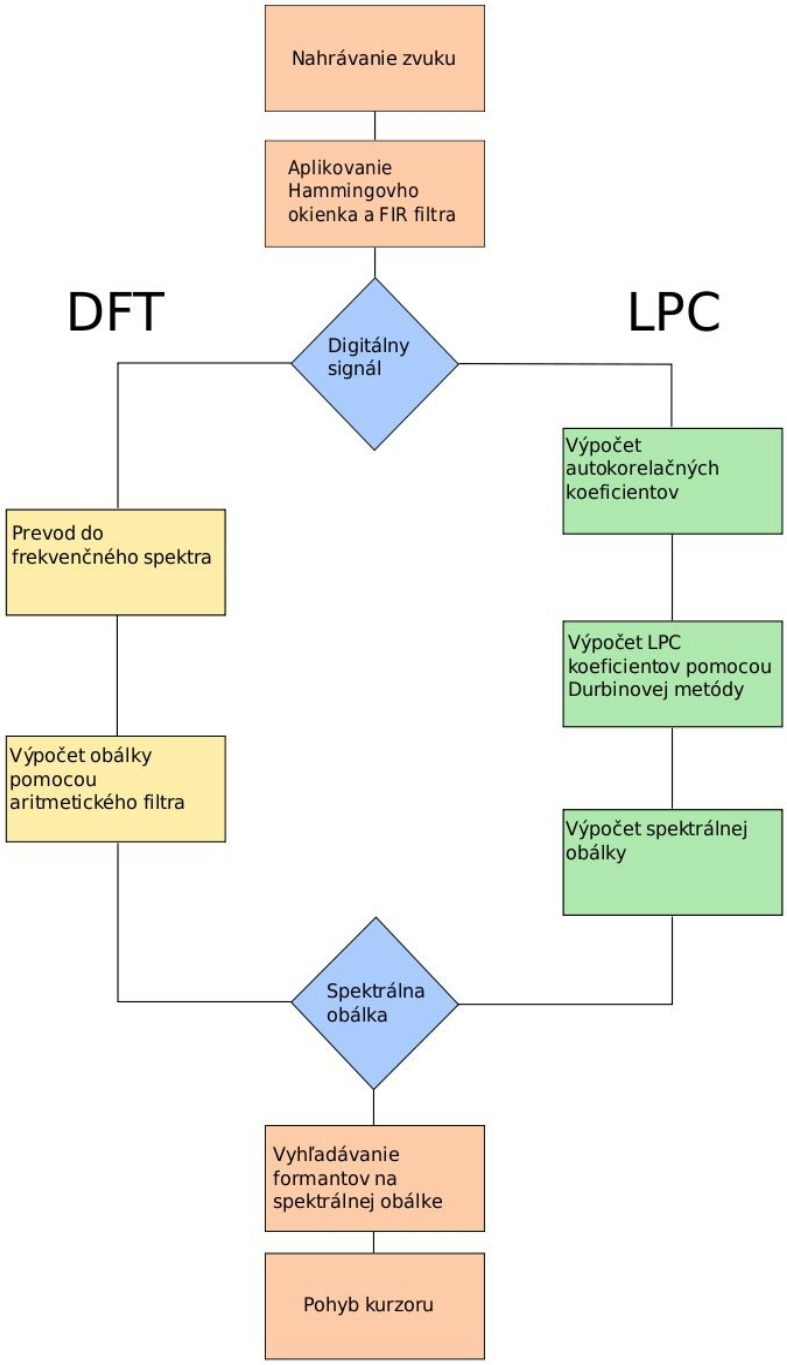
Druhým dôležitým bodom je spracovanie samotného digitálneho signálu. Aj v prípade DFT aj LPC je jedným z výsledkov analýzy spektrálna obálka. Pri DFT ju získame prevodom diskrétneho rečového signálu, získaného nahrávaním zvuku pomocou mikrofónu, do frekvenčného spektra. Obálku frekvenčného spektra následne získame jeho vyhladením použitím aritmetického filtra. LPC odhaduje parametre signálu z predchádzajúcich vzoriek signálu. Pracuje teda priamo s rečovým signálom. Vďaka LPC sme schopný získať auto korelačné koeficienty podľa vzťahu v13. Tieto následne využijeme pri Durbinovom algoritme podľa vzťahu v18 na vypočítanie koeficientov sústavy rovníc vyjadrenej vo v16. Durbinová metóda vo vzťahu v18 je vyjadrená rekurzívne. Dosadením koeficientov vypočítaných pomocou Durbinovej metódy je možné vyjadriť obálku frekvenčného spektra podľa vzťahu v20.

Frekvenčné spektrum a jeho obálka pre samohlásku “a” sú znázornené na obrázkoch č. 12 a 13 v kapitole 3.

Druhou časťou analýzy signálu je nájdenie prvých dvoch formantov. Na určenie prvých dvoch formantov môžme využiť vyhľadávanie lokálnych maxím v spektrálnej obálke. Nie každé takéto lokálne maximum však musí znamenať, že sa jedná o formant. Rozhodujúcim faktorom pri vyhľadávaní formantov je šírka pásma jednotlivých vrcholov. Pokiaľ je Šírka pásma väčšia ako 500Hz o formant sa spravidla nejedná.

Posledná tretia časť programu bude mať za úlohu priradenie nájdených formantov k samohláskam a posielanie príslušných súradníc operačnému systému.

Následujúci graf ukazuje priebeh programu. Po nahraní zvuku a získaní rečového signálu sa program delí na dve vetvy, ktoré predstavujú dva prístupy k získaniu obalky. Po vypočítaní spektrálnej obálky program hľadá formanty a nasleduje už len príkaz na pohyb kurzoru.



5. Implementácia

Program ovládaní kurzoru myši bol naprogramovaný pod systémom Ubuntu. V zmysle licencie GPL sme vyberali aj všetky prostriedky a ostatný software, ktorý sme použili pri vzniku tohto programu.

5.1. Použité nástroje

Jadro programu je naprogramované v jazyku C. Tento jazyk sme zvolili z dôvodu potreby rýchleho spracovaní nahratých dát a sile programovacieho jazyka. Z podobných dôvodov je v dnešnej dobe tento jazyk používaný pri väčšine programov ktoré sa zaoberajú spracovaním digitálneho signálu.

Pri nahrávaní zvuku sme použili knižnicu Simple.h z balíku knižníc produktu Pulse audio. Táto knižnica poskytuje jednoduché vytvorenie nahrávacieho servera a následné čítanie alebo zapisovanie dát z neho.

V tomto programe sme využili aj knižnicu GSL (GNU Scientific Library). Z nej sme využili optimalizovanú metódu pre výpočet FFT.

V nasledujúcich kapitolách podrobnejšie rozoberieme implementáciu jednotlivých funkcií programu.

5.2. Nahrávanie zvuku

Na nahrávanie zvuku sme použili triedu simple.h z projektu Pulse audio. Táto trieda umožňuje jednoduché vytvorenie servera, ktorý dokáže zaznamenávať zvuk. Na vytvorenie toho servera sa používa nasledujúci kód.

```
if (!(s = pa_simple_new(NULL, argv[0], PA_STREAM_RECORD, NULL, "record", &ss,   NULL,
NULL, &error)))

{

            fprintf(stderr, __FILE__": pa_simple_new() failed: %s\n", pa_strerror(error));

            goto finish;}
```

Argumenty použite v metóde pa_simple_new sú bližšie rozvedené v samotnej triede. Dôležitým argumentom v našom prípade je premenná ss, v ktorej sú zadefinované informácie o formáte výsledného signálu, počte použitých kanálov a vzorkovacej frekvencii. Vzorkovaciu frekvenciu sme pre tento program zvolili 16000. Takáto frekvencia by podľa Nyquistovej teórie mala stačiť na spoľahlivé zaznamenanie signálu s frekvenciami do 8 KHz a teda mala byť zároveň postačujúcou na zaznamenanie bežného hovoreného slova. Definovanie konštánt v premennej ss je zobrazené v nasledujúcom kóde.

```
static const pa_sample_spec ss =
{
	.format = PA_SAMPLE_S16LE,
	.rate = 16000,
	.channels = 1
};
```

Po tom ako sme vytvorili nahrávací stream je potrebné vedieť z neho čítať dáta a tieto načítané údaje v cykle zapisovať do potrebného formantu na ďalšie spracovanie. To nám zaručí nekonečný for cyklus v ktorom voláme metódu triedy simple.h na čítanie zo streamu pa_simple_read a metódu na zápis údajov loop_write. Táto funkčnosť je zachytená v nasledujúcom kóde.

```
for (;;)
{
	uint8_t buf[BUFSIZE];
	    if (pa_simple_read(s, buf, sizeof(buf), &error) < 0)
	{
	        fprintf(stderr, __FILE__": pa_simple_read() failed: %s\n",
pa_strerror(error));
	        goto finish;
	    }
	    if (loop_write(fn, buf, sizeof(buf)) != sizeof(buf)) {
	        fprintf(stderr, __FILE__": write() failed: %s\n", strerror(errno));
	        goto finish;
	    }
}
```

Metóda pa_simple_read zapisuje dáta do pola buf ktoré je typu uint8_t, čo je bezznamienkový integer, ktorý využíva na svoj zápis 8 bitov. Toto pole ma veľkosť počet prvkov BUFSIZE. BUFSIZE teda znázorňuje pocet vzoriek signálu ktoré budeme naraz

spracovávať. Tieto údaje potom posielame ako parameter do metódy loop_write ktora v cykle volá metódu na vypisovanie dát do konzoly. Metóda loop_write je zachytená v nasledujúcom kóde.

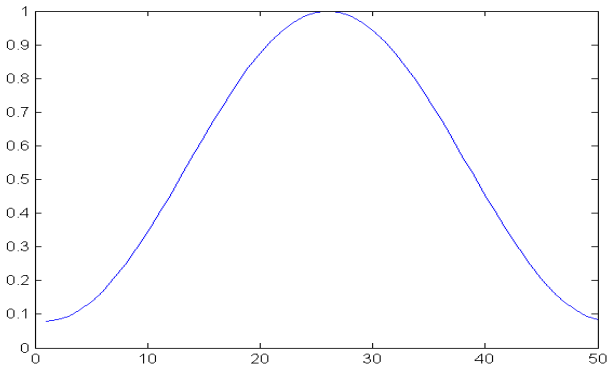
```
 static ssize_t loop_write(int fd, const void*data, size_t size)
{
        write(fd, data, size);
            int i;
            for (i = 0; i<sizeof(data); i+=2)
       {
                printf( "%d", ((short*)data)[i]);
                printf("\n");
            }
        return BUFSIZE;
}
```

Táto funkčnosť je dostačujúca na zachytenie potrebných dát, ktoré môžme následne spracovávať pomocou DFT a LPC a získať spektrálnu obálku.

5.3. Spracovanie signálu

Výsledkom nahrávania zvuku, pomocou triedy simple.h predstaveného v predošlej kapitole, je diskrétny digitálny rečový signál. Tento je potrebné ďalej spracovať. V tejto kapitole predstavíme implementáciu DFT a Durbinovho algoritmu. Oba tieto postupy by mali viesť k podobnému výsledku a to spektrálnej obálke.

Pri testovaní aplikácie sa ukázala nutnosť ďalšieho spracovania spracovania nahratého signálu z dôvodu nedostatočných výsledkov dosiahnutej obálky. Prvou úpravou bolo nahradenie pravouhlého okienka ktorým sme doteraz signál prechádzali okienkom Hammingovým. Pravouhlé okienko je v podstate okienko žiadne a signál sa necháva taký aký sa nahrá pomocou mikrofónu. Pri Hammingovom okienku sa jednotlivé vzorky signálu násobia koeficientami vypočítanými na základe funkcie Hammingovho okienka ktoré je znázornené na obrázku č. 15. Prenásobenie signálu vyzerá nasledovne.



Obr. č. 14: Hammingovo okienko

```
double HammingPole[BUFSIZE];
for(i = 0; i < BUFSIZE; i++)
{
	HammingPole[i] = (0.54 - 0.46*cos(2*M_PI*i/BUFSIZE-1))*pole[i];
}
```

Druhým spôsobom predspracovania signálu bolo prefiltrovanie signálu FIR (finite impluse response) filtrom. Tento lineárny filter funguje na násobení vzoriek koeficientami. Každá vzorka je nahradená súčtom samej seba a N-1 predchádzajúcich vzoriek. N je rád filtra. Každá z týchto vzoriek je naviac vynásobená koeficientom vypočítaným na základe vstupných parametrov, ktorými sú vzorkovacia frekvencia, hraničné frekvencie, ktoré chceme v signále zvýrazniť a samotného rádu modelu. Tieto koeficienty sme získali z voľne dostupného webového kalkulátora. Prefiltrovanie signálu FIR filtrom vykonáva nasledovný kód.

```
int FiRRad;
FiRRad = 51;
double HammingPoleFIR[BUFSIZE];
for(i = 0; i < BUFSIZE; i++){
	HammingPoleFIR[i] = 0.0;
	int MaxRad;
	if (i < FiRRad)  MaxRad = i+1;
	else   MaxRad = FiRRad;
	for (j = 0; j<MaxRad; j++){
		HammingPoleFIR[i] += HammingPole[i-j]*FiRKoef[j];
	}
}
```

V počítačových algoritmoch, ktoré rátajú DFT sa často používa zefektívnený algoritmus na výpočet DFT a to FFT. Na výpočet FFT sme využili optimalizovanú metódu knižnice GSL.

```
gsl_fft_real_radix2_transform(pole, 1, BUFSIZE/2);

gsl_fft_real_radix2_transform (double data[], size_t stride, size_t n)
```

Funkcia gsl_fft_real_radix2_transform ráta pole FFT dĺžky n. Vstupujúce parametre sú pole reálnych čísel s názvom pole, dĺžka kroku 1 a veľkosť výstupného pola BUFSIZE/2. Výstupom tejto metódy je polo-komplexná sekvencia uložená v poli. Usporiadanie polo-komplexného pola používa nasledujúcu schému: pre $k < n/2$ je reálna časť k-teho prvku uložená na indexe k výstupného pola a korešpondujúca imaginárna časť tohto prvku je uložená v lokácii pola s indexom n-k. Prvky s indexom $k > n/2$ môžu byť rekonštruované použitím symetrie z_k = z^*_{n-k}. Prvky k=0 a k=n/2 sú oba čisto reálne a teda vo výsledku nemáme ich imaginárnu zložku. Ich reálne časti sú uložené na lokáciach poľa s indexmi 0 a n/2 zatiaľčo ich imaginárne časti nie sú uložené.

Nasledujúca tabuľka ukazuje vzťah medzi výstupným poľom a ekvivalentnými výsledkami nadobudnutými uvažujúc vstupné dáta ako komplexnú sekvenciu s nulovou imaginárnou časťou a krokom 1.

```
        complex[0].real    =    data[0]
        complex[0].imag    =    0
        complex[1].real    =    data[1]
        complex[1].imag    =    data[n-1]
        ..............          ..............
        complex[k].real    =    data[k]
        complex[k].imag    =    data[n-k]
        ..............          ..............
        complex[n/2].real  =    data[n/2]
        complex[n/2].imag  =    0
        ..............          ..............
        complex[k'].real   =    data[k]        k' = n - k
        complex[k'].imag   =   -data[n-k]
        ..............          ..............
        complex[n-1].real  =    data[1]
        complex[n-1].imag  =   -data[n-1]
```

Tento výstup sme následne zaznamenali do dvojrozmerného poľa reálnych čísel.

```
polefurier[0]= log(sqrt(pole[0]*pole[0]));

polefurier[BUFSIZE/4]= log(sqrt(pole[BUFSIZE/4]*pole[BUFSIZE/4]));

for (i = 1; i<BUFSIZE/4; i++){

      polefurier[i] = log(sqrt((pole[i]*pole[i]) + (pole[BUFSIZE/2-i]*pole[BUFSIZE/2-
i])));

}
```

Polefurier sme naplnili v zmysle predchádzajúcej tabuľky.

Na získanie obálky frekvenčného spektra zo samotného frekvenčného spektra je potrebné postupne spriemerovať amplitúdy susediacich frekvencií. Túto funkčnosť vykonáva nasledujúci kód. Je možné si všimnúť, že veľkosť poľa reálnych čísel s ktorým pracujeme teraz, je už len štvrtinová oproti pôvodnému BUFSIZE. Tento jav spôsobil najskôr bitový posun pri konverzii s typu uint8_t na typ short a následne metóda, ktorou sme naplnili premennú polefurier, kde sme zrátali reálne a imaginárne zložky. Pri spriemerovaní poľa polefurier tento počet môžme ďalej redukovať. Ako vidíme v nasledujúcom kóde úvodné naplennei poľa obálka tvorí prvé spriemerovanie každého štvrtého prvku poľa polefurier so štyrmi okolitými prvkami. Ak teda chceme vypočítať hodnotu pre lokáciu obalka[n], je potrebné spriemerovať hodnoty poľa polefurier z indexov n-2, n-1, n, n+1 a n+2.

```
for (i = 2; i<BUFSIZE/16-2; i++)

{

      obalka[i] = (polefurier[i*4-2] + polefurier[i*4-1] + polefurier[i*4] +
polefurier[i*4 + 1] + polefurier[i*4 + 2])/5;

}

int k;
```

Hraničné hodnoty, ktoré musia byť vypočítané iným spôsobom sú teda pre pole obálka dĺžky k na indexoch 0, 1, k-2 a k-1. Tieto sa rátajú podľa nasledovného algoritmu.

```
double obalka[BUFSIZE/16];

obalka[0] = (polefurier[0] + polefurier[1] + polefurier[2])/3;

obalka[1] = (polefurier[0] + polefurier[1] + polefurier[2] + polefurier[3])/4;

obalka[BUFSIZE/16-1] = (polefurier[BUFSIZE/4-1] + polefurier[BUFSIZE/4-2]+
polefurier[BUFSIZE/4-3])/3;

obalka[BUFSIZE/16-1] = (polefurier[BUFSIZE/4-1] + polefurier[BUFSIZE/4-2]+
polefurier[BUFSIZE/4-3] + polefurier[BUFSIZE/4-4])/4;
```

Túto redukciu pri výpočte poľa obálka sme urobili z dôvodu, že jedno aplikovanie aritmetického filtra nie je dostatočné aby sa získané frekvenčné spektrum zachytené v poli polefurier dostatočne vyhladilo na to aby sme mohli spoľahlivo identifikovať prvé dva formanty. Počet spriemerovaní potrebných na dosiahnutie uspokojivého výsledku priamo-úmerne rastie s počtom prvkov vstupného poľa. Ak by sme pri pôvodných BUFSIZE/4 záznamoch potrebovali približne 200-250 iterácii teraz nám ich postačí približne 50. Je treba si uvedomiť, že tieto operácie sa vykonávajú nad každým prvkom poľa obalka počas každého spracovávaného časového úseku. Ak teda spracovávame časové intervaly dĺžky jednej sekundy, má teda výsledné pole obalka, pri vzorkovacej frekvencii 16000, 1000 prvkov, na ktoré je 50 krát aplikovaný aritmetický filter. Bez predošlej redukcie počtu prvkov v poli obalka by to bolo 4000 prvkov, teda o 3000*50 viac operácii spriemerovania. Ďalšie spriemerovania sú identické s predchádzajúcimi. Jediným rozdielom je, že vstupné hodnoty sa tentokrát berú priamo z poľa obalka aj zapisujú do poľa obalka.

Spriemerovanie poľa obalka v 50 iteráciách vykonáva nasledujúci kód.

```
for (k = 0; k<50; k++)

{

              obalka[0] = (obalka[0] + obalka[1] + obalka[2] + obalka[3])/4;

              obalka[1] = (obalka[0] + obalka[1] + obalka[2] + obalka[3])/4;

              obalka[BUFSIZE/16-1] = (obalka[BUFSIZE/16-1] + obalka[BUFSIZE/16-2]+
obalka[BUFSIZE/16-3])/3;

              obalka[BUFSIZE/16-2] = (obalka[BUFSIZE/16-1] + obalka[BUFSIZE/16-2]+
obalka[BUFSIZE/16-3] + obalka[BUFSIZE/16-4])/4;

       for (i = 2; i<BUFSIZE/16-2; i++)

       {

               obalka[i] = (obalka[i-2] + obalka[i-1] + obalka[i] + obalka[i + 1] +
obalka[i + 2])/5;

       }

}
```

Frekvenčné spektrum a následne jeho obálku získanú aritmetickým spriemerovaním amplitúd jeho frekvencií zobrazujú obrázky 13 a 14 v kapitole 3.2.3 (jedná sa o samohlásku “a”).

```
static void AutoKorelacneKoef(float *in, int Rad, float *out){
    int i, k;
    for(i = 0; i<=Rad; i++)
    {
        out[i] = 0.0;
        for (k = 0; k<(BUFSIZE-1-i); k++)
        {
            out[i] += in[k] * in[k+i];
        }
    }
}
```

Druhou metódou, ktorej algoritmus je možné vidieť v metóde AutoKorelacneKoef(), na získanie spektrálnej obálky je LPC s použitím Durbinovho algoritmu. Táto metóda je predstavená v kapitole 3.3. Pre implementáciu samotného Durbinovho algoritmu je potrebné najskôr vyrátať takzvané korelačné koeficienty podľa vzťahu v13 v kapitole 3.3.1 Matematická definícia LPC.

Tento algoritmus pracuje s pôvodným rečovým signálom a nie s amplitúdami z frekvenčného spektra. Parameter Rad ktorý vstupuje do metódy AutoKorelacneKoef je rád predikčného modelu LPC. V kapitole 3.3 je tento rád predstavovaný ako rád Q. Ďalšie parametre sú polia in a out. In je pole vstupujúcich hodnôt daného úseku rečového signálu a out je pole auto-korelačných koeficientov pre daný signál. Po výpočte auto-korelačných koeficientov je možne spustiť rekurzívny algoritmus pre výpočet koeficientov a[] Durbinovej metódy zachytenom v vzťahu v18 kapitoly 3.3. Tento algoritmus vyzerá nasledovne.

```
static void Durbin(double *r, int Rad, double *a, double *G){
        double k[Rad+1]; double aPred[Rad+1];
        double e;
        int i,j;
        *G = 0;
        e = r[0];
        for (i = 0; i <= Rad; i++)
            a[i] = 0.0;
        for(i = 1; i<=Rad; i++){
            k[i] = -r[i];
            for(j = 1; j<i; j++){
                aPred[j] = a[j];
                k[i] -= (a[j] * r[i - j]);
            }
            if (fabs((float)e) < FLT_EPSILON) {
                e = 0.0f;
                break;}
            k[i] /= e;
            a[i] = k[i];
            for (j = 1; j < i; j++)
                a[j] = aPred[j] + k[i] * aPred[i - j];
            e *= 1.0f - k[i] * k[i];
           }
        *G = sqrt(e);}
```

Vstupné parametre float *r, int Rad, float *a, float *G sú v tom poradí autokorelačné koeficienty AK vypočítané v predchádzajúcej metóde AutoKorelacneKoef, rád modelu, pole a[] veľkosti rád modelu +1 do ktorého sa uložia koeficienty a[] a &G do ktorého sa na záver uloží chyba výpočtu pri Durbinovej metóde. Pole k[], ktoré sa využíva v tejto metóde obsahuje hodnoty priebežných medzi-výsledkov PARCOR.

Posledným krokom k spektrálnej obálke je implementácia vzťahu v20. Vstupujúcimi parameterami float *a, float G, int Q, float *H sú v tomto poradí pole koeficientov a[], chyba predikcie G, rád modelu a pole do ktorého sa uložia výsledné hodnoty spektrálnej obálky. Implementácia tohto vzťahu je v nasledujúcej metóde Spektrum.

```
static void Spektrum( double *a, double G, int Q, double *H){
        double citatel;
        double pi = 3.14159265358979323846264338327950288419 7;
        citatel = G;
        int i, ii, f;
        for (ii = 0; ii<BUFSIZE/4; ii++){
            f = ii;
            double realna;
            double imaginarna;
            double menovatel;
            menovatel = 0;
            double = 0;
            imaginarna = 0;
            for (i = 1; i<=Q; i++){
                double += a[i] * cos(i*(2*pi*f/8000));
                imaginarna += a[i] * sin(i*(2*pi*f/8000));
            }
            realna ++;
            menovatel = sqrt(realna*realna + imaginarna*imaginarna);
            H[ii] = citatel/menovatel;
        }
    }
```

Čitateľom zo vzťahu v20 je v tejto metóde premenná citatel do ktorej sme priradili G a menovateľom je premenná menovatel vypočítaná podľa definície vo vzťahu v20 kde platí, že 0 = 1, 0≤ i≤ Q. ω - predstavuje kruhovú frekvenciu ktorú môžeme do vzťahov (v19) a (v20) dosadiť zo vzťahu $\omega=2\Pi f/F_v$, kde F_v[Hz] je vzorkovacia frekvencia signálu s(t) a f[Hz] je meniaca sa frekvencia. Platí, že $f\leq 0{,}5F_v$.

5.4. Pohyb kurzoru

Dôležitou časťou implementácie je samotný pohyb kurzoru. Na to aby sme mohli pohnúť kurzor príslušným správnym smerom je potrebné najskôr identifikovať formanty v spektrálnej obálke. Výsledok by mal byť rovnaký bez ohľadu na to, či sme spektrálnu obálku získali pomocou FFT alebo pomocou LPC. V nasledujúcom kóde uvádzame príklad pre identifikáciu prvých dvoch formantov.

```
double Formant1 = 0; double Formant2 = 0;
for (i = 1; i<BUFSIZE/16 -1; i++)
{
	if((obalka[i]>obalka[i-1]) && (obalka[i]>obalka[i+1]) && (obalka[i] > 12))
		if (((double)i*16000/(1024*2)) > 210){
			Formant1 = (double)i*16000/(1024*2);
			break;
		}
}
for (i = 1; i<BUFSIZE/16 -1; i++){
	if((obalka[i]>obalka[i-1]) && (obalka[i]>obalka[i+1]) && (obalka[i] > 12))
		if (((double)i*16000/(1024*2)) > Formant1){
			Formant2 = (double)i*16000/(1024*2);
			break;
		}
}
```

Tento kód slúži čisto na ukážku spôsobu vyhľadávania formantov. Možná optimalizácia tohoto kódu by bolo ohraničenie cyklov len po potrebné maximálne a minimálne hodnoty samohlások. Pre účely testovania sme však potrebovali vidieť aj nesprávne výsledky spektrálnych obálok a teda formanty na atypických miestach.

Pokiaľ sa jedná o samohlásku. Prvé dva formanty by sa mali nachádzať na frekvenciách uvedených v tabuľke v kapitole 3.1. Jednoduchou kontrolou formantov vieme teda určiť o akú samohlásku ide a poslať príslušný príkaz kurzoru myši. Je vhodné pohyb myši umiestniť do samostatného threadu a priebežne posielať informáciu o tom či a aká samohláska bola identifikovaná predchádzajúcimi algoritmami. Tento spôsob by mal zabezpečiť plynulosť pohybu myši.

Nasledujúci kód ukazuje kontrolu formantov pre samohlásku “a” a príkaz systému na pohyb myši vpravo o delta_x pixlov.

```
if ((Formant1 > 450) && (Formant1 < 700) && (Formant2 > 1000) && (Formant2 < 1200))
{
	Display *display = XOpenDisplay(0);
	// move pointer relative to current position
	XWarpPointer(display, None, None, 0, 0, 0, 0, delta_x, 0);
	XcloseDisplay(display);}
```

Druhým možným spôsobom určovania samohlásky pomocou formantov je výpočet euklidovskej vzdialenosti od priemernej hodnoty formantov danej samohlásky. Pri rozhodovaní sa ktorým smerom sa má kurzor pohnúť vyberáme najmenšiu hodnotu vypočítaných euklidovských vzdialeností. Premenné Formant1x a Formant2x obsahujú priemerné hodnoty formantov. Tieto môžeme určiť ručne podľa teórie, alebo pomocou kalibrácie, kedy nahráme a spriemerujeme niekoľko vzoriek pre každú samohlásku a vypočítané formanty spriemerujeme.

```
vzdialenostA = sqrt(((Formant1-Formant1a)*(Formant1-Formant1a))+((Formant2-
Formant2a)*(Formant2-Formant2a)));

vzdialenostE = sqrt(((Formant1-Formant1e)*(Formant1-Formant1e))+((Formant2-
Formant2e)*(Formant2-Formant2e)));

vzdialenostI = sqrt(((Formant1-Formant1i)*(Formant1-Formant1i))+((Formant2-
Formant2i)*(Formant2-Formant2i)));

vzdialenostU = sqrt(((Formant1-Formant1u)*(Formant1-Formant1u))+((Formant2-
Formant2u)*(Formant2-Formant2u)));
```

6. Výsledky

6.1. Zápis a vykresľovanie dát

V tejto kapitole predstavíme výsledky spracovaného signálu.

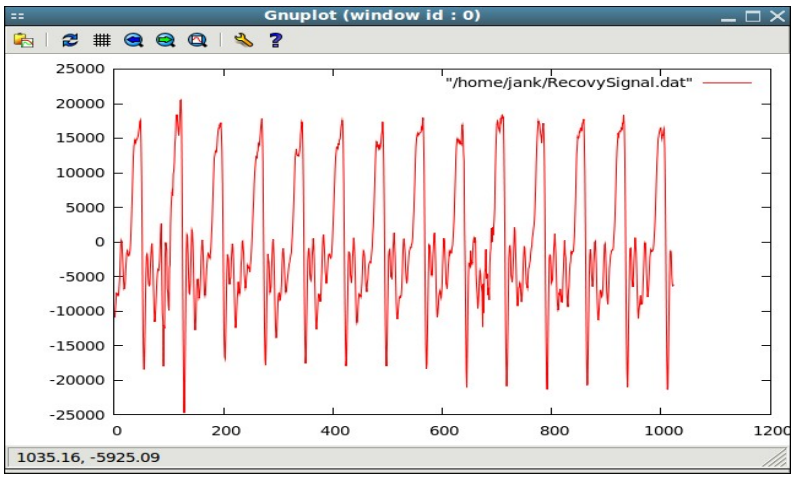
Na grafické znázornenie nahratých a spracovaných dát sme použili program gnuplot. Tento program je schopný vykresľovať grafy z dátových súborov. Do týchto súborov sme ukladali dáta o rečovom signále, frekvenčnom spektre a spektrálnych obálkach. Zápis do dátových súborov spracováva jednoduchý kód.

```
file = fopen("cesta/RecovySignal.dat", "w");
for(i = 0; i<BUFSIZE; i++){
    fprintf(file, "  %f     %f\n" , (double)i, pole[i]);
} fclose(file);
```

Vykresľovanie dát pomocou programu gnuplot sa realizuje z príkazovej konzoly. Najprv príkazom „gnuplot“ spustíme program a nasledovným príkazom

```
plot "cesta k dátovému súboru" with line
```

spustíme vykreslenie grafu podľa dátového súboru. Výsledok takéhoto vykreslenia pre rečový signál samohlásky „a“ je znázornený na obrázku 15.



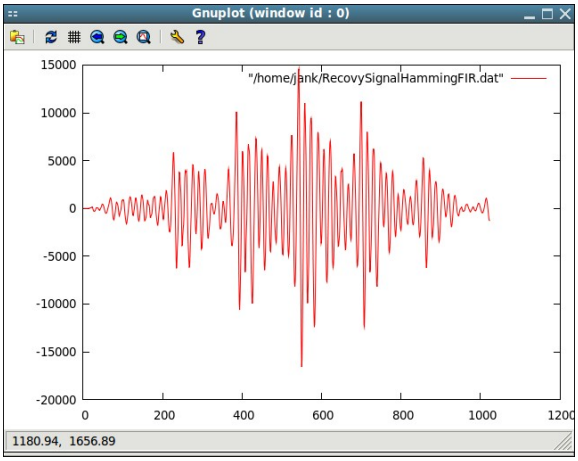
Obr.č. 15: Rečový signál samohlásky „a“ (1024 vzoriek)

6.2. Rečový signál

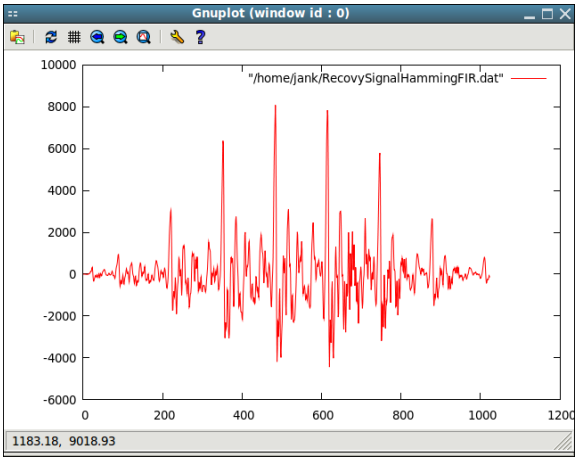
Prvým štádiom pri spracovaní digitálneho signálu je nahranie rečového signálu. Tento proces sa deje prevodom analógového signálu na diskrétny digitálny signál a je podrobnejšie popísaný v kapitole 2.

Na nasledujúcich obrázkoch je možné vidieť digitálny signál nahratý našim programom. Tento signál bol nahratý so vzorkovacou frekvenciou 16000. Na každom obrázku je znázornených 1024 vzoriek signálu.

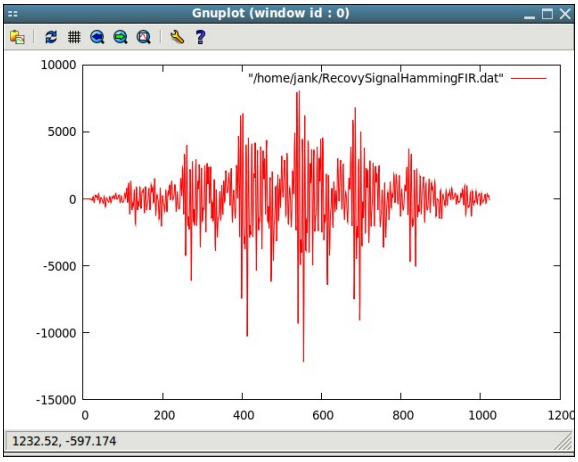
Badateľný rozdiel v rečovom signáli oproti obrázku č. 16 je spôsobený aplikovaním Hammingovho okienka a FIR filtra na signál.



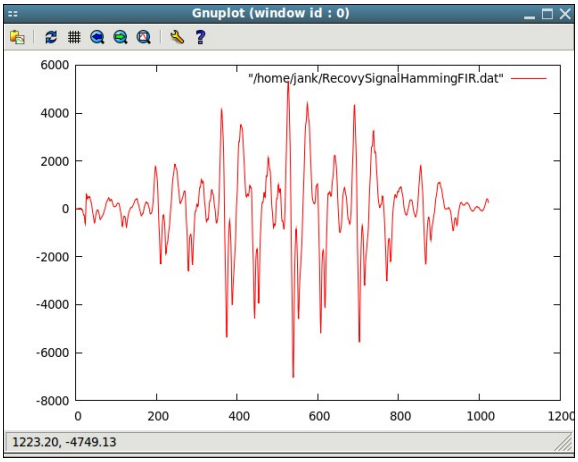
Samohláska „a“



Samohláska „e“



Samohláska „i“

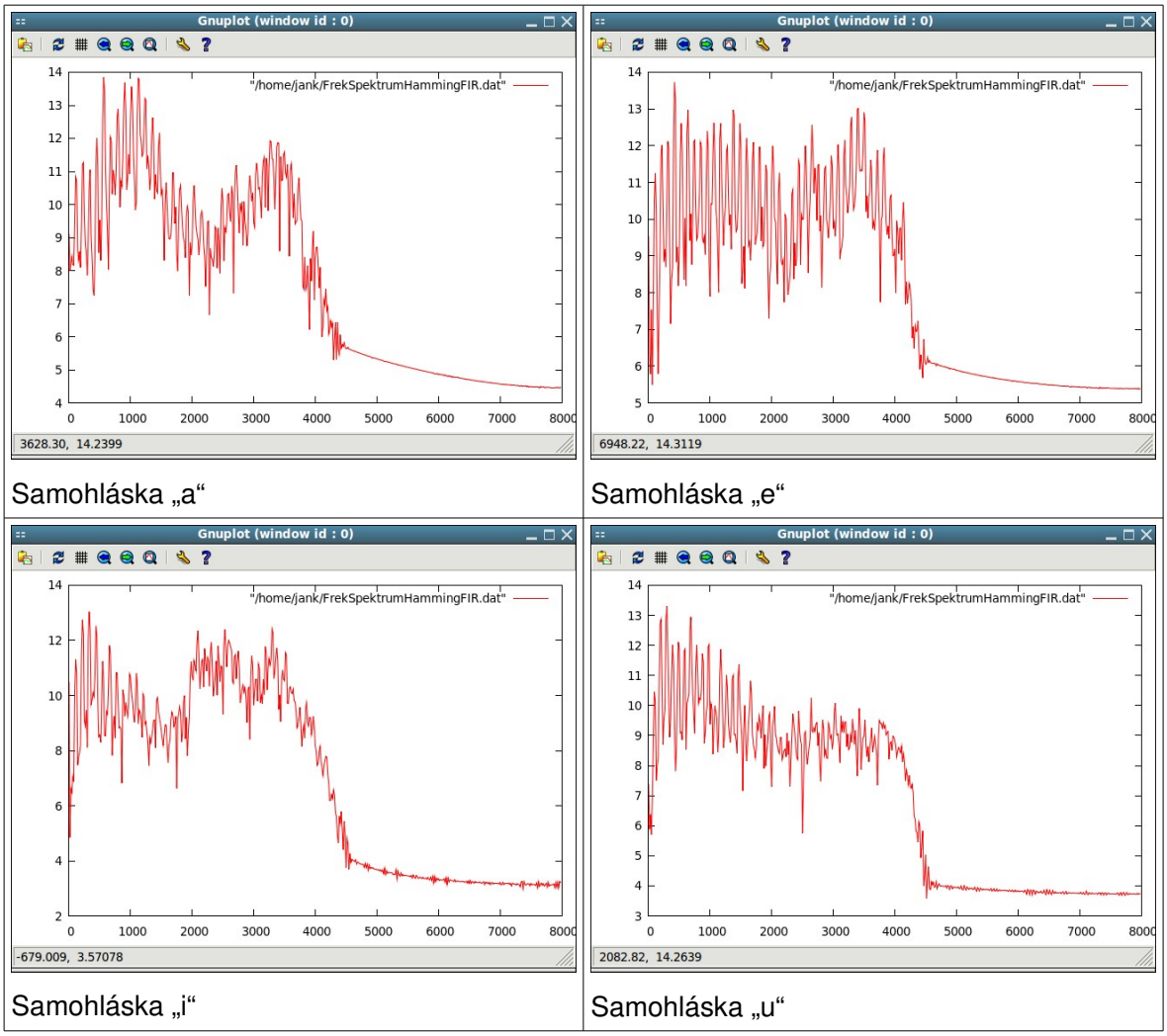


Samohláska „u“

Obr.č. 16: Rečový signál samohlások „a“, „e“, „i“ a „u“ po použití Hammingovho okienka a FIR filtra

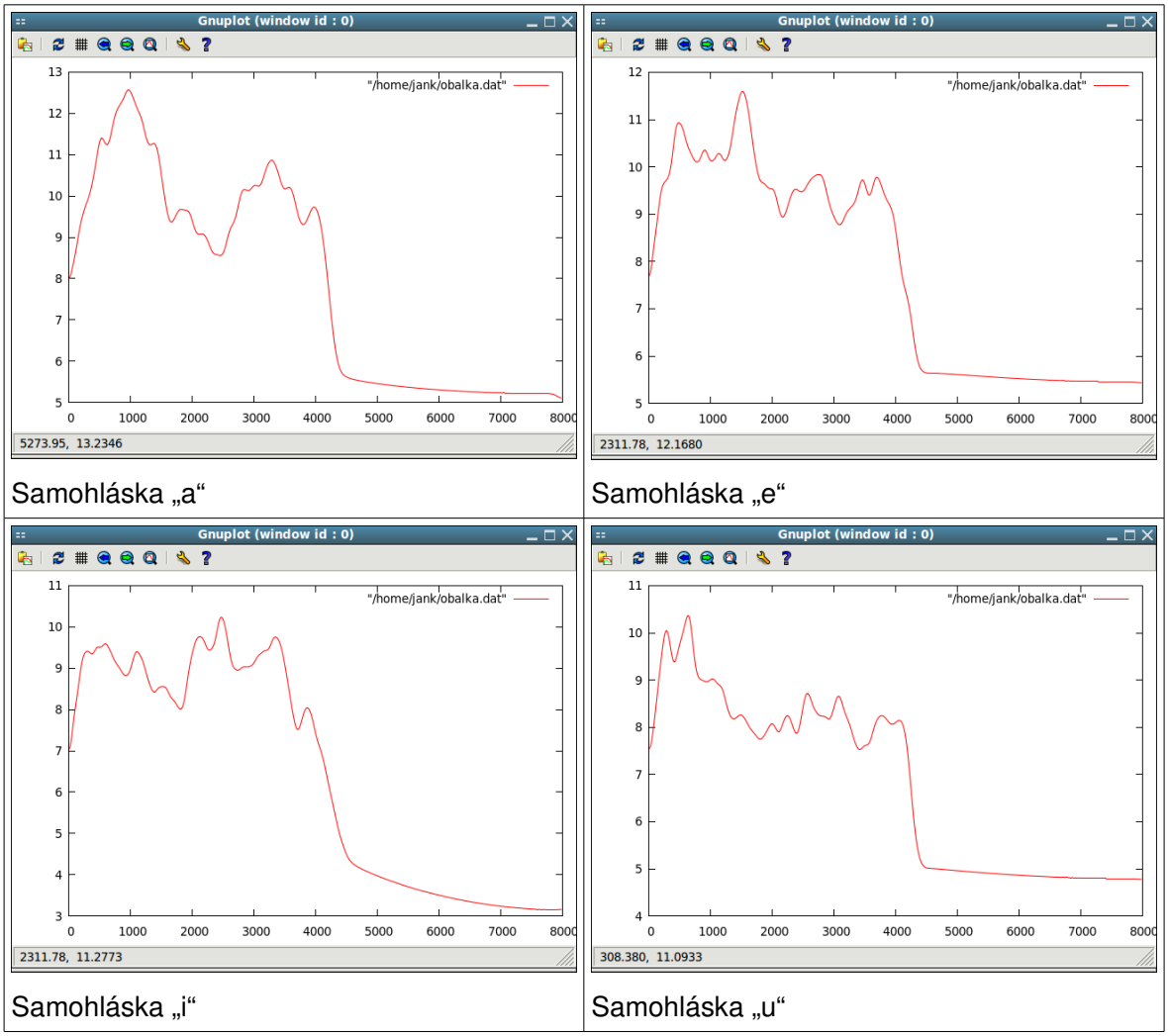
6.3. Výsledky DFT

Frekvenčné spektrum získané pomocou DFT, ktorú sme aplikovali na rečový signál upravený Hammingovým okienkom a FIR filtrom je znázornené na obrázku č. 17. Toto spektrum sa vyznačuje charakteristickým poklesom amplitúd frekvencií po hodnote 4000. Tento pokles spôsobil FIR filter, ktorého koeficienty boli vyrátané s ohľadom na túto hraničnú frekvenciu. Takýto FIR filter sme zvolili z toho dôvodu, že za touto hranicou sme už neočakávali dôležité informácie vo frekvenčnom spektre z hľadiska rozpoznania o akú samohlásku ide. Inými slovami druhý formant s najvyššou frekvenciu má samohláska „i“, ktorá ho má na intervale 2000-3000 Hz. Z hľadiska spektrálnej obálky je teda užitočné ak budú frekvencie za touto hranicou potlačené.



Obr.č. 17: Frekvenčné spektrum samohlások „a“, „e“, „i“ a „u“

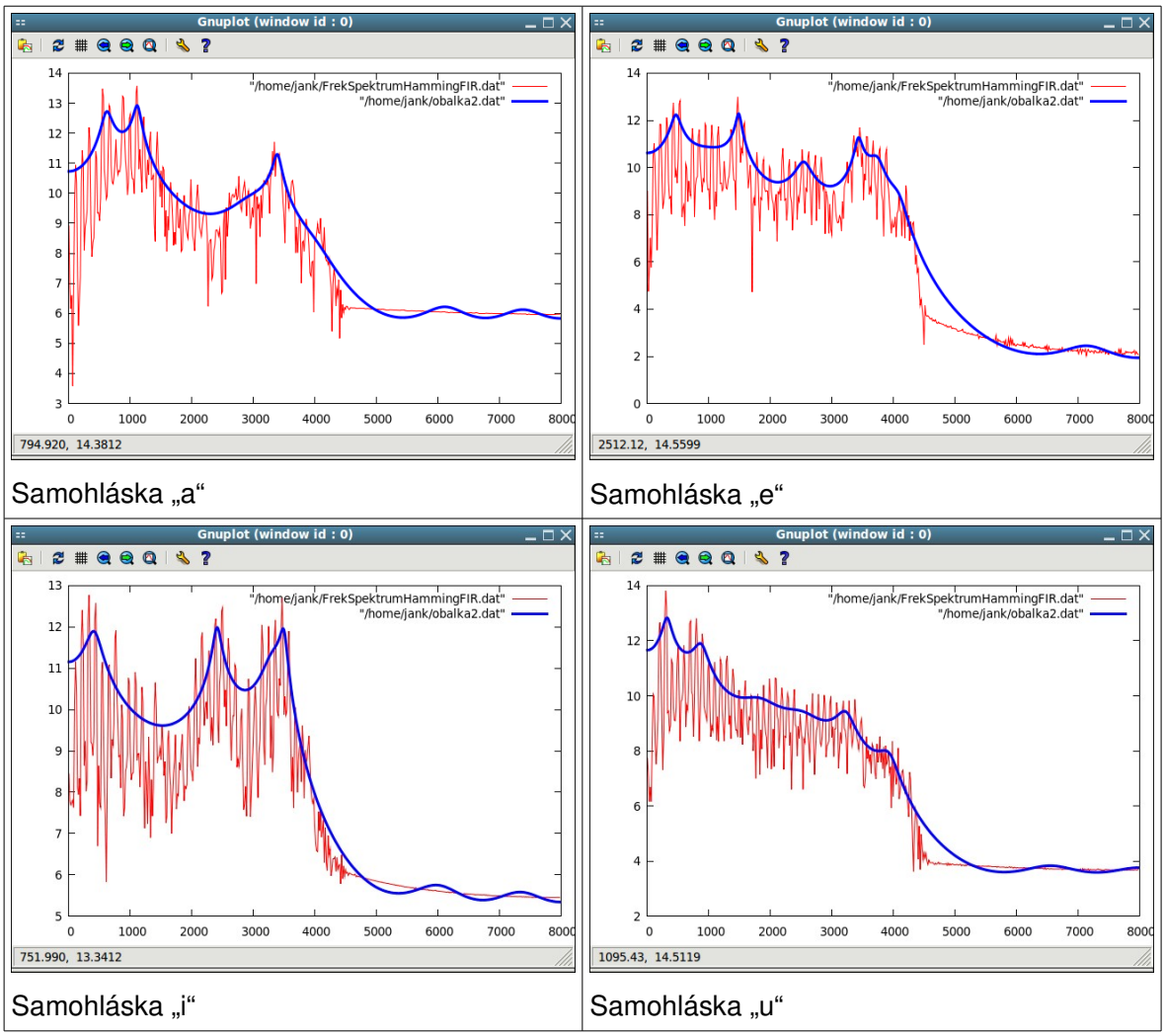
Pomocou spriemerovania amplitúd vo frekvenčnom spektre sme získali spektrálnu obálku. Na obrázku č. 18 je znázornená takáto obálka po sedem násobnom spriemerovaní. Vyhľadávanie formantov v takejto obálke bolo sťažené falošnými formantami, ktoré sa nachádzajú predovšetkým pri samohláskach „e“ a „i“. Tieto samohlásky majú oproti zvyšným dvom väčšiu vzdialenosť medzi prvými dvoma formantami. Týmto falošným formantom sa dá predísť viacnásobným spriemerovaním. To má však za následok posun frekvencií reálnych formantov. Pri zvolení tohto prístupu je teda potrebné na základe nájdených formantov pre jednotlivé samohlásky posunúť intervaly, s ktorými sa budú formanty porovnávať, alebo zvoliť metódu vhodnú metódu kalibrácie pred samotným pohybom kurzoru.



Obr.č. 18: Spektrálna obálka „a“, „e“, „i“ a „u“ získaná z frekvenčného spektra

6.3. Výsledky LPC

Druhým použitý spôsob vypočítania spektrálnej obálky je pomocou koeficientov LPC. Takto získaná obálka spolu s frekvenčným spektrom je znázornená na obrázku č. 19.



Obr.č. 19: Spektrálna obálka samohlások „a“, „e“, „i“ a „u“ získaná pomocou koeficientov LPC

V porovnaní s obálkou získanou pomocou piremerovania frekvenčného spektra sú formanty v LPC obálke jednoznačnejšie. U testovaného subjektu, ktorého výsledky sú zobrazené na obrázku č. 19 je možné jasne identifikovať prvé dva formanty všetkých štyroch testovaných samohlások. Je tiež možné vidieť, že frekvencie, na ktorých sa formanty nachádzajú sú identické s frekvenciami uvedenými v teoretickej časti práce.

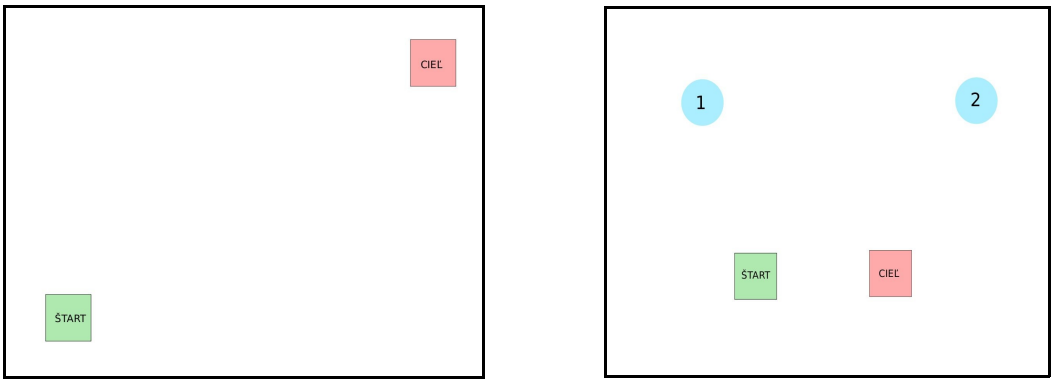
6.4 Kalibrácia formantov a testovanie

Podľa predošlých výsledkov sme na testovacie účely zvolili spektrálnu obálku získanú pomocou LPC koeficientov.

Na určovanie o akú samohlásku ide sme zvolili výpočet euklidovskej vzdialenosti nameraných formantov od priemernej hodnoty formantov.

Teoretický základ predstavený v kapitole 3.1 hovorí, že tieto formanty by sa mali nachádzať na daných intervaloch nezávisle od testovaného subjektu a teda nie je potrebná kalibrácia programu pri jednotlivých testoch. Naše testovanie však ukázalo, že takáto kalibrácia je potrebná. Ak boli priemerné hodnoty nastavené ako stredné hodnoty intervalov v ktorých by sa mali formanty nachádzať podľa tabuľky v kapitole 3.1 , z ôsmich testovaných subjektov len štyria dosahovali uspokojivé výsledky.

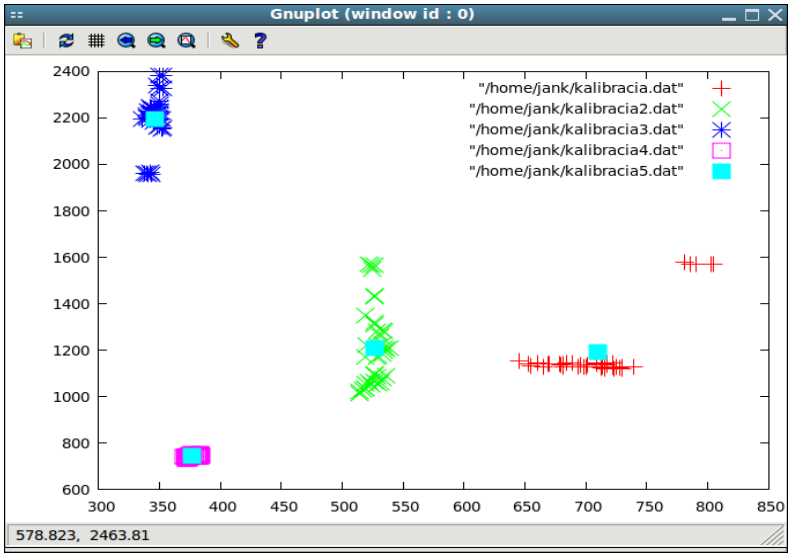
Subjekty boli testované v dvoch testoch zobrazených na obrázku č 20. V oboch testoch mali za úlohu prejsť kurzorom myši zo štvorčeka s nápisom štart do Štvorčeka s nápisom cieľ. V druhom teste museli navyše počas cesty prejsť kruhmi s číslami 1 a 2 v danom poradí. Druhý test bol navrhnutý tak aby boli respondenti nútený použiť všetky štyri smery pohybu a teda všetky použité samohlásky.



Obr. č. 20: schéma úloh použitých pri testovaní programu

Z dôvodu nízkej úspešnosti pohybu kurzoru sme sa rozhodli implementovať kalibráciu priemernej hodnoty formantov. Táto kalibrácia prebieha na začiatku programu, kde testovaný subjekt postupne nahráva vzorky pre každú z používaných samohlások. Z tých sú následne vypočítané formanty. Priemer týchto formantov sa potom uloží ako

priemerná hodnota od ktorej sa počas ovládania kurzoru počíta euklidovská vzdialenosť. Výsledok takejto kalibrácie je znázornený na obrázku č. 21. Na tomto obrázku je z každej samohlásky vypočítaných 40 prvých a druhých formantov. Prvý formant je nanesený na os x a druhý na os y. Priemerné hodnoty týchto formantov znázorňujú svetlomodré štvorčeky. Tieto hodnoty sú smerodajné pri následnom rozpoznávaní o akú samohlásku sa jedná a to tak že vypočítame euklidovskú vzdialenosť aktuálnych nájdených formantov od všetkých štyroch priemerných hodnôt a vyberieme najmenšiu.



Obr. č. 21: Priemerné hodnoty formantov po kalibrácii

Po takejto kalibrácii formantov vykazovalo šesť z ôsmich testovaných subjektov uspokojivé výsledky v oboch testoch, pričom nedostatočná úspešnosť rozpoznávania, pri zvyšných dvoch, mohla byť spôsobená okolitým hlukom Každý z týchto šiestich dokázal náročnejší test splniť do 10 sekúnd.

7. Záver

Cieľom tejto práce bolo navrhnúť a implementovať program, ktorý dokáže ovládať kurzor myši pomocou nepretržitého vyslovovania samohlások. Tento cieľ sa podarilo naplniť. Boli použité dva prístupy. Diskrétna Furierová transformácia a lineárne prediktívne kódovanie. Oba prístupy sa podarilo implementovať.

Na rozlišovanie samohlások bolo použité vyhľadávanie prvých dvoch formantov v spektrálnej obálke signálu. Podľa naštudovanej teórie sa tieto formanty nachádzajú na charakteristických intervaloch frekvenčného spektra signálu. Tieto intervaly by sa nemali posúvať v závislosti na rečníkovi. Toto tvrdenie sa nám pri testovaní nepotvrdilo vzhľadom na malú úspešnosť rozpoznávania pre rôznych rečníkov. Preto bolo potrebné implementovať kalibráciu týchto intervalov pre každého rečníka samostatne.

Po kalibrácii sa výsledky respondentov, na ktorých bol program testovaný, viditeľne zlepšili. Kedy šesť z ôsmich respondentov preukazovalo dobré výsledky v oboch predložených testoch predstavených v kapitole 6.4.

Horšie výsledky v prípade dvoch respondentov mohli byť spôsobené nie ideálnymi akustickými podmienkami v čase testovania a tiež nedostatočnou kalibráciou.

Z dvoch použitých prístupov sa ako účinnejšia a spoľahlivejšia ukázalo pre tento typ programu lineárne prediktívne kódovanie, vzhľadom na kvalitu spektrálnej obálky v ktorej sa vyhľadávali prvé dva formanty určujúce pre jednotlivé samohlásky.

Ako pomocné metódy, nutné na zvýraznenie charakteristických frekvencií signálu, boli použité Hammingovo okienko a FIR filter.

Prínos tejto práce spočíva v samotnom návrhu programu a implementovaní algoritmov, ktoré môžu byť použité pre vývoj programu použiteľného v praxi ľuďmi, ktorý nie sú schopní ovládať počítač pomocou dostupných technológií.

V ďalších fázach vývoja tohto programu je možne zdokonaliť rozpoznávanie samohlások aplikovaním viacerých kritérií, ako sú napríklad vzdialenosť medzi formantmi alebo vylepšená kalibrácia. Tiež je potrebná implementácia Ďalších funkcií kurzoru a používateľsky príjemného interfacu.

Bibliografia

[1] Doc. Ing. Psutka Josef, Csc. – Komunikace s počítačem mluvenou řečí, 1995, ISBN 80-200-0203-0

[2] Steven W. Smith – Digital Signal Processing, A practical guide for Engineers and Scientists, 2003, ISBN 0-750674-44-X

[3] Bradbury, Jeremy. "Linear Predictive Coding." Florida Institute of Technology. http://my.fit.edu/~vkepuska/ece5525/lpc_paper.pdf (2/22/11)

[4] Diemo Schwarz, Xavier Rodet - Spectral Envelope Estimation and Representation for Sound Analysis–Synthesis

[5]A.Robel,X.Rodet - Conference on Digital Audio Effects (DAFx'05), Madrid, Spain, September 20-22, 2005- Efficient spectral envelope estimation and its aplication to pitch shifting and envelope preservation