VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZEZTRÁTOVÉ KÓDOVÁNÍ ŘEČI Z MIKROFONNÍHO POLE

LOSSLESS CODING OF SPEECH FROM MICROPHONE ARRAY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID MYŠKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR MALENOVSKÝ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Myška David**
Program: Informační technologie
Název: **Bezeztrátové kódování řeči z mikrofonního pole**
Lossless Coding of Speech from Microphone Array
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se s principy bezztrátového kódování řeči, zejména pak kodeku FLAC, analyzujte jeho zdrojový kód.
2. Seznamte se s principy mikrofonních polí a obstarejte si vícekanálové nahrávky řeči z mikrofonního pole.
3. Navrhněte techniku pro bezeztrátové kódování vícekanálového řečového signálu pro mikrofonní pole a implementujte ji.
4. Vyhodnoťte kompresní poměr a rychlost kódování a navrhněte, jak je zlepšit.
5. Nejméně jedno zlepšení implementujte a vyhodnoťte.
6. Konsolidujte váš kód do formy knihovny nebo binárního nástroje, který bude možno použít z příkazové řádky. Váš kód řádně dokumentujte.

Literatura:

1. Dokumentace kodeku FLAC na https://xiph.org/flac/documentation.html
2. K. Kumatani, J. McDonough, and B. Raj. "Microphone array processing for distant speech recognition: From close-talking microphones to far-field sensors.", Signal Processing Magazine, IEEE, 29(6):127-140, 2012. https://ieeexplore.ieee.org/abstract/document/6296525
3. Další zdroje dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

1. Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Malenovský Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 9. května 2019

Abstrakt

Tato bakalářská práce se zabývá bezeztrátovým kódování řečových signálů z mikrofonních polích. Uvádí popis metod použitých v referenčním kodeku FLAC a jejich obměny pro zvýšení komprese pro signály z mikrofonních polí. Dále uvádí popis metod pro zarovnání kanálů pro jejich následné zpracování. Na konci práce jsou zhodnoceny dosažené výsledky v porovnání s referenčním kodekem FLAC.

Abstract

This bachelor‘s thesis deals with lossless coding of speech signals from microphone arrays. It describes the methods used in the FLAC reference codec and their variations to increase compression for signals from microphone arrays. The following describes methods for aligning channels for their subsequent processing. At the end of the work the results obtained are compared with the reference codec FLAC.

Klíčová slova

Bezeztrátové kódování, mikrofonní pole, normalizovaná křížová korelace, TDOA, FLAC, lineární predikce, Riceovo kódování

Keywords

Lossless coding, microphone array, normalized cross-correlation, TDOA, FLAC, linear prediction, Rice coding

Citace

MYŠKA, David. *Bezeztrátové kódování řeči z mikrofonního pole*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Malenovský, Ph.D.

Bezeztrátové kódování řeči z mikrofonního pole

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Malenovského, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .

David Myška

15. května 2019

Poděkování

Děkuji svému vedoucímu práce Ing. Vladimíru Malenovskému, Ph.D. za odbornou pomoc při vypracovávání této práce a za užitečné rady, jenž mi dal. Dále bych chtěl poděkovat všem, kteří mě při mé práci podporovali.

Obsah

Kapitola 1

Úvod

Bezeztrátové kódování, nebo také bezeztrátová komprese je metoda umožňující komprimaci dat s možností přesné rekonstrukce dat do původní podoby bez jakýchkoliv odchylek. Bezeztrátové kódování je většinou specializované na určitý druh dat, například obrázky, zvuk nebo prostý text, můžeme se však setkat i s univerzálními metodami, jenž komprimují různé druhy dat stejně efektivně. Tato práce se ale zaměřuje pouze na bezeztrátovou kompresi zvuku, navíc ještě zvuku, jenž pochází z mikrofonního pole. Mikrofonní pole je soustava více mikrofonů, rozmístěných do určitého vzoru, například na jedné přímce, či do kruhu nebo čtverce. Signály pocházející z mikrofonního pole jsou pak velice podobné, neboť všechny mikrofony zachytávají téměř stejné zvukové vlny, většinou jen trochu zpožděné. Této vlastnosti pak využívá kodek, jenž je výstupem této práce.

Pro srovnání se zde bude jako referenční nástroj používat kodek FLAC neboli „Free Lossless Audio Codec“, jenž také umožňuje bezeztrátovou kompresi zvuku. Na rozdíl od nástroje, jenž je výstupem této práce, není specializovaný na data pocházející z mikrofonního pole. FLAC jednotlivé kanály takových dat zpracovává zcela nezávisle, tj. nehledá mezi jednotlivými kanály dat jakékoliv podobnosti. Knihovna kodeku FLAC je v této práci také použita jako základ pro kódování a dekódování, pouze s několika, místy i rozsáhlými, úpravami kódu.

Kapitola druhá je věnována mikrofonním polím, kodeku FLAC, a i rozboru některých jeho metod, jako například lineární predikce, která slouží k odhadu vzorků predikovaného signálu z lineární kombinace předchozích vzorků s co nejmenší chybou oproti skutečnému signálu. Další metoda používaná kodekem FLAC je Riceovo kódování, což je bezeztrátová kompresní metoda generující komprimovaná data s proměnlivou délkou jednotlivých hodnot.

Kapitola třetí krátce popisuje vzorová data, která pochází z mikrofonního pole, jež mi byla k mé práci poskytnuta.

Kapitola čtvrtá se věnuje implementaci jednotlivých dílčích metod včetně popisu uživatelského rozhraní výsledné konzolové aplikace pro systémy Linux.

Dosaženým výsledkům a jejich srovnání s referenčním kodekem FLAC se pak věnuje kapitola pátá.

Kapitola 2

Teoretická část

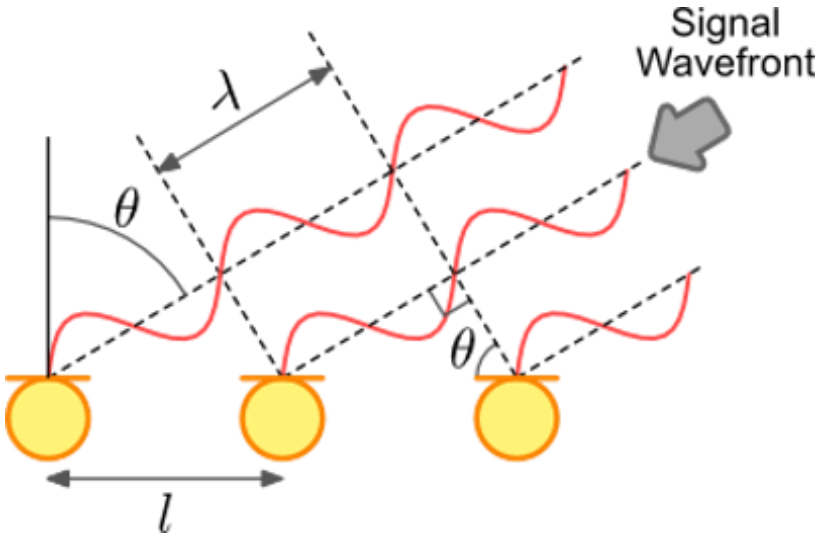
2.1 Mikrofonní pole

Mikrofonní pole je soustava mikrofonů rozmístěných do určitého vzoru. Rozlišujeme celkem tři typy mikrofonních polí:

- Lineární mikrofonní pole
- Rovinné mikrofonní pole
- Prostorové mikrofonní pole

Lineární mikrofonní pole si můžeme představit jako řadu mikrofonů umístěných na jedné přímce s konstantními rozestupy mezi sousedními mikrofony. V ideálním případě jsou pak zpoždění mezi sousedními mikrofony také konstantní. U rovinného mikrofonního pole jsou jednotlivé mikrofony rozmístěny na jedné rovině, například na stole nebo na nějaké desce. Vzorů, kterých takové pole může nabývat je už více, kruh, čtverce či větší mřížka mikrofonů. Co do možností vzorů je prostorové mikrofonní pole asi nejbohatší, neboť může nabývat opravdu libovolného prostorového tvaru, například krychle.

Pro jednoduchost se dále budeme věnovat pouze lineárním mikrofonním polím.



Obrázek 2.1: Lineární mikrofonní pole, s konstantními rozestupem mezi sousedními mikrofony l. Zvukové vlny přicházejí pod úhlem θ, zpoždění mezi sousedními mikrofony je λ. Převzato z http://p214.info/grating-lobes.html.

Signál dorazí pod úhlem θ jako první k mikrofonu vpravo, k prostřednímu mikrofonu dorazí signál o λ později než na první mikrofon a o 2λ později k mikrofonu nalevo. Zpoždění λ můžeme vypočítat následovně:

$$\lambda = \frac{l * \sin\theta}{v} \tag{2.1}$$

Kde v je rychlost zvuku (v je 346,3 m/s ve vzduchu při teplotě 25 °C).

Většinou však nemáme k dispozici úhel, pod kterým zvuková vlna přichází, ale naopak jsme schopni vypočítat zpoždění mezi mikrofony ze samotných zvukových signálů a z tohoto zpoždění vypočítat úhel θ. Pokud však nepoužijeme prostorové mikrofonní pole, pak nejsme schopni vypočítat přesnou polohu zdroje zvuku, například u rovinného mikrofonního pole nemůžeme poznat, zda zvuk přichází z přední strany nebo ze zadní. U lineárního pole vzniká dokonce celá kružnice možných poloh zdroje zvuku.

2.2 Free Lossless Audio Codec

Free Lossless Audio Codec [4] neboli FLAC je bezeztrátový audio kodek, podporující také streamování či archivaci dat. Vstupní signály jsou zpracovávány po rámcích. Signál se nejprve rozdělí na úseky signálu s konstantní délkou. Kromě posledního rámce ten bývá typicky kratší, až tyto rámce jsou dále zpracovány. Pro streamování je nezbytné, aby jednotlivé rámce dat byly pro dekódování nezávislé na následujících rámcích. Všechny potřebné informace pro dekódování musí být tedy součástí aktuálního rámce, popřípadě předchozího rámce. To ovšem FLAC splňuje a využívá při tom pouze aktuální rámec nikoliv i předchozí, je tedy plně nezávislý na okolních rámcích. Pro streamování je také vhodné, aby dekódování probíhalo rychle, FLAC při dekódování používá převážně celočíselnou aritmetiku napomáhající k rychlosti dekodéru. Kodér je pak pomalejší, neboť je mnohem složitější nežli dekodér a provádí více výpočtů s desetinnými čísly. Archivace vyžaduje, aby data bylo možné zpětně dekódovat do původní podoby bez jakýchkoliv odchylek, tj. aby vše bylo bezeztrátové. Kodek FLAC je distribuován zcela zdarma včetně zdrojových kódů pod open-source licencí. Nabízí četnou podporu mnoha platforem a operačních systémů, například Windows, Mac OS X, či Linux nebo BSD.

Pro svoji činnost využívá převážně lineární predikce a Riceova kódování. Při zpracování rámce pak vzniká ještě další dělení, a to na jednotlivé kanály, neboť každý kanál v jednom rámci může být zpracován odlišným způsobem. Při zpracovávání signálů má na výběr z více možných metod, jednou z nich je právě zmíněná lineární predikce a Riceovo kódování, kombinací těchto dvou metod vzniká velmi často nejvyšší komprese dat. Lineární predikce vytvoří reziduální signál, jenž má mnohem nižší energii než původní signál, toho pak využívá Riceovo kódování, které takový signál zakóduje mnohem úsporněji. Další možnosti jsou již méně časté, velmi okrajovým případem jsou pak konstantní rámce, kde všechny vzorky v jednom rámci mají tutéž hodnotu. Zde pro zakódování stačí onu hodnotu zapsat pouze jednou a k ní přidat počet kolikrát se opakuje. Kódování konstantních rámců je tedy velmi úsporné, avšak nastává jen výjimečně. Asi nejméně úspornou možností, jež se použije jen v případě kdy všechny ostatní metody nelze aplikovat nebo jsou zakázané, je zanechání signálu v původní podobě. Stále zde však velmi často vzniká úspora, a to tím, že se pro zápis hodnot použije jen nezbytné množství bitů. FLAC je schopen odstranit přebytečné bity vzorků nejen z nejvyšších bitů, ale také z těch nejnižších, avšak aby tak mohl učinit tak stejné množství bitů musí být přebytečné na všech vzorcích v rámci.

Velikost rámce lze nastavit pomocí parametru, či přímo v programu, FLAC nabízí celkem devět kompresních úrovní, čím vyšší úroveň, tím vyšší komprese, avšak také delší doba výpočtů. FLAC má také přednastavené výchozí velikosti rámců, jenž se použijí, pokud není velikost rámce zadána jiným způsobem. Použije-li se kompresní úroveň 0–2, pak je výchozí velikost rámce 1152 vzorků, v každém kanálu rámce. Velikost rámce 1152 pak odpovídá 72ms při vzorkovací frekvenci 16 kHz, jenž je použita například ve vzorových datech. Je-li kompresní úroveň vyšší neboli 3–8, pak se použije velikost 4096 vzorků na každý kanál. Tato velikost rámce pak odpovídá 256ms při stejné vzorkovací frekvenci 16 kHz. Hlavním rozdílem mezi těmito dvěma skupinami kompresních úrovní je to, zda se používá variabilní lineární predikce či nikoliv, při úrovních 0–2 se tato varianta lineární predikce nepoužívá (obě varianty lineární predikce budou popsány později). Zvolená kompresní úroveň pak ovlivňuje ještě několik výchozích hodnot různých parametrů, avšak všechny je možné manuálně přenastavit na požadovanou hodnotu. Objevuje se zde například omezení maximálního

řádu variabilní lineární predikce. I samotná kompresní úroveň je předem přednastavená na výchozí hodnotu 5.

Kompresní úroveň	Komprese		Doba trvání	
	2 kanály	8 kanálů	2 kanály	8 kanálů
0	37,20 %	37,18 %	27,492 s	1 min 48,732 s
1	36,85 %	37,18 %	28,240 s	1 min 48,884 s
2	36,76 %	37,18 %	31,604 s	1 min 48,916 s
3	35,57 %	35,54 %	34,636 s	2 min 18,068 s
4	35,06 %	35,42 %	39,824 s	2 min 23,196 s
5	35,03 %	35,42 %	48,836 s	2 min 23,464 s
6	35,00 %	35,41 %	1 min 14,732 s	3 min 16,576 s
7	34,97 %	35,36 %	1 min 20,356 s	3 min 28,556 s
8	34,94 %	35,35 %	1 min 56,168 s	4 min 42,484 s

Tabulka 2.1: Výsledky z experimentu s různou úrovní komprese nad vzorovými daty pro dva a osm kanálů. Je zde uvedena výsledná komprese oproti původní velikosti souborů a doba, jak dlouho samotná komprese trvala (včetně času potřebného pro načtení dat). Velikost rámce odpovídá 25ms, experiment byl proveden na mém notebooku.

2.2.1 Rozdělení kodeku FLAC na hlavní části

Kódování v kodeku FLAC můžeme rozdělit na čtyři hlavní části:

1. Rozdělení na rámce
2. Zpracování
   a) Doslovné zpracování
   b) Konstantní zpracování
   c) Lineární predikce
3. Reziduální kódování
4. Riceovo kódování

Rozdělení na rámce
Rozdělení dat na rámce je velmi jednoduché. Vstupní nezkomprimovaná data se rozdělí na úseky signálu o zadané délce rámce. Po zpracování všech kanálů jednoho rámce se začnou zpracovávat vzorky signálu nacházející se za právě zpracovaným rámcem, vezme se opět počet vzorků odpovídající velikosti rámce a data se zpracují jeden dílčí rámec po druhém. Dílčí rámec pak znamená jeden kanál v aktuálním rámci, každý kanál tedy odpovídá samostatnému dílčímu rámci. V kodeku FLAC se pak používá termín podrámec, ten však má standartně význam kratšího úseku rámce, proto je tedy v této práci zaveden termín dílčí rámec. Výjimku tvoří poslední rámec, jehož délka je typicky kratší než velikost ostatních rámců, jelikož na konci signálu nemusí být přesný počet vzorků odpovídající velikosti rámce.

Zpracování
Po rozdělení dat na rámce jsou signály zpracovány, FLAC nabízí celkem čtyři možnosti zpracování dílčích rámců: doslovné, konstantní, fixní lineární predikci a variabilní lineární predikci. Metoda, jenž FLAC použije k zakódování dílčího rámce se vybere automaticky, na základě odhadu potřebného místa pro uložení zkomprimovaného dílčího rámce včetně jeho nezbytných metadat. Zpracování pak probíhá na každém kanálu rámce zcela samostatně, každý dílčí rámec tedy může být zpracován odlišnou metodou než ostatní kanály v aktuálním rámci.

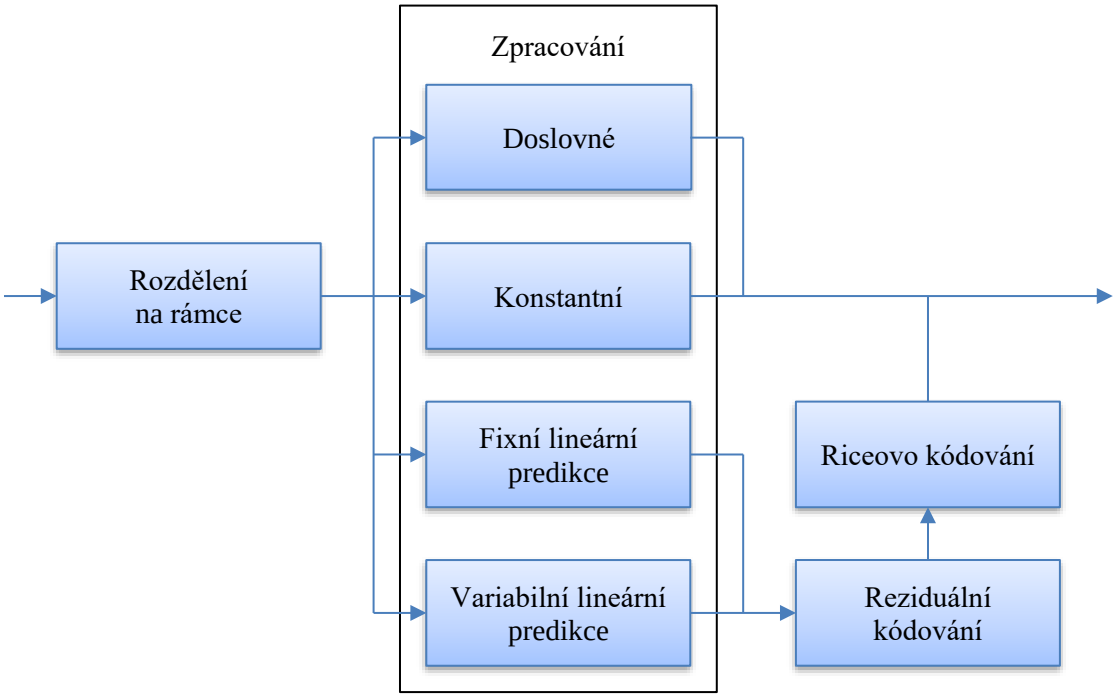
Doslovné zpracování
Doslovné zpracování dílčího rámce znamená, že data jsou přenesena beze změny. Pokud je to možné pak se odstraní přebytečné nulové bity ze všech vzorků rámce tohoto kanálu, a to nejen z nejvyšších bitů, ale také z nejnižších. Bity však může odstranit z jedné či z druhé strany jen v případě, že všechny vzorky tohoto rámce a tohoto kanálu mají všechny tyto bity nulové. Aby FLAC zjistil, které bity se mohou odebrat a které musí nechat, provede bitový OR mezi všemi vzorky dílčího rámce. Posčítáním nulových bitů získaného výsledku z jedné či z druhé strany zjistí počet bitů, které může z obou stran odebrat. Nejnižší bity odebírá pomocí bitového posunu jednotlivých vzorků, do výstupního zkomprimovaného souboru, mezi metadata daného dílčího rámce, se pak jen zapíše o kolik bitů vzorky posunul. Nejvyšší bity pak odebere jen tím, že sníží hodnotu obsahující počet bitů na jeden vzorek, tato hodnota se poté také zapisuje do metadat do výstupního souboru spolu se všemi vzorky, jenž uloží do datové části, o právě tomto počtu bitů. Doslovné zpracování dílčích rámců je ze všech metod nejméně úsporné, avšak díky odebrání případných nepotřebných bitů zde k jisté úspoře také dojde. Používá se však jen výjimečně, a to především v případech, kdy signál obsahuje náhodné vzorky, s nimiž si ostatní metody nedokáží efektivně poradit, nebo pokud jsou ostatní metody zakázány či je nelze aplikovat.

Konstantní zpracování
Naopak nejvíce úsporné je konstantní zpracování, to je však také nejvíce specializované, neboť se používá jen v případech, kdy všechny vzorky v dílčím rámci mají tutéž hodnotu. Tato metoda se tedy používá ještě méně často nežli doslovné zpracování. Vysoká úspora pak vzniká tím, že tuto hodnotu zapíše pomocí run-length kódování [8], což znamená, že tuto hodnotu zapíše pouze jednou následovanou počtem kolikrát se tato hodnota opakuje. V případě kodeku FLAC se však počet opakování hodnoty nachází již v hlavičce rámce, kde tato hodnota představuje velikost rámce, datová část tohoto dílčího rámce už pak obsahuje pouze jedinou hodnotu.

Lineární predikce
Nejčastěji se dílčí rámce zpracovávají pomocí jedné z metod lineární predikce, obě metody lineární predikce používají podobný princip zpracování dílčího rámce. Pro většinu dílčích rámců se pak zvolí variabilní lineární predikce, neboť ta je schopná dosáhnout mnohem přesnějších výsledků nežli fixní predikce. V případě fixní lineární predikce se používají pevně stanovené koeficienty pro predikci, FLAC pak pouze vybírá nejvhodnější řád predikce, u fixní lineární predikce má k dispozici celkem pět řádů, jenž může použít, zvolený řád se pak zapíše do hlavičky dílčího rámce. Variabilní lineární predikce má pouze omezený maximální řád predikce, ten je však vyšší než u fixní predikce. Maximální řád pak určuje zvolená kompresní úroveň, ale je opět možné nastavit maximální řád ručně i na vyšší hodnotu, než jakou nastavuje jakákoliv kompresní úroveň, a to až do hodnoty 32. Při použití kompresní úrovně 0–2 je pak variabilní lineární predikce vypnuta, je to však způsobeno pouze nastavením maximálního řádu na nulu. Koeficienty pro predikci pak nejsou pevně nastavené, ale vypočítávají se pro každý kanál rámec samostatně. Vypočítané koeficienty a zvolený řád jsou pak také zapsány do hlavičky dílčího rámce. Výstupem obou variant lineární predikce je pak predikovaný signál, jenž je podobný původnímu signálu dílčího rámce. Tento predikovaný signál se pak předává k dalšímu zpracování pomocí reziduálního kódování.

Reziduální kódování
Následující reziduální kódování se pak používá jen pokud byl dílčí rámec zpracován pomocí jedné z metod lineární predikce, pro doslovné a konstantní zpracování se vůbec nepoužívá. Reziduální kódování je jen mezikrok mezi lineární predikcí a Riceovým kódováním. V kódu kodeku FLAC je přímo součástí lineární predikce, tudíž skutečným výstupem lineárním predikce podle kódu není predikovaný signál, ale již jeho reziduální signál, zde je však popsán jako samostatný prvek. Výstupem reziduálního kódování je tedy již zmíněný reziduální signál vzniklý rozdílem původního signálu dílčího rámce a predikovaného signálu z lineární predikce. K zakódování reziduálního signálu je pak zapotřebí mnohem méně bitů, než při přímém zakódování původního či predikovaného signálu, neboť při přesnější predikci má reziduální signál mnohem nižší dynamiku signálu.

Riceovo kódování
Poslední z hlavních částí je Riceovo kódování, jimž je reziduální signál zakódován do posloupnosti bitů, jenž se ukládají do výstupního zkomprimovaného souboru, tato část se tedy opět uplatňuje pouze pro lineární predikci. Riceovo kódování pak rozdělí reziduální signál na několik stejně velkých oddílů, počet oddílů je pak určen tak, aby jejich počet byl mocninou dvou a zároveň aby každý oddíl obsahoval stejně dlouhý reziduální signál. Na rozdíl od velikosti rámců zde nemůže být poslední oddíl kratší než ostatní. V každém oddíle se pak určuje parametr Riceova kódování M zcela nezávisle na ostatních oddílech.



Obrázek 2.2: Blokové schéma kodéru v kodeku FLAC včetně směru toku dat při různých metodách zpracování dílčích rámců.

2.2.2 Lineární predikce

Tato podkapitola byla převzata z
http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf

Lineární predikce neboli zkráceně LP je matematická operace sloužící k odhadu následujícího vzorku diskrétního signálu $\tilde{s}(n)$ z lineární kombinace předchozích vzorků s co nejmenší chybou $e(n)$ oproti skutečnému signálu $s(n)$.

Signál $\tilde{s}(n)$ je definován jako:

$$\tilde{s}(n) = -\sum_{i=1}^{P} a_i s(n-i) \tag{2.2}$$

Kde P je počet předchozích vzorků, nebo také řád lineární predikce.

Hledáme takové parametry a_i aby energie chyby $e(n)$ byla co nejmenší. Chyba $e(n)$ je definována jako:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \left(-\sum_{i=1}^{P} a_i s(n-i) \right) = s(n) + \sum_{i=1}^{P} a_i s(n-i) \tag{2.3}$$

Energie chyby $e(n)$ je dána:

$$E = \sum_{n=0}^{N+P-1} e^2(n) \tag{2.4}$$

Kde N je celkový počet vzorků daného rámce. Hodnoty $s(n)$ a $\tilde{s}(n)$ jsou pro $n > N-1$ nulové.

Pro nalezení minima se tato rovnice musí parciálně derivovat podle všech a_i. Derivace položíme rovny nule:

$$\frac{\delta}{\delta a_j} \left\{ \sum_{n=0}^{N+P-1} \left(s(n) + \sum_{i=1}^{P} a_i s(n-i) \right)^2 \right\} = 0 \qquad pro\ 1 \le j \le P \tag{2.5}$$

Úpravou dostaneme:

$$\sum_{i=1}^{P} a_i \sum_{n=0}^{N+P-1} s(n-i)s(n-j) = -\sum_{n=0}^{N+P-1} s(n)s(n-j) \tag{2.6}$$

Označíme-li:

$$\sum_{n=0}^{N+P-1} s(n-i)s(n-j) = \phi(i,j) \tag{2.7}$$

Pak dostaneme soustavu lineárních rovnic:

$$\begin{aligned} \phi(1,1)a_1 + \phi(2,1)a_2 + \cdots + \phi(P,1)a_p &= -\phi(0,1) \\ \phi(1,2)a_1 + \phi(2,2)a_2 + \cdots + \phi(P,2)a_p &= -\phi(0,2) \\ &\vdots \\ \phi(1,P)a_1 + \phi(2,P)a_2 + \cdots + \phi(P,P)a_p &= -\phi(0,P) \end{aligned} \tag{2.8}$$

Použijeme-li pro výpočet $\phi(i,j)$ korelační metodu, která vzorky $n < 0$ a $n > N-1$ považuje za nulové, pak jsou splněny rovnosti $\phi(i,j) = \phi(i+C, j+C)$ a $\phi(i,j) = \phi(j,i)$, tudíž můžeme zjednodušit značení na $R(|i-j|) = \phi(i,j)$, vznikne pak výsledná soustava rovnic:

$$\begin{aligned} R(0)a_1 + \quad R(1)a_2 + \cdots + R(P-1)a_P &= -R(1) \\ R(1)a_1 + \quad R(0)a_2 + \cdots + R(P-2)a_P &= -R(2) \\ &\vdots \\ R(P-1)a_1 + R(P-2)a_2 + \cdots + \quad R(0)a_P &= -R(P) \end{aligned} \tag{2.9}$$

Energii chyby predikce pak můžeme vyjádřit i jako:

$$E = \sum_{n=0}^{N+P-1} e^2(n) = R(0) + \sum_{i=1}^{P} a_i R(i) \tag{2.10}$$

Vyřešením výsledné soustavy rovnic, například metodou Levinson-Durbin [2], dostaneme optimální parametry a_i pro daný rámec. Uvažujeme-li o fixní lineární predikci pak jsou koeficienty a_i předem pevně nastaveny a není tedy nutné je počítat, tudíž vybíráme pouze řád predikce.

2.2.3 Kvantizace koeficientů lineární predikce

Pro správné dekódování zkomprimovaných dat je nutné do nich uložit i samotné LP koeficienty a_n. Pomocí kvantizace však můžeme dosáhnout vyšší komprese těchto hodnot a tím ušetřit další bity z komprimovaných dat.

V kodeku FLAC je použita skalární kvantizace, přenášející chybu v zaokrouhlování do výpočtu následujícího koeficientu. Kvantované koeficienty jsou pak vyjádřeny stanoveným počtem bitů, tato hodnota pak odpovídá hodnotě původního koeficientu a_n posunutého o předem vypočítaný počet bitů s a následně zaokrouhleného na celé číslo.

Přesnost, nebo také počet bitů, na které budou vyjádřeny kvantované koeficienty, se pak odvíjejí od velikosti rámce, programově nastavené, či získané podle zvolené kompresní úrovně.

Kvantované koeficienty q_n získáme následovně:

$$q_n = round(e(n-1) + a_n * 2^s) \qquad pro\ 1 \leq n \leq P \tag{2.11}$$

Kde P je řád variabilní lineární predikce, parametrem n se pak prochází přes všechny LP koeficienty a_n, s je již zmíněný posun bitů a $e(n-1)$ je chyba v zaokrouhlování z výpočtu předchozího koeficientu. Funkce $round()$ pak představuje zaokrouhlení na celé číslo.

Během kvantizace však vzniká chyba v zaokrouhlování $e(n)$, pro její snížená se pak započítává i do kvantizace následujícího koeficientu, počáteční chyba pro kvantování prvního koeficientu je nulová.

$$e(n) = \begin{cases} 0 & pro\ n = 0 \\ e(n-1) + a_n * 2^s - q_n & pro\ 1 \leq n \leq P \end{cases} \tag{2.12}$$

Hodnota s pak představuje bitový posuv, získáme jej následně:

$$s = p - \log_2 \left(\max_{1 \leq n \leq P} |a_n| \right) - 2 \tag{2.13}$$

Kde p je dříve zmíněný počet bitů neboli přesnost.

2.2.4 Riceovo kódování

Riceovo kódování je bezeztrátová kompresní metoda generující komprimovaná data s proměnlivou délkou jednotlivých hodnot. Pro hodnoty bližší nule generuje kratší binární kód nežli pro hodnoty více vzdálené. Z čehož plyne, že toto kódování je vhodné pro případy, kdy pravděpodobnost nízkých hodnot je mnohem vyšší než hodnot vysokých.

Je zde použit nastavitelný parametr M, jehož pomocí se vstupní hodnota $x \geq 0$ rozdělí na dvě části:

$$q = \lfloor x/M \rfloor \quad (2.14)$$
$$r = x - qM \quad (2.15)$$

Kde q odpovídá celočíselnému dělení hodnotou M a r je pak zbytek po celočíselném dělení. Pro Riceovo kódování musí být hodnota M mocninou dvou. Podíl q je zakódován pomocí unárního kódování, do výstupního řetězce bitů se vloží q-krát logická 0 a následně jedna logická 1. Zbytek r se následně zakóduje přímo v binárním kódování na b bitů.

$$b = \log_2(M) \quad (2.16)$$

Příklad v tabulce 2.2:

Hodnota ***x***	**Celočíselný podíl** ***q***	**Zbytek** ***r***	**Výstup**
0	0	0	1 00
1	0	1	1 01
2	0	2	1 10
3	0	3	1 11
4	1	0	01 00
5	1	1	01 01
6	1	2	01 10
7	1	3	01 11
8	2	0	001 00
9	2	1	001 01
10	2	2	001 10
11	2	3	001 11
12	3	0	0001 00
13	3	1	0001 01
14	3	2	0001 10
15	3	3	0001 11

Tabulka 2.2: Příklad Riceova kódování pro čísla do hodnoty 15, pro $M = 4 \Rightarrow b = log_2(4) = 2$.

V základní variantě však Riceovo kódování dovoluje zakódovat jen nezáporná celá čísla. Pro zakódování záporných celých čísel je nutné zavést posloupnost hodnot: 0, -1, 1, -2, 2, -3, 3, -4, 4, …

$$x' = \begin{cases} 2x & pro\ x \geq 0 \\ -2x - 1 & pro\ x < 0 \end{cases} \quad (2.17)$$

Kde x' je hodnota, která se bude kódovat místo x.

Druhou možností je posunutí nuly do středu rozsahu vstupních hodnot, tím však nebudou hodnoty blízké nule vyjádřeny na nejmenší počet bitů.

2.2.5 Získání parametru M Riceova kódování

Diference vstupního signálu $s(n)$ řádu $p \geq 0$ pro fixní lineární predikci je definována jako:

$$e(n,p) = \begin{cases} e(n,p-1) - e(n-1,p-1) & pro\ p > 0 \\ s(n) & pro\ p = 0 \end{cases} \tag{2.18}$$

Jednotlivé řády pak můžeme vyjádřit pouze za použití vstupního signálu (v kodeku FLAC jsou použity řády 0–4):

$$\begin{aligned} e(n,0) &= s(n) \\ e(n,1) &= s(n) - s(n-1) \\ e(n,2) &= s(n) - 2s(n-1) + s(n-2) \\ e(n,3) &= s(n) - 3s(n-1) + 3s(n-2) - s(n-3) \\ e(n,4) &= s(n) - 4s(n-1) + 6s(n-2) - 4s(n-3) + s(n-4) \end{aligned} \tag{2.19}$$

Celková chyba řádu p je pak definována jako:

$$E(p) = \sum_{n=0}^{N-1} |e(n,p)| \tag{2.20}$$

Kde N je počet vzorků zkoumaného rámce vstupního signálu.

Reziduální bity z každého vzorku pak dostaneme jako:

$$r(p) = \frac{\ln\left(\frac{\ln(2) * E(p)}{N}\right)}{\ln(2)} \tag{2.21}$$

Hledaný parametr M je dán jako:

$$M = \begin{cases} 2^{r(p)+1} & pro\ r(p) > 0 \\ 2 & pro\ r(p) \leq 0 \end{cases} \tag{2.22}$$

Zvolíme takový řád p, jehož celková chyba $E(p)$ je co nejmenší.

Tento postup je odvozen přímo z kódu kodeku FLAC, konkrétně pro fixní lineární predikci, standardně se ale pro určování parametru M používá medián vstupních hodnot. Medián je prostřední hodnota vzestupně seřazené řady čísel, pro sudý počet hodnot pak aritmetický průměr dvou hodnot nejblíže ke středu.

Kapitola 3

Vzorová data

Vzorová data, jež jsou při práci používány, pochází z lineárního mikrofonního pole s osmi mikrofony. Z obdržených dat nelze určit vzdálenost mezi mikrofony ani to, zda se opravdu jednalo o lineární mikrofonní pole, neboť v datech se posun jednotlivých kanálů vyskytuje jen zřídka, popřípadě je posun nepatrný. Tento jev mohou mít za následek dvě příčiny. První je, že mikrofony byly umístěny s velmi malou vzdáleností a druhá příčina, že všichni mluvčí, jenž se v datech objevují se nacházeli ve větší vzdálenosti od mikrofonního pole, k čemuž také napovídá to, že hlasy mluvčích jsou velmi tlumené. Tím se také snižuje rozdíl ve zpoždění mezi jednotlivými mikrofony. Popřípadě se může jednat i o obě příčiny najednou. Data byla také velmi zašuměná, míra šumu je také nejspíše způsobená utlumením mluvčích, neboť ve srovnání s hlasy mluvčích je šum opravdu velmi hlasitý. V datech se nachází celkem čtyři mluvčí, kde jeden z hlasů byl značně hlasitější než ostatní, z toho tedy vyplívá, že byl blíže k mikrofonnímu poli nežli ostatní.

Sada vzorových dat také obsahuje pět kanálů z headsetů, to znamená, že dalších pět mluvčích komunikovalo vzdáleně, to by také vysvětlovalo četné oblasti v datech z mikrofonního pole obsahující pouze šum. Tyto data však při práci použita nejsou.

Data jsou uložena do souborů formátu wav, s jedním kanálem v každém souboru, každý soubor pak obsahuje záznam o délce přibližně jedné a půl hodiny se vzorkovací frekvencí 16 kHz, kde každý vzorek je uložen jako 16bitové celé číslo se znaménkem.

Pro testovaní pak byly vytvořeny zkrácené verze těchto signálů o délce přibližně dvou minut a patnácti sekund, jednalo se o jednoduchou konkatenaci několika vybraných úseků původních signálů bez žádného přechodu či útlumu na hranici úseků.

Vzorová data pochází z AMI datasetu [9], který zahrnuje přes 100 hodin nahrávek. Při práci se však využívají pouze nahrávky z jednoho mikrofonního pole nahrávané na meetingu EN2001a. Vzorová data, použitá při práci, jsou pak také dostupná na serveru FIT VUT matylda5 v adresáři /mnt/matylda5/iveselyk/KALDI_AMI_WAV/EN2001a/audio/.

Kapitola 4

Implementace vlastních metod

Výsledkem celé práce je konzolová aplikace pro systémy Linux. Jedná se o jednoduchý kompresní a dekompresní nástroj. Výstupem je jediný spustitelný soubor, nazvaný *mlac*, provádějící jak kompresi, tak dekompresi na základě zadaných parametrů. Omezením je, že dokáže komprimovat pouze soubory ve formátu wav, a druhým omezením pak je, že může zpracovat maximálně osm kanálů, toto omezení vychází přímo z kodeku FLAC a k navýšení tohoto limitu by byly zapotřebí rozsáhlé změny v samotném kódu kodeku.

Projekt je napsaný kombinací jazyků C a C++, jazyk C je využit v knihovně kodeku FLAC, jenž je zde se změnami zachována a v jazyku C++ je napsáno hlavní jádro obsahující například první fázi zarovnání kanálů a práci se soubory ve formátu wav. Pro zjednodušení práce se pro zacházení se soubory formátu wav používá volně šiřitelná knihovna AudioFile [5]. I ta však nezůstala beze změny, neboť data ukládala normalizovaně do pole čísel s pohyblivou řádovou čárkou v rozsahu hodnot -1 až 1. To bylo nutné předělat na 32bitová celá čísla se znaménkem. Celý projekt, pak neobsahuje jeden samostatný algoritmus, jenž by řešil vše potřebné ke zvýšení komprese, nýbrž několik dílčích metod, jenž každá částečně přispívá ke výšení komprese. Většina práce pak spočívala v úpravě metod a postupů obsažených v použité knihovně kodeku FLAC, nikoliv ve psaní vlastních metod.

Jelikož součástí knihovny kodeku FLAC je také velké množství zdrojových souborů, u kterých nedošlo k žádné změně oproti původní implementaci, je v hlavičce každého zdrojového souboru napsaný seznam provedených změn. Jedinou změnou, která pak proběhla ve všech souborech z knihovny kodeku je korekce direktivy #include, neboť ty byly vázány vůči struktuře vytvořené při překladu projektu pomocí Microsoft Visual Studia.

4.1 Uživatelské rozhraní nástroje

Komprese vstupních souborů probíhá pomocí příkazu:

./mlac -e <output_mlac_file> <wav_files_list>

Kde *<output_mlac_file>* je výstupní komprimovaný soubor včetně případné relativní či absolutní cesty a *<wav_files_list>* je seznam vstupních souborů ve formátu wav, obsahující vstupní signály pro kódování. Oddělovačem mezi jednotlivými soubory v seznamu je mezera. Soubory mohou být zadány opět včetně relativní nebo absolutní cesty. Pro úspěšnou kompresi je vyžadován minimálně jeden soubor s minimálně jedním kanálem. Vstupní soubory však mohou obsahovat celkem pouze osm kanálů. Tato aplikace však také podporuje vstupní soubory s různými počty kanálů. Při dekódování jsou pak kanály správně rozděleny mezi příslušné soubory.

Opačným úkonem pak je dekomprese, lze ji spustit příkazem:

./mlac -d <input_mlac_file> [-o <wav_files_folder>]

Kde *<input_mlac_file>* je vstupní zkomprimovaný soubor ve formátu mlac včetně případné cesty a *<wav_files_folder>* je adresář, do kterého se uloží původní výstupní soubory. Parametr *-o* s touto hodnotou je volitelný, nebude-li zadán použije se adresář v němž je umístěn samotný nástroj *mlac*. Ve zkomprimovaném souboru mlac jsou také uloženy názvy vstupních souborů, ovšem bez cesty. Při okamžité zpětné dekompresi, jenž byla během testů velmi častá, by došlo k přepsání původních dat, proto je zde přidán parametr *-o*, pomocí které je možné určit jiný adresář.

K dispozici je také možnost pro vypsání základních informací o zkomprimovaném souboru bez jeho dekomprese a to příkazem:

./mlac -i <input_mlac_file>

Tímto příkazem, kde *<input_mlac_file>* je opět zkomprimovaný soubor ve formátu mlac, se pouze vypíší základní údaje o datech obsažených ve zkomprimovaném souboru, například technické parametry, jako počet kanálů nebo vzorkovací frekvence signálu, nebo názvy obsažených souborů včetně počtu kanálů jenž do daného souboru náleží.

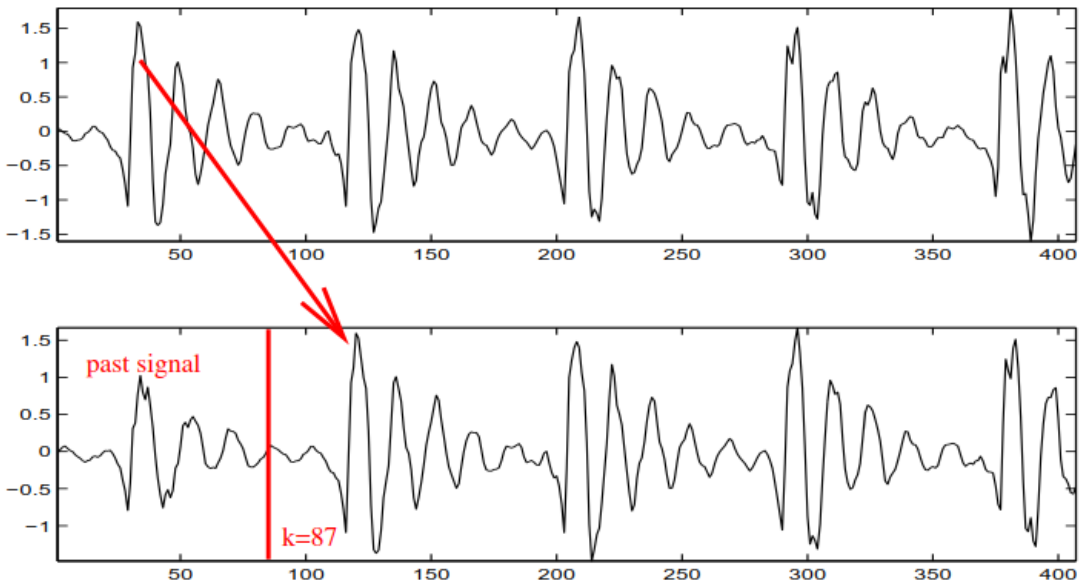
Poslední možností pak je vypsání jednoduché nápovědy obsahující základní popis parametrů, jenž je možné zde použít, a to pomocí příkazu:

./mlac [-h]

K vypsání stejné nápovědy dojde i v případě, kdy není zadán žádný parametr, parametr *-h* je tedy volitelný. Jakmile je však parametr *-h* zadán, tak k výpisu nápovědy dojde a ostatní parametry jsou ignorovány, nejedná se tedy o chybový stav, parametr *-h* má však vyšší prioritu nežli ostatní parametry.

4.2 Zarovnání kanálů vícekanálových dat

Ačkoli jsou signály v jednotlivých kanálech velmi podobné, tak největší rozdíl mezi nimi tvoří zpoždění signálů z jednotlivých mikrofonů. Abychom získaly signály ještě mnohem více podobné je vhodné toto zpoždění eliminovat tím, že jednotlivé kanály zarovnáme. Jelikož však neznáme ani vzdálenost mikrofonů, jejich rozložení a ani úhel pod kterým přicházel zvuk od zdroje k jednotlivým mikrofonům, nemůžeme hledané zpoždění jednoduše vypočítat z těchto parametrů, musíme zpoždění vypočítat jinak. Odpovědí je křížová korelační funkce. Konkrétně v této implementaci je použita normalizovaná křížová korelace [1], kde se pomocí normalizace eliminují chyby vzniklé rozdílnou energií porovnávaných signálů. Ilustrace zpoždění jednotlivých kanálů je znázorněna na obrázku 4.1.



Obrázek 4.1: Ilustrace posunutých signálů. Zpoždění spodního kanálu oproti hornímu je o k=87 vzorků, eliminací tohoto zpoždění pak vznikne vyšší shoda kanálů. Převzato z http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf, upraveno.

Pro zarovnání jednotlivých kanálů pomocí normalizované křížové korelační funkce, musíme nejprve vypočítat korelační koeficienty pro každý přijatelný posun mezi porovnávaným a zvoleným referenčním kanálem. Jako referenční kanál se zde používá první kanál, neboť v tuto chvíli nemáme představu o tom, který kanál je ve skutečnosti nejméně zpožděný, proto také je nutné vypočítat korelační koeficienty pro posun na obě strany. Srovnáním získaných koeficientů se pak vybere nejvhodnější hledaný posun neboli TDOA (time delay of arrival), nalezená hodnota TDOA odpovídá indexu největšího koeficientu. Hodnota koeficientů se blíží k jedné s tím, čím více si jsou navzájem posunuté signály podobné, pokud tedy dojde k přesné shodě, mezi porovnávaným a referenčním kanálem, pak bude hodnota odpovídajícího koeficientu rovna jedné. Tento jev je však velmi výjimečný, neboť vzorky obou signálů musí obsahovat jen čistý signál, bez jakéhokoliv šumu. Dále by musela přesně odpovídat fáze ve vzorkování signálů z obou mikrofonů, aby amplitudy odpovídajících vzorků v obou signálech byly shodné. Aby se u dvou signálů mohla najít přesná shoda musely by tyto signály být vytvořeny například syntézou, nikoliv pocházet z mikrofonního pole.

Jelikož se charakteristiky signálu mohou postupně měnit, ať už změnou mluvčího nebo i pouhým vyslovováním jiné hlásky, musí být koeficienty pro křížovou korelaci vypočítávány vždy jen pro malý úsek signálu zvaný rámec, neboť v následujícím rámci mohou být koeficienty naprosto rozdílné. Velikost rámce je v této aplikaci zvolena na počet vzorků, jenž odpovídá signálu o délce 25ms. I když se pracuje pouze s jedním rámcem tak kvůli posunům se přistupuje i ke vzorkům rámce předcházejícího, avšak velikost i tohoto posunutého rámce je stále stejná. Výjimku pak tvoří první a velmi často i poslední rámec signálu, neboť u prvního rámce není žádný předchozí rámec, do kterého by bylo možné přistupovat, tudíž jsou tyto hodnoty při výpočtu považovány za nulové. Poslední rámec pak je ve většině případů kratší nežli předešlé rámce, posunutý rámec je poté také zkrácen na tutéž velikost.

Normalizovaná křížová korelační funkce pro posun m, pak má tento vzorec:

$$NCCF(m) = \frac{\sum_{n=zr}^{zr+N-1} s_1(n-m) * s_2(n)}{\sqrt{E_1 E_2}} \qquad pro\ m \geq 0 \tag{4.1}$$

Kde zr je začátek aktuálního rámce, N je počet vzorků odpovídající velikosti rámce, s_1 a s_2 jsou dva porovnávané kanály vstupních signálů a E_1 a E_2 jsou energie signálů obou porovnávaných kanálů v aktuálním rámci, tuto energii pak získáme následovně:

$$E_1 = \sum_{n=zr}^{zr+N-1} s_1^2(n-m) \qquad E_2 = \sum_{n=zr}^{zr+N-1} s_2^2(n) \tag{4.2}$$

Tento vzorec pro normalizovanou křížovou korelaci však dává smysl jen pro kladné hodnoty posunu m, neboť při záporných hodnotách bychom museli přistupovat k budoucím vzorkům signálu s_2, ke kterým nemusíme mít vždy přístup, proto je nutné pro záporné hodnoty posunu m použít lehce odlišný vzorec navíc s prohozenými signály s_1 a s_2:

$$NCCF(m) = \frac{\sum_{n=zr}^{zr+N-1} s_1(n) * s_2(n+m)}{\sqrt{\sum_{n=zr}^{zr+N-1} s_1^2(n) * \sum_{n=zr}^{zr+N-1} s_2^2(n+m)}} \qquad pro\ m < 0 \tag{4.3}$$

Energie signálů E_1 a E_2 jsou v tomto vzorci již rozepsány, neboť ke stejným změnám došlo i zde.

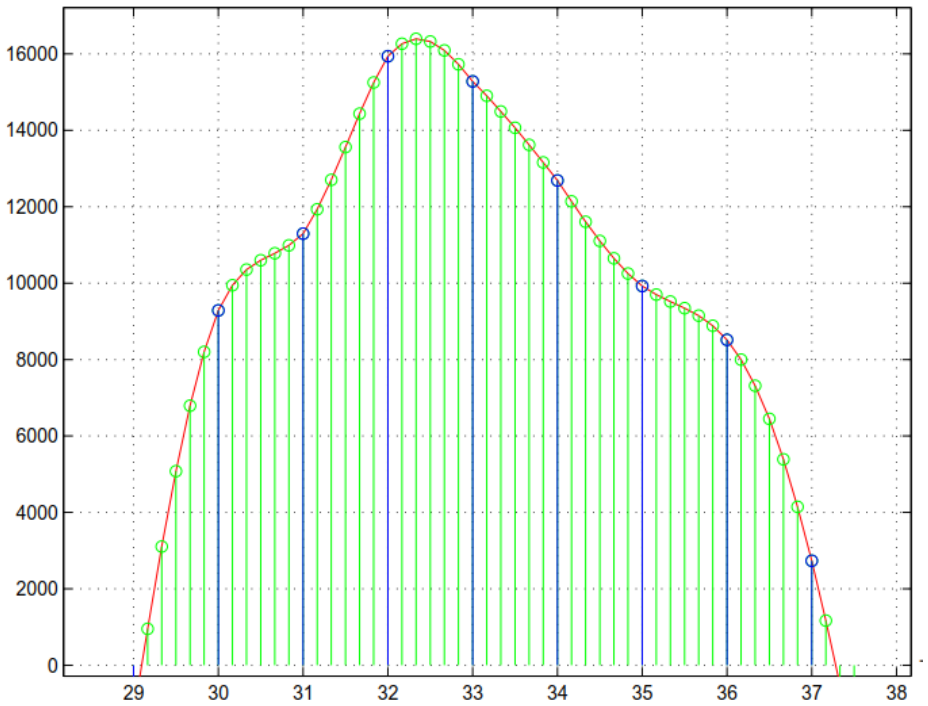
Samotný výpočet koeficientů křížové korelace probíhá ve dvou fázích. V první se pokoušíme najít přibližné zpoždění posouváním signálů parametrem m pouze po celých číslech, tj. posouváme signály vždy o celý vzorek. Parametr m se pak v této fázi pohybuje mezi hodnotami $-M$ a M, kde M je polovina velikosti aktuálního rámce, tím posouváme signál maximálně o polovinu rámce, tato hodnota je pak ještě omezena na maximální hodnotu 120, aby se celý rozsah hodnot od $-M$ do M vešel do osmi bitů, Parametr m je pak zvyšován s krokem 1. Ze získané sady koeficientů, se pak vypočítá nejvhodnější TDOA, jenž představuje přibližné zpoždění signálu vůči referenčnímu kanálu, hodnota TDOA je jinými slovy takový posun, jehož odpovídající korelační koeficient je nejvyšší. Toto přibližné zpoždění je tedy v rozlišení na jeden vzorek, tj. jedná se stále jen o celé číslo.

4.3 Zpřesněné zarovnání kanálů pomocí interpolace

V druhé fázi zarovnání kanálů se pak vypočítané zpoždění ještě čtyřnásobně zpřesní. Parametr m se v této fázi pohybuje okolo hodnoty TDOA nalezené v první fázi, konkrétně se pohybuje mezi hodnotami $TDOA - 0{,}75$ a $TDOA + 0{,}75$. Parametr m se pak zvyšuje o 0,25, získáme tedy celkem sedm možných posunutí signálů. Stejným způsobem určíme nejvhodnější posun tentokrát v rozlišení na jednu čtvrtinu vzorku, tím se také přiblížíme k přesnému zpoždění mezi signály. Ovšem posunutím signálu o takovou hodnotu na sebe nebudou vzorky obou signálů navazovat, vzorky neposunutého signálů budou připadat do oblasti mezi vzorky posunutého signálu. Pro správné porovnání však potřebujeme mít v tomto místě také vzorky, ty tam však nejsou, proto je nutné je získat jiným způsobem a tím způsobem je tzv. interpolace, pomocí níž můžeme odhadnou hodnotu vzorků v oblasti mezi vzorky kde žádné vzorky nejsou. Nejjednodušší možností interpolace je tzv. lineární interpolace, ta je však velmi nepřesná, neboť provádí pouze lineární kombinaci dvou okolních vzorků, jednoho z každé strany, například při interpolaci mezi vzorky 32 a 33 na obrázku 4.2 by správně neurčila hodnoty vzorků mezi nimi a za lokální maximum by stále považovala přímo vzorek 32, i když lokálním maximem je ve skutečnosti vzorek mezi těmito dvěma vzorky. Jendou z přesnějších možností, jenž je použita i v této aplikaci, je použití filtru typu sinc, interpolace filtrem typu sinc pak vypadá následovně:

$$s'(i) = \begin{cases} s(i) & pro\ i \in \mathbb{Z} \\ \sum_{n=a}^{b} s(n) * sinc\big(\pi * (i-n)\big) & pro\ i \notin \mathbb{Z} \end{cases} \tag{4.4}$$

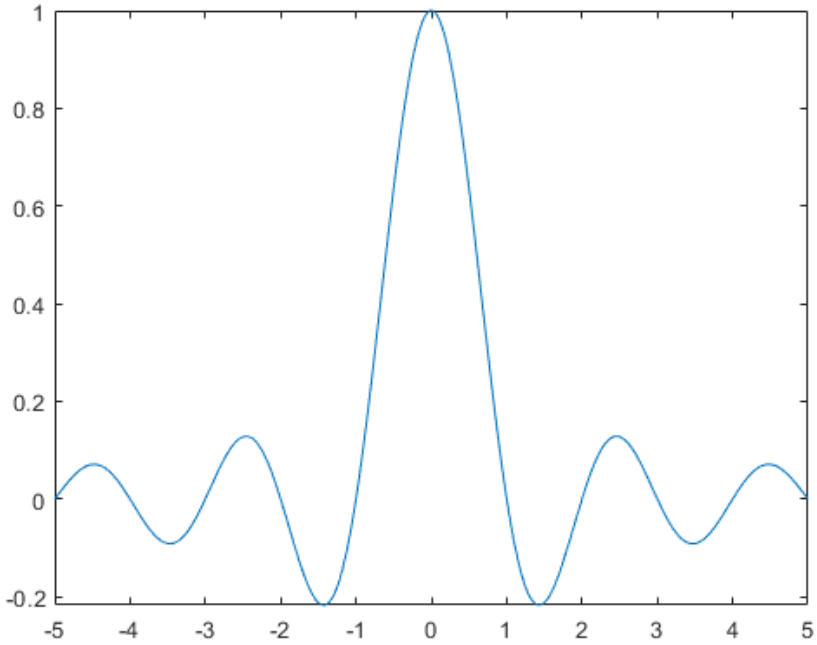
Kde i je index v desetinném tvaru, kde celá část představuje index vzorku ve vstupním signálu a desetinná část pak je index mezi-vzorku následujícího za vzorkem identifikovaným celou částí indexu. V kódu jsou pak tyto indexy reprezentovány jako čísla s pevnou desetinnou čárkou se dvěma bity pro desetinou část. Meze sumy pro interpolaci a a b jsou určeny tak, aby suma obsáhla vždy pět vzorků před požadovaným indexem i a pět za tímto indexem, pokud však tento interval přesáhne jednu z hranic rámce, pak je tento interval posunut o potřebný počet vzorků, aby výpočet probíhal vždy v jednom rámci, je-li rámec menší, než tento interval pak je interval zkrácen na celý rámec. Standartně se pro interpolaci filtrem sinc používá celý signál, to je ale velmi časově náročně a celá operace by pak trvala mnohem déle. Příliš vysoká náročnost se pak projevuje i při aplikování filtru sinc na celý rámec. To je hlavním důvodem proč se zde používá jen malé okno okolo požadovaného indexu, nevýhodou pak je nižší přesnost interpolace.



Obrázek 4.2: Interpolace signálu pomocí vložení nových pěti vzorků mezi původní vzorky. Červená křivka představuje spojitý signál, modré vzorky jsou původní vzorky signálu získané vzorkováním spojitého signálu a zelené vzorky jsou výstupem přesné interpolace. Převzato z http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf.

Samotná funkce sinc je definováno následovně:

$$sinc(x) = \begin{cases} \frac{\sin x}{x} & pro\ x \neq 0 \\ 1 & pro\ x = 0 \end{cases} \tag{4.5}$$



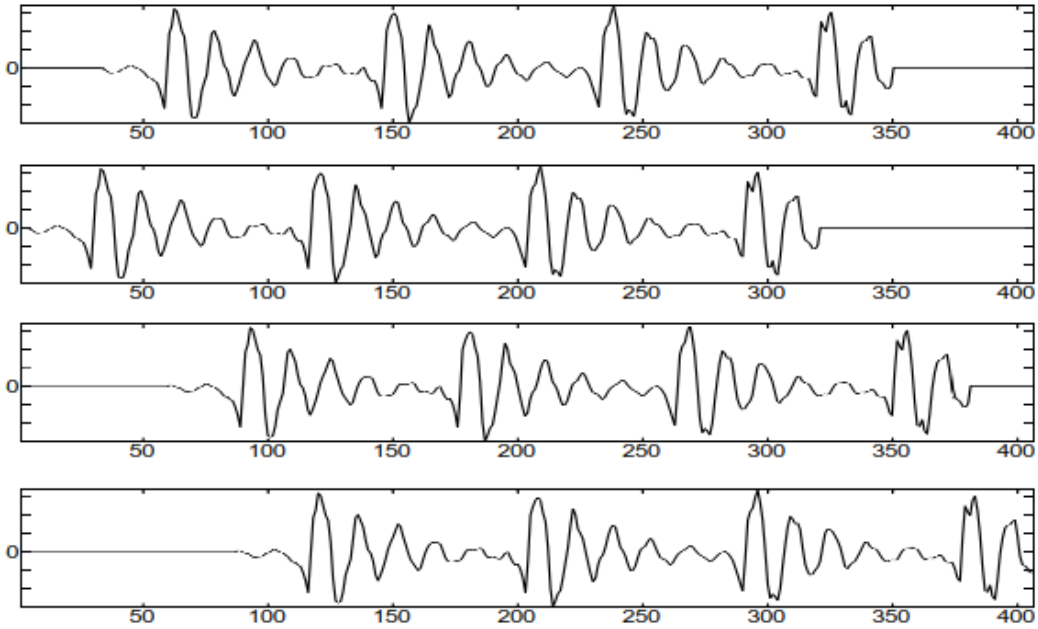
Obrázek 4.3: Funkce y=sinc(πx). Převzato z https://www.mathworks.com/help/symbolic/sinc.html.

Celý výpočet hodnot TDOA do této chvíle probíhal ve více vláknech, každý kanál, kromě prvního, u kterého se hodnota TDOA nevypočítává, tak obsadil vlastní vlákno, tudíž výpočet hodnot TDOA pro všechny ostatní kanály probíhal paralelně, jeden z kanálů pak zpracovávalo i hlavní vlákno, aby nečinně nečekalo, dokud vedlejší vlákna nedokončí výpočet hodnot TDOA. Například máme-li celkem osm kanálů, tak první je pro výpočet TDOA referenční, u něj se nic nepočítá, druhý kanál se zpracovává přímo v hlavním vlákně a zbylých šest kanálů se pak zpracovává v šesti vedlejších vláknech.

4.4 Výběr referenčního kanálu

Po získání zpoždění TDOA pro všechny kanály v aktuálním rámci je nutné zjistit skutečný referenční kanál a v případě potřeby provést korekci hodnot TDOA aby se vztahovaly oproti referenčnímu kanálu, nikoliv vůči prvnímu kanálu. Sada hodnot TDOA je totiž vypočítávána oproti prvnímu kanálu vstupních signálů, pro první kanál je tedy hodnota TDOA vždy rovna hodnotě nula, pro ostatní kanály je pak vypočítána pomocí výše uvedeného postupu, kde právě signál s_1 v rovnicích 3.1 a 3.3 je první kanál a signál s_2 je pak jeden z ostatních kanálů, pro který hledáme TDOA. Je-li hodnota TDOA kladná, pak kanál, jemuž tato hodnota náleží, je zpožděn oproti prvnímu kanálu, pokud je hodnota záporná, pak je naopak zpožděn první kanál oproti tomuto kanálu, kterému hodnota TDOA patří.

Jako referenční kanál se pak vybere takový kanál, který není zpožděn oproti žádnému jinému kanálu, jinými slovy referenční kanál je ten, který má nulovou případně i zápornou hodnotu TDOA. Výběr referenčního kanálu je naznačen v obrázku 4.4. Má-li více kanálu shodnou a zároveň nejnižší hodnotu TDOA, pak je vybrán jako referenční kanál první v pořadí z těchto kanálů. Pokud je hodnota TDOA referenčního kanálu záporná, pak je nutné provést korekci, aby se celá sada hodnot TDOA vztahovala vůči referenčnímu kanálu místo vůči prvnímu kanálu. Korekce se pak provede jednoduchým přičtením absolutní hodnoty této hodnoty TDOA referenčního kanálu ke všem hodnotám TDOA, tím se zajistí, že referenční kanál bude mít vždy tuto hodnotu rovnu nule. Jelikož to byla také nejnižší hodnota došlo navíc k eliminaci všech záporných hodnot, které jsou pro následné zpracování nežádoucí, od této chvíle jsou tedy všechny hodnoty TDOA nezáporné. Spolu s tím také nastane druhý efekt, neboť referenční kanál má hodnotu vždy rovnu nule, to se pak využívá v dekodéru k určení referenčního kanálu, který je potřeba i v dekodéru znát.



Obrázek 4.4: Výběr referenčního kanálu na ilustračních datech: Jako referenční kanál bude vybrán druhý kanál, neboť jeho zpoždění vůči prvnímu kanálu bylo záporné. To znamená, že byl v předstihu o přibližně 35 vzorků, a nikoliv zpožděn, hodnota 35 se poté přičte ke všem hodnotám zpoždění TDOA. Převzato z http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf, upraveno.

V tuto chvíli jsou vstupní signály předávány do knihovny kodeku FLAC, avšak ještě před tím dochází k menší úpravě samotných dat a tím je mírná změna pořadí kanálů pro aktuální rámec a to tak, že referenční kanál je předán jako první a ostatní pak následují v jejich původním pořadí pouze však přeskočí referenční kanál, který byl předán jako první. Tato změna pořadí se však projevuje pouze u předávání dat aktuálního rámce do knihovny kodeku, pro následující rámce je pořadí kanálů zachováno v původním pořadí, v následujícím rámci se pak znovu vyhodnotí nové pořadí z výchozího stavu. Hodnoty TDOA se pak nakonec uloží do hlavičky rámce, která je společná pro všechny kanály neboli dílčí rámce aktuálního rámce.

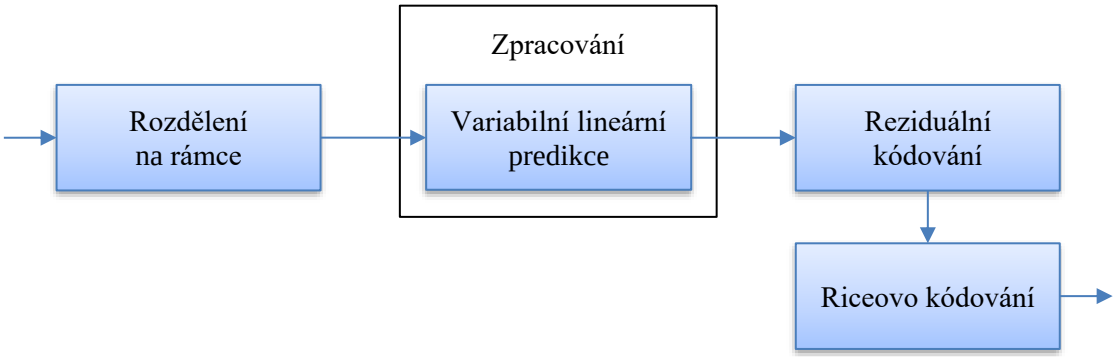
4.5 Úpravy knihovny kodeku FLAC

Knihovna kodeku FLAC prošla četnými změnami v kódu oproti původním zdrojovým kódům dostupným na oficiálních stránkách [4]. Jak již bylo řečeno, tak FLAC používá celkem čtyři možnosti zpracování: doslovné, konstantní a fixní a variabilní lineární predikci. První změnou provedenou uvnitř knihovny kodeku FLAC je to, že při zpracování rámce se nyní používá pouze variabilní lineární predikce, ostatní možnosti zpracování se nepoužívají, a dokonce jsou i odstraněny. Tento krok bylo nutné provést, aby každý dílčí rámec měl k dispozici LP reziduum z předchozího rámce, výjimku pak samozřejmě tvoří první rámec, který žádný předchozí rámec nemá. Přesněji řečeno, LP reziduum z předchozího rámce je potřebné pouze u referenčního kanálu, u ostatních kanálů se nijak nevyužívá. LP reziduum je reziduální signál, jenž vnikl pomocí reziduálního kódování. LP reziduum se vypočítá následovně:

$$R_{lp}(n) = s(n) - \tilde{s}(n) \tag{4.6}$$

Kde $s(n)$ je původní signál a $\tilde{s}(n)$ je predikovaný signál, jenž vznikl pomocí lineární predikce, index n pak prochází přes všechny vzorky v aktuálním dílčím rámci.

Termín LP reziduum je zde zaveden především proto, že později bude uveden další reziduální signál se zcela odlišným významem. Blokové schéma kodéru je pak značně zjednodušeno, jak je také vidět na obrázku 4.5.



Obrázek 4.5: Upravené blokové schéma kodéru v kodeku FLAC včetně směru toku dat.

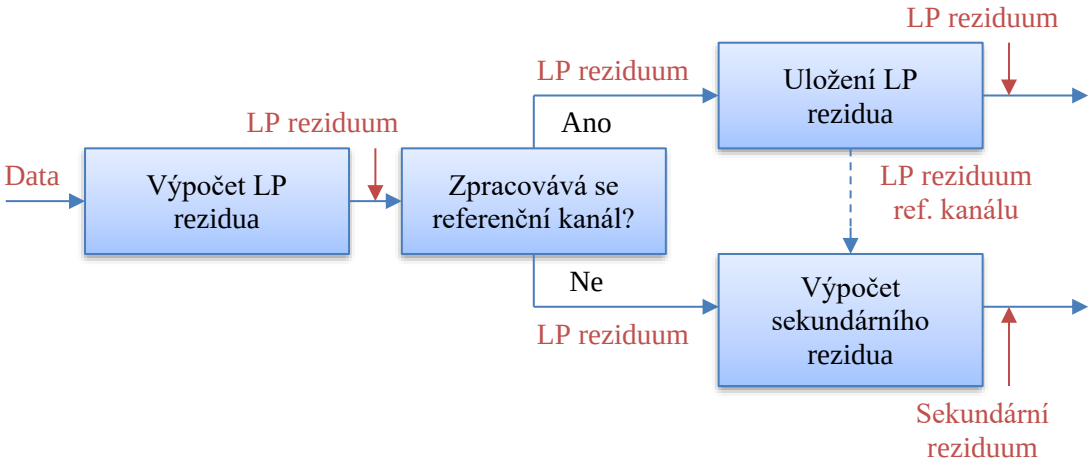
Tato změna se pak projeví, nepatrným zvýšením velikosti výstupního zkomprimovaného souboru, pokud kodek FLAC při zpracování narazí na konstantní dílčí rámec, zpracuje jej v tomto stavu také pomocí variabilní lineární predikce a tím zvětší velikost dat. Konstantní dílčí rámec je však velmi výjimečný, tudíž se v datech příliš často neobjevuje, navzdory tomu se ve vzorových datech konstantní dílčí rámce nacházejí. Navýšení velikosti tímto jevem se však vyrovná pomocí ostatních metod, jenž výslednou velikost zkomprimovaného souboru naopak snižují.

Vzhledem k vysoké podobnosti signálů v jednotlivých kanálech je možné použít shodné koeficienty pro lineární predikci pro data všech kanálů. Použitím totožných koeficientů pak také musíme zvolit stejný řád predikce, neboť pro jiný řád predikce by také bylo potřeba použít jiné koeficienty, čemuž jsme se právě tímto krokem chtěly vyhnout, abychom mohly do výsledného zkomprimovaného souboru zapsat koeficienty pouze jednou jen pro jeden z dílčích rámců a tím snížit velikost metadat, jenž musíme zapsat do výstupního souboru. Tento dílčí rámec pak odpovídá referenčnímu kanálu. To byl také jeden z důvodů, proč se před vstupem do knihovny kodeku FLAC mění pořadí kanálů, aby referenční kanál byl v aktuálním rámci zpracován vždy jako první. Koeficienty lineární predikce a její řád se pak vypočítávají vždy na referenčním kanálu.

Jak už bylo řečeno původní verze kodeku FLAC zpracovávala každý rámec, a i každá kanál zcela nezávisle, to však znamenalo, že pokud pro daný dílčí rámec byla zvolena jedna z forem lineární predikce, bylo nutné několik úvodních vzorků na začátku dílčího rámce uložit v původní podobě, aby dekodér měl k dispozici první vzorky, z nichž mohl provádět samotnou predikci a následnou rekonstrukci signálu. Počet těchto úvodních vzorků se pak odvíjel od zvoleného řádu predikce, odpovídající počtu vzorků, z nichž se provádí lineární kombinace při lineární predikci. Aby bylo ale možné přistupovat k LP reziduu minulého rámce je také nutné mít celé LP reziduum z aktuálního rámce, aby tento reziduální signál neobsahoval žádné mezery. Další změnou provedenou v knihovně kodeku FLAC bylo vynechání těchto úvodních vzorků. Místo použití úvodních vzorků dílčího rámce se používají poslední vzorky z předchozího rámce odpovídajícího kanálu o stejném počtu vzorků. Samotné LP reziduum pak tedy vzniká z celého rámce už od prvního vzorku dílčího rámce, nikoliv až ze vzorků následujících za úvodními vzorky. Výjimku pak tvoří opět první kanál, kde úvodní vzorky nadále zůstaly, neboť zde není žádný předchozí rámec, ze kterého bychom tyto vzorky mohly použít. Tím, že místo úvodních vzorků se do výstupního souboru ukládají pouze jejich rezidua se také ušetřily další bity ve výsledné velikosti výstupního souboru.

Druhá část zarovnání kanálů pak probíhá až po dokončení výpočtu LP rezidua. V této části je potřeba mít přístup k LP reziduu předchozího rámce, přesněji řečeno je potřeba jen LP reziduum z předchozího rámce pouze u referenčního kanálu. Tudíž toto je důvod proč se využívá pouze variabilní lineární predikce, také je zde potřeba mít souvislé LP reziduum bez jakýchkoliv mezer, tj. je potřeba LP reziduum z celého rámce, proto bylo nutné se zbavit úvodních vzorků na začátku každého dílčího rámce, aby v LP reziduu nevnikaly nežádoucí mezery. Teprve v této fázi dochází k samotnému posunu signálů o patřičnou hodnotu TDOA, dosud proběhl pouze výpočet těchto posunů a určení referenčního kanálu, všechny signály však byly dosud zpracovávány bez jakéhokoliv zarovnání signálů. Zpracování dílčích rámců však není pro všechny kanály stejné, výjimku zde představuje referenční kanál, jenž je zpracován odlišným způsobem. U referenčního kanálu se do výsledného zkomprimovaného souboru ukládá přímo LP reziduum tohoto dílčího rámce, dále se pak pro pozdější použití ukládá LP reziduum referenčního kanálu aktuálního rámce do pole, jenž obsahuje LP reziduum z předchozího rámce jeho referenčního kanálu, referenční kanál předchozího rámce nemusí odpovídat referenčnímu kanálu aktuálního rámce. Toto LP reziduum z předešlého rámce je pak ihned následováno LP reziduem referenčního kanálu z rámce aktuálního, toto pole obsahující tedy dvě LP rezidua z předešlého a aktuálního rámce se pak používá při zpracování ostatních kanálů.

Zpracování ostatní kanálů začíná stejným způsobem, a to vytvoření LP rezidua ze signálu aktuálního dílčího rámce. K posouvání signálů pak dochází pouze u zpracování ostatních kanálů. Avšak se neposouvá s tímto signálem, respektive s tímto LP reziduem, ale s LP reziduem referenčního kanálu uloženého v již zmíněním poli. To je také důvod, proč je v tomto poli uchováno i LP reziduum z předchozího rámce, neboť díky posunutí se přistupuje i do tohoto rezidua. Po dokončení výpočtu LP rezidua daného kanálu se tedy vytvoří druhé reziduum, dále označované jako sekundární, toto sekundární reziduum vzniká odečtením LP rezidua aktuálního rámce od posunutého LP rezidua referenčního kanálu. Pokud však hodnota posunu TDOA není celé číslo, musí se opět uplatnit interpolace filtrem typu sinc. Až toto sekundární reziduum se pak ukládá do výstupního zkomprimovaného souboru. LP reziduum referenčního kanálu se posouvá vždy směrem doprava, díváme-li se na tento signál na časové ose, jinými slovy přistupujeme pouze ke vzorkům více v minulosti. To je tedy důvod, proč je nutné si uchovat LP reziduum referenčního kanálu z předchozího rámce a také proč je nutné používat pouze variabilní lineární predikci.



Obrázek 4.6: Blokové schéma zpracování signálu pomocí variabilní lineární predikce, červené popisky představují druh dat, která se v daném místě nachází. Přerušovaný šipka pak značí zpřístupnění jejích dat bez následování této cesty během zpracování. Jak je vidět LP reziduum referenčního kanálu se používá při výpočtu sekundárního rezidua u ostatních kanálů, proto se referenční kanál musí zpracovávat vždy jako první.

Sekundární reziduum pak tedy získáme následovně:

$$R_s(n) = R_{ref,lp}(n - T) - R_{lp}(n) \tag{4.7}$$

Kde R_s je sekundární reziduum aktuálního rámce daného kanálu neboli dílčího rámce, $R_{ref,lp}$ je LP reziduum referenčního kanálu, kde záporné indexy představují LP reziduum z předchozího rámce a nezáporné z aktuálního rámce, index nula pak připadá na první hodnotu LP rezidua aktuálního rámce referenčního kanálu, T je zpoždění TDOA pro požadovaný kanál v aktuálním rámci a R_{lp} je LP reziduum daného kanálu. Index n pak prochází přes všechny vzorky aktuálního dílčího rámce daného kanálu.

Touto operací vznikne signál s nižší energií než původní LP reziduum a tím se ušetří další bity ve výsledném zkomprimovaném souboru.

Kapitola 5

Testování a dosažené výsledky

5.1 Metrika testů

Metrika byla zvolena velmi jednoduše, a to na celkovou velikost zkomprimovaného souboru, ta je pak udávána ve třech různých hodnotách. První tedy je celková velikost tohoto souboru a druhá pak procentuální vyjádření této velikosti oproti celkové velikosti vstupních dat. Třetí hodnotou pak je opět procentuální vyjádření, tentokrát však rozdílu velikostí zkomprimovaného souboru z této aplikace a z referenčního kodeku FLAC oproti velikosti výstupního souboru z referenčního kodeku. Tato hodnota tedy odpovídá relativnímu snížení či navýšení velikosti výstupního souboru vůči referenčnímu kodeku. Ve všech třech případech se počítá s výslednou velikostí souborů, nikoliv jen s velikostí samotných dat bez nezbytných metadat. Pro některé účely by však bylo vhodnější měřit právě velikost samotných zkomprimovaných dat, nikoliv včetně souvisejících metadat, ale ty jsou pro správné dekódování nezbytné, tudíž bez nich by se nedalo data zpětně dekódovat. Druhou vlastností pak je doba, jak dlouho celá operace trvala, od spuštění nástroje až po jeho úspěšné ukončení. Doba trvaní je tedy druhou metrikou, na kterou se přihlíží.

5.2 Průběh testování

Testování probíhalo průběžně během celé práce. Po aplikování nové metody, nebo po provedení úpravy nějaké ze stávajících metod bylo provedeno testovací kódování, buďto se vzorovými daty, nebo se zkrácenými testovacími signály. Při testování se pak výsledky porovnávaly s předchozími výsledky, a hlavně s výsledky z referenčního kodeku FLAC. Během testování pak bylo provedeno mnoho dalších testů, u kterých se nepřihlíželo na velikost zkomprimovaného souboru, ale pouze na některé z vedlejších pomocných výpisů používaných při vývoji aplikace. Příkladem těchto vedlejších výpisů je například výpis vypočítaného TDOA nebo vybraný referenční kanál. Testy zaměřené pouze na tyto vedlejší výpisy však nebyly zachovány pro následné porovnávání, neboť to nebylo ani jejich účelem. Tyto testy tedy sloužily pouze pro ověření správné funkčnosti některých dílčích metod, nikoliv pro ověřování výsledného programu jako celku.

Testování bylo hlavně zaměřeno na celkovou velikost zkomprimovaného souboru a na bezztrátovost celého procesu kódování a dekódování. Doba trvání výpočtu, jenž je zde zavedena jako druhá metrika, pak byla nejméně prioritní vlastností, na kterou v podstatě nebylo přihlíženo, ale zavedením výpočtu TDOA pomocí vláken se celý výpočet značně urychlil. Výpočet je v tuto chvíli přibližně třikrát rychlejší než bez použití vláken, i tak je ve srovnání s referenčním kodekem FLAC stálé velmi pomalý.

Aplikace během vývoje také obsahovala velké množství již zmíněných pomocných ladících výpisů, ať už vypisujících přímo do konzole nebo při větším objemu dat stejného typu do samostatného souboru, tyto vedlejší výpisy však byly pouze dočasné a v tuto chvíli již v aplikaci žádné nejsou.

5.3 Výsledky testů

Výsledky testů není možné příliš dobře porovnávat, neboť testy byly spouštěny nad různými daty. Některé testy byly spuštěny nad celými vzorkovými daty, některé pouze nad dvěma kanály vzorových dat a ostatní jen nad vytvořenými testovacími signály. Kromě toho bylo provedeno mnoho testů bez výstupního zkomprimovaného souboru, pouze s vedlejšími výstupy. Tyto testy pro vedlejší výstupy pak byly převážně prováděny nad zkrácenými testovacími signály. Testovací signály byly vytvořeny později během vývoje pro urychlení testování, v úvodních fázích vývoje k dispozici tyto signály nebyly.

Jak je vidět i v následující tabulce 5.1, jenž obsahuje porovnání finální verze aplikace a referenčního kodeku, výsledky nedopadly dle očekávání, neboť výsledný soubor je o přibližně 2,14 % větší než soubor, jenž byl výstupem kodeku FLAC. To může být způsobeno vysokou úrovní šumu obsaženého ve vzorových datech, neboť díky šumu je možné že se nepodařilo správně zarovnat jednotlivé kanály a výsledné sekundární reziduum mělo dokonce vyšší energii nežli LP reziduum, z něhož bylo sekundární reziduum vypočítáno. Pravděpodobně se také nepodařilo ušetřit dostatek bitů na metadatech po prodloužení reziduálních signálů na celou délku rámce, kde toto prodloužení výslednou velikost opět zvýšilo.

	FLAC	MLAC
Velikost souboru v MB	476,1 MB	486,3 MB
Velikost souboru v %	35,42 %	36,18 %
Doba trvání výpočtu	2 min 23,464 s	44 min 39,049 s

Tabulka 5.1: Porovnání referenčního kodeku FLAC a nástroje, jenž je výstupem této práce. Velikost souboru v procentech je oproti celkové velikosti souborů se vstupními daty, doby trvání jsou pak uvedeny včetně samotného načítaní dat. Experiment byl proveden na mém notebooku.

Jelikož průběžné testy byly prováděny nad různými daty, nemůžeme porovnávat všechny dílčí metriky. K porovnávání nám tedy musí stačit pouze procentuální vyjádření velikosti zkomprimovaného souboru oproti velikosti původních souborů, i tak, zde budou uvedeny i velikosti v MB. Následující tabulka 5.2 pak uvádí velikosti vstupních souborů pro lepší orientaci v datech.

Vstupní data	Celková velikost souborů v MB
Vzorová data, dva kanály	336 MB
Vzorová data, osm kanálů	1344 MB
Testovací zkrácené signály, osm kanálů	35,5 MB

Tabulka 5.2: Celkové velikosti vstupních souborů různých použitých vstupních dat.

Provedená úprava	Velikost souboru	Delta	Vstupní data
Kodek FLAC	35,03 % (117,7 MB)	±0 %	Vzorová data, 2 kanály
Kodek FLAC	35,42 % (476,1 MB)	±0 %	Vzorová data, 8 kanálů
* Nevyhlazené TDOA	35,65 % (119,8 MB)	+1,78 %	Vzorová data, 2 kanály
* Vyhlazené TDOA	35,57 % (119,5 MB)	+1,53 %	Vzorová data, 2 kanály
* Vyhlazené TDOA 2	35,48 % (119,2 MB)	+1,27 %	Vzorová data, 2 kanály
* Vyhlazené TDOA 2	37,31 % (501,4 MB)	+5,31 %	Vzorová data, 8 kanálů
Pouze LP dat	34,92 % (469,3 MB)	-1,43 %	Vzorová data, 8 kanálů
* Pouze LP reziduálního signálu	37,17 % (499,6 MB)	+4,94 %	Vzorová data, 8 kanálů
* Pouze LP reziduálního sig. + gain	37,12 % (498,9 MB)	+4,79 %	Vzorová data, 8 kanálů
Pouze LP, sekundární reziduum	35,04 % (470,9 MB)	-1,09 %	Vzorová data, 8 kanálů
Pouze LP, bez úvodních vzorků	34,04 % (457,5 MB)	-3,91 %	Vzorová data, 8 kanálů
Pouze LP, řád pevně nastaven na 8	42,82 % (15,2 MB)	NaN	Testovací sig., 8 kanálů
Pouze LP, řád pevně nastaven na 12	42,25 % (15,0 MB)	NaN	Testovací sig., 8 kanálů
Pouze LP, řád pevně nastaven na 16	42,82 % (15,2 MB)	NaN	Testovací sig., 8 kanálů
Pouze LP, řád pevně nastaven na 20	42,54 % (15,1 MB)	NaN	Testovací sig., 8 kanálů
Pouze LP, max. řád zvýšen na 20	42,82 % (15,2 MB)	NaN	Testovací sig., 8 kanálů
Pouze LP, použita interpolace sinc	42,25 % (15,0 MB)	NaN	Testovací sig., 8 kanálů

Tabulka 5.3: Srovnání některých testů. V tabulce je také znovu uvedena komprese samotného kodeku FLAC pro jednodušší porovnání. Jak je vidět z dat, tak komprese testovacích signálů nebyla stejně efektivní jako u celých vzorových dat. To je ale způsobené vyšší dynamikou signálů v těchto zkrácených testovacích signálech. Jednotlivé testy jsou pak seřazeny ve stejném pořadí, v jakém byly prováděny. Sloupec „Delta" pak představuje o kolik procent se navýšila, popřípadě snížila celková velikost souboru oproti velikosti souboru z referenčního kodeku. Hodnota NaN je pak uvedena u testů, u kterých není k dispozici reference z kodeku FLAC. Experimenty byly provedeny na mém notebooku.

* Knihovně kodeku FLAC nebyla předána vstupní data, nýbrž rozdíl mezi posunutým referenčním kanálem a kanálem dat, ke kterému tento reziduální signál náleží. Jinými slovy k tomuto odečtení v prvních fázích vývoje docházelo již před vstupem do knihovny kodeku. Pouze referenční kanál byl předávám v původní podobě.

Jak je vidět podle tabulky 5.3, vytvoření reziduálního signálu přímo mezi vstupními daty nebylo nejlepším řešením, později se pak ukázalo, že odečtení zarovnaných kanálů LP reziduí je mnohem účinnějším řešením. Aby však bylo možné toto provést bylo nutné zajistit, aby knihovna kodeku FLAC používala výhradně variabilní lineární predikci a aby její LP rezidua pokrývaly celý rámec.

Vyhlazování TDOA probíhalo jednoduchým váženým průměrem mezi právě vypočítanou hodnotou TDOA a hodnotou z předchozího rámce, váhy byly pak voleny tak, aby se navzájem doplňovaly do hodnoty jedna. Testován byl poměr 0,5 na 0,5 a 0,25 na 0,75 s vyšší váhou pro novou hodnotu TDOA. Vyhlazením se sice snížila celková velikost souboru, ale pro později přidanou interpolaci to bylo nevhodné, kvůli nepřesnostem, jež vyhlazením vznikaly. Proto se ve finální verzi aplikace vyhlazování TDOA nepoužívá. Ničemu také nevyhlazené TDOA nebrání, neboť v datech nevnikají žádné mezery nebo překryvy.

Dále stojí za zmínku úprava v tabulce 5.3 pojmenovaná „Pouze LP dat", kde již v knihovně kodeku FLAC bylo nastavené výhradní používání variabilní lineární predikce a také používání stejných koeficientů lineární predikce pro všechny kanály. Jelikož se koeficienty predikce ukládaly do výstupního souboru pouze u jednoho kanálu ušetřilo se tím dostatek bitů, aby celková velikost souboru byla dokonce menší než u referenčního kodeku.

Další dva testy nazvané „Pouze LP reziduálního signálu“ a „Pouze LP reziduálního sig. + gain“ pak používaly obdobný přístup ke knihovně kodeku FLAC jako předchozí test. Pouze na vstup knihovny nevkládaly samotná data bez zarovnání, ale již odečtené zarovnané kanály, tento reziduální signál tedy vznikal ještě před vstupem do knihovny kodeku. Tento signál však ztratil správnou strukturu signálu natolik, že lineární predikce se stejnými koeficienty jako pro referenční kanál, jenž byl jako jediný předáván knihovně v původní podobě, nedokázala vytvořit LP reziduum s dostatečně nízkou energií. Gain u druhého z těchto testů pak představoval vyrovnání energií signálů před samotným odečtením. V tom souboru však hodnoty gain uloženy nejsou, tudíž po jejich přidání by výsledný soubor byl pravděpodobně větší než z testu bez aplikování gainu.

V testu „Pouze LP, sekundární reziduum“, je již aplikovaný výpočet sekundárního rezidua ovšem jsou zde stále zanechány úvodní vzorky na začátku rámce pro lineární predikci. Díky těmto úvodním vzorkům, ale při posunutí LP rezidua referenčního kanálu toto reziduum správně nenavazovalo na LP reziduum referenčního kanálu z předchozího rámce. Tento test tedy sice vypadal úspěšně, neboť výsledná soubor byl menší než u referenčního kodeku, avšak tento přístup byl chybný.

Test pojmenovaný „Pouze LP, bez úvodních vzorků“, pak dopadl, ze všech testů nejlépe, zdaní však může klamat, neboť výsledný soubor z tohoto testu nebylo možné dekódovat. LP reziduum, a i sekundární reziduum bylo vypočítáváno na celém rámci, tedy bez použití úvodních vzorků, místo kterých se používaly vzorky na konci předchozího rámce. Problém však vznikl při kódování a následném zápisu do výstupního souboru reziduí pomocí Riceova kódování. Riceovo kódování totiž používalo reziduum, které bylo zkrácené o počet úvodních vzorků, jelikož se ale úvodní vzorky již nepoužívaly, tak těchto pár vzorků na konci rezidua zůstalo nezpracovaných. Tyto zkrácená rezidua pak byly také zapsané do výstupního souboru. Při dekódování pak vznikly kratší signály, než jaké byly na vstupu během kódování.

Následující čtyři testy počínaje „Pouze LP, řád pevně nastaven na 8“ a končící „Pouze LP, řád pevně nastaven na 20“ byly směřovány na testování různého nastavení řádu pro variabilní lineární predikci. Řád pro lineární predikci byl v každém z těchto testů nastaven na hodnotu uvedenou v názvu testu, všechny dílčí rámce pak byly zpracovávány výhradně s tímto řádem. Tyto testy pak byly prováděny nad testovacími signály. Výsledky těchto testů byly téměř identické, nejlépe pak dopadl test, kde byl řád nastaven napevno na hodnotu 12.

V předposledním testu „Pouze LP, max. řád zvýšen na 20“ v tabulce 5.3 byl nastaven maximální řád variabilní predikce na hodnotu 20. Zvýšením maximálního řádu vznikl prostor pro výběr vhodnějšího řádu lineární predikce, neboť v pomocném výpisu, jenž obsahoval zvolený řád predikce se občas objevovali případy kdy vybraný řád byl vyšší než původně nastavená hodnota maximálního řádu osm. Tento test dopadl velmi podobně jako testy s pevně nastaveným řádem, kupodivu však nedopadl nejlépe, pevně nastavený řád 12 měl stále lepší výsledky.

Poslední uvedený test „Pouze LP, použita interpolace sinc“ pak testoval nově přidanou metodu, jenž pomocí interpolace filtrem typu sinc vypočítávala zpoždění TDOA s vyšší přesností, než jaká byla perioda vzorkovací frekvence. Tento test pak téměř odpovídá finálnímu stavu této aplikace, opravou narušené bezztrátovosti již vnikla finální verze programu. Výsledný soubor měl podobnou velikost jako pevně nastavený řád 12, avšak došlo zde k porušení bezztrátovosti nástroje, jeho opravením se pak výsledný soubor mírně zvětšil, jelikož bezztrátovost stále má vyšší prioritu nežli velikost souboru.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo vytvořit nástroj pro bezeztrátovou kompresi zvukových dat pocházejících z mikrofonního pole, cíl bakalářské práce se podařilo úspěšně splnit. Jelikož data pochází z mikrofonního pole, první použitou metodou je výpočet zpoždění mezi kanály a následně jejich zarovnání, neboť pro zarovnané kanály lze použít další metody zvyšující finální kompresi dat. K výpočtu zpoždění TDOA neboli „time delay of arrival“, se pak využívá normalizovaná křížová korelace. Spolu s interpolací pak dovoluje vypočítat toto zpoždění s vyšší přesností, než je perioda vzorkovací frekvence. Na základě vypočítaných zpoždění se pak určuje také referenční kanál pro následující použité metody.

Úpravami knihovny kodeku FLAC, jenž byl v této práci použit jako referenční kodek a zároveň i jako základ aplikace, například použitím stejných koeficientů v lineární predikci pro všechny kanály, se pak ušetřily další bity v metadatech. Tyto metadata jsou ve výsledném zkomprimovaném souboru nezbytné pro správné dekódování dat do původní podoby. Lineární predikce se pak používá k vytvoření reziduálního signálu neboli LP rezidua, jenž je rozdílem původního a predikovaného signálu. Toto LP reziduum pak má mnohem nižší energii a tím je pomocí Riceova kódovaní zakódováno na mnohem méně bitů než původní signál. Zarovnání kanálů pak probíhá mezi lineární predikcí a Riceovým kódováním. LP reziduum referenčního kanálu se zarovná s LP reziduem aktuálního kanálu a tyto dva signály se opět odečtou, tím vznikne nový signál, v této práci také nazývaný sekundární reziduum, s ještě nižší energií, teprve ten je zakódovaný pomocí Riceova kódování. Výjimkou pak je referenční kanál, jehož LP reziduum je přímo zakódován Riceovým kódováním. Odstraněním úvodních vzorků na začátku každého rámce se zajistilo, že LP reziduum je vypočítáváno z celého rámce a tím i jednotlivá LP rezidua z navazujících rámců na sebe také navazují.

Přidání výpočtu zpoždění a interpolace způsobilo značné zpomalení celého kódování, rychlost výpočtu je s těmito metodami přibližně 18,5krát pomalejší nežli samotný referenční kodek. Pro výpočet zpoždění je v aplikaci použita paralelizace, bez ní, by rychlost výpočtu byla ještě několikanásobně pomalejší. Výsledná komprese pak také nedopadla dle očekávání, neboť výsledná velikost zkomprimovaného souboru při kódování vzorových dat vzrostla přibližně o 2,14 % oproti výstupnímu souboru z referenčního kodeku. Pravděpodobnou příčinou je nepřesné zarovnání kanálů díky čemuž pak výsledné sekundární reziduum nemá dostatečně nízkou energii oproti LP reziduu z odpovídajícího kanálu. Nepřesné zarovnání, popřípadě vyšší energii LP rezidua, potažmo i sekundárního rezidua, může způsobovat také nadměrný šum obsažený ve vzorových datech.

Zpřesnění výpočtu TDOA přidáním dalších mezi-vzorků při interpolaci by mohlo napomoct ke snížení celkové velikosti výstupního souboru, na druhou stranu by značně vzrostla délka výpočtu, který by tím byl ještě o to více pomalejší. Další možností by mohlo být případné vyrovnání energii obou LP rezidui, neboť jejich energie nemusí být nutně stejné nebo alespoň velmi podobné, vzniklé sekundární reziduum by pak mělo mít ještě nižší energii. Otázkou pak je, zda uložením hodnoty představující poměr energii by se nenavýšily metadata natolik, že by tato metoda byla neúčinná. Popřípadě, by se každý rámec mohl ještě rozdělit na kratší úseky a poměr energii pak vypočítat pro každý úsek samostatně, ovšem znamenalo by to také přidání více hodnot do metadat a opět se nabízí stejná otázka, zda by tato metoda nebyla neúčinná. Z mého osobního pohledu je však referenční kodek FLAC napsán zřejmě tak efektivně, že jej lze jen těžko vylepšit.

Literatura

[1] Černocký J.: *Zpracování řečových signálů – studijní opora*, Vysoké učení technické v Brně, Fakulta informačních technologií. [Online; navštíveno 09.12.2018]
URL http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf

[2] *Discrete-Time Signal Processing: The Levinson-Durbin Recursion*, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.
[Online; navštíveno 15.12.2018]
URL https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-341-discrete-time-signal-processing-fall-2005/lecture-notes/lec13.pdf

[3] Salomon D.: *Data Compression: the complete reference. 4th ed.* London: Springer, 2007. California State University, Computer Science Department. ISBN 978-1-84628-602-5.

[4] *Hlavní webová stránka kodeku FLAC*, Xiph.Org Foundation [Online; navštíveno 19.12.2018]
URL https://xiph.org/flac/

[5] Stark A.: *Knihovna C++ pro práci se soubory ve formátu wav* [Online; navštíveno 4.1.2019]
URL https://github.com/adamstark/AudioFile

[6] Yaroslavsky L. P.: *Signal sinc-interpolation: A fast computer algorithm. Bioimaging. 4.*, s. 225-231. DOI 10.1002/1361-6374(199612)4:4<225::AID-BIO1>3.0.CO;2-G.

[7] TKADLEC, J. *Lokalizace vzdáleného zdroje zvuku polem mikrofonů.* Brno, 2011, Bakalářské práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí bakalářské práce byl Ing. Zdeněk Havránek, PhD.
URL https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=40757

[8] GHARAVI, H. Conditional Run-Length and Variable-Length Coding of Digital Pictures. *IEEE Transactions on Communications [online]*. IEEE, 1987, Vol.35(6), s. 671-677. ISSN 0090-6778. DOI: 10.1109/TCOM.1987.1096817.

[9] *AMI Corpus Overview* [Online; navštíveno 27.4.2019]
URL http://groups.inf.ed.ac.uk/ami/corpus/overview.shtml