VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Kompiuterio valdymas balsu

Atliko: Mag. 2 kurso, 9 grupės studentas

Raimundas Karalevičius

Darbo vadovas:

doc. dr. Algirdas Bastys

Recenzentas:

dr. Pijus Kasparaitis

Vilnius
2007

Turinys

Anotacija

Šiame darbe gilinamasi į kompiuterio valdymo lietuvių kalba galimybes, apžvelgiant jau pasaulyje taikomus algoritmus bei pabrėžiant balso technologijų svarbą ateityje. Pagrindinis tikslas yra realizuoti kai kuriuos iš apžvelgtų atpažinimo algoritmų ir imituoti tam tikras valdymo balsu situacijas. Kaip kalbos atpažinimo pagrindas yra naudojamas dinaminės laiko skalės algoritmas (DTW) spektrinės poros (LSP) parametrams. Pagrindinės realizavimo priemonės – Java programavimo kalba, mikrofonas ir WAVE garsiniai failai. Pasiekti rezultatai leidžia teigti, kad įgyvendintas atpažinimo algoritmas gana tiksliai randa tariamus žodžius ir su nedideliu žodynu (iki 20 žodžių) efektyviai vykdo nurodytas komandas.

Summary

Control of Computer Using Voice

Nowadays speech processing should become very important because it is one of the best alternatives of present control means like keyboard and mouse. So this work studies:

- The use of speech signal processing.
- Various methods and algorithms of speech recognition.
- The implementation of Lithuanian speech recognition in this work.
- Comparison with other algorithms.
- Simulation of voice control.

There are described and analyzed these algorithms that are used in speech processing: Dynamic Time Warping [1], Linear Predictive Coding [2], Linear Spectral Pair (or Frequencies) ([2], [3], [8]), Vector Quantization [4], Adaptive Quantization [5] and Hidden Markov Models [9].

In the implementation of Lithuanian speech recognition were used Linear Predictive Coding, Linear Spectral Pair, Vector Quantization and Dynamic Time Warping algorithms, Java programming language, microphone and WAVE format. The speech recognition system were tested with more noisy and less noisy microphones, with various WAVE files recorded in different conditions and with real time speaking. Further for simulation of voice control were modeled three situations: calculator with four operations (sum, subtraction, multiplication and division), changing background colors and moving mouse cursor.

To conclude it should be noted that:

- Implemented speech recognition algorithm mostly gives around 90% recognition.
- The precision of testing depends on microphone and surrounding noise, length and acoustic likeness of recognizable words.
- Word finding and Dynamic Time Warping algorithms are optimized for most cases, but surely they can be improved for better results.

Įvadas

Dabartinio kompiuterio kol kas negalime įsivaizduoti be klaviatūros ir pelės, nes tai yra pagrindinės priemonės, kuriomis galime įvesti informaciją ir nurodymus. Bet tokie apribojimai neleistų automatizuotų technologijų efektyviai pritaikyti bendrai daugelyje gyvenimo sričių kaip namų ar kitos aplinkos valdymas (šviesos uždegimas, durų atidarymas ar uždarymas ir panašiai), gamybos įrenginių greitas valdymas, robotų kontrolė, galbūt ateityje transporto priemonių valdymas ir kita. Taigi kaip alternatyva būtų galimybė apdoroti kuo daugiau iš aplinkos sklindančios garsinės ir vaizdinės informacijos. Dabar jau įprasta pasienyje ar oro uoste (ypač JAV) patikrinti pirštų atspaudus, siekiama plačiau taikyti akies rainelės skanavimą, aktyviai tobulinamos automobilių numerių atpažinimo sistemos ir kitos vaizdo apdorojimo priemonės, o garsinės informacijos panaudojimas irgi ima įgauti vis didesnį pagreitį. Todėl šiame darbe atidžiau panagrinėsime kalbos atpažinimo galimybes ir problematiką.

Pirmiausiai net įprasto kompiuterio valdymą galime įsivaizduoti kaip tiesiog simbolių arba dar geriau žodžių sakymą. Tokiu atveju ir klaviatūros poreikis būtų tik antraeilis, nes, pavyzdžiui, teksto diktavimas tampa kur kas greitesnis ir nereikalauja papildomų pastangų įsimenant įvairių klavišų vietų. Analogiškai būtų galima pritaikyti žodines komandas, kurios mažiau suvaržytų judėjimo laisvę ir leistų valdyti kompiuterį per didesnį atstumą nei dabar. Visgi kalbos atpažinimas nėra toks paprastas, kaip gali pasirodyti iš pirmo žvilgsnio, nes įvairiose šalyse žmonės kalba skirtingomis kalbomis, be to, dar ir kiekvieno asmens kalbėjimo pobūdis gali skirtis gana ženkliai. Taip pat reikia atsižvelgti ir į tai, kad aplinkoje gali būti įvairaus pašalinio triukšmo, kuris slopina tariamus žodžius. Todėl kalbos atpažinimo sistema turi atsižvelgti į kiek tik įmanoma numatyti pašalinių faktorių.

Žvelgiant iš kitos pusės, gali kilti problemų atskiriant garsus ar žodžius vieną nuo kito, pavyzdžiui, paprasti lietuviški žodžiai „tu“ ir „du“ yra labai panašūs ir lengvai sumaišomi. O jeigu dar pridėsime triukšmą ir skirtingą tartį, tai kalbos atpažinimo kokybė labai priklausys nuo to, kaip sistema atsižvelgs į įvairią žodžius apibūdinančią informaciją ir atskirs reikalingus duomenis nuo aplinkinių trukdžių.

Kitas svarbus aspektas yra tai, kaip greitai atpažinimo sistema gali rasti žodžio reikšmę. Juk turint realų kalbos žodyną reikia taip susisteminti kiekvieno žodžio informaciją, kad galėtume greitai ir be užlaikymo atlikti palyginimus galbūt net su visais žodyno elementais. Tarkim atpažinimas nespėja su realiu laiku tariamais žodžiais, tai apie jo efektyvų pritaikymą negali būti jokių kalbų.

Taigi šiame darbe išdėstomi šie dalykai:

- Balso technologijų taikymo reikšmė.
- Įvairių kalbos atpažinimo algoritmų apžvalga.
- Detali darbe realizuoto kalbos atpažinimo algoritmo analizė.
- Darbe realizuoto algoritmo rezultatų palyginimas su kitų autorių algoritmais.
- Kompiuterio valdymo balsu imitavimas.

Kaip darbo rezultatas yra pateikiama programa, galinti realiu laiku atpažinti tariamus žodžius bei imituojanti kelias kompiuterio valdymo balsu situacijas.

1. Kalbos atpažinimo apžvalga

1.1. Balso technologijų taikymo reikšmė

Šiuo metu labiausiai yra išvystytos užsienio balso technologijos, ypač JAV ir Japonijos, todėl norint neatsilikti svarbu plėtoti ir lietuvių kalbos atpažinimo sistemas.

Iš pradžių reikia apibrėžti, kaip galima taikyti kalbos technologijas:

- Kalbos atpažinimas – kompiuteris pagal konkrečiai nurodytą žodyną atpažįsta tariamus žodžius.
- Kalbos sintezė – kompiuteris pagal turimus žodžių garsinius šablonus taria žodžius.

Šiame darbe gilinsimės į pirmąją taikymo sritį. Visgi antroji sritis irgi yra plačiai taikoma (*Microsoft Windows* turi programą "Narrator", taip pat atskiros programos kaip "Power Text to Speech Reader", "TextAloud", internetinė "Text-talk" ir daug kitų), nes tai gali būti naudinga ne tik regėjimo negalią turintiems žmonėms (teksto skaitymas), bet ir kuriant bendravimo su kompiuteriu sistemas. Juk turbūt daugeliui būtų maloniau gauti žodinį atsakymą ar klausimą negu ekrane išvedamą teksto eilutę.

Atsižvelgiant į sudėtingumą galime išskirti kalbos atpažinimą, kadangi jis apima žmogaus ištartų garsų interpretavimą ir įvairių algoritmų taikymą norint pasiekti kuo didesnį tikslumą. Pirmiausiai sistema turi būti apmokyta tam tikro žmogaus balsui su konkretaus dydžio žodynu, o jau tik tada galima automatiškai nustatyti, kokie žodžiai ar komandos yra tariamos sistemai.

Taigi balso technologijų taikymo spektras yra labai platus. Pabandykime jį sugrupuoti į tam tikras didžiausią poreikį turinčias sritis:

- Kompiuterio ar kitos automatinės sistemos valdymas (ši sritis gali būti susijusi ir su likusiomis).
- Automatinis vertimas iš vienos kalbos į kitą.
- Mokymo procesų pagalba.
- Įvairių paslaugų patogesnis pateikimas.
- Įvairių kitų procesų alternatyva (saugumas, medicina, kultūros ar istorinių įrašų apdorojimas ir kt.).

Pirmoji sritis labiau apima robotų technologijas ir komandų pateikimą kompiuteriui. Tai ypač naudinga tiek regėjimo negalią, tiek judėjimo sutrikimų turintiems neįgaliesiems, kurie neturi galimybių dirbti su dabartinėm valdymo priemonėm. Žinoma, efektyvus ir greitas kompiuterio valdymas balsu ateityje galėtų visiškai pakeisti klaviatūrą ir pelę visiems vartotojams. Na, o robotų technologijos be žodinio bendravimo tuo labiau neįsivaizduojamos.

Kiek kitoks balso technologijų taikymas yra automatinis kalbos vertimas, nes šiuo atveju reikia tiesiog kuo tiksliau išversti tam tikros kalbos žodžius į kitą kalbą taip, kad sakinio struktūra būtų rišli ir aiškiai suprantama. Tai labai artimai siejasi su lingvistika, o žinant, jog kiekvienos kalbos struktūra turi tik jai būdingų savybių, tokia užduotis tampa gana nelengvu iššūkiu.

Taip pat su lingvistika yra susijęs ir mokymo priemonių rengimas. Pavyzdžiui, mokantis gimtosios ar užsienio kalbos labai praverstų įvairūs automatiniai kalbėjimo treniruokliai, galintys atkreipti dėmesį į teisingą tarimą ir imituojantys realų pokalbį.

Kitas svarbus taikymas – įvairių paslaugų pateikimas – turi panašumų su automatinės sistemos valdymu, kadangi šiuo atveju sistema turi išklausyti kliento pageidavimus (nurodymus) ir, jei įmanoma, jį aptarnauti. Tai ypač aktualu tose paslaugų sferose, kur dažnai trūksta aptarnaujančio personalo arba kai reikia atlikti užsakymus per atstumą (tarkim internetinė sistema užfiksuoja, ką reikia atlikti, ir per tam tikrą ryšių sistemą praneša artimiausiam aptarnavimo skyriui).

Be jau minėtų balso technologijų taikymo sričių galima išskirti daug smulkesnių arba kol kas turinčių mažesnį poreikį. Kaip išskirtinis pavyzdys galėtų būti saugumo užtikrinimas atskiriant asmens tapatybę pagal balsą. Šis procesas jau pradedamas taikyti šalia kitų patikrinimų (pirštų ar delno atspaudų, akies rainelės, DNR ir kt.), o virtualioje erdvėje balso atpažinimas tikrai turėtų tapti nepakeičiama asmens parašo alternatyva. Visgi nereikia pamiršti ir kitų sferų kaip medicina, teisėsauga, krašto apsauga, kultūros vertybių tyrimai ir kt. Juk vien istorinių įrašų archyvuose yra sukaupta tiek informacijos, kad žmonės galėtų juos tyrinėti metų metus ir vis tiek nepastebėti galbūt svarbių detalių. Todėl automatizuotos sistemos galėtų būti kaip papildoma, o gal net esminė, pagalba stengiantis išsaugoti kultūros ir istorijos vertybes.

Kaip dabartinio pasaulio pasiekimų pavyzdį galima pateikti *Microsoft Windows Vista* [7] kalbos atpažinimo galimybes. Ši operacinė sistema gali būti valdoma anglų, japonų, kinų, prancūzų, ispanų ir vokiečių kalbomis. Tarp pagrindinių komandų yra įtraukta programų paleidimas ir uždarymas, failų atidarymas, kūrimas, kopijavimas ir trynimas, konkrečių užduočių vykdymas (programos langų pakeitimas, dokumento keitimas ir išsaugojimas, tam tikro internetinio puslapio atidarymas, konkretaus programos veiksmo vykdymas ir panašiai). Kaip atskira funkcija iš ankstesnių *Windows* versijų atėjo teksto diktavimas teksto redaktoriuose bei elektroninio pašto programose. Čia

kiekvienas simbolis (kablelis, taškas, dvitaškis ir kt.) turi žodinį atitikmenį kaip ir įprasti diktuojami žodžiai. Taip pat yra suteikiama galimybė pataisyti neteisingai atpažintą žodį, tačiau pabrėžiama, kad sistema automatiškai mokosi ir toliau vis tiksliau prisitaiko prie vartotojo tarties. Na, ir paskutinė svarbi sistemos funkcija yra pagalba („How do I“), kuri leidžia paklausti kaip galima atlikti konkrečius veiksmus.

Vadinasi, pasaulyje jau atsiranda gana patogių balso technologijomis veikiančių sistemų, bet kol kas lietuvių kalba lieka nuošalyje, todėl šiame darbe patyrinėsime mūsų gimtosios kalbos galimybes valdant automatines sistemas.

1.2. Kalbos atpažinimui naudojami algoritmai

Šiuo metu galima rasti be galo daug balso technologijose taikomų algoritmų. Kaip pagrindinius galima išskirti:

- Tiesinės prognozės kodavimas (angl. *Linear Predictive Coding* – LPC [2]).
- Spektrinės poros parametrai (angl. *Linear Spectral Pair* (arba *Frequencies*) – LSP [2], [3], [8]).
- Kvantavimo algoritmai (angl. *Vector Quantization* [4], *Adaptive Quantization* [5]).
- Dinaminės laiko skalės tempimo algoritmai (angl. *Dynamic Time Warping* – DTW [1]).
- Paslėpti Markovo modeliai (angl. *Hidden Markov Models* [9]).

Visi šie algoritmai darbo eigoje bus apžvelgti, o kai kurie realizuoti taip, kad kalbos atpažinimo rezultatų tikslumas būtų kuo didesnis.

1.3. Tiesinės prognozės kodavimas

Kadangi dirbant su garsu turime labai daug duomenų, reikia sumažinti atpažinimui skirtų duomenų kiekį. Garso signalų kompresijai yra naudojamas tiesinės prognozės kodavimas (LPC) papildytas spektrinės poros (angl. *Line Spectrum Pair* – LSP) algoritmu. Pirmiausiai garsinio failo bloko duomenims paskaičiuojami LPC parametrai, pagal kuriuos po to randami LSP parametrai,

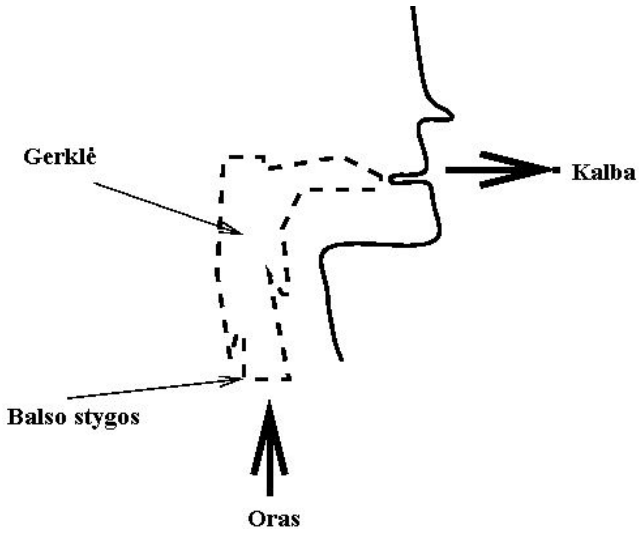
pasižymintys surūšiavimu didėjančia tvarka bei kitomis kalbos kompresijai ir atpažinimui naudingomis savybėmis.

1.3.1. Fizinis kalbos modelis

Iš pradžių reikia apibrėžti, kaip žmogaus balso garsai yra pateikiami skaitmeniniais signalais. Fizinis kalbos modelis yra apibūdinamas taip:

- Žmogus stumia orą iš plaučių pro gerklę, kur yra balso stygos, ir iš burnos pasigirsta kalbos garsai.
- Konkrečiam kalbos garsui mūsų balso stygos vibruoja skirtingai, t.y. atsidaro arba užsidaro skirtingu dažniu. Būtent šis balso stygų vibravimo dažnis apibrėžia garso aukštumą, pavyzdžiui, moterys ir vaikai paprastai kalba aukštu balsu (greitas vibravimo dažnis), o vyrai – žemu balsu (lėtas vibravimo dažnis).
- Tam tikriems dusliems ir skardiems garsam (priebalsiam) balso stygos nevibruoja, t.y. tam tikrą laiką išlieka atsidariusios.
- Taip pat garsui didelę įtaką turi gerklės forma, pavyzdžiui, kalbant ji nuolat kinta, kad pasigirstų skirtingi garsai.
- Visgi gerklės forma keičiasi gana lėtai (maždaug per 10 – 100 milisekundžių).
- Balso garsumą nulemia iškvepiamo oro kiekis, t. y. kuo daugiau oro išeina iš plaučių tuo garsiau galime kalbėti.

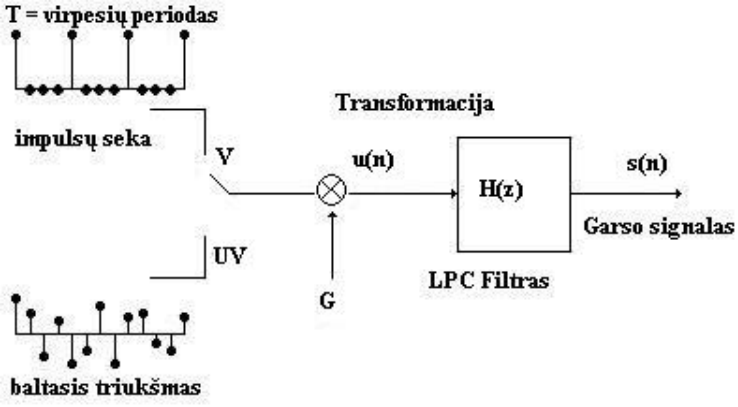
Šis fizinis kalbos modelis gali būti pavaizduotas taip (1 pav.):



1 pav. Žmogaus kalbėjimo modelis

1.3.2. Tiesinės prognozės parametrų radimas

Pereinant prie matematinės išraiškos [2] panagrinėkime tokią schemą (2 pav.):



2 pav. Kalbinių garsų ir triukšmo patekimas į garsinius duomenis

Kaip matome iš schemos LPC filtrui yra pateikiama garsinių impulsų seka (angl. *impulse train*) arba baltasis triukšmas (angl. *white noise*). Šiuo atveju garsinių impulsų seką *V* atitinka balso stygų vibravimas, kurio dažnio periodas yra pažymėtas *T*, baltąjį triukšmą – duslieji ir skardieji priebalsiai (žymėsime *UV*), kai balso stygos neturi įtakos, garsumą – dydis *G*, oro srautą patenkantį į gerklę – seka ***u(n)***, gerklės formą – LPC filtras ***H(z)*** ir galutinį garsinį signalą – seka ***s(n)***.

Dabar apibrėžkime patį LPC filtrą. Tarkim filtro eilė *P* yra lygi 10, tada ***H(z)*** galime apibrėžti taip:

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{10} z^{-10}}.$$

Iš čia galime gauti tiesinę pradinių duomenų ir rezultatų ryšio lygybę:

$$s(n) + \sum_{i=1}^{10} a_i s(n-i) = u(n).$$

Taigi LPC modelis gali būti apibūdinamas tokiais parametrais:

$$A = (a_1, a_2, \ldots, a_{10}, G, V/UV, T).$$

Žinoma, atliekant skaitmeninio signalo kompresiją naudosime tik $a_1, \ldots, a_{10}$ parametrus.

Mūsų atveju yra imami 20 milisekundžių (0,02 s) skaitmeninio garsinio signalo blokai, todėl per sekundę turime 50 tokių blokų, kurių kiekvieną šiuo atveju (kai *P=10*) apibūdina po 10 parametrų. Tuo tarpu imdami tiesiog visus garsinio signalo duomenis turėtume kur kas daugiau perteklinės informacijos. Tarkim, turėdami standartinį 8000 kadrų per sekundę dažnį, kiekvienam bloke

turėsime po 160 narių, taigi panaudojus LPC parametrus 16 kartų sumažėjo duomenų kiekis. Todėl galima sakyti, kad LPC parametrų seką ***A*** atitinka skaitmeninio garsinio signalo seka ***S***:

$$S = (s(0), s(1), ..., s(159)).$$

Taigi pagrindinis LPC kompresijos uždavinys yra sekai *S* rasti geriausius parametrus *A*. Pirmiausiai apibrėžkime papildomą parametrą ***f***:

$$f = \sum_{n=0}^{159} u^2(n).$$

Kadangi mes žinome, kaip iš rezultatų ***s(n)*** galima gauti pradinius duomenis ***u(n)***, tai čia nežinomi bus tik LPC parametrai $\boldsymbol{a_1}$, ... , $\boldsymbol{a_{10}}$. Kad gautume lygtis, imame *f* išvestines pagal parametrus a_i ir prilyginam jas nuliui:

$$df / da_1 = 0,$$
$$df / da_2 = 0,$$
$$.....$$
$$df / da_{10} = 0.$$

Dabar turime 10 lygčių su dešimt nežinomųjų, ką galima pavaizduoti matricomis:

$$\begin{pmatrix} R(0) & R(1) & ... & R(8) & R(9) \\ R(1) & R(0) & ... & R(7) & R(8) \\ ... & ... & ... & ... & ... \\ R(8) & R(7) & ... & R(0) & R(1) \\ R(9) & R(8) & ... & R(1) & R(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ ... \\ a_9 \\ a_{10} \end{pmatrix} = \begin{pmatrix} -R(1) \\ -R(2) \\ ... \\ -R(9) \\ -R(10) \end{pmatrix}$$

Čia matricos nariai ***R(k)*** vadinami autokoreliacine seka, gaunama iš ***s(n)***:

$$R(k) = \sum_{n=0}^{159-k} s(n)s(n+k).$$

Ši lygčių sistema gali būti išspręsta pritaikius Levinson – Durbin rekursijos algoritmą, kuris yra apibrėžiamas taip:

$$E^{(0)} = R(0),$$
$$k_i = [R(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R(i-j)] / E^{(i-1)}, \text{ kai } i = 1,2,...,10;$$
$$\alpha_i^{(i)} = k_i,$$
$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}, \text{ kai } j = 1,2,...,i-1;$$
$$E^{(i)} = (1 - k_i^2) E^{(i-1)}.$$

Taigi visa tai atlikus dešimt kartų randame LPC parametrus a_i:

$$a_i = -\alpha_i^{(10)}.$$

1.3.3. Spektrinės poros parametrų radimas

Radus tiesinės prognozės parametrus a_i, galime pereiti prie spektrinės poros algoritmo (toliau LSP) [3], kuris randa naujus parametrus, geriau apibūdinančius garsinę informaciją ir lengviau kvantuojamus. Pirmiausiai reikia apibrėžti funkcijas **A(z)**, ***P(z)*** ir ***Q(z)***:

$$A(z) \equiv A_p(z) = 1 + \sum_{p=1}^{P} a_p z^{-p},$$

$$P(z) \equiv A(z) + z^{-(p+1)} A(z^{-1}) = 1 + \sum_{p=1}^{P} (a_p + a_{p+1-p}) z^{-p} + z^{-(p+1)},$$

$$Q(z) \equiv A(z) - z^{-(p+1)} A(z^{-1}) = 1 + \sum_{p=1}^{P} (a_p - a_{p+1-p}) z^{-p} - z^{-(p+1)}.$$

Čia formulės yra užrašytos bendru atveju visoms eilėms *P*. Akivaizdu, kad ***P(z)*** yra simetrinis polinomas, o ***Q(z)*** – antisimetrinis, todėl galioja tokia lygybė:

$$A(z) = \frac{P(z) + Q(z)}{2}.$$

P(z) ir ***Q(z)*** turi tris svarbias savybes:

1. Kai ***P(z)*** = *0* arba ***Q(z)*** = *0*, z (toliau vadinsime šaknimis) priklauso vienetiniam apskritimui.
2. ***P(z)*** ir ***Q(z)*** šaknys yra persidengę tarpusavyje.
3. Po ***P(z)*** ir ***Q(z)*** šaknų kvantavimo įmanoma atvirkštinė transformacija, neprarandant duomenų.

Pirmos dvi savybės yra naudingos randant ***P(z)*** ir ***Q(z)*** šaknis, o trečioji užtikrina filtro stabilumą. Kadangi visos polinomų šaknys yra ant vienetinio apskritimo, tai $\boldsymbol{z = e^{jw}}$. Čia ***w*** yra LSP parametrai.

Kai eilė *P* yra lyginis ir didesnis už 2 skaičius, tai galioja dar dvi savybės:

4. Kai ***P(z)*** šaknis yra -1, tai ***Q(z)*** šaknis yra 1.
5. Be šios sprendinių poros dar egzistuoja kitos eilės *P/2* sprendinių poros abiem funkcijoms.

Pagal paskutiniąją savybę su naujais parametrais $\boldsymbol{w_k}$ ir $\boldsymbol{\theta_k}$ ($k=1, \ldots, P/2$) šios funkcijos užrašomos taip:

$$P(z) = (1+z^{-1})\prod_{i=1}^{P/2}(1-z^{-1}e^{j\omega_i})(1-z^{-1}e^{-j\omega_i}) = (1+z^{-1})\prod_{i=1}^{P/2}(1-2\cos(\omega_i z^{-1})+z^{-2}),$$

$$Q(z) = (1-z^{-1})\prod_{i=1}^{P/2}(1-z^{-1}e^{j\theta_i})(1-z^{-1}e^{-j\theta_i}) = (1-z^{-1})\prod_{i=1}^{P/2}(1-2\cos(\theta_i z^{-1})+z^{-2}).$$

Be to, parametrai w_k ir ir θ_k yra surūšiuoti didėjančia tvarka ir apriboti tokiu būdu:

$$0 < \omega_1 < \theta_1 < \omega_2 < \theta_2 < \ldots < \omega_{P/2} < \theta_{P/2} < \pi.$$

Būtent ši savybė padeda susieti vieną bloką su kitu, nes galime lyginti atitinkamo bloko atitinkamus parametrus. Tai panagrinėsime 2.6 skyriuje.

1.3.4. Čebyševo polinomai

Čebyševo polinomai (angl. *Chebyshev polynomials* [8]) padeda išvengti trigonometrinių funkcijų skaičiavimų arba trigonometrinių lentelių saugojimo kompiuterio atmintyje. LSP parametrai tiesiog suvedami į $x = \cos(w)$, t. y. :

$\cos(m\,w) = T_m(x)$, kur $T_m(x)$ yra m-tos eilės Čebyševo polinomas.

Tuo tarpu Čebyševo polinomai turi tenkinti tokią rekursiją:

$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, kai pradinės sąlygos $T_0(x) = 1$ ir $T_1(x) = x$.

Taip pat Čebyševo polinomai yra naudingi dar ir tuo, kad taip galima išvengti nežinomojo x laipsnių. Galutinę išraišką galima suvesti į tokią seką:

$Y(x) = c_0T_0(x) + c_1T_1(x) + \ldots + c_{N-1}T_{N-1}(x)$, kur c_i – koeficientai, o N – polinomo eilė.

Tokiu atveju įmanomas atvirkštinės rekursijos ryšys:

$b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + c_k$, kur $b_k(x)$ – kita Čebyševo polinomų išraiška.

Tokiu atveju pradinės sąlygos jau yra $b_N(x) = b_{N+1}(x) = 0$. Ši rekursinė išraiška reikalinga $b_0(x)$ ir $b_2(x)$ skaičiavimui, nes polinomo $Y(x)$ reikšmė būtent gaunama per šiuos polinomus. Pabandykime parodyti visą išvedimo eigą:

$$Y(x) = [b_0(x) - 2xb_1(x) + b_2(x)]T_0(x) + \ldots + [b_{N-1}(x) - 2xb_N(x) + b_{N+1}(x)]T_{N-1}(x) =$$
$$= b_0(x)T_0(x) + b_1(x)T_1(x) - 2xb_1(x)T_0(x) + (b_2(x)[T_2(x) - 2xT_1(x) + T_0(x)] + \ldots +$$
$$+ b_{N-1}(x)[T_{N-1}(x) - 2xT_{N-2}(x) + T_{N-3}(x)]) = b_0(x) - xb_1(x) = (b_0(x) - b_2(x) + c_0) / 2 .$$

Vadinasi, polinomo $Y(x)$ reikšmei rasti reikės dar ir koeficientų c_i .

Kad galėtume pereiti prie ***P(x)*** ir ***Q(x)*** polinomų išraiškų, reikia iš pradžių jas šiek tiek pritaikyti naujai situacijai. Kadangi ***P(x)*** ir ***Q(x)*** polinomų šaknys apima visus parametrus ***w***, tai kiekvieno polinomo eilę galime sumažinti iki ***P/2 = m***, tačiau tokiu atveju reikia perskaičiuoti ir koeficientus. Juos randame irgi rekursiškai:

$$p_i = a_{m-i} + a_{m+i+1} - p_{i+1}\text{ , kai } p_m = 1 \text{ ir } \mathbf{i = m - 1, \dots , 0;}$$

$$q_i = q_{i+1} + a_{m-i} - a_{m+i+1}\text{, kai } q_m = 1 \text{ ir } \mathbf{i = m - 1, \dots , 0.}$$

Taigi dabar galima rekursiškai paskaičiuoti ir polinomus ***P(x)*** ir ***Q(x)*** :

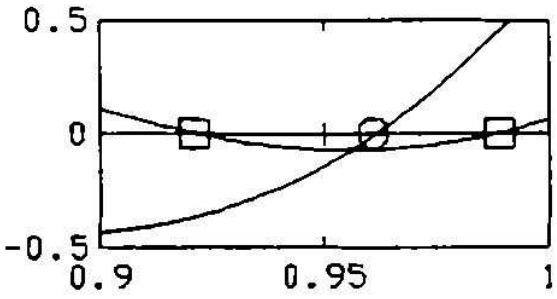
$$b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + p_k \text{ , kai } b_{m+1}(x) = b_{m+2}(x) = 0 \text{ ir } k = m, \dots , 0;$$

$$P(x) = (b_0(x) - b_2(x) + p_0) / 2;$$

$$d_k(x) = 2xd_{k+1}(x) - d_{k+2}(x) + q_k \text{ , kai } d_{m+1}(x) = d_{m+2}(x) = 0 \text{ ir } k = m, \dots , 0;$$

$$Q(x) = (d_0(x) - d_2(x) + q_0) / 2.$$

Mums reikalingos šaknys yra išsidėstę intervale ***-1 < x < 1***. Kraštinės šaknys 1 (nuo ***Q(x)***) ir -1 (nuo ***P(x)***) iš karto atmetamos, o šaknų paieška prasideda nuo ***x*** = 1 tikrinant polinomo ***P(x)*** ženklo pokyčius. Kad rastume kuo tikslesnę šaknį, reikia pasirinkti pakankamai mažą ***x*** perėjimo pokytį ***pokytis1***. Jeigu jis būtų per didelis, tai tam tikru momentu gali būti nepastebėtas polinomo reikšmės ženklo pasikeitimas (iš teigiamo į neigiamą arba atvirkščiai). Tai gerai iliustruoja 3 pav. :



3 pav. Dviejų polinomų *P(x)* ir *Q(x)* grafikai ir jų šaknys

Kadangi vieno polinomo šaknys gali būti gana arti viena kitos ir tuo labiau kito polinomo šaknies, tai dažnai tenka ***pokytis1*** parinkti atsižvelgus į ieškomų parametrų ***w*** skaičių ***P***, t. y. kuo didesnis jų skaičius, tuo didesnė tikimybė rasti juos vienas šalia kito.

Kai toks pasikeitimas yra užfiksuojamas, pagal antrąjį pokytį ***pokytis2*** bandome grįžti atgal ir dar labiau patikslinti ieškomą šaknį $x_š$. Po šių veiksmų pirmasis LSP parametras w_1 randamas taip: $w_1 = \arccos(x_š)$.

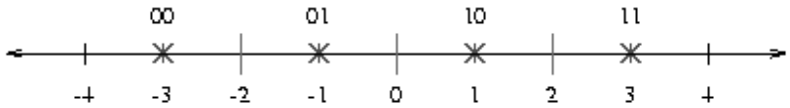
Tada tuos pačius veiksmus nuo $x_š$ atliekame su polinomu $Q(x)$ ir viską taip kartojame pakaitomis, kol randame visus parametrus w , kurių yra P.

1.4. Vektoriaus kvantavimo algoritmai

Jeigu turime daug garsinės komandos blokų, tai jų skaičių galime sumažinti atlikdami vektoriaus kvantavimą. Mūsų atveju vektorių atitinka vieną bloką apibūdinantys LSP parametrai, kurių skaičius nepakis ir po kvantavimo, tačiau rezultate turėsime žymiai mažesnį vektorių skaičių, o tai labai naudinga norint surinkti kuo daugiau informacijos apie vieną komandos šabloną, pavyzdžiui, daug kartų pakartojant žodį ar žodžių junginį.

1.4.1. Paprastas vektoriaus kvantavimo algoritmas

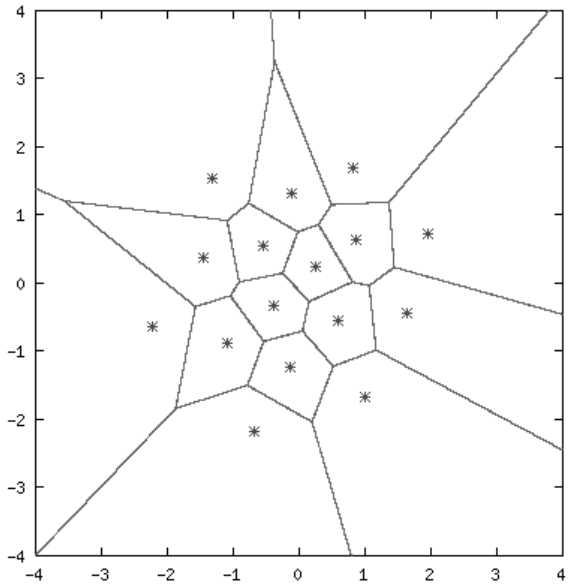
[4]Paprastai tariant, vektoriaus kvantavimas yra tam tikras vektoriaus reikšmių apvalinimas. Pavyzdžiui, vienmačiu atveju turime realių skaičių intervalą nuo -4 iki 4, tačiau mes norime, kad šį intervalą atitiktų keturios reikšmės. Tada skaičiams nuo -4 iki -2 suteikiam reikšmę -3, skaičiams nuo -2 iki 0 – -1, skaičiams nuo 0 iki 2 – 1 ir skaičiams nuo 2 iki 4 – 3. Grafiškai tai galima pavaizduoti 4 pav.:



4 pav. Vienmačio kvantavimo pavyzdys

Beje, suapvalintas reikšmes galime išreikšti per du bitus, todėl šį pavyzdį galima vadinti vienmačiu 2 bitų vektoriaus kvantavimu (angl. *Vector Quantization* – VQ). Taip pat jo dažnis yra 2 bitai/dimensija.

Dvimatis pavyzdžio (5 pav.) atveju, jeigu taškas patenka į kurią nors sritį, tai jis apvalinamas iki tą sritį apibūdinančios žvaigždutės, kurių kaip ir sričių turime 16. Taigi turime dvimatį 4 bitų VQ, kurio dažnis 2 bitai/dimensija.



5 pav. Dvimačio kvantavimo pavyzdys

Dabar pereisim prie terminų. Pavyzdžiuose žvaigždutėmis žymėti vektoriai vadinami *kodo vektoriais* (angl. *codevectors*), o linijomis pažymėtos sritys (vienmačiu atveju atkarpos, dvimačiu daugiakampiai) – *kodavimo sritimis* (angl. *encoding regions*). Visų kodo vektorių aibė vadinama *kodo žodynu*, o kodavimo sričių aibė – *erdvės padalinimas*.

Tarkim yra duota vektorių aibė, iškraipymo matas, kodo vektorių skaičius, tada reikia rasti kodo žodyną bei erdvės padalinimą, kurio vidutinis iškraipymas yra mažiausias.

Laikykime, kad turime M duomenų vektorių:

$$\tau = \{ x_1 , x_2 , \dots , x_M \}.$$

Be to, M turi būti pakankamai didelis, kad turėtume kuo daugiau statistinių duomenų apie nagrinėjamą objektą, pavyzdžiui, ištartų žodžių seką. Taip pat tarkime, kad duomenų vektorių dimensija yra k:

$$x_m = (x_{m,1} , x_{m,2} , \dots , x_{m,k}) , \text{ kai } m = 1, 2, \dots , M.$$

Jeigu reikia N kodo vektorių, tai kodo žodynas bus toks:

$$C = \{ c_1 , c_2 , \dots , c_N \}.$$

Kodo vektorių dimensija yra lygi k (taigi kaip ir duomenų vektorių):

$$c_n = (c_{n,1} , c_{n,2} , \dots , c_{n,k}) , \text{ kai } n = 1, 2, \dots , N.$$

Tada sritys S_n , susietos su kodo vektoriais c_n , sudarys erdvės padalinimą:

$$P = \{ S_1 , S_2 , \dots , S_N \}.$$

Jeigu pradinių duomenų vektorius x_m patenka į sritį S_n , tada jo artinys bus kodo vektorius c_n. Šį apvalinimą galime pažymėti tokia funkcija:

$$Q(x_m) = c_n \text{ , jei } x_m \in S_n .$$

Vidutinis iškraipymas apskaičiuojamas pagal tokią formulę:

$$D_{vid} = \frac{1}{Mk}\sum_{m=1}^{M}\left\|x_m - Q(x_m)\right\|^2.$$

Šiuo atveju vektoriaus normos apibrėžimas yra toks:

$$\|e\|^2 = e_1^2 + e_2^2 + \ldots + e_k^2.$$

Taigi pagrindinė problema yra minimizuoti vidutinio iškraipymo reikšmę.

Taip pat rezultatai turi tenki dvi sąlygas: *artimiausio kaimyno* ir *centroido*.

Artimiausio kaimyno sąlygą galima apibrėžti taip:

$$S_n = \{ x : \|x - c_n\|^2 \le \|x - c_{n'}\|^2 \text{ visiem } n' = 1, 2, \ldots, N \}.$$

Taigi kodavimo sritis S_n turi apimti visus pradinius vektorius, kurie yra arčiau kodo vektoriaus c_n nei nuo kitų kodo vektorių. Išimtiniu atveju, kai vektorius yra ant kodavimo sričių ribos galima apibrėžti tam tikras pasirinkimo galimybes.

Centroido sąlyga teigia, kad kodo vektorius c_n turi būti vidurkis visų pradinių vektorių, patenkančių į sritį S_n :

$$c_n = \frac{\sum_{x_m \in S_n} x_m}{\sum_{x_m \in S_n} 1}, \text{ kai } n = 1,2,\ldots,N.$$

Taigi pagal šią sąlygą į kodavimo sritį turi patekti bent vienas vektorius.

Dabar galime apibrėžti iteracinio LBG (Linde, Buzo, Gray) vektoriaus kvantavimo algoritmo žingsnius. Pirmiausiai pradiniam vektorių žodynui $C^{(0)}$ randame pradinį kodo vektorių, kuris yra visų duomenų vektorių vidurkis. Pats algoritmas yra pagrįstas šio kodo vektoriaus iteraciniu skaidymu tol, kol apskaičiuojame reikiamą skaičių kodo vektorių:

1. Duota pradinių vektorių aibė τ iš M vektorių ir fiksuotas mažas skaičius $\varepsilon > 0$.
2. Tegu $N = 1$ ir kodo vektorius lygus:

$$c_1^* = \frac{1}{M}\sum_{m=1}^{M} x_m.$$

Paskaičiuojam vidutinį iškraipymą:

$$D_{vid}^* = \frac{1}{Mk}\sum_{m=1}^{M}\left\|x_m - c_1^*\right\|^2.$$

3. Atliekamas visų ***N*** kodo vektorių skaidymas. Tarkim ***i*** = 1, ... , ***N*** , tada:

$$c_i^{(0)} = (1+\varepsilon)c_i^*,$$
$$c_{N+i}^{(0)} = (1-\varepsilon)c_i^*.$$

Taip pat pakeičiam ***N*** = 2***N*** .

4. Atliekami iteracijos veiksmai. Tegu pradinis iškraipymas yra $D^{(0)}_{vid} = D^*_{vid}$ ir iteracijos skaitliukas ***i*** = 0. Tada atliekami tokie veiksmai:

a) Visiems ***m*** = 1, ... , ***M*** randame minimalią reikšmę:

$$\left\| x_m - c_n^{(i)} \right\|^2.$$

Čia ***n*** = 1, ... , ***N***. Tarkime su indeksu n^* yra ieškomas minimumas, tada pažymime:

$$Q(x_m) = c_{n^*}^{(i)}.$$

b) Visiems ***n*** = 1, ... , ***N*** randame naujus kodo vektorius:

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_n^{(i)}} x_m}{\sum_{Q(x_m)=c_n^{(i)}} 1}.$$

c) Nustatome ***i*** = ***i*** +1.

d) Paskaičiuojame naują vidutinį iškraipymą:

$$D_{vid}^{(i)} = \frac{1}{Mk} \sum_{m=1}^{M} \left\| x_m - Q(x_m) \right\|^2.$$

e) Tikriname sąlygą:

$$(D_{vid}^{(i-1)} - D_{vid}^{(i)}) / D_{vid}^{(i-1)} > \varepsilon.$$

Jeigu taip, tai grįžtame į žingsnį a).

f) Išsisaugome $D^*_{vid} = D^{(i)}_{vid}$ ir visiems ***n*** = 1, ... , ***N*** :

$$c_n^* = c_n^{(i)}.$$

5. Kartojame 3 ir 4 žingsnius tol, kol pasiekiame reikiamą skaičių kodo vektorių.

Patartina, kad duomenų vektorių skaičius ***M*** būtų žymiai didesnis už planuojamą kodo vektorių skaičių ***N***, pavyzdžiui, ***M*** > 1000***N***.

1.4.2. Adaptyvus diapazono kvantavimas

Adaptyvaus diapazono kvantavimo metodas (angl. *Adaptive Quantization Range Method* [5]) yra taikomas konkrečiai LSP parametrams apvalinti ir atsižvelgia į jų surūšiavimo savybes. Kiekvieno LSP parametro kvantavimo diapazonas yra nustatomas pagal prieš tai kvantuotų parametrų reikšmes, t.y. kvantuojam LSP vektorių koordinates paeiliui ir panaudojam prieš tai surinktą informaciją tam, kad pašalintume galimas paklaidas bei įvairius triukšmo sukeltus faktorius.

Tegu LSP parametrų ribos yra :

$$0 = \omega_0 < \omega_1 < \ldots < \omega_p < \omega_{p+1} = 0.5.$$

Čia ***p*** yra LSP parametrų eilė. Šiuo atveju imkime ***p*** lygų 10. Tarkim turime ***n*** tokių LSP parametrų vektorių, kurių atitinkami elementai yra išsidėstę netoli tam tikro vidurkio, paskaičiuojamo taip:

$$B_i = \frac{0.5i}{p+1}.$$

Tada apskaičiuojami nauji suvienodinti parametrai:

$$\tilde{\omega}_i(n) = \omega_i(n) - B_i.$$

Kvantatoriui yra pateikiama spėjama paklaida:

$$e_i(n) = \tilde{\omega}_i(n) - 0.90625\,\hat{\omega}_i(n-1).$$

Kvantatoriumi gauname naują paklaidą:

$$\hat{e}_i(n) = Q_{\omega i}^{-1}\left[Q_{\omega i}\left[e_i(n)\right]\right].$$

Po to paskaičiuojamas naujas LSP parametras:

$$\hat{\omega}_i(n) = \hat{\omega}_i^{\,p}(n) + \hat{e}_i(n).$$

Čia ***i***-tajam LSP parametrui pritaikome tiesinį kvantatorių $\boldsymbol{Q_{wi}}$:

$$Q_{\omega i}(x) = \max\left[0, \min\left(2^N - 1, Q_{ti}(x)\right)\right].$$

Šioje vietoje $\boldsymbol{Q_{ti}}$ apskaičiuojamas:

$$Q_{ti}(x) = round\left(\frac{(2^N - 1)(x + e_{i\max}}{2e_{i\max}}\right).$$

Taip pat reikia paminėti, kad ***N*** yra kvantuojamų bitų skaičius, ***round(x)*** – apvalinimo funkcija iki artimiausio sveikojo skaičiaus, o maksimalus kvantavimo diapazonas $e_{i\ max}$ yra paimtas iš 1 lentelės:

1 lentelė. Maksimalus kvantavimo diapazonas pagal atitinkamus parametrus

LSP parametras	Maksimalus diapazonas (e_{imax})
ω_1	0.03
ω_2	0.04
ω_3	0.07
ω_4	0.07
ω_5	0.06
ω_6	0.06
ω_7	0.05
ω_8	0.05
ω_9	0.04
ω_{10}	0.04

Maksimalus kvantavimo rėžis palaipsniui gali kisti, pavyzdžiui, iš pradžių imame dešimtą LSP parametrą, kuriam taikysime reikšmę iš lentelės, tačiau kitiems parametrams galėsime taikyti ir mažesnes ribas, nes atsižvelgsime į naujai paskaičiuotus parametrus. Tikrinimo koeficientas gaunamas iš kvantuoto ***(i+1)***-osios eilės parametro atėmus senąją ***i***-tosios eilės parametro reikšmę:

$$z = \omega_{i+1}^{q}(n) - \omega_{i}^{p}(n).$$

Jeigu **z** yra mažesnis už $e_{i\ max}$, tada kvantatoriaus formulėje naudosime **z** reikšmę, t. y. sumažinsime kvantavimo diapazoną iki [$-e_{i\ max}$, **z**].

Taigi šį algoritmą sudaro tokie žingsniai:

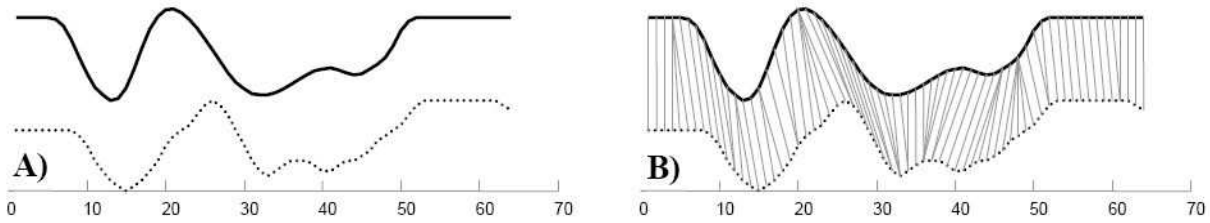
1. Kvantuojame parametrus $w_{10}(n)$ pagal kvantavimo diapazoną [$-e_{10\ max}$, $e_{10\ max}$] ir nustatom ***i*** lygų 9.
2. Randame tikrinimo koeficientą **z**.
3. Jei |**z**| < $e_{i\ max}$ tai parametrus $w_i(n)$ kvantuojame diapazone [$-e_{i\ max}$, **z**], kitu atveju imame diapazoną [$-e_{i\ max}$, $e_{i\ max}$].
4. Jei ***i*** = 1, stabdome veiksmus, kitu atveju ***i*** = ***i*** – 1 ir grįžtame į 2 žingsnį.

Taip apskaičiuoti nauji suapvalinti LSP parametrai turi gana geras savybes ir jiems, jei reikia, galima pritaikyti įprastą arba pakoreguotą vektoriaus kvantavimą.

1.5. Dinaminės laiko skalės algoritmas

Šio algoritmo užduotis yra palyginti dvi nuo laiko priklausančias duomenų sekas. Mūsų nagrinėjamu atveju tai yra garso duomenys, tačiau dinaminės laiko skalės (DTW [1]) technologiją

galima pritaikyti medicinoje, pramonėje, įvairiuose moksliniuose stebėjimuose ir panašiai. Dviejų sekų palyginimą galima pavaizduoti 6 pav.:



6 pav. Panašių kreivių susiejimas DTW algoritmu

Akivaizdu, kad viršutinė ir apatinė sekos turi panašią formą, tačiau kai kuriais laiko momentais viena seka pralenkia ar atsilieka nuo kitos sekos. Būtent DTW algoritmas turi surasti tuos momentus ir išlygiuoti panašiausius sekų atitikmenis. Paveikslėlyje **B)** matome, kad tokios nesutapimų dalys sueina į vieną tašką arba atvirkščiai labiau išsiskaido.

1.5.1. Klasikinis dinaminės laiko skalės algoritmas

Dabar apibrėžkime patį DTW algoritmą [1]. Tarkim turime dvi sekas Q ir C atitinkamai n ir m ilgio:

$$Q = q_1 , q_2 , \dots , q_i , \dots , q_n ;$$

$$C = c_1 , c_2 , \dots , c_j , \dots , c_m .$$

Šių sekų palyginimui reikia sukonstruoti n ant m dydžio matricą, kurios elementai (i, j) laiko atstumą tarp dviejų sekos taškų $d(q_i , c_j)$. Paprastai šis atstumas paskaičiuojamas pagal Euklido atstumo formulę:

$$d(q_i , c_j) = (q_i - c_j)^2.$$

Kiekvienas matricos elementas atitinka sekų narių q_i ir c_j palyginimą, todėl galime rasti tokį atitikimo kelią W, kai šie atstumai yra mažiausi:

$$W = w_1 , w_2, \dots , w_k , \dots , w_K ,$$ kai $$max(m, n) \le K < m+n - 1.$$

Šiuo atveju $w_k = (i, j)_k$. Be to, atitikimo kelias W yra apribotas tokiomis savybėmis:

1. Ribų sąlygos: $w_1 = (1, 1)$ ir $w_K = (n, m)$. Taigi kelias prasideda ir baigiasi matricos įstrižainės kampuose.
2. Tęstinumas: $w_k = (a, b), w_{k-1} = (a', b')$, kur $a - a' \le 1$ ir $b - b' \le 1$. Taigi galime imti tik gretimas matricos reikšmes, bet nebūtinai vien iš matricos įstrižainės.

3. Monotoniškumas: $w_k = (a, b)$, $w_{k-1} = (a', b')$, kur $a - a' \geq 0$ ir $b - b' \geq 0$. Taigi negalima grįžti atgal.

Pagal šias savybes egzistuoja labai daug atitikimo kelių, tačiau mums tinka tik labiausiai minimizuojantis. Dėl to yra įvedama minimalios atitikimo paklaidos savoka, kuri yra apskaičiuojama taip:

$$DTW(Q,C) = \min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right/K.$$

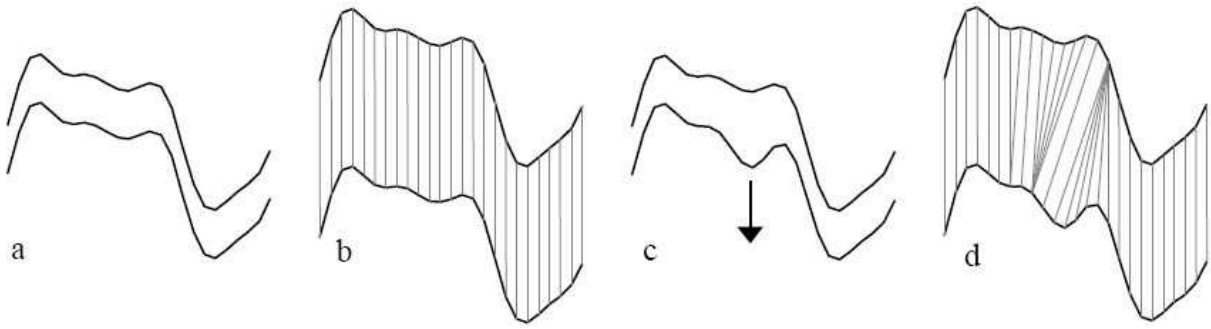
Šis minimizuotas kelias gali būti rastas pritaikius dinaminį programavimą, t. y rekursyviai ieškant suminio atstumo $\gamma(i, j)$ kaip ir $d(i, j)$. Todėl skaičiuodami suminį atstumą tikriname prieš tai buvusius gretimus matricos narius:

$$\gamma(i,j) = d(q_i,c_j) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\}.$$

Algoritmo minimali atitikimo paklaida būna lygi nuo 0 iki 1, t. y. kuo arčiau nulio, tuo geresnis atitikimas tarp sekų.

1.5.2. Išvestinės laiko skalės algoritmas

DTW algoritmas paprastai būna sėkmingas, kai lyginam panašias sekas su vietiniais neatitikimais laiko skalėje, tačiau kai labiau skiriasi ir reikšmės, tada atsiranda rimtų problemų. Pavyzdžiui, jeigu vienos sekos minimumas yra kur kas mažesnis už kitos sekos, tai atitikimas gali būti ir nerastas (7 pav.):



7 pav. a) ir b) dalyse tos pačios kreivės lyginimas ją paslinkus, o c) ir d) dalyse apatinė kreivė turi gilesnį įdubimą

Kaip matome, yra paimtos visiškai vienodos sekos ir **c** paveikslėlyje pagilintas vienas įdubimas. Na, o **d** paveikslėlyje jau pastebime, kad tas gilesnis įdubimas susiejamas net su pakilimu, nors to

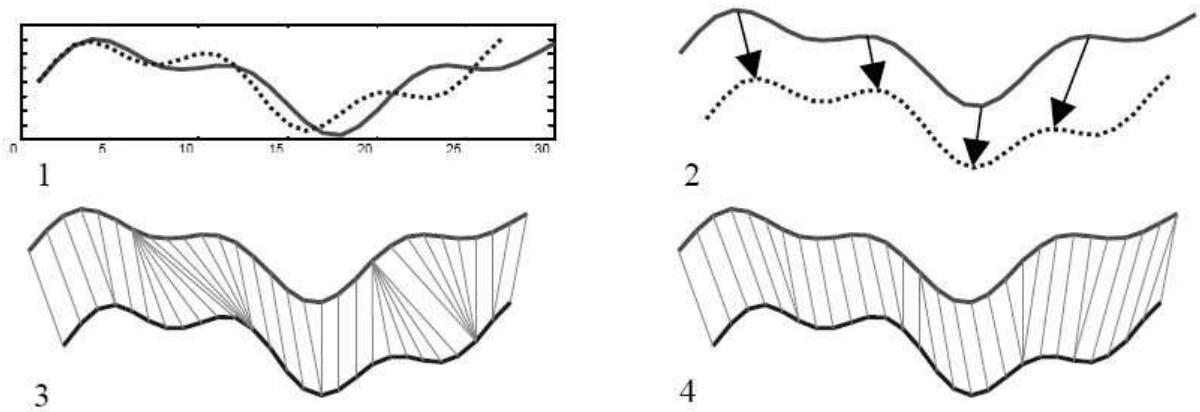
neturėtų būti. Taip atsitiko todėl, kad vienos sekos pakilimo taškas sutapo su kitos sekos nusileidimo tašku, ir taip gavome problematišką atitikimą.

Vienas iš šios problemos sprendimų gali būti išvestinės laiko skalės algoritmas (*Derivative Dynamic Time Warping* – DDTW [1]), kuris ima ne tiesiogines sekų reikšmes, o jų išvestines. Dėl paprastumo išvestinė skaičiuojama pagal tokią formulę:

$$D_x[q] = ((q_i - q_{i-1}) + ((q_{i+1} - q_{i-1}) / 2)) / 2 \text{ , kai } 1 < i < m .$$

Taip paskaičiuojamas pasvirimo vidurkis, kuris po to imamas taikant visus DTW algoritmo žingsnius, t. y. pagal šias reikšmes ieškome atstumo tarp sekų, sudarome atstumų matricą ir pagaliau ieškome minimalaus atitikimo kelio. Kadangi yra imamos pasvirimo vidurkio reikšmės, dėl to turėtų būti išspręsta neteisingo atitikimo problema, kai pakilimas susiejamas su nusileidimu. Visgi turbūt abu algoritmai turi savų pliusų ir minusų, todėl vienu atveju gali būti, kad geriau veiks viena algoritmo modifikacija, antru – dar labiau pakeistas variantas, o trečiu – jau klasikinis algoritmas. Todėl realizuojant balso atpažinimą tenka įvesti įvairių pakeitimų skirtingoms problemoms išspręsti arba bent minimizuoti.

Dabar pabandykime palyginti klasikinį ir išvestinį DTW algoritmus konkrečiu atveju(8 pav.):



8 pav. DTW (3 dalis) ir DDTW (4 dalis) algoritmų palyginimas

Trečiame paveikslėlyje yra parodyti DTW, ketvirtame – DDTW rezultatai. Akivaizdu, kad DDTW kur kas geriau susiejo panašias sekas nei DTW.

1.6. Paslėpti Markovo modeliai (HMM)

Alternatyva dinaminės laiko skalės tempimo algoritmui yra paslėpti Markovo modeliai (toliau HMM – angl. *Hidden Markov Models* [9]). Jie naudojami ne tik balso technologijose, bet ir rašto,

gestų bei kituose atpažinimuose. Trumpai kalbant HMM – tai tam tikras baigtinio automato variantas, tačiau kitaip nei baigtiniuose automatuose HMM procesai nėra deterministiniai, t.y. čia išnaudojami atsitiktinumo statistiniai duomenys.

Taigi HMM – tai baigtinių būsenų aibė, o kiekviena būsena yra susieta dažniausiai su daugiamačiu tikimybiniu skirstiniu. Perėjimai tarp būsenų yra valdomi tikimybių aibe – *perėjimų tikimybėmis*. Kiekvienoje būsenoje atsižvelgiant į turimą tikimybinį skirstinį gali būti sugeneruotas rezultatas. Tuo tarpu pati būsena išorėje nėra matoma – iš čia ir kilo pavadinimas paslėpti Markovo modeliai.

Norint apibrėžti HMM, reikia tokių parametrų:

- N – modelio būsenų skaičius.
- M – skirtingų stebėjimo būsenų (simbolių) skaičius (abėcėlės dydis). Jeigu stebėjimas yra besitęsiantis (t.y. nežinome, kiek yra simbolių) M yra begalybė.
- Būsenų perėjimų tikimybių aibė $\Lambda = \{ a_{ij} \}$, kai

  $$a_{ij} = p\{ q_{t+1} = j \mid q_t = i \}$$, kur $1 \le i, j \le N$, o q_t – dabartinė būsena.

  Perėjimų tikimybės turi tenkinti tokias atsitiktinumo sąlygas:

  a. $a_{ij} \ge 0$, kai $1 \le i, j \le N$.

  b. $a_{i1} + a_{i2} + \ldots + a_{iN} = 1$, kai $1 \le i \le N$.

- Kiekvienos būsenos tikimybių skirstinys $B = \{ b_j(k) \}$, kai

  $$b_j(k) = p\{ o_t = v_k \mid q_t = j \}$$, kur $1 \le j \le N$, $1 \le k \le M$.

  Čia v_k – k– tasis stebėjimų abėcėlės simbolis, o o_t – dabartinis parametrų vektorius.

  Tikimybių skirstinys turi tenkinti šias tikimybines sąlygas:

  a. $b_j(k) \ge 0$, kai $1 \le j \le N$, $1 \le k \le M$.

  b. $b_j(1) + \ldots + b_j(M) = 1$, kai $1 \le j \le N$.

- Pradinės būsenos skirstinys $\pi = \{ \pi_i \}$, kur

  $$\pi_i = p\{ q_1 = i \}$$, kai $1 \le i \le N$.

Taip pat galima naudoti sutrumpintą HMM žymėjimą $\lambda = (\Lambda, B, \pi)$.

Bendrai kalbant galima išskirti tris pagrindines paslėptų Markovo modelių sprendžiamas problemas:

1. Kokybės įvertinimas, t.y. kai turime HMM λ ir stebėjimų rezultatų aibę $O = o_1 , o_2 , \ldots , o_T$, kokia tikimybė, kad šie stebėjimai yra sugeneruoti modeliu $p\{ O \mid \lambda \}$? Šios problemos sėkmingas sprendimas padeda atpažinti atskirus žodžius, žodžių junginius ar kitokius garsų darinius.

2. Dekodavimas, t.y. kokia labiausiai tikėtina būsenų seka galėjo duoti tokius stebėjimo rezultatus? Tai susiję tiek su nuolatiniu stebėjimų atpažinimu, tiek su tam tikrų segmentų palyginimu.

3. Apmokymas, t.y. kaip suderinti modelio parametrus $\{ \Lambda, B, \pi \}$, kad galėtume maksimizuoti $p\{ O \mid \lambda \}$? Žinant tai, kad visada labai svarbus atpažinimo tikslumas, apmokymo optimizavimas tampa išskirtiniu uždaviniu.

Visgi galima teigti, kad HMM algoritmas dabar labai populiarėja ir tampa vienu iš pagrindinių balso technologijų vystymosi varikliu. Ypač pabrėžiama šių modelių nuolatinio mokymosi galimybė, tačiau dėl HMM matematinių operacijų ir parametrų gausumo, toks algoritmas palyginus su kitais gali būti gana lėtas. Be to, HMM apmokymas yra reiklus ne tik duomenų dydžiu, bet ir tuo, kad būtina pateikti tik teigiamą informaciją, t.y. maksimizuojamos stebėjimų patekimo į tam tikrą klasę tikimybės, bet neminimizuojamos patekimo į kitas klases tikimybės. Vadinasi, prieš pasirenkant tam tikrą kalbos atpažinimo algoritmą tenka atsižvelgti ir į technines galimybes, ir į tai, kaip galima optimizuoti visų procesų veikimą.

2. Praktiniai taikymai

Šiame darbe taikymams buvo pasirinkti dinaminės laiko skalės tempimo (DTW) , tiesinės prognozės (LPC), spektrinės poros (LSP) ir kvantavimo algoritmai.

2.1. Kalbos atpažinimo algoritmo žingsniai

Kad programa galėtų atpažinti nuskaitomus žodžius, iš pradžių turi būti atliktas apmokymas su tam tikru žodynu, todėl kalbos atpažinimo algoritmą galima suskaidyti į du stambius etapus:

I. Garsinių šablonų sukūrimas ir kalbos atpažinimo programos apmokymas. Šiame etape turi būti surinkta visa reikiama informacija apie visus nurodytus žodžius ar žodžių junginius.

II. Kalbos atpažinimas. Čia surenkama informacija apie pateiktą žodį ar žodžių junginį ir palyginama su visais šablonais, tarp kurių randamas mažiausiai besiskiriantis nuo testuojamos komandos.

Visgi šie du etapai bent iš pradžių turi nemažai panašumų, kurie apima informacijos apie garsinį failą (ar mikrofono duomenis) rinkimą:

1. Garsinio failo sukūrimas (šablonams) ir nuskaitymas (failo arba mikrofono duomenų).
2. Nuskaitytų duomenų suskaidymas į 0,02 sekundžių blokelius.
3. Kiekvieno blokelio tiesinės prognozės parametrų (LPC) radimas.
4. Spektrinės poros parametrų (LSP) radimas pagal LPC parametrus.
5. LSP parametrų kvantavimas.

Taigi kuriant šablonus garsinius duomenis sudaro pasikartojančių žodžių seka, ir šie penki žingsniai kartojami visiems šablonams atskirai. O kalbos atpažinimo etape viskas atliekama tik nuskaitytai komandai. Be to, čia dar pridedamas surinktos informacijos lyginimo žingsnis:

- Rastų parametrų lyginimas su šablonų parametrais naudojant Dinaminės laiko skalės tempimo (DTW) algoritmą.

2.2. Komandų šablonų ir kitų garsinių failų sukūrimas

Iš pradžių reikia įrašyti standartines kompiuterio valdymo operacijas į WAVE failus, kuriuos laikysime kiekvienos komandos šablonais. Taip pat įrašomi ir bandymams skirti garsiniai failai. Galimi keli šablonų sukūrimo būdai:

- Garsiniuose failuose laikyti po vieną žodį ar, jei komandą sudaro keli žodžiai, kelis žodžius.
- Šablono garsinį failą sudarys kelis kartus pakartoti tie patys žodžiai ar žodžių junginiai. Šiuo atveju bus galima surinkti daugiau tą žodį ar žodžių junginį apibūdinančios informacijos.

Beje, iš šablono failo pavadinimo sužinome, kokia tai komanda, ir atpažįstant žodžius (realiu laiku arba iš failų) galime naudoti jau turimą vardą tolimesniems veiksmams.

2.3. Garsinio failo nuskaitymas

Kadangi WAVE failų formatai gali būti skirtingi, reikia paminėti jų savybes. Taigi imame PCM formato WAVE failus, t. y. be jokios duomenų kompresijos.

Pats mažiausias garsinių duomenų kiekis vadinamas kadru (angl. *Frame*), kurį gali sudaryti vienas, du arba keturi baitai. Taip pat kadrą gali sudaryti vienas arba du garsiniai kanalai, t. y. mono arba stereo. Kitas garsinį failą apibūdinantis parametras yra kadrų skaičius per vieną sekundę (angl. *Frame Rate*), kurį žymėsime FPS. Na, ir paskutinis parametras yra garsinio failo dydis.

Taigi nuskaitant garsinį failą reikia atkreipti dėmesį, kiek baitų sudaro vieną kadrą, ir atitinkamai suformuoti vieną garsinių duomenų vienetą, pavyzdžiui, vieno baito (t. y. 8 bitų) mono garsinis kadras gali būti nuskaitytas taip:

- Baitą nuskaitome kaip mažą skaičių (nuo -128 iki 127).
- Skaičių transformuojam į intervalą 0 - 256: *kadras = (short)(skaičius & 0xff)*, t. y. atliekame bitinį „ir“ su vienetukais ir imame kaip *short* tipo (2 baitai) skaičių.

Čia reikia atkreipti dėmesį, kad 8 bitų formate (tiek mono, tiek stereo) reikia imti tik teigiamus skaičius nuo 0 iki 255. Tuo tarpu 16 bitų (dviejų baitų vienam kanalui) formate jau turime sveikus skaičius nuo -32768 iki 32767. Todėl 16 bitų mono garsinis kadras nuskaitomas taip:

- Abu baitus nuskaitome kaip mažus skaičius *skaičius1* ir *skaičius2* (nuo -128 iki 127).

- Tada kadro reikšmė gaunama taip:

kadras = (short)((skaičius1 & 0xff) | ((skaičius2 & 0xff)) <<8)).

Čia ženklas „|“ reiškia bitinį „arba“, o „<<“ – bitų postūmį į kairę.

Visus duomenis nuskaitome į skaičių (nuo 0 iki 255 arba nuo -32768 iki 32767) seką. Jeigu turime stereo garsinį failą, tai reikės imti arba abiejų kanalų reikšmių vidurkį, arba naudoti vieną iš dviejų kanalų, arba sudaryti dvi atskiras sekas abiem kanalams. Be to, stereo kadre turime iš pradžių 8 arba 16 bitų pirmojo kanalo duomenų, o po to seka antrojo kanalo duomenys, t. y. abiejų kanalų informacija yra saugoma tuo pačiu laiko momentu, o ne pirmiau laikant visus vieno kanalo kadrus bei po jų kito kanalo kadrus.

2.4. Nuskaitytų duomenų skaidymas

Kadangi duomenų seką gali sudaryti kelios dešimtys ar net šimtai tūkstančių narių, todėl tolesniems skaičiavimams toks didelis masyvas nėra tinkamas, t.y. gali užtrukti labai daug laiko. Taigi jį galima suskaidyti į mažesnius blokus. Tarkim kiekvienas blokas turi laikyti po 0,02 sekundės informacijos, taigi per vieną sekundę turėsime 50 blokų. Garsinio failo trukmė yra apskaičiuojama taip:

*Trukmė = failo dydis / (Kadro dydis * FPS).*

Tada galime rasti vieno bloko dydį:

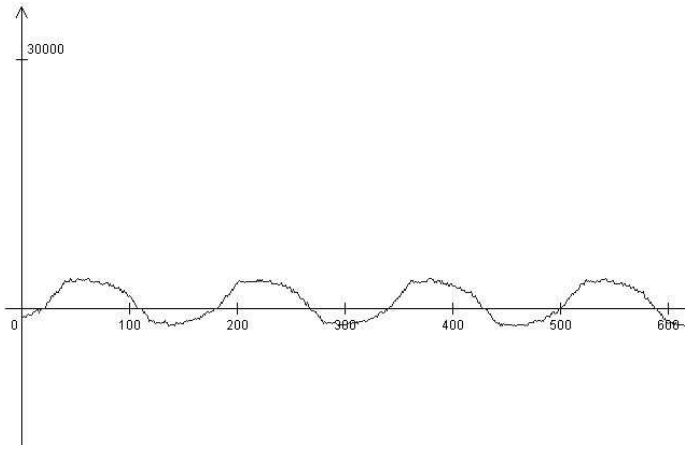
*Dydis = 0,02 * Sekos dydis / Trukmė.*

Taigi kuo didesnė trukmė, tuo daugiau blokų turėsime. Taip pat reikia atkreipti dėmesį į galimą duomenų persidengimą, pavyzdžiui, kitas blokas galėtų apimti pusę (ar daugiau) ankstesnio bloko ir pusę naujų duomenų. Tokiu atveju turėtume tam tikrą priklausomybę tarp blokų, o tai galėtų suteikti tam tikro stabilumo kalbos atpažinimo rezultatams. Visgi taip padidintume blokų skaičių, todėl išaugtų ir laiko sunaudojimas, kuris ir taip gali sukelti nemažai problemų. Dėl to svarbu įvertinti vieno ar kito pakoregavimo privalumus ir trūkumus.

2.5. Triukšmo įtakos mažinimas

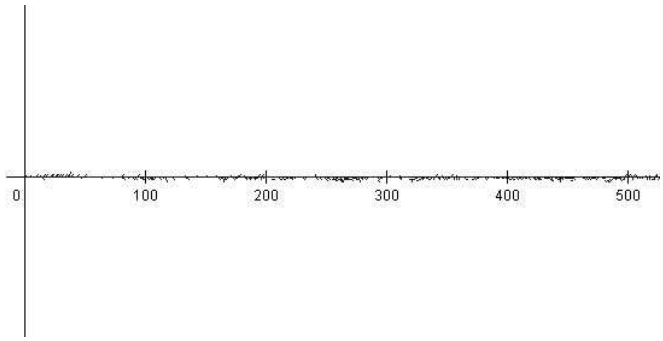
Taigi pirmiausiai buvo nuspręsta panagrinėti fono be kalbėjimo įtaką ir įsivesti papildomus parametrus, padedančius nustatyti, kur yra spėjama žodžio pradžia ir pabaiga.

Fonas be kalbėjimo su triukšmingu mikrofonu atrodo maždaug kaip sinusoidė (9 pav.):



9 pav. Tylos grafikas įrašius su triukšmingu mikrofonu

Eksperimentiškai nustatyta, kad fono periodas yra apie 0.02s , pvz.: 8000 kadrų per sekundę dažnio duomenims ciklas apima 160 skaičių. Prafiltravus grafikas turėtų atrodyti maždaug kaip 10 pav.:



10 pav. Tylos grafikas po filtravimo

Gana efektyvus filtras trumpoms sekoms (pvz.: kelių sekundžių) buvo imti garsinių duomenų (skaičių) sekos pradžią (pvz.: 160 skaičių) žinant, kad tai tikrai fonas be kalbėjimo, ir tolesniuose sekos kadruose atimti atitinkamus šios sekos narius (spėjamą sinusoidę). Visgi ilgesniems garsinių duomenų segmentams toks filtravimas nėra efektyvus, nes dažnai yra būdingas sinusoidės periodo nukrypimas į vieną ar kitą pusę. Todėl tokiais atvejais būtinas dinaminis spėjamos sinusoidės periodo atnaujinimas:

- Tarkim imant vieną iš 16 bitų garsinio signalo kadrų, turime du skaičius nuo -128 iki 127. Pirmasis skaičius atlieka nedidelę reikšmę, t.y. keičia grafiko reikšmę tik savo ribose, o antrasis yra esminis, nes apima diapazoną maždaug nuo -32000 iki 32000. Taigi nustačius, kad šis antrasis skaičius vis labiau nesutampa su spėjama sinusoide, ir žinant, kad tikrai turime foną be kalbėjimo, galime atnaujinti sinusoidės reikšmes.
- Kitas būdas būtų nelaukti, kol reikšmės ims nesutapti, o iš karto jas atnaujinti.

Kad dinaminis atnaujinimas veiktų efektyviai reikėjo tiksliai nustatyti, kur yra fonas be kalbėjimo. Tam įvedžiau papildomus parametrus, pagrįstus garsinių duomenų skirtumų vidurkiu (panašiai kaip DDTW algoritme):

$$\frac{1}{N-1}\sum_{i=1}^{N-1}\left|\frac{(x_i - x_{i-1}) + \frac{x_{i+1} - x_{i-1}}{2}}{2}\right|, \text{ kai turime seką } x_0,\ldots,x_N.$$

Čia x_i – garsiniai duomenys (skaičiai), o N – bloko dydis. Pažymėkime, šiuos parametrus $\boldsymbol{vid_0}$, ... , $\boldsymbol{vid_{n-1}}$, kur $\boldsymbol{n}$ – blokų skaičius (parametrų skaičius).

Taigi eksperimentiškai nustatyta, kad fono be kalbėjimo parametrai $\boldsymbol{vid_n}$ neviršija tam tikros ribos (be didesnio triukšmo šios reikšmės yra labai panašios ir gali skirtis vos per kelis vienetus) , todėl norint imti tik kalbėjimo segmentus užtenka skaičiuoti LSP parametrus tik, kai viršijama ši riba. Parametram $\boldsymbol{vid_n}$ vėl sumažėjus žemiau iki ribos , galime laikyti, kad žodis baigėsi. Visgi dažnai pasitaiko, kad kai kuriems priebalsiams reikšmės irgi gali nukristi žemiau paskaičiuotos ribos, todėl įvedami papildomi tikrinimai, ar pakankamai arti prieš šį sumažėjimo intervalą ir po jo tęsiasi kalbėjimo segmentas, t.y. yra parametrų $\boldsymbol{vid_n}$ viršijančių mūsų ribą. Be to, galimas ir priešingas atvejis – parametrų laikinas padidėjimas segmente be kalbėjimo, todėl tikrinama, ar kalbėjimo segmentas yra pakankamai ilgas (jeigu ne, tai triukšmo segmentą atmetam).

Kaip papildomus parametrus įvedžiau garsinių duomenų minimumų ir maksimumų skaičiavimus kiekviename bloke, nes kartais pagal parametrus $\boldsymbol{vid_n}$ tam tikri suaktyvėjimai (garsai) gali būti neatspindėti. Čia taip pat galima išvesti tam tikrą ribą, kurią viršijus fiksuojame tikėtiną žodžio pradžią arba tęsinį.

Taip pat triukšmo filtravimui taikiau *Hamming'o* lango funkciją:

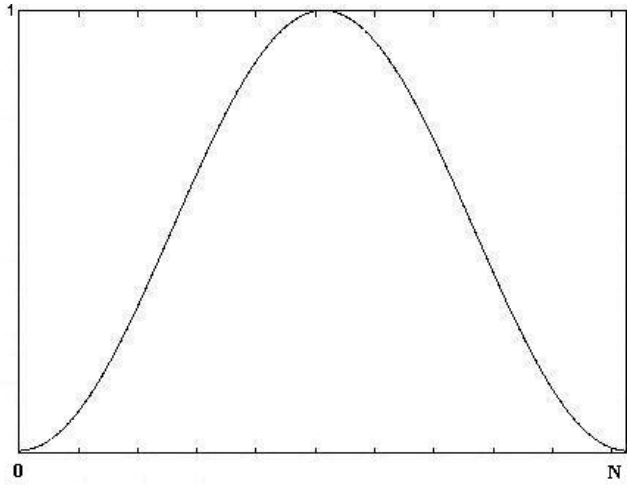
$\boldsymbol{w_i = 0.5(\ 1 + cos(\ (x - 0.5)2\pi\ )\ )}$, kur $\boldsymbol{x = i / N}$, $\boldsymbol{0 \le x \le 1}$ ir $\boldsymbol{0 \le i \le N}$, kai N – bloko dydis.

Šios funkcijos grafikas pavaizduotas 11 pav.

Tada iš bloko garsinių duomenų $\boldsymbol{u_i}$ (kai $\boldsymbol{0 \le i \le N}$) naujieji duomenys $\boldsymbol{v_i}$ gaunami imant gretimų duomenų skirtumus ir dauginant iš atitinkamo *Hamming'o* lango funkcijos koeficiento $\boldsymbol{w_i}$:

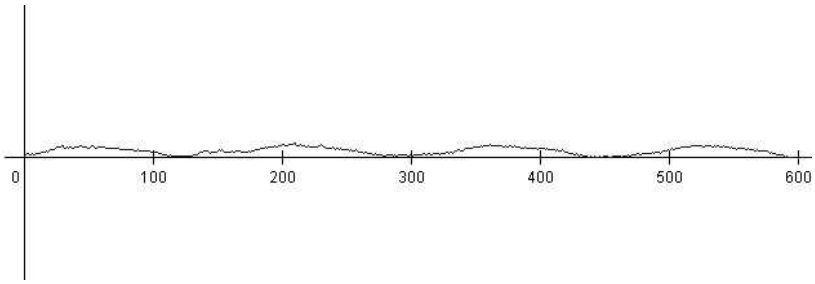
$$\boldsymbol{v_i = w_i\ (\ u_i - u_{i-1}\ )}\ , \text{ kai } \boldsymbol{0 < i \le N}.$$

Toks garsinių duomenų perskaičiavimas gana sėkmingai sumažina galimo periodinio triukšmo įtaką.



11 pav. Hamming‘o lango funkcija N ilgio blokui

Beje, su mažesnio triukšmingumo mikrofonu tylos grafikas (12 pav.) kiek geresnis:



12 pav. Tylos grafikas įrašius su mažesnio triukšmingumo mikrofonu

Tokiu atveju papildomas triukšmo filtravimas nereikalingas ir užtenka naudoti tik *Hamming‘o* lango funkciją.

2.6. Parametrų skaičiavimas ir atpažinimas

Kiekvienam blokui, kuris patenka į surasto žodžio sritį, reikia paskaičiuoti spektrinės poros (LSP) parametrus, o po to jiems pritaikyti iteracinį LBG kvantavimą.

Pirmiausiai randami tiesinės prognozės parametrai (LPC) pagal *Levinson – Durbin* rekursijos algoritmą, o tada pagal juos remiantis Čebyševo polinomais apskaičiuojami LSP parametrai. Be to, taikomas ¾ blokų persidengimas, kuris leidžia surinkti daugiau informacijos palyginimui. Algoritme naudojama 10 LPC parametrų ir analogiškai 10 LSP parametrų kiekvienam blokui. Taip pat skaičiuojant LSP parametrus atsižvelgiama į WAVE formato dažnį, t.y. kadrų per sekundę skaičių, nes kuo jis didesnis, tuo daugiau duomenų pateikiama parametrų ieškojimo algoritmui.

Kai turime visus LSP parametrus, atliekamas iteracinis LBG kvantavimas 8 kvantų (kodo vektorių) skaičiavimas (toks kvantų skaičius buvo optimalus tiek kokybės, tiek laiko atžvilgiu).

Visi šie veiksmai tinka tiek šablonams, tiek kalbos atpažinimo žodžiams (iš failų arba tiesiogiai iš mikrofono). Jei norime žodį atpažinti naudojamas Dinaminės laiko skalės tempimo algoritmas. Jis buvo šiek tiek modifikuotas:

- Kad žodžiai būtų palyginti nenutolstant nuo atitinkamų jų segmentų (pirmas parametras su pirmu, paskutinis su paskutiniu ir panašiai), įvedamas lyginimas atsižvelgiant į testuojamo žodžio ilgį. Tarkim šablono ilgis yra M , atpažįstamo žodžio – N, tai turim koeficientą $k = M / N$ ir imant i-tąjį atpažįstamo žodžio parametrą reikia jį lyginti su j-tuoju šablono parametru, kai $j = i*k$.
- Leidžiama nutolti nuo atstumų matricos įstrižainės per dvi jos eilutes arba du stulpelius, t.y. imam i-tąjį ir j-tąjį, $(i + 1)$-tąjį ir $(j – 1)$-tąjį, $(i – 1)$-tąjį ir $(j + 1)$-tąjį, $(i + 2)$-tąjį ir $(j – 2)$-tąjį, $(i – 2)$-tąjį ir $(j + 2)$-tąjį parametrus ir panašiai.
- Jei žodžių ilgiai stipriai skiriasi (pavyzdžiui 1.5 karto), leidžiama nutolti nuo atstumų matricos įstrižainės per vieną eilutę arba stulpelį.

Šablonui su mažiausia paklaidų suma priskiriama pirma reitingo vieta, antram pagal mažumą – antra ir taip iki penktos vietos.

2.7. Atpažinimo kokybės įvertinimas

Norint įvertinti, koks atpažinimo tikslumas yra su dideliu duomenų kiekiu, reikia apibrėžti reitingo kreivės modelį, kuris yra supaprastintas ROC (angl. *Receiver operating characteristic*) kreivės variantas, kai ROC kreivė parodo sistemos jautrumą nuo atmetimo slenksčio dydžio. Tarkim, jeigu turime dvi būsenas – atpažino ir neatpažino, tai mūsų atveju pradinis slenkstis yra pirma vieta ir atmetame žemiau pirmos vietos esančius objektus, su antru slenksčiu jau atmetame žemiau antros vietos esančius objektus ir kt. Kadangi turime reitingus nuo 1 (atpažino) iki penktos vietos, tada kreivės duomenys yra penki taškai, t.y. penki skaičiai iš intervalo [0, 1], pavyzdžiui, [0.8, 0.85, 0.9, 0.95, 0.99] reiškia, kad pirmoje vietoje turime 80% testų, pirmoje ir antroje – 85%, pirmoje, antroje ir trečioje – 90% ir kt. Idealiu atveju reikėtų imti tiek reitingo vietų, kiek žodyne yra žodžių.

2.8. Programos galimybės

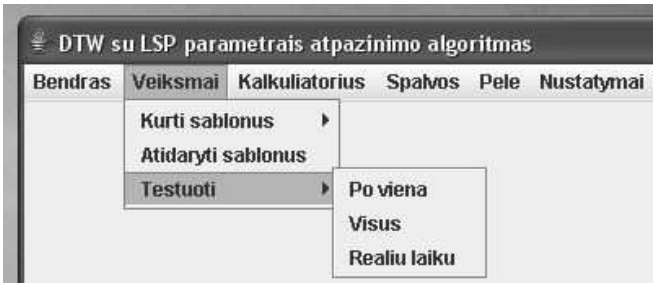
Programavimui buvo naudojama Java kalba su „javax.sound.sampled.*“ paketu WAVE formatui nuskaityti ir kurti. Pagrinde buvo naudojamas dviejų kanalų 8000 kadrų per sekundę su 16 bitų vienam kanalui formatas (tokiu formatu programa įrašo WAVE failus ir nuskaito iš mikrofono vykstant atpažinimui realiu laiku), tačiau programa priima ir kitus (8 bitų vienam kanalui, mono ir su didesniu kadrų per sekundę dažniu) failus. Taigi tarp pradinių funkcijų galima paminėti WAVE failų grojimą ir įrašymą (13 pav.):

13 pav. Programos „Bendras“ meniu skiltis

Antroji meniu skiltis ***Veiksmai*** yra skirta atpažinimo apmokymui ir testavimui. Šablonų duomenys išsaugomi atskiruose *.sab failuose, kuriuos galima kurti dviem būdais: arba nuskaityti po vieną WAVE failą atskirai, arba pažymėjus (Shift + rodyklių ir End klavišai klaviatūroje) visus reikiamus failus iš karto (14 pav.):

14 pav. Šablonų kūrimo pasirinkimas

Šablonų (WAVE failų) pavadinimai turi sutapti su juose įrašytais žodžiais ar žodžių sekomis. Tokiu atveju *.sab failuose išsaugomi visi apskaičiuoti parametrai, kuriuos dar galima tiesiog atidaryti jau iš egzistuojančių *.sab failų pasirinkus ***„Atidaryti sablonus“*** (to daryti nebūtina, jei prieš tai buvo kuriamas tas pats šablonų failas). Testavimas įmanomas irgi keliais būdais (15 pav.) – po vieną failą, keliais pažymėtais (kaip su šablonais) bei realiu laiku sakant į mikrofoną:



15 pav. Testavimo pasirinkimai

Atpažįstant po vieną failą konsolėje ir programos lange parodomi penkios pirmos reitingo vietos pagal mažiausias paklaidas. Antruoju atveju testai turi būti pavadinti tokiais pačiais pavadinimais kaip ir šablonai, nes kitaip negalėtume suprasti, ar teisingai buvo atpažinta. Kaip rezultatas konsolėje išvedama kiekvieno testo informacija (arba atpažinta, arba žemesnė reitngo vieta iki 5, arba neatpažinta, jei žemiau 5) ir pabaigoje bendri rezultatai pagal reitngo kreivę. Kai realiu laiku į mikrofoną tariami žodžiai, tada rezultatai išvedami į konsolės langą kaip ir pirmuoju atveju su atskirais testais iš failų. Šiuo atveju yra įvestas vienos minutės laiko limitas.

Taip pat galima pakeisti kai kuriuos programos nustatymus (16 pav.) – įjungti arba išjungti kvantavimą skaičiuojant šablonų arba atpažįstamų žodžių parametrus, su arba be *Hamming'o* lango garsiniams duomenims, su arba be triukšmo šalinimo ypač triukšmingiems garsiniams duomenims, ieškant žodžių imti didesnę arba mažesnę žodžio ilgio ribą:

16 pav. Nustatymų meniu

Na, ir likę trys meniu skiltys yra skirtos kompiuterio valdymo balsu imitavimo situacijoms: kalkuliatoriui (17 pav.), lango spalvos keitimui ir pelės žymeklio padėties keitimui. Tam, kad būtų leidžiama pradėti imitaciją, reikia atidaryti šablonų failą su reikiamu žodynu. Kalkuliatoriaus žodynas apima skaitmenis nuo 0 iki 9 ir penkias operacijas – sudėti, atimti, dauginti, dalinti ir rezultatas:

17 pav. Kalkuliatoriaus imitavimo pavyzdys

Jei skaičiai yra keliaženkliai, tai skaitmenis reikia išvardinti iš kairės į dešinę. Taip pat galima atlikti kelias operacijas iš eilės, pavyzdžiui, 2 * 10 / 4. Lango spalvos keitimo žodynas susideda iš dešimties spalvų – balta, raudona, žalia, mėlyna, geltona, oranžinė, rožinė, žydra, violetinė ir juoda – ir veiksmų pabaigos operacijos „nutraukti“. Analogiškai pelės žymeklio padėčiai keisti naudojami žodžiai kairėn, dešinėn, viršun, apačion ir nutraukti.

2.9. Rezultatų analizė

Testavimas buvo atliekamas keliomis kryptimis: su triukšmingu mikrofonu, su mažatriukšmiu mikrofonu, su kolegos magistranto Daliaus Dobravolsko duomenimis bei su Matematikos ir informatikos instituto duomenimis.

Pirmiausiai panagrinėsime triukšmingo mikrofono duomenis su 22 žodžių baze (skaitmenys nuo 0 iki 9 ir komandos atgal, atidaryti, atnaujinti, ieškoti, išsaugoti, naujas, ne, pirmyn, taip, taisyti, trinti, uždaryti). Šablonams buvo naudojami žodžių pasikartojimai po tris kartus kiekviename faile, o testams failai po vieną žodį. Iš viso kaip testai buvo panaudoti 210 skirtingi failai ir rezultatai palyginti su trimis skirtingomis sąlygomis: su *Hamming‘o* langu ir be triukšmo šalinimo, be *Hamming‘o* lango ir su triukšmo šalinimu bei be *Hamming‘o* lango ir be triukšmo šalinimo (2 ir 3 lentelės):

2 lentelė. Triukšmingo mikrofono atpažinimo rezultatai pagal reitingo vietas procentais

	atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Tik su *Hamming‘o* langu	84.76%	5.71%	1.43%	2.38%	1.9%
Tik su triukšmo šalinimu	78.57%	10.48%	2.38%	1.43%	0.95%
Be nieko	75.24%	8.57%	3.3%	3.3%	2.86%

3 lentelė. Triukšmingo mikrofono atpažinimo rezultatai pagal reitingo kreivės taškus

	1	2	3	4	5
Tik su *Hamming‘o* langu	0.8476	0.9048	0.919	0.9429	0.9619
Tik su triukšmo šalinimu	0.7857	0.8905	0.9143	0.9286	0.9381
Be nieko	0.7524	0.8381	0.8714	0.9048	0.9333

Taigi pagal reitingo kreivės taškus matome, kad pagal pirmas vietas geriausiai veikia Hamming‘o lango filtras, tačiau jau prie pirmų pridėjus antras ir trečias vietas jį beveik pasiveja triukšmo šalinimo filtras. Visgi tolesniems testavimams naudosime tik *Hamming‘o* lango filtrą, nes triukšmo lygis bus mažesnis.

Mažatriukšmiui mikrofonui testavimas buvo atliktas realiu laiku su kalkuliatoriaus (skaitmenys ir penkios operacijos), spalvų (10 spalvų) ir krypčių (keturios kryptys) žodynais. Kaip šablonai buvo imami failai po vieną atitinkamą žodį. Su kalkuliatorium ir spalvomis buvo ištarta po 200 žodžių, o su kryptimis – 100 ir pagal rezultatus (4 ir 5lentelės) galima teigti, kad su tokiom sąlygom galima sėkmingai imituoti valdymą balsu:

4 lentelė. Mažatriukšmio mikrofono atpažinimo rezultatai pagal reitingo vietas procentais

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Kalkuliatoriaus žodynas	94.5%	3.5%	1.5%	0.5%	0%
Spalvų žodynas	93.5%	5.0%	0.5%	0.5%	0.5%
Krypčių žodynas	95.0%	5.0%	0%	0%	0%

5 lentelė. Mažatriukšmio mikrofono atpažinimo rezultatai pagal reitingo kreivės taškus

	1	2	3	4	5
Kalkuliatoriaus žodynas	0.945	0.98	0.995	1.0	1.0
Spalvų žodynas	0.935	0.985	0.99	0.995	1.0
Krypčių žodynas	0.95	1.0	1.0	1.0	1.0

Visgi pagal šiuos testavimus galima teigti, ilgesni (daugiau negu du skiemenys) žodžiai geriau atpažįstami negu trumpi (vienas arba du skiemenys). Pavyzdžiui, su spalvomis daugiausia problemų kilo su dviskiemeniais žodžiais (balta, juoda, žalia ir žydra).

Dalius Dobravolskas sudarė šablonų ir testų bazę (WAVE failai) tik su skaitmenimis, bet su 20 skirtingomis sąlygomis (skirtingi balsai, triukšmas, lėta arba greita tartis). Iš viso susidarė po 10 šablonų kiekvienom sąlygom ir bendrai 197 testai. Rezultatai pateikiami 6 lentelėje:

6 lentelė. Atpažinimo rezultatai su Daliaus Dobravolsko duomenimis

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Reitingo vietos procentaiss	81.7%	9.1%	4.6%	2.5%	1%
Reitingo kreivės taškai	0.817	0.909	0.954	0.98	0.99

Visgi vidutiniai rezultatai rodo tik bendrą vaizdą, nes palankiomis sąlygomis įrašyti duomenys iš karto duodavo beveik šimtaprocentinį atpažinimą, tuo tarpu su labai greitai tariamais žodžiais kildavo nemažų problemų.

Panaši situacija susidarė ir su Matematikos ir informatikos instituto baze (7 lentelė):

7 lentelė. Bendri atpažinimo rezultatai su Matematikos ir informatikos duomenimis

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Reitingo vietos procentaiss	79.4%	4.6%	2%	1.5%	1.3%
Reitingo kreivės taškai	0.794	0.84	0.86	0.875	0.888

Na, šiuo atveju mažesni procentai su 2-5 vietomis gaunasi dėl to, kad turime net 111 dydžio žodyną (akis, antras, asmuo, aštuoni, atlikti, atvejis ir kt.). Iš viso yra 17 skirtingų balsų su šablonais ir testais po vieną žodį, todėl testavimas buvo atliekamas į abi puses, t.y. iš pradžių kaip šablonus imame pirmąją failų grupę ir testuojame su antrąją, o po to – atvirkščiai. Palyginimui pateikiame geriausius rezultatus su 16-tu balsu (8 lentelė) ir blogiausius su 8-tu balsu (9 lentelė), kur ypač stiprūs pašaliniai mikrofono garsai, o pilni rezultatai pateikiami priede Nr. 1:

8 lentelė. Atpažinimo rezultatai su 16-to balso duomenimis

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Reitingo vietos procentaiss	99%	0%	0%	0%	0%
Reitingo kreivės taškai	0.99	0.99	0.99	0.99	0.99

9 lentelė. Atpažinimo rezultatai su 8-to balso duomenimis

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Reitingo vietos procentaiss	44%	11%	3%	3%	1%
Reitingo kreivės taškai	0.44	0.55	0.58	0.61	0.62

Taigi tiktai su nepalankiom sąlygom įrašytais duomenimis atpažinimas būna gana blogas, tačiau daugeliu atveju atpažinimo procentas buvo apie 90% arba netgi geresnis. Beje, mažesnis atpažinimo tikslumas buvo su akustiškai panašiais žodžiais kaip geras ir gerai arba metas ir metai.

3. Kitų autorių darbų apžvalga

Dabar apžvelgsim Kauno Technologijos universiteto [6] ir Matematikos ir informatikos instituto projektų, susijusių su lietuvių kalbos atpažinimu, pasiekimus.

3.1. Kauno Technologijos universiteto projektas

„Balso technologijų taikymo lietuvių kalbai analizė ir perspektyvinių veiklos krypčių pagrindimas“ [6] projekto tikslai buvo detaliai išnagrinėti balso technologijų pasiekimus pasaulyje ir Lietuvoje bei parodyti, kokiose srityse galima jas efektyviai pritaikyti. Buvo paruoštos demonstracinės programos kalbiniam dialogui su kompiuteriu (užduoti klausimus ir gauti atsakymus), balso įrašų stenografavimui (kalbos įrašas į tekstinį redaktorių), ilgų pokalbių apdorojimui (galima atrinkti segmentus pagal kalbėtoją), balsinei internetinei naršyklei, balso panaudojimui namų automatikoje (įjungti/išjungti elektros lemputę), automatiniam lietuviško teksto skaitymui balsu (žodžių sintezė pagal tekstą) ir kitiems dalykams. Darbe pabrėžiama, kad paruošta

programinė ir aparatūrinė įranga yra skirta elektros ir matavimo prietaisams valdyti balsu bei rezultatams išvesti balsu. Buvo pasirinktas projekcinis kalbos atpažinimo algoritmas, pagrįstas dinaminio laiko ištiesinimo algoritmo (DTW) modifikacija. Be to, naudojamas fonetiškai segmentuotų kalbos signalų parametrų vidurkinimas ir originalus balso komandų ribų aptikimo algoritmas. Darbo išvadose akcentuota, kad atpažinimo tikslumas labai priklauso nuo algoritmo apmokymo, mikrofono kokybės ir diktoriaus treniruotumo vienodai ištarti žodį. Tarp rezultatų galima paminėti valdymo balsu įrangą 8-iems elektros prietaisams, lietuvių kalbos sintezatorių "Aistis" ir elektrinių matavimų valdymo balsu įrangą.

Bandymams su šio projekto algoritmu panaudojau pateiktą programą žodžių palyginimui. Kaip šablonus paėmiau po vieną žodį iš 22-iejų žodžių žodyno įrašyto su triukšmingu mikrofonu ir po vieną testą kiekvienam žodžiui. 10 lentelėje pateikiu eksperimentų rezultatus:

10 lentelė. Atpažinimo rezultatai su 22 žodžių žodynu

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
Reitingo vietos procentaiss	81.8%	13.6%	0%	4.5%	0%
Reitingo kreivės taškai	0.818	0.954	0.954	1.0	1.0

Visgi patikrinimas užtruko, nes buvo galima lyginti tik po du failus, t.y. kad įvertintume vieno žodžio atpažinimą reikėjo jį palyginti su 22-iem šablonais.

3.2. Matematikos ir informatikos instituto projektai

Iš pradžių reikia padėkoti Matematikos ir informatikos institutui, nes be jų suteiktos 111-os žodžių bazės nebūtų pavykę detaliai pratestuoti mano darbe parengto lietuvių kalbos algoritmo.

Tarp pagrindinių projekto tikslų yra tirti atskirai pasakytų žodžių požymius, kurti lietuvių kalbos atpažinimo programinę įrangą, nagrinėti ir realizuoti įvairius kalbos atpažinimo algoritmus, atsižvelgti į garsinio signalo triukšmus, kurti įvairias kalbos atpažinimo taikymo priemones ir kiti darbai. Buvo išbandyti ir realizuoti vektoriaus kvantavimo, kepstrinių požymių skaičiavimo, paslėptų Markovo modelių, neuroninių tinklų algoritmai ir jų įvairios modifikacijos. Kaip projekto rezultatus galima paminėti lietuvių kalbos atpažinimo programinę įrangą, interneto puslapių atidarymo balsu prototipą, audiovizualinės vartotojo sąsajos su kompiuteriu kūrimą ir įvairius balso technologijų taikymo tyrimus.

Testuojant su pateikta programa (2005-2006 metų nepriklausomas nuo balso lietuviškų žodžių atpažintuvas su 111 žodžių žodynu) kilo keblumų parenkant tinkamus parametrus: kaip skaičiuoti žodžio aptikimo slenksčius (automatiškai ar su konkrečiu signalo energijos slenksčiu), po kiek kadrų

skirti žodžio pradžios ir pabaigos aptikimui, kokie žodžio pradžios ir pabaigos koeficientai labiausiai tinka. Su pradiniais parametrais kažkodėl nieko neatpažino. Be to, testavimo eigą apsunkino ir atpažinimas tik iš vieno failo. Visgi po kelių konfigūracijos bandymų pavyko gauti gana gerus rezultatus. 11 lentelėje pateikiu 111 žodžių bazės 8-to, 16-to balsų (mano algoritmo rezultatai 8-toje ir 9-toje lentelėse), ir kelių kitų balsų rezultatus su pirmais 20 žodžių:

11 lentelė. MII programos rezultatai su 111 žodžių baze

	Atpažino
8-tas balsas	55%
16-tas balsas	80%
1-as balsas	65%
2-as balsas	75%
7-tas balsas	90%

Šios programos privalumas yra tas, kad jos nereikia apmokyti, ir atlikti atpažinimą su bet kokiu balsu, tačiau dėl to nukenčia rezultatų tikslumas.

Išvados

- Buvo suprogramuotas lietuvių kalbos atpažinimo algoritmas pagrįstas dinaminės laiko skalės tempimo ir spektrinės poros algoritmais.
- Darbo metu buvo sėkmingai imituotos kompiuterio valdymo situacijos: kalkuliatorius, lango spalvos keitimas ir pelės žymeklio judinimas.
- Atlikti išsamūs suprogramuoto atpažinimo algoritmo testavimai parodė, kad rezultatų kokybė labai priklauso nuo kompiuterio mikrofono keliamo triukšmo, diktoriaus žodžių tarimo greičio ir pašalinių aplinkos triukšmų.
- Pagal eksperimentų rezultatus galima teigti, kad ilgesnių žodžių (daugiau negu du skiemenys) atpažinimo tikslumas yra didesnis negu trumpų žodžių (vienas arba du skiemenys).
- Taip pat pastebėta, kad akustiškai panašūs žodžiai atpažįstami blogiau negu neturintys jokio panašumo su visais žodyno žodžiais.
- Realizuojant kalbos atpažinimo algoritmą daugiausia problemų kilo su tiksliu žodžio pradžios ir pabaigos radimu. Nors atpažinimas daugeliu atvejų veikia gana gerai, visgi galima teigti, jog šios problemos sprendimą dar įmanoma optimizuoti.
- Bandant Dinaminės laiko skalės algoritmą buvo atlikta nemažai modifikacijų, kurios padėjo pakelti atpažinimo tikslumą, tačiau kai kada pagerėjimas jausdavosi tik tam tikrose situacijose, todėl galutinė realizacija yra tik optimalus variantas daugeliui atvejų, bet nebūtinai tiksliausias iš anksčiau bandytų.

Literatūros sąrašas

[1] Eamonn J. Keogh, Michael J. Pazzani. Derivative Dynamic Time Warping. 2001 m. - URL: http://www.cs.rutgers.edu/~mlittman/courses/lightai03/DDTW-2001.pdf. 146 KB

[2] Speech Compression. – URL: http://www.data-compression.com/speech.html

[3] Fang Zheng, Zhanjiang Song, Ling Li, Wenjiang Yu, Fengzhou Zheng, Wenhu Wu. The Distance Measure For Line Spectrum Pairs Applied To Speech Recognition. 1998 m. – URL: http://www.work.caltech.edu/ling/pub/icslp98lsp.pdf

[4] Vector Quantization. – URL: http://www.data-compression.com/vq.html

[5] Insung Lee, Hong Chae Woo. Encoding of Speech Spectral Parameters Using Adaptive Quantization Range Method. 2001 m. ETRI (Electronics and Telecommunications Research Institute) Journal - URL: http://etrij.etri.re.kr/Cyber/servlet/GetFile?fileid=SPF-1042441730228. 101 KB

[6] dr. A. Rudžionis, dr. K. Ratkevičius, dr. V. Rudžionis, dr. P. Kasparaitis, dr. B. Šalna „Balso technologijų taikymo lietuvių kalbai analizė ir perspektyvinių veiklos krypčių pagrindimas“ 2001 m. KTU, VU, TEC URL: http://www.likit.lt/frames/balso_tech/balsotech_st.htm.

[7] Microsoft internetinė informacinė sistema URL: http://www.microsoft.com.

[8] R. P. Ramachandran, P.Kabal „The Computation of Line Spectral Frequencies Using Chebyshev Polynomials“ 1985 m. , INRS – Telecommunications Canada

[9] Hidden Markov Models URL: http://jedlik.phy.bme.hu/~gerjanos/HMM/node2.html

[10] prof. hab. dr. L. Telksnys, doc. dr. A. Lipeika, doc. dr. J. Lipeikienė, dr. M. Filipovič ir kiti „Automatinio lietuvių šnekos atpažinimo darbai“ 2002-2006 m. , Matematikos ir informatikos institutas, Vilnius.

Priedas Nr. 1

Atpažinimas su Matematikos ir informatikos instituto duomenimis

12 lentelė. Balso vidutiniai atpažinimo rezultatai procentais

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
1-as balsas	89.5%	3.3%	1.35%	0.85%	1.5%
2-as balsas	54.55%	7.45%	4.4%	2.75%	3.3%
3-as balsas	93.25%	1.35%	0.9%	0%	0.25%
4-as balsas	67.45%	4.2%	4.35%	3.5%	2%
5-as balsas	77.7%	9.7%	2%	2.5%	1.5%
6-as balsas	84.9%	3.6%	2.5%	1.5%	1%
7-as balsas	91.9%	3.1%	1%	0%	1.5%
8-as balsas	42.5%	9.5%	3.5%	3.4%	2.1%
9-as balsas	66%	7.5%	2.5%	2%	3%
10-as balsas	91.5%	4%	1.5%	0%	0%
11-as balsas	77%	3.5%	4%	1.5%	1.25%
12-as balsas	87%	3%	0.5%	3.5%	0%
13-as balsas	91%	3%	1%	0%	0.5%
14-as balsas	77.5%	6.3%	0.5%	2.7%	1.5%
15-as balsas	83.5%	2.5%	3%	0.5%	1.5%
16-as balsas	99%	0%	0%	0%	0%
17-as balsas	76%	5.5%	1%	1%	2%

13 lentelė. Balso vidutiniai atpažinimo rezultatai pagal reitingo kreivės taškus

	Atpažino	2 vieta	3 vieta	4 vieta	5 vieta
1-as balsas	0.895	0.928	0.9415	0.95	0.965
2-as balsas	0.5455	0.62	0.664	0.6915	0.7245
3-as balsas	0.9325	0.946	0.955	0.955	0.9575%
4-as balsas	0.6745	0.7165	0.76	0.795	0.815
5-as balsas	0.777	0.87	0.89	0.915	0.93
6-as balsas	0.849	0.885	0.92	0.925	0.935
7-as balsas	0.919	0.95	0.96	0.96	0.975
8-as balsas	0.425	0.52	0.555	0.589	0.61
9-as balsas	0.66	0.735	0.76	0.78	0.81
10-as balsas	0.915	0.955	0.97	0.97	0.97
11-as balsas	0.77	0.805	0.845	0.86	0.8725
12-as balsas	0.87	0.9	0.905	0.94	0.94
13-as balsas	0.91	0.94	0.95	0.95	0.955
14-as balsas	0.775	0.838	0.843	0.87	0.885
15-as balsas	0.835	0.86	0.89	0.895	0.91
16-as balsas	0.99	0.99	0.99	0.99	0.99
17-as balsas	0.76	0.815	0.825	0.835	0.855