VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

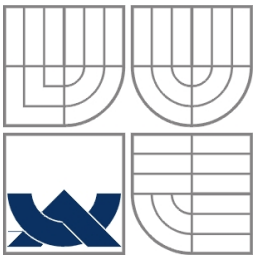
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SADA JAVAAPPLETŮ PRO DEMOSTRACI ZPRACOVÁNÍ ŘEČI

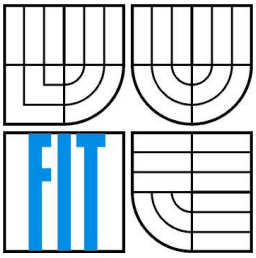
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Michal Kudr
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SADA JAVAAPPLETŮ PRO DEMOSTRACI ZPRACOVÁNÍ ŘEČI

SET OF JAVAAPPLETS DEMOSTRATIONS FOR SPEECH PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Michal Kudr
AUTHOR

VEDOUCÍ PRÁCE — doc. Dr. Ing. Jan Černocký
SUPERVISOR

BRNO 2010

Abstrakt

Cílem práce je seznámit se s metodami a technikami využívanými při zpracování řeči. Pomocí získaných znalostí navrhuji tři JavaApplety demonstrující vybrané metody. V této práci můžeme nalézt teoretický rozbor vybraných problémů.

Abstract

The goal of the thesis is being familiar with methods a techniques used in speech processing. Using the obtained knowledge I propose three JavaApplets demonstrating selected methods. In this thesis we can find the theoretical analysis of selected problems.

Klíčová slova

Řeč, signál, rámec, příznaky, základní tón, autokorelace, rozpoznávání, dynamické borcení času, skryté Markovovy modely, vektor, spektrum, trénování, likelihood.

Keywords

Speech, signal, frame, features, pitch, autocorrelation, recognizing, dynamic time warping, hidden markov's models, vector, spectrum, training, likelihood.

Citace

Kudr Michal: Sada JavaAppletů pro demonstraci zpracování řeči, diplomová práce, Brno, FIT VUT v Brně, 2010

Sada JavaAppletů pro demostraci zpracování řeči

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Dr. Ing. Jana Černockého.
Další informace mi poskytl Ing. Zdeněk Jančík.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Kudr
30.7.2010

Poděkování

Chtěl bych poděkovat doc. Dr. Ing. Janu Černockému a Ing. Zdeňku Jančíkovi za veškerou pomoc a rady, které mi poskytli.

Obsah

1 Úvod

Komunikace prostřednictvím mluvené řeči je základní a nejpřirozenější způsob dorozumívání mezi lidmi. Není proto divu, že při současných zvyšujících se možnostech výpočetní techniky usilují vědci a technici o to, aby se plnohodnotným partnerem člověka v mluveném dialogu mohl stát i počítač. Takový způsob komunikace by mohl být člověku velmi prospěšný a často by mu mohl i výrazně zjednodušit život. Pro dosažení takového cíle je potřeba algoritmicky a technicky vyřešit několik relativně komplikovaných úloh, které se týkají zejména zpracování řečového signálu, počítačové syntézy a automatického rozpoznávání řeči, atd.

Bohužel plnohodnotný dialogový režim člověka s počítačem prostřednictvím přirozené, plynule mluvené řeči není plnohodnotně vyřešen a je v současnosti stále ve fázi výzkumu, na kterém se podílejí výzkumné týmy z celého světa. Je to způsobeno především stálými problémy s rozpoznáváním spontánní řeči a dále i omezenými možnostmi součinnosti procesu klasifikace řečového signálu a procesu porozumění smyslu klasifikovaných vět.

I přes celou řadu zatím nedořešených problémů, s nimiž se systémy hlasové komunikace člověka s počítačem potýkají, dochází stále častěji k jejich praktickému nasazování v průmyslové a společenské praxi. Ve většině aplikací přitom jde o problémově orientovaná řešení, kdy komunikace je tématicky omezená, rozpoznávaný slovník se týká určité konkrétní oblasti a systém pracuje v prostředí s definovaným rušivým pozadím apod.

V mojí diplomové práci jsem se rozhodl demonstrovat 3 úlohy zpracování řeči ve formě JavaAppletů. Jedná se o detekci základního hlasivkového tónu, rozpoznávání izolovaných slov pomocí dynamického borcení času a rozpoznávání izolovaných slov pomocí skrytých Markovových modelů..

Proč JavaApplety? V dnešní době je stále populárnější (a hlavně snadnější) získávání znalostí a odpovědí na své otázky prostřednictvím internetu. Pro snadnější pochopení dané problematiky je možné na internetu nalézt velké množství demonstračních programů a simulací, které mohou například umožnit vyzkoušení si daného problému či algoritmu v praxi. I já jsem při studiu těchto možností hojně využíval a proto jsem se rozhodl pro implementování vybraných úloh právě ve formě JavaAppletů s možností jejich pozdějšího umístění právě na internet, kde mohou být dostupné pro všechny uživatele, zajímajících se o danou tématiku.

Popis vybraných úloh a metod pro jejich řešení se nachází dále v teoretické části práce.

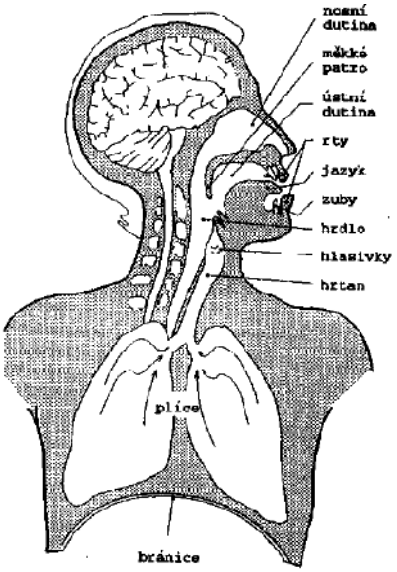
2 Vytváření a vlastnosti mluvené řeči

Mluvená řeč se přenáší **komunikačním kanálem** ve formě akustických vln, tzv. **akustického signálu**. Podstatou akustického (nebo též řečového) signálu je vlnění elastického prostředí v oboru slyšitelných frekvencí. Komunikačním kanálem rozumíme prostředí, kterým je akustický signál, tj. řeč, přenášen od svého zdroje, úst řečníka, až k příjemci informace, uším posluchače. V akustickém signálu řeči je zakódováno několik druhů informace. Vedle samotné akustické složky (amplitudově frekvenčního časového spektra) bývá z hlediska komunikace nejdůležitější informace lingvistická, daná například svojí fonetickou, morfologickou, syntaktickou, sémantickou či pragmatickou strukturou, protože vyjadřuje význam sdělované myšlenky. Akustický signál dále obsahuje jisté specifické informace o mluvčím, které respektují charakteristiky hlasového traktu řečníka a způsob artikulace (intonaci, rytmus řeči, barvu hlasu atd.), včetně případných anomálií, jako jsou například vady řeči apod., a také informace o emocionálním stavu řečníka (stresu, rozčilení, smutku, atd.).

2.1 Proces vytváření řeči

V této kapitole bude popsán proces vytváření řeči člověkem a popsány ústrojí, které se na tvorbě řeči podílejí.

Pro vytváření řeči existuje v lidském těle několik skupin orgánů, které se souhrnně nazývají **řečové (artikulační) orgány**, či **jen mluvidla (artikulátory)**. Na rozdíl od ucha, které se vyvinulo speciálně pro slyšení, nebývá produkce řeči zpravidla primárním úkolem těchto orgánů – jejich základní funkce jsou v lidském těle různé a často spolu navzájem nesouvisejí (např. dýchání, přijímaní potravy, cítění). Spojuje je tedy až spoluúčast na procesu vytváření řeči. Z hlediska tvorby řeči tyto řečové orgány tvoří hlasový trakt. U dospělého mužského jedince je celková délka hlasového traktu od hrtanu až ke rtům přibližně 17 cm a plocha jeho příčného průřezu se mění od nuly (v případě úplného uzavření traktu) až cca k 20 cm^2. Hlasový trakt lze rozdělit na tři základní ústrojí: dechové, hlasové a artikulační[1]. Hlasový trakt člověka je zobrazen na obr. č. 1.



Obrázek č. 1 – Hlasový trakt člověka[2]

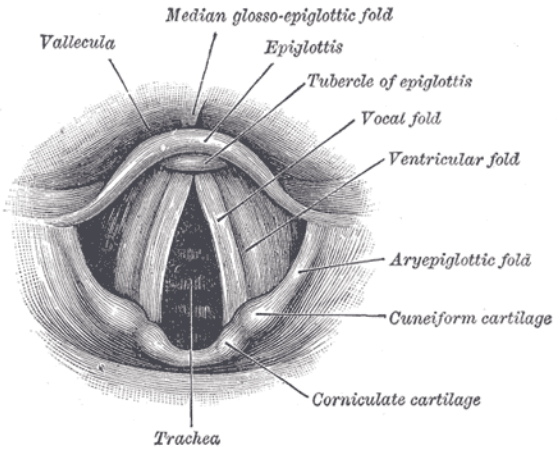
2.1.1 Dechové ústrojí

Dechové ústrojí představuje fundamentální zdroj energie pro řeč. Je umístěno v hrudním koši a tvořeno přívodní dýchací cestou, plícemi a s nimi funkčně spjatými dýchacími svaly (bránicí). Kapacita plic dospělého muže je v klidu asi 4-5 litrů, přičemž 1-2 litry tvoří tzv. zbytkovou kapacitu plic, která musí být vždy zachována. Vitální kapacita plic, tj. maximální rozdíl mezi úplně naplněnými a vypuštěnými plícemi, je přitom asi 5 litrů. Při nádechu dochází k pohybu vzduchu, který tak poskytuje zdroj energie pro řeč. Při výdechu potom vzniká v plicích výdechový proud vzduchu, který je v zásadě základním materiálem pro tvorbu řeči. Při dlouhém dýchání je do plic přivedeno a poté z plic odvedeno asi 0.5 litru vzduchu každých 3-5 sekund. Výdechový proud vzduchu je z plic odváděn průdušnicí (tracheou), a pak prochází hrtanem a nadhrtanovými dutinami, kde se modifikuje, a jako řečový signál je vyzařován rty do okolního prostoru. Rychlost, s jakou vzduch opouští plíce, je přibližně konstantní a je rovna asi 0.2 litru za sekundu. Trvání výdechu má vliv na to, jak dlouhý úsek řeči lze vytvořit bez přerušení. Síla výdechového proudu ovlivňuje způsob fungování hlasového ústrojí, a tím má vliv na sílu hlasu a částečně i jeho výšku. K vytvoření „slyšitelné“ řeči je zapotřebí z plic vytlačit v rozmezí několika sekund více než 0.5 litru vzduchu. Během běžné řeči se spotřebuje asi polovina vitální kapacity plic, zatímco při velmi hlasité řeči až 80%. Další nádech pak opět doplňuje vzduch do plic, čímž mimo jiné dodává nový materiál pro tvorbu řeči. Navenek se nádech projeví jako pauza v jinak souvislé řeči[1].

2.1.2 Hlasové ústrojí

Pojmem **hlasové ústrojí** (hlasový trakt, angl. vocal tract) se často také označuje celý systém pro vytváření řeči. Hlasové ústrojí je uloženo v **hrtanu** (larynx), viz. obr. č. 2, který je s plícemi spojen průdušnicí. Z hlediska tvorby řeči nejdůležitější část hlasového ústrojí tvoří **hlasivky** (plicae vocales), které se nacházejí v hrtanové dutině přímo za „ohryzkem“. Jsou to dvě ostré slizniční řasy, které vedou napříč hrtanem v místě jeho nejužšího průchodu. Jejich typická délka je asi 15 mm pro muže a 13 mm pro ženy. Z jedné strany jsou napojeny na chrupavky hlasivkové a z druhé strany na chrupavku štítnou. Jsou pokryty sliznicí a jejich základ tvoří hlasový vaz a hlasový sval. Prostor mezi hlasivkami tvoří hlasivková štěrbina trojúhelníkového tvaru. Jestliže člověk mlčí, pak hlasivky drží hlasivkovou štěrbinu odkrytou (asi 8 mm na šířku), takže jí může bez odporu proházet vzduch k dýchání. Hlasové ústrojí tak využívá klidového postavení hlasivek.

Při vytváření hlasu (fonaci) plní hlasivky jinou funkci – nacházejí se v tzv. hlasovém (fonačním) postavení. Výdechový proud vzduchu postupuje bez odporu z plic průdušnicí až k hrtanu. Zde se mu do cesty postaví překážka vytvořená hmotou hlasivek, které cestu vzduchu úplně uzavřou. Stažené hlasivky se pod tlakem vzduchu stávají pružnými a začínají kmitat. Hlasivky se přitom střídavě postupně otevírají a prudce uzavírají. Maximální velikost otevřené plochy hlasivkové štěrbiny během kmitání je u mužů asi 20mm^2 a u žen 14 mm^2. V důsledku kmitání hlasivek se vzduchový proud (do té doby homogenní) „rozdrobí“ tak, že se víceméně pravidelně střídá vždy kvantum hustšího a řidšího vzduchu – vzniká tzv. vzduchová vlna, kterou vnímáme jako zvuk. Tento periodický proud vzduchových pulsů tvoří základ lidského hlasu. Bývá označován termínem **základní (hlasivkový) tón** a představuje nosný zvuk řeči. Frekvence kmitání hlasivek se nazývá **fundamentální frekvence** nebo **frekvence základního hlasivkového tónu**. Tato frekvence je fyzikální charakteristikou řečového signálu a odpovídá výšce hlasu tak, jak ji vnímá posluchač[1].



Obrázek č. 2 – Hrtan a hlasivky[12]

2.1.3 Artikulační ústrojí

Artikulační ústrojí je posledním ústrojím, které se podílí na tvorbě řeči. Jeho význam spočívá v tom, že umožňuje vytvářet velké množství různých zvuků, které charakterizují mluvený jazyk. Skládá se jednak z **nadhrtanových dutin** a jednak z **artikulačních orgánů**, které jsou v těchto dutinách uloženy nebo je obklopují. Mezi nadhrtanové dutiny řadíme dutinu hrdelní, ústní a nosní. Hranici mezi těmito dutinami tvoří čípek (uvula), špička měkkého patra (velum), které zamezuje nebo umožňuje přístup vzduchu z dutiny hrdelní do dutiny nosní. Zatímco se nadhrtanové dutiny účastní procesu tvorby řeči pasivně (nepohybují se), artikulační orgány (artikulátory) se účastní tvorby řeči většinou aktivně – tvoří pohyblivé součásti artikulačního ústrojí a svým pohybem mění velikosti nadhrtanových dutin. Z hlediska vytváření řeči mezi nejvýznamnější artikulátory patří jazyk, rty, měkké patro, neboť se podílejí na vytváření největšího počtu různých zvuků. Dalšími artikulátory potom jsou zuby, tvrdé patro nebo čelisti. Artikulátorem je také hrtan, který kromě toho, že se zapojuje do tvorby znělosti, se také může pohybovat nahoru a dolů a měnit tak délku celého hlasového traktu. Vůbec nejsložitějším a nejdůležitějším artikulátorem je jazyk. Skládá se ze tří částí: hrotu (špičky), hřbetu a kořene, které jsou součástí stejné svalové struktury jazyka, ale každá může do jisté míry fungovat nezávisle. Jazyk je tak velice pružný a přizpůsobivý, schopný tvořit mnoho tvarů a rychle přecházet z jedné pozice do druhé. Právě variabilita umisťování jazyka vytváří nesčetné tvary ústní i hrdelní dutiny a vede tak k vytváření různých zvuků řeči.

Výrazných změn ve svém složení doznává zvuk v nadhrtanových dutinách, kam jako výdechový proud vzduchu (v podobě periodického proudu vzduchových pulsů nebo jako prostý proud vzduchu) postupuje z hlasového ústrojí. Tyto změny jsou dvojího druhu:

- **Vytváření tónové struktury**. Při průchodu základního hlasivkového tónu nadhrtanovými dutinami dochází vlivem rezonance uvnitř hrdelní, ústní a popř. i nosní dutiny ke změně rozložení akustické energie ve vznikajícím řečovém signálu. Akustická energie se soustředí kolem určitých frekvencí, kterým říkáme **formantové frekvence**. Oblasti koncentrace (zesílení) akustické energie se nazývají **formanty** a označují se čísly, počínaje formantem s nejnižší frekvencí, F_1, F_2, ..., F_n. Pokud se do procesu vytváření řeči zapojí i nosní dutina, dochází navíc vlivem jejich antirezonančních vlastností k potlačení některých frekvenčních oblastí, tzv. **antiformantů**. Vzniká složený zvuk tónového charakteru, kterému říkáme hlas. Tvoří podstatu znělých částí řeči, především samohlásek. Různé zvuky (tj. zvuky s různou

spektrální strukturou) přitom vznikají tak, že pohyblivé artikulátory mění tvar nadhrtanových dutin, a tím i frekvenční vlastnosti vytvářených zvuků.

- **Vytváření šumové složky.** Artikulátory ovlivňují průchod vzduchu nadhrtanovými dutinami. Svým pohybem mohou průchod na různých místech zúžit, úplně uzavřít nebo naopak uvolnit. Výdechový proud vzduchu se pak prodírá přes vytvořené překážky a podle charakteru překážky vzniká šum různého druhu. Šum tvoří základ těch částí řeči, kterým při popisu jazyka říkáme souhlásky. Různé zvuky se v tomto případě vytvářejí pomocí různých typů překážek umístěných na různých místech hlasového traktu[1].

2.2 Reprezentace řeči

Řeč jakožto mluvenou podobu jazyka je možné popisovat z různých lingvistických hledisek – akustického, artikulačního, fonetického, morfologického, syntaktického, sémantického či pragmatického. Vzhledem k náplni mé práce se dále budeme zabývat akustickou úrovní reprezentace řeči[1].

2.2.1 Akustická úroveň

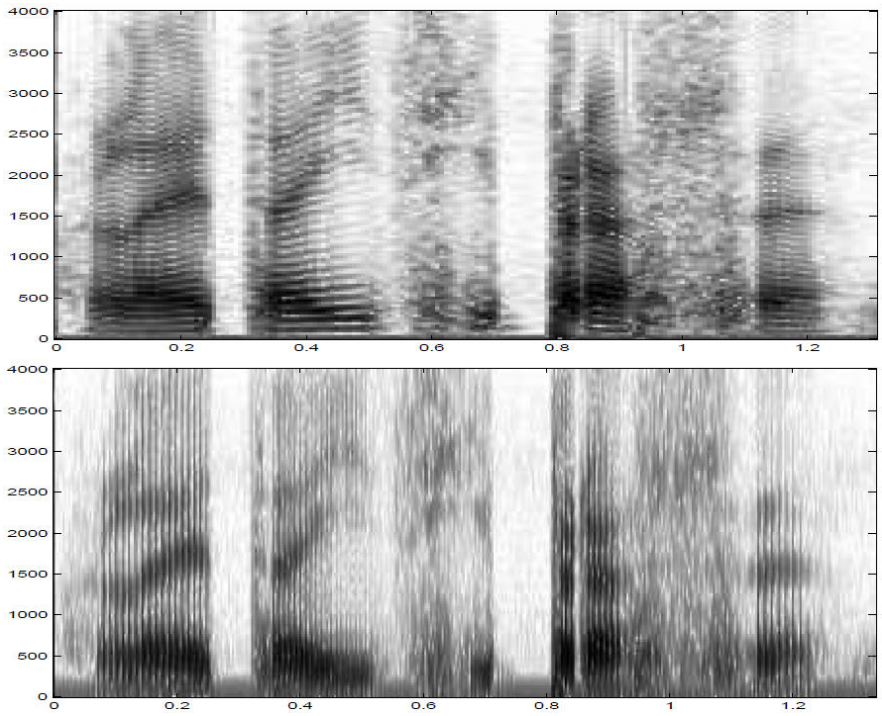
Akustická reprezentace popisuje řeč jako výstupní signál procesu vytváření řeči. Podstatou akustického signálu je vlnění elastického prostředí v oboru slyšitelných frekvencí vznikající kontrolovanými pohyby struktur hlasového ústrojí. Pro snímání řeči se používá mikrofon, který řeč převádí na elektrický signál. Elektrický signál se digitalizuje a reprezentuje jako posloupnost řečových vzorků. Akustickými rysy řeči jsou frekvence základního tónu, intenzita a rozdělení spektrální energie.

V časové oblasti se řečový signál reprezentuje pomocí **časového průběhu vlny**, který zobrazuje vývoj amplitudy signálu v čase. Časové průběhy odhalují střídaní téměř periodických úseků (tzv. kvaziperiodicita) s frekvencí F_0 a s poměrně velkou amplitudou, šumových úseků s podstatně nižší amplitudou a úseků ticha s téměř nulovou amplitudou.

Frekvenční oblast je vhodná k popisu spektrálních vlastností řeči. K výpočtu spektrálních charakteristik se používá Fourierova transformace (viz. kapitola 4.2). Spektrální povaha signálu se zobrazuje pomocí tzv. amplitudového spektra, které znázorňuje vývoj amplitudy signálu ve frekvenci. Znělé úseky řeči v něm vystupují jako úzké spektrální vrcholky, střední frekvence těchto vrcholků jsou přitom v harmonickém vztahu s frekvencí základního tónu. Na druhou stranu spektra neznělých zvuků jsou v podstatě stochastická. Celkový tvar spekter obou typů zvuků, tzv. **spektrální obálka**, pak vykazuje široké vrcholky a údolí, které odpovídají jednotlivým formantům a antiformantům hlasového traktu. Vývoj jejich středních frekvencí a šířek pásem v čase udává zabarvení hlasu odpovídajících zvuků – **témbr**. Je zřejmé, že u znělých zvuků je většina akustické energie soustředěna na nižších frekvencích (s velmi přibližným rozdělením rozdělením jednoho formantu na jeden kilohertz šířky frekvenčního pásma). Naproti tomu u neznělých zvuků převládají vysokofrekvenční složky.

Spektrogramy zobrazují časové a spektrální vlastnosti řeči. Představují časově frekvenční reprezentaci řečového signálu. Spektrogramy zobrazují pouze frekvenční amplitudu, informace o bázi je, stejně jako u amplitudových spekter, ignorována. Rozlišení detailů v čase nebo frekvenci (spektru) je omezeno obecně platným principem neurčitosti, který dává do souvislosti přesnost zachycení detailů v časové a frekvenční oblasti. Podle toho, v jaké oblasti chceme detaily zachytit, se používají dva typy spektrogramů: širokopásmové a úzkopásmové (viz. obr.č.3), které se liší délkou tzv. váhového okénka (viz. kapitola 3.4) použitého při jejich výpočtu. Širokopásmové spektrogramy

používají krátká váhová okénka (asi 5-10 ms). Zobrazují detaily v časové oblasti, podle principu neurčitosti však nejsou schopny zobrazit příliš jemné frekvenční detaily. Spíše než vlastní frekvence zvýrazňují spektrální obálku řečového signálu, a tak lze pomocí nich sledovat vývoj jednotlivých formantů v čase. Znělé úseky jsou na obr.č.3 zobrazeny jako posloupnosti různě tmavých vertikálních proužků. Úzkopásmové spektrogramy využívají delší váhové okénko (kolem 30 ms). Zobrazují detaily ve frekvenční oblasti – zvýrazňují spíše jemnou spektrální strukturu, a tak z nich lze vyčíst jednotlivé harmonické frekvence odpovídající frekvenci základního hlasivkového tónu, které se ve spektrogramu zobrazují jako horizontální proužky. Na druhou stranu ale neposkytují podrobné rozlišení v časové oblasti, nejsou tedy vhodné například k podrobné spektrální analýze rychle se měnících zvuků řeči, jako jsou plozivy apod. Pokud nechceme, aby se ve spektrogramech objevovaly parazitní vertikální nebo horizontální čáry, můžeme použít kompromis. Váhové okénko bývá veliké přibližně 2-3násobku lokální periody základního tónu[1].



Obrázek č. 3 – úzkopásmový (nahoře) a širokopásmový spektrogram[2]

3 Předzpracování řečového signálu

Cílem této kapitoly je představit základy předzpracování řeči, které se budou později využívat při teoretickém rozboru jednotlivých metod i jejich implementaci.

3.1 Digitalizace

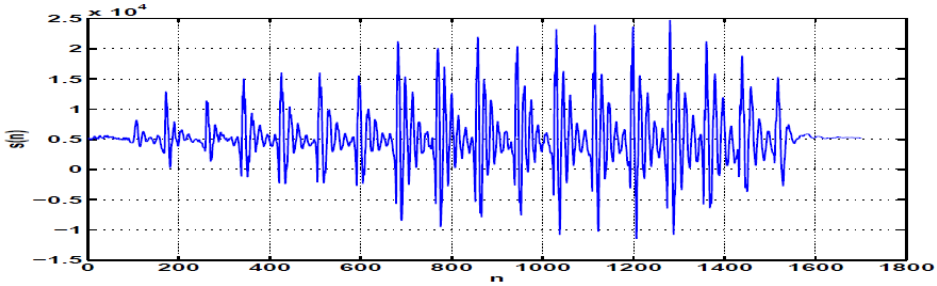
Abychom mohli akustický signál zpracovat počítačem, je nutné jej nejprve převést z analogového tvaru na číslicový. Tento převod se nazývá digitalizace a jeho výsledkem je vyjádření spojitého analogového signálu posloupností čísel, které nabývají hodnot ze zvoleného intervalu. Tohoto lze dosáhnout pomocí vzorkování a kvantizace. Celý proces je v počítači prováděn v analogově-digitálním převodníku, který je obsažen ve zvukové kartě.

3.2 Ustředění

Stejnosměrná složka (dc-offset) nenese žádnou užitečnou informaci, naopak může být rušivá (např. výpočet energie). Proto je vhodné ji odstranit odečtením střední hodnoty. V našem případě se bude pracovat s celým signálem, takže střední hodnota se spočítá průměrováním po ukončení signálu (offline střední hodnota).

$$s = \frac{1}{N}\sum_{i=1}^{N} s[n] \tag{1}$$

Lze vypočítat střední hodnotu i během nahrávání signálu (online střední hodnota), ale protože to v mé práci není použito, nebudu se jí zabývat. Více informací v [1].



Obrázek č.4 – neustředěný signál[2]

3.3 Segmentace

Řečový signál považujeme za náhodný a pro metody odhadu parametrů by měl být stacionární. Signál proto dělíme na kratší úseky, které nazýváme rámce (někdy též segmenty, mikrosegmenty, frames), kde by signál již měl být stacionární. Rámec má 3 parametry. Jsou to délka (length) l_{ram} , překrytí (overlap) p_{ram} a posun (frame shift) $s_{ram} = l_{ram} - p_{ram}$. [2]

Délka rámců by měla být dostatečně malá, aby bylo možno považovat danou část signálu za stacionární, ale na druhou stranu by měla být i dostatečně velká na to, aby bylo možné dostatečně přesně odhadnout požadované parametry. Délka rámců je typicky 20-25 ms (odpovídá 160-200

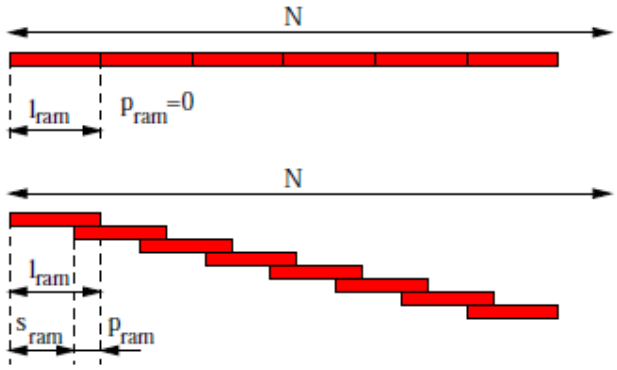
vzorkům pro vzorkovací frekvenci F_s = 8000). Délka překrytí bývá typicky 10 ms (tedy 100 rámců za sekundu).[2]

U rámců bez překrytí (vhodné pro kódování) dostaneme jejich počet jednoduchým vztahem:

$$N_{ram} = \left\lfloor \frac{N}{l_{ram}} \right\rfloor \tag{2}$$

U rámců s překrytím (běžné pro rozpoznávání) je počet rámců dán vztahem:

$$N_{ram} = 1 + \left\lfloor \frac{N - l_{ram}}{s_{ram}} \right\rfloor \tag{3}$$

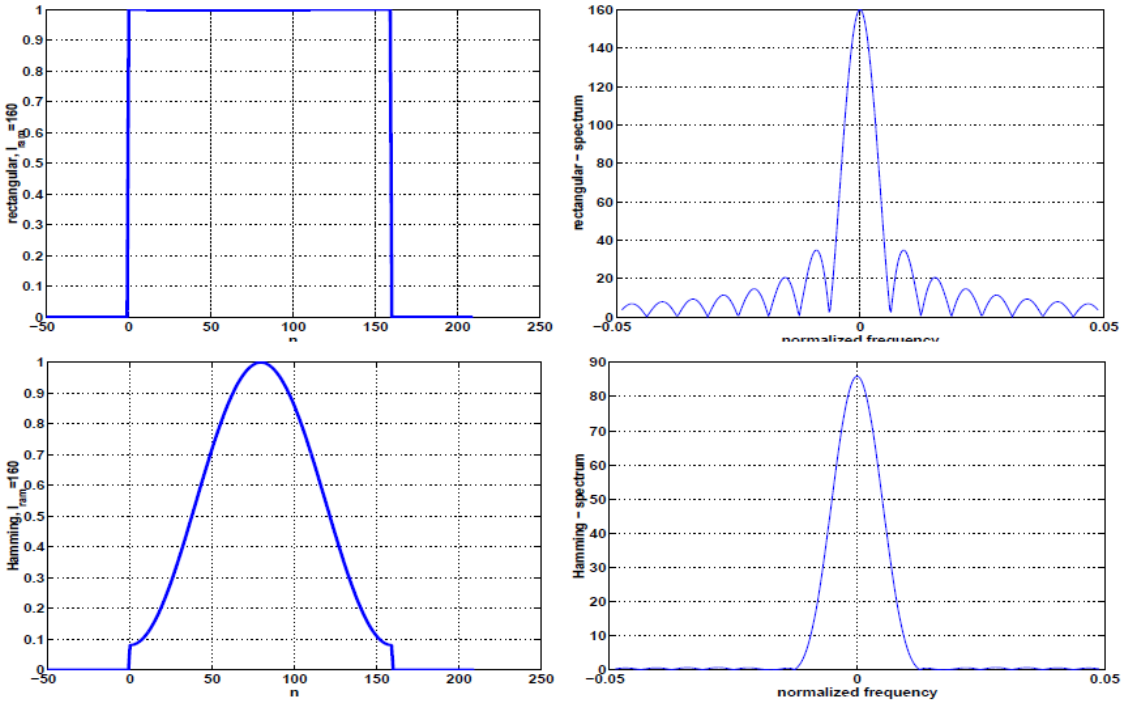


Obrázek č.5 – Rámce bez překrytí a s překrytím[2]

3.4 Váhování oknem

Předpokládáme, že zvukový signál v okolí rámce je periodický s periodou uvnitř rámce. Není-li perioda shodná s délkou rámce, popř. mají-li rámce nulové překrytí, můžeme se dopustit chyby ve zpracování. Proto se pro „vykrojení“ rámce se používá „okno“ (windowing function). Účelem použití okna je vybrat vzorky signálu v analyzovaném rámci a vyhladit jejich průběh přidělením váhy. Existuje několik typů, ale pro ilustraci vybírám pouze dvě nejpoužívanější.

- Pravoúhlé (rectangular) okno – se signálem neudělá nic.
- Hammingovo okno – je nejpoužívanější, utlumí signál na okrajích.



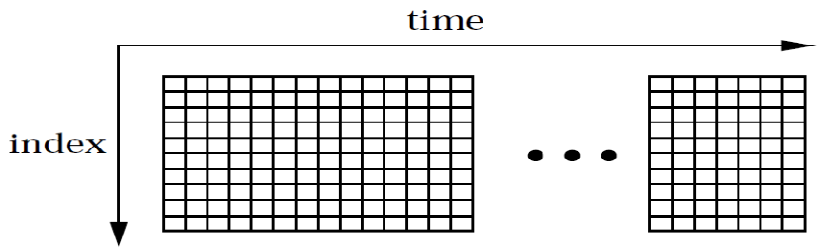
Obrázek č. 6 – srovnání pravoúhlého a hammingova okna [2]

4 Parametrizace

Pomocí parametrizace (feature extraction) popisujeme řečový signál jako omezené množství hodnot. Metody popisu jsou rozděleny na

- Neparametrický popis, který je založen pouze na poznatcích o zpracování signálu (např. Fourierova transformace, banky filtrů)
- Parametrický popis, který je založen na poznatcích o tvorbě řeči

Druhá technika ovšem používá mnoho technik neparametrického popisu, takže tyto dvě skupiny není snadné (a někdy ani žádoucí) oddělit. Vypočtené hodnoty se v obou případech nazývají parametry.



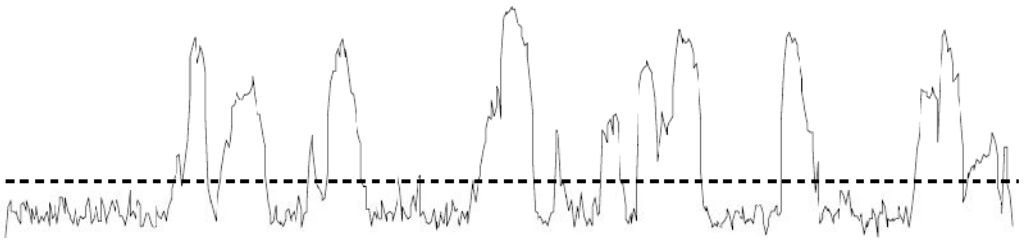
Obrázek č.7 – sada vektorů parametrů[2]

4.1 Zpracování v časové oblasti

Parametry, které získáme z časového průběhu řeči, vyžadují nejmenší nároky na výpočet. Bohužel z nich lze získat pouze základní informace. Dají se využít například při rozhodování, zda se jedná o řeč nebo šum, zda je rámec znělý nebo neznělý. Lze je také využít ve spojení s příznaky vypočtenými z frekvenční oblasti. Mezi parametry získané z časového průběhu patří například střední krátkodobá energie a počet průchodů nulou. Parametr, který nás zajímá, je střední krátkodobá energie, která je použita v detektoru řečové aktivity, který ve vstupním signálu objeví jeho část, reprezentující slovo a ta část signálu se bude dále zpracovávat. Střední krátkodobou energii můžeme vypočítat následujícím vztahem:

$$E = \frac{1}{N}\sum_{k=1}^{N} x(k)^2 \tag{4}$$

Vzhledem k tomu, že se při výpočtu používá druhá mocnina, bývá tento parametr navíc logaritmován. Více o zpracování v časové oblasti v [1].



Obrázek č.8 – detektor řečové aktivity založený na energii [4]

4.2 Zpracování ve frekvenční oblasti

Podobně jako v časové oblasti lze v jednom rámci považovat za konstantní i spektrální charakteristiky řeči. Hovoříme o krátkodobé spektrální analýze. Ve frekvenční oblasti je mluvená řeč reprezentována zastoupením jednotlivých frekvencí, neboli svým spektrem. Většina metod zpracování ve frekvenční oblasti využívá Fourierovu transformaci. Protože v našem případě pracujeme s navzorkovaným signálem s N vzorky, využijeme její diskrétní verzi, která je dána vztahem:

$$F_k = \sum_{n=0}^{N-1} s(n).e^{-j\frac{2\pi kn}{N}} \tag{5}$$

Samotné spektrum se ale pro klasifikaci rámců nepoužívá. Problémem je, že spektrum, získané pomocí diskrétní Fourierovy transformace, stále obsahuje velké množství redundantních informací, obsahuje v sobě buzení hlasového traktu a je korelované (tzn. jednotlivé frekvence jsou na sobě závislé). Je proto potřeba spektrum zpracovat dále. Je obvykle rozloženo do několika pásem, kde v každém z nich je zpracováno samostatně a je vypočítáno tzv. *cepstrum,* které můžeme popsat vztahem:

$$c(n) = DFT^{-1}\left\{\ln\left|DFT\left(s(n)\right)\right|^2\right\} \tag{6}$$

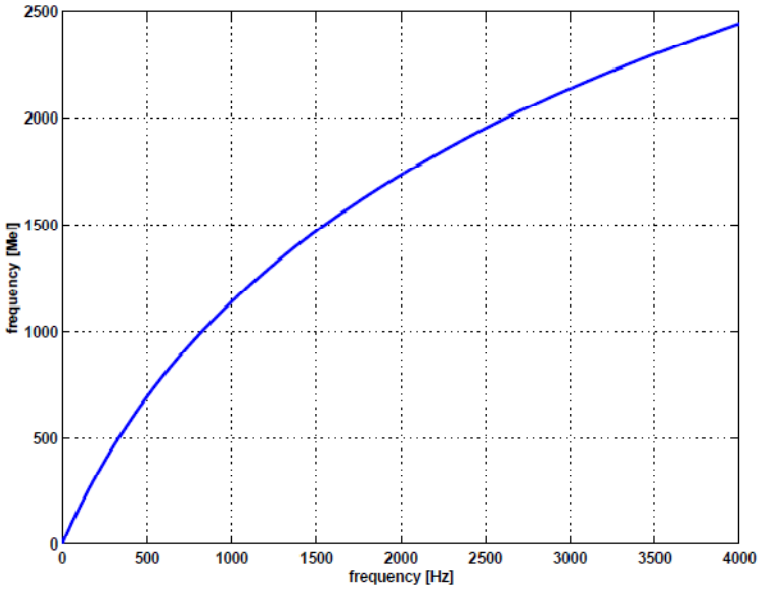
kde *c(n)* jsou vypočítané cepstrální koeficienty, *DFT* je diskrétní Fourierova transformace, DFT^{-1} je inverzní diskrétní Fourierova transformace a *s(n)* jsou vstupní vzorky.

Cepstrální analýza nám umožňuje ze signálu řeči oddělit nežádoucí parametry buzení a hlasového ústrojí. Nejběžnějšími metodami parametrizace jsou LPCC (Linear prediction cepstral coefficients) nebo MFCC (Mel frequency cepstral coefficients). V mé práci využívám pouze parametrizaci MFCC, která je popsána podrobněji v následující kapitole. O parametrizaci LPCC je možné vyhledat informace např. v [1].

4.2.1 Mel-frekvenční cepstrální koeficienty (MFCC)

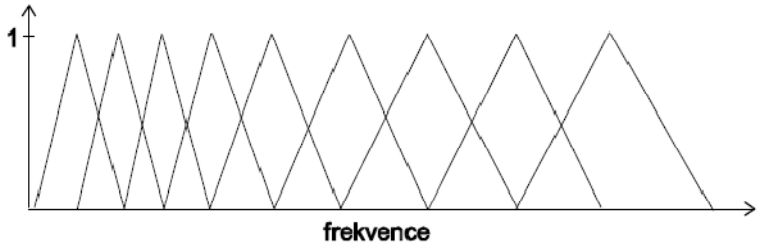
Zpracování pomocí MFCC je navrženo tak, aby do jisté míry respektovalo nelineární vlastnosti vnímání zvuků lidským uchem. Snaží se kompenzovat zejména nelineární vnímání frekvencí, a to s využitím banky trojúhelníkových filtrů s lineárním rozložením frekvencí v tzv. melovské frekvenční škále, jež je definována vztahem:

$$f_m = 2595\ \log_{10}\left(1 + \frac{f}{700}\right) \tag{7}$$



Obrázek č. 9 – Nelineární převod Hertzů na Mely[2]

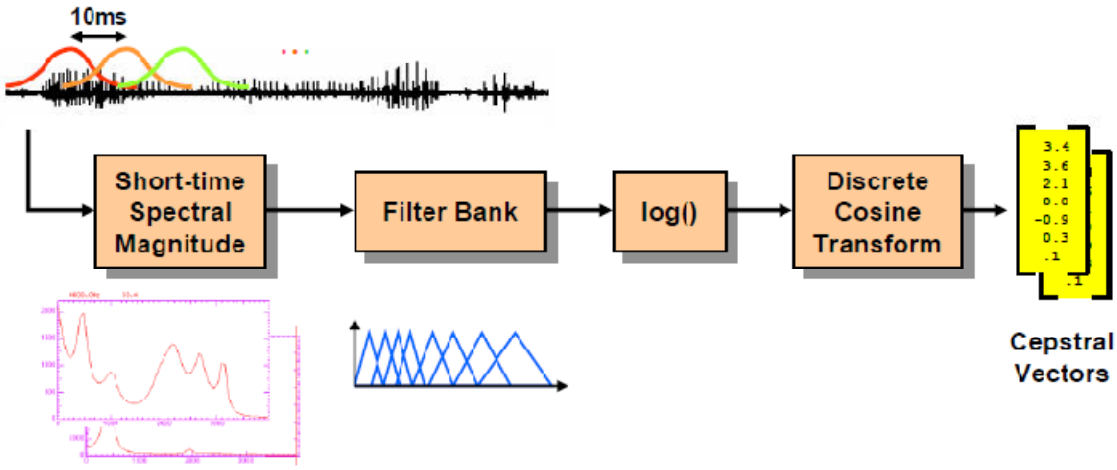
Kde *f*[Hz] je frekvence v lineární škále a f_m je frekvence v nelineární melovské škále a z každého pásma je vypočítána energie. Banka filtrů v melovském měřítku je označena na následujícím obrázku.



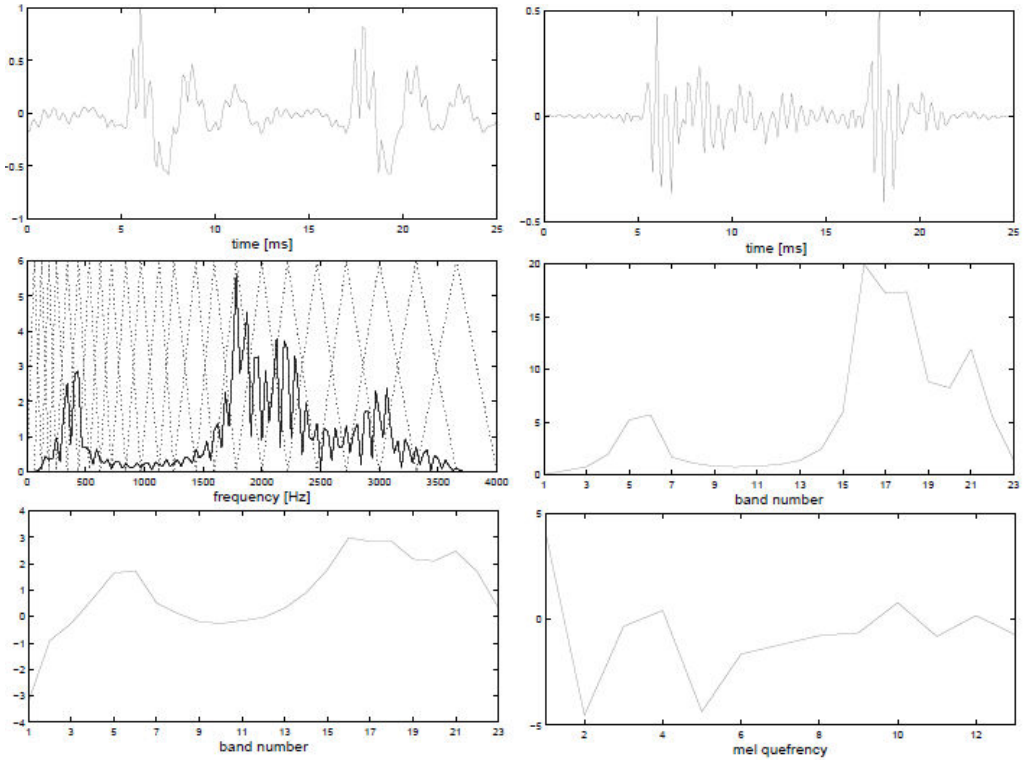
Obrázek č.10 – Banka melovských trojúhelníkových filtrů

Následně jsou energie získané z jednotlivých pásem zlogaritmovány a vektor energií je transformován do cepstrální oblasti. Pro výpočet cepstrálních koeficientů se místo Fourierovy transformace požívá diskrétní cosinova transformace. Celý proces je znázorněn na obrázku č. 11. Definice diskrétní cosinovy transformace je uvedena zde:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi . i}{N}(j - 0.5)\right) \tag{8}$$



Obrázek č.11 – Postup získání MFCC[9]



Obrázek č. 12 – Demonstrace MFCC[9]

Pro popis rámce řeči vypočítáme obvykle 13 koeficientů. Pro zvýšení kvality popisu rámce řeči se obvykle přidává první a druhá časová derivace každého koeficientu, neboli tzv. Δ a ΔΔ-koeficienty. Δ-koeficienty lze vypočítat pomocí vztahu:

$$d_t = \frac{\sum_{i=1}^{D}(c_{t+i} - c_{t-i})}{2\sum_{i=1}^{D} i^2} \tag{9}$$

kde D je tzv. *delta okno* a většinou se volí délky 2. Na začátku výpočtu bývají chybějící vektory nahrazeny prvním a při ukončení posledním. Pokud na získané Δ-koeficienty aplikujeme vzorec ještě jednou, získáme ΔΔ-koeficienty. Výsledný vektor potom obsahuje 39 koeficientů.

5 Určování frekvence základního tónu

5.1 Frekvence základního tónu

Frekvence základního tónu (pitch) F_0 je základním kmitočtem, na kterém kmitají hlasivky (viz. kapitola 2.1.2). Využívá se například u syntezátorů řeči (generování melodie) a kódování. Periodu základního tónu (pitch period) můžeme vypočítat jako převrácenou hodnotu frekvence.

$$T_0 = \frac{1}{F_0} \tag{10}$$

Dalším důležitým pojmem je „lag“, který označuje periodu základního tónu vyjádřenou ve vzorcích. Výpočet lagu bude ve tvaru

$$L = F_0 F_s \tag{11}$$

5.1.1.1 Charakteristiky základního tónu

- F_0 může nabývat hodnot od 50 Hz (muži) až do 400 Hz (děti). Při F_s = 8000 Hz tyto frekvence odpovídají lagům L = 160 až 20 vzorků. Je patrné, že při malých hodnotách F_0 se blížíme délkám běžně používaných oken (20 ms, což odpovídá 160-ti vzorkům).
- Frekvence základního tónu je různá pro různé řečníky a může se i pro jednotlivce měnit při promluvách v poměru až 2:1.
- Pro různé hlásky mívá základní tón typické průběhy, malé změny po prvním kmitu (delta F_0 < 10 Hz) charakterizují mluvčího, ale obtížně se zjišťují. V radiotechnice se takovým posuvům říká „jitter“.
- Základní tón je ovlivněn spoustou faktorů, např. větnou melodií, náladou, únavou, atd. Velikosti změn jsou větší u profesionálních mluvčích. Zejména díky většímu „modulování“ hlasu.

5.1.1.2 Problémy při určování základního tónu

- Ani znělé hlásky nejsou zcela periodické. Čistě periodický může být pouze velmi čistý zpěv. Při generování řeči s F_0 = konst. je výsledná řeč monotónní.
- Nevyskytuje se čistě znělé nebo neznělé buzení. Většinou je buzení smíšené (šum na vyšších frekvencích).
- Vysoký základní tón může být ovlivněn nízkým formantem F_1 (ženy, děti).
- Při přenosu řeči v telefonním pásmu (300-3400 Hz) máme k dispozici pouze násobky (vyšší harmonické) základního tónu. Filtrace za účelem získání F_0 by tedy k ničemu nevedla.

5.1.1.3 Metody pro určování základního tónu

Existuje řada metod. Základní tón můžeme určit například pomocí autokorelační funkce, využitím prediktoru chyby lineární predikce nebo cepstrální metodou. V mé aplikaci demonstrující určení základního tónu jsem zvolil metodu využívající autokorelační funkci.

5.2 Metoda využívající autokorelační funkci

5.2.1 Autokorelační funkce (ACF)

Odhad periody základního tónu v časové oblasti je většinou založen na využití ACF (autocorrelation function). Tato funkce má některé užitečné vlastnosti, které indikují jak energetické, tak frekvenční charakteristiky řečového signálu. Pokud má tato funkce obsahovat informace o velikosti periody základního tónu, musí být délka okénka při zpracování signálu větší než jedna perioda základního tónu, tj. alespoň L = 20 - 40 ms. Uvážíme-li pouze jeden segment zpracovávaného signálu a pravoúhlé okénko, lze funkci zapsat ve tvaru

$$R(m) = \sum_{n=0}^{L-1-m} s(n)\, s(n+m) \tag{12}$$

a periodu základního tónu bychom pak mohli hledat pro takové m* > 0, které vyhovuje rovnici

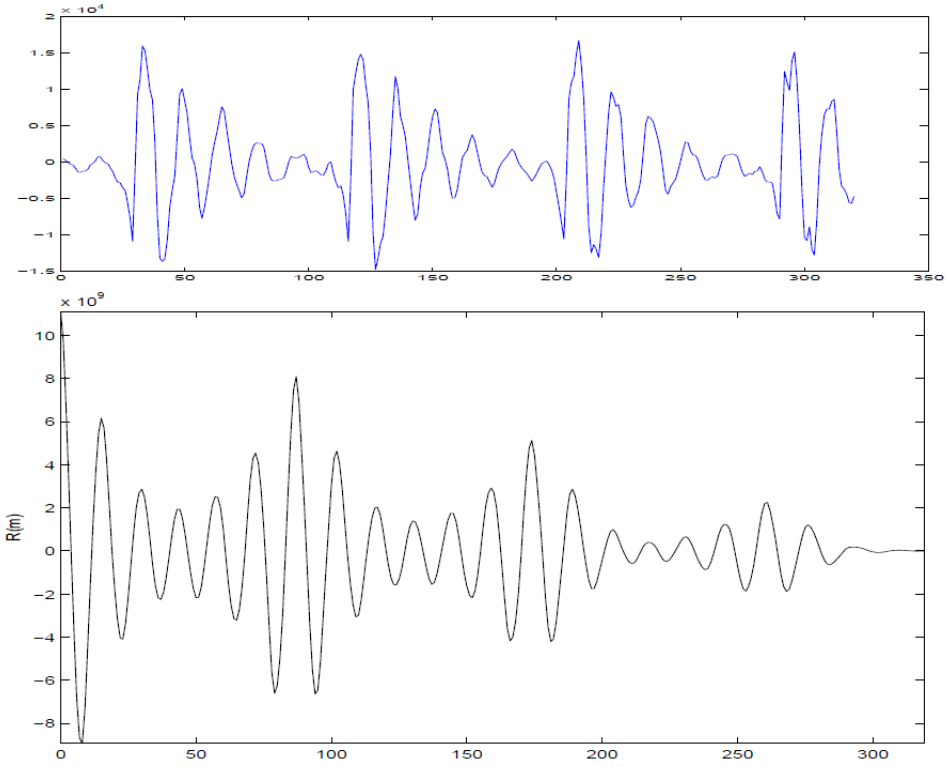
$$m^* = \underset{m}{\operatorname{argmax}}\, R(m) = \underset{m}{\operatorname{argmax}} \sum_{n=0}^{L-1-m} s(n)\, s(n+m) \tag{13}$$

Je zřejmé, že ACF obsahuje informaci o periodě základního tónu (maximum ACF), ale její průběh vykazuje také mnoho dalších vrcholů. Tyto vrcholy jsou způsobeny formantovou strukturou a mohou se stát snadno zdrojem chyb. Sondhi a Rabiner navrhli již před mnoha lety velmi účinné postupy nelineárního předzpracování, které mají za cíl potlačit formantovou strukturu využitím ACF aplikované na řečový signál, jenž byl předtím centrálně, popř. centrálně a amplitudově, omezen[1]. Vstupně-výstupní funkci centrálního omezovače lze vyjádřit vztahem

$$c_1\big(s(k)\big) = \begin{cases} s(k) - h_L & pro\ s(k) > h_L \\ 0 & pro\ |s(k)| \le h_L \\ s(k) + h_L & pro\ s(k) < -h_L \end{cases} \tag{14}$$

A pro centrální a amplitudový omezovač lze obdobnou funkci vyjádřit vztahem

$$c_2\big(s(k)\big) = \begin{cases} 1 & pro\ s(k) > h_L \\ 0 & pro\ |s(k)| \le h_L \\ -1 & pro\ s(k) < -h_L \end{cases} \tag{15}$$



Obrázek č.13 – Rámec řeči a vypočítaná ACF [3]

5.2.2 Výpočet lagu a určení znělosti

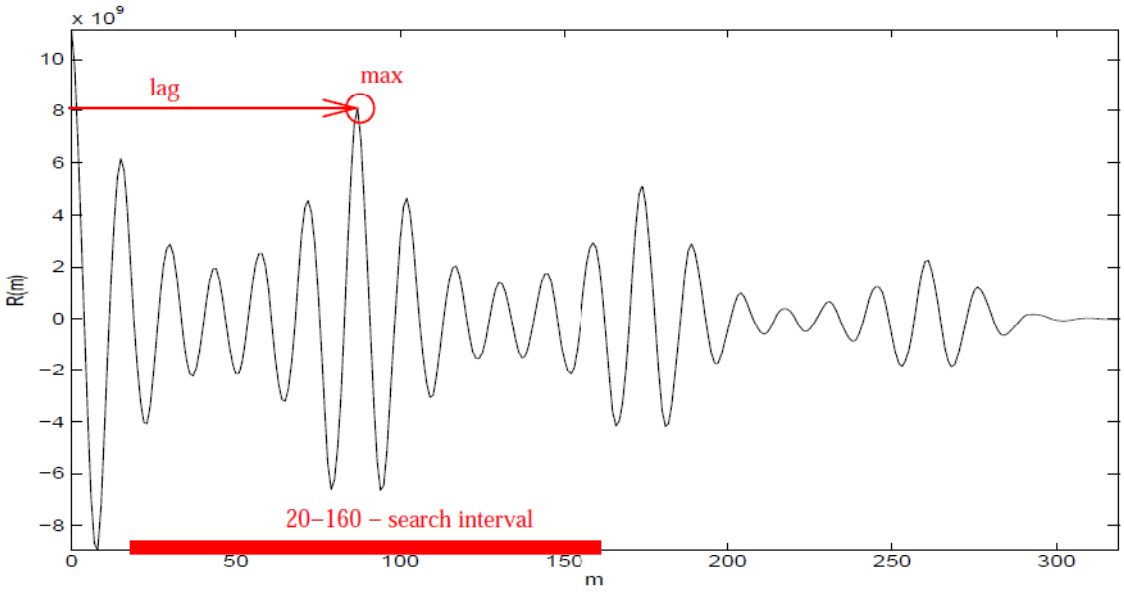
Autokorelační funkce je normalizována a je určeno její maximum pro m = 20 až m = 160. Jestliže je řečový rámec klasifikován jako znělý, tak podle polohy maxima je stanovena velikost frekvence základního tónu (pro m = 20 až 160 leží odpovídající hodnoty v pásmu F_0 = 50 až 400 Hz).

Znělost rámce můžeme odhadnout porovnáním nalezeného maxima s nultým (maximálním) autokorelačním koeficientem. Konstanta α se musí zvolit experimentálně.

$$R_{max} < \alpha R(0) \rightarrow neznělý \quad (16)$$

$$R_{max} \geq \alpha R(0) \rightarrow znělý \quad (17)$$

Nalezení maxima ACF je ilustrováno na následujícím obrázku, kde L = 87



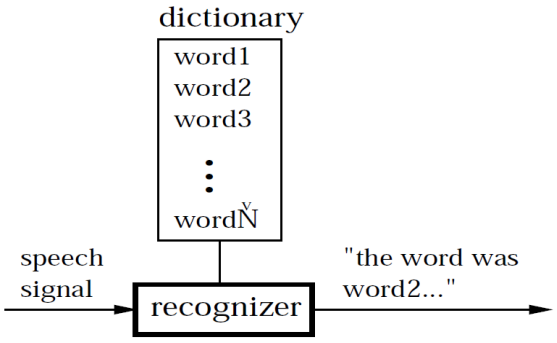
Obrázek č.14 – hledání maxima ACF u znělého rámce [3]

Hodnotu frekvence základního tónu pro L = 87 a vzorkovací frekvenci F_s = 8000 vypočítáme následovně

$$F_0 = \frac{F_s}{L} = \frac{8000}{87} \doteq 92\ Hz$$

6 Rozpoznávání izolovaných slov

Tato kapitola se zabývá rozpoznáváním izolovaných slov. Rozpoznávač má za úkol klasifikovat příchozí matici parametrů **O** = [o(1),…,o(T)] jako jedno ze slov $w_1 \ldots w_{\check{N}}$, kde Ň je počet slov.



Obrázek č.15 – Princip rozpoznávání izolovaných slov[4]

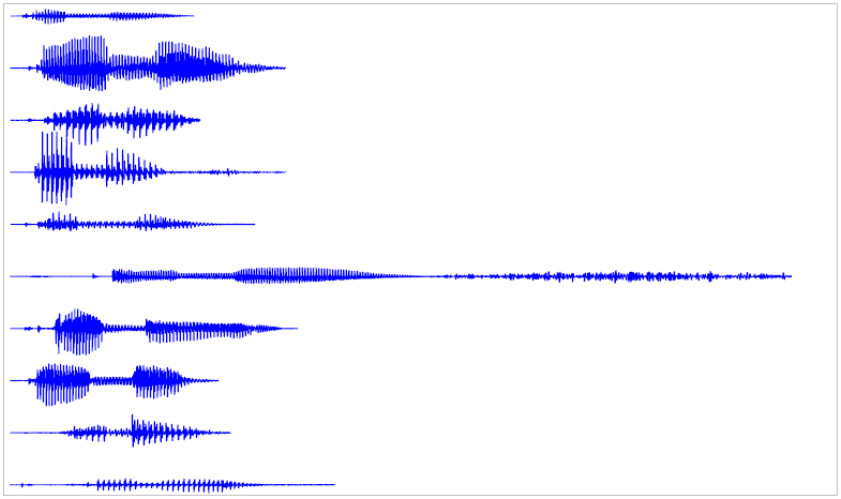
V následujících kapitolách budou popsány dvě metody rozpoznávání. Metoda využívající dynamické borcení času (viz. kapitola 6.1) a metoda využívající skryté Markovovy modely (viz. kapitola 6.2).

Při rozpoznávání je nejprve důležité izolovaná slova najít:

- Push-to-talk (zmáčkni a mluv)
- Detektor řečové aktivity

6.1 Dynamické borcení času (DTW)

Metoda pracuje na principu **porovnávání se vzory.** Slovo je zde zpracováno jako celek, přičemž je klasifikováno do té třídy (třídy jsou tvořeny jednotlivými slovy ve slovníku), k jejímuž **vzorovému obrazu** (vzorovému slovu reprezentovanému posloupností příznakových vektorů) má nejmenší vzdálenost. Při určování vzdálenosti se hledá taková nelineární transformace časové osy jednoho z obrazů, při níž dojde k porovnání obou obrazů s nejmenší výslednou vzdáleností. Uvedený mechanismus vyplynul z důkladného rozboru signálu získaného vyslovením stejného slova několikrát týmž řečníkem. Při tomto rozboru se zjistilo, že základní odlišnosti mezi odpovídajícími signály nejsou ve spektrální oblasti, ale v časovém členění, tj. v nestejné délce slov a zejména v nepoměru mezi délkami odpovídajících částí (fonémů, hlásek) uvnitř slova. Algoritmus pracuje s efektem nelineární časové normalizace, přičemž kolísání v časové ose je modelováno časově nelineární „bortivou“ funkcí s přesně specifikovanými vlastnostmi. Časové rozdíly mezi dvěma řečovými obrazy jsou přitom eliminovány „borcením“ jedné z časových os takovým způsobem, že je dosaženo maximální shody s druhým obrazem[1].



Obrázek č.16 – ukázka rozdílného časování při vyslovení stejného slova[4]

Ve slovníku se nacházejí referenční matice parametrů pro slova, která chceme rozpoznávat

$$\mathbf{R}_1 \ldots \mathbf{R}_N$$

Na vstup rozpoznávače přijde testovací matice parametrů O. Cílem je určit, ke kterému referenčnímu slovu test patří. Pokud by měla slova pouze jeden vektor, bylo by řešení relativně jednoduché:

$$d(\mathbf{o}, \mathbf{r}_i) = \sqrt{\sum_{k=1}^{P} |o(k) - r_i(k)|^2} \tag{18}$$

Nakonec by došlo k vybrání minimální vzdálenosti.

Ve skutečnosti slova jeden vektor nemají. Je potřeba určit vzdálenost (či podobnost) referenční sekvence vektorů o délce *R*:

$$\mathbf{R} = [\mathbf{r}(1), \ldots, \mathbf{r}(R)]$$

a testovací sekvence vektorů o délce *T*:

$$\mathbf{O} = [\mathbf{o}(1), \ldots, \mathbf{o}(T)]$$

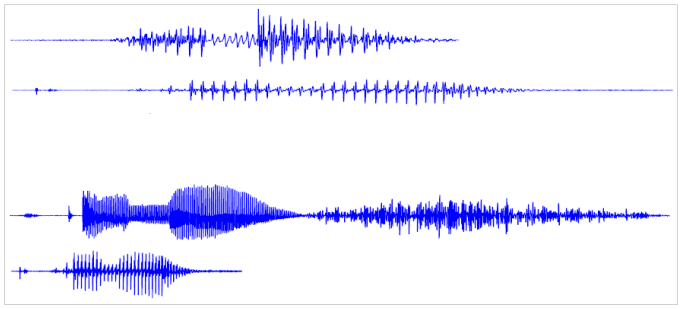
Slova **nejsou nikdy stejně dlouhá**, takže platí

$$R \neq T$$

Můžeme provést lineární srovnání

$$D(\mathbf{O}, \mathbf{R}) = \sum_{i=1}^{R} d[\mathbf{o}(w(i)), \mathbf{r}(i)] \tag{19}$$

kde *w(i)* je definována tak, aby srovnání bylo lineární. Tato technika je velmi nespolehlivá a snadno může dojít k chybě. Následující obrázek ilustruje 2 případy. V 1. případě by výsledek byl funkční, ale v 2. případě ne.



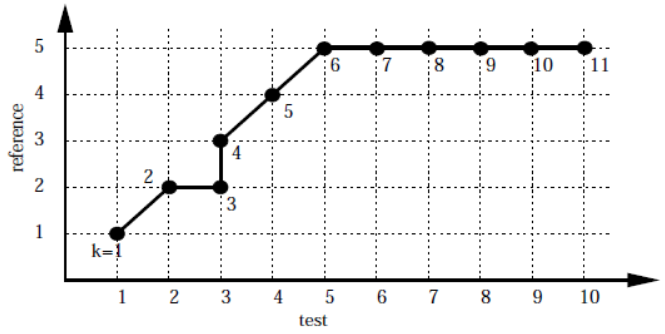
Obrázek č.17 – vhodný a nevhodný signál pro lineární srovnání[4]

Podstatně lepším řešením je, když je srovnání řízeno přímo vzdálenostmi jednotlivých vektorů (Dynamické borcení času).

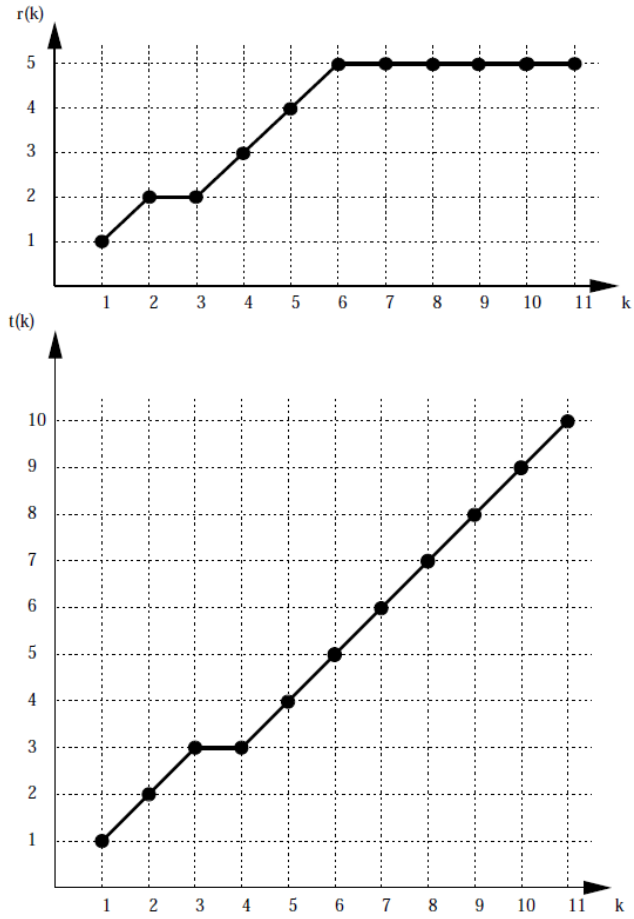
Definujeme obecnou časovou proměnnou *k* a zavedeme dvě transformační funkce:

- *r(k)* pro referenční sekvenci
- *t(k)* pro testovací sekvenci

Poté je možné srovnání jednotlivých vektorů zakreslit pomocí **cesty** (viz. obr. 18)**.** Počet kroků cesty označíme jako K. Referenci budeme zobrazovat na svislé ose, test na vodorovné. Z této cesty můžeme odvodit průběh funkcí *r(k)* a *t(k)*, které „krokují“ jednotlivé sekvence (viz. obr. 19).



Obrázek č.18 – srovnání vektorů zakreslené pomocí cesty[4]



Obrázek č.19 – odvozený průběh funkcí *r(k)* a *t(k)* [4]

Cesta *C* je jednoznačně dána svou délkou K_C a průběhem funkcí $r_C(k)$ a $t_C(k)$. Pro tuto cestu je vzdálenost sekvencí **O** a **R** dána jako:

$$D_C(\mathbf{O},\mathbf{R}) = \frac{\sum_{k=1}^{K_C} d\left[\mathbf{o}\left(t_C(k)\right),\mathbf{r}\left(r_C(k)\right)\right] W_C(k)}{N_C} \tag{20}$$

kde $d[\mathbf{o}(\cdot), \mathbf{r}(\cdot)]$ je vzdálenost dvou vektorů, $W_C(k)$ je váha odpovídající *k*-tému kroku cesty a N_C je normalizační faktor závislý na vahách.

Vzdálenost sekvencí **O** a **R** je dána jako minimální vzdálenost přes soubor všech možných cest (všechny možné délky, všechny možné průběhy)

$$D(\boldsymbol{O},\boldsymbol{R}) = \min_{\{C\}} D_C(\mathbf{O},\mathbf{R}) \tag{21}$$

Je potřeba vyřešit 3 důležité věci:

1. průběhy funkcí *r(k)* a *t(k)* musejí být přípustné. Nesmí docházet k situacím, že by se cesta vracela zpět, „skákala“ přes několik vektorů, atd.
2. musí se definovat váhovací funkce a normalizační faktor.
3. je žádoucí vyrobit takový algoritmus, který vypočítá *D*(**O**, **R**) co nejrychleji.

Cesta má následující omezení:

1. Počáteční a koncové body

$$\left.\begin{matrix} r(1)=1 \\ t(1)=1 \end{matrix}\right\} \text{začátek} \qquad \left.\begin{matrix} r(K)=R \\ t(K)=T \end{matrix}\right\} \text{konec} \tag{22}$$

2. Lokální souvislost a lokální strmost

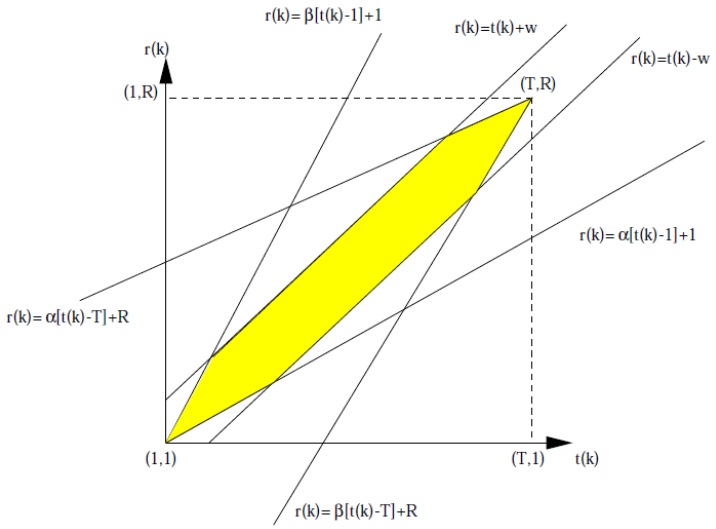
$$\begin{matrix} 0 \leq r(k) - r(k-1) \leq R^{\star} \\ 0 \leq t(k) - t(k-1) \leq T^{\star} \end{matrix} \tag{23}$$

   v praxi se volí $R^{\star}$, $T^{\star}$ = 1, 2, 3 (úplně nejčastěji 1).
   - $R^{\star}, T^{\star} = 1$: Každý vektor se bere alespoň 1x! *r(k) = r(k – 1)* znamená, že se opakuje.
   - $R^{\star}, T^{\star} > 1$: Vektor(y) se mohou přeskočit

3. Globální vymezení cesty
   Vymezení přípustné oblasti pomocí přímek:

$$\begin{matrix} 1 + \alpha[t(k)-1] \leq r(k) \leq 1 + \beta[t(k)-1] \\ R + \beta[t(k)-T] \leq r(k) \leq R + \alpha[t(k)-T] \end{matrix} \tag{24}$$

   Tyto cesty vymezují maximální „strmost“ nebo „placatost“ cesty DTW.



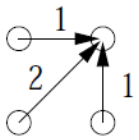
Obrázek č.20 – Globální vymezení cesty DTW pomocí přímek [4]

Definice váhových funkcí

Váhová funkce *W(k)* závisí na lokálním „posunu“ cesty. Existují 4 typy:

- Typ a) symetrická:

$$W_a(k) = [t(k) - t(k-1)] + [r(k) - r(k-1)] \tag{25}$$

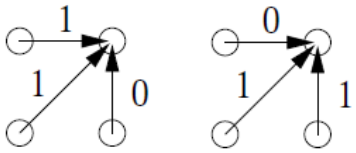


Obrázek č.21 – Symetrická váhová funkce [4]

- Typ b) asymetrická:

$$W_{b1}(k) = t(k) - t(k-1) \tag{26}$$

$$W_{b2}(k) = r(k) - r(k-1) \tag{27}$$



Obrázek č.22 – asymetrická váhová funkce W_{b1} *a* W_{b2} [4]

- Typ c)

$$W_c(k) = \min\{t(k) - t(k-1), r(k) - r(k-1)\} \tag{28}$$

- Typ d)

$$W_d(k) = \max\{t(k) - t(k-1), r(k) - r(k-1)\} \tag{29}$$

Normalizační faktor

Obecně lze normalizační faktor zapsat ve tvaru:

$$N = \sum_{k=1}^{K} W(k) \tag{30}$$

Pro chovací funkci typu a) je normalizační faktor:

$$N_a = \sum_{k=1}^{K} [t(k) - t(k-1) + r(k) - r(k-1)] = T + R \tag{31}$$

Pro váhovací funkci typu b1) je normalizační faktor $N = T$

Pro váhovací funkci typu b2) je normalizační faktor $N = R$

Pro typy c) a d) je faktor silně závislý na průběhu cesty. Je lepší použít konstantu: $N = T$

Efektivní výpočet *D(*O, R*)*

Výpočet minimální vzdálenosti je relativně jednoduchý, pokud normalizační faktor N_C není funkcí cesty a může se psát $N_C = N$ pro $\forall C$

Toto naštěstí většinou platí a tedy:

$$D(\mathbf{O}, \mathbf{R}) = \frac{1}{\mathrm{N}} \min_{\{C\}} \sum_{k=1}^{K_C} d\left[\mathbf{o}\left(t_C(k)\right), \mathbf{r}\left(r_C(k)\right)\right] W_C(k) \tag{32}$$

Postup bude následující:

1. Do mřížky **d** o velikosti $T \times R$ si zapíšeme vzdálenosti referenčních a testovacích vektorů, každý s každým

2. Definujeme mřížku **g** s částečnou kumulovanou vzdáleností. Oproti mřížce **d** má **g** navíc ještě nultý řádek a nultý sloupec, které inicializujeme na:

$$g(0,0) = 0 \quad \text{a} \quad g(O, m \neq 0) = g(n \neq 0, 0) = \infty$$

3. Částečnou kumulovanou vzdálenost spočítáme pro každý bod takto:

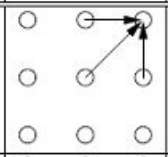
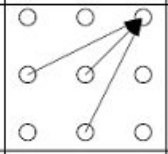
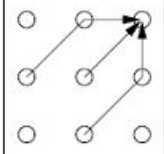
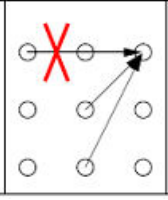
$$g(m,n) = \min_{\forall \text{předchůdci}} [g(\text{předchůdce}) + d(m,n)\, w(k)] \tag{33}$$

- Možní předchůdci jsou dáni pomocí tabulky lokálních omezení cesty
- Váha *w(k)* odpovídá pohybu z předchůdce do bodu [*m, n*].
- Vztahy pro výpočet částečné kumulované vzdálenosti jsou tabelovány v tabulce lokálních omezení cesty.

4. Konečná minimální normovaná vzdálenost je pak dána:

$$D(\mathbf{O}, \mathbf{R}) = \frac{1}{N}\, g(T,R) \tag{34}$$

Lokální omezení cesty

Typ DTW		α	β	Typ $w(k)$	$g(n,m)$
I.		0	∞	a	$\min \begin{Bmatrix} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-1,m)+d(n,m) \end{Bmatrix}$
				d	$\min \begin{Bmatrix} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m)+d(n,m) \end{Bmatrix}$
II.		$\frac{1}{2}$	2	a	$\min \begin{Bmatrix} g(n-1,m-2)+3d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+3d(n,m) \end{Bmatrix}$
				d	$\min \begin{Bmatrix} g(n-1,m-2)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-2,m-1)+d(n,m) \end{Bmatrix}$
III.		$\frac{1}{2}$	2	a	$\min \begin{Bmatrix} g(n-1,m-2)+2d(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+2d(n-1,m)+d(n,m) \end{Bmatrix}$
IV.		$\frac{1}{2}$	2	b1	$\min \begin{Bmatrix} g(n-1,m)+kd(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m-2)+d(n,m) \end{Bmatrix}$ kde $k=1$ pro $r(k-1) \neq r(k-2)$ $k=\infty$ pro $r(k-1)=r(k-2)$

Obrázek č.23 – Tabulka obsahující typy lokálních omezení a z nich vyplývající faktory α a β [4]

Trénování – tvorba referencí neboli vzorů

1. Nejjednodušší: ve třídě ω_r se nachází jen jedna reference $\mathbf{R}_r$.
2. Složitější: pro každou třídu ω_r existuje více referencí.
3. Vytvoření průměrného vzoru pro každou třídu ω_r
   - Lineární průměrování – je brán v potaz průměr vektorů lineárně srovnaných sekvencí

- Dynamické průměrování
    a) Nalezneme vzor s vyhovující délkou
    b) Průměrování se provádí na základě vektorů přiřazených k tomuto prvnímu vzoru pomocí cesty DTW

4. Vytváření vzorů shlukováním – shluky jsou tvořeny tak, aby si byly reference uvnitř shluků co nejvíce podobné a reference mezi shluky co nejméně podobné

Rozpoznávání (klasifikace)

Je-li každá třída reprezentována jen jednou referencí, je řešení jednoduché:

$$\omega_r^\star = \arg\min_r \mathrm{D}(\boldsymbol{O}, \mathbf{R}_r) \text{ pro } r = 1, \dots, N \tag{35}$$

Existuje-li pro každou třídu více referencí, je možné použít dvě metody

1. 1-NN (Nearest Neighbor) neboli nejbližší soused

$$\omega_r^\star = \arg\min_r \mathrm{D}(\boldsymbol{O}, \mathbf{R}_r) \quad \text{pro} \quad \begin{matrix} r = 1, \dots, N \\ i = 1, \dots, N_r \end{matrix} \tag{36}$$

2. *k*-NN (*k* Nearest Neighbors) neboli *k* nejbližších sousedů
    - pro každou třídu jsou vypočítány všechny vzdálenosti a seřazeny podle velikosti od nejlepší po nejhorší

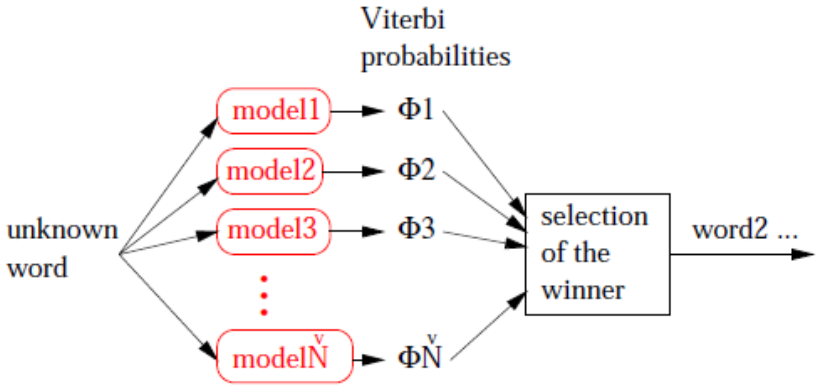
$$D\left(\mathbf{O}, \mathbf{R}_{r(1)}\right) \leq D\left(\mathbf{O}, \mathbf{R}_{r(2)}\right) \leq \cdots \leq D\left(\mathbf{O}, \mathbf{R}_{r(N_r)}\right) \tag{37}$$

    - klasifikaci **O** do třídy ω_r je poté rozhodnuto podle průměrné vzdálenosti *k* nejbližších sousedů:

$$\omega_r^\star = \arg\min_r \frac{1}{k} \sum_{i=1}^{k} \mathrm{D}\left(\boldsymbol{O}, \mathbf{R}_{r_{(i)}}\right) \tag{38}$$

6.2 Skryté Markovovy modely (HMM)

Nechceme-li používat referenční matice, musíme použít statistické modely jednotlivých rozpoznávaných slov. Struktura takového rozpoznávače je ilustrována na obrázku.



Obrázek č.24 – Rozpoznávač se statistickými modely slov

6.2.1 Stavební bloky rozpoznávače

Teorie statického rozpoznávání vzorů

Základní rovnice (Bayesův vztah) je následující:

$$P(\omega_j|\mathrm{x}) = \frac{p(\mathrm{x}|\omega_j)P(\omega_j)}{p(\mathrm{x})} \tag{39}$$

kde:

- $p(\mathrm{x}|\omega_j)$ je likelihood (česky „věrohodnost“, ale dále bude používán anglický výraz „likelihood“) datového vektoru x, známe-li třídu ω_j
- $P(\omega_j)$ je prior třídy ω_j
- $p(\mathrm{x})$ je evidence (normalizační funkce taková, aby $p(\mathrm{x}|\omega_j)$ byla slušná pravděpodobnost)

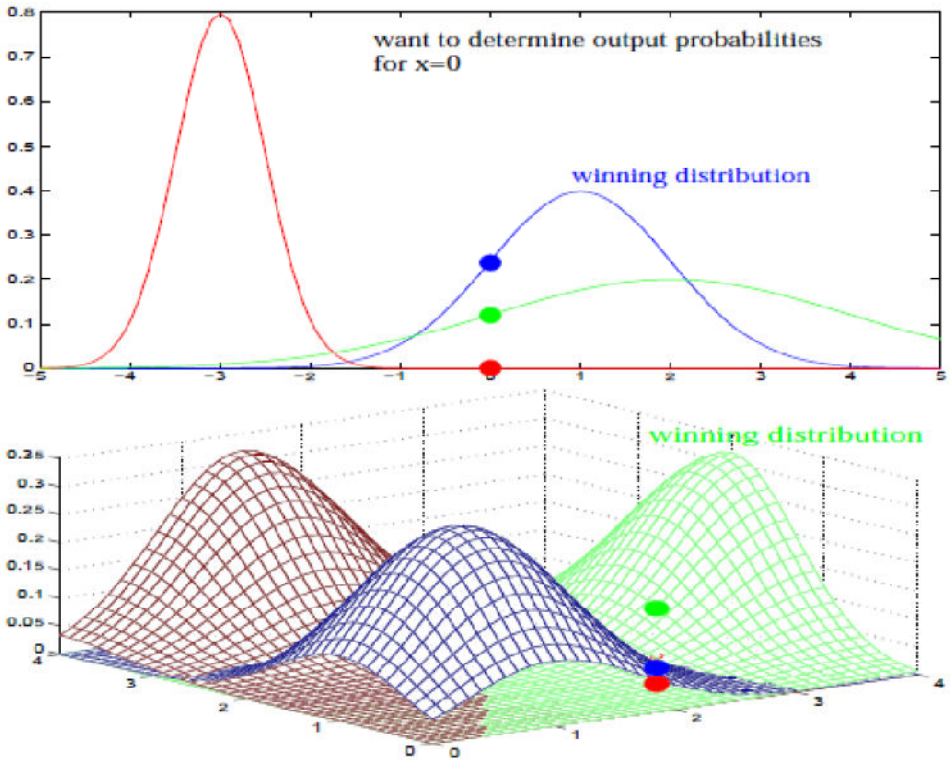
Na evidenci $p(\mathrm{x})$ se nehledí, protože je stejná pro všechny třídy a u jednoduchých rozpoznávačů se předpokládá, že priory všech tříd budou stejné $P(\omega_j) = \omega$. Proto pro nalezení rozpoznaného slova stačí hledat maximální likelihood:

$$\omega_j^{\star} = \max_j p(\mathrm{x}|\omega_j) \tag{40}$$

Modelování jednotlivých tříd

Pro rozpoznávání jednotlivých vektorů dokážeme třídy namodelovat Gaussovými rozděleními nebo dokonce jejich směsí:

$$p(\mathbf{x}|\omega_j) = \sum_{i=1}^{M} \alpha_{ij} N(\mathbf{x}; \boldsymbol{\mu}_{ji}, \Sigma_{ji}) \tag{41}$$



Obrázek č.25 – Ilustrace modelování jednotlivých tříd pomocí jednorozměrných a dvourozměrných Gaussových rozdělení [5]

Jejich parametry rozdělení **Θ** (míchací koeficienty) dokážeme odhadnout pomocí iterativního algoritmu EM:

1. Parametry na začátku nějak odhadneme (výběr vektoru nebo generování náhodného čísla)
2. S parametry $\boldsymbol{\Theta}^{(i-1)}$ se pro každé Gaussovo rozdělení spočítá „occupation count“:

$$\gamma_l = \frac{\alpha_l^{(i-1)} N(\mathbf{x}; \boldsymbol{\mu}_l^{(i-1)}, \Sigma_l^{(i-1)})}{\sum_{i=1}^{M} \alpha_i^{(i-1)} N(\mathbf{x}; \boldsymbol{\mu}_i^{(i-1)}, \Sigma_i^{(i-1)})} \tag{42}$$

3. Nové parametry jsou dány:

$$\alpha_l^{(i)} = \frac{1}{n} \sum_{k=1}^{n} \gamma_l(k) \tag{43}$$

$$\boldsymbol{\mu}_l^{(i)} = \frac{\sum_{k=1}^{n} \gamma_l(k) \mathbf{x}_k}{\sum_{k=1}^{n} \gamma_l(k)} \tag{44}$$

$$\Sigma_l^{(i)} = \frac{\sum_{k=1}^{n} \gamma_l(k) (\mathbf{x}_k - \boldsymbol{\mu}_l^{(i)})(\mathbf{x}_k - \boldsymbol{\mu}_l^{(i)})^T}{\sum_{k=1}^{n} \gamma_l(k)} \tag{45}$$

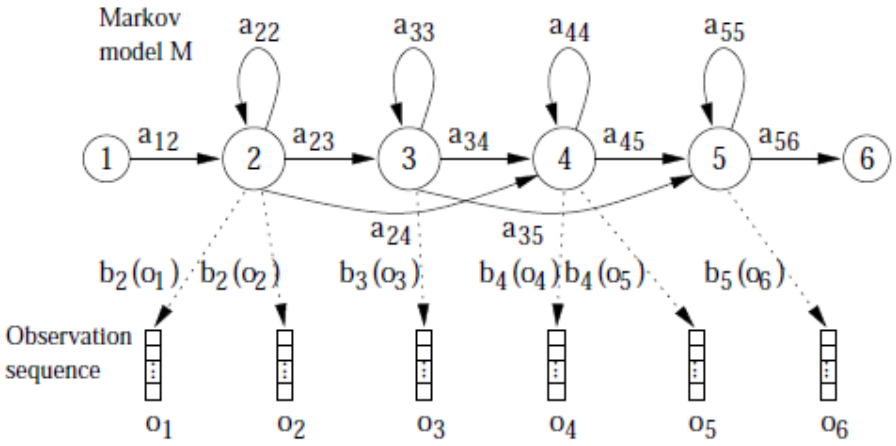
4. Opakují se body 2 a 3, dokud se nesnižuje celkový log-likelihood:

$$\ln p(D|\Theta) = \sum_{k=1}^{n} \ln \sum_{i=1}^{M} \alpha_{ij} N(\mathbf{x}_k; \boldsymbol{\mu}_{ji}, \Sigma_{ji}) \tag{46}$$

Popřípadě můžeme nastavit fixní počet operací.

6.2.2 Struktura skrytého Markovova modelu

Struktura modelu je znázorněna na obrázku č.26. Stavy vyznačené většími kruhy jsou **vysílací** (emitting), reprezentují tedy vždy nějaký vstupní vektor. Termín „vysílací“ je poněkud zavádějící, protože stavy ve skutečnosti žádné vektory nevysílají, ale produkují likelihoody, se kterými se dané vektory vyslaly. První a poslední stav jsou nevysílací. Používáme je jako konektory při napojování modelů. V této kapitole budeme uvažovat model pouze pro jednu třídu. Ostatní modely jsou podobné, ale s jinými parametry.



Obrázek č.26 – Struktura Skrytého Markovova modelu[6]

Přechodové pravděpodobnosti a_{ij}

Kvantifikují, jak pravděpodobné je v modelu přeskočit ze stavu i do stavu j při posunu času z t do t+1. Jedná se o skutečné pravděpodobnosti, takže musí platit:

$$\sum_j a_{ij} = 1 \tag{47}$$

V příkladu, který je na obrázku, jsou pouze 3 typy:

- $a_{i,i}$ je pravděpodobnost setrvání ve stavu
- $a_{i,i+1}$ je pravděpodobnost přechodu do následujícího stavu
- $a_{i,i+2}$ je pravděpodobnost přeskočení následujícího stavu (používá se pouze někdy pro modely „krátkého ticha“)

$$\mathbf{A} = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrázek č.27 – Matice přechodových pravděpodobností [5]

Likelihood, že je ve stavu *j* vyslán vektor o(*t*)

„Vysílací“ likelihood $b_j[\mathbf{o}(t)] = p(\mathbf{o}(t)|j)$ může být:

- opravdová pravděpodobnost – u diskrétních HMM jsou vektory “předkvantovány” pomocí vektorové kvantizace na symboly, stavy pak obsahují tabulky vysílacích pravděpodobností jednotlivých symbolů.
- Hodnota funkce hustoty rozdělení pravděpodobnosti, která je dána většinou směsí Gaussových funkcí

$$b_j[\mathbf{o}(t)] = \sum_{i=1}^{M} \alpha_{ji} N(\mathrm{x}; \mu_{ji}, \Sigma_{ji}) \tag{48}$$

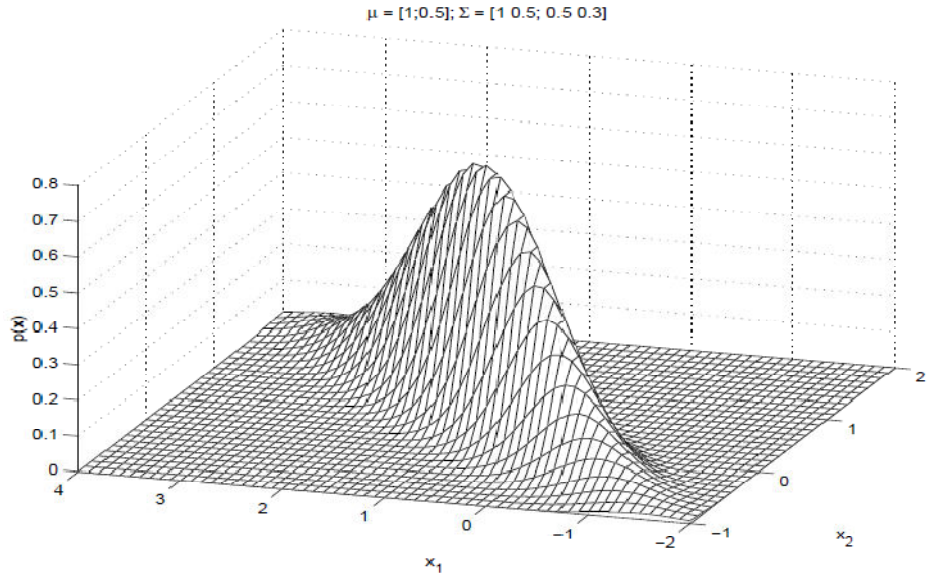
- Nebo pouze jednou Gaussovou funkcí

$$b_j[\mathbf{o}(t)] = N(\mathrm{x}; \mu_j, \Sigma_j) \tag{49}$$

Tyto modely se nazývají CD-HMM (Continuous density hidden Markov models)

Pokud by měl vektor **o**(t) jen jeden prvek, „vysílací“ likelihood $b_j[\mathbf{o}(t)]$ by se vypočítal následovně

$$b_j[\mathbf{o}(t)] = \frac{1}{\sqrt{(2\pi)^P |\Sigma_j|}} e^{-\frac{1}{2}(\mathbf{o}(t) - \mu_j)\Sigma_j^{-1}(\mathbf{o}(t) - \mu_j)^T} \tag{50}$$



Obrázek č.28 – Gaussovo rozložení pro korelované parametry [5]

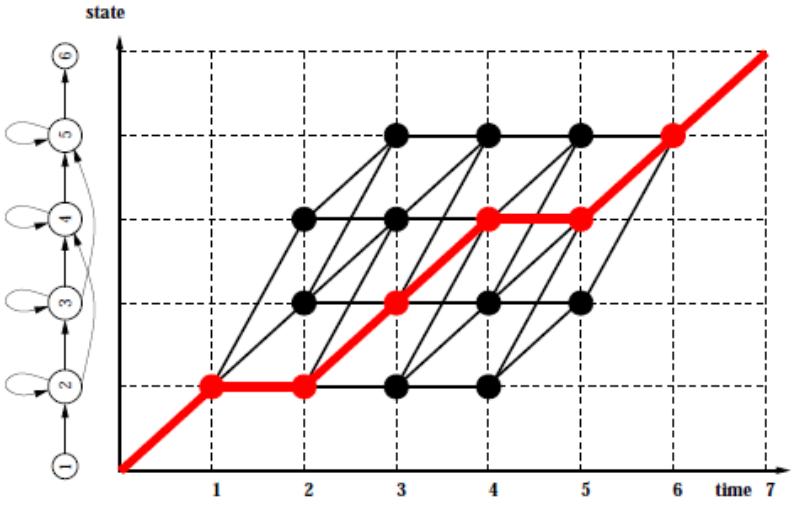
Pokud nejsou parametry korelované, kovarianční matice je diagonální, tzn. místo *P* x *P* koeficientů postačí *P* směrodatných odchylek. Hodnota rozdělení je pak dána pouze součinem 1-rozměrných Gaussových rozdělení a je dána vztahem

$$b_j[\mathbf{o}(t)] = \prod_{i=1}^{P} \frac{1}{\sigma_{ji}\sqrt{(2\pi)}} e^{\frac{-(\mathbf{o}(t)-\mu_{ji})^2}{2\sigma_{ji}^2}} \tag{51}$$

6.2.3 Likelihood generování celé sekvence O modelem *M*

Postupuje se po krocích. Nejprve se nadefinuje stavová sekvence, která stavy přiřadí vektorům, např. *X* = [1 2 2 3 4 4 5 6]. Ilustrace je na obrázku č.29.

Stavová sekvence má *T*+2 prvků, protože první a poslední stav tam musí být vždy a nepatří k nim žádný vektor. Při práci s časem umisťujeme první stav do neexistujícího času 0 a poslední stav do neexistujícího času *T*+1.



Obrázek č.29 – Stavové sekvence definující přiřazení vektorů ke stavům[6]

Jak ovšem definovat jeden likelihood generování sekvence vektorů modelem?

a) Baum-Welch:

$$p(\mathbf{O}|M) = \sum_{\{X\}} p(\mathbf{O}, X|M) \tag{52}$$

Suma přes všechny stavové sekvence délky $T + 2$.

b) Viterbi:

$$p^{\star}(\mathbf{O}|M) = \max_{\{X\}} p(\mathbf{O}, X|M) \tag{53}$$

je likelihood optimální cesty.

6.2.4 Trénování parametrů

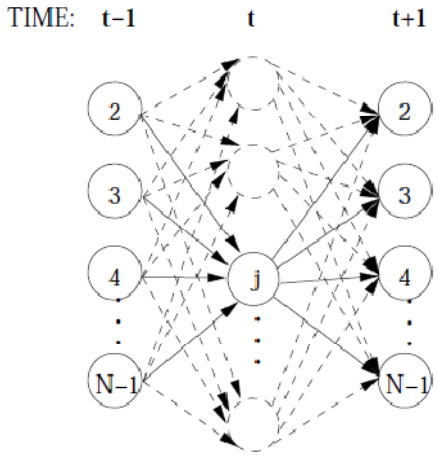
1. Parametry modelu jsou zhruba odhadnuty:

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{o}(t) \qquad \boldsymbol{\Sigma} = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{o}(t) - \boldsymbol{\mu})(\mathbf{o}(t) - \boldsymbol{\mu})^T \tag{54}$$

2. Výpočet funkce $L_j(t)$ (state occupation function).
3. Update parametrů

Výpočet state occupation function

Likelihood bytí ve stavu j v čase t lze spočítat jako sumu všech cest, které v čase t projdou stavem j.



Obrázek č.30 – Cesty procházející stavem *j* v čase *t* [5]

Pro zjištění state occupation counts je potřeba vypočítat dopředné (α) a zpětné (β) likelihoody pro všechny stavy a všechny časy. Výpočet se skládá ze 3 kroků.

1. Inicializace

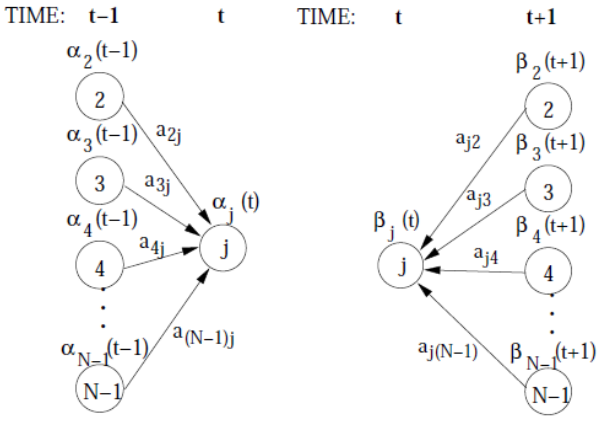
$$\alpha_j(1) = a_{1j} b_j[\mathbf{o}(1)] \quad \text{pro } 2 \leq j \leq N-1 \tag{55}$$

$$\beta_j(T) = a_{jN} \quad \text{pro } 2 \leq j \leq N-1 \tag{56}$$

2. Výpočet pro zbývající stavy a všechny časy

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij}\right] b_j[\mathbf{o}(1)] \quad \text{pro } 2 \leq j \leq N-1 \tag{57}$$

$$\beta_j(t) = \sum_{i=2}^{N-1} \beta_i(t+1) a_{ji}\, b_i[\mathbf{o}(t+1)] \quad \text{pro } 2 \leq j \leq N-1 \tag{58}$$



Obrázek č.31 – Výpočet dopředného a zpětného likelihoodu ve stavu *j* v čase *t* [5]

3. Uzavření algoritmu

$$\alpha_N(T+1) = \sum_{i=2}^{N-1} \alpha_i(T) a_{iN} \tag{59}$$

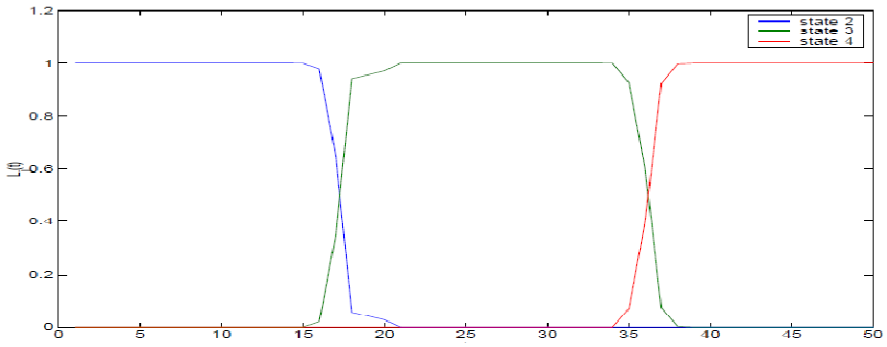
$$\beta_1(0) = \sum_{i=2}^{N-1} \alpha_i(1) a_{1i} b_i[\mathbf{o}(1)] \tag{60}$$

V posledním kroku jsme vypočítali sumu likelihoodů všech cest, které projdou jakýmkoliv stavem v čase *t*, což je vlastně Baum-Welchům likelihood. Dále již lze pomocí vypočítaných hodnot vypočítat samotná funkce $L_j(t)$, která je normalizovaná Baum-Welchovým likelihoodem, aby bylo zajištěno, že se bude jednat o skutečné pravděpodobnosti, následovně

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{p(\mathbf{O}|M)} \tag{61}$$

Update parametrů

Při odhadu $\boldsymbol{\mu}_j$ a $\boldsymbol{\Sigma}_j$ slouží occupation counts $L_j(t)$ jako „rozhazovač“ vektorů mezi stavy.



Obrázek č.32 – State occupation function [6]

Nové hodnoty parametrů budou vypočítány budou vypočítány ze všech vektorů, vážených příslušnou $L_j(t)$ podle vzorců:

$$\boldsymbol{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}(t)}{\sum_{t=1}^{T} L_j(t)} \qquad \boldsymbol{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}(t) - \boldsymbol{\mu})(\mathbf{o}(t) - \boldsymbol{\mu})^T}{\sum_{t=1}^{T} L_j(t)} \tag{62}$$

Odhad nových přechodových pravděpodobností:

$$a_{ij} = \frac{\sum_{t=1}^{T} \alpha_i(t) a_{ij} b_j\big(\mathbf{o}(t+1)\big)\beta_j(t+1)}{\sum_{t=1}^{T} L_j(t)} \tag{63}$$

6.2.5 Rozpoznávání s HMM

Rozpoznávání musí pracovat podle schématu na obr.24

- Máme rozpoznat neznámou sekvenci **O**.
- Ve slovníku máme $\check{N}$ slov $w_1 \ldots w_{\check{N}}$.
- Pro každé slovo máme natrénovaný model $M_1 \ldots M_{\check{N}}$
- Cílem je zjistit, který model by generoval **O** s největším likelihoodem

Bude se hledat takový model, který pro sekvenci vektorů **O** dá maximální likelihood:

$$i^\star = \arg\max_i \{p(\mathbf{O}|M_i)\} \tag{64}$$

Pro hledání jediného likelihoodu „vyslání“ sekvence **O** modelem M_i použijeme Viterbiho likelihood pro nejpravděpodobnější stavovou sekvenci:

$$p^\star(\mathbf{O}, X|M) = \max_{\{X\}} p(\mathbf{O}, X|M) \tag{65}$$

takže platí:

$$i^\star = \arg\max_i \{p^\star(\mathbf{O}|M_i)\} \tag{66}$$

Teoreticky by se měly vyhodnotit likelihoody všech možných stavových sekvencí X a najít Viterbiho likelihood optimální cesty, ale to by bylo velmi náročné a pro složitější modely a dlouhé promluvy by rozpoznávání patrně nikdy neskončilo[6].

Viterbiho trik

Naštěstí nemusíme nejprve počítat likelihood všech cest a pak teprve rozhodovat, která je nejlepší, ale můžeme evaluovat všechny cesty současně a rozhodnutí o nejlepší cestě dělat ne až na konci, ale pro každý čas t a každý stav j.

Definujeme částečný Viterbiho likelihood jako likelihood nejlepší cesty končící ve stavu j v čase t:

$$\Phi_j(t) = p^\star(\mathbf{o}(1) \ldots \mathbf{o}(t), x(t) = j|M) \tag{67}$$

a počítáme ji iterativně:

- Inicializace:

$$\Phi_j(1) = a_{1j} b_j[\mathbf{o}(1)] \text{ pro } 2 \le j \le N-1 \tag{68}$$

- Cyklus pro všechny časy t a všechny stavy j:

$$\Phi_j(t) = \max_i \{\Phi_i(t-1) a_{ij}\} b_j[\mathbf{o}(t)] \text{ pro } 2 \le j \le N-1 \tag{69}$$

- Uzavření algoritmu:

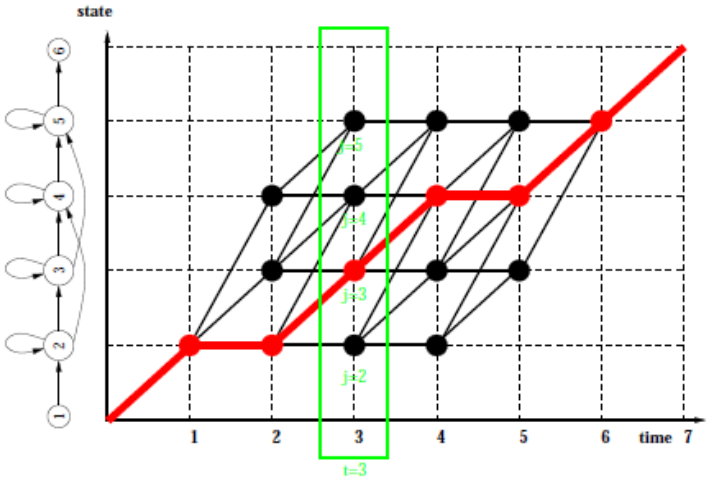
$$\Phi_N(T+1) = \max_i \{\Phi_i(T) a_{iN}\} \tag{70}$$

Poslední Φ je požadovaný Viterbiho likelihood:

$$p^{\star}(\mathbf{O}|M) = \Phi_N(T+1) \tag{71}$$

Výpočet obvykle probíhá v logaritmické oblasti: máme menší problémy s dynamikou a všechny součiny přejdou na součty:

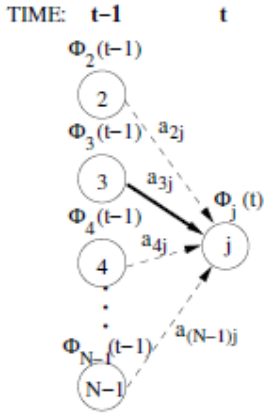
$$\Psi_j(t) = \max_i\{\Psi_i(t-1) + \log a_{ij}\} \log b_j[\mathbf{o}(t)] \tag{72}$$



Obrázek č.33 – Viterbiho algoritmus: rozhodnutí o nejlepší cestě děláme v každém čase *t* a v každém stavu *j* [6]

Implementace Viterbiho algoritmu pomocí token-passing

Viterbiho algoritmus se velmi dobře implementuje pomocí algoritmu token-passing, kde definujeme struktury, které přecházejí mezi stavy a při přechodu akumulují logaritmické pravděpodobnosti a likelihoody.



Obrázek č.34 – Jeden krok Viterbiho algoritmu: Výpočet částečného Viterbiho likelihoodu pro stav *j* a stav *i*. Tlustá šipka značí maximální hodnotu[5]

7 Implementace

Applety byly přirozeně programovány v programovacím jazyce Java. Konkrétní verze, která byla použita při programování a testování běhu programu, byla JRE 1.6.0_13. Jako vývojové prostředí bylo použito Netbeans 6.1 a pro testování aplikací v něm integrovaný program AppletViewer. Applety byly testovány na operačních systémech WINDOWS XP, WINDOWS VISTA a WINDOWS 7.

Kromě vlastních zdrojových souborů byla použita knihovna JAMA pro práci s maticemi, která umožňuje snadnější manipulaci s maticemi, než v případě použití např. klasických dvourozměrných polí v jazyce Java.

Při nahrávání signálu z mikrofonu je použit 1 kanál, vzorkovací frekvence 8000 vzorků za sekundu, kde každý vzorek je reprezentován pomocí 8 bitů. Pro práci se zvukem byl použit standardní balíček `javax.sound`.

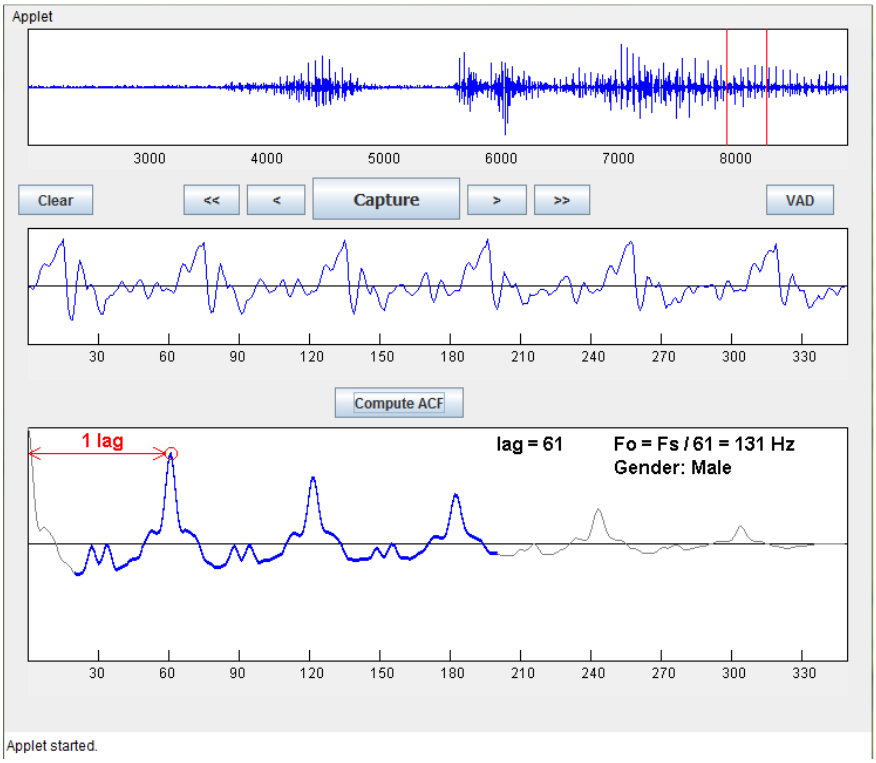
Pro ukládání dat (slovník) jsou použity klasické textové soubory s názvy *prislusne_slovo.txt*.

7.1 Detekce základního tónu

Applet provádějící detekci základního tónu a určování pohlaví mluvčího je pojmenován *PitchDetector*.

Uživatelský návrh

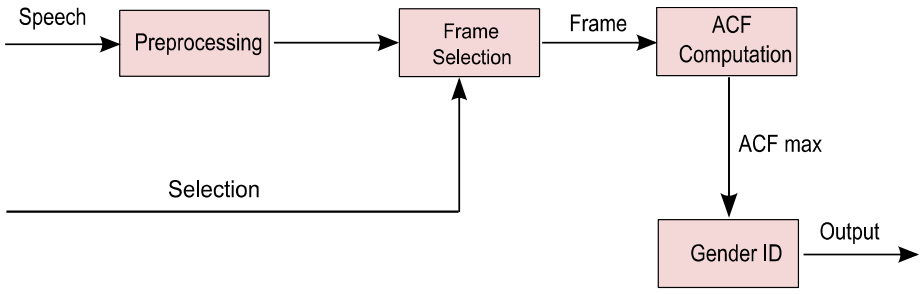
Účelem appletu *PitchDetector* je demonstrování průběhu výpočtu autokorelační funkce na vybraném rámci řeči. Uživateli je umožněno si vybrat libovolnou část vstupního signálu a jeho průběh detailněji zobrazit. Signál je možné nahrát libovolně dlouhý a pomocí uživatelského rozhraní je možné prohledávat celý signál. Pro zjednodušení nalezení signálu je součástí uživatelského rozhraní jednoduchý detektor řečové aktivity, který najde první výskyt řeči v signálu. Pro zajímavost není výběr rámce omezen, takže je možné aplikovat autokorelační funkci např. na šum nebo neznělé části řeči. Pro demonstraci průběhu autokorelační funkce je spuštěna jednoduchá animace.



Obrázek č. 35 – Applet *PitchDetector* při detekci základního tónu na mužském hlase

Implementace

Implementace programu nebyla obtížná. Applet je navržen poměrně jednoduše, viz.schéma na obrázku č.36.



Obrázek č.36 – Schéma appletu *PitchDetector*

Dále jsou uvedeny nejdůležitější součásti programu:

- *PitchApplet.java* – řídící třída programu
- *AudioIO.java* – vstup/výstup zvuku
- *PitchDetection.java* – veškeré zpracování signálu, včetně autokorelační funkce

Základem detektoru je metoda `calcACF(byte[] audiodata)`, která jako parametr obdrží vybranou část signálu a jsou vypočítány autokorelační koeficienty. Pro ilustraci je zdrojový kód metody uveden níže:

```
//výpočet autokorelační funkce
public int[] calcACF(byte[] testedFrame)
{
  //pole pro autokorelací koeficienty
  int[] r = new int[FRAME_LENGTH];
  //algoritmus detekce základního tónu
  for (int m=0; m<testedFrame.length; m++)
  {
     int suma = 0;
     for (int n=0;n<testedFrame.length-1-m; n++)
        suma += testedFrame[n]*testedFrame[n+m];
     r[m] = suma;
  }
  return r;
}
```

Z autokorelačních koeficientů je nalezeno maximum a vypočítána frekvence základního tónu. Při určování pohlaví je práh nastaven na hodnotu 180 Hz, tzn. řečník, jehož frekvence základního tónu je menší než 180 Hz, je identifikován jako muž.

Testování

Při testování se projevily všechny negativa spojené s určováním základního tónu. Zejména velké kolísání základního tónu mezi jednotlivými promluvami, ale i v rámci jedné promluvy díky změnám hlasu, náladě apod. Při testování bylo jedním mluvčím namluveno 10 odlišných promluv a z každé promluvy bylo vybráno náhodně 10 znělých rámců. Statistiky v tabulce č.1 ukazují veliké rozdíly mezi jednotlivými naměřenými hodnotami v podobě rozsahu naměřených hodnot pro každou promluvu a průměrné hodnoty.

Promluva	Rozsah hodnot [Hz]	Průměrná hodnota [Hz]
1	123-149	131
2	118-138	125
3	131-152	138
4	126-140	132
5	109-143	122
6	118 - 134	128
7	123-148	136
8	111-130	125
9	120-147	133
10	126-155	137

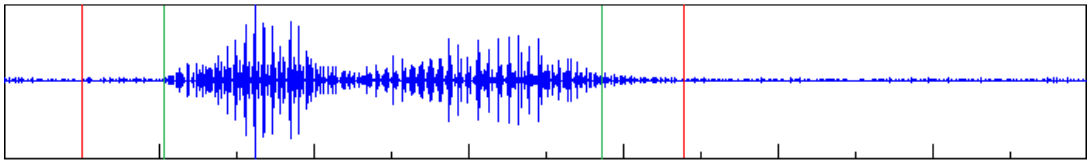
Tabulka č.1 – Statistika měření detekce základního tónu

7.2 Rozpoznávání izolovaných slov

Oba applety umožňují kromě rozpoznávání i trénování slov ve slovníku, popř. jejich smazání.Po nahrání promluvy je provedeno předzpracování signálu.Signál je ustředěn a rozdělen na rámce, jehož velikost jsem zvolil 25 milisekund (200 vzorků, pokud uvažujeme požadovanou vzorkovací frekvenci

8000 vzorků za vteřinu) a překrytí rámců 10 milisekund (120 vzorků), což ve výsledku dává 100 rámců každou vteřinu. Jako „okénkovací funce“ je v obou případech použito Hammingovo okno. Pro reprezentaci každého rámce jsem zvolil 13 mel-frekvenčních cepstrálních koeficientů v případě dynamického borcení času na rozdíl od appletu využívajících HMM, kde jsou ke standardním 13 koeficientům ještě vypočítány tzv. Δ a ΔΔ koeficienty (viz. kapitola 4.2.1), tzn. každý rámec je reprezentován pomocí 39 parametrů.

Dalším důležitým krokem je identifikace části vstupního signálu, který reprezentuje rozpoznávané (popř. trénované) slovo. Za tímto účelem jsem navrhl jednoduchý detektor řečové aktivity, který je založen na střední krátkodobé energii, který porovnává dvojice po sobě jdoucích rámců se zvolenou hodnotou prahu. Pokud je energie obou rámců vyšší, je první rámec považován za začátek promluvy. Obdobným způsobem je zjištěn i konec promluvy. Při detekci v nějakých případech docházelo k ignorování málo znělých částí slova, proto je počátek řeči posunut o 5 rámců směrem k počátku nahrávky. Naopak při určování konce promluvy směrem ke konci nahrávky. Tento krok se při testování aplikací ukázal jako velmi prospěšný. Užitečnost tohoto kroku je znázorněna na obrázku č. 37, kde by došlo k ignorování části signálu na konci slova. Rámce, které reprezentují slovo jsou uloženy a je provedena parametrizace, ostatní rámce jsou zahozeny, protože pro účely rozpoznávání nemají žádný význam, naopak při dlouhé nahrávce by mohly způsobit zpomalení programu díky zbytečnému provádění parametrizace. Pro lepší spolehlivost je možné citlivost detektoru zvolit explicitně z několika předdefinovaných úrovní (Extreme Low, Very Low, Low, Medium, High, Very High, Extreme High). Při testování se ukázaly jako dostačující hodnoty Low, Medium a High. Detektor se pro účely demonstrace algoritmů ukázal jako dostatečně funkční. Existují i dokonalejší algoritmy pro detekci řečové aktivity, např. pomocí cepstrální analýzy apod., ale rozhodl jsem se je nepoužít, neboť už přesahují rámec mojí práce.



Obrázek č. 37 – Původní (zeleně) a modifikovaný detektor řečové aktivity

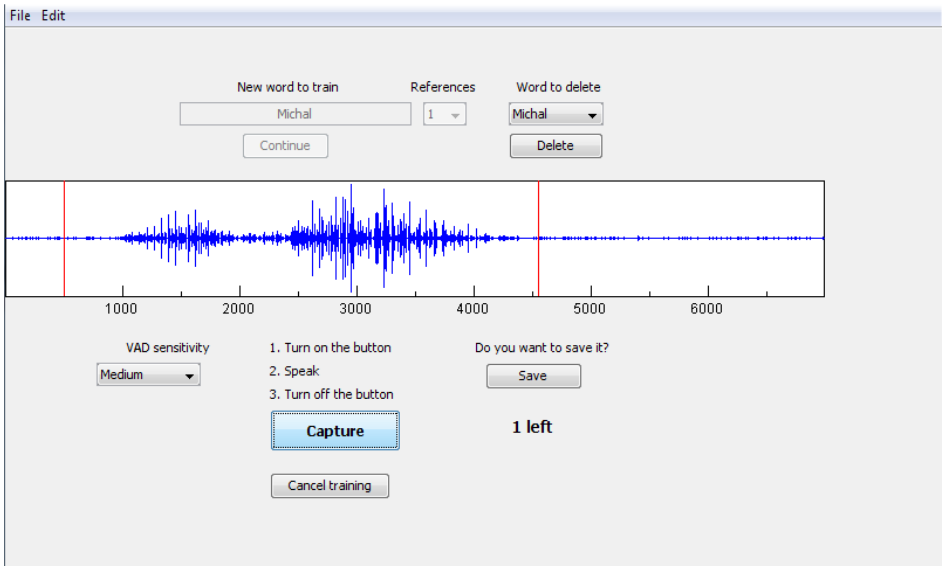
7.2.1 Dynamické borcení času

JavaApplet je pojmenován *DTWrecognizer* a má za úkol demonstrovat rozpoznávání izolovaných slov pomocí algoritmu dynamického borcení času.

Uživatelský návrh

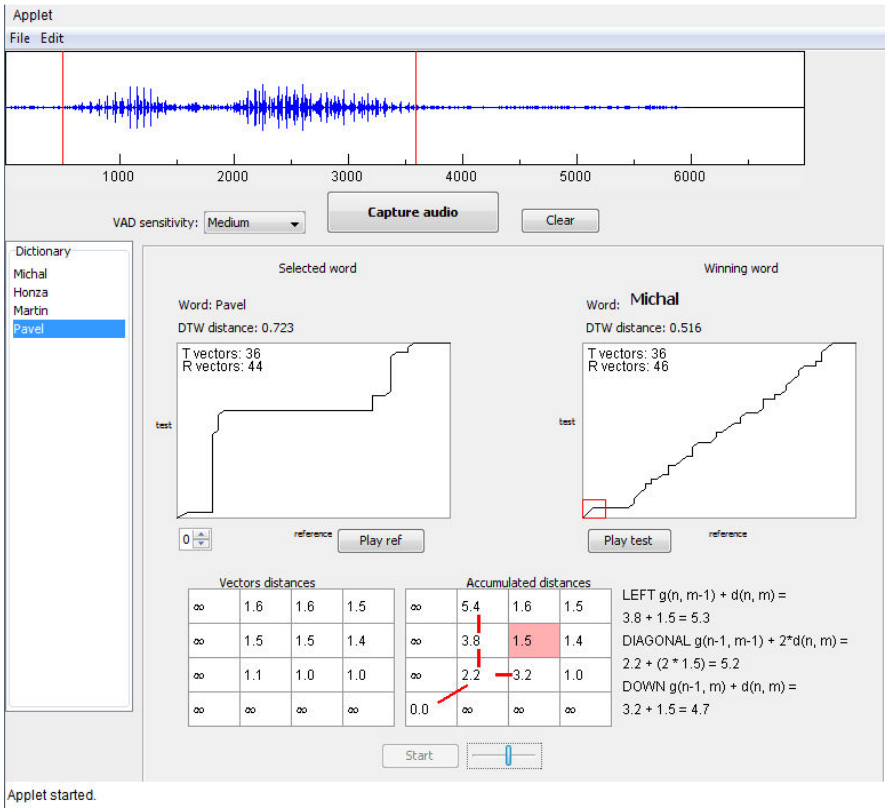
Jak již bylo zmíněno, applet umožňuje jak rozpoznávání, tak i práci se slovníkem v podobě trénování nových slov a mazání slov již uložených.

Pro práci se slovníkem je vytvořeno jednoduché uživatelské rozhraní (viz. obr.č. 38). Při trénování slov je zadáno slovo v textové podobě a počet promluv, které budou dané slovo reprezentovat a bude pomocí nich prováděno rozpoznávání. Poté následuje samotné nahrání řeči pomocí mikrofonu. Pro případ nepovedené nahrávky (např. šum nebo selhání detektoru řečové aktivity) je umožněno nahrávku „přeskočit“ a pokusit se o nové nahrání a uložení promluvy, kterou uživatel shledá jako kvalitní. Pro pozdější možnost porovnávání a demonstrace je ukládán spolu s mel-frekvenčními cepstrálními koeficienty také číslicový signál.



Obrázek č. 38 – *DTWrecognizer* při trénování slova „Michal“

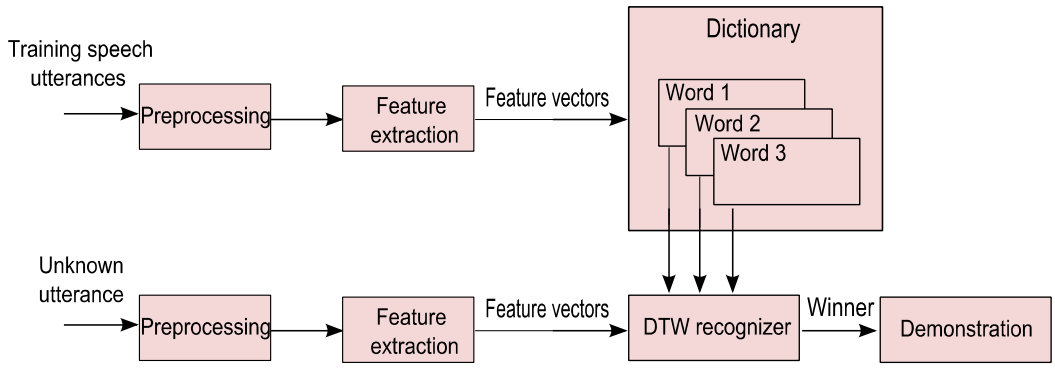
Při rozpoznávání je kladen důraz hlavně na demonstraci výsledků a použití algoritmu. Po nahrání promluvy je automaticky provedeno rozpoznávání. Následuje zobrazení výsledků rozpoznání ve formě rozpoznaného slova, DTW vzdálenosti vítězného slova a testovací promluvy a vykreslení optimální DTW cesty. Jak již bylo zmíněno, existuje možnost přehrání uloženého slova pro možnost porovnání, zda se pod slovem ve slovníku požadované slovo opravdu skrývá, popř. jak kvalitně nahráno je, podobně toto lze i pro promluvu testovací. Pro možnost porovnání výsledků pro jednotlivá slova ve slovníku s vítězným slovem je umožněno zobrazení optimální DTW cesty a výsledné DTW vzdálenosti k libovolnému slovu, nacházejícímu se ve slovníku. Pro demonstraci samotného algoritmu dynamického borcení času je uživateli umožněno vybrání libovolné části matice částečných kumulovaných vzdálenosti pomocí myši, na které je demonstrován algoritmus naplňování matice pomocí jednoduché animace. Applet *DTWrecognizer* v režimu rozpoznávání je zobrazen na obrázku č. 39.



Obrázek č. 39 – *DTWrecognizer* při rozpoznávání slova „Michal“ a porovnání se slovem „Pavel“

Implementace

Schéma rozpoznávače je uvedeno na obrázku č. 40



Obrázek č.40 – Schéma rozpoznávače *DTWrecognizer*

Níže jsou uvedeny některé důležité součásti programu:

- *DtwApplet.java* – řídící třída appletu
- *AudioIO.java* – vstup/výstup zvuku
- *DtwTraining.java* – trénování a odstraňování slov
- *DTW.java* – rozpoznávání
- *MFCC.java* – extrakce mel-frekvenčních cepstrálních koeficientů

- *SignalFrame.java* – reprezentuje rámec a všechny informace o něm

Klíčovou fázi rozpoznávání obstarává metoda `findRoute(double[][] dMatrix)`, kde parametr `dMatrix` reprezentuje matici vzdáleností mezi vektory. Pro ilustraci uvádím některé důležité součásti metody:

```
//hodnoty při jednotlivých přechodech, vybere se minimum
double left, down, diagonal;
//inicializace matice
routeMatrix = new RouteData[dMatrix.length][dMatrix[0].length];
//inicializace levého sloupce a spodního řádku
//první hodnota vlevo dole zůstane stejná jako v dMatrix * 2
routeMatrix[0][0]=new RouteData(2*dMatrix[0][0],"diagonal",dMatrix[0][0]);
//naplnění levého sloupce
for (int i=1;i<=height;i++)
   routeMatrix[i][0]=new RouteData(routeMatrix[i-1][0].value +
                                   dMatrix[i][0],"down",dMatrix[i][0]);
//naplnění spodního řádku
for (int i=1;i<=width;i++)
   routeMatrix[0][i] = new RouteData(routeMatrix[0][i-1].value +
                                     dMatrix[0][i],"left",dMatrix[0][i]);
//naplnění zbytku matice pomocí algoritmu DTW
for (int i=1;i<=width;i++) {
   for (int j=1;j<=height;j++)    {
       //výpočet hodnot při přechodech možných
       down = routeMatrix[j-1][i].value + dMatrix[j][i];
       left = routeMatrix[j][i-1].value + dMatrix[j][i];
       diagonal = routeMatrix[j-1][i-1].value + (2*dMatrix[j][i]);
       double min = 0;
       String dir = null;
       //najdeme minimum
       if (down < left && down < diagonal) {
          min = down;
          dir = "down";
       }
       if (left < down && left < diagonal) {
          min = left;
          dir = "left";
       }
       if (diagonal < down && diagonal < left) {
          min = diagonal;
          dir = "diagonal";
       }
       routeMatrix[j][i] = new RouteData(min, dir, dMatrix[j][i]);
   }
}
```

Testování

Pro účely testování bylo natrénování jedním mluvčím 7 slov v podobě křestních jmen, kdy ke každému slovu byly uloženy 3 reference. Testování bylo provedeno mluvčím, který trénoval slovník (mluvčí 1) i „nezávislým mluvčím“, kdy oba mluvčí postupně vyzkoušeli rozpoznat 20x všechny

slova uložená ve slovníku, a to jak v naprosto nehlučném prostředí, tak i v prostředí s výraznějším hlukem. Trénování a testování bylo provedeno na stejném počítači. Statistiky testování v tichém prostředí jsou uvedeny v následující tabulce.

Slovo	Úspěšnost mluvčího 1 [%]	Úspěšnost mluvčího 2 [%]	Celkem [%]
Michal	85	60	72.5
Honza	95	75	85
Pavel	90	65	77.5
Jirka	100	75	87.5
Aleš	100	80	90
Tomáš	85	75	80
Radim	90	70	80
	92	71.4	81.8

Tabulka č.2 – úspěšnost rozpoznávání pomocí DTW v nehlučném prostředí

Při testování v hlučném prostředí jsou statistiky rozpoznávání poněkud horší, jak vypovídá tabulka č. 3. Předpokladem pro testování bylo optimální nastavení detektoru řečové aktivity pro dané prostředí.

Slovo	Úspěšnost mluvčího 1 [%]	Úspěšnost mluvčího 2 [%]	Celkem [%]
Michal	75	45	60
Honza	80	60	70
Pavel	70	60	65
Jirka	100	65	82.5
Aleš	90	70	80
Tomáš	65	65	65
Radim	80	70	75
	80	62	71

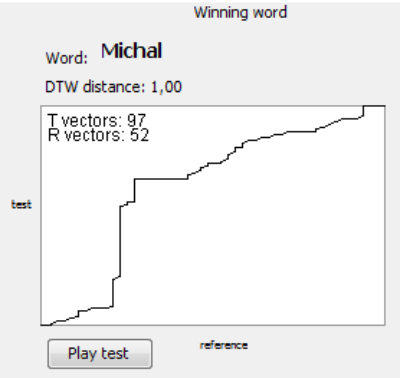
Tabulka č.3 – úspěšnost rozpoznávání pomocí DTW v relativně hlučném prostředí

Z výsledků testování vyplývá, že rozpoznávač nevykazuje ideální spolehlivost rozpoznávání. Nevýhodou je mírná závislost na řečníkovi. To je způsobeno pravděpodobně nepřesnostmi při parametrizaci. Při nezávislém testování se projevily i rozdíly při rozpoznávání slov stejným mluvčím při použití jiného PC, popř. jiného mikrofonu, než bylo použito při trénování. Navzdory tomu při testování mluvčím, který nahrával slovník, je úspěšnost poměrně uspokojivá. Úspěšnost rozpoznávání je též ovlivněna velikostí slovníku. Při použití malého slovníku, např. dvě slova, je úspěšnost obou mluvčích velice dobrá. Tabulka č. 4 uvádí statistiky rozpoznávání slov ANO/NE stejnými mluvčími, jako v prvních dvou případech.

Slovo	Úspěšnost mluvčího 1 [%]	Úspěšnost mluvčího 2 [%]	Celkem [%]
ANO	100	95	97.5
NE	100	100	100
	100	97.5	98.75

Tabulka č.4 – úspěšnost rozpoznávání slov Ano/Ne

Rozpoznávač se umí uspokojivě vypořádat s výrazným rozdílem délky testovací a referenční promluvy. Na obrázku č. 41 lze pozorovat výsledky rozpoznání slova „Mííííííchal“ i odpovídající průběh DTW cesty.



Obrázek č. 41 – Úspěšné rozpoznání slova s výrazně rozdílnou délkou

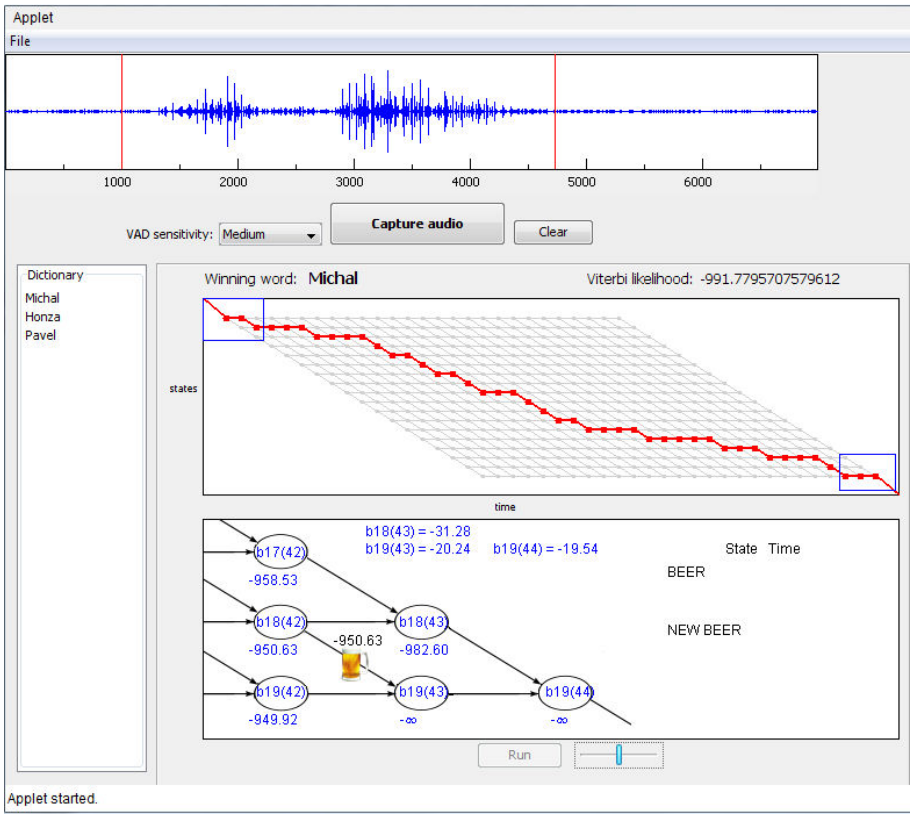
7.2.2 Skryté Markovovy modely

Applet je pojmenován *HMMrecognizer* a jeho úkolem je demonstrovat rozpoznávání slov pomocí skrytých Markovových modelů a algoritmu token passing.

Uživatelský návrh

Obdobně jako u dynamického borcení času podporuje applet i režim trénování modelů s velmi podobným uživatelským rozhraním, kdy je zadáno trénované slovo v textové podobě a počet promluv, na kterých bude model trénován. Program je výhradně zaměřen na demonstraci algoritmu pro rozpoznávání, tzn. trénovací algoritmus zůstává uživateli skrytý, což se nabízí jako jedno z možných vylepšení do budoucna.

Při rozpoznávání je uživateli zobrazeno výsledné slovo, hodnota Viterbiho likelihoodu a je vykreslena optimální stavová sekvence. Pro rozpoznávání a demonstraci je použit algoritmus token passing, který je možné pomocí jednoduché animace demonstrovat na počátku a konci promluvy (viz. obr. 42). Oproti demonstraci DTW jde o omezení, ale v tomto případě se animace chodu algoritmu na jakékoliv části promluvy ukázala jako obtížně programovatelná, proto jsem zvolil tuto jednodušší variantu, která pro demonstrování algoritmu plně dostačuje.

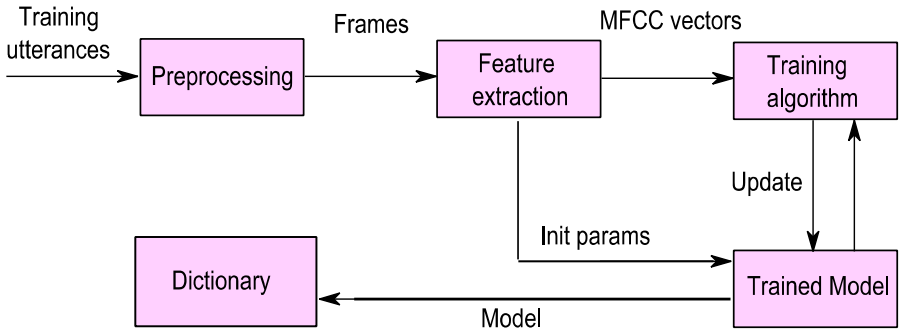


Obrázek č. 42 – *HMMrecognizer* při rozpoznávání slova „Michal“

Implementace

Model obsahuje 20 stavů, přičemž každý stav je reprezentován nejjednodušším způsobem v podobě střední hodnoty a jednoho gaussova rozdělení. Přechodové pravděpodobnosti jsou nastaveny na pevnou hodnotu 0.5. Vzhledem ke skutečnosti, že jsou pro rozpoznávání použity mel-frekvenční cepstrální koeficienty, které jsou dostatečně nekorelované, lze místo kovarianční matice brát v úvahu pouze směrodatné odchylky, což značně zjednoduší výpočet a časovou náročnost algoritmu. Lze vynechat časově náročné operace, jako je například výpočet inverze a determinantu kovarianční matice, která má, vzhledem k počtu parametrů rozměry 39x39, což je celkem 1521 hodnot. Díky nekorelovanosti MFCC koeficientů a využití pouze směrodatných odchylek je počet hodnot, se kterými je třeba počítat, omezen na 39 hodnot ležících na diagonále kovarianční matice.

Samotná implementace se ukázala jako ne zcela triviální, převážně v případě trénování modelů. Jako počáteční hodnoty stavů jsou vypočítány globální hodnoty z celé promluvy, popř. ze všech trénovacích promluv. Následuje update parametrů, kde se ukázalo při výpočtu *state occupation counts* být velikým problémem podtékání registru počítače při výpočtu zpětných a dopředných likelihoodů. Problém jsem vyřešil pomocí logaritmů, což ovšem podstatně znesnadnilo implementaci. Při logaritmování součtu byla použita funkce `LogAdd(double x, double y)`, jejíž zdrojový kód v jazyce C mi poskytnul Ing. Petr Schwarz Ph.D, a která má, oproti klasické implementaci logaritmu sčítání, výhodu v tom, že zvládá větší dynamický rozsah.



Obrázek č.43 – Schéma trénování modelu

Níže jsou uvedeny některé důležité součásti programu:

- *HmmApplet.java* – řídící třída appletu
- *AudioIO.java* – vstup/výstup zvuku
- *HmmTraining.java* – trénování a odstraňování slov
- *HmmRecognition.java* – rozpoznávání
- *MFCC.java* – extrakce mel-frekvenčních cepstrálních koeficientů
- *SignalFrame.java* – reprezentuje rámec a všechny informace o něm
- *HmmModel.java* – reprezentuje model a všechny informace o něm
- *State.java* – reprezentuje stav modelu a všechny informace o něm

Testování

Při testování přesnosti rozpoznávače byl kladen důraz na porovnání výsledků v závislosti na kvalitě natrénovaných modelů. Ukázalo se, že kvalita a použitelnost modelu jsou silně závislé na počtu trénovacích promluv. V extrémních případech (počet promluv 1-5) docházelo k chybnému natrénování a ve výsledku byl model ve většině případů nepoužitelný. K případům správného rozpoznání docházelo při použití přibližně 10-ti trénovacích promluv. Při dále rostoucím počtu promluv se přesnost rozpoznávače zvyšovala. Tabulka č.5 porovnává výsledky rozpoznávání 5-ti stejných slov, které jsou v 1. případě natrénovány pomocí 10-ti promluv a ve 2. případě pomocí 20-ti promluv. Trénování i testování bylo provedeno jedním mluvčím.

Slovo	10 promluv [%]	20 promluv [%]
Michal	30	90
Honza	60	100
Petr	100	100
David	70	90
Pavel	90	100
	68	96

Tabulka č. 5 – Statistiky rozpoznávání appletu *HMMrecognizer*

Nevýhodou appletu *HMMrecognizer* je možnost nahrávání promluv pouze z mikrofonu. Vzhledem k potřebě velkého množství trénovacích promluv je tento postup poněkud zdlouhavý.

7.3 Omezení

Při tvorbě mé práce jsem narazil na několik výrazných problémů. Při testování se ukázaly rozdíly mezi jednotlivými použitými mikrofony, kdy některé mikrofony např. vykazovaly vysokou hlasitost, naopak některé mikrofony nízkou hlasitost. Rozdíl jsem zaznamenal i v kvalitě mikrofonů, kde se např. u starších mikrofonů objevoval výrazný šum. Co se týče přenositelnosti docházelo k problémům na počítači se systémem Windows XP při použití méně kvalitního mikrofonu.

Jako největší problém se ukázalo testování JavaAppletů v internetových prohlížečích. Ukázalo se, že JavaApplety mají implicitně zakázáno nahrávání z mikrofonu, aby nemohly odposlouchávat uživatele. Problém je možné vyřešit podepsáním appletů důvěryhodnou autoritou, což v praxi stojí peníze, proto se tato alternativa ukázala jako nevhodná. Druhou možností je lokální povolení *policy* na každém PC, na kterém by uživatel applet spouštěl, což je opět nevhodné. Problém jsem vyřešil pomocí programu *appletViewer*, který byl přidán ke všem programům a pomocí dávkového souboru je program spuštěn jako klasická „okénková“ aplikace.

8 Závěr

Hlavním bodem mé diplomové práce bylo demonstrovat algoritmy používané při zpracování a rozpoznávání řeči. To dle mého názoru moje práce splňuje. Práce byla určitě velice zajímavá a během ní jsem si rozšířil znalosti o zpracování řečového signálu a algoritmech pro rozpoznávání.

Samotné JavaApplety, minimálně applety pro detekci základního tónu a rozpoznávání pomocí DTW, vykazují poměrně uspokojivou spolehlivost, ale v praxi určitě své využití nenajdou. Nabízí se jejich využití spíše pro studenty jako pomůcka pro pochopení dané problematiky.

Co se týče možností do budoucna, applety jistě nabízí velké množství možných zdokonalení. Zdokonalení a zpřesnění parametrizace, možnost využití jiných parametrů, pro demonstraci určování frekvence základního tónu lze naprogramovat více metod, u skrytých Markovových modelů lze modely rozšířit tak, aby jednotlivé stavy, pro zvýšení přesnosti rozpoznávání, neobsahovaly jedno Gaussovo rozdělení, ale jejich směs. Rozšíření rozpoznávačů o možnost rozpoznávání spojených slov atd. Určitě lze také demonstrovat algoritmy dokonalejšími grafickými animacemi.

Literatura

[1] Psutka, J., Müller, L., Matoušek, J., Radová, V.: *Mluvíme s počítačem česky*. Academia, Praha, 2006. ISBN 80-200-1309-1

[2] Černocký, J.: *Předzpracování řeči, tvorba řeči, cepstrum.* FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/03_prepro_model_ceps/03_prepro_model_ceps.pdf>

[3] Černocký, J.: *Určování základního tónu řeči.* FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/05_pitch/05_pitch.pdf>

[4] Černocký, J.: *Rozpoznávání řeči – úvod a DTW*. FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/08_rozp_dtw/08_rozp_dtw.pdf>

[5] Černocký, J.: *Rozpoznávání řeči - HMM*. FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/09_hmm/09_hmm.pdf>

[6] Černocký, J.: *Zpracování řečových signalů – Studijní opora.* FIT VUT v Brně. Dostupné na URL: < http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf>

[7] Černocký, J.: *SRE 03 – Skryté Markovovy Modely HMM*. FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/SRE/public/prednasky/04a_hmm_honza/sre_03_hmm.pdf>

[8] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., Woodland, P.: *The HTK Book*, Microsoft Corporation, 1995-1999. Dostupné na URL: <http://nesl.ee.ucla.edu/projects/ibadge/docs/ASR/htk/htkbook.pdf>

[9] Burget, L.: *Parametrizace řeči*. FIT VUT v Brně. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/11_priznaky/11_priznaky.pdf>

[10] Psutka, J.: *Komunikace s počítačem mluvenou řečí.*Academia Praha, 1995. ISBN 80-200-0203-0

[11] Chalupníček, K.: *Rozpoznávání diktované řeči pro medicínské aplikace [DP]*. FEKT VUT v Brně, 2004. Dostupné na URL: <http://www.fit.vutbr.cz/~chalupni/publikace/DP/DP.pdf>

[12] WWW stránky: *Anatomie lidského těla*. < http://anatomie-lidskeho-tela.kvalitne.cz/>

Seznam příloh

Příloha 1. CD

Obsahuje 4 složky. Složky *PitchDetector, DTWrecognizer, HMMrecognizer* obsahující jednotlivé applety a složku *Report* obsahující technickou zprávu

Všechny složky obsahují:
BUILD – obsahuje složku classes s přeloženými třídami
SRC – zdrojové kódy
Applet.policy – nastavení práv appletů
AppletViewer.exe – pro spuštění
jli.dll – knihovna pro spuštění appletVieweru
msvcr71.dll – knihovna zajišťující přenositelnost
jmeno_programu.bat – spouštěcí dávkový soubor
manual.pdf – návod k použití

Složky *DTWrecognizer* a *HMMrecognizer* navíc obsahují složku pro ukládání slov.