VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ KNIHOVNY ALIZE PRO IDENTIFIKACI MLUVČÍCH

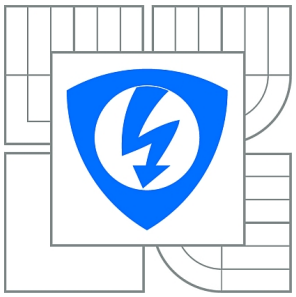
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK SKULÍNEK
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ KNIHOVNY ALIZE PRO IDENTIFIKACI MLUVČÍCH

SPEAKERS IDENTIFICATION USING ALIZE LIBRARY

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

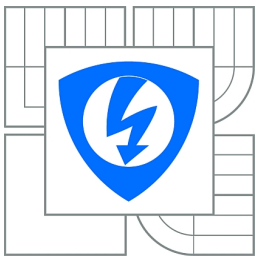
AUTOR PRÁCE — ZDENĚK SKULÍNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ PŘINOSIL, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Zdeněk Skulínek	***ID:***	16865
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Využití knihovny ALIZE pro identifikaci mluvčích

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s knihovnou ALIZE a prostudujte její možnosti pro návrh algoritmu identifikace osob na základě jejich řečového projevu. Na základě získaných teoretických poznatků proveďte vlastní návrh algoritmu s využitím funkcí knihovny ALIZE v programovacím jazyku C/C++, přičemž uvažte využití různých typů řečových příznaků a klasifikátorů. Realizovaný algoritmus ověřte na skupině 20-40 osob.

DOPORUČENÁ LITERATURA:

[1] Psutka J.. Komunikace s počítačem mluvenou řečí. Academia, Praha 1995.
[2] Psutka J., Müller L., Matoušek J., Radová V.. Mluvíme s počítačem česky. Academia, Praha 2006.

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: Ing. Jiří Přinosil, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

ABSTRAKT

V posledních letech, kdy dochází k výraznému nárůstu výkonu počítačů, se na popředí dostaly mimo jiné i systémy pro rozeznávání mluvčího. Vznikla řada knihoven, ať již open–source či proprietárních, které se této problematice věnují. Jednou z takových knihoven je například ALIZE, která byla vyvinuta na univerzitě v Avignonu. Tato knihovna nabízí obecný koncept práce s biometrickými údaji, statistickými výpočty a kompletní framework pro zamýšlenou aplikaci. Úkolem této práce je tuto knihovnu vyzkoušet, a to postavením praktické aplikace pro rozeznávání mluvčího.

Tato práce popisuje úspěšné sestavení této aplikace. Aplikace je platformně nezávislá a je také vyzkoušená na všech třech základních platformách. Ty jsou přílohou práce jako soubory do virtuálního stroje VirtualBox. Práce si všímá praktických zkušeností s knihovnou ALIZE, popisuje její strukturu a architekturu. Pro testování je použito celkem 50 vzorků od 9ti mluvčích, v délkách okolo 20ti sekund. Pro tyto délky funguje rozeznávání naprosto spolehlivě s dodanými vzorky. Proto je na závěr v tezi uskutečněn test skutečné délky potřebné k rozeznání mluvčího.

KLÍČOVÁ SLOVA

Knihovna ALIZE, identifikace řečníka, GMM systém

ABSTRACT

Recently computing power has been significantly increasing and speaker identification systems have been drawing to the foreground among others. Many libraries have been developed, no matter if open–source or proprietary. One of them is for example ALIZE that was developed at Avignon university. This library offers general concept for staff with biometric values, statistic computations and complex framework for intended application. The goal of this thesis is to test this library, namely by constructing a practical application for speaker identification.

This thesis describes successful assembly of that application. Application is platform independent and it is also tested on three basic platforms. These platforms are attached as files into VirtualBox virtual machine. This thesis takes notice of practical experiences with ALIZE library and also describes it's structure and architecture. There are 50 patterns of 9 speakers used for testing in average length 20 seconds. Application recognizes speakers very reliable in these given patterns. Therefore real neccesary length test for speaker recognition was accomplished at the end of this thesis.

KEYWORDS

ALIZE library, speaker identification, GMM system

SKULÍNEK, Zdeněk *Využití knihovny ALIZE k identifikaci mluvčích*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 60 s. Vedoucí práce byl Ing. Jiří Přinosil, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Využití knihovny ALIZE k identifikaci mluvčích“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jiřímu Přinosilovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Rád bych také poděkoval paní Ing. Marii Thirouard, za překlad dokumentace z francouzštiny do angličtiny, kterého se ujala nejen velmi profesionálně, ale i rychle a nad to opravila řadu jazykových chyb, které tam autoři dokumentace zanechali.

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této bakalářské práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ. 1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Téma

Tématem této práce je vyzkoušet knihovnu ALIZE k identifikaci mluvčího.

Techniky rozpoznávání řeči

Řeč obecně obsahuje spoustu informací, mezi které patří pohlaví, nálada, zpráva, identita a psychická kondice atd. Pro člověka je často velmi jednoduché rozeznat tyto „odlišnosti“ nezávisle na sobě. 90letá žena má hlas znějící jako „hlas 90leté ženy“ a unavený muž zní „hlasem unaveného muže“ a to i přes to, že jejich řeč nese stejné sdělení. Tyto přidané informace, které ve skutečnosti nejsou potřebné pro přenos vlastní zprávy, jsou užitečné pro odhad důvěryhodnosti, relevance a mnoha dalších vlastností sdělení.

V posledních letech, kdy dochází k výraznému nárůstu výkonu počítačů, se na popředí dostaly dvě oblasti výzkumu. Rozeznávání řeči a rozeznávání mluvčího. Rozeznávání řeči je technika, která určuje *co* bylo řečeno, a rozeznávání mluvčího je technika říkající *kým* to bylo řečeno. Výsledky by samozřejmě měly respektovat dialekt, zdravotní stav, náladu mluvčího, tak aby výsledky nesly vždy jednu relevantní a stejnou informaci.

Jsou dva přístupy k rozeznávání mluvčího. *Textově–závislé* a *textově–nezávislé*. Textově–závislé jsou takové, kde textově vyjádřený hlas mluvčího je srovnáván s natrénovaným hlasem mluvčího s týmž textovým vyjádřením. Naproti tomu textově–nezávislé jsou takové, kde natrénovaná a reálná řeč může obsahovat věty zcela rozdílné. Nicméně, používají se obě tyto možnosti, protože různé aplikace vyžadují různé přístupy. Například, když sledujeme identitu řečníků v konferenci, je většinou nutné použít textově–nezávislý systém. Tato oblast zpracování řeči je nyní intenzivně zkoumána a cílem vědců je nalézt obecný systém, který nepotřebuje žádné další předpoklady pro svou funkcionalitu.

Jak šel čas, rozdělila se úloha o identifikaci řečníka na dvě subúlohy:

Ověření řečníka, kde máme jako vstup segment řeči a jeden či několik segmentů k natrénování systému. Otázka tedy je „Vyslovil řečník tento segment“ a odpověď je „ano“ nebo „ne“.

Identifikace řečníka, kde mám opět jako vstup segment řeči a nějaký počet segmentů namluvených různými řečníky. Otázka tady je „který řečník vyslovil tento segment“ a odpověď je „identifikace jednoho z řečníků“, který vyslovil tento segment.

Alespoň teoreticky je velký rozdíl ve výpočetní složitosti těchto dvou případů. Ověřovací případ nepotřebuje porovnávat řeč s mnoha dalšími vzorky. Nicméně v praktických aplikacích, nebude jednoduchý model dávat dostatečnou specifičnost modelu pro solidní rozhodnutí, pokud nebude znát nic o řeči a bude hledat v jediném samotném modelu. Jinak řečeno, není možno identifikovat mluvčího, pokud nedefinujeme nějakou množinu tříd. Taková třída může být definována s využitím předpočítané směsi modelů (s nebo bez modelu řečníka), nebo spuštěním systému proti jiným jiným segmentů (nazývaných podvodných- „impostor“).

Detekce řečníka vyžaduje algoritmy, které vyberou nejpravděpodobnější možnost. To je dosaženo nějakým způsobem skóringu. Výstupem takovéhoto algoritmu bude model řečníka s nejvyšším skóre.

Rozdělení dokumentu

Kapitola 1 je takovým úvodem do problematiky zpracování řeči, bude v ní napsáno vše, co jsem si musel nastudovat, abych práci zdárně dokončil.

Kapitola 2 jsou moje postřehy ze zkoumání ALIZE ze zdrojových kódů. Popis ALIZE není detailní, ačkoli není až tak špatný. Více o dokumentaci je popsáno v úvodu kapitoly.

Kapitola 3 jsou moje postřehy ze zkoumání ALIZE z matematické stránky. Protože mi nebyly jasné některé matematické obraty, začal jsem se o ně více zajímat v této kapitole. Závěrem jsem porovnal znovu své bádání a chápání ALIZE.

Kapitola 4 popisuje realizaci projektu v C++. Součástí řešení je nejen můj přínos do projektu, ale hlavně praktické zkušenosti s biometrickou knihovnou ALIZE.

Kapitola 5 uvádí test závislosti spolehlivosti identifikace na délce promluvy. Ukázalo se, že provedením tohoto testu se přišlo na řadu velmi zajímavých skutečností a podnětům k další práci.

Kapitola A je jednoduchým, ale přesto docela obsáhlým návodem, jak aplikaci a knihovnu ALIZE rozchodit na všech třech základních platformách a začít s její úpravou.

Kapitola B je krátké, ale výstižné seznámení s ovládáním projektu.

1 ÚVOD DO ZPRACOVÁNÍ ŘEČOVÝCH SIGNÁLŮ

1.1 Limity autentizace hlasem

1.1.1 Využití biometrie v právních souvislostech

Je třeba zdůraznit limity autentizace hlasem v právních souvislostech. Dobře to rozvedli ve své knize Bonastre J.F a kol.[Bon03]. Závěr tohoto dokumentu navrhuje toto: V současné době není možno naprosto přesně určit, zda podobnost mezi dvěma nahrávkami je dána mluvčím, či nějakými jinými vlivy, zejména když a) mluvčí nespolupracuje b) nemáme pod kontrolou nahrávací zařízení c) podmínky záznamu nejsou známy d) není známo zda hlas není zamaskován méně důležitými zvuky e) jazykový obsah zprávy není řízen. Obezřetnost a soudnost musíme mít na zřeteli když používáme techniky rozeznávání mluvčího, kde člověk nebo automat počítá s těmito neřízenými faktory. V mnoha omezených nebo modelových situacích, nebo v situacích, kde je třeba něco objasnit, může být rozumné využití těchto technik vhodné, ovšem nemohou být považovány na neomylné. V současnosti žádný vědecký proces nedovoluje jednoznačně charakterizovat hlas člověka nebo identifikovat s absolutní jistotou z hlasu.

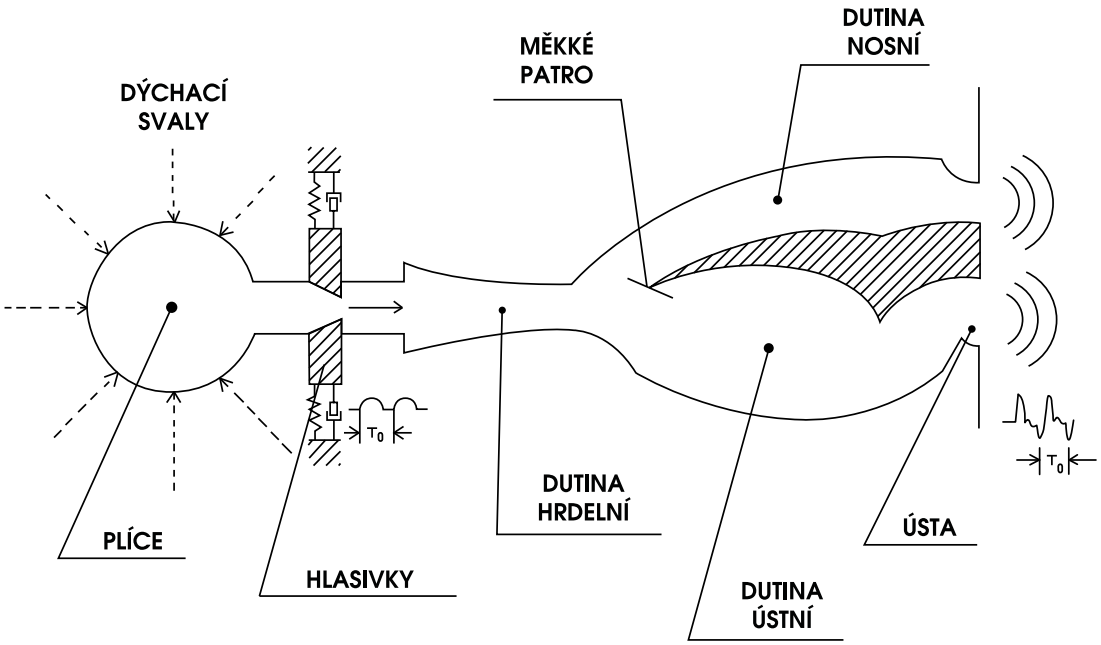
1.2 Model hlasového traktu pro analýzu a syntézu řeči

Na ilustraci 1.1 je obecný řez hlasovým ústrojím člověka. Pochází z publikace [Psut95]. My se na tento obrázek ale můžeme podívat jako na systém, který se skládá z generátoru(hlasivky) a dvou filtrů (dutina ústní, vyzařování zvuku). Viz ilustrace 1.2.

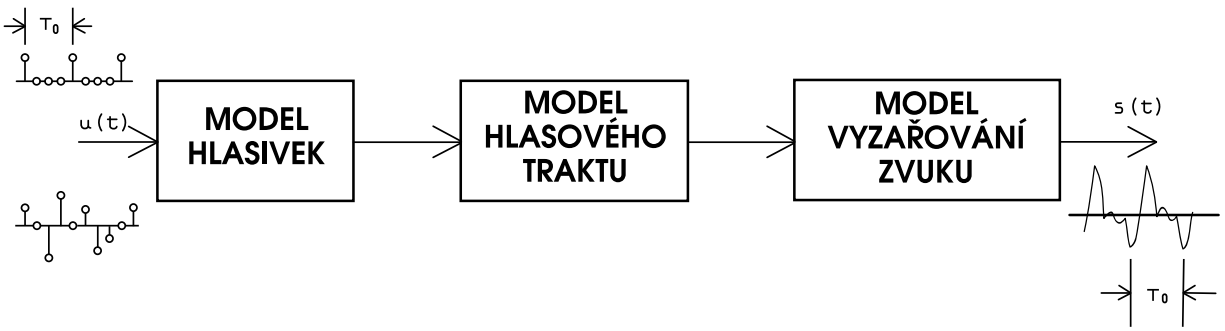
1.3 Model hlasivek

Frekvence vibrací závisí na tlaku vzduchu vháněného z plic, hmotnosti a pružnosti hlasivek a také na velikosti hlasivkové štěrbiny. Rychlost kmitání lze dobře modelovat jako výstup nízko-pásmového filtru druhého řádu, jehož parametry jsou pro každého řečníka jiné. Jeho přenosovou funkci lze uvažovat ve tvaru:

$$G(z) = \frac{1}{[(1 - \exp^{-cT})z^{-1}]^2}. \tag{1.1}$$



Obr. 1.1: Model hlasového ústrojí člověka



Obr. 1.2: Signálový model hlasového ústrojí člověka

Podrobnější informace lze nalézt v [Psut95].

1.4 Model hlasového traktu

Model hlasového traktu by měl především respektovat vlastnosti dutiny hrdelní dutiny ústní a pro nosovky i dutiny nosní. Signálové parametry závisí na rozměrech dutin. Tyto parametry se neustále mění, jelikož ústa se při mluvení otevírají a zavírají také pohybem jazyka a dalšími vlivy. Matematicky se to dá popsat jako několik dvoupólových rezonátorů zapojených do kaskády. Rezonanční frekvence těchto rezonátorů odpovídají frekvencím jednotlivých formantů. Přenosovou funkci lze tedy nalézt ve tvaru:

$$V(z) = \frac{1}{\Pi_{i=1}^{K}[1 - \exp^{-\alpha_i T}\cos(\beta_i T)z^{-1} + \exp^{-2\alpha_i T}\, z^{-2}]}. \tag{1.2}$$

1.5 Model vyzařování zvuku

[Psut95] Uvádí vyzařovací funkci ve tvaru:

$$L(z) = 1 - z^{-1}. \tag{1.3}$$

Pro tuto práci ji není třeba více rozebírat.

1.6 Model produkce řeči

Sjednocením výše uvedených vzorců dostáváme model produkce řeči. Protože tyto filtry jsou zapojeny do kaskády, stačí přenosové funkce prostě vynásobit:

$$H(z) = G(z)V(z)L(z) = \tag{1.4}$$

$$\frac{1 - z^{-1}}{[(1 - \exp^{-cT})z^{-1}]^2\,\{\Pi_{i=1}^{K}[1 - \exp^{-\alpha_i T}\cos(\beta_i T)z^{-1} + \exp^{-2\alpha_i T}\, z^{-2}]\}}. \tag{1.5}$$

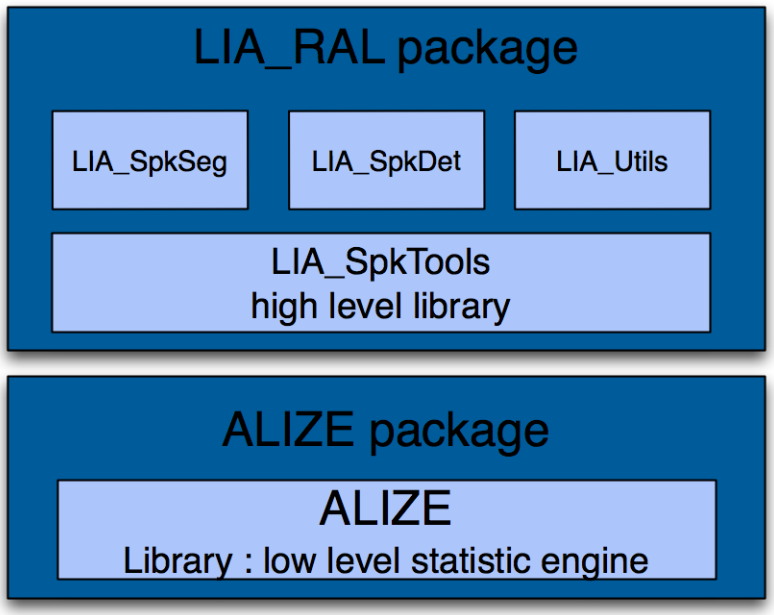
2 POPIS KNIHOVNY ALIZE

2.1 Několik slov o ALIZE

ALIZE je open-source platforma pro biometrockou identifikaci. Je distribuována pod LGPL licencí. Hlavním cílem ALIZE je umožnit vývoj biometrických aplikací všem, poskytnutím řady low-level a high-level funkcí. ALIZE je multi-platformní, funguje jak pod Windows, tak pod Linuxem a MacOSX. Knihovnu naprogramoval Frederic Wils a nyní ji vede profesor Jean Francois Banstre.

2.2 Struktura knihovny ALIZE

Alize má dvě základní části. Je to nízkoúrovňová ALIZE 2.3 a dále vyšší LIA_RAL 2.4. LIA_RAL se dále dělí na nižší část SpkTools 2.4.3 a na vyšší část, která má tři subčásti. Jde o SpkDet 2.4.5, SpkSeg 2.4.4 a Utils 2.4.6.



Obr. 2.1: Struktura knihovny ALIZE

2.3 Balík ALIZE

2.3.1 Dokumentace

ALIZE má dva typy dokumentace. Cca 48stránkový dokument v angličtině a francouzštině sloužící jako návod k použití a pak dokumentaci v hlavičkových souborech, ze které dostaneme ucelený dokument v HTML programem doxygen. Ten druhý zmiňovaný dokument je jen v angličtině. Většina funkcí a tříd dokumentaci nemá nebo je dokumentována velmi sporadicky. Na druhou stranu názvy proměnných a funkcí se zdají být výstižné. V konfiguračním souboru pro doxygen je pozapínána celá řada voleb pro maximální dokumentovatelnost. Výsledkem je celkem velký dokument, který ale obsahuje relativně málo zásadních informací.

2.3.2 Překlad ALIZE

ALIZE jsem přeložil pod Ubuntu Linuxem napoprvé a bez jakýchkoli problémů.

2.3.3 Obecně

ALIZE si potrpí na vyžívání výjimek. Soubor `Exception.h` jich definuje hned několik, vesměs pro IO operace.

2.3.4 Object

Základní třída balíku ALIZE. Jsou z ní odvozeny veškeré další typy. `Object` definuje několik výčtových typů a také převod těchto typů na a z typu `String`, což bude potřebné pro načítání z textového (XML) souboru. Další funkcionalita jsou jednoduché funkce pro potřeby ladění. Jedná se o počítadla vytvořených objektů, dále o metodu `getClassName` a metodu `toString`.

2.3.5 Základní typy

- `AutoDestructor` - smazáním tohoto objektu se smaže i jeho parametr - hlídaný objekt
- `Exception`

2.3.6 Základní datové typy

- `String`
- `XLine` - seznam `String` objektů
- `XList` - seznam `XLine` objektů

- `Vector`
- `ULongVector`
- `RefVector`
- `RealVector`
- `Matrix`
- `DoubleSquareMatrix`
- `BoolMatrix`

2.3.7 Objekty z charakterem kontejnerů

- `Config`
- `CmdLine`
- `Label`

2.3.8 Audio Input Stream

`AudioInputStream` je interface určená k načítání audio souborů. Dědí z ní skutečná načítací třída `AudioFileReader`, která vlastní třídu `FileReader` která pouze zapouzdřuje čtení ze souboru. K je tak snadno modifikovatelná a také snadno zaměnitelná např. z čtení ze zařízení nebo čtení s API operačního systému, což může mít nějaké výhody, třeba možnost paměťově mapovaného souboru (za cenu multiplatformnosti).

2.3.9 Feature - vektor parametrů

Výsledky výpočtů jsou uloženy do vektoru `Feature`. Ten je možno uložit čí načíst ze souboru a je zde použito stejné techniky jako v předcházejícím případě pro možnost snadné modifikace. Hlavním důvodem je však porovnávání se známými hodnotami pomocí objektu `Distrib`.

2.3.10 Distrib - objekt pro výpočet shody

`Distrib` má v podstatě jen jednu metodu. Jedná se o metodu `computeLK` – Vypočítej věrohodnost *likelihood*. Vzorec, který tato metoda implementuje bohužel moc nechápu, dokonce si myslím, že je minimálně z poloviny empirický, jmenovitě nechápu $\pi^{N/2}$. Ale v kostce mohu říct, že se vezme známý vektor, v tomto objektu označený a uložený jako `Mean` a počítá vzdálenost od tohoto vektoru k vektoru `Feature`, který je parametrem funkce. Výsledek se následně snaží normalizovat. Výsledná hodnota je tedy jakýmsi ukazatelem podobnosti, čím menší tím podobnější, mezi naměřenými

a uloženými parametry. Vzorec jsem objevil až později, a detailněji jsem ho popsal v kapitole Statistika 3.

2.3.11 XML parser

ALIZE obsahuje vlastní implementaci SAX parseru. Na přínos tohoto kódu se dívám dost skepticky. Tato implementace nepřináší naprosto nic oproti standardnímu libxml2. Jediné pozitivum je tedy odstranění závislosti na další knihovně.

2.3.12 Shrnutí

Výše uvedené informace lze najít jednak v hlavičkových souborech, ale hlavně ve zdrojových kódech ALIZE. Vyžaduje to hodně zkoumání, popisy jsou téměř minimální a hlavně většinou jen kopírují název funkce. Například:

```
/// Returns a constant reference to the mean vector
/// @return a constant reference to the mean vector
///
const DoubleVector& getMeanVect() const;
```

ALIZE jinak vypadá užitečně pro zamýšlenou funkcionalitu, kde je po ní požadováno jen několik málo úkolů.

V dokumentaci k ALIZE jde zjistit jak tyto funkce používat, zejména načítání ze souborů. Vlastně je to řečeno velmi elegantně a API je jednoduché na použití.

2.4 Balík LIA_RAL

2.4.1 Dokumentace

Jsou v podstatě tři druhy dokumentace. Popis funkcí v programu Doxygen, „Dokumentace LIA_RAL“ a příklady.

Jestliže v ALIZE jsem hodnotil dokumentaci v doxygenu spíše hůře, zde jsem nenašel jediné slovo v celých 22MB zdrojového textu. Navíc, názvy proměných nejsou čitelné pro „návštěvníky“ projektu. Viz:

```
ULongVector  _idx
ULongVector  _nbg
DoubleVector  _snsw
DoubleVector  _snsl
unsigned long  _nt
unsigned long  _nbgcnt
```

, ačkoli je pravda, že jde v tomto případě o prvky chráněné (`protected`).

2.4.2 Překlad LIA_RAL

Knihovnu LIA_RAL jsem přeložil na Ubuntu Linux napoprvé a bez jakýchkoli obtíží. Stejně tak se přeložily i příklady, kterých je většina. Samotná LIA_RAL je velice malá.

2.4.3 LIA_SpkTools

Zde je několik programů, které využívají některé z níže uvedených částí knihovny.

2.4.4 LIA_SpkSeg

LIA_SpkSeg je soubor několika programů, sloužících jako ukázky segmentace řeči.

2.4.5 LIA_SpkDet

LIA_SpkDet je pro nás základní část knihovny. Téměř většina toho, co je zde nabízeno se dá při vývoji aplikace na identifikaci řečníka použít.

2.4.6 LIA_Utils

LIA_Utils jsou zcela obecné podprogramy, které se dají využít. Možné lepší pojmenování by bylo „nezařazeno“.

3 STATISTIKA

3.1 Obecně

Tato část rozebírá statistiku v části ALIZE, kterou považuji za nutné bezpečně zvládnout.

3.2 Úplná a podmíněná pravděpodobnost

Ze základní rovnice pravděpodobnosti

$$p(A, B) = p(A|B)p(B) = p(B|A)p(A), \tag{3.1}$$

vychází volným přepisem Bayesův vzorec:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}, \tag{3.2}$$

který využíváme v situaci, kdy máme jednu nebo více omezujících podmínek. Nejčastěji má omezující podmínka charakter náhodného jevu, který musí před zkoumaným jevem nastat. Mluvíme pak o podmíněné pravděpodobnosti.

Pokud jsou $A_i s$ nezávislé, můžeme rovněž psát:

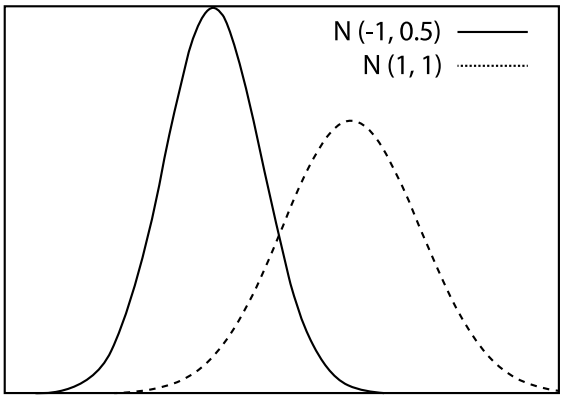
$$p(B) = \sum_i p(B|A_i)p(A_i), \tag{3.3}$$

a z toho Bayesův vzorec pro více proměnných,

$$p(A|B) = \frac{p(B|A)p(A)}{\sum_i p(B|A_i)p(A_i)}. \tag{3.4}$$

3.3 Gaussovo (normální) rozdělení

Normální (nebo Gaussovo) rozdělení pravděpodobnosti je jedno z nejdůležitějších rozdělení pravděpodobnosti spojité náhodné veličiny. Slovo „normální“ zde není použito v obvyklém smyslu „obyčejné“, „běžné“. Jeho použití se vztahuje k staršímu



Obr. 3.1: Normální rozdělení různých parametrů

významu „řídící se zákonem, předpisem nebo modelem“ [Heb88].

Tímto rozdělením pravděpodobnosti se sice přesně řídí jen málo náhodných veličin, ale jeho význam spočívá v tom, že za určitých podmínek dobře aproximuje řadu jiných pravděpodobnostních rozdělení (spojitých i diskrétních) [Heb88].

V souvislosti s normálním rozdělením jsou často zmiňovány náhodné chyby, např. chyby měření, způsobené velkým počtem neznámých a vzájemně nezávislých příčin. Proto bývá normální rozdělení také označováno jako zákon chyb. Podle tohoto zákona se také řídí rozdělení některých fyzikálních a technických veličin [Heb88].

Hodnota pravděpodobnosti je definována vztahem

$$f(x) = \Theta[\mu, \rho^2] = \frac{1}{\rho\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\rho^2}}, \tag{3.5}$$

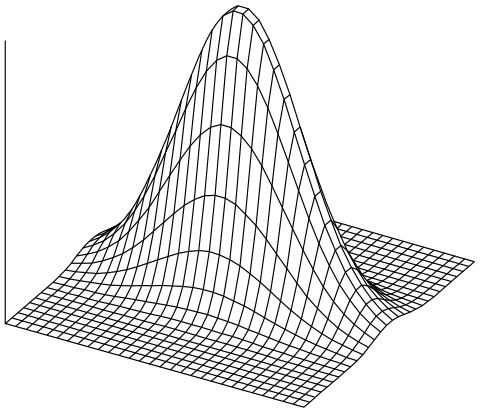
kde ρ^2 je rozptyl ($D(X)$), x je zkoumaná hodnota a μ je střední hodnota ($E(X)$) a současně i medián ($x_{0.5}$).

3.3.1 Příklad Gaussova rozdělení

- Výška Pygmejů – $\Theta[\mu, \rho^2] = \Theta[4, 1]$ stop
- Výška Křováků – $\Theta[\mu, \rho^2] = \Theta[6, 1]$ stop

Otázka: Pokud volně vybereme člověka z populace ⇒ jaká je pravděpodobnost, že výška nabude nějakého určitého rozmezí hodnot?

Jestliže volně vybereme Pygmeje, řekněme x, pak:



Obr. 3.2: Dvojdimenzionální normální rozdělení

$$p(výškax = 4'1'') = \frac{1}{\sqrt{2\pi 1}} \mathrm{e}^{-\frac{1}{2.1}(4'1''-4)^2}, \tag{3.6}$$

Poznámka: zde je konstantní rozptyl a střední hodnota, mění se jen výskyty x.
Závěr: člověk výšky 4'1" *je pravděpodobněji* Pygmej.

Pokud budeme pozorovat více výšek osob ze stejné populace – řekněme 3'6", 4'1", 3'8", 4'5", 4'7", 4', 6'5" ⇒ budeme mít větší jistotu, že populace jsou Pygmejové.

Tedy více pozorování ⇒ lepší rozhodnutí.

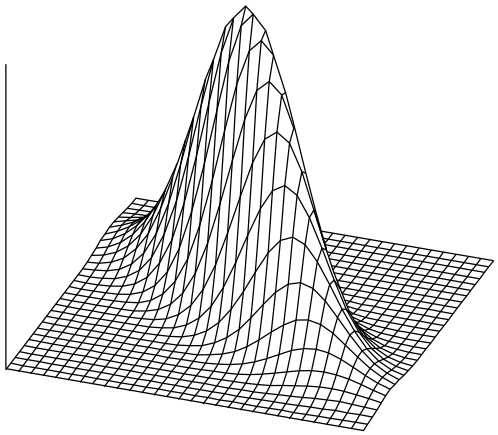
3.4 Vícerozměrné Gaussovo rozdělení

Pro nás ale bude zajímavé rozdělení vícerozměrné, protože se setkáme vektory tvořenými celou řadou parametrů.

Sdružená hustota pravděpodobnosti pro S-rozměrný vektor má obecně tvar

$$f(x_1, x_2, \ldots x_S) = \frac{1}{\sqrt{(2\pi)^S \left|C\right|}} \mathrm{e}^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)}, \tag{3.7}$$

kde C je kovarianční matice o které napíši v následující sekci.



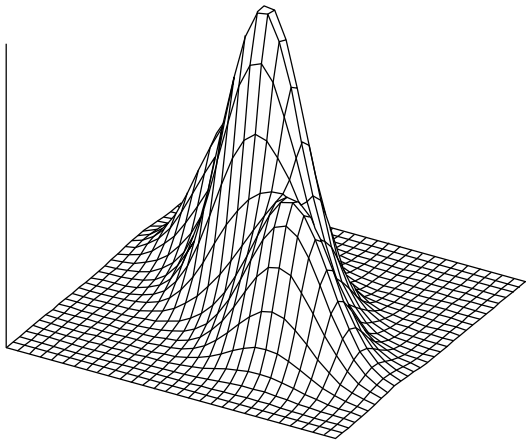
Obr. 3.3: Dvojdimenzionální závislé (natočené) normální rozdělení

3.5 Kovarianční matice

V minulé sekci, ve vzorci 3.7 vystupuje kovarianční matice. Pojďme si teď ukázat nějaké její vlastnosti:

- Je čtvercová
- Je pozitivně definitní
- Je symetrická podle diagonály
- Na hlavní diagonále jsou variance ρ_i^2
- Mimo diagonálu leží kovariance $cov(x_i, x_j)$

Kovariance je je jakýsi vztah mezi hodnotami vektoru, přesněji, $cov(x_i, x_j) = E[(x_i - E(x_i)).(x_j - E(x_j))]$. Její praktický význam je v tom, že nám ukazuje natočení grafu „zvonu“ v prostoru. Srovnejte 2D normální rozdělení 3.2 a natočené normální rozdělení 3.3. ALIZE má dva režimy prací s kovarianční maticí, jeden ji bere jako klasickou matici, tedy předpokládá natočení „zvonu“, a druhá varianta má pouze prvky na hlavní diagonále, tedy pracujeme s normálním rozdělením a výpočty jsou pak rychlejší. Jenže pokud by by bylo třeba díky protáhlému tvaru zvonu graf(model) „narovnat“ a tento graf se nenarovnal, pak i tento vektor by se musel otočit, pak by se vlastně násobilo maticí a objem operací by byl ještě větší (násobení maticí a ještě maticí s diagonálou). Jinými slovy, je lépe narovnat model, než násobit maticí při testování.



Obr. 3.4: Směs tří Gaussovek ve 3D prostoru

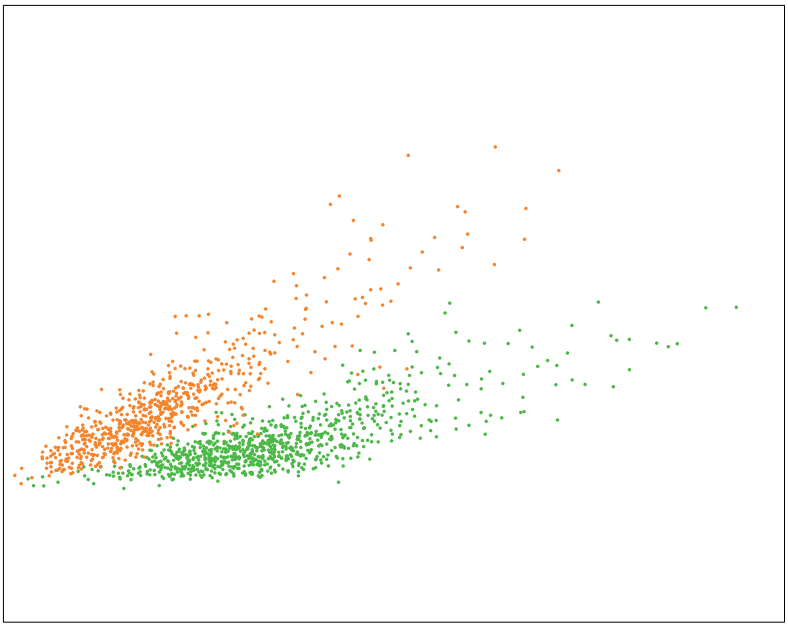
3.6 Srovnání s ALIZE

Vztah 3.7 je přesně to, co počítá funkce `computeLK` počítacího objektu `Distrib`. Před samotným výpočtem naplnit vektor `mean` (reprezentující μ). Pak je nutno zavolat funkci `computeAll`, která vytvoří kovarianční matici a spočítá matici k ní inverzní. Dále počítá tu první konstantní část vzorce $\frac{1}{\sqrt{(2\pi)^S \left|C\right|}}$ a uloží do proměné `constante`.

Vlastní výpočet už je velmi jednoduchý, spočítá vektor $(x-\mu)$ a vynásobí inverzní kovarianční maticí.

3.7 Směs Gaussových rozdělení

Často se stává, že hodnoty vektorů příznaků patřící k jednomu vzorku neleží jen v jediné oblasti a dochází k situaci, kdy je možno stanovit několik oblastí výskytu vektorů. V tomto případě není moc dobrý nápad aproximovat je jediným rozdělením, protože toto rozdělení by bylo velmi obecné(veliký rozptyl) a bylo by velmi podobné s jinými vzorky, což by následné rozpoznávání velmi ztížilo a znepřesnilo. Proto se zavádí směs Gaussovek. Je to v podstatě vektor Gaussových rozdělení(vektorů a rozptylů). V grafu je to jednoduché, je to vlastně součet všech hustot pravděpodobnosti, viz obr 3.4. V praxi je vyzkoušeno, že těchto Gaussovek by mělo být 3-10. Pokud je jich totiž velmi mnoho, dochází k opačnému extrému, totiž že zkoumaný vzorek se do těchto malých oblastí „netrefí“ a nebude tím pádem ani rozpoznán.



Obr. 3.5: Obecné možné rozložení parametrů v 2D prostoru

3.8 Parametrizace modelu

V této kapitole se nebudu zabývat tím jak parametry získat, ale pouze tím jak je zpracovat.

3.8.1 Stav po výstupu z extrakce parametrů

Na úvod se podívejme na obrázek 3.5, na kterém je nějaké z možných rozložení parametrů, pro příklad v 2D prostoru. Už na první pohled má nějaké vady. My bychom pro identifikaci totiž chtěli aby rozložení parametrů:

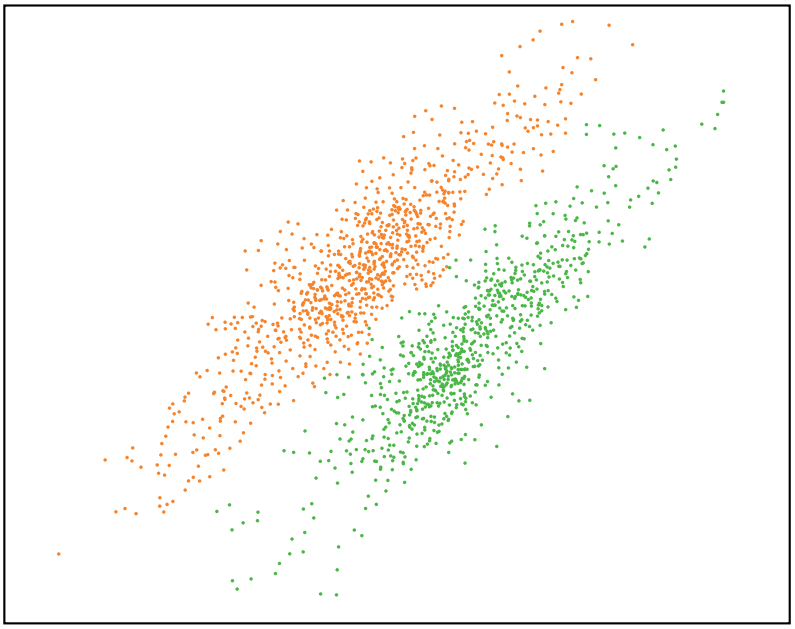
- Bylo Gaussovského rozložení
- Bylo dekorelované
- Bylo málo dimenzionální

.

Jak vidíme naše není ani jedno z toho.

3.8.2 Úprava rozložení na Gaussovské

Když se blíže podíváme na tvar toho shluku příznaků, připomíná něco jako „kometu“. Můžeme provést třeba třetí odmocninu všech koeficientů. Výsledek je vidět na ob-



Obr. 3.6: Komprimované rozložení parametrů v 2D prostoru

rázku 3.6. Rozložení se tak nelineárně deformuje — Komprimuje.

3.8.3 Dekorelace rozložení

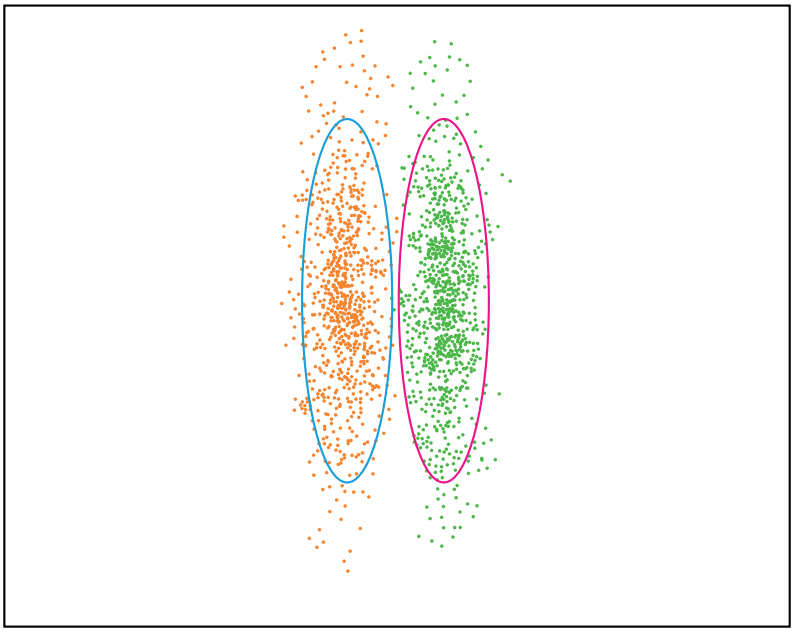
Jak bylo vysvětleno v kapitole 3.5 o kovarianční matici, korelované koeficienty znamenají, že je rozložení otočené. Není tedy problém otočit rozdělení tak, aby bylo v základním postavení podle všech os. Otočení v prostoru je vlastně násobení maticí. Výsledek našeho snažený by měl vypadat nějak takhle – 3.7.

3.8.4 Omezení dimenze

Na úrovní statisty je tato operace velmi těžko realizovatelná, zejména proto, že jde o významnou ztrátu informace. O takovém zásahu je nutné něco vědět o významu těch parametrů, proto je lépe přesunout toto téma do kapitoly extrakce příznaků. Jedno ale udělat můžeme. Když se podíváme ještě jednou na obrázek, zjistíme, že dimenze y je již zbytečná, protože oblasti se z pohledu od osy y zcela překrývají.

3.9 Trénování

V minulé sekci jsem uvedl, že existuje nějaké rozdělení, které modeluje rozdělení příznaků. Otázka je jak toto rozdělení zjistit. Tuto problematiku velmi pěkně rozepsal



Obr. 3.7: Otočené(dekorelované) rozložení parametrů

ve své knize J. Psutka [Psut05]. Ten napsal pro stanovení pravděpodobnosti vzorku O:

$$P(O|\lambda) = \sum_S P(O,S|\lambda) = \sum_S P(O|S,\lambda)P(S|\lambda) = \sum_S a_{s(0)s(1)} \prod_{t=1}^{T} b_{s(t)}(o_t) a_{s(t)s(t+1)}, \tag{3.8}$$

kde O je vektor vzorku, S je vektor vzoru a λ je hledaný model.

Tento výraz má ale dost zásadní chybu – je v něm ukryto velice mnoho výpočetních operací, je prakticky téměř nerealizovatelný. Proto byl vymyšlen algoritmus EM.

3.9.1 Pozorování směsi Gaussovek

Následují úvahu provedl velmi názorně Samudravijaya([Sam09]).

Máme urnu a v ní tři druhy koulí. Vytahujeme se zavřenýma očima tyto koule z urny...

červená koule	$\Rightarrow$	generuje x_i z $N(x;\mu_1,\rho_1)$
modrá koule	$\Rightarrow$	generuje x_i z $N(x;\mu_2,\rho_2)$
zelená koule	$\Rightarrow$	generuje x_i z $N(x;\mu_3,\rho_3)$

a my mám jen pozorování $x_1, x_2, ..., x_N$.

Takže

$$p(x[i];\theta) = c_1 N(x;\mu_1,\rho_1) + c_2 N(x;\mu_2,\rho_2) + c_3 N(x;\mu_3,\rho_3) \qquad (3.9)$$

ale nevíme ze které komponenty x[i] pochází!

Můžeme vůbec určit $\theta = [c_1 c_2 c_3 \mu_1 \mu_2 \mu_3 \rho_1 \rho_2 \rho_3]^T$ z pozorování?

$$arg\ max_\Phi p(X;\Phi) = arg\ max_\Phi \prod_{i=1}^{N} p(x_i;\Phi) \qquad (3.10)$$

Jednodušší problém: **Známe komponentu každého pozorování**

Poz	x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]	x[9]	x[10]	x[11]	x[12]
Hod	1	2	2	1	3	1	3	3	2	2	3	3	3

$X_1 = \{x[0], x[3], x[5]\}$ pokrývá $p_1(x;\mu_1;\rho_1)$
$X_2 = \{x[1], x[2], x[8], x[9]\}$ pokrývá $p_2(x;\mu_2;\rho_2)$
$X_3 = \{x[4], x[6], x[7], x[10], x[11], x[12]\}$ pokrývá $p_3(x;\mu_3;\rho_3)$

Z $X_1 = \{x[0], x[3], x[5]\}$ zjistíme
$\hat{c_1} = \frac{3}{13}$
$\hat{\mu_1} = \frac{1}{3}\{x[0] + x[3] + x[5]\}$
$\hat{\rho_1^2} = \frac{1}{3}\{(x[0] - \hat{\mu_1})^2 + (x[3] - \hat{\mu_2})^2 + (x[5] - \hat{\mu_3})^2\}$

my ale v praxi nevíme ze kterého komponenty (červené, modré, zelené) koule pochází.

3.9.2 Kompletní a nekompletní data

Zkusme si představit ...

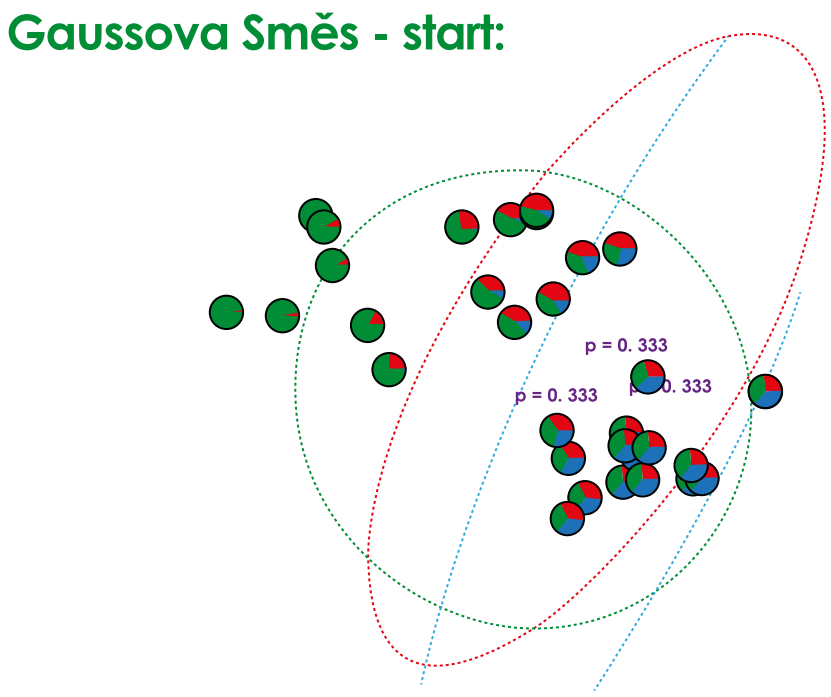
$x[0], x[1], ..., x[N-1] \Rightarrow$ nekompletní data

A nyní mějme jinou množinu hodnot $y[0], y[1], ..., y[N-1]$

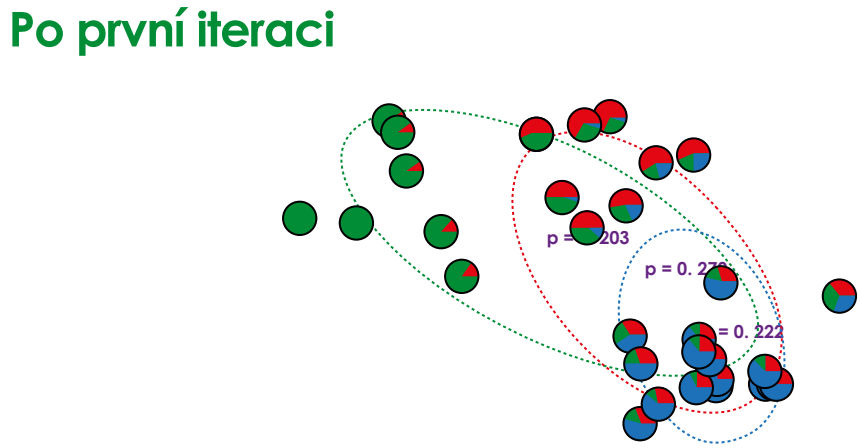
3.9.3 Algoritmus EM (Expectation-Maximization)

1. Odhadněte parametry $\theta^g = c_1^g c_2^g c_3^g \mu_1^g \mu_2^g \mu_3^g \rho_1^g \rho_2^g \rho_3^g$
2. Pro známé parametry θ^g najdeme pravděpodobnost $x[i]$ pro jtou komponentu.

   $p[y[i] = j | x[i]; \theta^g]$ pro i [1,2..N] - N počet vzorků
   pro j [1,2..M] - M počet komponent
3. $\hat{c_j^{nové}} = \frac{1}{N} \sum_{i=1}^{N} p(y[i] = j | x[i]; \theta^g)$
4. $\mu_j^{nové} = \frac{\sum_{i=+}^{N} x_i . p(y[i]=j|x[i];\theta^g)}{\sum_{i=1}^{N} p(y[i]=j|x[i];\theta^g)}$
5. $(\rho_j^2)^{nové} = \frac{\sum_{i=+}^{N} (x_i - \mu_i)^2 . p(y[i]=j|x[i];\theta^g)}{\sum_{i=+}^{N} p(y[i]=j|x[i];\theta^g)}$
6. Opakovat od bodu 2, dokud až po konvergenci

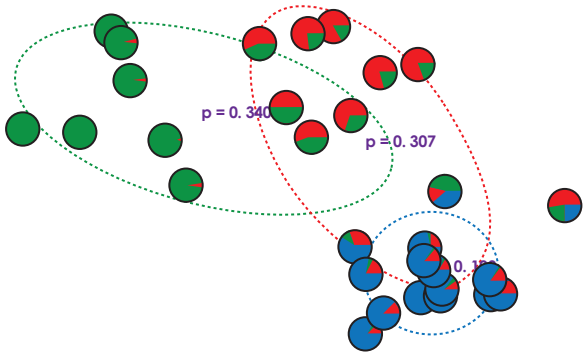


Obr. 3.8: Algoritmus EM - start



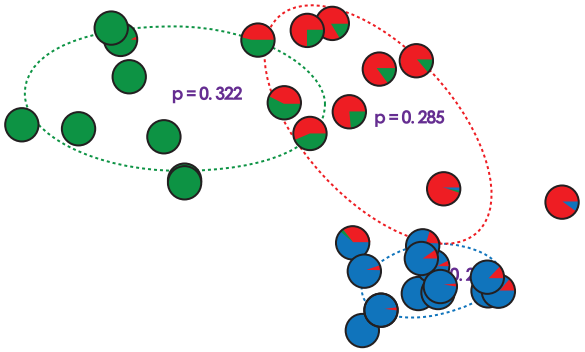
Obr. 3.9: Algoritmus EM - po 1. iteraci

Po třetí iteraci

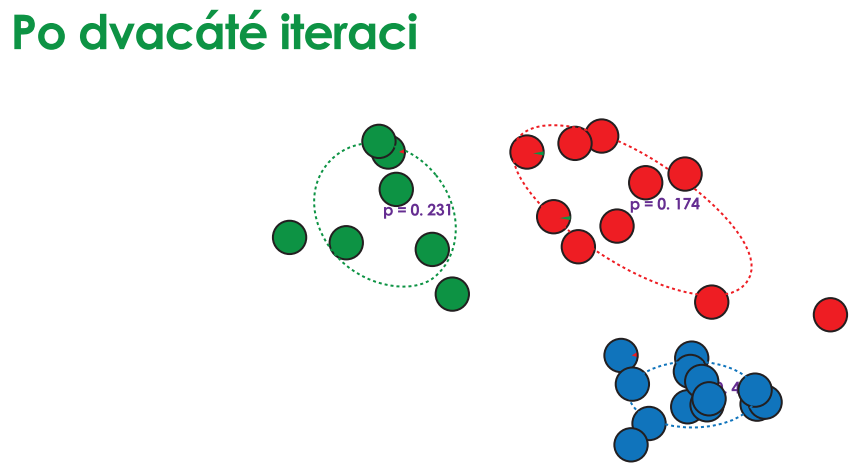


Obr. 3.10: Algoritmus EM - po 3. iteraci

Po páté iteraci



Obr. 3.11: Algoritmus EM - po 5. iteraci



Obr. 3.12: Algoritmus EM - po 20. iteraci

4 PRÁCE S ALIZE

4.1 Úvod

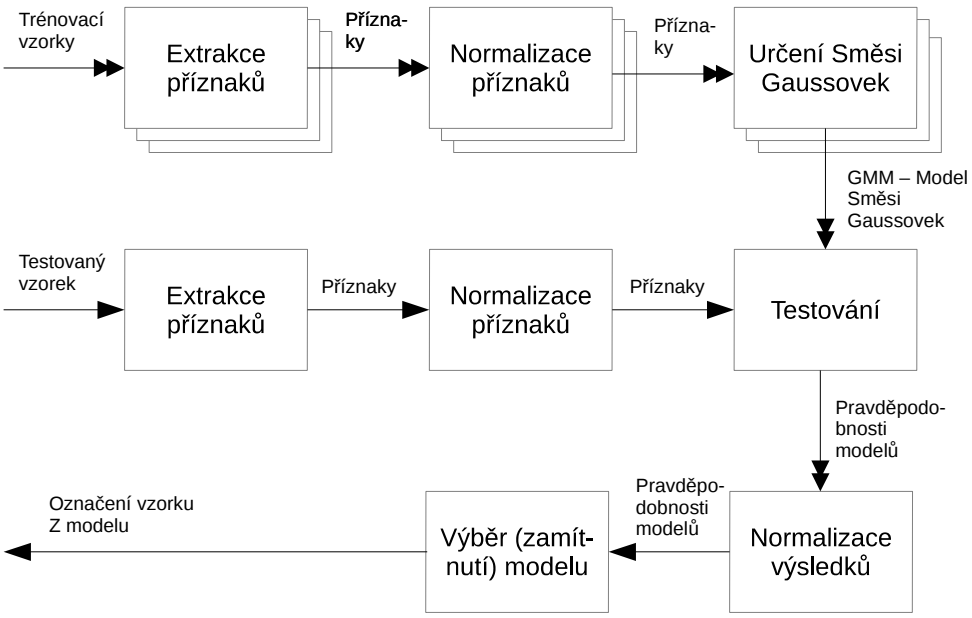
V předchozích kapitolách jsem se zabýval nějakou teorií o zpracování řeči a ALIZE z pohledu průzkumu kódu. Tato kapitole je tedy trochu retrospektivní a budu se v ní věnovat pouze praktickým zkušenostem s ALIZE. Ostatní informace jsou uvedeny v ostatních kapitolách. Práce na projektu začala stažením projektu ze stránek univerzity v Avignonu. Stažení ALIZE z internetu je ale jen začátek. Současně s ALIZE je k dispozici ještě knihovna LIA_RAL, dokumentace k ALIZE a několik příkladů, z nichž pro mě je zajímavý jen příklad „GMM/UBM system with ALIZE“.

4.2 Blokové schéma aplikace

Právě GMM/UBM system s ALIZE se zdá být dobrý výchozí bod pro rozpoznávač. Dělá přesně to, co potřebujeme, aby dělal. Načte příznaky, normalizuje je, spočítá z nich směs Gaussových rozdělení a pak už jen testuje s novými vzorky. Jeho blokové schéma je na obrázku 4.1. Celý systém jsou vlastně jen tři shellové skripty, volající programy, které jsou součástí LIA_RAL. V projektu jsou dále konfigurační soubory pro ALIZE a testovací data. Na extrakci příznaků je použit „nějaký“ program, o kterého moc nevím co dělá, mám jen spustitelný soubor bez zdrojového textu. To ovšem nevadí, protože já mám příznaky vlastní, generované matlabovým skriptem pana Dr. Přinosila.

4.3 Překlad dokumentace k ALIZE

Jedním z klíčových problému, které je třeba řešit, je dokumentace. Ta originální, jak bylo již uvedeno, je z jedné poloviny ve francouzském jazyce a zbytek v jazyce anglickém. Našel jsem na webových stránkách univerzity KTH (Švédsko), diplomovou práci T. Gannerta [Gan07], který se s jazykovým problémem snažil nějak vypořádat. Bez dokumentace by bylo nutno použít nějakou formu reverzního inženýrství, což není lehké a je to časově náročné. Zkusil jsem tedy napsat obecnou emailovou adresu projektu ALIZE s prosbou o překlad do angličtiny. Odepsal mi pan Prof. Bonastre, který se vyjádřil velmi pozitivně o mém úmyslu dokumentaci přeložit a nabídl mi svou pomoc. Zjistil jsem, že lidí, kteří jsou schopni přeložit takto odborný text z francouzštiny do angličtiny, je velice málo, zejména v naší zemi. Nakonec jsem našel překladatelku schopnou dokumentaci přeložit a navázal s ní spolupráci. Dílo



Obr. 4.1: Základní blokové schéma GMM/UBM systému

bylo hotovo v druhé části listopadu. Překlad dokumentace byl velký pokrok v práci s ALIZE a ihned jsem se pustil do praktické realizace bakalářské práce.

4.4 Realizace projektu v C++

ALIZE je napsána v C++ a je multiplatformní, jak již bylo uvedeno dříve. Vytvořil jsem tedy projekt v prostředí NetBeans. Mezi hlavní důvody tohoto prostředí patří zejména multiplatformnost(je naprogramováno v JAVĚ), podpora C++, stabilita, vestavěná podpora debugeru a uživatelská přítulnost.

Vlastní program je velmi jednoduchý, obsahuje zdrojové texty všech programů v příkladu GMM/UBM system, pouze trochu upravené, tak aby neočekávaly parametry z příkazové řádky. Rozdíl tam ovšem být musí. Z programu volám GNU Octave s matlabovskými skripty pro extrakci příznaků. Skript pro extrakci příznaků jsem proto rozšířil tak, aby očekával parametry z příkazového řádku a výsledky uložil do binárního souboru. V tomto případě je to matice, jejíž rozměry jsou počet příznaků jako sloupce a počet řádků je roven počtu segmentů extrahovaných z promluvy. S tím souvisí další změna. Výsledek, který byl očekáván ve formátu SPRO4, nyní byla jen řada čísel typu float. Prohlídkou zdrojových textů (ani v manuálu to moc popsáno není) jsem zjistil, e by mělo stačit natavit ALIZE tak, aby čekala příznaky ve formátu RAW a přidat nějaké další parametry, například velikost vektoru příznaků.

4.5 Výchozí stav projektu s ALIZE

Ovšem ALIZE po změně vstupního formátu na RAW začne padat na výjimku. Jako důvod je uvedena maska vektoru, tedy technika, kde je možno vybrat z vektoru příznaků jen některé hodnoty. Musel jsem po hledání zmodifikovat ALIZE tak, aby již masku nechtěla a mohl jsem se dostat dále. V parseru masky je chyba zadání masky. Podle dokumentace nefunguje správně.

Tím ovšem nekončí práce na projektu. ALIZE vyhodí výjimku při trénování modelu směsi Gaussovek. Na toto si již několik lidí stěžovalo na bugzille projektu ALIZE, ovšem autoři věc nechávají již přes dva roky bez odezvy. Laděním ALIZE se ukázalo, že chyba je v načítání. ALIZE neumí načíst normalizované příznaky, které si sama uložila do souboru. Formát výstupního i vstupního souboru je stejný, a to SPRO4.

Protože zdrojové kódy ALIZE máme, je možno s předchozí situací něco dělat. Prvně bude možné v bakalářské práci opravit maskování vektorů tak, aby bylo možno zadat nějakým způsobem všechny kanály, nebo maskovací stream vůbec nepoužít.

Druhá chyba je rovněž řešitelná. Možná bude stačit používat jen formát RAW, který jako jediný známe, a tudíž bude možno případné chyby v ALIZE opravit. Ještě lepší možnost je neukládat mezivýsledky vůbec a uložit do souboru pouze výsledek rozhodování a model směsi Gaussových rozdělení.

Celkově se dívám velmi optimisticky na možnost opravit nebo obejít chyby v ALIZE. Samozřejmě hrozí případ kdy se opravením jedné chyby objeví několik dalších a po jejich opravení ještě další.

4.6 Konfigurační programování v ALIZE

Nakonec jsem našel jen jednu chybu, kterou jsem nahlásil autorům ALIZE.

Použil jsem všude pro příznaky formát RAW. Zafungovalo to. Je to jen matice čísel typu float, takže se nyní jde lépe zjistit, co je v souboru uložena a ladit další chyby.

Sestavení projektu je vlastně jen problém napsání konfiguračních souborů. Zásadní komplikace je totiž nedostatek dokumentace. Pokud je zavolána jakákoli funkce v ALIZE, většinou bere jako parametr jméno konfiguračního souboru a v dokumentaci je také popsáno, že parametr je to jméno konfiguračního souboru. Co ale popsáno není, je jaké parametry a v jakém formátu tato funkce z konfiguračního souboru čte. Jediná možnost, jak to zjistit, je reverzní inženýring. Musel jsem program spustit, počkat na výjimku, dát před ní breakpoint, spustit ještě jednou, vydedukovat co se asi stalo a která proměnná je nastavena špatně. Z trochou štěstí dám breakpoint o kousek v programu dříve a proces opakuji, dokud nedojdu ke čtení z konfiguračního

souboru. Když si uvědomíme , že je potřeba asi 13 konfiguračních souborů a každý má v průměru okolo 20 aktivních řádků, je jasné, že tato část práce je opravdu zdlouhavá.

Výsledkem práce je, jak jinak, soubor s výsledky. Vytvořil jsem z něj nový, doplnil ho pro přehlednost jmény vzorků, setřídil a vypsal na konzoli.

4.7 Výběr segmentů pomocí detekce energie

Souvislý blok příznaků nazývají autoři ALIZE segmentem. Ve druhé části programu, ihned po normalizaci příznaků jsou vytvořeny seznamy promluv nalezených ve vzorku a to na základě detekce energie. Seznam je uložen v souborech prm/lst/*.lst . ALIZE pracuje tak, že spočítá nějakou statistickou (Gaussovo rozdělení) funkci za všech vzorků a pak určí prahovou úroveň–hranici, pro kterou by se vybral určitý počet vzorků a vybere všechny vzorky vyšší než stanovený práh.

Systém ovšem, tak jak je, vybral i méně než 3% vzorků, přestože jde o normální promluvu. Pokud se vybere 80% vzorků s nejvyšší energií, kvalita testování se podstatně zlepší. Pořád je tu ale problém, že nevíme kolik procent času mluvčí mluví a kolik je doba šumu mezi promluvami. Tak se může stát, že porovnáváme šum z okolí, kvalitu mikrofonu, nežádoucí pozadí.

Jako řešení jsem zvolil 3 průchody původní, ale odlehčenou funkcí. V prvním vyberu promluvy, 95% všech. Pak projdu výsledné segmenty a vyberu právě ty, které vybrány nejsou. Funkce rovněž vrátí práh. Máme tak vzorky, které by zcela jistě měly být šum mezi slovy. Z těchto vyberu 95% nejvyšších a dostanu nový práh. Na základě jeho vezmu o 20 procent větší prahovou úroveň.

$$final\_threshold = newthreshold + (threshold - newthreshold) * 0.2 \tag{4.1}$$

Je trošku těžké toto nějak otestovat, ale nové procento vybraných segmentů nyní více odpovídá rozložení mezer a slov v promluvě.

5 TESTOVÁNÍ MINIMÁLNÍ POTŘEBNÉ DÉLKY VZORKU

5.1 Cíle testu

Spolehlivost je základním kritériem identifikace řečníka. Proto by bylo potřeba napsat nějaký zátěžový test, který pro pro maximální počet vzorků prověřil, zda identifikace funguje.

Výpočetní výkon je asi druhým nejzákladnějším kriteriem. Ideálně bychom chtěli vypočítat shodu testovaného vzorku se vzorky natrénovanými co nejrychleji.

Délka vzorku je důležitá pro porovnávání v reálném čase, kde kromě co nejmenšího procesorového času potřebujeme začít s testováním co nejrychleji po začátku nahrávání vzorku.

Efektivita se kterou testujeme. Pokusíme se zjistit poměr mezi délkou vzorku na trénování a délkou vzorku pro test.

5.2 Algoritmus testu

V principu provedeme tři vnořené cykly.

1. Pro různé délky promluvy, konkrétně hodnoty 0.032, 0.048, 0.08, 0.112, 0.16, 0.32, 0.48, 0.8, 1.12, 1.6, 3.2, 4.8, 8, 11.2, 16 natrénujeme standardním způsobem.
2. Pro různé délky promluvy, konkrétně hodnoty 0.032, 0.048, 0.08, 0.112, 0.16, 0.32, 0.48, 0.8, 1.12, 1.6, 3.2, 4.8, 8, 11.2, 16 testujeme všechny vzorky standardním způsobem.
3. Přesněji vzorky ze souboru **all.lst** a spočítáme kolik vzorků uspělo.

Nové je zkrácení vzorků. Aplikace pro detekci energie vygeneruje seznam segmentů s promluvou. Více o této problematice je uvedeno v kapitole 4.7. Nová funkcionalita testu tento seznam zkrátí tak, aby celková délka všech segmentů byla právě jedna z délek uvedených výše. Porovnáváme tak určitě délku promluv, nikoli délku vstupního souboru i s mezerami mezi slovy.

Protože vzorky od jednoho mluvčího se liší ve jménu souboru pouze posledním dvoučíslím, je porovnaní, zda se srovnání podařilo, velmi jednoduché. Bohužel, zejména pro krátké segmenty se často stává, že výsledkem je *Inf* nebo *NaN*, někdy

je hodnota všech výsledků *0.0*. Pokud něco z tohoto nastane, zvýším počitadlo nesmyslných výsledků. Takže kromě tabulky shod bude vytvořena i podobná tabulka neúspěchů.

5.3 Průběh testů

Před testem jsem musel opravit asi tři chyby v ALIZE, protože zejména pro krátké vzorky řeči vyhazovala výjimky.

Rovněž je problém, pokud je pouze jeden „podvodník“. Testy velmi často dopadnou neúspěchem, tedy vracením *NaN* nebo *Inf*.

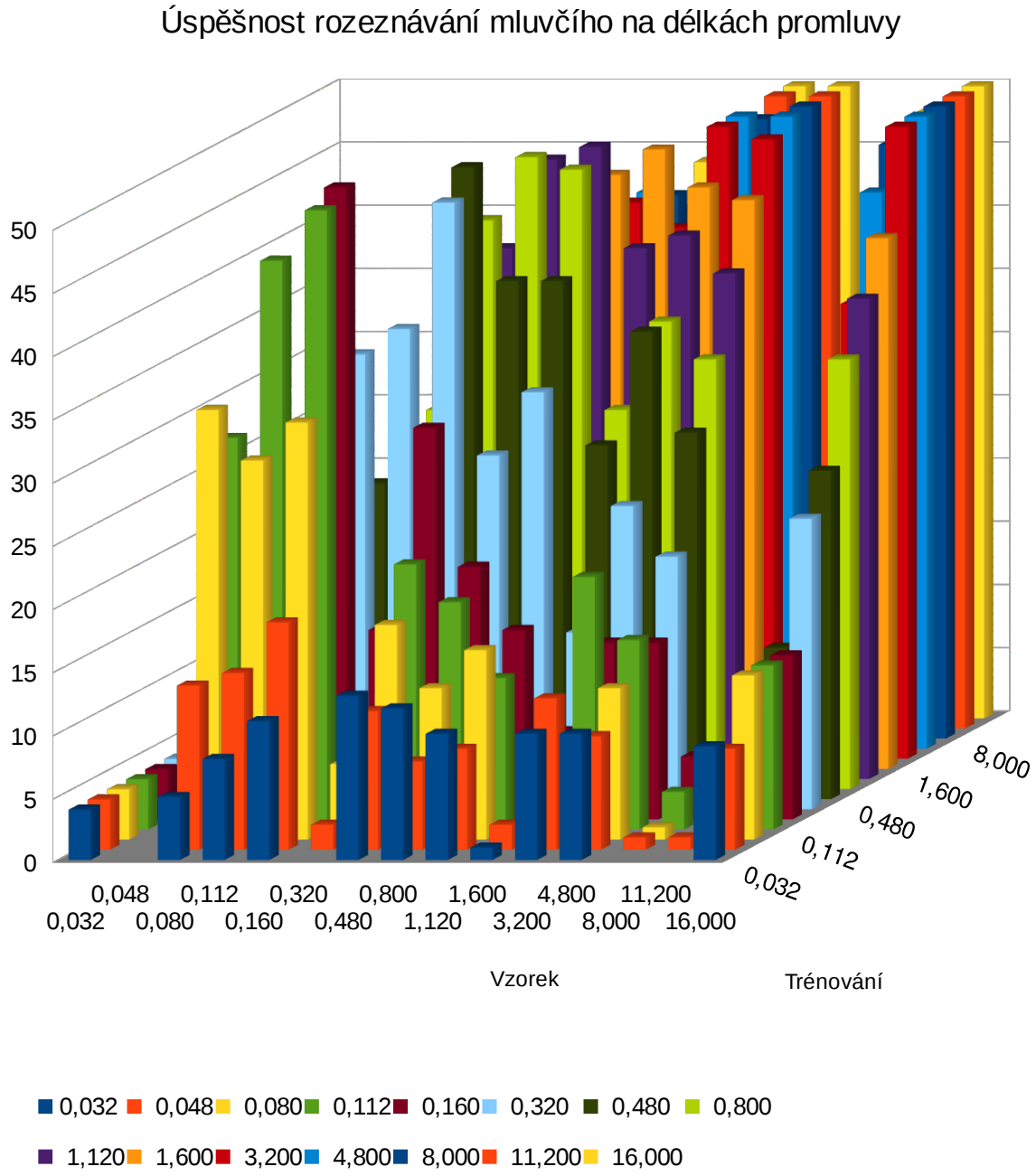
Test trvá několik hodin pro 27 vzorků natrénovaných a 23 vzorků k testování.

5.4 Výsledky testů

5.4.1 Test každého s každým – 50 vzorků

Jako první jsem udělal test každého s každým ze všech vzorků, co jsem měl k dispozici. Tento test je pro velký počet kombinací extrémně náročný na procesorový čas, trval přes 5 hodin na mém AMD Vishera. Věrohodnost je možná diskutabilní, protože by měl vždy vybrat jeden a tentýž vzorek.

Trén.	Délka vzorku														
délka	0.032	0.048	0.080	0.112	0.160	0.320	0.480	0.800	1.120	1.600	3.200	4.800	8.000	11.200	16.000
0.032	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0.048	0	0	0	0	0	3	3	2	5	7	6	9	9	9	5
0.080	5	13	34	31	22	16	13	12	15	11	13	16	14	14	14
0.112	8	14	30	45	31	17	20	15	13	13	15	15	18	16	16
0.160	11	18	33	49	50	36	25	21	20	20	19	14	17	15	14
0.320	0	2	6	3	15	38	29	30	21	17	17	18	20	22	18
0.480	13	11	17	21	31	48	50	45	42	40	32	31	34	34	33
0.800	12	7	12	18	20	28	41	50	49	44	40	34	35	32	31
1.120	10	8	15	12	15	33	41	49	50	47	44	44	43	41	44
1.600	1	2	1	3	7	14	28	30	42	49	42	37	33	33	36
3.200	10	12	8	20	14	24	37	37	43	46	50	50	49	50	50
4.800	10	9	12	15	14	20	29	34	40	45	49	50	50	50	50
8.000	0	1	1	3	5	1	4	2	9	9	10	21	23	27	25
11.200	0	1	0	1	3	8	12	11	20	21	36	44	47	47	48
16.000	9	8	13	13	13	23	26	34	38	42	50	50	50	50	50

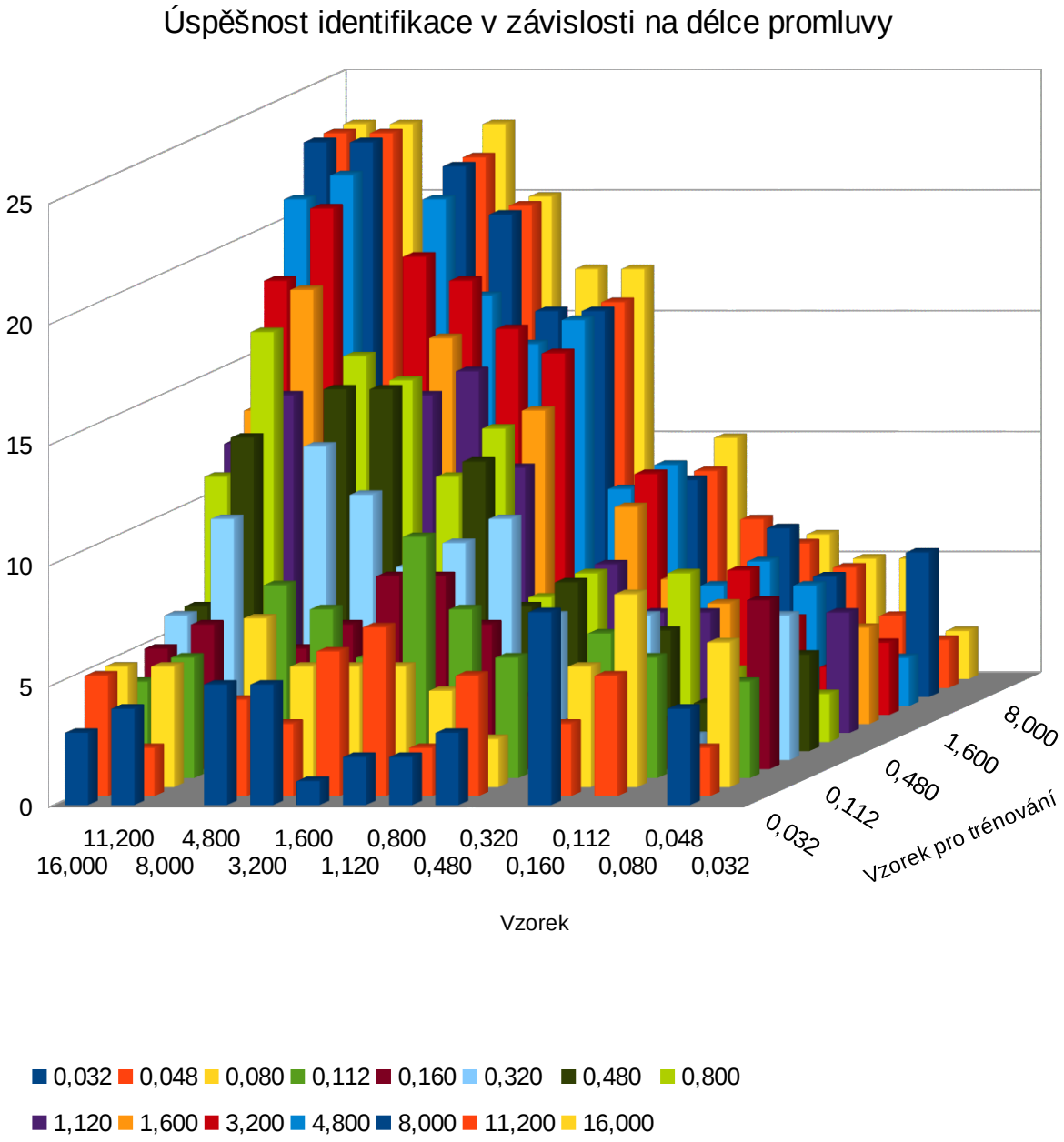


Obr. 5.1: Test každého s každým

5.4.2 Objektivnější test

V tomto testu je rozdělen počet vzorků na 2 skupiny. Testovací skupina má 23 vzorků a vzorky použité k trénování mají 27 vzorků. Test by měl být objektivnější, protože se nemůže stát, že se boudou porovnávat stejné vzorky.

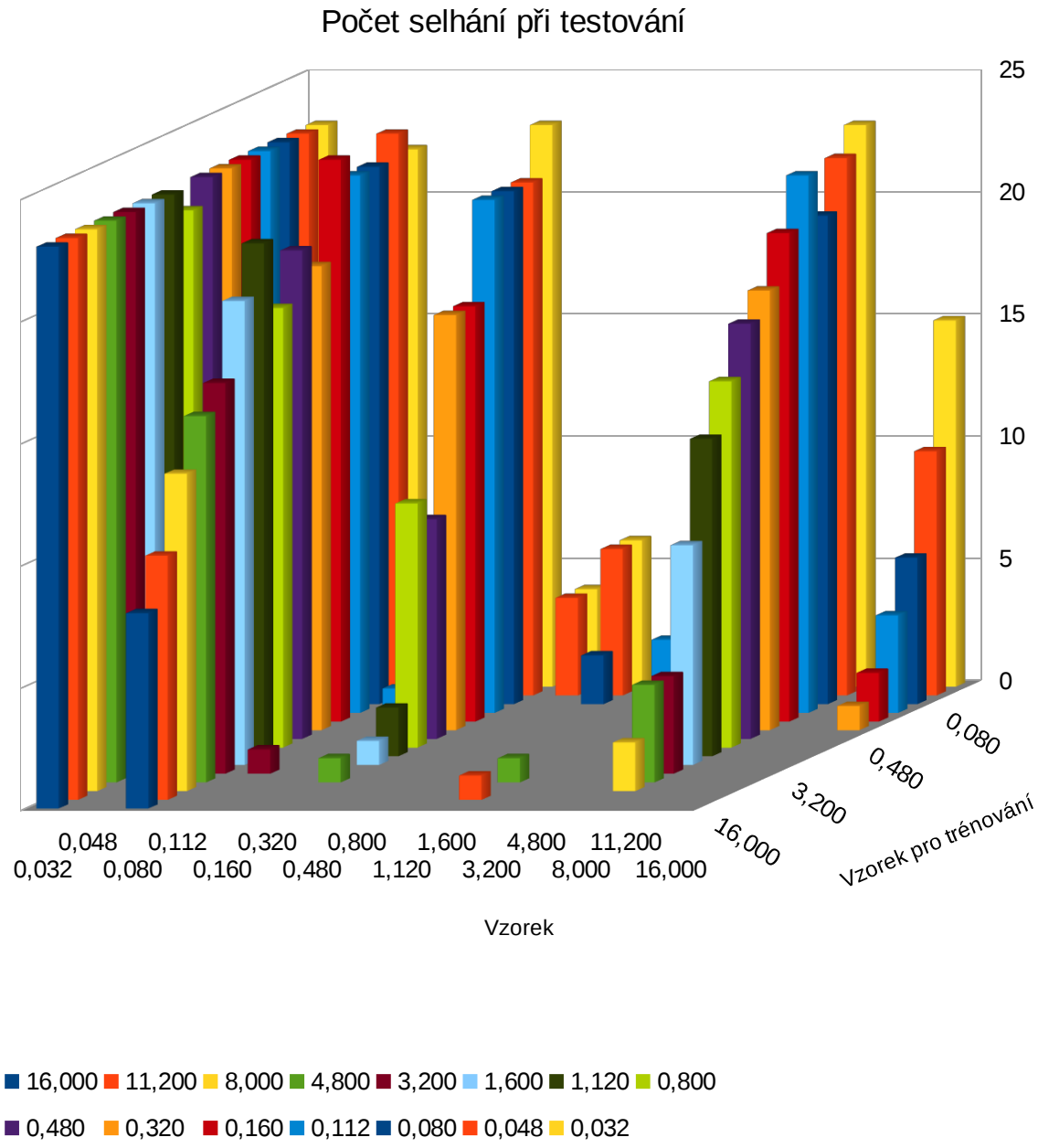
Trén.	Délka vzorku														
délka	16.000	11.200	8.000	4.800	3.200	1.600	1.120	0.800	0.480	0.320	0.160	0.112	0.080	0.048	0.032
16.000	23	23	23	21	18	13	12	11	6	6	5	4	5	5	3
11.200	23	23	23	22	21	18	14	17	13	10	6	5	5	2	4
8.000	16	12	10	12	9	8	3	2	1	1	0	0	0	0	0
4.800	23	22	22	21	19	13	14	16	15	13	5	8	7	4	5
3.200	20	20	20	17	18	16	14	15	15	11	6	7	5	3	5
1.600	17	14	16	15	16	11	15	11	7	8	8	5	5	6	1
1.120	17	16	16	16	15	13	11	13	12	9	8	10	5	7	2
0.800	8	7	7	9	5	5	4	6	6	10	6	7	4	2	2
0.480	10	9	9	10	10	9	7	7	7	6	3	5	2	5	3
0.320	4	7	4	5	3	6	5	4	3	2	2	1	0	0	0
0.160	6	6	7	6	6	5	5	7	5	6	5	6	5	3	8
0.112	5	5	5	5	4	3	1	3	2	1	2	5	8	5	0
0.080	5	3	2	2	2	1	0	1	1	1	0	0	0	0	0
0.048	2	2	6	2	3	4	5	2	4	6	7	4	6	2	4
0.032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Obr. 5.2: Test objektivnější

Často se stává, že aplikace nejen že neoznačí správného řečníka, ale dokonce vrátí tabulku *NaN* nebo *Inf* nebo celou tabulku nulovou. To jsem rovněž zaznačil do tabulky jako počet selhání.

Trén.	Délka vzorku														
délka	16.000	11.200	8.000	4.800	3.200	1.600	1.120	0.800	0.480	0.320	0.160	0.112	0.080	0.048	0.032
16.000	0	0	0	0	0	0	0	0	0	1	2	4	6	10	15
11.200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8.000	0	0	2	4	4	9	13	15	17	18	20	22	20	22	23
4.800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.600	0	1	0	1	0	0	0	0	0	0	0	3	3	6	12
1.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.800	0	0	0	0	0	0	0	0	0	0	0	0	2	6	6
0.480	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4
0.320	0	0	0	1	0	1	2	10	9	17	17	21	21	21	23
0.160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.112	0	0	0	0	1	0	0	0	0	0	0	1	0	2	7
0.080	8	10	13	15	16	19	21	18	20	19	23	22	22	23	22
0.048	0	0	0	0	0	0	0	0	0	0	1	1	0	4	0
0.032	23	23	23	23	23	23	23	22	23	23	23	23	23	23	23



Obr. 5.3: Test selhání

5.5 Závěrem k testování

Z grafu „objektivnějšího“ je vidět řada zajímavých věcí. Závislost přesnosti na délce vzorku je přibližně logaritmická, jak pro délku testovací promluvy, tak pro délky promluvy na trénování. Přitom není téměř rozhodující, zda se jedná o vzorek pro trénování nebo pro test. Na špičce, kde jsou všechny hodnoty 23 (100procentní), je také vidět, že je lépe použít delší vzorky pro trénování.

Pokud tedy budeme trénovat promluvy v délkách na 11 vteřin a testovat s promluvami délek alespoň 8 vteřin, bude **spolehlivost** velmi slušná, ačkoli se dá očekávat, že s větší databází mluvčích bude ještě stále docházet k omylům.

Z testu selhání je vidět, že aplikace pro délky promluv pod asi 0,5 vteřiny je nejen neúčinná, ale nefunguje vůbec.

Ještě je z grafu(graf každý s každým) vidět, že délka vzorku k testování a délka vzorku k natrénování by měly být přibližně stejné.

6 ZÁVĚR

ALIZE je velmi zajímavým projektem v oblasti zpracování řeči. Je provedena ve standardní kvalitě a působí dokončeným dojmem.**Celkově je ALIZE napsaná velmi pěkně**, má dobrou strukturu. Pan Wills, který ji psal, byl zcela jistě dobrý programátor. Všechny problémy, popsané v této práci poukazují na to, že ALIZE je málo dokumentovaná, a co horší, je nevyzkoušená, takže v ní nějaké chyby prostě jsou. Také je nutné počítat s faktem, že ALIZE je „po smrti“, tj. že minimálně dva roky už do ní nikdo nepřispívá a ani ji neopravuje.

Důležitým přínosem pro mě, ale i zájemce o zpracování řeči, je překlad dokumentace z francouzštiny do angličtiny. Dále jsem navrhl vhodnou koncepci projektu a to na základě řady nashromážděných zkušeností z několika univerzit, kde se analýzou řeči zabývají. Součástí takovéto práce byl zejména výběr vhodných příznaků, nahrání vhodných testovacích vzorků, výběr platformy a vývojového prostředí, návrh použitelné kostry programu, implementace v jazyce C++ a testování výsledků. Není ani nepodstatné, že jsem strávil nějaký čas studováním a laděním ALIZE a mám tedy velmi cenné zkušenosti do další práce. Opatřil jsem si několik, cca 10 vzorků mluvčích pro otestování funkcionality, nicméně je jich méně než je v zadání. Přesto se potvrdilo, že systém jako celek funguje. Na závěr jsem udělal test závislosti spolehlivosti na délce promluvy.

Práce na této tezi je výzvou, jak nejen splnit zadání bakalářské práce, získat zajímavé znalosti, ale i podílet se na vývoji open–source knihovny, která bude sloužit všem zájemcům o problematiku zpracování řeči.

LITERATURA

[Bon03] Bonastre J.F., Bimbot F., Boe L.J., Campbell J.P., Douglas D.A., Magrin-chagnolleau I., *Person Authentification by Voice: A Need of Caution*: Eurospeech, 2003

[Psut95] Psutka J., *Komunikace s počítačem mluvenou řečí*: Academia, 1995

[Psut05] Psutka J., Müller L., Matoušek J., Radová V., *Mluvíme s počítačem česky*: Academia, 2005

[Heb88] Hebák P., Kahounová J., *Počet pravděpodobnosti v příkladech* : 3. vydání, SNTL 1988

[Gan07] Gannert T., *A Speaker Verification System Under The Scope: Alize* : KTH, 2007

[Sam09] Samudravijaya K., *Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM)* : Tata Institute of Fundamental Research, Mumbai

SEZNAM PŘÍLOH

A INSTALACE PROJEKTU

A.1 Instalace virtuálních strojů

Protože je ALIZE multiplatformní, rozhodl jsem se toho využít a pokusit se o překlad a spuštení na všech třech základních platformách. Nainstaloval jsem si tři virtuální stroje v programu VirtualBox od Oracle. VirtualBox má tu výhodu, že je jednoduchý na ovládání a obrazy disků jsem v něm 100procentně přenositelné. Na každé z těchto platforem je nějaké úskalí, proto bude vhodné se o své zkušenosti podělit.

Hotové virtuální stroje jsou uloženy na blue–ray jako příloha této práce.

Všechny operační systémy mají společného uživatele *virtualbox* a společné heslo *410virtual410*.

A.1.1 Instalace na OSX 10.10 Yosemite

Napsání této kapitoly stálo dost zkoušení a shánění informací na internetu. Návodů je sice dost, ale ne každý návod na zprovoznění Mac je opravdu funkční. Mohlo by se zdát, že to nepotřebuji, protože sám vlasním počítač Apple, ale chtěl jsem, aby bylo možno vyzkoušet projekt na akademické půdě a porovnat tak s ostatními platformami.

Důvodem té „složitosti“ je svázání operačního s určitým konkrétním hardwarem, který je třeba ve VirtualBox nasimulovat.

Operační systém OSX je zdarma ke stažení na Apple AppStore. Přestože je zdarma, Apple jej licencuje uživatelům Applího hardware, k instalaci do virtuálních strojů určen není.

Malým nedostatkem tohoto návodu je závislost na OSX, nemůžete tento návod tedy provádět na Linuxu nebo Windows. Na druhou stranu,vytvořený virtuální stroj je přenositelný bez problému. Spouštění virtuálních strojů na operačním systému, na kterém nebyly vytvořeny, jsem prakticky ověřil.

Pokud si jej tedy stáhneme, získáme aplikaci, která po zpuštění na předchozí verzi OSX 10.9 Mavericks povýší systém na OSX 10.10 Yosemite. Co je ale důležité, z balíčku aplikace je možné vytvořit obraz disku, který se dá do virtuálního stroje nainstalovat.

Jako první je dobré ověřit, zda Váš procesor podporuje virtualizaci. Pokud ne, nemám následující řádky ověřeny.

```
sysctl -a | grep machdep.cpu.features
```

Pokud vidíte jako jednu z položek „VMX“ nebo „VT-x“, tak Váš počítač podporuje virtualizaci.

Nejprve je třeba získat program iesd. Pro instalaci jsem použil instalátor gem. Měl by být standardní součástí každého počítače od Applu.

```
sudo gem install iesd
```

Použijeme ho k tomu apbychm provedli extrakci obrazu (Applího) z aplikace, v našem případě „Install OS X 10.10 Developer Preview“ do souboru yosemite.dmg.

```
iesd -i /Applications/Install\ OS\ X\ Yosemite.app -o yosemite.dmg
-t BaseSystem
```

Následně vytvoříme balík, který není komprimovaný.

```
hdiutil convert yosemite.dmg -format UDSP -o yosemite.sparseimage
```

Je třeba připojit obraz staženého disku, který je součástí instalačního balíčku OSX.

```
hdiutil mount /Applications/Install\ OS\ X\ Yosemite.app/Contents/
SharedSupport/InstallESD.dmg
```

A také námi vytvořeného.

```
hdiutil mount  yosemite.sparseimage
```

Potom se dají kopírovat soubory.

```
cp /Volumes/OS\ X\ Install\ ESD/BaseSystem.* /Volumes/
OS\ X\ Base\ System/
```

Nyní můžeme oba svazky odpojit. Dá se použít příkaz umount, nebo prostě kliknou na ikonku vysunutí disku ve Finderu.

```
hdiutil convert osx-yosemite.sparseimage -format UDZO -o
osx-yosemite3.dmg
```

Vytvořený soubor yesemite3.dmg je ten pravý obraz disku, který připojíte ve VirtualBoxu jako CD a bude fungovat.

Ještě než se ale tak stane, je třeba vytvořit a nakonfigurovat virtuální stroj. Začnete tím, že vyberete z menu vytvořit nový a jako hostovaný systém zvolíte OSX. Dále je třeba nastavit hardware. V nastavení, v položce sytém, záložce základní

deska, zvolíme čipovou sadu ICH9, povolíme IO APIC a povolíme EFI. Hardwarový čas naopak odškrtneme. V záložce procesor zvolíme jedno jádro (více jader bohužel virtuální stroj nějak neumí využít) a zaškrtneme povolit PAE/NX. Poslední důležité nastavení je v záložce akcelerace, kde zaškrtneme obě položky, povolit VT-x/AMD-V a povolit přímou komunikaci s hardwarem.

Pokud tedy mám virtuální hardware vytvořený, musíme jej upravit napsáním následujícího příkazu. Ačkoliv se jedná o funkcionalitu VirtualBoxu, „vyklikat“ to nejde.

```
VBoxManage modifyvm <Name of virtual machine> --cpuidset 00000001
000306a9 00020800 80000201 178bfbff
```

Tím už jsme skoro u konce. Ještě rozlišení našeho nového Maca je velmi malé. Následující řádek to pomůže zlepšit.

```
vboxmanage setextradata <Name of virtual machine> VBoxInternal2/EfiGopMode <N>
```

Kde N je 0,1,2,3,4,5 znamenajících 640x480, 800x600, 1024x768, 1440x900, 1900x1200.

A.1.2 Instalace na Ubuntu 14.04

K instalaci je potřeba iso soubor - obraz disku Ubuntu. Já jsem ho získal z webu Ubuntu, konkrétně z `http://www.ubuntu.cz/ziskejte/stahnout` (64bitová verze).

Stačí jen zvolit defaultní nastavení pro Windows 8.1 při vytváření stroje, ale je potřeba větší disková kapacita, tak okolo 50GB.

Po instalaci je je vše plně funkční, rozlišení je však velice malé. To lze vylepšit instalačním příkazem níže. Pozor! Zadává se do hostovaného virtuálního stroje.

```
sudo apt-get install virtualbox-guest-dkms virtualbox-guest-utils
virtualbox-guest-x11
```

Rozlišení je nyní stejné jako rozlišení monitoru.

Ubuntu jsou ve virtuálním stroji (Macbook pro s Intel i7) svižné, výkon je mnohem vyšší než výkon potřebný k přehrávání videa s Youtube.com.

A.1.3 Instalace na Windows 8.1

K instalaci je potřeba iso soubor – obraz disku windows. Já jsem získal školní windows na webu DreamSpark, přístupného přes `http://msdnaa.feec.vutbr.cz/web/`.

Stačí jen zvolit defaultní nastavení pro Windows 8.1 při vytváření stroje, ale je potřeba větší disková kapacita, tak okolo 50GB.

```
VBoxManage modifyhd <Jmeno_vaseho_stroje> --resize 60000
```

Rozlišení lze měnit, ale k dispozici je jen poměr 4:3 a maximální rozlišení 1600x1200.

Windows jsou ve virtuálním stroji (Macbook pro s Intel i7) svižné, výkon je mnohem vyšší než výkon potřebný k přehrávání videa s Youtube.com.

To rozlišení jde opravit, stejně jako v předchozím případě napsáním

```
VBoxManage.exe setextradata "Win 8 dev" CustomVideoMode1 1680x1050x32
```

A.2 GNU C++

A.2.1 OSX Yosemite

Není dobré pokoušet se instalovat GNU C++ z internetových stránek. Apple má na překladači nějaké modifikace a je tedy vhodné použít jeho překladač. Já jsem to udělal tak, že jsem nainstaloval vývojové prostředí Xcode a jako jeden z jeho balíčků přidal Xcode command line tools.

A.2.2 Ubuntu Linux

C++ je jeden z balíčků každé distribuce. Stačí napsat:

```
sudo apt-get install g++
```

A.2.3 Windows 8.1

Pro Windows existuje disrtibuce GNU cygwin, kterou je možno stáhnout a nainstalovat z webu `sourceware.org/cygwin`.

Důležité je nezapomenout zašktnout devel a debug v seznamu balíčků k instalaci.

A.3 NetBeans IDE

A.3.1 OSX Yosemite

V případě NetBeans lze ze stránek `netbeans.org/downloads` klasický instalační dmg balíček. Při instalaci nejsou problémy, ale je nutné povýšení virtuálního stroje javy, nejlépe z URL `www.java.com/en/download`.

Další problém je, že nefugují debuggery. Apple má problém s debugerem gdb. Pro své aplikace je nabízen gdb s Applí verzí gdb–apple. Ten ovšem s NetBeansem nefunguje. Další kandidát je Applí lldb. Ani ten nefunguje správně s NetBeans. Nakonec

jsem nainstaloval nejnovější debugger pomocí balíčkovacího nástroje homebrew. Ten se dá dostat do počítače takto.

```
ruby -e \"$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
install/master/install)\"
```

Pak debugger.

```
brew install https://raw.github.com/Homebrew/homebrew-dupes/master/
gdb.rb
```

Následující je nutné. Debugger je nutno podepsat, jinak jej apple odmítne spustit. Zde je postup na OSX Yosemite.

- Otevřete „Klíčenku“.
- Otevřete menu „Klíčenka/Průvodce certifikací/Vytvořit certifikát“.
- Vyberte jméno (gdb-cert), typ identity na „Kořen podepsaný sám sebou“, typ certifikátu na „Podepisování kódu“ a zašktnout „Nechat přepsat výchozí hodnoty“.
- Asi budete chít prodloužit platnost 3650 dní
- Několikrát kliněte na pokračovat, než se dostanete na stránku „Umístění pro certifikát“ a nastavte „Systém“.
- nakonec je potřeba jít do seznamu certifikátu v klíčence a kontextovém menu nad gdb-cert zvolit info. v řádku „Podepisování kódu“ vybrat „Vždy důvěřovat“

Pak je nutné uzavřít démona „taskgated“

```
sudo killall taskgated
```

A nakonec podepíšeme příkazem „codesign“.

```
codesign -s gdb-cert -f /usr/local/Cellar/gdb/7.9/bin/gdb
```

Bohužel, ani toto řešení není dokonalé. Pokud se totiž, zanořím třeba do desáté funkce, ztratí se nějak informace o zásobníku a další práce přestává být možná.

A.3.2 Ubuntu Linux

NetBeans je distupný jako klasický deb balíček, bohužel verze nabízená na repozitáři je zastaralá. Proto je lepší stáhnout si ze stránek `netbeans.org/downloads` aktuální. I zde je vhodné povýšit verzi viruálního stroje javy, vyzkoušel jsem open-jdk-7. Instalace je bez problémů.

A.3.3 Windows 8.1

Instalce z `netbeans.org/downloads` aktuální. I zde je vhodné povýšit verzi viruálního stroje javy.

A.4 GNU Octave

A.4.1 Funkce LPC v Octave

Funkce LPC není implementována standartdně, dokonce ani v balíčku signal tak jako v matlabu. Proto je nutné ji doinstalovat z balíčku tsa.

A.4.2 OSX Yosemite

GNU octave lze nainstalovat ze stránek `octave.org`.

Ještě je potřeba balíček tsa. Pokud octave spustíme napíšeme do něj:

```
pkg install -forge tsa
```

A.4.3 Ubuntu Linux

Balíček octave je v každé distribuci, stačí tedy napsat:

```
sudo apt-get install octave
```

Ještě je potřeba balíček tsa kvůli funkci LPC.

```
sudo apt-get install octave-tsa
```

A.4.4 Windows 8.1

GNU octave lze nainstalovat ze stránek `octave.org`.

Ještě je potřeba balíček tsa. Pokud octave spustíme napíšeme do něj:

```
pkg install -forge tsa
```

A.5 Verzovací systém Git a grafické rozhraní SourceTree

A.5.1 Git

Verzovací systém Git používám nejen v zaměstnání, ale i v domácích projektech a má své místo i v této práci. I v bakalářské práci je totiž potřeba přenášet aplikaci mezi

mým počítačem, notebookem a všemi třemi virtuálními stroji. Může být důležité moci se v případě nutnosti vrátit k přechozím verzím projektu.

Vytvořil jsem speciální repozitář na svém testovacím serveru. Přihlašovat se jde výhradně pomocí klíče, který najdete na CD–příloze této práce. Projekt stačí už jen naklonovat příkazem,

```
git clone virtualbox@46.167.245.175:bp
```

případně adresu vložit do SourceTree.

K programu Git vznikl vynikající nástroj SourceTree od společnosti Atlassian. V něm má uživatel dobrý přehled o příspěvcích do jeho pracovního i společného repozitáře a hlavně aktuálnosti jeho pracovní kopie. Je ke stažení zdarma na stránkách `https://www.atlassian.com/software/sourcetree/overview`. Je podporován OSX a Windows, Linux bohužel podporován není.

A.5.2 OSX Yosemite

Nejefektivnější je pouze naistalovat SourceTree z webových stránek uvedených výše. Tím se automaticky nainstaluje i Git.

S klíči se zachází standartním způsobem, měly by být uloženy v adresáři ~/.ssh .

A.5.3 Ubuntu Linux

Git je standatdní balíček snad všech distribucí. Instale je jako obvykle napsáním

```
sudo apt-get install git
```

S klíči se zachází standartním způsobem, měly by být uloženy v adresáři ~/.ssh .

Pro Git je na Linuxu celá řada grafických prostředí, ale já po zkušenostech s nimi raději používám Git z příkazové řádky. Žádné z nich nedosahuje kvalit SourceTree.

A.5.4 Windows 8.1

Nejefektivnější je pouze naistalovat SourceTree z webových stránek uvedených výše. Tím se automaticky nainstaluje i Git.

SourceTree pro Windows má v menu „nástroje“ nové položky „klíče“ a „spustit ssh agenta“. Do prvního je třeba zadat klíč z CD a druhý krok je spustění agenta. Ssh agent funguje, ačkoli po jeho vybrání z menu to vypadá, že se nic nestalo.

B POUŽÍVÁNÍ PROGRAMU

B.1 Formát dat

Vstupní data programu jsou uložena v adresáři *data*. Typově se jedná jen o dva druhy souborů. Audio soubory ze vzorky a seznamy. Jako audio soubor je zvolen...

```
*.wav, 16000Hz, mono, cca 20sec dlouhý
```

Souborů tohoto typu může být neomezené množství, ale zpracování pak zabere svůj čas.

Další dva soubory mají stejný formát. Je to seznam, kde na každém řádku je jméno souboru (.wav) bez přípony. První je seznam všech vzorků, má jméno **data.lst** a druhý je seznam „podvodníků“ **impostor.lst**. Impostor.lst je podmnožina souboru data.lst a všechny řádky v něm musí být i v souboru data.lst.

B.2 Trénování

Když si připravíme data jako dle instrukcí v předchozí sekci, stačí spustit trénování...

```
/project/dist/Debug/GNU-Linux-x86/project --train
```

Chvíli to trvá, zejména extrakce příznaků v GNU Octave je pomalá.

B.3 Testování vzorku

Testování vzorků je podstatně rychlejší operace než trénování, zejména se extrahují příznaky pouze jednoho souboru.

```
/project/dist/Debug/GNU-Linux-x86/project --test <soubor_s_cestou>
```

Poslední řádky výpisu jsou tabulka. Na prvním místě je soubor s největší shodou a jeho jméno. Měl by patřit tomu samému mluvčímu jako testovací soubor. Tabulka je rovněž uložena v souboru **result.txt** a aktuálním adresáři.

B.4 Test minimální délky promluvy

Vstupní data jsou jako obvykle v adresáři data. Nový je soubor **all.lst**, ve kterém jsou očekávány jména všech vzorků pro testování proti standardnímu souboru **data.lst**.

```
/project/dist/Debug/GNU-Linux-x86/project --ultest
```

Vykonávání pro 50 vzorků je na několik hodin.
Více o tomto testu je v kapitole 5.