VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Václav Martinek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

HLEDÁNÍ SLOVNÍKU PRO AUDIOSIGNÁLY
SEARCH FOR A DICTIONARY FOR AUDIOSIGNALS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Václav Martinek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Václav Martinek ***ID:*** 171620
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Hledání slovníku pro audiosignály

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku reprezentačních systémů pro jednorozměrné signály, včetně Fourierovy a krátkodobé Fourierovy syntézy. Nastudujte algoritmy z oblasti strojového učení, které signálový slovník hledají tak, aby byl přizpůsobený nějakému typu signálu. Implementuje takovéto algoritmy v MATLABu. Vyhodnoťte úspěšnost algoritmů jak objektivně, tak neformálně subjektivně, s použitím malé databáze zvuků, kterou je možno přejmout/adaptovat z dostupných volných zdrojů. Bakalářská práce by měla odpovědět na otázku, zda učení slovníku přináší výhody z hlediska úspornosti (tj. řídkosti) reprezentace a za jakou cenu.

DOPORUČENÁ LITERATURA:

[1] Hrbáček, R., Rajmic, P., Veselý, V., Špiřík, J. Řídké reprezentace signálů: úvod do problematiky, Elektrorevue, 2011. ISSN 1213-1539

[2] Špiřík, J., Rajmic, P., Veselý, V. Reprezentace signálů: od bází k framům, Elektrorevue, 2010. ISSN 1213-1539

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Mgr. Pavel Rajmic, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce by měla odpovědět z čeho se skládá audio signál. Zabývá se tedy učením slovníků pro audio signály na základě řídkých reprezentací. Jsou zde popsány jednotlivé algoritmy, které jsou důležité k vytvoření přizpůsobených slovníků. Dále v práci nalezneme srovnání reprezentace signálu pomocí fourierovy transformace a natrénovaného slovníku. Je zde popspána tvorba databáze hudebních zvuků.

KLÍČOVÁ SLOVA

Fourierova transformace, K-SVD, řídká reprezentace signálů, učení slovníků

ABSTRACT

This thesis should answer what constitutes audio signal. It deals with the dictionary learning based on sparse representations of signals. This thesis describing algrithms, which are important for the creation of learning dictionaries. We find a comparison of signal representation using Fourier transform and learned vocabulary. It describes the creation of a database of musical sounds.

KEYWORDS

dictionary learning, Fourier transform, K-SVD, sparse representation of signals

MARTINEK, Václav *Hledání slovníku pro audiosignály*: diplomová práce. Místo: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 37 s. Vedoucí práce byl doc. Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Hledání slovníku pro audiosignály“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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.

Místo ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Místo ....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Libovolný periodický signál lze vyjádřit, rozložit na součet stavebních bloků. Tyto stavební bloky bývají vyjádřeny pomocí harmonických funkcí. Systém (slovník) vytvořený tímto způsobem nazýváme jako univerzální. Úkolem této bakalářské práce je nastudovat, popsat a navrhnout algoritmy, které jsou schopny vytvářet slovníky přímo na „míru“ pro daný audio signál.

V poslední době se pro zpracování zvuku a obrazu začíná využívat tzv. řídké reprezentace. Reálné zvuky např. hudebních nástrojů se skládají poměrně z malého počtu stavebních bloků. Pochopení řídké reprezentace signálu má široké využití v reálném světě. Zahrnuje spoustu matematických disciplín a stále se objevují nové aplikace. Úkolem této práce je subjektivně i objektivně porovnat úspěšnost těchto algoritmů.

První kapitola se zaměřuje na vysvětlení problematiky řídkosti, je zde vysvětleno, čemu říkáme slovník. Druhá kapitola by nám měla vysvětlit, jaké typy slovníků máme a jaký slovník je pro naši aplikaci výhodný. Jsou zde popsány jednotlivé algoritmy. Další kapitola se zabývá prostředím MATLAB, návrhem algoritmu, samotným řešením a provnáním úspěšnosti. Součástí bakalářské práce je vytvoření databáze hudebních zvuků, signálů. O vytváření databáze a úvahou nad vhodným záznamem pojednává čtvrtá kapitola.

1 ŘÍDKÁ REPREZENTACE SIGNÁLŮ

V dnešní době oblast řídké reprezantace signálů zaměstnává spousty vědců, kteří stále objevují nová odvětví, kde se tato problematika může aplikovat. Historie této problematiky sahá do počátku 90. let 20. století. Nejsledovanější aplikací je komprimované snímání. Tak zvaná řídká reprezentace signálu (sparse signal representation) lze chápat jako nedoručený systém lineárních rovnic s existujícím řešením, které má velmi málo nenulových proměných. Zjednodušeně můžeme říct, že existuje více neznámých než lineárních vztahů mezi nimi. Můžeme tedy nalézt nekonečně mnoho řešení. Základní úlohu definujeme jako $\mathbf{Dx} = \mathbf{y}$, kde $\mathbf{D}$ je matice koeficientů, kterou nazýváme slovník, $\mathbf{x}$ je vektor neznámých, ve kterém nacházíme řídkost, jako $\mathbf{y}$ značíme námi známý vektor (většinou reálný naměřený signál). Nejvíce atraktivní jsou pro nás řešení, které obsahují co nejvíce nulových neznámých. Taková řešení nazýváme jako řídká řešení a jsou velmi výhodné z důvodů usnadnění interpretace a komprese dat. Podle očekávání se teorie opírá o mnoho matematických odvětví, mimo lineární algebry také o optimalizace, aproximace atd. [3]

1.1 Základní značení

Pro lepší přehlednost budeme v této práci značit matice velkými tučnými písmeny např. $\mathbf{A}$. Vektory označujeme malými tučnými písmeny například $\mathbf{a}$. Počet prvků množiny značíme jako absolutní hodnotu $|\{0, 2, 5, 1, 0\}| = 5$. Tuto hodnotu nazýváme také jako kardinalita. Předpokládaná indexace prvků vektoru začíná jedničkou $\mathbf{x} = [x_1, \ldots, x_n]$ s velikostí N. Nosičem vektoru myslíme všechny jeho prvky, ve kterých má vektor nenulové hodnoty. Taková množina se potom značí $supp(\mathbf{x})$. Tzn. $supp(\mathbf{x}) = \{i \mid_i \neq 0\}$. Pro námi již zmíněný signál $\mathbf{x} = [0, 2, 5, 1, 0]$ je $\text{supp}(\mathbf{x}) = \{2, 5, 1\}$ tedy $|supp(\mathbf{x})| = 3$. V řídké reprezentaci signálů je pro nás nejdůležitější tzv. řídký vektor, který obsahuje spoustu nulových hodnot, čím menší je nosič vektoru a zároveň větší počet prvků, tím je pro nás vektor atraktivnější, jelikož je řídší.

Normou vektoru se myslí číslo, které nám říká, jaká je velikost vektoru. Široké veřejnosti nejznámější a nejpoužívanější je norma eukleidovská. Tato norma se v našich aplikacích využívá dosti zřídka. Proto zde definujeme další typy norem: [3]

$$\|\mathbf{x}\|_p := (\sum_{i=1}^{N} |x_i|^p)^{1/p} \quad \text{pro } 0 \leq p \leq 1, \tag{1.1}$$

$$\|\mathbf{x}\|_p := \sum_{i=1}^{N} |x_i|^p \quad \text{pro } 0 < p < 1, \tag{1.2}$$

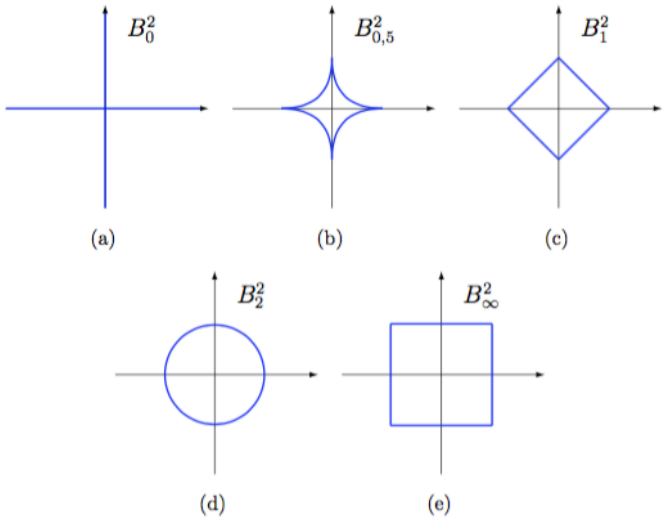
$$\|\mathbf{x}\|_\infty := \max_i |x_i|, \tag{1.3}$$

$$\|\mathbf{x}\|_0 := |\text{supp}(\mathbf{x})|. \tag{1.4}$$

Správně bychom měli označovat normu pouze v případě, že $1 \leq p \leq \infty$. [3] Pro přehlednost a zjednodušení používáme pro všechna p jednotné označení l_p-norma. V našich aplikacích a popisech je nejdůležitější tzv. jedničková norma $\|\mathbf{x}\|_1 = \sum_i |x_i|$, která je součtem absolutních hodnot prvků vektoru. Druhou často využívanou normou je norma nulová $\|\mathbf{x}\|_0$, ta představuje počet nenulových složek vektoru, tedy absolutní hodnotu nosiče vektoru. Pro lepší pochopení jednotlivý norem je dobrá vizualizace pomocí jednotkových koulí v jednotlivých normách. Jednotkovou koulí rozumíme všechny body od počátku ve vzdálenosti 1 a menší.

$$B_p^N := \{x \in C^N \mid \|\mathbf{x}\|_p \leq 1\} \tag{1.5}$$

Avšak v každé z norem jsou velikosti 1 jiné vzdálenosti od počátku. Jednotková koule v normě l_0 kopíruje osy souřadnicového systému, jak vyplývá z definice normy l_0.



Obr. 1.1: Ilustrace jednotkovych kouli (a) B_0^2, (b) $B_{0,5}^2$ (c) B_1^2 a (d) B_2^2 (e) B_∞^2

Dále je dobré si určit řídký vektor a stupeň řídkosti, do kterého bude $\mathbf{x}$ stále řídké. Za k-řídký vektor se dá považovat takový, který má počet nenulových složek

rovný nebo menší než k. Jinými slovy pokud je nultá norma rovna nebo menší než k $\|\mathbf{x}\|_0 \leq k$. Relativní řídkostí vektoru $\mathbf{x}$ o délce N se rozumí poměr $\frac{k}{N}$. K označujeme jako maximální počet nenulových složek. [3]

Reálné řídké signály, se kterými budeme pracovat, nejsou řídké tak, jak jsme si je definovali. Většinou obsahují třeba šum a potom nejsou hodnoty přímo rovny nule, ale jsou nule velmi blízké. Pro tento stav je dobré definovat chybu aproximace. Chybou aproximace se rozumí chyba mezi $\mathbf{x}$ a k-řídkým vektorem. Je-li $\mathbf{x}$ přímo k-řídký vektor, potom bude chyba rovna nule pro jakoukoliv normu. Při výpočtu chyby záleží s jakou normou počítáme.

$$\sigma_k(\mathbf{x})_p := \inf_{z \in \sum_k^N} \|\mathbf{x} - \mathbf{z}\|_p \tag{1.6}$$

Číslo $spark(\mathbf{X})$ je maximální počet lineárně závislých sloupců v matici. Tento údaj matice je důležitý pro rozhodování existence řešení. Čím je spark matice menší, tím větší musí být řídkost vektoru $\mathbf{x}$, aby bylo zajištěno řešení.

Vektorový prostor můžeme chápat jako množinu čísel, která je těmito vektory vymezená (popsána), jedná se tedy o zobecnění nějaké množiny čísel (reálné, komplexní...). Bází vektorového prostoru je mnozina linearne nezavislych vektoru, jejichz linearni kombinaci lze dosahnout na libovolny vektor v uvazovanem vektorovém prostoru. Báze vektorového prostoru můžeme dělit na ortogonální a ortonormální. V ortogonální bázi jsou dvojice bázových vektorů vzájemně kolmé. Jako ortonormální bázi označujeme ortogonální bázi, jejichž dvojice bázových vektorů jsou jednotkové a tedy i kolmé. Konečně si tedy můžeme definovat pojem frame. Frame vzniká, když rozšíříme bázi vektorového prostoru o další vymezující („nadbytečný“) vektor, který už bude lineárně závislý a tedy počet generátorů prostoru bude větší než dimenze prostoru. [2]

1.2 Aplikace řídkosti

Jak již bylo zmíněno, teorie řídkých signálů má velké využití ve zpracování signálu, především jedná-li se o zvuk nebo obraz. Aplikovat můžeme všude, kde o signálu víme, že je signál řídký v nějakém systému. Mezi významné aplikace řadíme: kompresi signálu, potlačování šumu, separaci různých prvků signálu, zvyšování rozlišení, doplňování chybějících vzorků signálu a další. Využití můžeme také najít v konstrukci nových typů A/D převodníků. Nejatraktivnější aplikací teorie řídkých signálů je komprimované snímání, které využívají vysokorychlostní kamery, fotoaparát s jedním pixelem, magnetická rezonance a další. [3]

1.3 Metody řešení úloh s aplikací teorie řídkosti

V předešlých kapitolách jsme si vysvětlili základní pojmy a tudíž můžeme prozkoumat metody řešení. Dělíme je do dvou hlavních skupin podle použitého typu algoritmu. Dva nejznámější typy algoritmů jsou algoritmy relaxační a hladové. Všechny typy algoritmů jsou iterativní tedy cyklické a jsou založeny na aproximaci. Algoritmus nalezne výsledek, který je velmi málo odlišný od přesného řešení daného problému.

$$(P_0) \min_x \|\mathbf{x}\|_0 \quad \text{vzhledem k} \quad \mathbf{y} = \mathbf{D}\mathbf{x} \tag{1.7}$$

1.3.1 Relaxace

Jelikož l_0 je nekonvexní funkce, tak je v aplikacích neupočitatelná. Nejbližší konvexní norma, která je zároveň vhodná pro náš problém je norma l_1. Potom můžeme relaxační úlohu definovat jako:

$$arg_x \min\|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \mathbf{D}\mathbf{x} = \mathbf{y} \tag{1.8}$$

Reálný signál může obsahovat šum, potom můžeme relaxovanou úlohu upravit jako:

$$arg_x \min\|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \|Dx - y\|_2 \leq \delta \tag{1.9}$$

1.3.2 Hladové Algoritmy

Jak již vyplývá z názvu, jsou tyto algoritmy „násilné, agresivní“. V angličtině takový algoritmus nazýváme jako greedy. Jedná se o iterační algoritmy, které v každém kroku zvolí jeden nebo více atomů, které nejlépe aproximují signál. Tyto atomy se poté podílejí na dalších iteracích a tak ovlivní celé řešení. Hladové algoritmy jsou velice rychlé, nemusí však dosáhnout optimálního minima. Jedním ze zástupců hladových algoritmů, je algoritmus OMP (Orthogonal Matching Pursuit). Tento hladový algoritmus je volán jako jeden krok algoritmu K-SVD, který bude podrobně popsán v kapitole 2.2.2. V problému 1.8 je podmínkou hladových algoritmů, že matice slovníku $\mathbf{D}$ musí mít $spark(\mathbf{D}) > 2$ a jediné řídké řešení $\mathbf{D}\mathbf{x} = \mathbf{y}$. Potom o $\mathbf{y}$ říkáme, že je skalárním součinem daného sloupce slovníku a řešení je jednoznačné. Hladový algoritmus v j-té iteraci vynásobí všechny atomy $\mathbf{D}$ s vektorem $\mathbf{y}$ a vytvoří j složek vektoru $\epsilon(j)$: [1]

$$\epsilon(j) = \min_{\mathbf{z}_j} \|\mathbf{d}_j \mathbf{z}_j - \mathbf{y}\|_2^2 \tag{1.10}$$

pro $\mathbf{z}_j$:

$$\mathbf{z}_j = \mathbf{d}_j^T \mathbf{y}^T / \ \|\mathbf{d}_j \mathbf{z}_j - \mathbf{y}\|_2^2 \tag{1.11}$$

2 SLOVNÍKY

Teorii řídkých signálů rozumíme jako systém nedoručených lineárních rovnic, kde platí: $\mathbf{y} = \mathbf{D}\mathbf{x}$. Matici $\mathbf{D}$ označujeme jako slovník, vektor $\mathbf{x}$ jako neznámé (řídký vektor) a vektor $\mathbf{y}$ jako naměřené (pozorované) hodnoty tedy signál. Jednotlivým sloupcům matice (slovníku) říkáme atomy. Slovník může být libovolný zástupce báze nebo framu. Takový slovník nazýváme jako univerzální nebo statický. Nastává však otázka, zda-li si můžeme vytvořit, naučit slovník speciální, přímo „na míru“ tak, aby bylo docíleno co nejřidšího a co nejpřesnějšího vyjádření signálu. Takový slovník pak budeme nazývat přizpůsobený. Natrénování slovníku vyžaduje mnohem větší výpočetní náročnost.

2.1 Statické slovníky

Jako statický slovník označujeme matice sestavené z pevné funkce. Podle funkce, která se podílela na vzniku slovníku, obsahují jednotlivé koeficienty tzn. atomy informace. Pro zvuk jsou to například frekvence, fáze, amplitudy. Čím více parametrů takový slovník nese, tím více by měl zmenšit chybu při syntéze původního signálu y. Mezi nejznámější statické slovníky řadíme slovníky DCT či Gaborův.

2.1.1 DCT slovník

Zkratka DCT je zkratkou pro diskrétní cosinovu transformaci (Discrete Cosine Transform). Tento slovník je používaný právě pro svoji nenáročnost a jednoduchost. DCT může být vícerozměrné, proto si zde uvedeme předpis, jak získat koeficienty DCT. Vždy budeme uvažovat $\mathbf{Y}$ jako původní signál a $\mathbf{X}$ jako signál transformovaný. Koeficienty jednorozměrné DCT získáme jako: [4]

$$\mathbf{X}_i = \alpha_i \sum_{j=0}^{n-1} \mathbf{Y}_i \cos\left(\frac{\pi(2j+1)i}{2n}\right) \tag{2.1}$$

kde α_i určíme jako:

$$\alpha_i = \begin{cases} \sqrt{\frac{1}{n}}, & \text{pro } i = 0 \\ \sqrt{\frac{2}{n}}, & \text{pro } 1 \leq i \leq n-1 \end{cases} \tag{2.2}$$

Koeficienty pro dvourozměrné DCT získáme jako transformaci nejdříve po řádcích a potom po sloupečcích. Obecně vícerozměrnou DCT získáváme jako sérii jednorozměrných transformací v každém rozměru. Koeficienty dvourozměrné DCT tedy získáme jako: [4]

$$\mathbf{X}_{i_1,i_2} = \alpha_{i_1}\alpha_{i_2}\sum_{j_1=0}^{n-1}\sum_{j_2=0}^{m-1}\mathbf{Y}_{i_1,i_2}\cos\left(\frac{\pi(2j_1+1)i_1}{2n}\right)\cos\left(\frac{\pi(2j_2+1)i_2}{2m}\right) \tag{2.3}$$

kde $\alpha_1,\ \alpha_2$ určíme jako:

$$\alpha_{i_1} = \begin{cases}\sqrt{\frac{1}{n}}, & \text{pro } i = 0\\ \sqrt{\frac{2}{n}}, & \text{pro } 1 \leq i \leq n-1\end{cases} \tag{2.4}$$

$$\alpha_{i_2} = \begin{cases}\sqrt{\frac{1}{m}}, & \text{pro } i = 0\\ \sqrt{\frac{2}{m}}, & \text{pro } 1 \leq i \leq m-1\end{cases} \tag{2.5}$$

2.1.2 Gaborův slovník

Gaborův slovník je oproti DCT slovníku lehce dokonalejší, protože součástí slovníku je informace o fázi každé frekvence. Aplikace algoritmu je potom složitější, komplikovanější a zvyšuje výrazně výpočetní náročnost. Odměnou je potom řidší výsledek. [5] Jednotlivé koeficienty matice **D** pak získáme takto:

$$d^g_{j,\varphi}(t) = W_d(t)\cos\left(\frac{\pi}{K_g}\left(t+\frac{1}{2}\right)\left(j+\frac{1}{2}\right)+\varphi\right) \tag{2.6}$$

kde K_g určuje rozměr slovníku

2.1.3 Fourierova a kratkodobá Fourierova synteza

Abychom mohli mluvit o Fourierově syntéze, musíme nejdříve popsat analýzu. Harmonická analýza je operace, při níž je periodický signál rozkládán na dílčí harmonické složky. Syntézu pak chápeme jako složení signálu z dílčích harmonických složek. Kmitočet harmonických složek určují n-násobky základní harmonické složky, každá harmonická složka má svou amplitudu a počáteční fázi. Je to tedy integrální transformace převádějící signál z časové oblasti do oblasti frekvenční. Protože se zde zabýváme signály v diskrétním čase, tak nás bude zajímat diskrétní Fourierova transformace (DFT). Výsledkem DFT bude diskrétní spektrum. Pokud je použito správné vzorkování signálu se spojitým časem, tak je DFT definována jako:

$$s_{id}(t) = \sum_{n=-\infty}^{\infty} s[nT]\delta(t-nT). \tag{2.7}$$

Pokud si chceme signál syntetizovat ze spektra, tak nás zajímá zpětná (inverzní) diskrétní Fourierova transformace.

$$S_{id}(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} S(\omega - k\omega_{vz}). \tag{2.8}$$

kde

$$\omega_{wz} = \frac{2\pi}{T}, fvz = \frac{1}{T}. \tag{2.9}$$

V roce 1964 byl pospán velmi efektivní algoritmus výpočtu DFT, tzv. rychlou Fourierovu transformaci (FFT – Fast Fourier Transform). Rychlá nebo-li krátkodobá Fourierova transformace vychází z DFT. Proto algoritmus FFT tak neleznem téměř ve všech matematických programech. Tento algoritmus je vysoce efektivní, protože jeho asymptotická složitost je linearitmická.

2.2 Přizpůsobené slovníky

Protože natrénování slovníku je výpočetně náročné, tak otázka, zda-li je vůbec možné nalézt optimální slovník, začala být studována až v roce 1996. [4] Jak již bylo řečeno, přizpůsobeným slovníkem myslíme takový slovník, který je přímo vytvořený na míru danému signálu. Prakticky se však využívají speciální algoritmy (MOD, K-SVD), které nejprve vytvoří slovník obecný a následně ho upraví, natrénují, aby byl co nejvýhodnějším slovníkem pro daný signál. Uvažujeme tedy následující optimalizační úlohu:

$$\min_{\mathbf{D},\{\mathbf{x_i}\}_{i=1}^{M}} \sum_{i=1}^{M} \|\mathbf{x}_i\|_0 \quad \text{vzhledem k } \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2 \leq \epsilon, \quad 1 \leq i \leq M. \tag{2.10}$$

Řešením takové úlohy je potom slovník $\mathbf{D}$, který je optimální pro daný signál, takový slovník nám zajišťuje pro všechny složky řídkost k_0.

2.2.1 MOD - metoda optimálních směrů

V angličtině je tato metoda známá pod názvem Method of Optimal Directions. Jak již bylo řečeno, jedná se o algoritmus, který vytváří slovník na míru danému signálu. Tato metoda přistupuje k danému problému jako ke složenému minimalizačnímu problému. Vnitřní minimalizace ovlivňuje řídkost vektoru x_i s pevně daným (fixním) slovníkem $\mathbf{D}$. Vnější minimalizace se pak týká samotného slovníku $\mathbf{D}$. Potom už samotná minimalizace vypadá celkem jednoduše. Protože se jedná o algoritmus iterační, tak v každém k-tém kroku použijeme $\mathbf{D}_{k-1}$ a tak řešíme matici $\mathbf{X}_k$. Jakmile vyřešíme $\mathbf{X}_k$, tak hledáme $\mathbf{D}_k$ jako celek pomocí metody nejmenších čtverců. [1]

$$\mathbf{D}_{(k)} = \arg\min_{\mathbf{D}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}_{(k)}\|_F^2 = \mathbf{Y}\mathbf{X}_{(k)}^T \left(\mathbf{X}_{(k)}\mathbf{X}_{(k)}^T\right)^{-1} = \mathbf{Y}\mathbf{X}_{(k)}^+ \tag{2.11}$$

Chybu počítáme pomocí Forbeinovy normy, takto můžeme opakovaně hledat řešení, dokud nejsme spokojení s výsledkem.

2.2.2 K-SVD

Oproti MOD algoritmu je K-SVD lehce složitější. Jedná se o zobecněný algoritmus K-means a SVD. K-SVD hledá slovník po částech v každé své iteraci, ne jako celek. Postupně tedy určuje každý atom zvlášť. Naším cílem je opět najít k-řídký vektor $\mathbf{x}$ a vhodný slovník $\mathbf{D}$. Po natrénování $\mathbf{D}$ vzniká podobná matice jako u DCT. Algoritmus pracuje ve dvou hlavních krocích. V prvním kroku hledá nejvhodnější vektor $\mathbf{x}$ s pevně zafixovanou maticí slovníku $\mathbf{D}$. To se většinou děje pomocí nějakého hladového algoritmu např. OMP, o kterém hovoří následující subkapitola 2.2.3. Ke každému $\mathbf{y}_i$ mi najde $\mathbf{x}_i$, parametrem může být, jak moc k-řídký $\mathbf{x}$ chci najít. V druhém kroku musím matici slovníku $\mathbf{D}$ upravovat tak, aby po zpětném vynásobení $\mathbf{D}\mathbf{x}$ byl výsledek co nejvíce podobný původnímu signálu $\mathbf{y}$. Rozdíl mezi původním signálem a signálem, který vznikne vynásobením $\mathbf{D}$ a $\mathbf{x}$, určuje tzv. chybová matice $\mathbf{E}$.

Pokud v matici slovníku $\mathbf{D}$ necháme všechny atomy stejné, kromě j_0-tého ($\mathbf{d}_{j0}$), tak jeho hodnoty můžeme měnit pomocí koeficientů z $\mathbf{X}$, které ho násobí. Když se budeme snažit vyjádřit ($\mathbf{d}_{j0}$)z výrazu $\|\mathbf{Y} - \mathbf{D}\mathbf{X}_\mathbf{k}\|_\mathbf{2}^\mathbf{F}$, tak dojdeme k výrazu:

$$\|\mathbf{Y} - \mathbf{D}\mathbf{X}_k\|_F^2 = \|\mathbf{Y} - \sum_{j=1}^{m} \mathbf{d}_j \mathbf{x}_j^T\|_F^2 = \left\| \left(\mathbf{Y} - \sum_{j \neq j_0} \mathbf{d}_j \mathbf{x}_j^T \right) - \mathbf{d}_{j_0} \mathbf{x}_{j_0}^T \right\|_F^2 \tag{2.12}$$

$\mathbf{x}_{Tj}$ nám vyjadřuje x-tý řádek matice $\mathbf{X}$. V současné chvíli se zajímáme o vhodné vyjádření $\mathbf{d}_{j0}$ a $\mathbf{x}_{j0T}$. Jako chybovou matici pak označujeme výraz v závorce. Tuto chybovou matici označujeme jako $\mathbf{E}_{j_0}$:

$$\mathbf{E}_{j_0} = Y - \sum_{j \neq j_0} \mathbf{d}_j \mathbf{x}_j^T. \tag{2.13}$$

Vhodné $\mathbf{d}_{j_0}$ a $\mathbf{x}_{j_0 T}$ můžeme získat pomocí SVD rozkladu chybové matice, bohužel většinou tento rozklad vede ke vzniku nenulových prvků, což není naším cílem, protože hledáme řídké vyjádření. Aby byl SVD rozklad použitelný, tak omezíme $\mathbf{E}_{j0}$ jen na sloupečky a řádky, které využívají j_0-tý atom. Z tohoto důvodu se zde zavádí speciální parametr P_0, kterým se násobím chybová matice $\mathbf{E}_{j0}$ – omezující operátor.

Potom už můžeme použít SVD rozklad a najít optimální slovník. [1] Při použití SVD rozkladu je pro nás nejzajímavější první sloupec matice $\mathbf{U}$, který určuje aktualizaci atomu $\mathbf{d}_{j_0}$.

2.2.3 OMP

Algoritmus je známý jako Orthogonal Matching Pursuit. Jedná se o hladový algoritmus, který je využíván algoritmem K-SVD ve svém prvním kroku, kdy hledá vhodné $\mathbf{x}$ k zafixovanému slovníku $\mathbf{D}$. Jak již bylo definováno, algoritmus má za úkol řešit rovnici 1.8, která je ekvivalentní s úlohou hledání nejmenšího $\epsilon(j)$ a největšího násobku mezi residuem $r^{(k-1)}$ a normovanými atomy matice $\mathbf{D}$. Poté přejde na vzorec l_1 normu

$$\epsilon(j) = \|\mathbf{r}^{k-1}\|_2^2 - \frac{\left(\mathbf{d}_j^T \mathbf{r}^{k-1}\right)}{\|\mathbf{d}_j\|_2^2} \tag{2.14}$$

Prakticky potom funguje algoritmus tak, že na začátku vytvoří prázdný sloupcový vektor $\mathbf{x}$ o určité délce a prázdný nosič $\mathbf{S}^k$ vektoru $\mathbf{x}$. Do residua r přiřadí vektor $\mathbf{y}$. V každé iteraci přidá minimální index složky $\epsilon(j)$ do nosiče podle 2.14. V daných iteracích nepracujeme s celým slovníkem $\mathbf{D}$, proto ho můžeme omezit na část, se kterou pracujeme a to na $\mathbf{D}_{sk}$, jehož druhý rozměr (sloupce) má stejný rozměr jako absolutní hodnota $\mathbf{S}^k$. Vektor $\mathbf{x}_{sk}$ je vytvořen z původního vektoru $\mathbf{x}$ a nosiče $\mathbf{S}^k$. Residuum vypočítáme jako $r^k = \mathbf{y}^{\smile}\mathbf{D}_{sk}\mathbf{x}_{sk}$. Matice $\mathbf{A}_{sk}$ a $\mathbf{r}^k$ jsou ortogonální. Poslední krok iterace je porovnání normy residua r^k a předem zvoleného parametru (např. k), pokud nerovnost vyhovuje, může se pokračovat další iterací. Takto postupně najdeme námi hledaný k-řídký vektor $\mathbf{x}$. [1]

3 PROSTŘEDÍ MATLAB

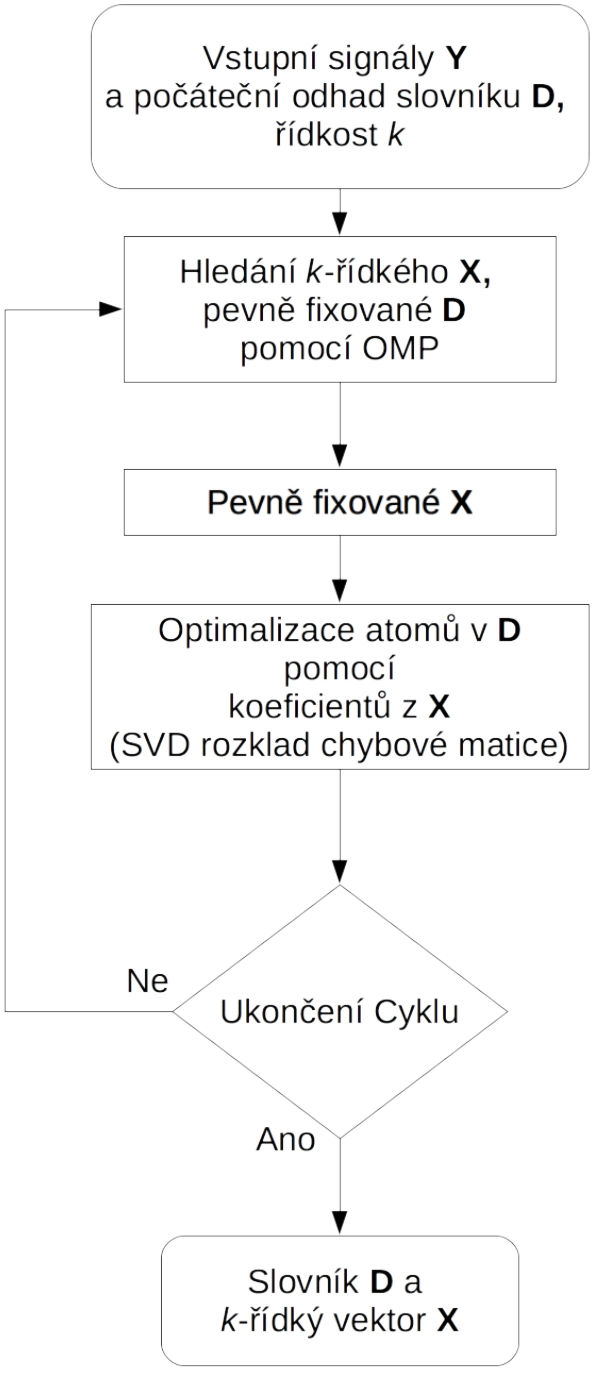
Interaktivní prostředí MATLAB je velice vhodné pro technické a vědecké výpočty. Výpočty jsou založeny na počítání s maticemi. Název vznikl od matrix laboratory. Umožňuje řešit spoustu technických problémů, aniž bychom museli programovat složitý vlastní program. Obsahuje spoustu knihoven (toolboxů), které jsou specializovány na určitý problém. Matlab také umožňuje pohodlnou vizualizaci dat v podobě 2D a 3D grafů. Trénování a testování slovníků pro audiosignály bylo prováděno ve verzi R2016a.

3.1 Návrh algoritmu

Z obrázku 3.1 je patrné, že se jedná o algoritmus K-SVD, který pro hledání k-řídkých $\mathbf{x}$ využívá hladového algoritmu OMP. V praxi má algoritmus více vstupních parametrů než je uvedeno na obrázku. Ty nejdůležitější jsou na obrázku uvedené. Matice Y ve svých sloupcích obsahuje signály $\mathbf{y}$ a vzniká rozkouskováním audio signálu na hodnoty o stejném počtu prvků, které reprezetují sloupce matice $\mathbf{Y}$. Zde by mělo platit pravidlo, kdy počet atomů ve slovníku je větší jak počet hodnot v atomu matice signálu $\mathbf{Y}$ (počet sloupců ve slovníku je větší než počet řádků matice $\mathbf{Y}$). Další proměnou je matice $\mathbf{D}$, kterou označujeme jako inicializační slovník, který ve svých sloupcích obsahuje harmonické kmity nebo náhodná čísla. Počet iterací celého algoritmu určuje iterátor i. Pokud je i nastaveno na nízkou hodnotu, tak algoritmus není schopen natrénovat slovník správně. Ideální nastavení je takové, kdy chyba $\|\mathbf{Y} - \mathbf{DX}\|_2^F$ začíná konvergovat k určité chybové hodnotě. Posledním parametrem je řídkost k, ta nám určuje, kolik hodnot mají obsahovat sloupce (vektory) matice $\mathbf{X}$. Výsledkem je pak natrénovaný slovník $\mathbf{D}$ pro vzorky $\mathbf{Y}$ a matice $\mathbf{X}$, jejíž sloupce obsahují k-řídké vektory.

3.2 Trénování a testování

Pokud bych měl v jedné větě vysvětlit rozdíl mezi trénování a testování, tak trénování chápeme jako analýzu a testování jako syntézu signálu pomocí natrénováného slovníku. Proces trénování slovníku je v podstatě popsán v subkapitole 3.1. Jak již bylo zmíněno, výsledkem je natrénovaný slovník $\mathbf{D}$ pro zadaný signál. Úspěšnost trénování nám určuje forbeinova norma chybové matice $\|\mathbf{E}\|_F^2$, kterou vypočítáváme po procesu natrénování jako $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$. Čím menší je $\|\mathbf{E}\|_F^2$, tím bylo trénování úspěšnější a slovník $\mathbf{D}$ nám umožní lépe aproximovat signál, na který byl natrénovaný. Naopak testování, je proces, kde využijeme námi natrénovaný slovník k apro-



Obr. 3.1: Návrh algoritmu

ximaci (syntéze) jiného signálu než na kterém byl slovník natrénován. Tento signál si označíme jako $\mathbf{Y}_1$. Většinou se snažíme použít podobný zvukový signál, například stejný hudební nástroj, hrající podobný hudební motiv. Zde můžeme předpokládat, že testovaný signál $\mathbf{Y_1}$ je možno aproximovat pomocí natrénovaného slovníku a k-řídkého $\mathbf{X}_1$. K nalezení k-řídkého $\mathbf{X}_1$ nám slouží algoritmus OMP, protože OMP je výpočetně náročné a je použito v každém kroku testování, tak je efektivnější vypočítat $\mathbf{X_1}$ pomocí pseudo inverze. Pseudo inverze nám umožňuje stejně jako OMP najít ortogonální $\mathbf{x}$ k hyperovině slovníku a to však mnohem rychleji, kde velikost chyby vypočítáme pomocí pseudo inverze jako:

$$E = \|\mathbf{Y}_1 - \mathbf{D}\left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{Y}_1\|_2^F \tag{3.1}$$

Efektivnost nám opět ukáže velikost chybové matice. Výsledek je možný porovnat i subjektivně když si syntetizovaný signál $\mathbf{Y}_1$, tedy $\mathbf{DX}_1$ poslechneme. Jestliže nebude poslechový rozdíl znatelný, můžeme konstatovat, že ze subjektivního hlediska trénování i testování proběhlo úspěšně. Analyzovat a syntetizovat signály je také možné pomocí Fourierovy transformace. Protože námi testované signály jsou reálné a spojité, můžeme v matlabu pomocí funkce fft analyzovat signál na spektrum. Pokud seřadíme jednotlivé harmonické podle jejich velikosti (absolutní hodnoty) a k syntéze (zpětná Fourierova transformace) použijeme jen k-největších, tak vzniká k-tá aproximace $\mathbf{Y}_1$. Využijeme-li všechny složky spektra, bude chyba testování nulová. Chybu získáme obdobně jako při testování natrénovaného slovníku, je to rozdíl mezi původmín signálem a syntetizovanou aproximací. Porovnat Fourierův slovník a natrénovaný slovník lze několika způsoby.

První způsob je natrénovat slovník o maximálním počtu atomů (sloupců), kdy rozměr slovníku je stejný jako rozměr matice trénovacích dat. Tento slovník natrénujeme tak, aby se jeho chyba začala ustalovat. Takto natrénovaný slovník použijeme k syntéze signálu $\mathbf{Y}_1$, kdy pomocí OMP najdeme neřídká $\mathbf{x}$. V první iteraci testování pak vybereme podle absolutní hodnoty největší prvek z každého atomu $\mathbf{X}$ a zbytek doplníme nulovými hodnotami, vznikne řídké $\mathbf{X}_i$, které bude obsahovat jen jednu hodnotu. Ve druhé iteraci testování bude $\mathbf{X}_i$ obsahovat dvě hodnoty v každém atomu, takto řídkými $\mathbf{X}_i$ násobíme natrénovaný slovník a zkoumáme chybu. Tento způsob testování popisuje funkce `TrenovaniTestovani1.m`.

Parametrem pro druhý způsob testování je počet atomů natrénovaného slovníku. Tento způsob testování popisuje funkce `TrenovaniTestovani2.m`. Podle počtu sloupců slovníku se odvíjí počet řádků matice $\mathbf{X}_i$, kde v první iteraci je natrénován slovník $\mathbf{D}_i$ o jednom sloupci a matice $\mathbf{X}_i$ o jednom řádku. Proto řídkost trénování nastavujeme na počet řádků matice $\mathbf{X}_i$. Výsledkem trénování je pak slovník o i sloupcích, ke kterému najdeme pomocí pseudo inverze neřídkou matici $\mathbf{X}$ a vypočítáme

chybu. Řídkost v tomto způsobu testování nacházíme v tom, že jsme schopni aproximovat signál pomocí malého počtu atomů slovníku. Proto je tento způsob testování mnohem atraktivnější.

3.3 Ukázka kódů v MATLABu

Hlavní funkce pro trénování a testování jsou `TrenovaniTestovani1.m` a funkce `TrenovaniTestovani2.m`. Tyto funkce si vyvolají další potřebné funkce, které jsou součástí přílohy. Každá z funkcí vyžaduje několik vstupních dat, které jsou popsány v komentářích funkcí. Funkce `TrenovaniTestovani2.m` pak vypadá takto:

```
function [vysledky] = TrenovaniTestovani2 (s,s1,c,n,i,K,I)
%s - trenovaci signal
%s1 - testovaci signal
%c - casovy usek rozsekani signalu
%n - inicializacni slovnik 1 = nahodny cisla, 2 = haromnicke
%i - pocet iteraci trenovani
%K - pocatecni iterace testovani (musi byt mensi nez I)
%I - pocet iteraci testovani
if K>I
   K = I-30;
   warning('Pocateni iterace byla nastavena na %d',K)
end
A = cut(s1,c);
[E_Fourier] = test_fourier (s1,c); %Vypocet chyb Fourierova slovniku
param = parametry(s,c,size(E_Fourier,2),n,0,i); %vytvoreni iniciacni struktury
Dmax = param.initdic; %ulozeni iniciacniho slovniku o max poctu sloupcu
E = zeros (1,I); % prazdny vektor pro ukladani chyb KSVD
E_OMP = zeros (1,I); %Prazdy vektor pro ukladani chyb OMP
E_testself = zeros (1,I);
if I>size(A,2)
   I = size(A,2);
   warning('Maximalni pocet iteraci byl zmenen na %d',I)
end
for k = K:I %size(E_Fourier,2)
     param.initdic =  Dmax(:,1:k);
     param.N = k;
     param.Tdata = k;
     [D,X] = ksvd_upravene(param);
```

```
      E(k) = norm((param.data - (D*X)),'fro');
      E_OMP(k) = norm(A - D*inv(D'*D)*D'*A,'fro');
      E_testself (k) = norm(param.data - D*inv(D'*D)*D'*param.data,'fro');
      if E_OMP(k) < E_Fourier(k)
            disp(sprintf('%d. iterace, K-SVD slovnik je efektivnejsi',k));
       else
             disp(sprintf('%d. iterace, Fourier je efektivnejsi',k));
       end
end
x = 1:size(E_Fourier,2);
x1 = 1:size (E_OMP,2);
plot (x, E_Fourier, x1, E_OMP)
axis ([K I 1 E_OMP(1)])
title('Trenovani a testovani)
xlabel('Pocet iteraci [-]')
ylabel('Velikost chyby [-]')
legend ('Fourieruv slovnik','Natrenovany slovnik')
vysledky.chyba_fourier = E_Fourier;
vysledky.chyba_OMP = E_OMP;
vysledky.D = D;
vysledky.X = X;
vysledky.E_testself = E_testself;
end
```

3.4 Vyhodnocení

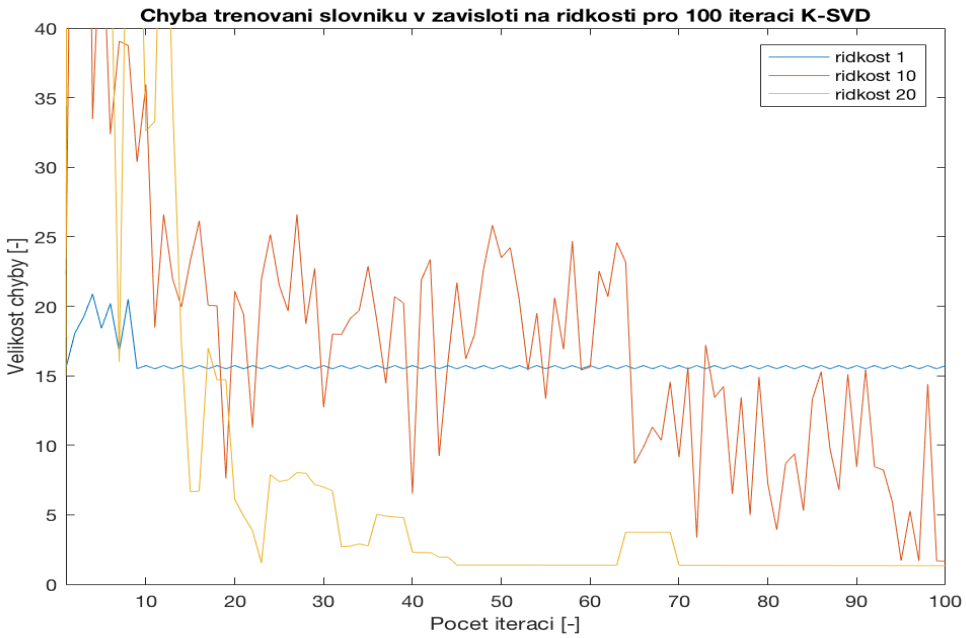
Nejdříve bylo třeba zjistit, jak ovlivňují jednotlivé parametry výstup algortimu K-SVD. Z trénování slovníku jsem zjistil, že na výstupní chybu nemá vliv inicializační slovník. Významný vliv má zvolená řídkost trénování, počet atomů slovníku a počet iterací algoritmu. U správně natrénovaného slovníku se chyba po určité době ustálí. Na grafu 3.2 pozorujeme srovnání chyb natrénovaného slovníku pro baskyatru o dvaceti atomech v závislosti na zvolené řídkosti. Jako inicializační slovník byla použita náhodná čísla a počet iterací K-SVD byl nastaven na 100. Pro řídkost nastavenou na číslo jedna začne chyba konvergovat s chybou patnáct již po desti iteracích K-SVD. Nejpřesnější aproximace vzniká, když je řídkost nastavená alespoň na počet atomů slovníku, tzn. matice $\mathbf{X}$ již není řídká.

První způsob testování viz. 3.2 jsem prováděl na zvuku baskytary na vzorcích `Basa1.wav` a `Basa2.wav` o délce $t = 2$ s a vzorkovací frekvencí $f_{vz} = 48$ kHz. Vstupní

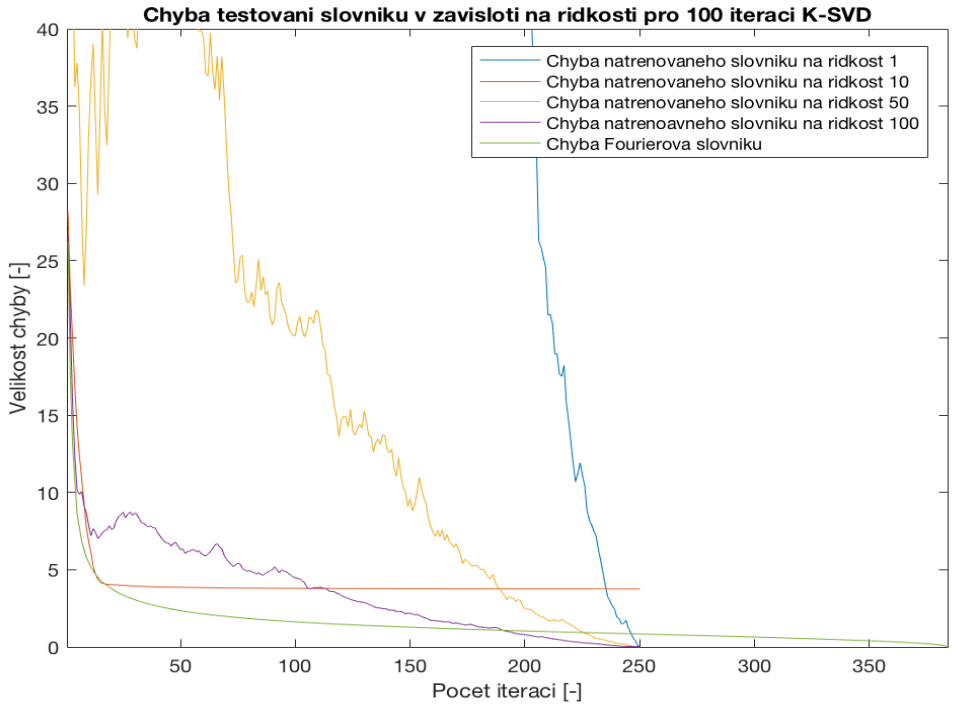
matice testovacích a trénovacích dat vznikla rozkouskováním signálu, kdy jeden sloupec matice obsahoval 0.008 sekund záznamu. Postupně jsem natrénoval čtyři různé slovníky o 250 atomech, s řídkostmi trénování nastavenými na 1,10, 50 a 100, vždy pro 100 iterací K-SVD. Celý proces trénování a testování byl výpočetně náročný. Nejmenší chyba testování byla 0.0018 pro baskytaru a 0.0053 pro piano se slovníky natrénovanými s řídkostí nastavenou na 100. Výsledek zobrazuje graf 3.3. Takto jsem pokračoval i se vzorky piana (Piano1.wav a Piano2.wav), výsledek je vynesen v grafu 3.4.

Druhý způsob testování viz. 3.2 jsem pro srovnání opět prováděl na baskytaře na vzorcích `Basa1.wav` a `Basa2.wav`. Podle očekávání byl natrénovaný slovník učínější již po několika iteracích testovaní, proto je v grafu 3.5 uvedena jen část testování, která má pro nás význam. Předpokládáme, že dále budou obě chyby klesat.

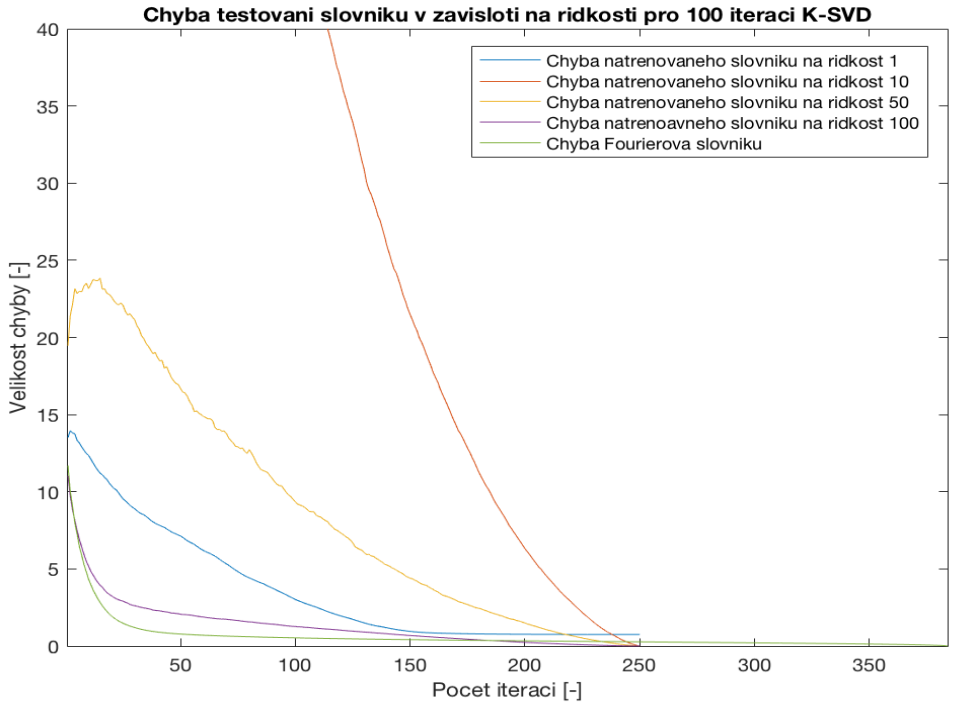
Z obou druhů testování můžeme konstatovat, že natrénovaný slovník je schopný aproximovat signál s větší přesností i při menším počtu iterací než Fourieruv slovník. To však za cenu větší výpočetní náročnosti. Zajímavé bylo druhé testování na zvuku piana na vzorcích `Piano1.wav` a `Piano2.wav`, kde tóny (frekvence), které vzorky obsahují odpovídají tónům ve vzorcích pro baskytaru (`Basa1.wav` a `Basa2.wav`). Testování piana nebylo tak efektivní jako testování baskytary. Lepších výsledků dosáhl natrénovaný slovník až od 70-té iterace. Výsledek zobrazuje graf 3.6.



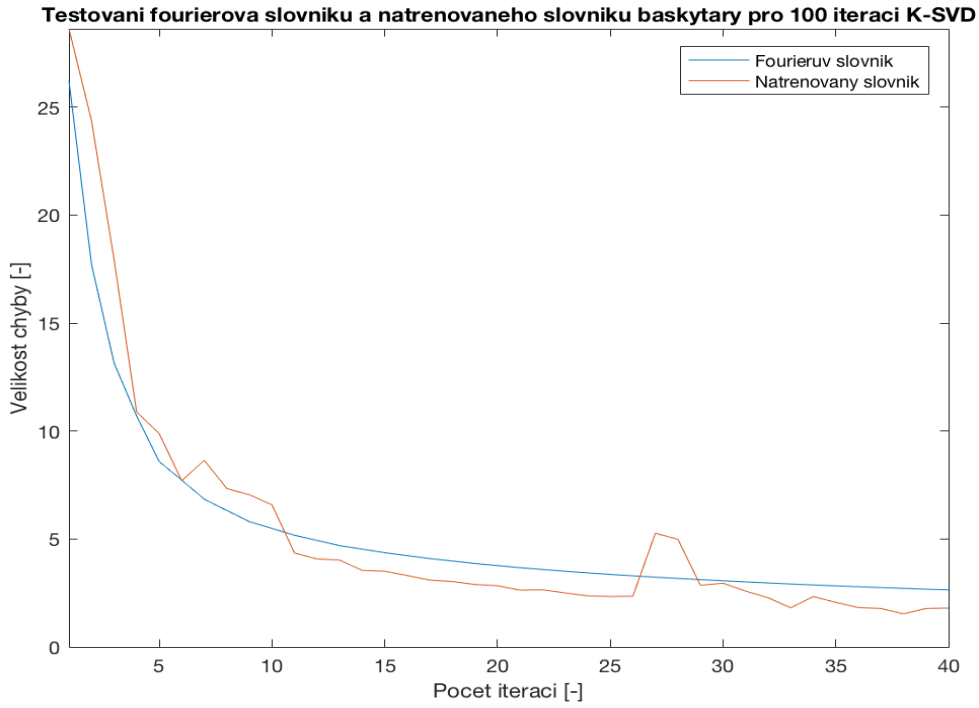
Obr. 3.2: Chyba trénování slovníku.



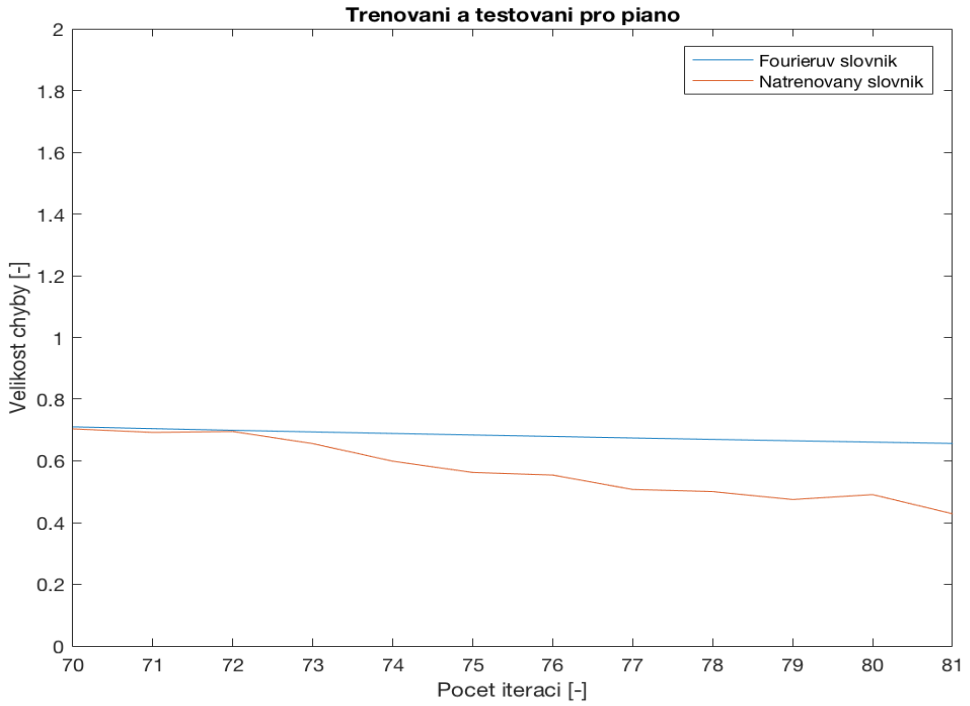
Obr. 3.3: První způsob testování, srovnání Fourierova slovníku a natrénovaného slovníku pomocí K-SVD pro baskytaru s různými řídkostmi trénování.



Obr. 3.4: První způsob testování, srovnání Fourierova slovníku a natrénovaného slovníku pomocí K-SVD pro piano s různými řídkostmi trénování.



Obr. 3.5: Druhý způsob testování, srovnání Fourierova slovníku a natrénovaného slovníku pomocí K-SVD pro baskytaru.



Obr. 3.6: Druhý způsob testování, srovnání Fourierova slovníku a natrénovaného slovníku pomocí K-SVD pro piano.

4 VYTVÁŘENÍ DATABÁZE HUDEBNÍCH ZVUKŮ

Součástí zadání semestrální práce je vytvoření databáze hudebních zvuků různých charakterů (bicí nástroje, harmonické nástroje...). Databáze bude sloužit pro natrénování slovníků a poté jejich testování. Takto získáme své vlastní a jedinečné signály, pro které vytvoříme jedinečné slovníky. Můžeme předpokládat, že například obecně zvuk houslí nese určité znaky, které jsou charakteristické pro všechny housle. Díky tomuto předpokladu by natrénované slovníky mohly mít široké využití v hudebních aplikacích. Například pro analýzu druhu hudebního nástroje nebo přiřazování zvuku na základě podobnosti jinému zvuku.

4.1 Zvuk

Jako zvuk označujeme mechanické vlnění hmotného prostředí zvukového pole. V dusledku pruznosti prostredi se vytvareji pri kmitani castic mista se zhustenim nebo zredenim prostredi postupujici od zdroje kmitani predavanim kineticke energie castic prostredi. Zvuk se šíří prostředím pomocí zvukové vlny. Místa, kam dospěje vlna za stejnou dobu od zdroje zvuku se stejnou fázi, nazýváme vlnoplochou.

4.2 Elektroakustický řetězec

Za elektroakustický řetězec označujeme vše, co vstupuje zvukovému signálu do cesty při záznámu a reprodukci. Počínaje elektroakustickým měničem (mikrofonem), záznamem, zesílením a reprodukcí. Pro danou aplikaci se typy elektroakustických řetězců mohou lišit. Také prostor, ve kterém je zvuk zaznamenáván či reprodukován, ovlivňuje kvalitu zvuku. Protože většina zařízení elektroakustického řetězce je zapojena v sérii, tak kvalita řetězce dosahuje takových kvalit jako je kvalita nejslabšího prvku řetězce. Na elektroakustický řetězec můžeme tedy nahlížet jako na systém s přenosovou funkcí $H(z)$.

4.2.1 Volba elektroakustického řetězce

Jak již bylo řečeno, elektroakustický řetězec má velký vliv na záznam zvuku hudebního nástroje. V oblasti záznamu hudebního zvuku se používají jiné elektroakustické řetězce než například při měření parametrů místností, hudebních nástrojů, spekter atd. Pro záznam zvuku, který bude sloužit pro měřící účely, budeme určitě používat elektroakustický řetězec, který nám bude minimálně ovlivňovat záznam zvuku. Jinými slovy budeme chtít, aby byl přenos takového řetězce konstantní v celém

spektru. Naopak pro záznam hudebního zvuku určený pro umělecké účely nám nevadí využití nelineární přenosové charakteristiky. Většinou je to i žádoucí a volbou elektroakustického řetězce získáváme (hudebníky označovaný) tzv. pěkný zvuk.

4.3 Výběr a vlastnosti prostoru

Rozhodl jsem se jít nejjednodušší cestou, všechny nástroje jsem nahrával ve svém domácím nahrávacím studiu. Zároveň jsem zvolil elektroakustický řetězec, který bude jednoduchý a jeho přenos bude co nejvíce konstantní v celém spektru. Rozměrově se jedná o místnost o velikosti 6m x 3,7m x 2,6m. Jsou zde nainstalovány prvky upravující akustiku místnosti. V rozích místnosti jsou nainstalované basové pasti, vyrobené z akustické pěny.

f [Hz]	125	250	500	1000	2000	4000
α [-]	0,70	0,91	0,65	0,70	0,64	0,78

Tab. 4.1: Tabulka koeficientů útlumu α pro basovou past.

Část zdí je prokryta jehlany z akustické pěny, která má jiné vlastnosti než pěna u basových pastí, proto její vlastnosti můžeme vidět v tabulce. Bohužel nemám dozvuk změřený, ale místnost můžeme označit spíše za akusticky suchou, bez velkého dozvuku.

f [Hz]	125	250	500	1000	2000	4000
α [-]	0,14	0,40	0,75	0,98	0,98	1,00

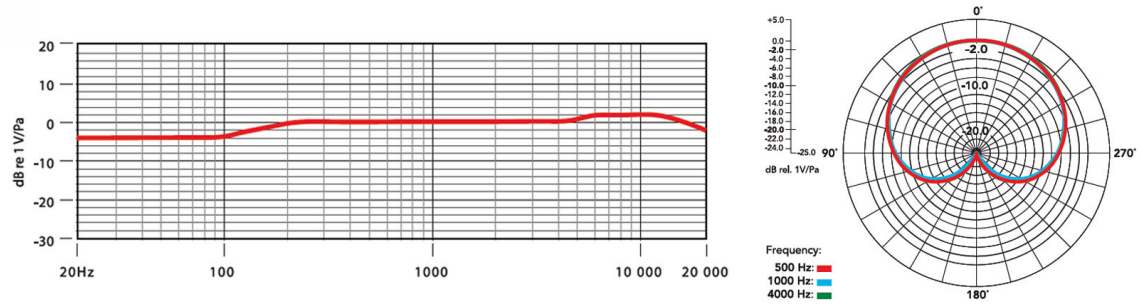
Tab. 4.2: Tabulka koeficientů útlumu α pro jehlany.

4.4 Elektroakustický měnič

Pro záznam jsem se rozhodl zvolit pouze jeden kondenzátorový mikrofon australské značky RODE, model NT5. Jedná se o malomembránový kondenzátorový mikrofon, jehož frekvenční charakteristika je vcelku konstantní. Směrová charakteristika je ledvinová (kardioda), to s sebou nese značné výhody i nevýhody. Výhody jsou například, že potlačuje zvuky přicházející „ze zadu“ a tak snímaný zvuk obsahuje více užitečného signálu než šumu. Nevýhodou je to, že mikrofony s touto směrovou charakteristikou mají velký proximity efekt tzn. velký nárůst basových (nízkých) frekvencí s přibližujícím se zdrojem zvuku. Proto není vhodné nahrávat

Obr. 4.1: Domácí studio

zvuk v bezprostřední blízkosti (jednotky centimetrů). Všechny ostatní parametry jako vlastní šum, dynamický rozsah a maximální akustický tlak SPL jsou uváděné výrobcem a jsou uvedeny v následující tabulce. Nástroje budu tedy snímat pouze jedním vhodně umístěným mikrofonem.



Obr. 4.2: Frekvenční a směrová charakteristika mikrofonu RODE NT5.

4.5 Předzesilovač a A/D převodník

Další zásadní volbou záznamového elektroakustického řetězce je volba mikrofonního předzesilovače a A/D převodníku. Často se v praxi změnou mikrofonního předzesi-

Vlastní šum	16 dB(A)
Maximální akustický tlak	143 dB SPL
Výstupní impedance	100 Ω
Maximální výstupní napětí	13,9 mV
Kapsle	1/2"se zlatem pokrytou membránou

Tab. 4.3: Další důležité parametry mikrofonu RODE NT5 .

lovače dá docílit zcela jiné barvy zvuku. To ale nebude naším záměrem, naším cílem je zvolit předzesilovač s nízkou hodnotou vlastního šumu a jehož přenosová charakteristika bude v celém hudebním spektru konstantní. Proto jsem zvolil předzesilovač přímo zabudovaný v převodníku RME Fireface 800, který má podle výrobce odstup signálu od šumu větší jak 110 dB a frekvenční rozsah 5 Hz - 200 kHz.

4.6 Obsah záznamu

Pro jednoduchost jsem se rozhodl od každého melodického nástroje, který mi byl k dispozici a který je schopný vytvořit tón s určitou frekvencí zaznamenat část chromatické stupnice od tónu A (110Hz) nebo od tónu a (220Hz). U bicích nemelodických nástrojů jsem vybral jednoduchý rytmický patern. Nejdříve jsem zaznamenával celé chromatické stupnice, ale protože jedna sekunda záznamu se vzorkovací frekvencí 48kHz obsahuje 48 000 hodnot a trénování slovníků je výpočetně náročné, tak jsem volil signály, jejichž délka nepřesahovala 2 až 4 sekundy záznamu. Tyto signály tedy neobsahují celou chromatickou stupnici ale jen její část. Každý zvukový vzorek jsem musel zaznamenat dvakrát, jeden slouží pro trénování slovníku a druhý pro testování.

5 ZÁVĚR

V bakalářské práci bylo úkolem nastudovat a popsat algoritmy z oblasti strojového učení, ktere signalovy slovnik hledaji tak, aby byl prizpusobeny nejakemu typu signalu a porovnat je jak objektivně tak subjektivně s použitím malé databáze zvuků. Z mnoha používaných algoritmů byl použit a popsán algoritmus K-SVD a hladový algoritmus OMP (Orthogonal Matching Pursuit). V první kapitole bakalářské práce je vysvětleno základní značení a definice, které slouží pro lepší pochopení následujících kapitol. Popisu jednotlivých slovníků se věnuje druhá kapitola, nejdříve jsou zde rozebrány slovníky statické. Je zde popsán slovník DCT a Gaborův. Více usílí jsem věnoval pochopení a popsání přizpůsobených slovníků a Fourierovy transformace. Toto téma vidím jako téma stěžejní v této práci. Kapitola s číslem tři hovoří o práci v programu MATLAB, popisuje blokové schéma algoritmu K-SVD a ukázky kódů v MATLABu, správnou volbu parametrů jako je řídkost, vliv inicializačního slovníku a počet atomů slovníku pro natrénování přizpůsobených slovníků. Dále je zde popsáno, jak je možné otestovat tyto natrénované slovníky a jak efektivní jsou. V poslední kapitole je popsáno, jak jsem vytvářel databázi zvuků.

Srovnání Fourierova slovníku a natrénovaného slovníku pomocí algoritmu K-SVD ukázalo, že ve většině případech má smysl slovník natrénovat. Nevýhoda K-SVD je výpočetní náročnost. Pokud bychom v praxi obdrželi již natrénovaný slovník, ve většině případech by byl schopný aproximovat signál s mešní chybou a s menším počtem kroků než Fourieruv slovník. Výhoda Fourierovy transformace je, že ve většině matematických programech je již implementována a její výpočetní čas je velmi krátký.

Je zajímavé, že pro zvuk baskytary byl slovník velice efektivní, ale pro zvuk klavíru hrající tóny o stejné frekvenci jako baskytara už byla efektivita mnohem menší pouze při jednom způsobu testování. Pro kytaru, bicí sadu, housle a trumpetu také nebyl druhý způsob testování K-SVD slovníku tak efektivní jako při testování na zvuku baskytary. Při subjektivním porovnávání jsem nebyl schopen poznat rozdíl mezi originálním signálem a syntetizovaným signálem s odchylkou menší jak 0,5.

LITERATURA

[1] ELAD, M. Sparse and redundant representations: from theory to applications in signal and image processing. New York: Springer, c2010. ISBN 9781441970114.

[2] RAJMIC, Pavel. Řídké a nízkohodnostní reprezentace signálů s aplikacemi. Brno: VUTIUM, 2015. Vědecké spisy Vysokého učení technického v Brně. Habilitační a inaugurační spisy. ISBN 978-80-214-5195-7

[3] HRBACEK, R., RAJMIC P., VESELY V. a SPIRIK J.: *Ridke reprezen- tace signalu: uvod do problematiky.* Elektrorevue. 2011, c. 50. ISSN 1213 - 1539. Dostupne URL: <http://elektrorevue.cz/cz/clanky/zpracovani-signalu/0/ridke-reprezentace-signalu--uvod-do-problematiky/>.

[4] NAVRÁTILOVÁ, Barbora. *Aplikace řídkých reprezentací dat.* Brno, 2014. Diplomová práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Vedoucí práce Pavel Rajmic.

[5] ADLER, Amir, et al. *Audio Inpainting* [online]. 2011. Dostupne URL: <http://hal.inria.fr/docs/00/57/70/79/PDF/RR-7571.pdf/>..

[6] SPIRIK, J., RAJMIC a VESELY. *Reprezentace signalu: od bazi k framum.* Elek- trorevue - Internetovy casopis [online]. 2010, c. 111 [cit. 2012-03-15]. ISSN 1213 - 1539. Dostupne z URL: <http://elektrorevue.cz/cz/clanky/zpracovani-signalu/0/ridke-reprezentace-signalu--komprimovane-snimani/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A/D	Převodník analogového signálu na digitální
OMP	Hladový algoritmus Orthogonal Matching Pursuit
K-SVD	Algoritmus využívaný k učení slovníku
MOD	Algoritmus využívaný k učení slovníku - method of directions
DCT	Diskrétní cosinova transformace
SVD	Část algoritmu K-SVD
SPL	Sound Pressure Level

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Na přiloženém CD naleznete elektronickou verzi bakalářské práze ve formě PDF. Dále se zde nachází dvě složky. První s názvem `Zvuky 48kHz` obsahuje databázi pro trénování a testování slovníků. Druhá složka s názvem `Zdrojové kódy` obsahuje všechny zdrojové kódy vytvořené v programu MATLAB 2016a, které jsou potřebné pro spuštění hlavních funkcí `TrenovaniTestovani1.m` a `TrenovaniTestovani1.m`. Pro snadné pochopení jsou všechny zdrojové kódy řádně okomentované.