VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ROZPOZNÁNÍ HLASOVÝCH PŘÍKAZŮ V AUDIOSIGNÁLU

VOICE COMMANDS RECOGNITION IN AUDIOSIGNAL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ŠRÁMEK

AUTHOR

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

SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Martin Šrámek

který/která studuje v **bakalářském studijním programu**

obor: **Mechatronika (3906R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Rozpoznání hlasových příkazů v audiosignálu

v anglickém jazyce:

Voice commands recognition in audiosignal

Stručná charakteristika problematiky úkolu:

Zpracujte výpočetně nenáročný systém na detekci a rozpoznání předem určených jednoslovných příkazů v audiosignálu.

Cíle bakalářské práce:

1. proveďte rešeršní studii způsobu detekce a rozpoznávání vzorů v audiosignálu
2. vyberte implementačně nenáročnou metodu
3. metodu zpracujte nejprve offline např. prostředí Matlab
4. v případě úspěšného zpracování offline implementujte metodu online.

Seznam odborné literatury:
http://www.mperfect.net/noReco/

Vedoucí bakalářské práce: Ing. Jiří Krejsa, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne 22.10.2009

L.S.

prof. Ing. Jindřich Petruška, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc. Děkan fakulty

Abstrakt

Držíte v rukách bakalársku prácu, ktorá sa zaoberá návrhom a realizáciou systému pre rozpoznávanie izolovaných hlasových príkazov. Motiváciou k vytvoreniu práce bol záujem o diaľkové ovládanie robotických mechanizmov pomocou hlasu a bližšie skúmanie problému spracovania rečových signálov. V dnešnej dobe je široko rozvinutý. Práca je rozdelená na dve časti. Prvá časť sa zaoberá zosumarizovaním poznatkov o rozpoznávaní, v druhej časti sú tieto poznatky využité v návrhu systému.

Abstract

You are holding in your hands the Bachelor thesis which deals with design and realizing of isolated voice recognition system. The motivation of this thesis was an interest in remote control of robotic mechanisms by voice and a research of speech signal processing. It is widely developed these days. The thesis is divided into two parts. The first one is concerned with summarizing of recognition knowledge, in the second one this knowledge is used in design of a system.

Kľúčové slová

rozpoznávanie príkazov, rozpoznávanie reči, dynamické skreslenie časovej osi, skryté markovove modely, neurónové siete, spektrogram

Key words

commands recognition, speech recognition, dynamic time warping, hidden Markov models, neuron nets, spectrogram

Bibliografická citácia

ŠRÁMEK, M. *Rozpoznání hlasových příkazů v audiosignálu.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 45 s. Vedoucí bakalářské práce Ing. Jiří Krejsa, Ph.D.

Čestné prehlásenie

Ja, Martin Šrámek, čestne prehlasujem, že som túto prácu vypracoval samostatne pod dohľadom vedúceho práce a s prispením zdrojov uvedených v zozname použitej literatúry.

_____

Martin Šrámek

Poďakovanie

Na tomto mieste by som chcel poďakovať predovšetkým vedúcemu mojej bakalárskej práce pánovi Ing. Jiřímu Krejsovi, Ph.D. za cenné pripomienky a pomoc pri návrhu systému a riešení problémov. Ďalej patrí vďaka mojej rodine za podporu pri štúdiu a v neposlednom rade chcem poďakovať Marekovi Huškovi za nahratie hlasových dát do slovníka.

Obsah

Kapitola 1

Úvod

Rozpoznávanie reči je v súčasnej dobe veľmi rozvinutým fenoménom. Rozvojom robotiky produkujúcej mechanizmy inšpirované biologickou (dvojnohé, štvornohé roboty) i priemyselnou (hybridné, kolesové roboty) sférou, vzrastala tendencia rečovej komunikácie s týmito strojmi. Rozpoznávače slov (angl. voice recognizers) dnes poznáme z mobilných telefónov, počítačových aplikácií, telefónnych záznamníkov a softvérov pre hendikepovaných ľudí.

Cieľom tejto práce je vyvinúť softvér, ktorý bude rozpoznávať jednoduché príkazy – povely. Po úspešnom spracovaní algoritmu bude vložený ako aplikácia do robotického systému. Systém reaguje na hlasové podnety z prostredia, pričom je schopný daný príkaz zaradiť do reprezentujúcej triedy vo vopred nadefinovanom slovníku. Ide o rozpoznávač s malou databázou vzorov, kde sa pracuje iba s povelmi týkajúcimi sa chodu robota. Tzn. príkazy „Jeď“, „Vlevo“, „Vpravo“, „Stát“ a i.

Práca je rozdelená na dve časti. Prvá – rešeršná časť vťahuje čitateľa do problematiky spracovania rečových signálov, vysvetľuje základné pojmy a tvorí základ pre pochopenie rozpoznávania rečových príkazov. V tejto časti je spracovaný prehľad najpoužívanejších spôsobov rozpoznávania vzorov v audiosignále. Druhá časť sa zaoberá implementáciou vlastného algoritmu pre rozpoznávač. Úloha je spracovávaná v prostredí Matlab, ktorý je spolu Signal Processing Toolboxom silným nástrojom a dobrým pomocníkom pri spracovaní reči.

Kapitola 2

Hlasový signál

V tejto kapitole sú zhrnuté základné informácie o reči a jej technických parametroch. Pre rozpoznanie reči je nutné najprv daný rečový signál istým spôsobom spracovať. Nasledujúce riadky sa venujú analýze audiosignálu pre danú úlohu rozpoznania príkazov.

2.1 Produkcia reči

Keď vydávame hlas, vzduch prúdi z pľúc najprv cez hlasivkové ústrojenstvo a potom prechádza cez hrtan a ústnu dutinu. V závislosti na artikulácii a pôvode budenia môžeme signál reči rozdeliť na tri možné kategórie [1] :

- Hlasové budenie

  Hlasivky sú zavreté. Tlak vzduchu ich núti periodicky sa otvárať a tak generujú periodický sled impulzov. Frekvencia týchto signálov väčšinou leží v istom rozmedzí. Záleží od pohlavia, veku človeka a iných podmienok.

- Nehlasové budenie

  Hlasivky sú otvorené a vzduch prúdi priamo do hrtanu a úst. Dôsledkom toho vzniká turbulentné prúdenie, ktoré vytvára rušivý signál (chrčanie).

- Prechodové (impulzné) budenie

  Uzavretie v hrtane alebo v ústnej dutine spôsobí zvýšenie tlaku vzduchu. Po následnom uvoľnení priechodu tlak prudko poklesne. Výsledkom takéhoto budenia sú napríklad spoluhlásky („p, k , c“...).

V klasickej reči sa tieto tri druhy budenia vyskytujú v rôznych kombináciách. Farba, tón hlasu a artikulácia úzko súvisia s tvarom hlasového traktu (hrtan, jazyk, pery a zuby). Zmenou tvaru tejto cesty sa mení tvar spektra rečového signálu.

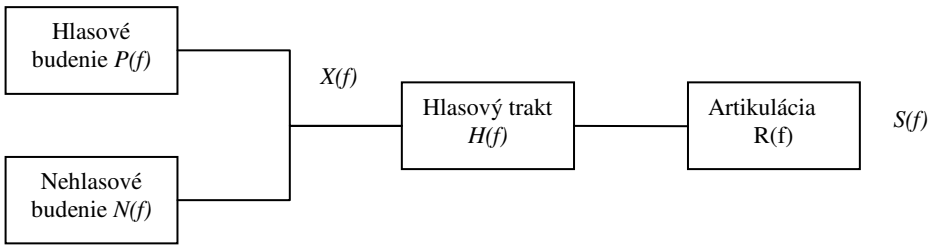
2.2 Technická charakteristika reči

Zdrojom energie pre produkciu reči sú naše pľúca. Moduláciu energie spôsobuje hrtan a na hlasivkách závisí budenie, ktoré je rozoberané vyššie. Výsledný signál *s(t)* je v konečnej fáze daný konvolúciou viacerých vstupov [2] :

$$s(t) = x(t) * h(t) * r(t), \tag{2.1}$$

kde *x(t)* je signál hlasového a nehlasového budenia, *h(t)* je signál hlasového traktu a *r(t)* je artikulačný signál. Konvolúcia je matematická operácia, ktorá kombinuje funkcie a v istom zmysle reprezentuje ich prekrývanie. Hlbšie sa konvolúcií venuje [8] .

Keďže zdroj zvuku a hlasový trakt sú relatívne nezávislé, môžeme ich modelovať oddelene. Na Obr. 2.1 je znázornený model hlasového ústrojenstva.



Obr. 2.1: Model produkcie reči (podľa [1])

Hlasové budenie je modelované ako pulzový generátor, ktorý generuje reťazec impulzov. V kmitočtovej oblasti je vyjadrené funkciou *P(f)*. Nehlasové budenie sa modeluje ako generátor šumu so spektrom *N(f)*. Spojenie týchto spektier vyvolá zmiešanie impulzov so šumom. Výstup oboch generátorov prechádza do vrchnej časti hlasového traktu. Zmenou akustickej trubice sa produkuje signál s prechodovou funkciou *H(f)*. Nakoniec je signál upravený pohybom pier a artikuláciou, modelovanými ako *R(f)*. V kmitočtovej oblasti je konvolúcia vyjadrená súčinom. Spektrum signálu reči *S(f)* je teda dané rovnicou [1] :

$$S(f) = \big(P(f) + N(f)\big).H(f).R(f) = X(f).H(f).R(f) \qquad (2.2)$$

2.2.1 Šírka pásma

Vo všeobecnosti sa pod šírkou pásma signálu reči rozumejú 4 kHz. V skutočnosti dosahujú napr. sykavky oveľa väčšiu frekvenčnú hladinu. Z použitia analógového telefónu však vieme, že v šírke pásma 4 kHz sú obsiahnuté všetky informácie, ktoré potrebujeme pre porozumenie ľudského hlasu.

2.2.2 Základný tón

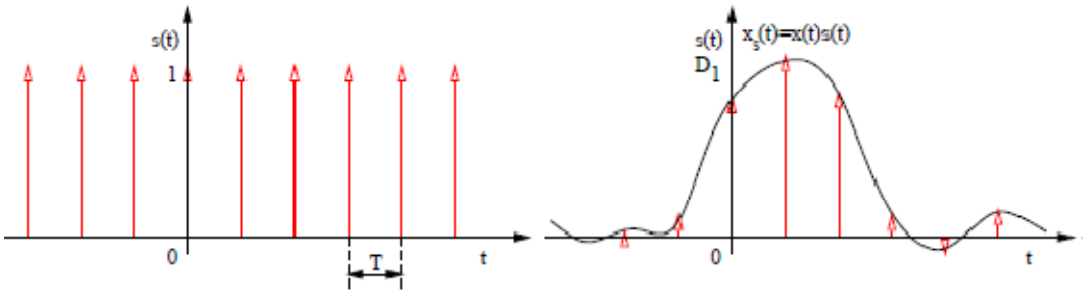
Ako už bolo spomínané vyššie, hlasové budenie vyvoláva sled impulzov, tzv. *základný tón reči*. Táto fundamentálna frekvencia môže nadobúdať hodnôt od 50 Hz (muži) až do 400 Hz (deti). Kolísanie tónu u jedného hovoriaceho môže byť až v pomere 2:1. Hlasové budenie sa používa, keď vyslovujeme samohlásky. Pri sykavkách používame nehlasové budenie. V týchto prípadoch základný tón nedokážeme zistiť.

2.3 Základy spracovania reči

Automatické spracovanie reči umožňuje hlasovú komunikáciu medzi ľuďmi navzájom (kódovanie) alebo medzi človekom a strojom.[2]

2.3.1 Vzorkovanie signálu

Ak chceme so signálom reči pracovať, musíme ho previesť jeho spojitú (analógovú) štruktúru na diskrétnu (číslicovú). Typická schéma spracovania signálu je na Obr. 2.2. Veľmi často sa spätný D/A prevod neuskutočňuje, pretože si vystačíme s číslicovou informáciou, čo sa využíva aj v rozpoznávaní reči.



Obr. 2.2: Násobenie periodickým sledom Diracových impulzov (prevzaté z [2])

Vzorkovaný signál dostaneme tak, že pôvodný vynásobíme istými periodickými hodnotami. Väčšinou to býva sled Diracových impulzov (nekonečná výška, nulová šírka a plocha o veľkosti 1). Opäť dostaneme sled Diracových impulzov, ktoré sa budú opakovať s periódou *T*, ale veľkosť ich plochy bude nadobúdať hodnoty pôvodného signálu v bodoch *nT*.

T teda vyjadruje veľkosť periódy, prevrátená hodnota periódy je vzorkovacia frekvencia F_s. Jednoducho povedané, vzorkovacia frekvencia udáva množstvo vzoriek, ktoré sú nahustené v danej časovej jednotke.

$$F_s = \frac{1}{T} \tag{2.3}$$

Viac je o vzorkovaní signálu pojednávané v [2] .

2.3.2 Signál vo frekvenčnej oblasti – spektrum

Na začiatku spracovania je signál v spojitom čase definovaný všade od -∞ do +∞. Pre reprezentáciu signálu vo frekvenčnej oblasti používame Fourierovú transformáciu [2] :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt, \tag{2.4}$$

kde funkcii *X(f)* hovoríme spektrálna funkcia, alebo skrátene spektrum. Sú tu však isté obmedzenia. Spektrálna funkcia sa nedá spočítať kvôli nekonečnu v rovnici (2.4). V praxi ju preto dokážeme iba odhadnúť pomocou diskrétnej Fourierovej transformácie (DFT).

Pre vzorkovaný signál odhadujeme spektrum rovnicou:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{nk}{N}} \quad pre\ k \in \langle 0, N-1 \rangle \tag{2.5}$$

kde *N* je počet vzoriek vo zvukovej stope. Výsledkom je periodicky sa opakujúce spektrum s periódou *N* vzoriek.

2.4 Predspracovanie reči – pre-processing

Úlohou tzv. *parametrizácie* je vyjadriť rečový signál obmedzeným počtom hodnôt. Znamená to nájsť také charakteristické črty daného signálu, aby sme mohli ďalej pokračovať v post-processingu a rozpoznávať reč.

Parametre delíme na [3] :

- skalárne – vyjadrené jedným číslom na celý rečový úsek
- vektorové – vyjadrené vektorom na rečový úsek. Ak je viac rečových úsekov, hodnoty sa radia do matíc.

2.4.1 Ustrednenie

V signále sa nachádza (mimo dát, ktoré nesú informáciu o reči) jednosmerná zložka, tzv. *dc-offset.* Táto časť signálu môže byť pre ďalšie spracovanie rušivá. Odstraňujeme ju teda odčítaním strednej hodnoty:

$$s'[n] = s[n] - \mu_s \tag{2.6}$$

Strednú hodnotu off-line počítame takto:

$$\mu_s = \frac{1}{N}\sum_{n=1}^{N} s[n] \tag{2.7}$$

2.4.2 Prevzorkovanie (resampling)

Mnohokrát berieme do úvahy dáta, ktoré majú veľkú vzorkovaciu frekvenciu. Pri rozpoznávaní reči je nám tento „príliš“ kvalitný signál prebytočný, pretože spracovanie signálu značne spomaľuje. Navyše vysoké frekvencie majú v sebe okrem audio informácií veľa šumu. Pri telefónoch sa používa napr. vzorkovacia frekvencia 8000 Hz. Aby sme zmenšili množstvo dát, prevzorkujeme rečovú stopu na nižšiu frekvenciu.

2.4.3 Segmentácia

Rečový signál považujeme za náhodný. Pre odhadovanie parametrov, ktoré potrebujeme pri spracovaní, by mal byť stacionárny (časovo stály). Túto podmienku však všeobecne nemožno dodržať, preto audiosignál delíme na kratšie úseky – rámce. V týchto menších segmentoch už stacionárny je.

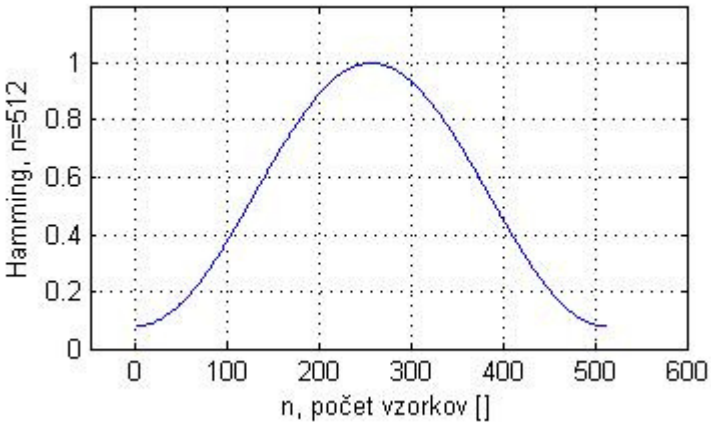
Najčastejšie sa používajú tri základné parametre rámcov:

- **Dĺžka rámcu**
  Zvyčajne sa volí dĺžka, ktorá rešpektuje zotrvačnosť hlasového ústrojenstva (20-25ms, teda 160-200 vzoriek pre F_S = 8000Hz).
- **Prekrytie rámcov**
  Ak zvolíme malé, získame tým rýchly časový posun v signále a tým pádom i malé nároky na procesor. Parametre sa však môžu medzi rámcami značne meniť. Ak zvolíme veľké prekrytie, zaistíme pomalý časový posun a veľké nároky na procesor. Kompromisom je typické prekrytie 10 ms.

2.4.4 Hammingovo okno

Pri vykrojení rámca je použité okno – tzv. okienková funkcia. Z rôznych typov sa najčastejšie používa Hammingovo okno, ktoré síce tlmí signál na okrajoch, ale zachováva spektrum v takmer nezmenenom stave [2] .

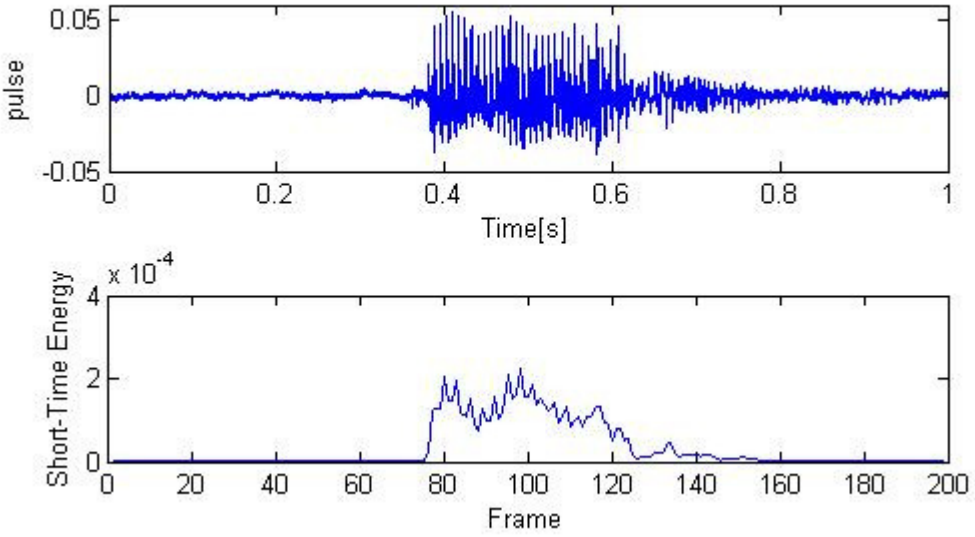
$$w[n] = \begin{cases} 0{,}54 - 0{,}46\cos\dfrac{2\pi n}{l_{ram}-1} & \text{pre } 0 \le n \le l_{ram}-1 \\ 0 & \text{inde} \end{cases} \tag{2.8}$$



Obr. 2.3: Hammingovo okno

2.4.5 Stredná krátkodobá energia

V rozpoznávaní reči je dôležité určiť, kedy slovo začína a kedy končí. Ako detektor hlasovej aktivity slúži stredná krátkodobá energia. Počíta sa pre každý rámec a jej priebeh je vidieť z Obr. 2.4. Pre izolované príkazy väčšinou platí, že pred a po slove sú tiché úseky. To znamená, že tam kde sa vyskytuje slovo, dosahuje energia väčšie hodnoty. Najväčšiu energiu majú samohlásky. Spoluhlásky sa vyznačujú menšou energiou, preto okrem výpočtu strednej krátkodobej energie býva prítomný aj výpočet priechodu cez nulu (tzv. Zero Crossing Rate).



Obr. 2.4: Stredná krátkodobá energia slova „Stůj“

2.4.6 Spektrum

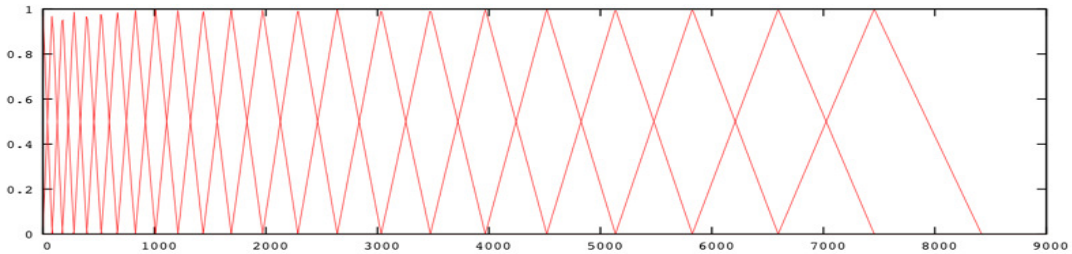
Z upraveného signálu Hammingovým oknom chceme počítať spektrum, pretože signál v časovej ose sa z teórie rečovej analýzy chová náhodne. Spektrum teda frekvenčne analyzuje rečový signál pomocou spektrálnej hustoty výkonu.

Ako som už vyššie uvádzal, spektrum odhadujeme pomocou DFT. Pre ďalšie spracovanie je dôležité iba výkonové spektrum, teda druhá mocnina komplexného spektra – $|X(k)|^2$.

Ľudské ucho nepočuje rozloženie frekvencií lineárne. Znamená to napr., že keď počúvame nižšie frekvencie dokážeme rozoznať rozdiel 50Hz, ale pri vyšších frekvenciách túto zmenu nevieme evidovať, pretože nám frekvencie splývajú. Aby sme sa vyrovnali s touto skutočnosťou, zavádza sa tzv. *Melova škála frekvencií.* Prepočet mel-frekvencie f_{mel} z frekvencie *f* vyjadruje vzťah [1] :

$$f_{mel}(f) = 2595.\log\left(1 + \frac{f}{700Hz}\right) \tag{2.9}$$

Aby sme priblížili rozpoznávaču reči ľudské počutie, musíme preložiť vypočítanú spektrálnu hustotu výkonu istými nelineárne rozloženými filtrami. Tieto bývajú väčšinou trojuholníkového tvaru (tzv. *triangle-shaped*) a najčastejšie je volený počet 22. Znamená to, že po aplikovaní melových filtrov na časový úsek spektra získavame 22 koeficientov, ktorým hovoríme melove spektrálne koeficienty m_k (*mel spectral coefficients*).



Obr. 2.5: Rozloženie filtrov v Melovej škále (prebraté z [10])

Lineárne rozloženie frekvencie v Melovej škále spôsobuje nelineárne rozloženie v štandardnej frekvenčnej ose.

2.4.7 Kepstrum

Pri spracovaní signálu nám do výpočtu spektra vchádzajú dve spektrá. Na jednej strane je to signál budenia *X(f)* a na druhej strane signál hlasového artikulačného traktu *H(f)*. Pre rozpoznanie reči je dôležité, aby sme tieto dva moduly od seba oddelili, pretože pracujeme len s *H(f)*. Používame kepstrálnu analýzu.

Kepstrum vychádza z inverznej Fourierovej transformácie. Premieňa signál z frekvenčnej oblasti znovu späť do závislosti na čase. Aby sme mohli dve spektrá *X(f)* a *H(f)* oddeliť, budeme musieť zlogaritmovať celé výsledné spektrum. Rovnica 2.10 vychádza z výkonových spektier, pre ktoré platí rovnaké pravidlo logaritmovania.

$$\begin{aligned}\log(|S(f)|^2) &= \log(|X(f)|^2.|H(f)|^2) \\ &= \log(|X(f)|^2) + \log(|H(f)|^2)\end{aligned} \tag{2.10}$$

Nakoniec nám zostáva použiť spätnú (inverznú) Fourierovu transformáciu. V prípade, že ide o Melovu škálu, ktorá je symetrická, čiže ide o náš prípad, môžeme inverznú Fourierovu transformáciu nahradiť jednoduchšou kosínovou transformáciou – DCT [2] :

$$c_{mf}(n) = \sum_{i=1}^{K} \log m_k \cos\left[n(k-0{,}5)\frac{\pi}{K}\right], n = 1, \dots, \kappa \qquad (2.11)$$

Výsledkom vzťahu sú *Mel-frekvenčné kepstrálne koeficienty (MFCC)*, ktoré posielame do rozpoznávača. Tieto koeficienty majú špecifické vlastnosti a črty, podľa ktorých sme schopní dané slovo rozpoznať. Obvykle sa volí $\kappa = 13$.

Použitím spätnej Fourierovej transformácie teoreticky aplikujeme dolnopriepustný filter. A zamedzíme tak vplyv spektra budenia $|X(f)|^2$. Po aplikovaní DCT dostaneme každých 10ms trinásť koeficientov.

Kapitola 3

Rozpoznávanie reči

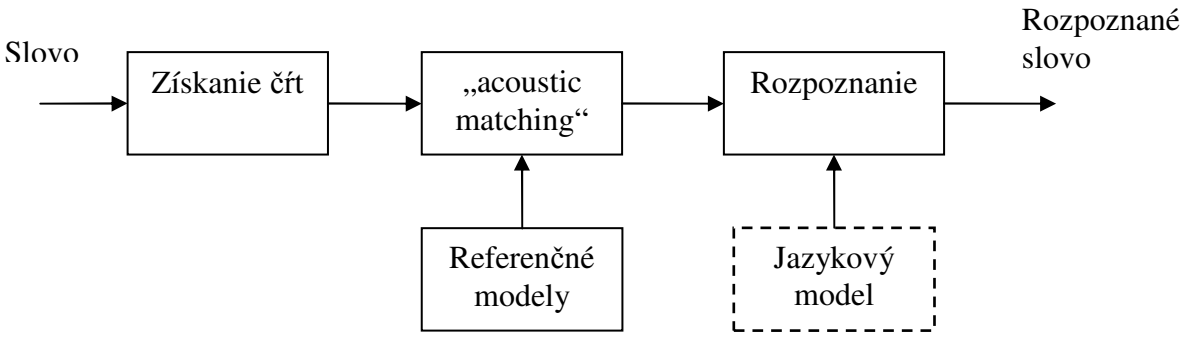
Technológie rozpoznávania reči umožňujú počítačom vybaveným zdrojom zvukového vstupu, ako je mikrofón, interpretovať ľudskú reč, na príklad pre účel zaznamenávania alebo pre alternatívnu metódu interakcie s počítačom a prístrojmi.[11]

Úlohou rozpoznávania reči je vhodnou metódou určiť, čo bolo povedané. To, čo bude rozpoznávané a akým spôsobom, nám určuje jednotlivé kategórie rozpoznávania. Podľa zložitosti klasifikujeme rozpoznávanie reči na jednotlivé kategórie:

- **Rozpoznávanie izolovaných slov**
  Vyznačuje sa najjednoduchším algoritmom a pomerne malou knižnicou slov.
- **Rozpoznávanie spojených slov**
  Majú väčšinou obmedzený slovník. Využitie má pri diktovaní telefónnych čísel, alebo zadávaní čísla kreditnej karty.
- **Rozpoznávanie plynulej reči**
  Má obrovský slovník, vyžaduje si informácie o akustike a o jazykových modeloch.

3.1 Štruktúra rozpoznávača

Funkcia všeobecného rozpoznávača je jasná z obrázku. Do algoritmu je zavedený vstup, ktorý je príkazom (testované slovo). Tento daný príkaz prejde potrebnými úpravami, aby sme z neho dostali vektor charakteristických čŕt. Zbavíme sa tak nadbytočných údajov (dc off-set, základná frekvencia) a obmedzíme veľkosť dát. Jeho vlastnosti sa potom porovnávajú s vopred nadefinovanou knižnicou slov. Ide o tzv. *acoustic matching*. Výsledky tohto triedenia sa potom spracovávajú, aby program dokázal správne priradiť slovo do danej kategórie. Ďalej je už len na rozpoznávači a jeho algoritme, ako prinesie rozpoznané slovo do výsledného výstupu.



Obr. 3.1: Všeobecná štruktúra rozpoznávača reči (podľa [2])

3.2 Rozpoznávanie izolovaných slov - príkazov

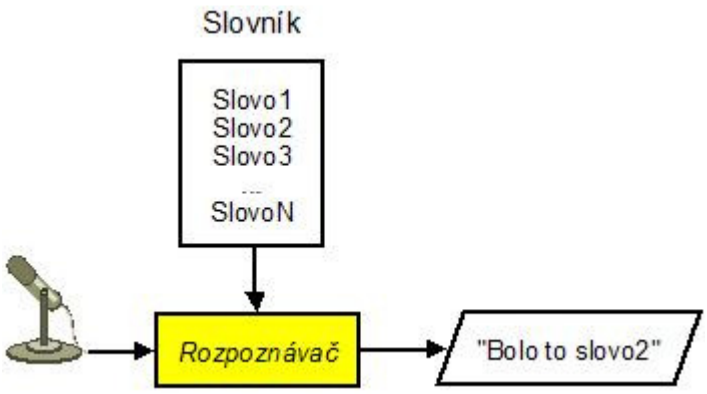
Rozpoznávanie izolovaných slov je najjednoduchším prípadom rozpoznávania reči v audiosignále. Jednoduchosť tohto druhu rozpoznávania spočíva v niekoľkých faktoch:

- Nie je náročný na vstupnú databázu slov. Knižnica môže obsahovať iba pár slov, čím je však slov viac, tým je referenčný model lepšie natrénovaný.
- Do algoritmu vstupuje menej údajov. Väčšinou je navrhnutý tak, aby nebol závislý na rečníkovi. Do výpočtov vstupujú iba dáta charakterizujúce artikulačný trakt.
- Kontroluje sa jedno testované slovo s referenciou. Slovo je určené jasnými hranicami. Test všeobecne neobsahuje komplikované funkcie ako pri iných rozpoznávaniach (syntaktická, sémantická analýza a i.).

Pri rozpoznávaní izolovaných slov si je potrebné uvedomiť, kedy slovo začína. Pre učenie tejto informácie sa využívajú dva prístupy:

- **Push-to-Talk**
  Testované slovo sa nahrá po stlačení tlačidla a ďalej prechádza procesmi rozpoznávača.

- **Detekcia hlasovej aktivity (Voice Activity Detection)**
  Nahrávanie beží po celý čas chodu programu. Ak hodnoty vystúpia nad danú hranicu, prebehne rozpoznávanie. Tento prístup je nevhodný v hlučnejšom prostredí (napr. v aute).



Obr. 3.2: Úloha rozpoznávača izolovaných príkazov

Kapitola 4

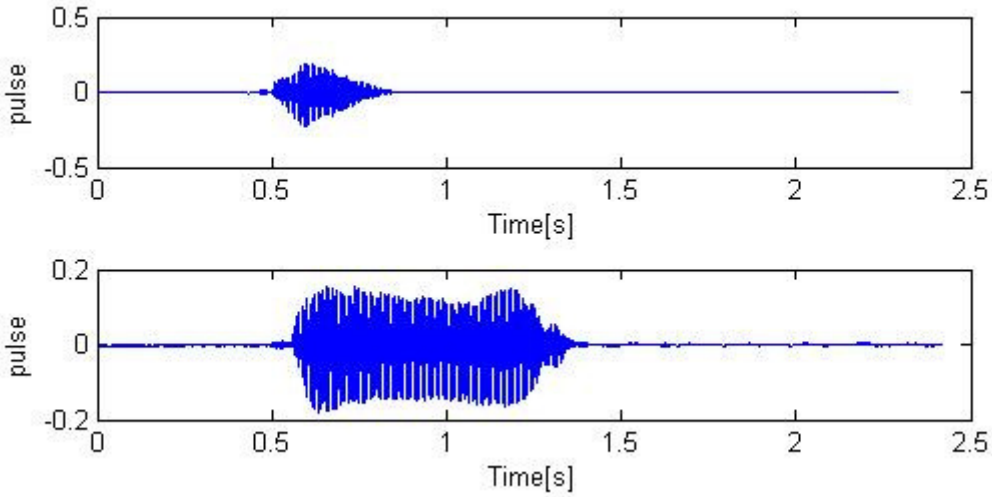
Spôsoby rozpoznávania hlasových príkazov

V tejto kapitole sú rozobrané tri základné typy rozpoznávania príkazov v audiosignále a je tu naznačený princíp metódy, ktorú som vybral pre implementáciu. Ide o stručný prehľad a základné informácie, preto pre detailnejšie informácie o daných spôsoboch nájdete v priloženej literatúre.

4.1 Dynamické skreslenie časovej osi (DTW)

Keď vyslovujeme slová, nikdy nepovieme jedno slovo úplne rovnako. Raz vyslovíme slovo kratšie, inokedy dlhšie, následne môže byť prítomný väčší šum, prípadne sa môžu zmeniť zvuky z prostredia. Preto nikdy nezískame rovnaké dáta.

Metóda rozpoznávania pomocou dynamického skreslenia časovej osi (Dynamic Time Warping - DTW) rieši hlavne prípad časovej variability slov. Znamená to, že dynamicky mení porovnávanie slov v čase. Napr. keď niekto povie „Čau“ a niekto iný „Čáááu“, metóda DTW sa dynamický snaží skrátiť, alebo predĺžiť písmeno „a“ v slove.



Obr. 4.1: Časový rozdiel medzi „Čau“ a „Čáááu“

Ako bolo už spomínané vyššie, pre každý rámec sa počíta vektor charakteristických parametrov, ktoré sú reprezentované napr. mel-frekvenčnými kepstrálnymi koeficientmi. Podľa dĺžky rámca a časového posunu sa počíta asi každých 10ms. V takom prípade dostávame vektor s trinástimi hodnotami každých 10ms. Koľko bude vektorov v danom prípade záleží na rýchlosti reči osoby. Dôležité je si uvedomiť, že sa nevyhodnocujú vektory jednotlivo, ale vyhodnocujú sa a prechádzajú algoritmom sekvencie vektorov.

4.1.1 Úloha DTW algoritmu

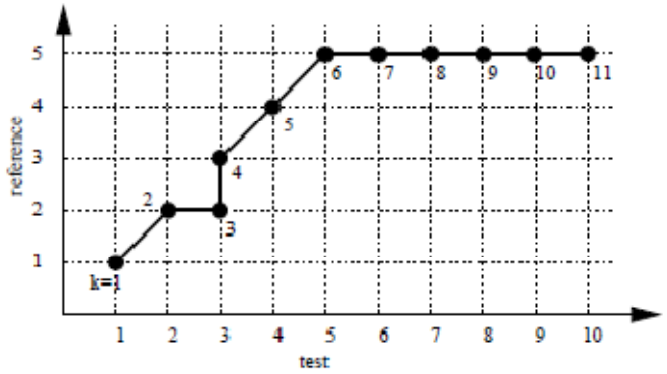
Ak si zoberieme referenčnú sekvenciu vektorov o dĺžke *R*, potrebujeme ju porovnať s testovanou sekvenciou vektorov o dĺžke *T*. Všeobecne je dĺžka sekvencie R a dĺžka sekvencie T rôzna. Obe však obsahujú vektory o konštantnej dĺžke (do vektorov posielame rovnaký počet MFCC, viď kap. 2).

$$R = [r(1), \dots, r(R)] \quad vs. \quad O = [o(1), \dots, o(T)] \tag{4.1}$$

Ako teda porovnáme jednotlivé vektory? Najjednoduchším a najefektívnejším spôsobom určenia toho, ako sa dva vektory na seba podobajú, je výpočet ich Euklidovskej vzdialenosti[2] :

$$d(o,r) = \sqrt{\sum_{k=1}^{P} [o(k) - r(k)]^2} \tag{4.2}$$

Úlohou metódy DTW je nájsť najkratšiu cestu medzi jednotlivými sekvenciami vektorov. Jedna z možných ciest je znázornená na Obr. 4.2. Porovnávame dve sekvencie rôznej dĺžky. Počet krokov je K.



Obr. 4.2: Príklad cesty DTW (prevzaté z [2])

Pre túto danú cestu sa dá spočítať celková vzdialenosť medzi sekvenciami ako suma vzdialeností medzi jednotlivými vektormi.

$$D(R,T,K) = \frac{\sum_{k=1}^{K} d_i\big(t(k), r(k)\big) \,.\, w_c(k)}{N_c}, \tag{4.3}$$

kde $d_i\big(t(k), r(k)\big)$ je vzdialenosť dvoch vektorov, w_c je váhový koeficient závislý na kroku a N_c je normalizačný faktor.

Vzdialenosť sekvencií T a K je daná minimom zo všetkých možných ciest[1] :

$$D_{min} = \min D(R,T,K) \tag{4.4}$$

Aby sme našli optimálnu cestu, našťastie nie je potrebné počítať všetky možné kombinácie. Dôležité je ale uvedomiť si niekoľko skutočností[2] :

- Je nutné vymedziť počiatočné a koncové body. Tzn.:

$$\begin{aligned} r(1) = 1, t(1) = 1 \ &\rightarrow začiatok \\ r(K) = R, t(K) = T \ &\rightarrow koniec \end{aligned} \tag{4.5}$$

- Potrebné je nadefinovať i lokálnu súvislosť a strmosť:

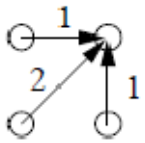
$$\begin{aligned} 0 \leq r(k) - r(k-1) \leq \tilde{R} \\ 0 \leq t(k) - t(k-1) \leq \tilde{T} \end{aligned} \tag{4.6}$$

  Ak zvolíme $\tilde{R} = \tilde{T} = 1$, znamená to, že každý vektor sa musí vziať aspoň jeden krát. Ak *r(k-1)=r(k)*, v tom prípade sa vektor opakuje.

- Aby sme dodržali plynulý chod algoritmu, je dobré vymedziť globálnu oblasť pre cestu DTW. Teda aby sme neporovnávali iba niekoľko vybraných vektorov, ale aby sa dynamicky prechádzalo cez referenčnú a testovaciu frekvenciu.

Váhová funkcia záleží na lokálnom posune cesty, teda na tom ako sa z bodu *k-1* dostaneme do bodu *k*. Najjednoduchšou je symetrická váhová funkcia:

$$w = [t(k) - t(k-1)] + [r(k) - r(k-1)] \tag{4.7}$$



Obr. 4.3: Váha troch možných smerov (prevzaté z [2])

Normalizačný faktor pre túto váhovú funkciu má potom tvar:

$$\begin{aligned} N = \sum_{k=1}^{K} [t(k) - t(k-1) + r(k) - r(k-1)] \\ = t(K) - t(0) + r(K) - R(0) = T + R \end{aligned} \tag{4.8}$$

4.1.2 Klasifikácia

Vo všeobecnosti sa používa viac druhov zaradenia slov, záleží na počte referencií v triede[2] :

- Ak je každá trieda reprezentovaná jednou referenciou, potom:

$$\omega_r = \arg\min D(T, R_r) \quad pre\ r = 1, \dots, N \tag{4.9}$$

- Ak je trieda reprezentovaná viacerými referenciami, je možné použiť dve metódy:
  a) **1-NN – Najbližší sused**
     Vyberie sa trieda, v ktorej referencia má najkratšiu vzdialenosť s testom.

b) **k-NN – k Najbližších susedov**
Vyberie sa k susedov, spriemeruje sa ich vzdialenosť k testu a z výslednej vzdialenosti sa určuje trieda, do ktorej patrí.

4.2 Skryté Markovove modely (HMM)

Druhou metódou rozpoznávania reči je metóda skrytých Markovovych modelov (angl. Hidden Markov Models - HMM). Tento spôsob využíva nadefinované štatistické modely.

4.2.1 Štatistické postupy v HMM

Na rozdiel od DTW, kde sa určovali vzdialenosti medzi vektormi rôznej dĺžky, využíva HMM stochastické modely určované pre jednotlivé referenčné triedy. Tieto potom využíva pri určovaní toho, či je hovorené slovo vhodné pre danú triedu.

Pre každé slovo ω_v v databáze máme model Λ_v, ktorý je definovaný istými parametrami λ_v. Pre sekvenciu vektorov $\tilde{X}$ je možné určiť hustotu pravdepodobnosti, že výskyt $\tilde{X}$ súvisí práve s modelom Λ_v. Túto funkciu, ktorá závisí na modely Λ_v značíme $p(\tilde{X}|\Lambda_v)$.

Pri rozpoznávaní nás však zaujíma, do ktorej triedy ω_v sa s najväčšou pravdepodobnosťou hodí sled vektorov $\tilde{X}$. Túto tzv. *posteriornú pravdepodobnosť* rieši teória štatistického rozpoznávania. Je daná Bayesovym vzťahom[2] :

$$P(\omega_v|\tilde{X}) = \frac{p(\tilde{X}|\omega_v)\,P(\omega_v)}{p(\tilde{X})}, \tag{4.10}$$

kde:

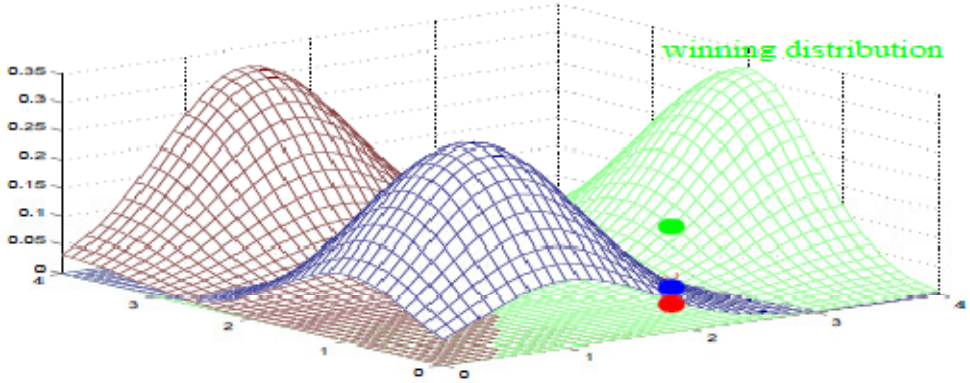
- $P(\omega_v|\tilde{X})$ je posterior triedy ω_v, ak poznáme radu vektorov $\tilde{X}$.
- $p(\tilde{X}|\omega_v)$ je likelihood (vierohodnosť) $\tilde{X}$, ak poznáme ω_v.
- $P(\omega_v)$ je prior triedy ω_v.
- $p(\tilde{X})$ je evidencia, alebo normalizačná funkcia.

Evidenciu $p(\tilde{X})$ nemusíme brať do úvahy, pretože je pre všetky triedy rovnaká. U jednoduchých rozpoznávačov tiež predpokladáme, že priory tried $P(\omega_v)$ sú taktiež rovnaké (napr. pre rozpoznávanie príkazov je pravdepodobnosť povelu „ŠTART“ a „STOP“ zhodná).

Preto pre rozpoznanie slova stačí hľadať maximálny likelihood[2] :

$$\omega_v{}^* = \max_v p(\tilde{X}|\omega_v) \tag{4.11}$$

Toto je ďalšia odlišnosť od DTW rozpoznávača. Zatiaľ čo v DTW hľadáme minimum vzdialenosti dvoch sekvencií vektorov, pri HMM hľadáme maximálnu posteriornú pravdepodobnosť.

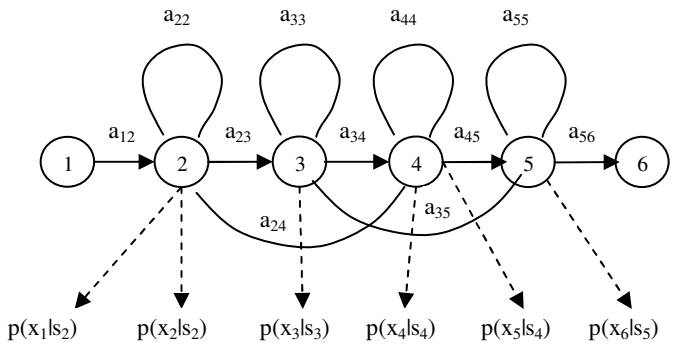


Obr. 4.4: Gaussove plochy – ilustrácia modelovania tried (prevzaté z [2])

4.2.2 Podstata HMM

Funkcie hustoty pravdepodobnosti $p(\tilde{X}|\Lambda_v)$ sa počítajú pre každý slovný model Λ_v, aby sa určila príslušnosť vektorov $\tilde{X}$ do triedy ω_v. Sekvencie vektorov $\tilde{X}$ však môžu mať rôznu dĺžku a rôzne spektrá. Preto je potrebný model, ktorý bude riešiť oba problémy. Takýmto modelom je práve HMM.

Štruktúra modelu HMM je znázornená na Obr. 4.5. Stavy 1 a 6 sú konektormi, ktoré používame pre napojenie modelov. Ostatné stavy sú tzv. *vysielacie* (nič nevysielajú, ale počítajú likelihoody, s ktorými by vyslali jednotlivé vektory). Prechodové pravdepodobnosti a_{ij} udávajú pravdepodobnosť preskočenia zo stavu i do stavu j. Pre jeden stav je súčet prechodových pravdepodobností rovný jednej.



Obr. 4.5: Schéma HMM (podľa [2])

Predstavme si, že máme sekvenciu vektorov $\tilde{X} = \{\vec{x}_0, \vec{x}_1, \dots, \vec{x}_{(T+1)}\}$, ktorá patrí do triedy ω_v. V každom čase T, kedy je HMM v stave s_t, model generuje vektor $\vec{x}_t$ s vysielacou pravdepodobnosťou $p(\vec{x}_t|s_t)$. Potom nastáva prechod zo stavu s_t do stavu s_{t+1}, s prechodovou pravdepodobnosťou a_{ij}. Takto dej pokračuje, až kým HMM neprejde cez celú sekvenciu $\tilde{X}$. Ak by sme mali iba sekvenciu vektorov $\tilde{X}$, nevieme v ktorom stave bol generovaný vektor $\vec{x}_t$ (preto názov Skryté Markovove modely).

Ak máme stavy, ktoré generujú $\tilde{X}$, môžeme spočítať hustotu pravdepodobnosti $p(\tilde{X}, \Theta|\Lambda)$, kde Θ je index stavovej sekvencie[1] :

$$p(\tilde{X}, \Theta|\Lambda) = a_{\theta(0)\theta(1)} \prod_{t=1}^{T} a_{\theta(t)\theta(t+1)} . p(\vec{x}_t|s_t) \qquad (4.12)$$

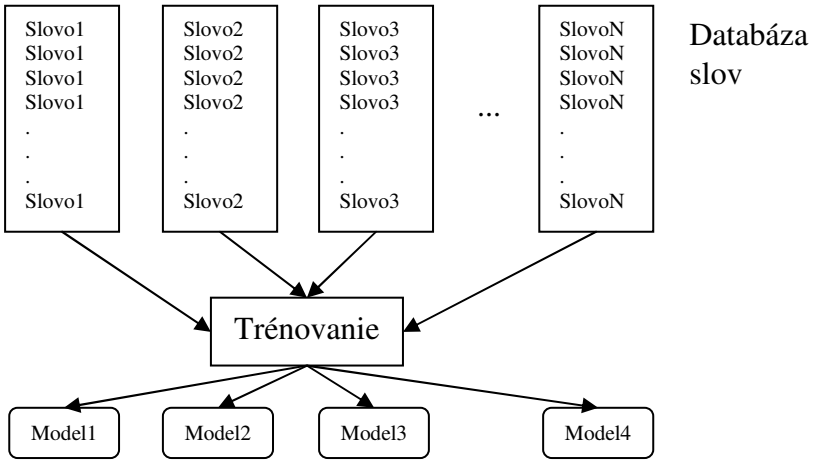
Tento vzťah nám hovorí, že do modelu pustíme vektory po stavovej ceste Θ a násobíme všetky pravdepodobnosti a likelihoody, ktoré sa tu priechodom vyskytnú.

Pre rozpoznávanie pomocou HMM však potrebujeme spočítať $p(\tilde{X}|\Lambda)$. Aby sa dosiahla táto hodnota, museli by sme spočítať rovnicu (4.12) cez všetky možné sekvencie stavov Θ o dĺžke T+2 a sčítať ich. Táto možnosť by vyžadovala veľa výpočtov, preto sa namiesto možných sekvencií Θ volí iba jedna stavová sekvencia, ktorá svojou pravdepodobnostnou hodnotou najviac prispieva do tejto sumy. Môžeme tak učiniť algoritmom, ktorému hovoríme Vertibiho algoritmus:

$$p^*(\tilde{X}|\Lambda) = \max_{\{\Theta\}} p(\tilde{X}, \Theta|\Lambda) \tag{4.13}$$

4.2.3 Trénovanie HMM

Proces trénovania modelov zjednodušene zachycuje Obr. 4.6.



Obr. 4.6: Trénovanie HMM z databázy

4.3 Neurónové siete

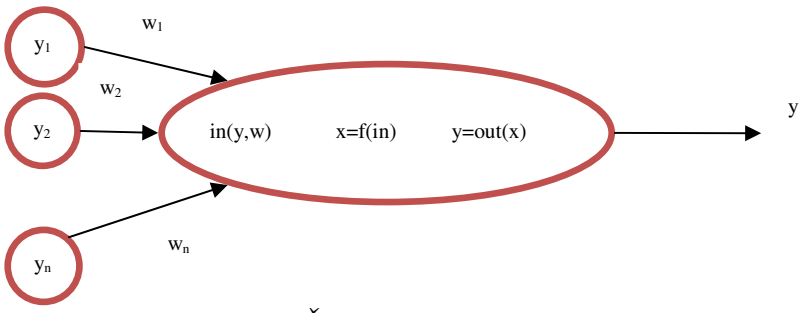
Neurónová sieť (NS) je masívne paralelný procesor, ktorý má schopnosť uchovávania informácií a ich ďalšieho využívania. Napodobňuje ľudský mozog, pretože[7] :

- poznatky sú získavané v procese učenia
- na ukladanie znalostí sa využívajú medzineurónové spojenia (synaptické váhy)

Je očividné, že pre princíp neurónových sietí je prebratý z biologických systémov. Najvýznamnejšou vlastnosťou NS je, že sú relatívne dobrým aproximátorom funkcií popisujúcich daný systém. Môže sa nám stať, že dostaneme systém, ktorý je náročné a takmer nemožné popísať funkciami. Máme však isté vstupy a pre ne charakteristické výstupy. NS sa snaží naučiť sa správať v zhode s daným systémom. Využíva pri tom v procese učenia spomínané dáta (vstupy a výstupy).

Štruktúra NS je vo všeobecnosti veľmi zložitá a ťažko sa analyzuje. Medzi najjednoduchšie patrí vrstvová, kde rozlišujeme 3 vrstvy (vstupnú, skrytú a výstupnú). Signály sa môžu v sieti šíriť len jedným smerom (dopredné siete) alebo sa môžu aj vracať,

čiže je možné šírenie i v opačnom smere (rekurentné siete). Vrstvy sa skladajú zo základných stavebných jednotiek NS – neurónov. Štruktúra neurónu je daná na Obr. 4.7.

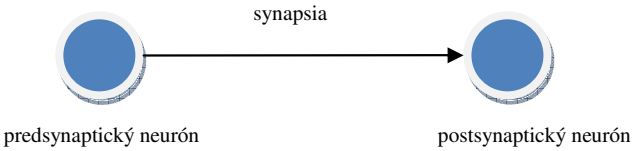


Obr. 4.7: Štruktúra neurónu (podľa [6])

Hlavnou úlohou neurónu je určiť jednu výstupnú hodnotu nelineárnej funkcie, ktorej premenná je získaná zhromažďovaním vstupov. Vstupmi do neurónu sú tzv. *dendrity* $y_1,..., y_n$, ktoré sú násobené váhami $w_1,..., w_n$. Váhy priraďujú dôležitosť informácii, ktorú posielame ďalej do NS a v štruktúre sa zobrazuje tzv. *synapsiou* – prepojením medzi neurónmi. Ďalej sa prejavujú dve funkcie – aktivačná *f* a výstupná *out*.

Podľa toho, čí je neurón pred synapsiou alebo za ňou, môžeme rozdeliť neuróny na dve skupiny:

- predsynaptické (zdrojové)
- postsynaptické (cieľové)



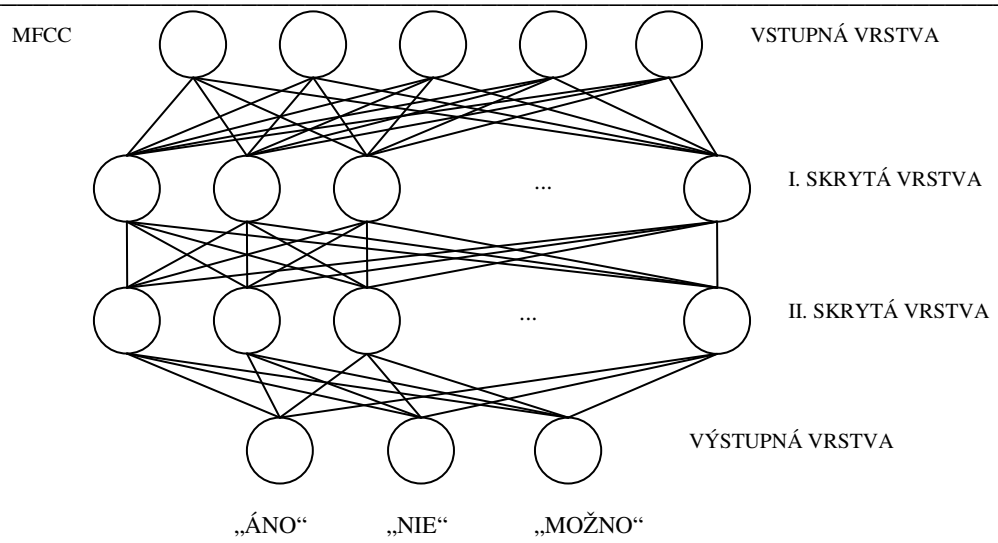
Obr. 4.8: Označovanie neurónov (podľa [6])

Rozlišujeme dve fázy činnosti neurónovej siete:

- fáza učenia – neurónovú sieť trénujeme. Je to prvá fáza vývoja, kedy sa upravujú váhy spojení medzi jednotlivými neurónmi. Sieť nadobúda vedomosti o prezentovaných dátach
- fáza života – sieť využíva poznatky, ktoré nadobudla v procese učenia. Na vstupy priraďuje výstupy.

4.3.1 Klasifikácia

Pre rozpoznávanie reči je možné použiť viacero topológií NS. Budeme preto vychádzať zo štandardne používanej doprednej NS (Obr. 4.9) [5] . Je zložená zo štyroch vrstiev. Vstupná vrstva obsahuje niekoľko neurónov. Ich počet zodpovedá parametrom (napr. MFCC), ktoré sme vybrali pre klasifikáciu. V prvej a druhej skrytej vrstve sa nachádza vyšší počet neurónov. Sú vzájomne prepojené. Výstupná vrstva obsahuje toľko neurónov, koľko slov rozpoznávame.



Obr. 4.9: Schéma doprednej NS (podľa [5])

Obr. 4.9 schematicky znázorňuje postup práce neurónovej siete. Vstupné neuróny obsahujú parametre (MFCC - vyjdú z pre-processingu), ktoré sa v skrytých vrstvách násobia váhami a formojú sa tak, aby sa na konci procesu dalo povedať, o ktoré slovo ide.

4.3.2 Trénovanie

Priebeh učenia NS je nasledovný:

Nadefinuje sa tzv. *trénovacia množina*, ktorá pozostáva z príznakov a cieľov. Príznaky sú koeficienty charakterizujúce dané slovo. Ako bolo už spomínané v kapitole 2, slovo môžeme charakterizovať koeficientmi MFCC. Z celkového počtu (napr. 13), vyberieme tie, ktoré sú smerodajné pre dané slovo (napr. 5). NS niekoľko krát prebehne knižnicu slov s príznakmi, pričom sa menia a prispôsobujú váhy tak, aby boli presne určené ciele dané príznaky. V priebehu života sa už váhy nemenia.

4.4 Porovnávanie spektrogramov

Ide o jednoduchú metódu, ktorá vychádza z grafickej podobnosti spektrálnej hustoty výkonu rovnakých slov. Zo zdigitalizovaného slova upraveného predspracovaním sa spočíta spektrogram. Spektrogramy rovnakých slov sa vyznačujú istou podobnosťou, naopak, ak si zoberieme obrázky spektrogramov rôznych slov, nachádzame isté odlišnosti [9] . Toto je základným princípom porovnávania spektrogramov.

Aby sme mohli spektrogramy užitočne porovnať, rozdelia sa na menšie polia, pre ktoré sa spočíta ich priemer. Ktoré spektrogramy sú si natoľko podobné, že ide o rovnaké slovo, prezradzuje metrika – vzdialenosť jednotlivých priemerov.

Spôsob rozpoznania reči pomocou spektrogramov som sa rozhodol spracovať v systéme, ako jeden z najmenej náročných spôsobov rozpoznávania. Implementácii sa venuje kapitola 5.

Kapitola 5

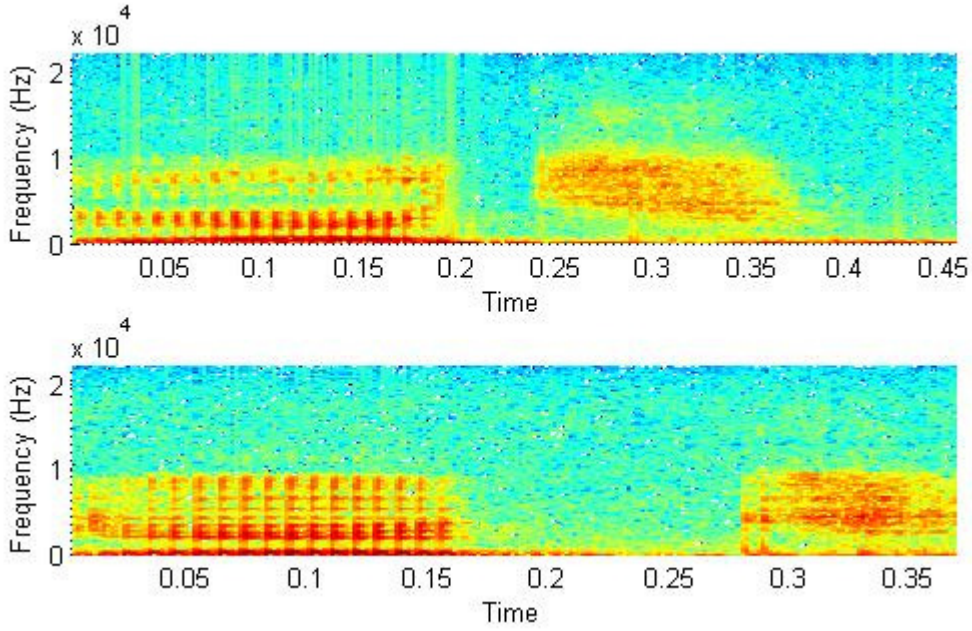
Implementácia algoritmu rozpoznania príkazov pomocou spektrogramov

V tejto kapitole sa dostávam k využitiu získaných poznatkov pre zostavenie vlastného algoritmu rozpoznania hlasových príkazov. Metódu rozpoznávania pomocou spektrogramov som spracoval v prostredí Matlab. Pre spracovanie signálov a tvorenie algoritmu je potrebný Signal Processing Toolbox.

5.1 Princíp metódy

Ako už bolo spomínané v rešeršnej časti práce, audiosignál sa považuje po digitalizácií za náhodný a nestacionárny. Ak chceme rozpoznávať reč, musíme sa s týmito problémami vyrovnať. Je teda nutné nájsť spôsob, ako získame podobné črty charakterizujúce dané signály.

Jeden zo spôsobov je rozpoznávanie pomocou spektrogramov. Spektrogram je grafické zobrazenie časovej zmeny spektrálnej hustoty. Laicky povedané, spektrogram nám ukazuje ako veľmi je v danom čase zastúpená daná frekvencia. V štúdiu spektrogramov rovnakých zvukových záznamov – v našom prípade rovnakých slov – postupne nachádzame istú podobnosť. Na Obr. 5.1 je ukážka spektrogramu pre slovo „JEĎ“. V prvom prípade ide o slovo vyslovené rečníkom 1, druhý prípad ukazuje slovo rečníka 2.



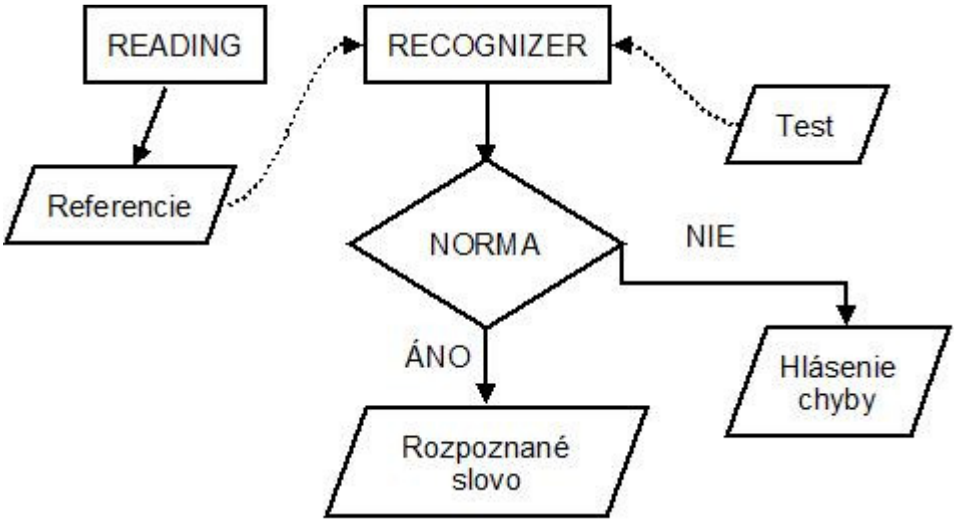
Obr. 5.1: Podobnosť spektrogramov slov od dvoch rozdielnych rečníkov

Z grafiky je vidieť, že hoci bolo slovo vyslovené inými osobami, môžeme sledovať isté spoločné črty. Táto skutočnosť je v našom prípade veľmi významná.

Ako teda využiť dané charakteristiky v rozpoznávaní? V číselnej podstate predstavujú spektrogramy matice, kde so vzrastajúcim indexom riadkov rastie frekvencia a so vzrastajúcim indexom stĺpcov rastie čas, v ktorom sa spektrálna hustota počíta. Aby sme nepracovali s exaktnými a početnými hodnotami, musíme si rozdeliť spektrogram na menšie sektory. Keď chceme nahradiť viac čísel jedným, je nutné zvoliť istú „funkciu náhrady“ (napr. aritmetický priemer). Týmto postupom dostaneme z hromady čísel získanej zo spektrogramu maticu o výrazne menších rozmeroch. Tieto matice už môžeme poslať do rozpoznávača. Jeho štruktúra je rovnaká ako na Obr. 3.1 (ibaže pre jednoduché rozpoznávanie izolovaných slov nepotrebujeme jazykový model).

5.2 Zloženie rozpoznávacieho systému

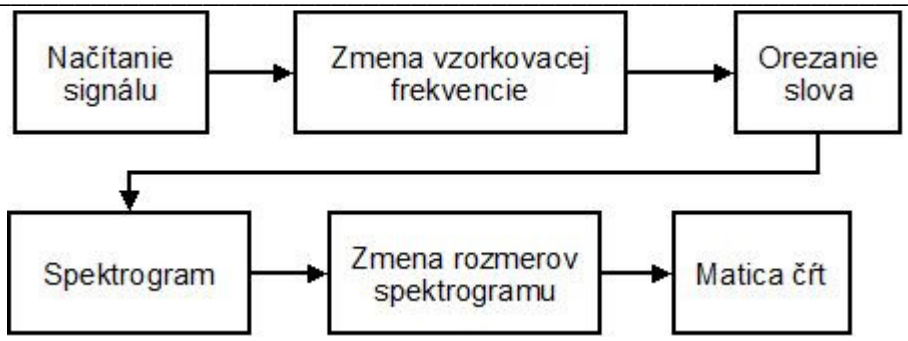
Celý systém pozostáva z dvoch hlavných častí. Prvou časťou je načítanie vstupných referencií – *READING*. Druhou časťou je už priamo rozpoznávač a klasifikátor – *RECOGNIZER*. Dôležité je povedať, že tieto časti sú izolované. Nie sú na sebe závislé. *Reading* načíta dáta, ktoré spracuje a uloží ich do súboru. *Recognizer* si ich odtiaľ zoberie a použije v porovnávaní. Z *recognizera* vychádzajú hodnoty, ktoré ak spĺňajú normu pre danú triedu, uskutoční sa klasifikácia. Ak danú normu nespĺňa, systém hlási chybu. Normy sa počítajú v pomocnom programe *NORMS*.



Obr. 5.2: Vývojový diagram systému pre rozpoznávanie príkazov

5.3 Tvorba vzorov - READING

Skôr než budeme hneď rozpoznávať, je potrebné, aby sme vytvorili databázu referencií, s ktorými sa bude testovaný signál porovnávať. Na Obr. 5.3 je znázornený postup všetkých funkcií ktoré nám do algoritmu vstupujú. Ich popis a príklady budú zahrnuté v tejto podkapitole. V databáze sa bude nachádzať sedem slov: jeď, vpřed, vlevo, vpravo, stůj, stát a stop.



Obr. 5.3: Schéma procesov v programe READING

Načítanie signálu

Proces načítania šikovne rieši integrovaná matlabovská funkcia `wavread`. Zvukový súbor vo formáte WAV uloží do premennej *signal*. Zvuky boli nahrávané so vzorkovacou frekvenciou 44100Hz (CD kvalita).

```
signal = wavread(p);
```

Zmena vzorkovacej frekvencie F_s

Frekvencia impulzov v CD kvalite znamená pre túto úlohu veľmi veľký počet dát za jednotku času. Preto je nutné zmenšiť množstvo informácií. To vykonáva funkcia *resamp*. Obsahuje vstavanú matlabovskú funkciu `resample` a ustrednenie signálu. V spracovaní reči je zvykom pracovať s F_s okolo 8000Hz. Zvolil som teda pätinu vzorkovacej frekvencie CD kvality, čo je presne 8820Hz.

```
4    function [SignalRes] = resamp(Signal)
5
6 -  s = Signal - mean(Signal); % ustrednenie signalu
7 -  SignalRes = resample(s,1,5); % resamplovanie
```

Obr. 5.4: Funkcia resamp

Orezanie slova

V systéme nesie funkcia, ktorá má za úlohu tento úkon, meno *cut*. Pri rozpoznávaní je jedným z najväčších problémov zistiť, kedy je slovo vyslovené. Spôsobov ako sa s týmto vyrovnať je viacero. Rozhodol som sa preto pre využitie strednej krátkodobej energie, spomínanej v kapitole 2. Počíta sa podľa vzťahu:

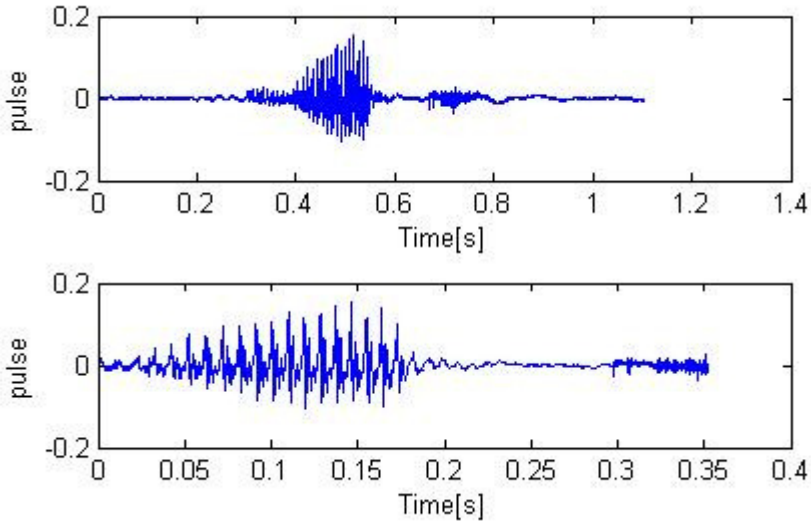
$$E = \frac{1}{l_{ram}} \sum_{n=0}^{l_{ram}-1} x^2[n] \tag{5.1}$$

Energetický pohľad na signál niekedy nestačí, pretože sa obmedzia slabiky s nižšou energiou. Býva zvykom použiť počítadlo priechodu cez nulu (ZCR). V systéme nie je použité, pretože po ustrednení signálu počas šumu nastávalo množstvo priechodov cez nulu. Zachytenie nižších energetických hladín je riešené konštantným posunom. Napr. slová Jeď, Stůj, Stát, Stop majú na začiatku i na konci spoluhlásky s menšou energiou. Preto sa zoberie rámec, ktorého energia presiahne zvolený prah (treshold) a posunie sa začiatok a koniec slova o konštantnú dĺžku. Tento spôsob je nenáročný na výpočet, má však nevýhodu, systém je tým pádom viac závislý na rečníkovi.

Vo funkcii sa využíva rámcovanie, o ktorom bolo hovorené vyššie. Stredná krátkodobá energia sa počíta pre každý rámec o dĺžke 10ms s prekrytím 5ms. Následne je uložená do

energetického vektoru, na ktorý sa aplikujú energetické prahy (tresholdy). Sú nadefinované dva prahy. Nižší prah sa spočíta z maximálnej energie a strednej energie šumu. Vyšší prah je počítaný, aby sa rýchlejšie hľadal nižší.

Funkcia vracia orezaný signál bez šumu, kde sa nachádza hovorené slovo.



Obr. 5.5: Pôvodný signál a orezané slovo „Jeď“

Spektrogram

Gro systému tvorí výpočet spektrálnej hustoty. Na to slúži funkcia `spectrogram` nadefinovaná v Matlabe. Jej volanie:

```
36 -        S = spectrogram(signalRC,win,nol,n,Fs);
```

Do funkcie vstupuje päť parametrov:

- `signalRC` – resamplovaný a orezaný signál
- `win` – Hammingovo okno dĺžky zodpovedajúcej dĺžke DFT
- `nol` – počet prekrývajúcich sa segmentov (s prekrytím 50%)
- `n` – dĺžka DFT
- `Fs` – vzorkovacia frekvencia

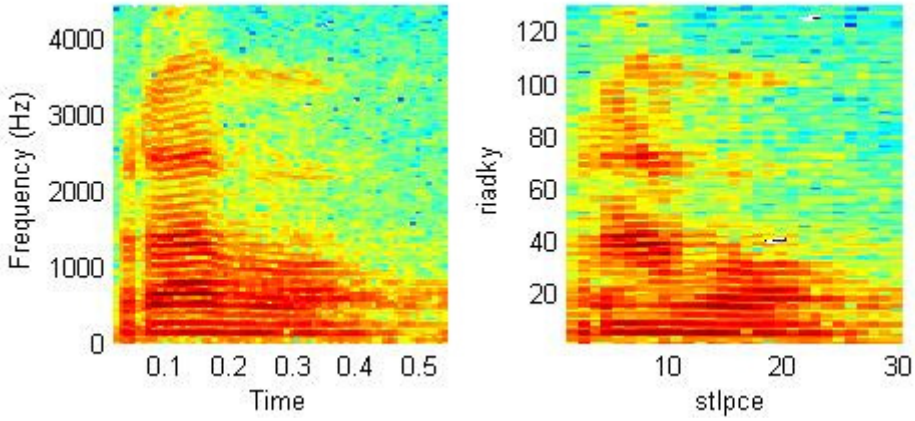
V systéme je použitá dĺžka DFT 256, čo je istým kompromisom medzi množstvom dát a rýchlosťou pri vzorkovacej frekvencii 8820Hz. Počet prekrývajúcich sa segmentov bol zvolený na 200. Funkcia vracia maticu komplexných čísel. Aby sme mohli ďalej pracovať s hodnotami, prevedieme ich do reálnej oblasti. Použitá je absolútna hodnota, aby sme získali modul komplexných údajov.

Zmena rozmerov spektrogramu

V kapitole 2 sme sa oboznámili s tým, že reč je nestacionárna. Získané spektrogramy majú preto odlišné rozmery. Našim cieľom je však získať porovnávateľné matice, čo vykonáva funkcia *resampSpec*. Má dva vstupné parametre:

- `Spectrogram` – vstupná matica
- `c_new` – počet stĺpcov (čas), na ktoré chceme maticu preformovať

Z výpočtov spektrogramov som stanovil `c_new` na konštantnú hodnotu 30. Rozdiel medzi pôvodnou a upravenou maticou spektrogramu je vidieť z Obr. 5.6.

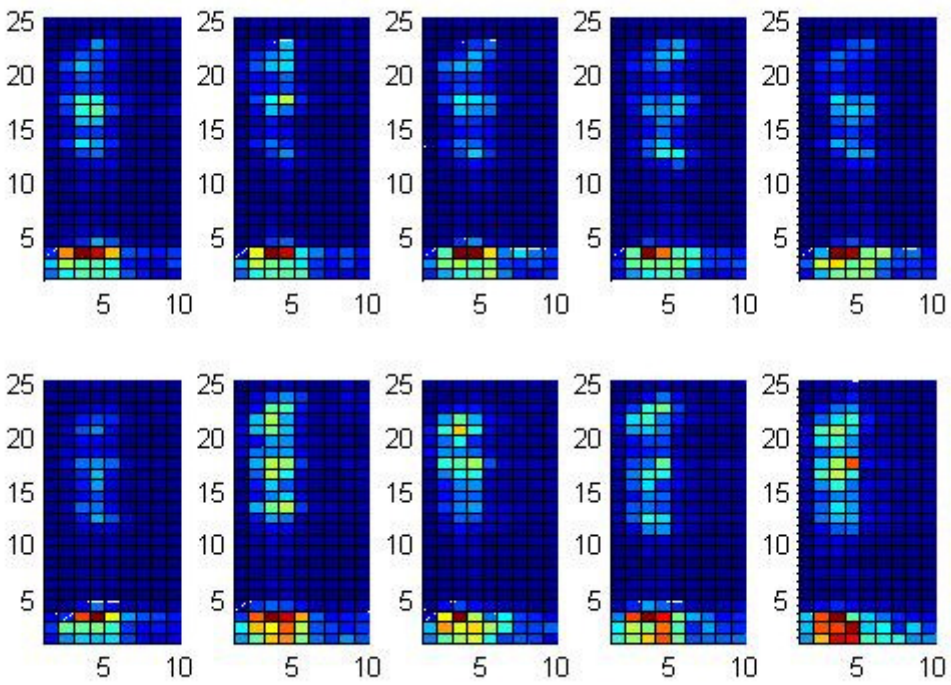


Obr. 5.6: Pôvodný a upravený spektrogram

Matica čŕt

Dostali sme spektrogramy konštantných rozmerov, ktoré sa dajú navzájom porovnávať. Ak by sme porovnávali samotné spektrogramy, bolo by to neefektívne a k tomu by sa len ťažko dosahovala istá variabilita. Z toho dôvodu spektrogramy rozdelíme na sektory, ktoré budú reprezentované jedným číslom.

Funkcia *fMatica* rieši práve tento problém. Je dôležité, aby sme zvolili vhodnú veľkosť sektorov. Ak by sme zvolili veľké rozmery, náhrada by nebola ideálna, rozpoznanie by nebolo presné (stačilo by to, ale pre úplne odlišné slová – napr. áno, nie). Na druhej strane, ak by sme zvolili príliš malé rozmery sektorov, výpočty by boli náročnejšie na čas a program by bol pomalší. Pre daný systém je zvolený časový posun tri stĺpce a frekvenčný posun päť stĺpcov. Náhrada sektoru, alebo črta, je charakterizovaná jeho aritmetickým priemerom. Obr. 5.7 znázorňuje podobnosť hovorených „Jeď".

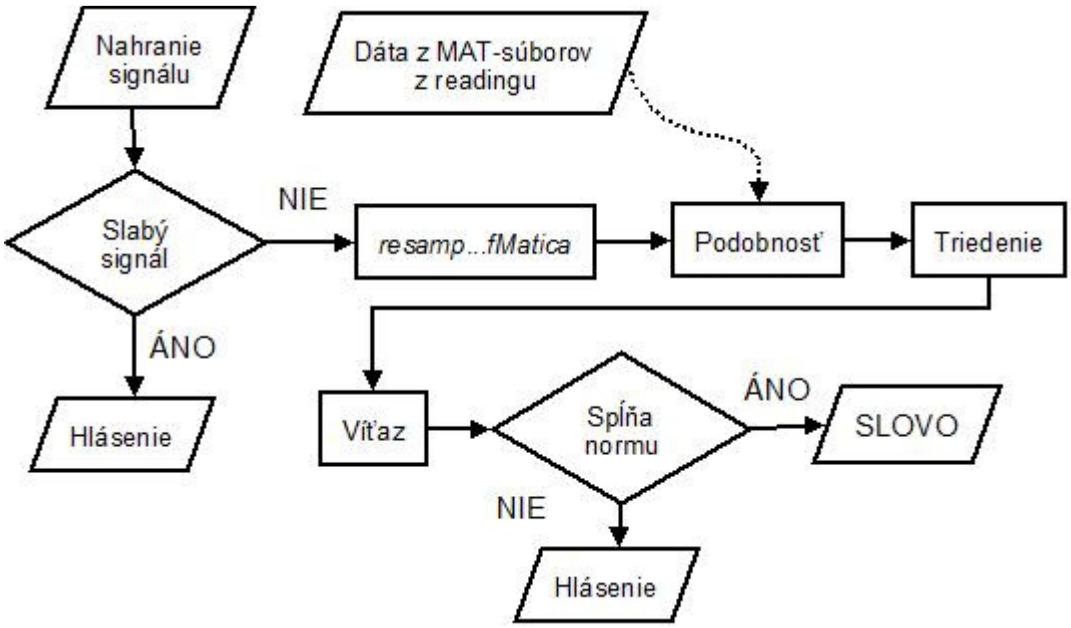


Obr. 5.7: Podobnosť príkazov Jeď v triede

Posledný krok ktorý program *Reading* vykoná je uloženie dát do štruktúry, ktorú potom prevedie do MAT-súboru, odkiaľ si údaje načíta program *Recognizer*.

5.4 Rozpoznávanie – RECOGNIZER

Po vytvorení vzorov a uložení dát môže začať rozpoznávanie. Na vstup sa privedie hovorené slovo, ktoré chceme rozpoznávať. V podstate sa vykonajú rovnaké operácie ako v načítaní dát, s tým rozdielom, že v *Recognizery* sa pokračuje vyhodnocovaním a triedením testovaného a referenčného slova. Na Obr. 5.8 je ukázaný postup príkazov v programe *Recognizer*. Podrobnejší popis je uvedený v tejto kapitole.



Obr. 5.8: Schéma procesov v Recognizery

Riešenie intenzity signálu

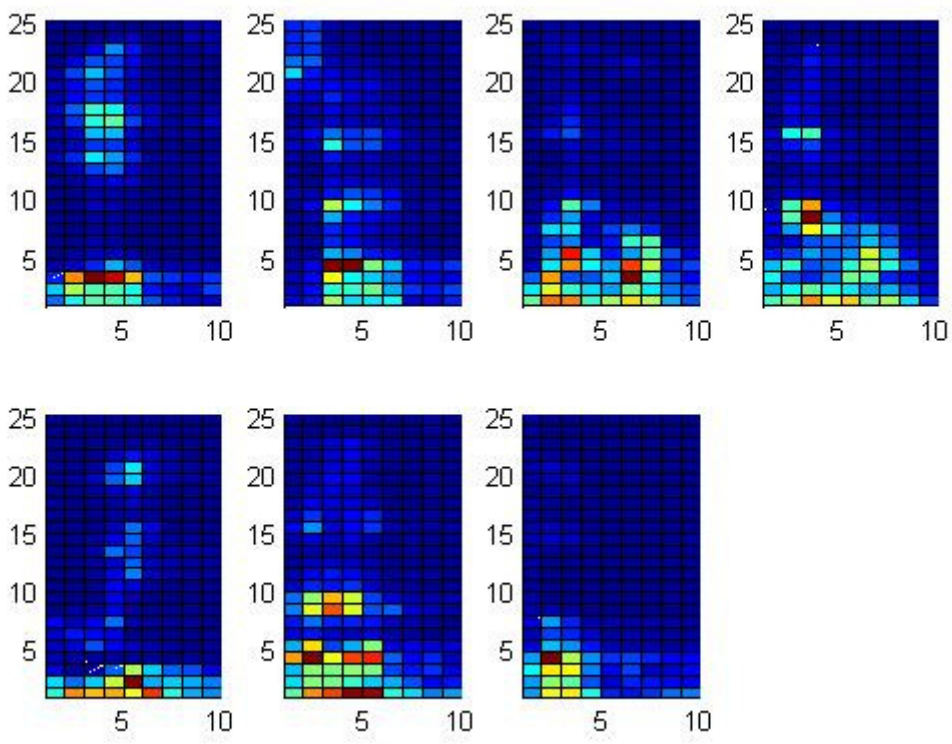
Pri nahrávaní sa môže stať, že je rečník príliš ďaleko od mikrofónu, alebo jeho citlivosť je príliš malá a nahratý signál je slabý. Alebo sa môže stať, že rečník nevysloví žiadne slovo. V takomto prípade sa neuskutoční rozpoznanie, pretože by jeho úspešnosť bola menšia ako so signálom v lepšej, intenzívnejšej kvalite. Namiesto toho program vypíše hlásenie, upozorní rečníka aby hovoril hlasnejšie. Podmienka je stanovená na hodnotu, ktorú nepresiahne šum v tichom prostredí.

Vymazanie nábehu mikrofónu

Nepatrnú odlišnosť od príkazov, ktorými prechádzal signál v *Readingu*, je vymazanie nábehu mikrofónu. Každý mikrofón má pri jeho zapnutí pulzový nábeh na nulu (tzv. recording onset). Tento problém zanikne použitím funkcie *vymazNabeh*. Kým príde signál k nule, pulzy majú zápornú hodnotu. Funkcia zmaže záporné hodnoty na začiatku signálu, kým nie je prvý krát dosiahnutá kladná hodnota pulzu.

Podobnosť

Po vykonaní všetkých procesov pre vstupný testovaný signál ako v *Readingu*, potrebujeme vedieť, ako sa matica čŕt referencie podobá matici čŕt testu. V skratke potrebujeme vedieť ako veľmi sú zhodné alebo odlišné sektory matíc. Najjednoduchším spôsobom ako to zistiť, je spočítať vzdialenosti medzi týmito sektormi. V systéme je používaná Euklidova vzdialenosť (viď rovnicu 4.2).



Obr. 5.9: Odlišnosť matíc čŕt. Hore zľava: Jeď, Vpřed, Vlevo, Vpravo, Stůj, Stát, Stop

Euklidova vzdialenosť je počítaná z testu pre každú referenciu. Jedna dvojica je preto určená jedným číslom.

Triedenie

Čísla sú následne usporiadané podľa toho, z akej triedy pochádzajú. Testované slovo vyhodnotí svoje vzdialenosti s referenciami pre danú triedu a uloží ich do matice, alebo vektoru s názvom `FeatureN`, kde N je číslo triedy. Dostaneme maticu, kde sa nachádzajú vzdialenosti medzi testovaným slovom a všetkými vzormi:

```
Features = [Feature1,Feature2,Feature3,Feature4,Feature5,Feature6,Feature7]
```

Vyhodnotenie víťaza

Pre určenie toho, ktoré slovo bolo povedané, hľadáme najmenšiu vzdialenosť matíc čŕt. Hľadáme teda minimum v matici `Features`. Podľa toho, kde sa dané minimum nachádza, určujeme triedu z ktorej pochádza.

5.5 Normy - NORMS

Aby sa zabránilo prípadu, že bude *Recognizer* rozpoznávať slová zo slovníka, aj keď bude na vstup nahraté slovo, ktoré z neho nepochádza, musí vzdialenosť spĺňať istú normu. Norma môže byť spočítaná v pomocnom programe a do *Recognizera* vložená ako konštanta, aby sme zbytočne nespomaľovali program. Program sa nazýva *Norms* a výpočet je nasledovný:

Spočíta sa priemerná vzdialenosť slov v jednej triede. Do priemeru sa zahrnie smerodajná odchýlka, ktorú dané hodnoty nadobudnú.

Tu je na mieste polemizovať, aké dáta z triedy vybrať pre výpočet noriem. Tzn. vyberieme jeden príkaz z triedy, od ktorého meriame vzdialenosť k iným príkazom v rovnakej triede, takže veľmi záleží práve na výbere tohto príkazu. Pre výpočet noriem v mojom systéme som vybral vždy prvý príkaz rečníka 2 (čiže mňa) v triede.

Vzdialenosti sa počítajú postupne pre dáta *rec_1, rec_2, rec_3.* A ukladajú sa do matice, z ktorej sa spočíta priemer a smerodajná odchýlka. Nakoniec sú normy uložené do MAT-súboru s názvom *normy.mat*.

5.6 Slovník, trénovanie a úspešnosť

Veľkosť slovníka ovplyvňuje v systéme veľa dôsledkov. Ak je požadované aby systém správne pracoval pre viacero rečníkov, musíme rozšíriť databázu. To však ovplyvňuje chod programu. Čím je počet referencií väčší, tým majú matice `FeatureN` väčší rozmer a tým je program pomalší. Možno to riešiť náhradou všetkých referencií nahratých jedným rečníkom za ich priemer. Ten však nepokryje dynamiku reči (ak niekto povie občas slovo značne odlišne). Preto je nutné zvoliť správny pomer požiadavky/veľkosť slovníku. V systéme sú tri zdroje referencií, pri ktorých trvá výpočet približne 3 sekundy.

Ďalším faktorom, ktorý ovplyvňuje rozpoznávanie je prostredie a spôsob, akým bol model slovníka natrénovaný. Systém, ktorý je produktom tejto práce bol trénovaný stolovým mikrofónom typu Trust MC-1500, v nehlučnom prostredí. Slová boli nahraté dvoma rečníkmi. Z toho jeden rečník nahrával databázu slov v dvoch odlišných priestoroch. Zložky rec_1 a *rec_2* obsahujú záznamy zo stredne veľkej izby, zložka *rec_3* bola nahrávaná vo väčšej miestnosti s malou ozvenou.

Úspešnosť systému pri vlastnom testovaní je zobrazená v tabuľke na Obr. 5.10. Percentuálna úspešnosť je vyjadrená z 20 pokusov pre jednu triedu.

Slovo	Úspešnosť	Nespĺňa normu	Iný príkaz
Jeď	80%	10%	10%
Vpřed	60%	25%	15%
Vlevo	75%	5%	20%
Vpravo	75%	0%	25%
Stůj	80%	15%	5%
Stát	75%	5%	20%
Stop	80%	10%	10%

Obr. 5.10: Tabuľka úspešnosti pri rozpoznávaní

Pri testovaní systému som sa snažil hovoriť rôznou intonáciou. Pri nižšom tóne systém prestával prepúšťať príkazy cez normu, alebo si ich plietol s inými príkazmi. Zo skúsenosti je teda dobré a vhodné artikulovať a hovoriť jasným hlasom.

Reakcia systému na iného rečníka je na Obr. 5.11.

Slovo	Úspešnosť	Nespĺňa normu	Iný príkaz
Jeď	75%	15%	10%
Vpřed	70%	25%	5%
Vlevo	70%	5%	25%
Vpravo	65%	5%	30%
Stůj	80%	10%	10%
Stát	80%	5%	15%
Stop	70%	25%	5%

Obr. 5.11: Tabuľka úspešnosti pri rozpoznávaní – iný rečník

Z tabuľky na Obr. 5.11 je vidieť, že úspešnosť mierne poklesla, no v niektorých prípadoch naopak stúpla. Konkrétne, v prípade slova „Vpřed“ sa mi nepodarilo kvôli spoluhláske „ř“ povedať slovo rovnako. Keďže rečník 2 je českej národnosti, úspešnosť rozpoznania mu tým pádom stúpla. Pri daných podmienkach sa dá tento výsledok považovať za vyhovujúci.

Pre správne fungovanie systému je potrebné, aby boli najskôr aspoň raz spustené programy *Reading* a *Norms*. Následne môže prebiehať rozpoznávanie toľko krát, koľko potrebujeme. Pri akejkoľvek zmene slovníka je nutné spustiť programy *Reading* a *Norms*, inak sú v systéme použité dáta z predchádzajúcej databázy.

Kapitola 6

Záver

Cieľom tejto štúdie bolo implementovať získané poznatky v rešeršnom skúmaní do vlastného systému pre rozpoznávanie príkazov. Cieľ bol splnený metódou, ktorá je menej rozvinutá ako tie, ktoré boli spomínané v rešerši. Príčiny môžu byť v tom, že tento spôsob je vhodný iba pre rozpoznávanie s malou databázou slov a je z časti závislý na rečníkovi.

Pri rozpoznávaní teda ovplyvňujú úspešnosť tieto faktory:

- prostredie – malo by byť nehlučné, alebo s konštantným šumom
- veľkosť slovníka – je vhodné mať v databáze viacero slov (treba však brať ohľad na rýchlosť rozpoznávania)
- rečník – je vhodné, aby v databáze boli aj príkazy nahraté rečníkom, ktorý testuje rozpoznávanie

Rozpoznávanie príkazov má využitie v robotike pri diaľkovom ovládaní mechanizmov. Pre skupinu rečníkov, ktorý budú mechanizmy ovládať hlasom, je vhodné nahrať databázu ich hlasom. Potom rozpoznanie prebieha bez väčších problémov. Na priloženom disku sa nachádza zdrojový kód systému. Na spustenie jednotlivých programov je nutné, aby bol v počítači nainštalovaný Matlab a Signal Processing Toolbox. Najskôr sa spúšťa *Reading* a *Norms*. Následne môže prebiehať rozpoznávanie, pomocou programu *Recognizer.*

Systém bol spracovaný off-line. To znamená, že po stlačení tlačidla prebieha nahrávanie – tzv. push-to-talk (viď kap. 3). Spracovanie on-line sa z časových dôvodov nepodarilo. Môže byť však predmetom ďalšieho skúmania a rozšírenia problematiky. On-line systém by umožňoval neustále nahrávanie s detektorom rečovej aktivity. Išlo by teda o vylepšenie tejto aplikácie, bez potreby stláčania tlačidla.

Literatúra

[1] PLANNERER, B., *An introduction to speech recognition*, 2005, Dostupné na internete: http://www.speech-recognition.de/pdf/introSR.pdf

[2] ČERNOCKÝ, J., *Zpracování řečových signálů – studijní opora*, 2006, Dostupné na internete: http://www.fit.vutbr.cz/study/courses/ZRE/public/

[3] WAIBEL, A., Lee K.-F., *Readings in speech recognition*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990

[4] ČERNOCKÝ, J., *Časové zpracování pro výpočet příznaků v rozpoznávání řeči*, Vutium, Brno, 2003

[5] JURÁČEK, D., *Klasifikátor izolovaných slov na bázi umělé neurónové sítě*, Dostupné na internete: http://dsp.vscht.cz/konference_matlab/matlab05/prispevky/juracek/juracek.pdf

[6] FEDOR, Z., Sinčák, P., *Inkrementálny systém pre rozpoznávanie slovných povelov*, 2008, Dostupné na internete: http://www.ai-cit.sk/source/mt/rozpoznavanie-slov.pdf

[7] HRIC, M., *Integrácia neurónových sietí typu ARTMAP s prvkami fuzzy systémov pre klasifikačné úlohy*, Dostupné na internete: http://www.ai-cit.sk/source/publications/thesis/master_thesis/2000/hric/html/index.html

[8] ČERNOCKÝ, J., *Systémy,* prednáška dostupná na internete: http://www.fit.vutbr.cz/study/courses/ISS/public/pred/syst_konv/syst_konv.pdf

[9] Homemade speech recognition, Dostupné z: http://www.mperfect.net/noReco/

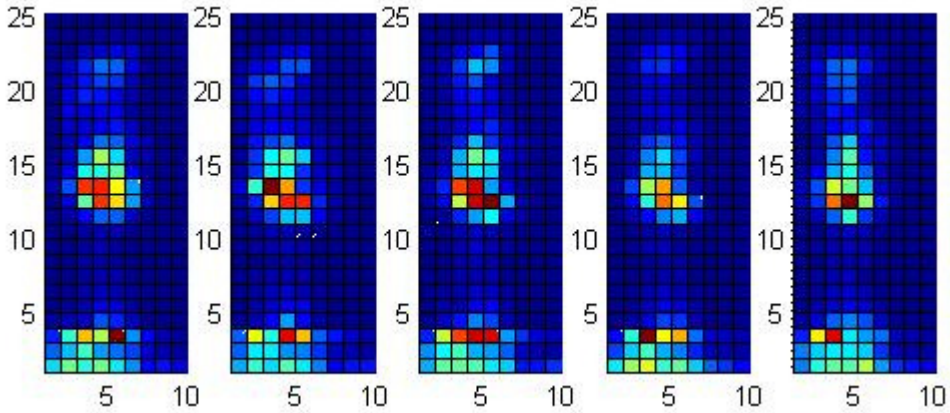
[10] WEICHEL, Ch., 32 leaves, September 26, 2009, Dostupné z: http://blog.32leaves.net/?m=200909

[11] http://sk.wikipedia.org/wiki/Rozpozn%C3%A1vanie_re%C4%8Di

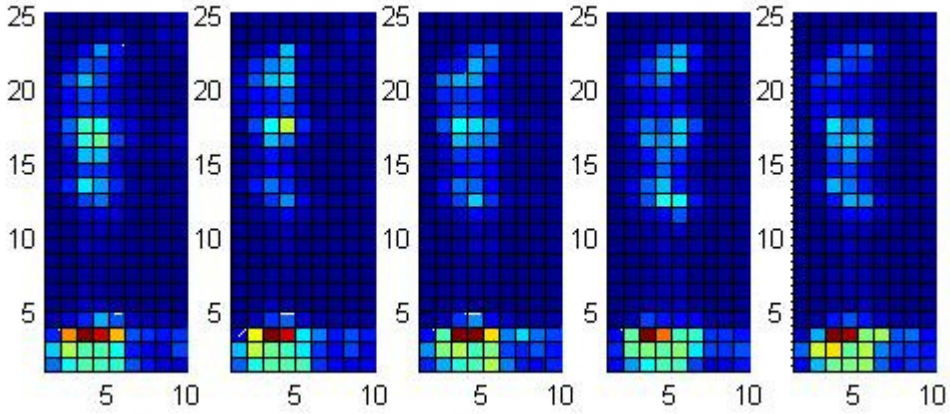
Zoznam príloh

Prílohy

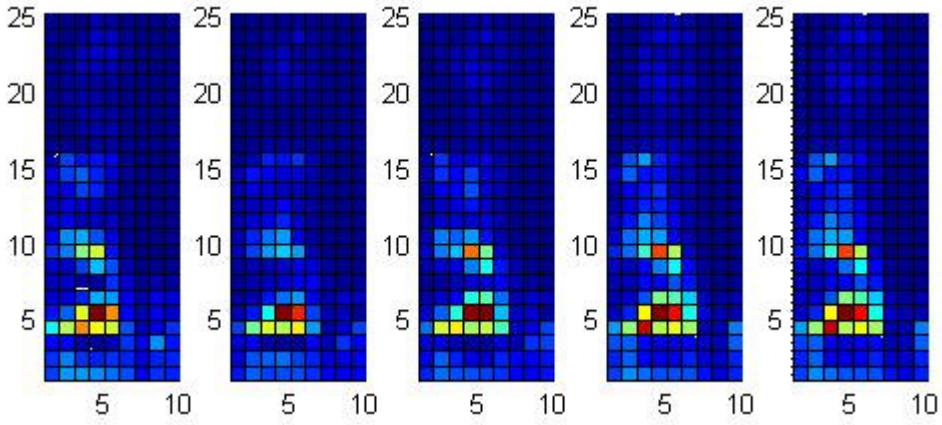
Príloha 1:



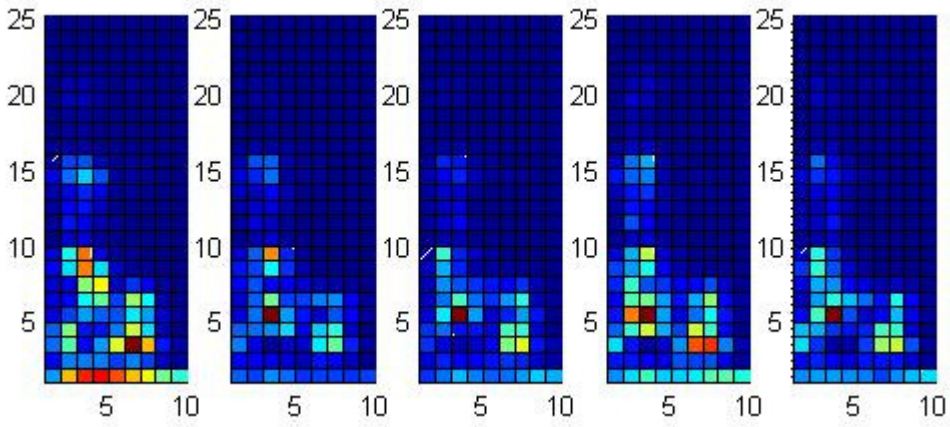
Príloha 2:



Príloha 3:



Príloha 4:



Príloha 5: