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

MULTIPLATFORMNÍ APLIKACE PRO VERIFIKACI MLUVČÍHO

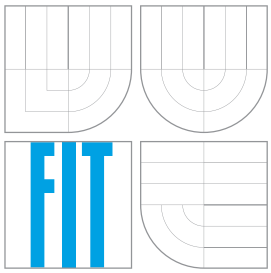
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN GÖRIG

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

MULTIPLATFORMNÍ APLIKACE PRO VERIFIKACI MLUVČÍHO

MULTIPLATFORM APPLICATION FOR SPEAKER VERIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN GÖRIG

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ONDŘEJ GLEMBEK

BRNO 2010

Abstrakt

Bakalářská práce se zabývá rozpoznáváním mluvčího bez znalosti textu sdělení. Zmiňuje dnes používané způsoby extrakce příznaků a jejich vyhodnocení pomocí směsice Gaussových hustotních funkcí. Praktickým výstupem práce je aplikace pro vizualizaci průběhu rozpoznávání. Návrh aplikace je multiplatformní a využívá knihoven Qt a BSAPI.

Abstract

Bachelor thesis considers speaker recognition without knowledge of spoken message. There are described current feature extraction methods and their evaluation using Gaussian mixture model. The practical output of this work is application for visualization of the recognition process. Developed application is cross platform and it uses Qt and BSAPI libraries.

Klíčová slova

verifikace mluvčího, identifikace mluvčího, multiplatformní aplikace, PCM, MFCC, STG, GMM, UBM, BSAPI, Qt

Keywords

speaker verification, speaker identification, multiplatform application, PCM, MFCC, STG, GMM, UBM, BSAPI, Qt

Citace

Jan Görig: Multiplatformní aplikace pro verifikaci mluvčího, bakalářská práce, Brno, FIT VUT v Brně, 2010

Multiplatformní aplikace pro verifikaci mluvčího

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ondřeje Glembka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Görig
17. května 2010

Poděkování

Na tomto místě děkuji zejména vedoucímu této práce Ing. Ondřeji Glembkovi za všechny inspirace, rady a připomínky. Dále chci poděkovat ostatním členům skupiny Speech@FIT za poskytnuté informace zejména skrze vyučované předměty. Děkuji také všem ostatním, kteří mne podporovali jakýmkoliv způsobem při vytváření této práce. Díky!

Obsah

Seznam obrázků

Kapitola 1

Úvod

Rozpoznávání mluvčího pomocí počítače se vzhledem ke stále výkonnějšímu hardwaru stává čím dál více účinnější a lze jej provozovat na velkém množství zařízení. Průběh rozpoznávání i poskytnuté výsledky však mohou být nic neříkající a příliš abstraktní. Proto bylo cílem této práce navržení jednoduchého programu vizualizujícího výsledky poskytnuté rozpoznávačem. Práce se však nezabývá jen vývojem aplikace, ale popisuje i techniky používané pro samotné rozpoznávání.

Kapitola Rozpoznávání mluvčího (2) rozděluje úlohu na dvě základní skupiny a uvádí příklady, jak lze rozpoznání mluvčího využít.

Další kapitola (3) se zabývá zpracováním řeči v počítači. Jsou zde uvedeny postupy, které se v současné době používají pro extrakci příznaků vhodných pro rozpoznání mluvčího. Dále je zde popsán způsob trénování modelu mluvčího a vyhodnocování podobnosti nahrávky s natrénovaným modelem.

Kapitola Multiplatformní aplikace (4) nastiňuje příčinu vzniku a způsob tvorby aplikací fungujících na více platformách. Je zde popsána historie a současnost knihovny Qt.

Následující kapitola (5) stručně popisuje vývojové nástroje použité pro vytvoření aplikace. Větší částí se pak věnuje funkci a základní struktuře implementovaných komponent.

Kapitola s názvem BSAPI (6) uvádí základní informace o knihovně určené pro rozpoznávání řeči vyvinuté na ÚPGM. Velká část kapitoly je věnována popisu použité konfigurace knihovny.

Kapitola Experimenty (7) obsahuje výsledky testování hardwarové náročnosti aplikace a spolehlivosti rozpoznávaní.

Závěrečná kapitola (8) obsahuje zhodnocení provedené práce a uvádí některé možnosti dalšího vývoje.

Kapitola 2

Rozpoznávání mluvčího

Úlohu o rozpoznávání mluvčího lze rozdělit na dvě základní oblasti. Lze provádět identifikaci a verifikaci mluvčího. Obě oblasti jsou do značné míry podobné a způsob jejich implementace se do značné míry prolíná. V některých literárních odkazech jsou dokonce tyto pojmy zaměňovány. Podrobnější rozbor lze nalézt v [4].

2.1 Verifikace mluvčího

Verifikace mluvčího se zabývá ověřením toho, zda dodaný hlas patří určitému řečníkovi. Verifikace tedy vyžaduje kromě vstupního zvuku také informaci o tom, za koho se daná osoba vydává. Výstupem systému je potom skóre určující šanci, že se opravdu jedná o domnělého řečníka. Pro binární rozhodnutí o přijetí či zamítnutí osoby, je nutné nastavit rozhodovací práh. Typickým využitím této technologie je umožnění přístupu privilegované osoby do nějakého systému a odmítnutí přístupu vetřelci.

2.2 Identifikace mluvčího

Identifikace se oproti tomu snaží přiřadit danou nahrávku konkrétnímu řečníkovi. Dá se říci, že jde o verifikaci každého mluvčího v databázi a následný výběr toho s největším skóre. Výstupem tak může být buď soubor čísel udávajících podobnost poskytnuté nahrávky k jednotlivým řečníkům nebo jedno konkrétní jméno. Ukázkovým použitím je například přiřazení jmen k probíhajícím telefonním hovorům.

Pokud bychom se striktně drželi této definice, pak by implementovaný program prováděl identifikaci mluvčího. Jak však bylo zmíněno, tyto pojmy jsou někdy zaměňovány a pro účely vizualizace je vhodnější úlohou identifikace mluvčího.

Kapitola 3

Zpracování řeči

Při zpracovávání řeči a obecně zvuku v počítači je nejprve potřeba vyřešit problém s reprezentací zvukových dat. Řeč je v reálném světě analogový signál a počítač je diskrétní výpočetní přístroj. Je tedy nutné převést zvuk do množství diskrétních dat. Vzhledem k dalšímu, časově náročnému, zpracovávání dat je při rozpoznávání řečníka navíc nutné provést redukci dat, aby nebylo nutné zpracovávat obrovské množství informací.

Jakmile jsou z řeči získány vhodné příznaky pro rozpoznání mluvčího, je použit nástroj stanovující, kterému mluvčímu z databáze je nahrávka více či méně podobná.

3.1 Pulsní kódová modulace

Pulsní kódová modulace (zkráceně PCM) je základní způsob převedení analogového signálu (zvuku) do digitální podoby. Postup digitalizace je rozdělen do dvou kroků: vzorkování a kvantizace s kódováním.

Pomocí **vzorkování** je analogový signál transformován na posloupnost analogových vzorků diskrétních v čase. Při provádění této operace je nutné dbát na omezení vyplývající ze vzorkovacího teorému. Je nutné aby platilo $f_v > 2f_m$, kde f_v je frekvence vzorkování a f_m je maximální frekvence signálu. Pro zamezení aliasingu, se signál filtruje dolní propustí.

Kvantizace s kódováním převádí navzorkované analogové hodnoty na číslicovou reprezentaci. U PCM je tento převod řešen lineárním analogově/digitálním převodníkem. Rozsah signálu je tedy pomocí stejného kroku rozdělen na 2^b úrovní, kde b udává počet bitů potřebných pro uložení jednoho vzorku. Je zřejmé, že počet bitů má zásadní vliv na přesnost číslicových hodnot.

PCM se vyznačuje vysokou redundancí dat a z toho vyplývajícími vysokými nároky na přenosové pásmo. Proto se v praxi využívají další kódování, která nejsou tak paměťově náročná. Jedná se zejména o kódování μ-law, A-law a diferenční pulsní kódovou modulaci.

Podrobnější informace ohledně PCM a dalších kódováních lze nalézt v [4], ze které jsem čerpal v této sekci.

3.2 Melovské kepstrální koeficienty

Melovské kepstrální koeficienty jsou v současné době základními příznaky používanými pro rozpoznávání řeči. Jsou známy pod zkratkou MFCC. Jejich výpočet kombinuje techniky známé ze zpracování signálů a reflektuje některé vlastnosti lidského ucha. Pro jejich získání se používá sekvence operací popsaných v této sekci.

Název je odvozen ze dvou základních principů využitých při výpočtu těchto koeficientů:

- **kepstrum** – způsob oddělení buzení a impulsní odezvy filtrů, dvou základních složek tvořících zvuk mluvené řeči. Principem je převedení signálu do frekvenční oblasti, zlogaritmování a opětovné převedení do časové oblasti [12].
- **melovská frekvenční škála** – úprava frekvencí způsobem jakým je vnímá lidské ucho [4].

3.2.1 Segmentace signálu

Nahrávku obsahující řeč nelze zpracovat najednou v celé délce. Znemožňují nám to nejen technické limity, ale i způsob, jakým je signál dále zpracováván. Nahrávku je tedy nutné rozdělit na menší části, které nazýváme **mikrosegmenty**. Délka těchto částí nemůže být zvolena náhodně, ale musí splňovat určité požadavky.

Mikrosegment musí být dostatečně krátký na to, aby bylo možné považovat obsaženou část signálu za stacionární a naopak musí být natolik dlouhý, aby obsahoval potřebné množství parametrů pro další zpracování [11]. Nejčastěji se používají mikrosegmenty o délce 10 až 30 ms [4].

Jelikož je v dalších krocích signál zpracováván pomocí Fourierovy transformace, není možné provést prosté rozsegmentování signálu. Kvůli zabránění vzniku rušení je nutné přenášet signál pomocí některé okénkové funkce. Ve většině případů se používá tzv. Hammingovo okénko, které filtruje signál zejména v hraniční oblasti segmentu. [7]. Délka tohoto okénka bývá určena tak, aby korespondovala s délkou mikrosegmentu a zároveň aby její délka byla mocninou dvou, kvůli možnosti použít FFT (rychlou Fourierovu transformaci) [4].

Dalším požadavkem je, aby se na sebe navazující segmenty příliš neměnily. Toho je v praxi dosahováno tak, že se jednotlivé segmenty navzájem překrývají. Je však potřeba dávat pozor na to, aby překrytí nebylo příliš velké. Pokud se totiž segmenty příliš překrývají, vzniká problém s velkou podobností segmentů a zvyšujícími se nároky na zpracování. Typicky je voleno takové překrytí, aby následující segment začínal 10 ms po předchozím. Získáme tak 100 mikrosegmentů za sekundu [11].

3.2.2 Rychlá Fourierova transformace

Rychlá Fourierova transformace, zkráceně FFT, je méně časově náročnou alternativou výpočtu diskrétní Fourierovy transformace (DFT). Její použití je podmíněno použitím vstupní sekvence o délce rovnající se mocnině dvou. Jak již bylo dříve zmíněno, tuto podmínku splníme zvolením vhodné velikosti mikrosegmentu, resp. okénka.

DFT slouží pro převod signálu z časové do frekvenční oblasti a je definována jako:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{3.1}$$

Další informace o DFT, stejně jako uvedenou definici, lze nalézt v [13].

3.2.3 Melovská filtrace

Cílem melovské filtrace je změna frekvenční charakteristiky zvuku tak, aby lépe odpovídal vlastnostem lidského ucha. Pro lidské ucho platí, že je schopno lépe rozlišovat zvuky o nižších

frekvencích než o frekvencích vyšších. Naproti tomu získané DFT koeficienty z minulého kroku jsou v celém frekvenčním pásmu lineární.

Nelinearity je dosaženo převedením frekvencí z lineární škály do melovské škály. Tento převod je definován následujícím vztahem:

$$f_m = 2595 \log_{10} \left(1 + \frac{f}{700}\right) \tag{3.2}$$

Převod se však neprovádí přepočítáváním pomocí této rovnice, ale je využita banka trojúhelníkových filtrů. Trojúhelníky jsou rozmístěny na frekvenční ose tak, aby reprezentovaly převod na melovskou frekvenční škálu. Dalším důsledkem této implementace je snížení počtu koeficientů, které vznikly pomocí DFT, na počet, který se rovná počtu použitých trojúhelníků.

Podrobněji viz [4].

3.2.4 Logaritmování

Koeficienty získané pomocí melovské filtrace jsou následně zlogaritmovány. Děje se tak ze dvou důvodů. Prvním důvodem je omezení dynamiky zvukového signálu, čímž simulujeme činnost lidského ucha [4]. Druhý důvod najdeme ve výpočtu kepstra, na kterém jsou MFCC koeficienty založeny. Zlogaritmování koeficientů způsobí převedení konvoluce buzení a impulsní odezvy filtrů na součet, a tím umožní jejich případné budoucí oddělení. [12]

3.2.5 Diskrétní kosinová transformace

Posledním zbývajícím krokem pro získání MFCC je provedení zpětné diskrétní Fourierovy transformace. Vzhledem k vlastnostem vstupních dat, ji lze nahradit diskrétní kosinovou transformací (DCT). Příjemným důsledkem této operace je vytvoření vzájemně nekorelovaných koeficientů.

Výpočet jednotlivých koeficientů je prováděn jako:

$$c_m(j) = \sum_{i=1}^{M^*} \log y_m(i) \cos\left(\frac{\pi j}{M^*}(i - 0,5)\right) \tag{3.3}$$

kde M^* je počet koeficientů získaných melovskou filtrací a m je číslo počítaného koeficientu od 0 do požadovaného počtu. Počet koeficientů se obvykle volí v rozmezí 10 až 13.

Detailnější popis výpočtu, stejně jako uvedené konkrétní informace, obsahuje [4].

3.3 Feature warping

Feature warping (název se nepřekládá, volný překlad by mohl být „borcení příznaků“) je metoda pro vylepšení vlastností příznaků používaná zejména při rozpoznávání řečníka. Použitím se snižuje nepříznivý vliv přítomnosti ruchů v nahrávce a rozdílné charakteristiky použitých mikrofonů. Principem je transformace příznaků tak, aby byly v intervalech o konstantní délce distribuovány podle vhodného rozdělení.

V praxi se využívá transformace na normální rozdělení. Tato varianta je zvaná „krátkodobá gaussianizace“ – anglicky Short Time Gaussianization, zkráceně STG. Tato transformace způsobí, že signál bude mít nulovou střední hodnotu, jednotkovou varianci a bude lokálně gaussovsky rozdělený [1].

Cíle metody je dosaženo následujícím postupem. Nejprve určíme velikost okna (v počtu mikrosegmentů), se kterým budeme pracovat a promítneme inverzní distribuční funkci cílového rozložení na interval diskrétních hodnot o velikosti okna. Následně posunujeme okno po příznacích a měníme vždy hodnotu příznaku uprostřed okna. Novou hodnotu příznaku zjistíme tak, že seřadíme vzestupně prvky uvnitř okna a nalezneme novou pozici zpracovávaného prvku. Tuto pozici namapujeme na hodnotu pomocí předpřipraveného diskrétního intervalu.

Tato metoda byla popsána svými autory v [3].

3.4 Dynamické koeficienty

Získané příznaky mají dobré vlastnosti pro rozpoznávání řečníka, ale jejich základním problémem je, že jsou statické. Každý příznak popisuje vlastnosti zvuku v krátkém časovém intervalu a nereprezentuje jeho souvislost s okolím. Naproti tomu řeč je charakteristická svou dynamikou, která může být použita jako další důležitý příznak pro rozpoznávání [2].

Pro zachycení dynamiky řeči se používají **dynamické koeficienty**. Tyto koeficienty zachycují časovou derivaci statických příznaků a jsou nazývány **delta** koeficienty. V praxi se používají i koeficienty druhého řádku, nazývané delta-delta nebo **akcelerační**.

Dynamické koeficienty se obvykle počítají najednou pomocí matice. První řádek odpovídá delta koeficientu a druhý řádek akceleračnímu. Matice je převzata ze zdrojových kódů BSAPI.

$$\begin{bmatrix} 0 & 0 & -0.2 & -0.1 & 0 & 0.1 & 0.2 & 0 & 0 \\ 0.04 & 0.04 & 0.01 & -0.04 & -0.1 & -0.04 & 0.01 & 0.04 & 0.04 \end{bmatrix} \tag{3.4}$$

Výhodou těchto koeficientů je jejich částečná nezávislost na konstantních charakteristikách spektra, které nesouvisí s řečí. Na druhou stranu postrádají některé důležité vlastnosti a nejsou samostatně vhodné pro rozpoznávání. Proto se používají v kombinaci se statickými koeficienty, například s MFCC. [2]

Podrobnější informace lze nalézt v [4].

3.5 Detekce řečových segmentů

Pro detekci řeči ve zvuku se používá anglický termín Voice activity detection, zkráceně VAD. Úkolem VAD je co nejpřesněji rozpoznat, kde se ve zvukové nahrávce nachází řeč a kde se nachází pouze okolní ruch. Tato činnost je velmi důležitá pro práci rozpoznávače, protože vzhledem k důrazu na maximalizaci přesnosti rozpoznávání, jsou příznaky voleny tak, aby co nejlépe reprezentovaly lidskou řeč. Jakýkoliv segment obsahující cokoliv jiného než řeč může systém zmást. Navíc zbytečné provádění algoritmů nad daty, která nemají žádnou vypovídající informaci, plýtvá výpočetním výkonem.

Systémy VAD se chovají nejčastěji binárně. Jejich výstupem je pouze to, zda je zvukový segment řečový či nikoliv. Z toho vyplývá nutnost dobrého nastavení systému. Pokud bude příliš tolerantní, projdou dále neřečové segmenty. Naopak příliš citlivé VAD způsobí zahození řečových segmentů a tedy i důležitých informací pro rozpoznávač.

Většina systémů pracuje s energií zvukového signálu. Bývá stanoven práh, který určuje nad jakou energetickou hladinu se jedná o řeč. Řada systémů má práh určený ne jako absolutní hodnotu, ale jako koeficient, který udává mírů odstupu řečového signálu od ruchu v pozadí. Z toho vyplývá nutnost stanovení hodnoty energie segmentů obsahujících pouze

okolní ruch. Na kvalitě algoritmu stanovujícího tuto energie závisí schopnost systému adaptovat se na změny okolních ruchů. Je zřejmé, že systémy pracující s konstantním prahem toto nedokáží.

Další informace viz [6].

3.6 Vyhodnocení skóre mluvčího

Pro stanovení podobnosti hlasu mluvčího a hlasu na nahrávce na základě získaných příznaků se využívá více metod. V posledních letech je asi nejpoužívanější metodou modelování pomocí směsice Gaussových hustotních funkcí (český název viz [4]), anglicky Gaussian Mixture Models, zkráceně GMM. Výstupem tohoto systému je skóre, které udává jak moc je nahrávka podobná danému mluvčímu v databázi.

Skóre je běžně vyjádřeno jako tzv. likelihood ratio[1], což je podíl likelihood pozitivní hypotézy (skóre mluvčího) lomeno likelihood záporné hypotézy (skóre všech ostatních mluvčích). V praxi se navíc používá zlogaritmovaná varianta, která se počítá jako rozdíl logaritmů těchto dvou hodnot.

Existují dva možné způsoby vypočítání skóre ostatních mluvčích. Jednou variantou je kombinace získaných likelihood všech ostatních natrénovaných mluvčích v databázi. Tento postup je však náročný, protože vyžaduje pro každého mluvčího uchování seznamu ostatních mluvčích. Druhou variantou je vytvoření univerzálního modelu pozadí, které poslouží pro všechny mluvčí. Tato varianta je v současné době asi rozšířenější a je dále popsána v této práci.

Všechny informace v sekci o vyhodnocení skóre mluvčího jsou čerpány z [5].

3.6.1 Směs Gaussových hustotních funkcí

GMM je nejpoužívanější metodou pro získání skóre při rozpoznávání mluvčího kdy neznáme text, který mluvčí pronáší [5]. Tento model, jak již vyplývá z názvu, modeluje vstupní data pomocí mnoha Gaussových hustotních funkcí. Výsledné skóre je ovlivňováno tím, jak moc dobře jsou data tímto modelem reprezentována a je počítáno jako vážený součet jednotlivých Gaussových funkcí:

$$p(x \mid \lambda) = \sum_{i=1}^{M} w_i p_i(x) \tag{3.5}$$

kde M je počet Gaussových funkcí, w_i je váha funkce a $p_i(x)$ je Gaussova funkce:

$$p_i(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(x-\mu_i)'(\Sigma_i)^{-1}(x-\mu_i)\} \tag{3.6}$$

Parametry Gaussových funkcí jsou označeny jako $\lambda = \{w_i, \mu_i, \Sigma_i\}$ pro $i = 1, ..., M$. Pro váhy funkcí musí platit $\sum_{i=1}^{M} w_i = 1, w_i > 0$

V praxi také platí, že pro reprezentaci Gaussových funkcí se používají pouze diagonální kovarianční matice Σ. Nevýhodou těchto matic je, že pro modelování korelovaných příznaků je nutné použít více Gaussových funkcí, ale tuto nevýhodu vyváží mnohem menší paměťová náročnost a méně kroků výpočtů než při využití plných kovariančních matic.

[1] Termín likelihood se dá přeložit jako věrohodnost, ale není jej zvykem překládat. Proto ho zde, stejně jako připojená slova, uvádím v anglické podobě.

3.6.2 Univerzální model pozadí

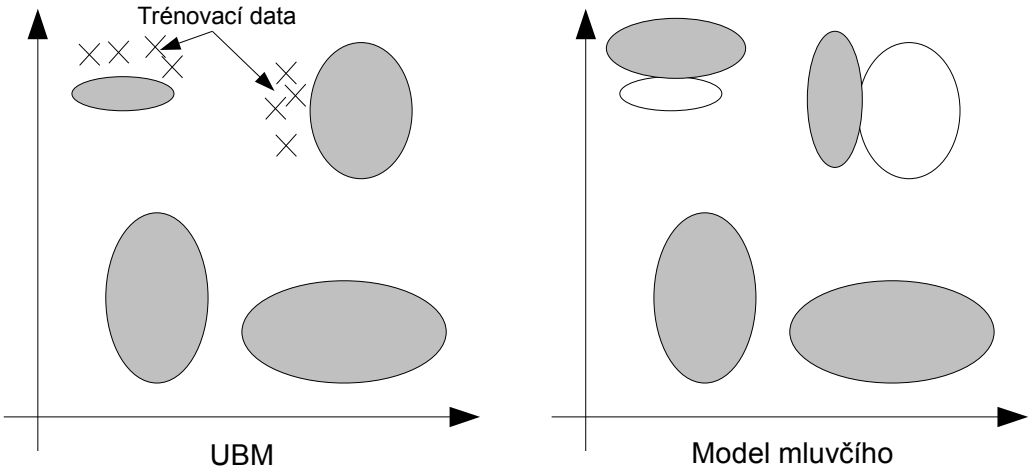
Univerzální model pozadí, anglicky Universal Background Model, zkráceně UBM, je model snažící se reprezentovat každého mluvčího. Jde o velký GMM, který je natrénován na dostatečně velkém množství zvukových dat. Tato data jsou tvořena kombinací zvukových nahrávek od různých mluvčích. Důležitým požadavkem při trénování je, aby vstupní data rovnoměrně reprezentovala potenciální uživatele systému. Pokud by byla některá podskupina uživatelů zastoupena více, odrazilo by se to ve výsledném chování systému. Proto je správné trénování UBM důležitým prvkem při ladění systému. Trénování UBM je nejčastěji prováděno iterativně pomocí tzv. EM algoritmu (viz [2]).

3.6.3 Adaptace modelu na řečníka

Model řečníka lze vytvořit několika způsoby. Základní variantou je, obdobně jako u UBM, natrénování GMM pomocí EM algoritmu. Zde však vznikají problémy, protože je třeba odhadnout velké množství parametrů na základě malého počtu dostupných dat. Proto se dnes více využívá jiná varianta, kdy se adaptuje UBM tak, aby reprezentoval konkrétního mluvčího.

Adaptace je prováděna pomocí Bayesovského trénování, které je také označováno jako MAP z „maximum a posteriori“. Principem je změna parametrů jednotlivých Gaussových funkcí tak, aby lépe popisovaly data mluvčího. Algoritmus postihuje nejvíce ty parametry, ke kterým máme od mluvčího nejvíce dat. Toho je dosaženo pomocí faktoru relevance, který je počítán z příchozích dat a pevné hodnoty dané parametrem. Věříme, že změna parametrů popisuje pouze mluvčího, ve skutečnosti však popisuje i kanál. Toto však v našem případě neřešíme.

Algoritmus je znázorněn na obrázku 3.1 a jeho podrobnější popis lze najít v [5].



Obrázek 3.1: Adaptace UBM na mluvčího, převzato z [5]

Kapitola 4

Multiplatformní aplikace

V oblasti počítačových systémů vždy vznikaly a stále vznikají vzájemně nekompatibilní zařízení. Nekompatibilita bývá způsobena více důvody, ať již technologickými, kdy je například nutné odstranit zastaralé součásti systému, právními, kdy není možné využít duševního vlastnictví konkurence, a dalšími více či méně podstatných důvody. Zároveň je však požadováno, aby vyvíjená softwarová aplikace fungovala na co nejvíce cílových zařízeních, aniž by bylo potřeba upravovat zdrojový kód aplikace.

Aplikace je multiplatformní, pokud je schopna pracovat na více rozdílných počítačových systémech. Může jít o různé hardwarové architektury případně operační systémy.

Multiplatformity je většinou dosahováno vkládáním vrstev mezi vyvíjenou aplikaci a systém, na kterém má pracovat. Existuje mnoho různých knihoven[1], které vytváří abstrakce nad nižšími vrstvami a tím umožňují jednodušší vytváření multiplatformních aplikací. Mezi nejznámější a nejkomplexnější patří knihovny GTK+, WxWidgets a níže zmíněný Qt framework. Tento postup vytváření abstrakčních knihoven má však i své nevýhody, protože se často stává, že v rámci zjednodušení kompatibility se vytváří další a další knihovny nad nižšími vrstvami a tak vzniká řetěz mnoha knihoven, které mohou ve svém důsledku způsobit zásadní zpomalení systému. Multiplatformní knihovny jsou ale často využívány, protože jejich výhody dalece převyšují tuto nevýhodu.

Význam multiplatformních aplikací se navíc v poslední době zvyšuje s příchodem výkonných kapesních počítačů a chytrých mobilních telefonů. Zvětšují se také uživatelské nároky na to, jaké aplikace by měly být dostupné například na mobilním telefonu. Tomuto trendu se přizpůsobují softwaroví vývojáři i tvůrci multiplatformních knihoven, které tak umožňují bezproblémový provoz aplikace na velmi rozdílných zařízeních.

4.1 Qt framework

Qt je multiplatformní toolkit pro vývoj aplikací a uživatelského rozhraní. Je napsán v programovacím jazyce C++, do kterého navíc přináší vlastní rozšíření známé z jiných jazyků. Jedná se především o systém signálů a slotů, jenž je realizován pomocí vlastního preprocesoru známého jako kompilátor meta objektů (moc).

Qt vzniklo v roce 1991 jako dílo Haavarda Norda a Eirika Chambe-Enga. O vývoj se následně starala společnost Quasar Technologies, později známá jako Trolltech. V roce

[1] Samozřejmě existují i jiné způsoby umožňující multiplatformní programování. Důležité jsou zejména multiplatformní programovací jazyky, bez kterých by toto nebylo vůbec technicky možné. Nejznámějšími příklady jsou například jazyky C, C++ nebo Java.

2008 byla tato společnost koupena finskou firmou Nokia, která je nyní hlavním vývojářem toolkitu [10].

Qt toolkit už zdaleka není jen knihovnou pro vytváření grafického rozhraní aplikací. Jeho možnosti se s každou další verzí rozšiřují a tak lze říci, že při vytváření většiny aplikací se programátor obejde bez používání jakýchkoliv systémových volání a vystačí s tím co nabízí Qt. Od verze 4.5 je navíc k dispozici multiplatformní integrované vývojové prostředí zvané Qt Creator, které usnadňuje vývoj aplikací včetně debugování.

V současné době (květen 2010) je aktuální verze 4.6, která vyšla v prosinci 2009. Tato verze je dostupná pro platformy Mac OS X, Windows, Linux/X11, Embedded Linux, Windows CE/Mobile, Symbian a Maemo a je licencována pod licencemi GPL, LGPL nebo vlastní proprietární licencí. Novinkou v této verzi je mimo jiné zavedení modulu QtMultimedia, který obsahuje nízkoúrovňový přístup ke zvukovému subsystému počítače. Tato nová vlastnost je důležitým prvkem použitým v aplikaci vyvíjené v rámci této bakalářské práce. [9]

I když je nativním jazykem toolkitu C++, existují rozhraní, které jej umožňují používat i v dalších jazycích. Ze známějších lze uvést například Javu a Python. Nejznámější projektem vystavěným nad tímto toolkitem je pravděpodobně unixové uživatelské rozhraní KDE.

Kapitola 5

Vývoj aplikace

Základním požadavkem na aplikaci, zmíněným i v samotném názvu bakalářské práce, byla schopnost běhu na různých platformách. Na základě minulé kapitoly bylo určeno, že aplikace bude postavena nad knihovnou Qt a jako implementační jazyk bude využito C++. Dále bylo rozhodnuto, že aplikace nebude implementovat algoritmy pro rozpoznávání řeči, ale využije služeb knihovny BSAPI. Jako překladač byl vybrán GCC, protože je k dispozici na velkém množství platforem a je nativním překladačem pro BSAPI.

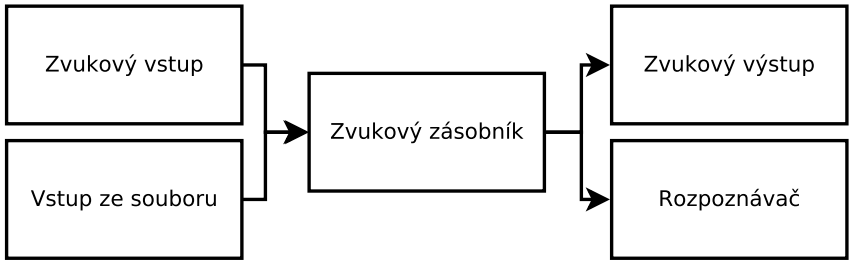
Samotný vývoj probíhal na operačním systému Windows Vista v integrovaném vývojovém prostředí Qt Creator. Zároveň probíhalo testovaní funkčnosti na linuxové distribuci Fedora 13.

Tato kapitola se dále zabývá jednotlivými částmi implementované aplikace.

5.1 Zvukový vstup a výstup

Komponenta pro práci se zvukem je implementována pomocí třídy `CAudio`. Její hlavní činností je zachytávání a přehrávání zvuku, což je realizováno pomocí rozhraní z modulu `QtMultimedia` knihovny Qt. Další důležitou rolí této komponenty je komunikace s rozpoznávačem. Tomu zajišťuje vstupní data a získává od něj skóre, které předává dalším komponentám systému. Schéma propojení jednotlivých částí komponenty je znázorněno na obrázku 5.1.

Při použití komponenty je nutné nastavit příznak určující zda je požadováno trénování nového modelu řečníka nebo rozpoznávání. Tento příznak zásadně ovlivňuje chování komponenty, a proto nelze toto nastavení v průběhu změnit.



Obrázek 5.1: Schéma zvukového vstupu a výstupu

Hlavní částí komponenty je **zvukový zásobník**, který umožňuje uchování zvuku za-

chyceného pomocí zvukové karty nebo nahraného ze souboru. Ze zásobníkem jsou potom propojeny veškeré další části zvukové komponenty, které jej používají jako úložiště nebo jako zdroj zvukových dat. Zásobník je realizován pomocí třídy `QBuffer` umožňující pracovat s pamětí jako se souborem s náhodným přístupem. Toho využívají různé částí komponenty, kdy některé pracují se souborem a některé s pamětí.

Zachytávání a přehrávání zvuku je realizováno ve **formátu** PCM. Parametry jsou nastaveny pomocí třídy `QAudioFormat` tak, aby odpovídaly požadavkům rozpoznávače. Frekvence je nastavena na 8000Hz, velikost jednoho vzorku je 16 bitů a pracuje se s jedním kanálem zvuku. Data jsou ukládána celočíselně znaménkově jako little endian. Do rozpoznávače jsou posílána po rámcích o délce 400 vzorků, což odpovídá 800 bytům nebo 50 milisekundám.

Načtení ze souboru je nejjednodušším způsobem získání dat pro zpracování v systému. Data jsou načtena pomocí třídy `QFile` ze souboru typu RAW (surové PCM data) do zásobníku a následně jsou zpracována rozpoznávačem. I když jsou data načtena najednou, do rozpoznávače jsou posílány po rámcích z důvodu jednotného chování systému.

Zachytávání zvuku je řešeno použitím třídy `QAudioInput`. Nejprve je inicializován rozpoznávač a je otestováno, zda vstupní zařízení podporuje vyžadovaný formát zvuku. Následně je zahájeno zachytávání do zvukového zásobníku. Navíc je každých 50 milisekund odeslán rámec do rozpoznávače. Zachytávání lze přerušit odesláním signálu `stop`. Vzhledem k současnému zápisu a čtení ze zvukového zásobníku, je zachytávání prováděno pomocí souborového rozhraní a čtení je realizováno nezávisle přímo z odpovídajícího místa v paměti.

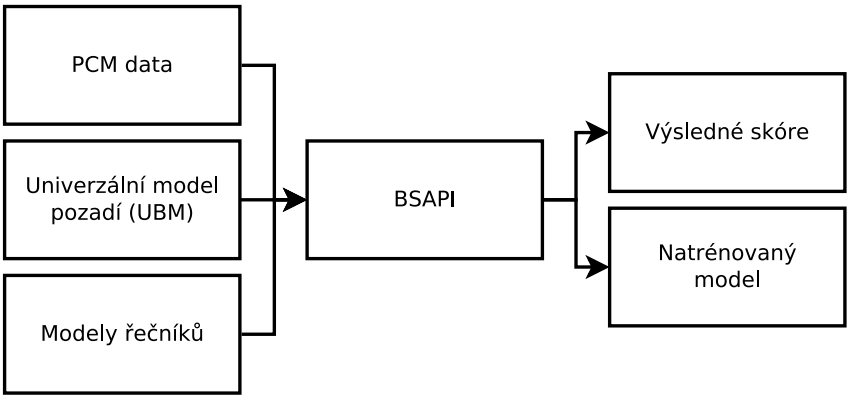
Přehrávání zvuku je vzhledem k použitému rozhraní velmi podobné zachytávání. Základem je třída `QAudioOutput` zajišťující zvukový výstup. Nejprve je, obdobně jako u zachytávání, zkontrolována podpora formátu a následně je spuštěno přehrávání, které je přerušitelné signálem nebo vyčerpáním zvukových dat v zásobníku.

Tato komponenta není vzhledem k množství prováděné práce příliš rozsáhlá. Toho je dosažena díky vysoké míře abstrakce zvukového vstupu a výstupu v Qt. Bohužel vzhledem k tomu, že modul `QtMultimedia` byl do Qt přidán teprve v aktuální verzi, nejsou všechny části úplně odladěny a na některých systémech dochází k pádům. Například na testovacím počítači s Windows Vista docházelo k ojedinělým pádům při zastavení zachytávání. Tento problém však nebylo možné zreprodukovat na jiných systémech ani verzích systému. Také na systému Fedora 13 docházelo k pádům. V okamžiku testování podpory formátů na některých typech zvukových karet došlo k nesplnění důležité podmínky a program byl ukončen. Obě chyby jsou však ojedinělé a výhody jednoduchosti a přenositelnosti rozhraní je dalece převažují. Navíc lze očekávat, že chyby budou v budoucích verzích knihovny nebo systému opraveny.

5.2 Rozpoznávač

Třída rozpoznávače `CSpeech` zapouzdřuje BSAPI pro použití v systému. Tato třída je používána výhradně z komponenty pro zvukový vstup a výstup a poskytuje dvě základní činnosti – **rozpoznávání řečníka** a **vytváření modelů** pro nové řečníky. Požadovanou činnost je třeba specifikovat, stejně jako v předchozím případě, již při vytváření objektu třídy a nelze ji později změnit. Rozpoznávač popisuje schéma na obrázku 5.2.

Chování třídy je pro obě poskytované činnosti velmi podobné a liší se pouze v detailech. Během inicializace je nejprve načtena konfigurace BSAPI a následně univerzální model pozadí. Dále jsou získány ukazatele na potřebné bloky rozpoznávacího systému. V režimu



Obrázek 5.2: Schéma rozpoznávače

rozpoznávání jsou navíc načteny jednotlivé modely mluvčích. Ve druhém případě je zahájeno trénování nového modelu.

Data jsou objektu zasílána pomocí metody `addWave`, která je předá prvnímu bloku v BSAPI. Pokud probíhá rozpoznávání, lze kdykoliv načíst aktuální skóre pomocí metody `getScores`. Počet modelů, které jsou k dispozici lze zjistit pomocí metody `getModelsCount`. V režimu trénování modelů je možné použít pro uložení metodu `saveModel`.

Spolupráce této třídy s BSAPI, konfigurační soubor, použité UBM a následná úprava a normalizace skóre jsou založeny na příkladu dodávaném s použitou verzí knihovny.

5.3 Uživatelské rozhraní

Uživatelské rozhraní je, stejně jako celá aplikace, založeno na Qt frameworku. Dále jsou použity ikony vytvořené v rámci projektu Tango[1], které jsou distribuovány jako public domain.

Samotná aplikace je rozdělena do několika dialogových oken, která jsou popsána v této sekci. Pro vytváření grafického vzhledu oken bylo využito Qt Designeru, který existuje samostatně i jako součást vývojového prostředí Qt Creator.

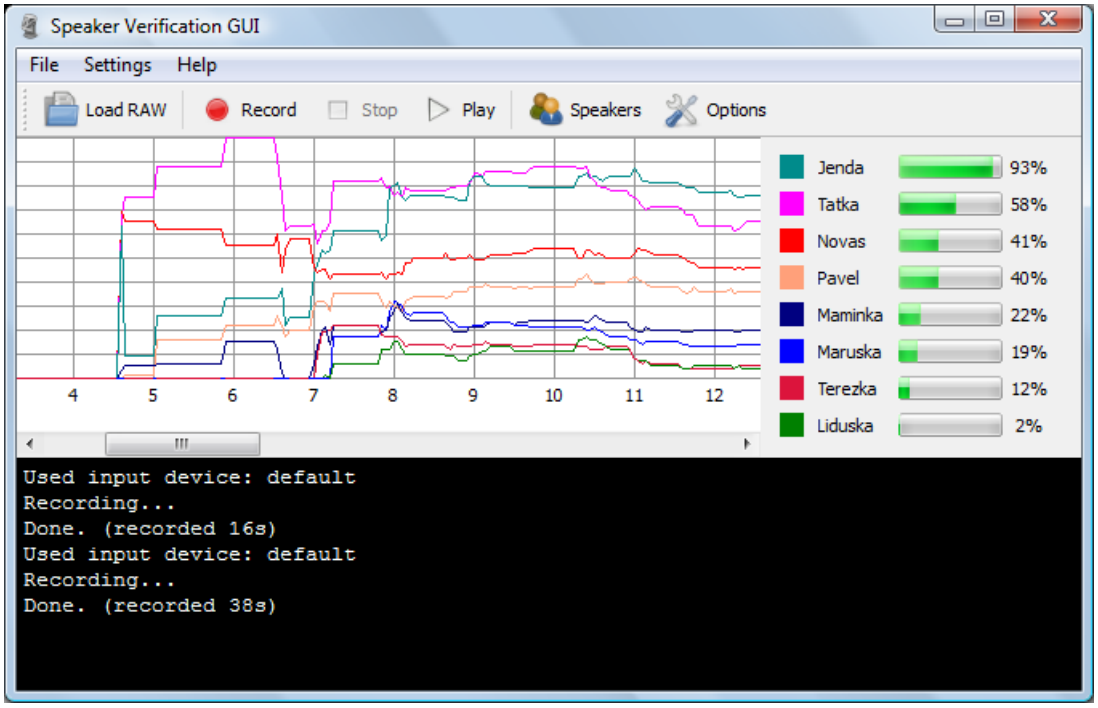
5.3.1 Hlavní okno aplikace

Hlavní okno je část aplikace sloužící především pro samotné rozpoznávání mluvčího. Pro tyto účely je zde zobrazován **graf vývoje skóre** jednotlivých mluvčích a panel obsahující **aktuální skóre**. Rovněž je zde zobrazeno informační okno zobrazující aktuální průběh rozpoznávání. Práce se zvukem a rozpoznávání mluvčího je řešeno voláním metod objektu třídy `CAudio`. Další možnosti aplikace jsou k dispozici pomocí nabídky a nástrojové lišty. Hlavní okno je zobrazeno na obrázku 5.3.

Graf vývoje skóre

Graf vývoje skóre je nejdůležitější vizualizační komponentou aplikace. Jelikož Qt neobsahuje v aktuální verzi grafovou komponentu vhodnou k vizualizaci skóre řečníků, byla tato

[1] Více informací o projektu a možnost stažení ikon lze nalézt na webové straně projektu http://tango.freedesktop.org/.



Obrázek 5.3: Hlavní okno aplikace

komponenta založena na třídách `QGraphicsView` a `QGraphicsScene`. Název implementované třídy je `CGraph`. Mimo samotného zobrazování umožňuje export vizualizovaných dat do souborů ve formátu PNG a SVG.

Pro export dat je využito třídy `QPainter`, která zajišťuje kreslení do jiných grafických tříd. Ukládání do formátu PNG je realizováno třídou `QImage` a ukládání do SVG je řešeno třídou `QSvgGenerator`.

Vývoj skóre každého mluvčího je spravován třídou `CPath`. Ta uchovává informace o průběhu skóre pomocí čar typu `QGraphicsLineItem` a usnadňuje jejich vkládání metodou `addLine`. Vytváření nových čar je prováděno na základě předání aktuálního času a nové hodnoty skóre.

Vkládání nového skóre do grafu je zajištěno pomocí metody `AddNode`, která zanáší do grafu aktuální údaje a zapouzdřuje použití třídy `CPath`. Zároveň se stará o posunutí zobrazované části grafu a aktualizaci velikosti mřížky pomocí metody `renderGrid`.

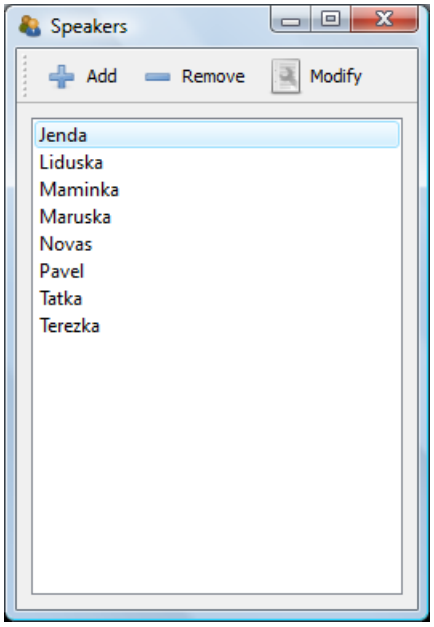
Panel aktuálního skóre

Panel zobrazuje mluvčí seřazené podle aktuálně dosaženého skóre a je realizován třídou `CScores`. Jako první je umístěn mluvčí, který je ohodnocen nejvyšším skóre, a je tedy nejpodobnější právě zpracovávané nahrávce. Počet zobrazených mluvčí v seznamu lze ovlivnit pomocí nastavení.

Před jménem každého mluvčího je zobrazena barva, kterou je vykreslen příslušný průběh skóre v grafu. Skóre je vedle procentuálního ohodnocení také graficky znázorněno ukazatelem průběhu. Po najetí myši na mluvčího je zobrazeno v plovoucím desetinném tvaru od 0 do 1.

5.3.2 Správa mluvčích

Správa mluvčích je realizována jednoduchým dialogovým oknem implementovaným pomocí třídy `CSpeakers`. Nejvýznamnější částí je seznam mluvčích, pro které je k dispozici natrénovaný model v adresáři daném konfigurací. Pomocí nástrojové lišty lze mluvčí přidávat, upravovat a mazat. Vzhled dialogového okna je na obrázku 5.4.



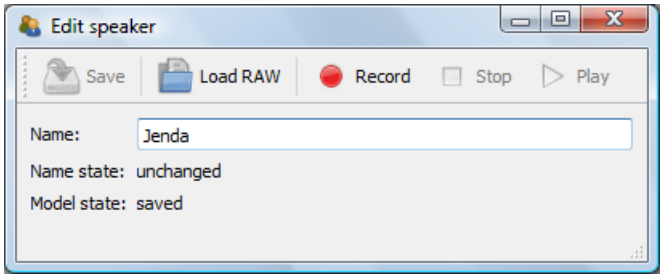
Obrázek 5.4: Okno správy mluvčích

5.3.3 Trénování mluvčího

Dialogové okno pro trénování modelu mluvčího je implementováno třídou `CSpeaker` a je vždy voláno z okna správy mluvčích. Tento dialog a hlavní okno jsou dvě místa aplikace využívající zvukový vstup a výstup. Veškerá práce se zvukem je, stejně jako v hlavním okně, řešena pomocí objektu třídy `CAudio`, který je však inicializován v módu trénování. Činnost okna je rozdělena na dvě činnosti: trénování nového mluvčího a úpravu vlastností již existujícího mluvčího. Rozdíly jsou však minimální a týkají se pouze inicializace a ukládání modelu. Dialogové okno je na obrázku 5.5.

Před zobrazením dialogu je nejprve inicializován zmíněný objekt pro zvukový vstup a výstup. Následně je nastaven titulek okna podle toho, jaká činnost se má provádět, a je nahrán případný název existujícího modelu. Při změně názvu nebo natrénovaného modelu je toto oznámeno pomocí příslušné stavové informace. Veškeré změny jsou uloženy až při stisknutí ukládacího tlačítka.

Ukládání je prováděno ve třech krocích. Nejprve je zjištěno, zda se jedná o nový model a zda již model se stejným názvem neexistuje. Následně je v případě úpravy existujícího modelu a změny názvu provedeno přejmenování modelu a ve třetím kroku je uložen změněný model pod novým názvem.



Obrázek 5.5: Okno trénování mluvčího

5.3.4 Konfigurační dialog

Tento dialog je určen pro změnu nastavení uloženého v objektu třídy `CConfig` zmíněné níže. Dialog je implementován v třídě `COptions`.

Při požadavku na vytvoření dialogu, jsou nejprve zjištěna přítomná vstupní a výstupní zvuková zařízení, která jsou poté umístěna do roletových nabídek dialogu. Následně je dialog vyplněn aktuálním nastavením.

Zadávané soubory a adresáře jsou ihned kontrolovány jestli existují. Pokud soubor či adresář neexistuje, je jeho název zobrazen červenou barvou. Při ukládání nastavení je navíc za název adresáře přidáno případné ukončující lomítko.

5.4 Konfigurace

Veškerá konfigurace aplikace mimo nastavení BSAPI je reprezentována třídou `CConfig`. Jde o velmi jednoduchou třídu, která zapouzdřuje potřebné informace a poskytuje možnost načítání a ukládání do XML souboru. To je řešeno pomocí XML rozhraní poskytovaného Qt. Pro zápis se využívá třída `QXmlStreamWriter` a pro načítání `QXmlStreamReader`. Třída `CConfig` pro tyto účely poskytuje metody `save` a `load`.

Objekt této třídy je vlastněn hlavním oknem. Ten je jako parametr konstruktoru předáván všem třídám, které vyžadují přístup ke konfiguraci. Změnu nastavení lze provést pouze pomocí dialogového okna pro úpravu konfigurace nebo pomocí změny konfiguračního souboru s názvem `config.xml`. Pokud není při startu tento soubor nalezen, je uživatel dotázán, zda má být vytvořena standardní konfigurace.

V konfiguraci jsou uloženy následující údaje:

- Cesta k adresáři obsahujícího modely mluvčích
- Cesta k UBM souboru
- Jméno vstupního zvukového zařízení
- Jméno výstupního zvukového zařízení
- Cesta ke konfiguračnímu souboru BSAPI
- Maximální počet zobrazených mluvčí při rozpoznávání mluvčího

Kapitola 6

BSAPI

Knihovna BSAPI byla vyvinuta skupinou Speech@FIT na Ústavu počítačové grafiky a multimédií FIT VUT. Následující dva odstavce představují tuto knihovnu a jsou překladem z její dokumentace (viz [8]).

Brno Speech Core (BSCORE) je soustava stavebních bloků pro jednoduchou a rychlou stavbu rozpoznávačů řeči. Implementuje celou řadu algoritmů od čtení zvukových souborů, vstupu z mikrofonu a zpracovávání seznamu souborů přes parametrizaci, transformaci příznaků, klasifikaci a dekódování až po rozpoznávání fonémů. Tyto algoritmy jsou určeny pro rozpoznávače řeči, zachytávání klíčových slov a identifikaci jazyka či řečníka. Implementačním jazykem je C++.

Brno Speech Application Interface (zkráceně BSAPI) je rozhraní umožňující spojení BSCORE a jiného softwaru. Aplikační rozhraní je objektově orientované a každý algoritmus je tvořen vlastní třídou. Ty jsou zpřístupněny pomocí rozhraní, implementovaných jako čistě virtuální třídy. Jejich instance lze vytvářet voláním funkce `BSAPICreateInstance()`.

V aplikaci bylo využito BSAPI ve verzi určené pro online zpracování signálu a optimalizované pro použití v mobilním telefonu. Tato verze je vyvinuta s důrazem na nízké hardwarové požadavky, a proto umožňuje práci i na méně výkonných počítačích.

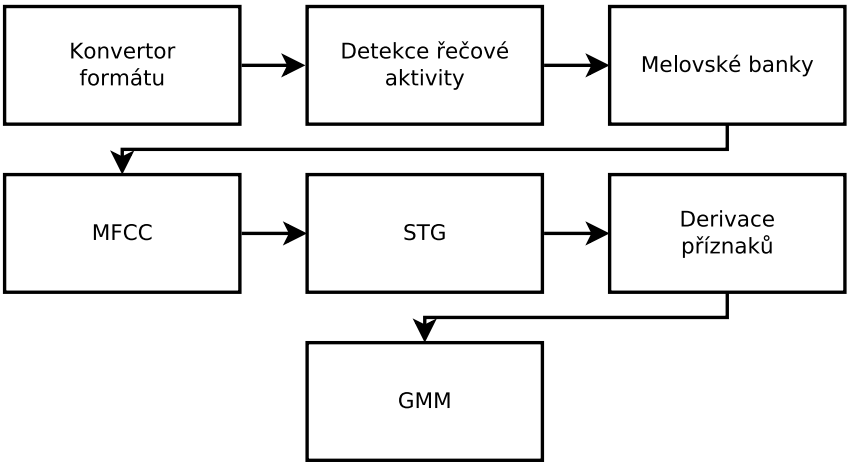
6.1 Konfigurace knihovny

Nastavení jednotlivých bloků a jejich vzájemné propojení je určeno pomocí textového konfiguračního souboru, který je načítán za běhu aplikace. Použité bloky jsou popsány v této sekci a jejich propojení je znázorněno na obrázku 6.1. Podrobný popis prováděných operací je v kapitole 3. Informace jsou čerpány z demonstrační konfigurace a zdrojových kódů knihovny.

Konvertor formátu

Data získaná pomocí PCM jsou obvykle celočíselná. V rámci této práce implementovaná aplikace pracuje se 16-bitovým znaménkovým typem reprezentovaným jako `short`. Naproti tomu BSAPI pracuje s čísly v plovoucí řádové čárce, které jsou datového typu `float`. Konvertor formátu je potom využit pro převod z aplikačního formátu do interní reprezentace BSAPI.

Konvertor dále dokáže provádět základní operace se signálem, ty však nejsou v rámci aplikace využity.



Obrázek 6.1: Schéma propojení bloků BSAPI

Detekce řečové aktivity

Detekce řečové aktivity je prováděna pomocí detektoru pracujícího pouze na základě energie signálu. Z důvodu jednoduchosti a rychlosti je práh nastaven absolutně. Číselná hodnota tohoto prahu je 13.

Melovské banky a MFCC

Výpočet MFCC koeficientů je rozdělen do dvou bloků. První blok provádí celý výpočet mimo diskrétní kosinovou transformaci. Tak je získáno 15 čísel, tzv. Melovských koeficientů. Druhý blok dokončuje výpočet MFCC provedením DCT. Výsledkem je 12 koeficientů a nultý koeficient, který je počítán samostatně.

Bloky jsou nastaveny pomocí řady parametrů. Vzorkovací frekvence je 8000 Hz, velikost rámce 200 vzorků a posun 80 vzorků. Frekvenční rozsah je stanoven jako interval od 64 Hz do 3800 Hz. Dalšími parametry jsou výše uvedené velikosti výstupních vektorů.

STG

Krátkodobá gaussianizace pracuje s oknem obsahujícím 150 prvků v minulosti a 150 prvků v budoucnosti. Celé okno je tak složeno z 301 mikrosegmentů, což odpovídá přibližně 3 sekundám zvukového záznamu.

Derivace příznaků

Blok derivace příznaků je nastaven tak, aby produkoval dynamické koeficienty 1. a 2. řádu.

GMM

Tento blok není propojen s ostatními přímo pomocí konfigurace BSAPI, ale je tak učiněno až v samotné aplikaci. Výstupem tohoto bloku jsou skóre jednotlivých mluvčích nebo nový natrénovaný model. Pro adaptaci mluvčího jsou využívány pouze střední hodnoty. Variance a váhy tak nejsou brány v potaz. Výpočet skóre probíhá po rámcích a koeficient faktoru relevance T pro MAP adaptaci je nastaven na hodnotu 19.

6.2 Univerzální model pozadí

Pro rozpoznávání je použit univerzální model pozadí dodaný s knihovnou. Jsou o něm známy následující informace.

Model je natrénován tak, aby byl nezávislý na pohlaví mluvčího a je reprezentován jako směsice Gaussových funkcí s diagonálními kovariančními maticemi. Trénování proběhlo na telefonních datech z databází NIST-SRE-2004 a 2005. Tyto databáze obsahovaly 171 hodin nahrávek od 376 žen a 138 hodin nahrávek od 294 mužů. Při iterativním EM trénování bylo použito prahování variancí, kde je každá nová odhadnutá variance menší než stanovený práh nahrazena tímto prahem. Práh byl zvolen jako desetina průměru variancí z předchozí iterace.

Kapitola 7

Experimenty

V rámci práce byly provedeny dva experimenty snažící se zjistit limity aplikace a spolehlivost rozpoznávání.

7.1 Hardwarová náročnost aplikace

Prvním experimentem byla snaha o zjištění hardwarové náročnosti aplikace. Testování proběhlo na počítači s procesorem Intel Core 2 Duo T5870, 4 GB operační paměti a s operačními systémy Windows Vista a Fedora 13. Jako metrika bylo zvoleno procentuální zatížení procesoru při rozpoznávání řečníka pomocí zvuku z mikrofonu. Pod systémem Windows byl tento údaj odečítán vždy po 20 sekundách rozpoznávání. Při testování na systému Fedora byla kvůli kolísání zatížení použita přibližná hodnota v okolí tohoto času. Test byl zopakován pro různé počty řečníků v databázi. Výsledky jsou zaznamenány v tabulce 7.1.

Počet řečníků	17	34	68	136	272
Windows Vista	15%	20%	40%	50%	50%
Fedora 13	22%	24%	31%	48%	62%

Tabulka 7.1: Zátěž procesoru při rozpoznávání pro daný počet řečníků

Z naměřených hodnot vyplývá skutečnost, že při zvyšování počtu řečníků roste zatížení systému. Na systému Windows však přestalo stoupat zatížení procesoru po dosažení hranice 50%. Tento jev se vyskytl i přesto, že aplikace zatížila obě jádra procesoru. Zcela jistě jde o chybu měření, protože odezva aplikace se při zvyšování počtu řečníků v databázi zhoršovala.

Vzhledem k problému při měření zátěže ve Windows se jako reprezentativnější jeví hodnoty naměřené pod Fedorou. I ty jsou ale jen ilustrační, protože dalšími experimenty bylo zjištěno, že pro zachování plynulého vykreslování grafu lze rozpoznávat s databází o přibližné maximální velikosti 200 řečníků. Při použití takové databáze se sice zatížení procesoru pohybuje pod hranicí 60%, ale jakékoliv další zvětšování má významný vliv na práci online rozpoznávače.

7.2 Spolehlivost rozpoznávače

Dalším experimentem bylo otestování spolehlivosti rozpoznávače. Trénování databáze proběhlo interním mikrofonem notebooku v tichém prostředí. Tak bylo vytvořeno 17 modelů

na základě nahrávek o přibližné délce jedné minuty.

Data pro testování byla vytvořena pomocí nahrávek od šesti mluvčích. Jejich hlasy byly zároveň obsaženy v natrénované databázi. Každý mluvčí nahrál čtyři nahrávky o délce 10 až 20 sekund, které obsahovaly stejnou promluvu, ale byly pořízeny jiným způsobem. Nahrávání probíhalo pomocí interního mikrofonu notebooku a USB telefonu. Tyto mikrofony byly využity pro získání nahrávek v tichém a hlučném prostředí. Hlučné prostředí bylo vytvořeno hlasitou hudbou z rádia. Získané skóre je ve formě procent zaznamenáno v tabulce 7.2.

Interní mikrofon notebooku – ticho

	Mluvčí A	Mluvčí B	Mluvčí C	Mluvčí D	Mluvčí E	Mluvčí F
Vlastní skóre	55%	52%	88%	71%	57%	50%
Nejlepší cizí	50%	74%	52%	50%	35%	38%

Interní mikrofon notebooku – hluk

	Mluvčí A	Mluvčí B	Mluvčí C	Mluvčí D	Mluvčí E	Mluvčí F
Vlastní skóre	55%	15%	37%	19%	22%	43%
Nejlepší cizí	36%	20%	28%	15%	19%	39%

Mikrofon USB telefonu – ticho

	Mluvčí A	Mluvčí B	Mluvčí C	Mluvčí D	Mluvčí E	Mluvčí F
Vlastní skóre	29%	41%	69%	59%	85%	22%
Nejlepší cizí	38%	51%	36%	40%	47%	19%

Mikrofon USB telefonu – hluk

	Mluvčí A	Mluvčí B	Mluvčí C	Mluvčí D	Mluvčí E	Mluvčí F
Vlastní skóre	48%	23%	59%	54%	65%	22%
Nejlepší cizí	38%	32%	43%	25%	22%	24%

Průměrný odstup od prvního cizího mluvčí

	Mluvčí A	Mluvčí B	Mluvčí C	Mluvčí D	Mluvčí E	Mluvčí F
Odstup skóre	6,25	-11,5	23,5	18,25	26,5	4,25

Tabulka 7.2: Skóre testovacích mluvčí

Vlastním skóre je míněno skóre mluvčího aktuálně pronášejícího řeč. *Nejlepší cizí* je skóre mluvčího s nejvyšším skóre mimo aktuálního.

Ze získaných skóre lze vyčíst řadu zajímavých informací. Mluvčí C,D a E byli rozpoznáni ve všech případech, mluvčí A a F ve třech případech a mluvčí B nebyl rozpoznán ani v jednom případě. Úspěšnost je tedy 75%. Tento údaj ale není příliš průkazný, protože neproběhlo zkoušení na mluvčích mimo databázi. Tabulka průměrného odstupu ukazuje rozdíl průměrných hodnot vlastních a nejlepších cizích skóre. Lze z ní vyčíst, že nejlépe šlo rozpoznat mluvčího E a nejhůře mluvčího B. Nejlepších výsledků bylo dosaženo v kombinaci mikrofon notebooku a ticho, nejhorší v kombinaci mikrofon notebooku a hluk.

Uvedené měření spolehlivosti rozpoznávání není příliš průkazné a oproti používaným metodám je značně nedokonalé. Zde je však uvedeno pro jednoduchou demonstraci možností systému.

Kapitola 8

Závěr

V rámci bakalářské práce jsem se seznámil s problematikou rozpoznávání řečníka a postupy, které jsou pro tuto činnost v současné době používány.

Hlavním produktem práce bylo vytvoření aplikace demonstrující průběh rozpoznávání řečníka. Ten je znázorněn pomocí grafového výstupu. Aplikace poskytuje možnost uložení výsledného grafu do souboru typu PNG nebo SVG. Dále je umožněno trénování modelů nových mluvčí, editace stávajících a mazání již nepotřebných.

Aplikace byla navržena jako multiplatformní a její funkčnost byla ověřena na operačních systémech Windows a Linux. V rámci vývoje jsem se seznámil s architekturou a konfigurací knihovny BSAPI vyvinuté na ÚPGM FIT VUT. Také jsem využil vlastnosti aktuální verze toolkitu Qt zejména v oblasti zachytávání a přehrávání zvuku.

8.1 Možnosti dalšího vývoje

Navržená aplikace obsahuje spoustu možností pro budoucí rozšíření.

První možností je vylepšení grafové komponenty. Ta je nyní velmi jednoduchá a neumožňuje žádnou složitější práci s grafem. Možným rozšířením by bylo přidání vylepšených možností navigace, například různých stupňů přiblížení, zobrazování hodnot skóre v libovolném čase nahrávky atd.

Aplikace je nyní založena na knihovně BSAPI, která je obalena třídou `CSpeech`. Případné rozšíření by mohlo implementovat všechny potřebné algoritmy samostatně a vyhnout se tak závislosti na této knihovně.

Dalším možným rozšířením by bylo vylepšení práce se zvukem, například přidáním možnosti rozpoznání řečníka z části nahrávky, načítáním jiných zvukových formátů či vykreslením průběhu zvukového signálu.

Vylepšit by se dala i správa modelů řečníků. Nyní jsou identifikováni pouze pomocí názvu souboru modelu. Daly by se přidat další komplexnější informace, které by šly využít při rozpoznávání.

Literatura

[1] Burget, L.: Systémy zpracování řeči. Brno: FIT VUT, přednáška, 19.11.2009.

[2] Gold, B.; Morgan, N.: *Speech and audio signal processing :processing and perception of speech and music*. New York: John Wiley and Sons, 2000, 537 s., iSBN 0-471-35154-7.

[3] Pelecanos, J.; Sridharan, S.: Feature Warping for Robust Speaker Verification. In *2001: A Speaker Odyssey*, Kréta, 2001.

[4] Psutka, J.; Müller, L.; Matoušek, J.; aj.: *Mluvíme s počítačem česky*. Praha: Academia, 2006, 746 s., iSBN 80-200-1309-1.

[5] Reynolds, D. A.; Quatieri, T. F.; Dunn, R. B.: Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 2000.

[6] Sakhnov, K.; Verteletskaya, E.; Simak, B.: Approach for Energy-Based Voice Detector with Adaptive Scaling Factor. *IAENG International Journal of Computer Science*, 2009.

[7] Szöke, I.: Zpracování řečových signálů. Brno: FIT VUT, přednáška, 25.2.2009.

[8] WWW stránky: BSAPI: Brno Speech Application Interface Documentation. `http://www.phonexia.com/docs/bsapi/index.html`.

[9] WWW stránky: Qt - A cross-platform application and UI framework. `http://qt.nokia.com`.

[10] WWW stránky: Wikipedia - Qt (framework). `http://en.wikipedia.org/wiki/Qt_(framework)`.

[11] Černocký, J.: Zpracování řečových signálů. slidy k předmětu, 2004-04-19 [cit. 2010-05-10].

[12] Černocký, J.: *Zpracování řečových signálů – studijní opora*. Brno: FIT VUT, 2006, 128 s.

[13] Černocký, J.: Signály a systémy. slidy k předmětu, 2010-02-04 [cit. 2010-05-10].

Příloha A

Obsah CD

- **bin** - Spustitelné soubory aplikace
- **src** - Zdrojové kódy aplikace
- **text** - Zdrojový kód práce v LaTeXu
- **bp.pdf** - Text bakalářské práce
- **readme.txt** - Základní informace o práci a obsahu disku