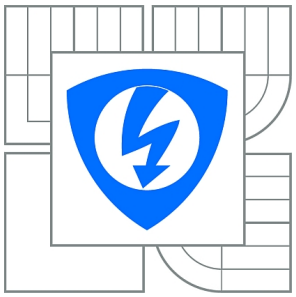
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

ROZPOZNÁVÁNÍ HUDEBNÍCH ZÁZNAMŮ

RECOGNITION OF MUSICAL RECORDINGS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

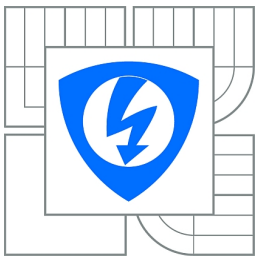
AUTOR PRÁCE — Bc. IGOR MASÁR

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. MILAN SIGMUND, CSc.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Igor Masár	***ID:***	119526
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Rozpoznávání hudebních záznamů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou formátů souborů s hudebními záznamy a s možnostmi jejich převodů. Vytvořte rešerši dostupných publikací o analýze hudebních signálů. Vytvořte a naprogramujte vybrané algoritmy pro určení základní melodie dané hudební nahrávky. Realizované metody a získané výstupy vzájemně porovnejte. Optimální algoritmus odlaďte a implementujte do autonomního systému, který bude automaticky rozpoznávat archivované hudební záznamy na základě zjednodušené centrální melodie.

DOPORUČENÁ LITERATURA:

[1] JAN, J. Číslicová filtrace, analýza a restaurace signálů. Brno: VUT v Brně, 2002.

[2] GUÉRIN, R. Velká kniha MIDI-standardy, hardware, software. Brno: Computer Press, 2005.

Termín zadání:	11.2.2013	***Termín odevzdání:***	24.5.2013

Vedoucí práce: prof. Ing. Milan Sigmund, CSc.
Konzultanti diplomové práce:

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá analýzou specifických zvukových signálů- hudbou. Jsou zde popsány základní metody analýzy hudebních signálů. Dále jsou zmíněny nejpoužívanější formáty hudebních souborů a možnost jejich vzájemného převodu. Jsou vysvětleny termíny z hudební teorie, které se v práci dále vyskytují. Jsou popsány a vytvořeny 3 způsoby detekce melodie. Na základě úspěšnosti detekce melodie z referenčních nahrávek je vybrán optimální algoritmus. Je vytvořeno uživatelské rozhraní v MATLAB GUI umožňující rozpoznávání nahrávek. Toto rozhraní je otestováno na několika melodiích.

KLÍČOVÁ SLOVA

Analýza signálů, hudba, wav, detekce tónu, autokorelace

ABSTRACT

This thesis analyzes the specific audio signal-music. It describes the basic methods of analysis of musical signals. The following are mentioned the most common music file formats and the possibility of cross transfer. There are explained terms of music theory, which are also present in this work. They are described and created three ways of detecting melody. It is selected optimal algorithm based on the successful detection of the reference melodies recordings. User interface is created in MATLAB GUI allows recognition of recordings. This interface is tested on few melodies.

KEYWORDS

Signal analysis, music, wav, tone detection, autocorrelation

MASÁR, I. *Rozpoznávání hudebních záznamů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2013. 51 s., 0 s. příloh. Diplomová práce. Vedoucí práce: prof. ing. Milan Sigmund, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Rozpoznávání hudebních záznamů 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 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.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce prof. ing. Milanu Sigmundovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

Úvod **1**

1 Analýza hudebních signálů **2**

1.1 Diskrétní Fourierova transformace ... 2

1.1.1 Rychlá Fourierova transformace ... 4

1.2 Časově frekvenční analýza ... 4

1.3 Korelogram ... 5

1.4 Kepstrum ... 5

1.5 Psychoakustická analýza ... 6

1.6 Publikace zabývající se zpracováním hudebních signálů ... 6

2 Počítačové formáty zvukových souborů **7**

2.1 PCM formát signálů ... 7

2.2 Formát souborů RIFF ... 7

2.3 Nekomprimovaný formát souborů WAV ... 8

2.4 Komprimované formáty souborů ... 8

2.4.1 MP3 ... 9

2.4.2 AAC ... 9

2.4.3 WMA ... 10

2.4.4 OGG Vorbis ... 11

3 Hudební teorie **12**

4 Algoritmy k určení melodie **14**

4.1 Korekce a úprava vstupního signálu ... 14

4.2 Detekce tempa ... 14

4.3 Segmentace a prahování ... 15

4.4 Spektrální metoda ... 17

4.5 Autokorelační metoda ... 19

4.6 Kepstrální metoda ... 21

4.7 Přiřazení tónů a prezentace výsledků ... 22

4.8 Porovnání metod ... 24

5 Databáze melodií a rozpoznání melodie 29

5.1 Detekce začátku melodie ... 30

5.2 Relativní vzdálenost tónů a korekce ... 30

5.3 Uložení do databáze a skladby v databázi ... 31

5.4 Rozpoznání skladby ... 31

6 Uživatelské rozhraní 33

6.1 Implementace algoritmů ... 33

6.2 Hlavní menu ... 34

6.3 Přidání skladby ... 35

6.4 Rozpoznání skladby ... 36

6.5 Databáze ... 37

6.6 Nastavení ... 38

6.7 Testování programu ... 39

7 Závěr 42

Literatura 43

Seznam symbolů, veličin a zkratek 45

Seznam příloh 46

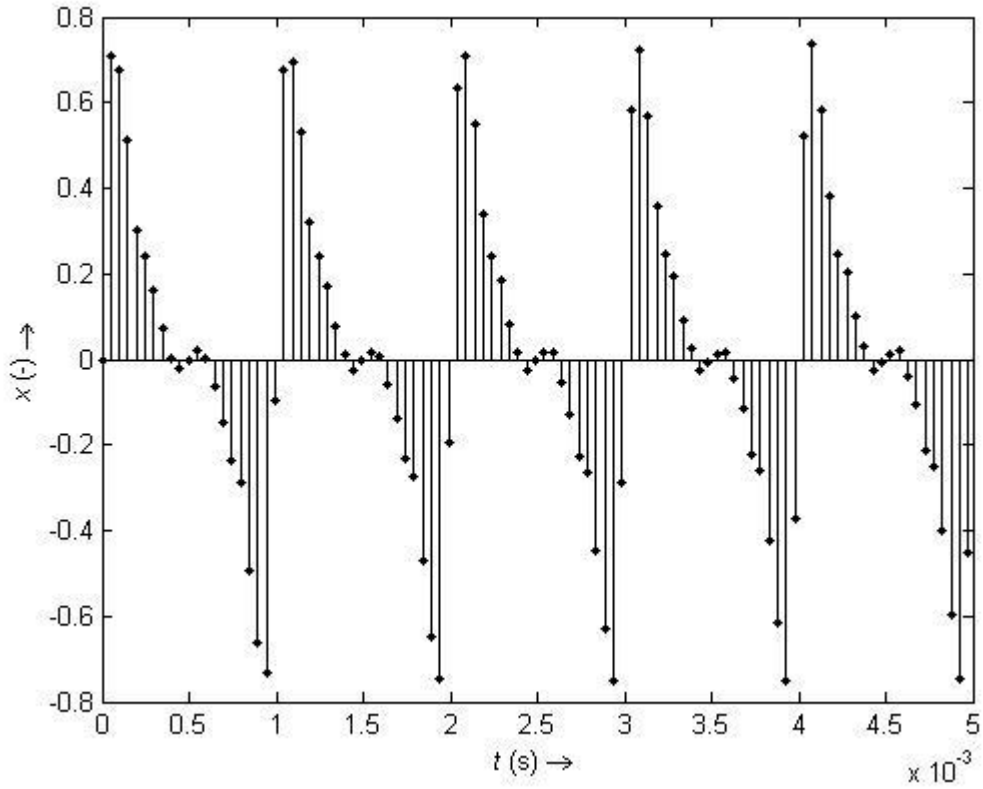
ÚVOD

V dnešní době obklopuje hudba člověka takřka na každém kroku. Hudba je vlastně zvuk, který má určitá pravidla a zákonitosti. Pokud se nejedná o „živou“ hudbu produkovanou hudebními nástroji či zpěvem, v drtivé většině případů je zdrojem digitálně zaznamenaný zvuk. Digitální zpracování hudby přináší celou řadu výhod, například prakticky neomezený počet reprodukcí dané nahrávky bez ztráty kvality, poměrně jednoduché korekce a úpravy zvuku dosažitelné i softwarovými metodami. Samozřejmě je zde i mnoho nevýhod, například zkreslení nedokonalostmi A/D a D/A převodníků, větší frekvenční šířka kanálu záznamu oproti analogové podobě, značné nároky na digitální systém (velká paměť, výpočetní výkon). Tyto nevýhody se časem snižují v důsledku neustálého vývoje daných komponent.[1]

Tato práce si klade za cíl vytvořit funkční algoritmus, který dokáže rozpoznat melodii ve skladbě a najít skladbu z databáze, která obsahuje daný úsek melodie.

1 ANALÝZA HUDEBNÍCH SIGNÁLŮ

Hudební signál je specifický zvukový signál. Pokud hovoříme o systému, který jakýmkoliv způsobem zpracovává tyto signály, v praxi bývá tento systém číslicový (digitální), z toho vyplývá, že i signál musí být v digitální podobě. Analogový zvukový signál lze nalézt pouze u jednoduchých systémů na ekvalizaci frekvenčního spektra daných signálů, systémů pro nelineární zkreslení používající jako nelineární prvek tranzistor či elektronku. Samozřejmě se objeví u digitálních systémů před vstupním A/D převodníkem, analogový signál z výstupu snímače zvuku a také za výstupním D/A převodníkem, kde dále putuje do výkonového zesilovače. Tato práce se proto dále zabývá pouze digitální podobou zvukových signálů.[2] Na obrázku 1.1 je ukázka, jak by mohla část jednoduchého hudebního signálu vypadat.



Obrázek 1.1: Diskrétní hudební signál v časové oblasti.

V následujících kapitolách jsou popsány nejpoužívanější způsoby analýzy takovýchto signálů.

1.1 Diskrétní Fourierova transformace

Diskrétní Fourierova transformace (DFT) vychází z poznatku, že každý signál konečný

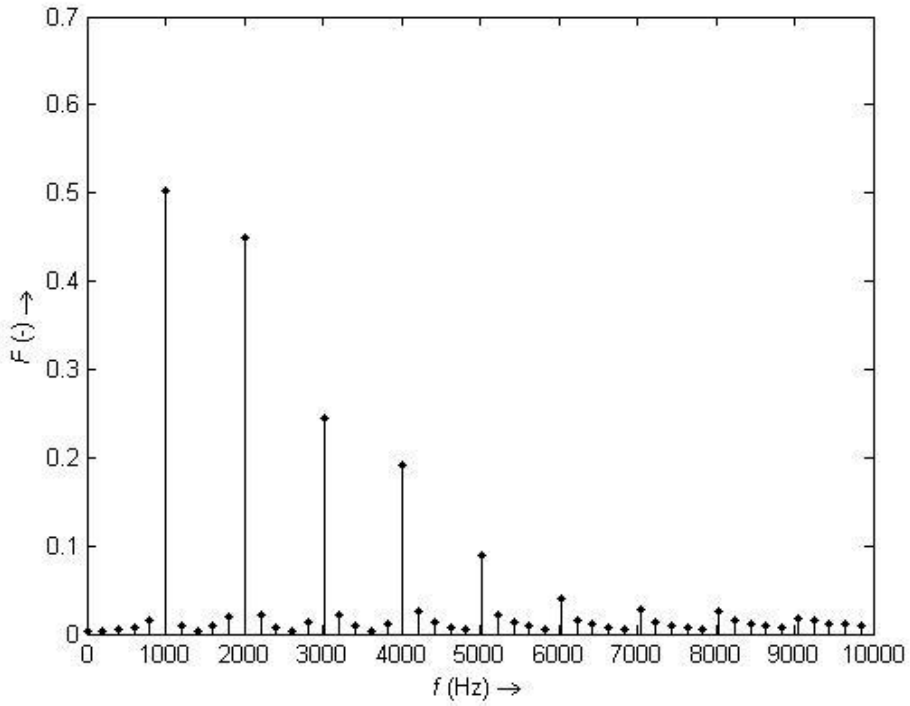
v čase lze vyjádřit součtem harmonických signálů, o různých parametrech (amplituda, frekvence, fáze). Definiční vztah této transformace:

$$F(k\Omega) = \sum_{n=0}^{N-1} s(nT) e^{-jk\Omega nT}, \tag{1}$$

a zpětná transformace:

$$s(nT) = \frac{1}{N} \sum_{k=0}^{N-1} F(k\Omega) e^{jnTk\Omega}, \tag{2}$$

kde *F* jsou komplexní hodnoty ve frekvenční oblasti, Ω je základní frekvence odpovídající frekvenčnímu rozlišení, *k* je celočíselný násobek této frekvence, *s* jsou hodnoty vzorků *n* v časové oblasti, *T* je perioda vzorkování, *e* je konstanta, *j* uvozuje označení pro imaginární část komplexních čísel. Platí, že počet výstupních hodnot v obou oblastech je stejný, proto pokud má vstupní signál více vzorků, frekvenční rozlišení bude větší, frekvenční rozlišení je nepřímo úměrné počtu vzorků signálu. Z definice je zřejmé, že se jedná o lineární transformaci. Výstupním hodnotám DFT zobrazených v grafu se obecně říká frekvenční spektrum. Toto spektrum je periodické a symetrické kolem poloviny vzorkovací frekvence a dále na celočíselných násobcích této frekvence. Na obrázku 1.2 je zobrazen modul frekvenčního spektra signálu z obrázku 1.1, zobrazena je oblast od 0Hz do poloviny vzorkovací frekvence.



Obrázek 1.2: Modulové frekvenční spektrum hudebního signálu.

Zajímavý je vztah DFT k Z-transformaci-výstupní hodnoty DFT určitého signálu jsou hodnoty Z-transformace stejného signálu, které jsou rovnoměrně rozloženy na jednotkové kružnici v Z-rovině.

Lze konstatovat, že většina analýz ve spektrální (frekvenční) oblasti vychází z DFT.[3]

1.1.1 Rychlá Fourierova transformace

Rychlá Fourierova transformace (FFT) je pouze označení pro efektivnější výpočet DFT, výsledné hodnoty jsou tedy stejné jako při použití DFT. Využívá se možnosti využít pro výpočet DFT vztahu:

$$F_k = \sum_{n=0}^{N-1} s_n W^{nk}, W = e^{-j\frac{2\pi}{N}}, \tag{3}$$

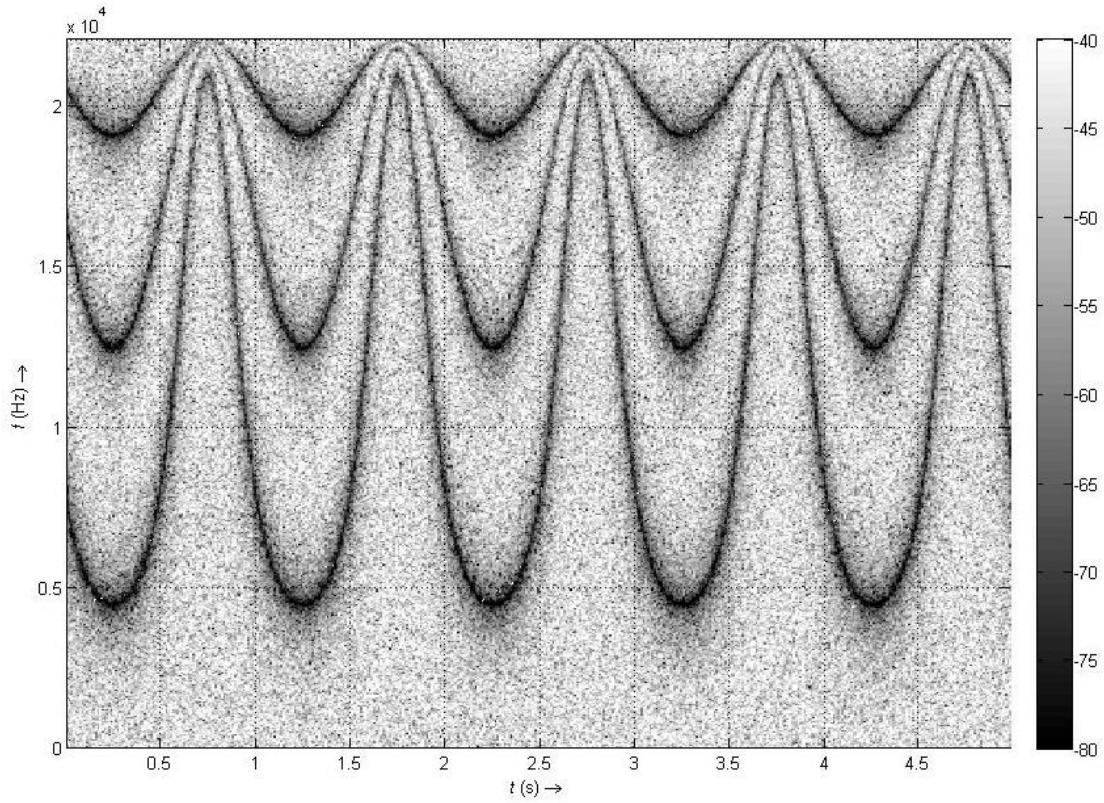
W_n jsou hodnoty koeficientů, které je možno odvodit z tabulky hodnot funkce sinus pro čtvrtinu periody. Byly vyvíjeny různé algoritmy pro výpočet FFT, důvodem tohoto počínání byla snaha o snadnou implementaci DFT do mikropočítačů, FPGA apod. Pro použití FFT v prostředí MATLAB je nutné, aby signál měl 2^N vzorků, přičemž N je celé číslo.[5]

1.2 Časově frekvenční analýza

Jedná se v podstatě o řadu tzv. krátkodobých spekter v čase, nazývanou spektrogram, kde na ose x je čas, na ose y je frekvence a amplitudy na daných frekvencích jsou dány barvou, případně odstínem šedi. Prakticky se signál rozdělí na M úseků zpravidla o stejné délce N, přičemž se jednotlivé úseky mohou navzájem částečně překrývat. Na tyto úseky se použije DFT. Jak již bylo řečeno v předešlé kapitole, délka signálu určuje frekvenční rozlišení, pro dobré frekvenční rozlišení je tedy potřeba provést DFT úseku M s co největší délkou N, to ale na druhou stranu způsobí nízké časové rozlišení, je tedy třeba najít kompromis mezi těmito požadavky, což se liší podle účelu analýzy, jednou z možností zlepšení těchto vlastností je použít překrytí vzorků. Druhý problém vzniká při rozdělování segmentů M, pokud se jednoduše pouze rozdělí, jedná se o konvoluci segmentu M s obdélníkovým oknem. Obdélníkové okno má ve frekvenční oblasti tvar funkce sinc, proto se výsledku objeví spektrum daného segmentu M vynásobené funkcí sinc, tento jev se nazývá prosakování spektra. Obecně tento jev vzniká, protože segment M s bázovou funkcí DFT (komplexní exponenciálou) není ortogonální. Pro potlačení této parazitní vlastnosti se používají okna různých typů, nejčastěji to jsou: Hammingovo, Hannovo, Blackmanovo, trojúhelníkové. [3]

Pro zlepšení vlastností mezi délkou okna M a časově-frekvenčním rozlišení bylo vyvinuto několik algoritmů, které využívají např. adaptivní délku okna. Mezi takové algoritmy patří například Metoda minimální energie spektrálního prosakování či Aproximovaná diskrétní Zolotarevova transformace, která používá jako bázovou funkci selektivní sinus a kosinus (zsin a zcos).[4]

Na obrázku 1.3 je ukázka spektrogramu bílého šumu na výstupu efektu Phaser.



Obrázek 1.3: Ukázka spektrogramu.

1.3 Korelogram

Jako jediná z dosud probraných analýz není ve frekvenční, nýbrž v časové oblasti, tedy nepoužívá se pro její provedení DFT. Jak název napovídá, používá se pro analýzu korelační respektive autokorelační funkce. Výstupem autokorelační funkce je míra podobnosti vzájemně posunutého stejného signálu. Definice této funkce pro jednostrannou korelaci je

$$R(k) = \frac{1}{N} \sum_{i=0}^{N-k-1} s(i)s(i+k), \tag{4}$$

přičemž N je počet vzorků signálu, $s(i)$ je hodnota vzorku signálu. Korelogram se potom získá obdobným způsobem jako u spektrogramu. Jako vstupy autokorelační funkce se použijí segmenty signálu M a poté se umísti v čase za sebe.[3]

1.4 Kepstrum

Kepstrum by se v podstatě dalo nazvat jako meziprodukt při homorfické slepé dekonvoluci. Jedná se o nelineární funkci, kdy se počítá logaritmus z modulu diskrétní Fourierovy transformace daného signálu a následně je na tento vektor aplikována zpětná diskrétní Fourierova transformace, což lze zapsat:

$$C_R(K) = \Re\{IFFT(\ln|FFT(s[i])|)\ \}, \quad (5)$$

kde C_R jsou hodnoty reálného kepstra, $\Re$ označuje reálnou část, *IFFT* značí zpětnou Fourierovu transformaci, *ln* je přirozený logaritmus, *FFT* Fourierova transformace a *s[i]* vzorky signálu v čase.[3]

1.5 Psychoakustická analýza

U psychoakusticé analýzy je určité skupině osob přehrána zvuková nahrávka či více nahrávek, tato skupina poté subjektivně hodnotí vjemy, případně porovnává různé nahrávky. Výsledné odpovědi jsou statisticky zpracovány, jedním z využití této analýzy je vytvoření psychoakustického modelu zohledňující získané poznatky. Toho je využito například u kódování MPEG1[2].

1.6 Publikace zabývající se zpracováním hudebních signálů

Publikace, které se zabývají detekcí melodie v jednohlasé skladbě, jsou např. **PLEVA, P.** ***Převod zvuku do MIDI formátu.*** Autor využívá autokorelační funkce, Fourierovu transformaci a reálné kepstrum, které aplikuje na segmenty hudebního signálu, počet segmentů je třeba zadat ručně a jeho délka odpovídá době trvání noty. Podobná publikace je **KRUPIČKA, J.** ***Převod not jednohlasé melodie ze zvukového signálu do protokolu MIDI.*** Další práce, která se zabývá identifikací nástrojů s velkým rozsahem tónů je E. Benetos, *et al*, "**Large scale musical instrument identification**," *Proceeding of 4th Sound and Music Computing Conf.*, 2007. Identifikaci nástrojů provádí umělá neuronová síť.

2 POČÍTAČOVÉ FORMÁTY ZVUKOVÝCH SOUBORŮ

V této kapitole jsou probrány nejpoužívanější formáty používané u zvukových souborů, dále způsob kódovaní zvukových souborů. Existují dva základní typy, první z nich je MIDI formát, jedná se o standardizovaný formát, kde obsah souboru netvoří vzorky hudebního signálu, ale záznam obdobný notovému zápisu. Výhoda tohoto tytu je malá velikost souboru, snadná změna tóniny a další hudební operace. Nevýhodou je kvalita závislá na způsoby syntézy tónů daných hudebních nástrojů. Nejlevnější varianty pracují pouze na principu tónových generátorů, dražší obsahují nahrané vzorky reálných nástrojů.[6] V poslední době vytlačuje tento formát VSTi (Virtual studio instruments).

Druhý formát jsou zakódované vzorky hudebního signálu daného formátu. V této kapitole bude probrán tento typ. V podkapitole 2.1 je popsán nejběžnější typ kódování samotných diskrétních vzorků do digitální podoby. V dalších podkapitolách jsou již uvedeny nejběžnější typy formátů souborů.

2.1 PCM formát signálů

Pulsně kódová (PCM) modulace je v současnosti stále nejpoužívanější formát modulace nejen audio souborů. Důvodem je její snadná realizace například pomocí R-2R sítí. Princip této modulace je vzorkování frekvencí f_s vstupního (analogového) v časově stejně vzdálených (ekvidistantních) okamžicích do n-bitového slova o q úrovních (kvantizačních hladinách) dané vztahem:

$$q = 2^n - 1, \tag{6}$$

přičemž samozřejmě je nutné dodržet Shannonův teorém, který říká, že vzorkovací frekvence f_{vz} musí být minimálně dvojnásobná jako nejvyšší harmonická frekvence obsažená ve vstupním signálu. [7]Příklad signálu tohoto formátu je na obrázku 1.1. Méně časté je použití modulace ADPCM, což je adaptivní PCM modulace.

2.2 Formát souborů RIFF

RIFF (Resource Interchange File Format) je jeden z nejpoužívanějších systémů ukládání souborů ve výpočetní technice nejčastěji používaný u audio a video souborů. Klíčovým prvkem je blok „chunk“, který má na začátku bloku 4 znakovou identifikaci o jaký blok se jedná a za ním následuje 32 bitová délka bloku. Tyto bloky je možné do sebe vnořovat, v praxi se používá maximálně 3 do sebe vnořené bloky. Výhodou tohoto formátu je, že pokud systém pracuje se souborem v tomto formátu a nezná některý blok, může jej jednoduše přeskočit, toho se dá využít ke kompatibilitě při zpracování v různých systémech. Struktura bloku informující o formátu souboru je v tabulce 2.1.

Datový typ	Název	popis
FOURCC	*ckid*	Identifikátor bloku
DWORD	*cksize*	Velikost bloku
WORD	*wFormatTag*	Formát kódování dat
WORD	*nChannels*	Počet audio kanálů
DWORD	*nSamplesPerSec*	Počet vzorků za sekundu
DWORD	*nAvgBytesPerSec*	Přenosová rychlost
WORD	*nBlockAlign*	Velikost datové jednotky
WORD	*wBitsPerSample*	Počet bitů na vzorek

Tabulka 2.1: Struktura Formát RIFF bloku.

[1]

2.3 Nekomprimovaný formát souborů WAV

Formát wav (Wave form audio format) používá k zápisu RIFF formát. Jedná se o bezeztrátový formát. Vzhledem k tomu, že tento formát není nijak komprimován, je možné jednoduše pracovat s libovolnými vzorky signálu, proto je tento formát vhodný k dalšímu zpracování signálů. Soubor obsahuje informace o vzorkovací frekvenci a bitové hloubce daného signálu a další informace v bloku Format RIFF blok, jehož obsah je uveden v tabulce 2.1, dále různé doplňkové informace potřebné u různých typů přehrávačů pro korektní přehrání souboru. Tyto různé doplňkové informace jsou uloženy v samostatných blocích, pokud některé nejsou třeba, nemusí být obsaženy, samotná data jsou také v samostatném bloku. Teoreticky formát wav nemusí být nekomprimovaný, dojde k tomu v případě, že je místo PCM modulace použita jiná, např. ADPCM.

Speciální formát, kde je obsah souboru pouze signál v PCM modulaci se nazývá raw formát. [1]

2.4 Komprimované formáty souborů

Důvodem jejich vzniku byla úspora místa, protože nekomprimovaný formát zabírá příliš mnoho místa v paměťovém médiu. Základní rozdělení je na bezeztrátové a ztrátové. Bezeztrátové mají výhodu, že zvuková kvalita je identická jako nekomprimovaný formát. Nevýhodou je menší dosažitelný kompresní poměr. Způsob komprese je založený na matematických metodách vyjádření různých informací. [6] Do této kategorie patří například formát FLAC (Free Lossless Audio Codec).

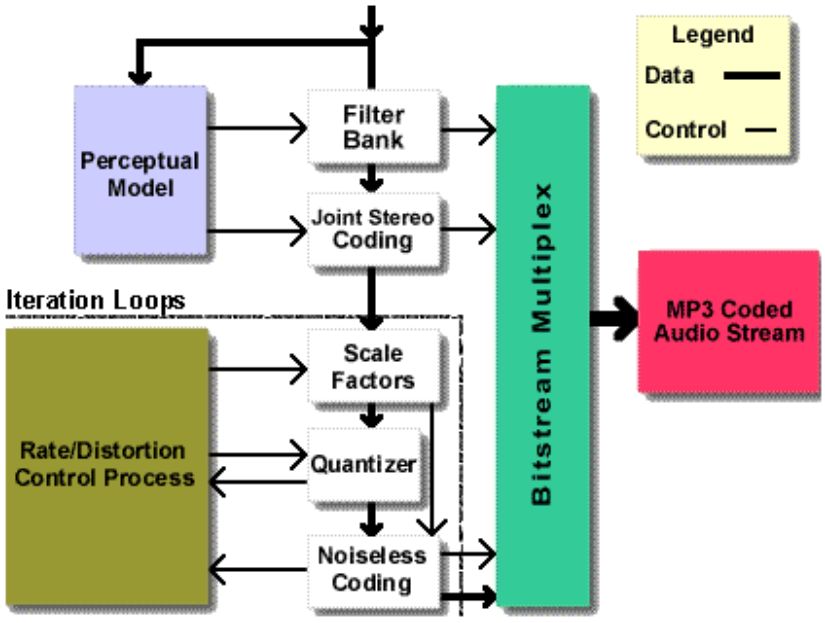
Ztrátové formáty jsou více užívané, zde se kromě metod bezeztrátové komprese

přidává odstranění redundantních dat v zvukovém souboru určené nedokonalostmi lidského ucha a vnímání zvuku člověkem. Tyto informace jsou získané z psychoakustické analýzy probrané v kapitole 1.4. [2]

V následujících podkapitolách jsou probrány nejběžněji používané ztrátové formáty.

2.4.1 MP3

Tento dnes stále ještě nejpopulárnější standardizovaný formát mp3 (ISO-MPEG Audio Layer-3) vznikl již v roce 1987. Blokové schéma tohoto kodéru je na obrázku 2.1.

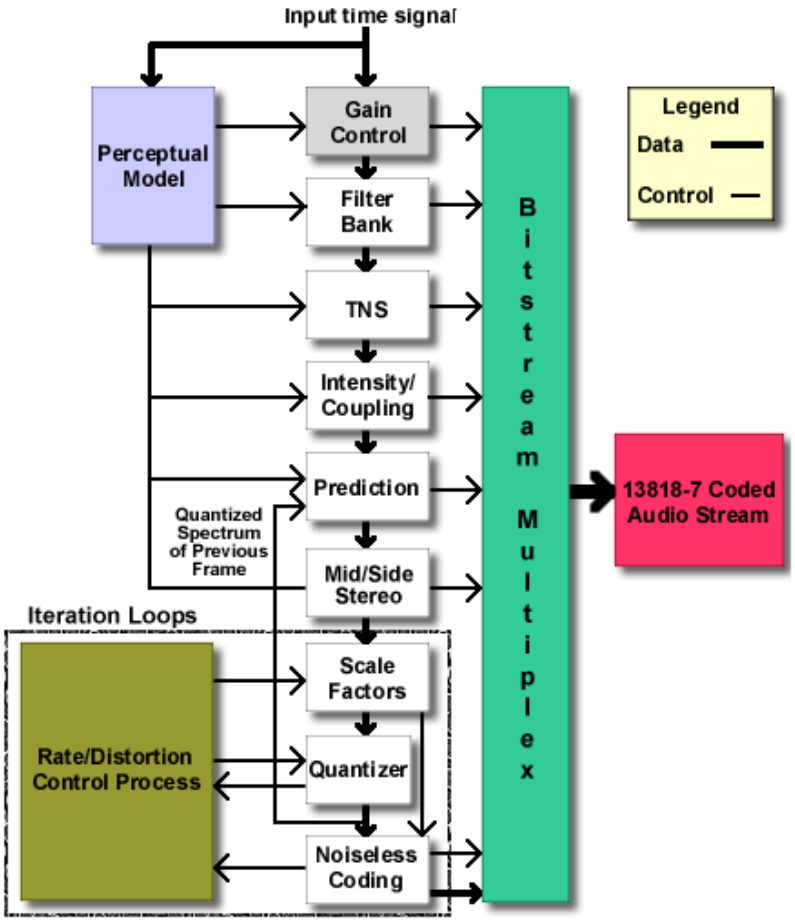


Obrázek 2.1: Blokové schéma kodéru mp3, převzato z [8].

Vstupní signál je bankou filtrů typu pásmová propust rozdělen do 32 frekvenčních pásem. V každém tomto pásmu je určen výkon, tyto hodnoty výkonů jsou poté porovnány s hodnotami v psychoakustickém modelu pro určení prahu maskování a počtu bitů respektive bitovou hloubku daného pásma, platí, že pásma s větším výkonem jsou zakódována do vyšší bitové hloubky. Tyto hodnoty jsou poté zakódovány Hufmannovým kódováním, což je v podstatě algoritmus bezeztrátové komprese. Mp3 soubor se skládá z rámců, kdy každý tento datový rámec obsahuje svou vlastní hlavičku a je tedy zcela nezávislý, výhoda této struktury je, že pokud by některý rámec byl nečitelný, soubor i tak půjde přehrát. Formát mp3 umožňuje maximálně dva zvukové kanály (stereo), neumí pracovat s vícekanálovým zvukem. [9]

2.4.2 AAC

Tento hudební formát AAC (Advanced audio coding) byl vyvinut na základě mp3 a dokončen byl v roce 1997. AAC spadá do standardu MPEG 2 a 4. Jedná se o jeden z nejefektivnějších ztrátových formátů. Blokové schéma kodéru AAC je na obrázku 2.2 a princip je obdobný jako u mp3. [8]



Obrázek 2.2: Blokové schéma kodéru AAC, převzato z [8].

Vstupní signál se rozdělí na bloky o délce 2048 vzorků, překrývání oken bylo zvoleno na 1024 vzorků, tedy 50%. Je možno velikost okna zmenšit na 256 vzorků, v případě potřeby. Na tyto bloky je aplikována Modifikovaná Kosinova transformace (MDCT). AAC obsahuje propracovanější psychoakustický model oproti mp3. Dále je v tomto formátu použit algoritmus percepční náhrady šumu (PNS), který má za úkol v signálu najít šumové složky, které následně tyto složky nekóduje v podobě vzorků, nýbrž parametrickým popisem. Jedná se tedy o zápis stochastického signálu. Protože se jedná o vylepšený standard mp3, je zřejmé, že bude mít lepší vlastnosti, výhoda oproti mp3 je například kvalita signálu při lepším kompresním poměru a podpora vícekanálového zvuku. [10]

2.4.3 WMA

Formát wma (Windows media audio), jehož tvůrcem je společnost Microsoft, má obdobné vlastnosti jako mp3. Je druhý nejrozšířenější formát po mp3 a to díky dominantní pozici společnosti Microsoft na trhu IT a multimédii. Používá formát souboru ASF (Advanced Systems Format). [10]

2.4.4 OGG Vorbis

Jedná se o nejpoužívanější opensource (software s otevřeným zdrojovým kódem) formát hudebních souborů. Označení OGG značí souborový kontejner, podobně jako RIFF v kapitole 2.2. U tohoto formátu se používá dělení vstupního signálu do bloků o délce 2^N vzorků, přičemž N je celé číslo a výsledný počet vzorků musí ležet mezi 64 až 8192 vzorků včetně. MDCT se aplikuje na okno o dvojnásobné velikosti. Bitová hloubka jednotlivých zastoupených frekvencí ve výsledném souboru je dána frekvenční charakteristikou, označovanou jako floor, vstupního bloku kódovanou lineární predikcí.[10]

3 HUDEBNÍ TEORIE

Nutnou prerekvizitou pro následující kapitoly jsou znalosti elementární hudební teorie. V této kapitole jsou tedy vysvětleny pojmy, které se dále v práci vyskytují. Jsou zde použita určitá zjednodušení, která z pohledu hudební teorie nemusí být úplně korektní, ale z technického pohledu jsou zcela dostačující.

Tón

Je to obecně vlnění, které má konstantní frekvenci. Popsán je sílou, výškou, délkou a barvou. **Síla** vyjadřuje amplitudu tohoto vlnění, nejčastěji se ke kvantitativnímu rozlišení používá pouze slovní popis, případně není uvedeno vůbec. **Výška** tónu může být vyjádřena buď absolutně frekvencí první harmonické, nebo relativně, kdy se uvádí poměr frekvencí daného tónu k dalšímu tónu, tento poměr, pokud je vyjádřen číslem, se nazývá interval. Poslední možností, nejčastěji užívanou, je uvedení názvu tónu a příslušné oktávy. Co je to oktáva, je vysvětleno v následující podkapitole. Dalším parametrem je **délka** tónu, ta vyjadřuje, jak dlouho tón trvá, tento parametr úzce souvisí s tempem skladby. Jedním ze způsobu zápisu je zlomek, který znamená jak dlouho je nota hraná oproti referenční hodnotě. Posledním parametrem je **barva** tónu, z technického hlediska jsou to obsažené vyšší harmonické. Z poměru amplitud těchto vyšších harmonických je možné rozpoznat, o jaký typ hudebního nástroje se jedná, přičemž platí, že první harmonická nemusí mít největší amplitudu, výrazné amplitudy se nazývají formanty a mají zásadní vliv na barvu nástroje. [13]

Tónová soustava, ladění

Tónová soustava, tedy tóny používané v klasické i moderní evropské hudbě, jsou pojmenovány: c, cis, d, dis, e, f, fis, g, gis, a, ais, h. Za těmito písmeny ještě může být uveden dolní index, do které oktávy daný tón patří. Oktáva je název pro interval, kdy jsou dva tóny v poměru 2:1. Těchto oktáv je celkem 10. Tóny odpovídající tomuto intervalu mají vždy stejný název, rozdíl jejich indexů je 1. Interval dvou sousedních tónů se nazývá malá sekunda, často se používá také termín půltón, číselná hodnota tohoto intervalu se liší dle použitého typu ladění, stejně tak se liší absolutní hodnoty frekvencí tónů. Nejpoužívanějším typem je temperované ladění. Hlavní důvod, proč se toto ladění používá v drtivé většině skladeb je, že skladba může využívat všech tónů ze všech oktáv, aniž by zněla rozladěně. Interval mezi půltóny je konstantní a odpovídá hodnotě:

$$q = \sqrt[12]{2}. \tag{7}$$

V tabulce 3.1 jsou uvedeny frekvence jednotlivých tónů.

oktáva	0	1	2	3	4	5	6	7	8	9
název tónů	frekvence tónů(Hz)									
c	16,35	32,7	65,4	130,8	261,6	523,2	1046,4	2092,8	4185,6	8371,2
cis	17,32	34,64	69,29	138,58	277,16	554,31	1108,62	2217,24	4434,49	8868,98
d	18,35	36,70	73,41	146,82	293,64	587,27	1174,54	2349,09	4698,18	9396,35
dis	19,44	38,89	77,77	155,55	311,10	622,19	1244,39	2488,77	4977,55	9955,09
e	20,60	41,20	82,40	164,80	329,60	659,19	1318,38	2636,76	5273,53	10547,05
f	21,82	43,65	87,30	174,60	349,19	698,39	1396,78	2793,55	5587,11	11174,21
fis	23,12	46,24	92,49	184,98	369,96	739,92	1479,83	2959,67	5919,33	11838,66
g	24,50	48,99	97,99	195,98	391,96	783,91	1567,83	3135,66	6271,31	12542,63
gis	25,95	51,91	103,82	207,63	415,26	830,53	1661,06	3322,11	6644,23	13288,45
a	27,50	54,99	109,99	219,98	440,00	879,91	1759,83	3519,66	7039,31	14078,62
ais	29,13	58,26	116,53	233,06	466,12	932,24	1864,47	3728,95	7457,89	14915,78
h	30,86	61,73	123,46	246,92	493,84	987,67	1975,34	3950,68	7901,36	15802,72

Tabulka 3.1: Frekvence tónů v temperovaném ladění.

Referenční hodnota pro tón a ve 4. oktávě je 440Hz od níž jsou odvozeny ostatní frekvence, tuto hodnotu určuje mezinárodně uznávaná norma. Řada tónů v jedné oktávě, kde jsou zastoupeny všechny její tóny, tedy je mezi nimi interval půltónu se nazývá **chromatická stupnice**.[11][13]

Tempo

Určuje jak rychle je skladba hraná, tedy jaký čas trvají jednotlivé tóny. Vyjadřuje se buď slovně, případně číselně, kdy se nejčastěji uvádí počet čtvrťových not za minutu, tzv. BPM(beats per minute). [11]

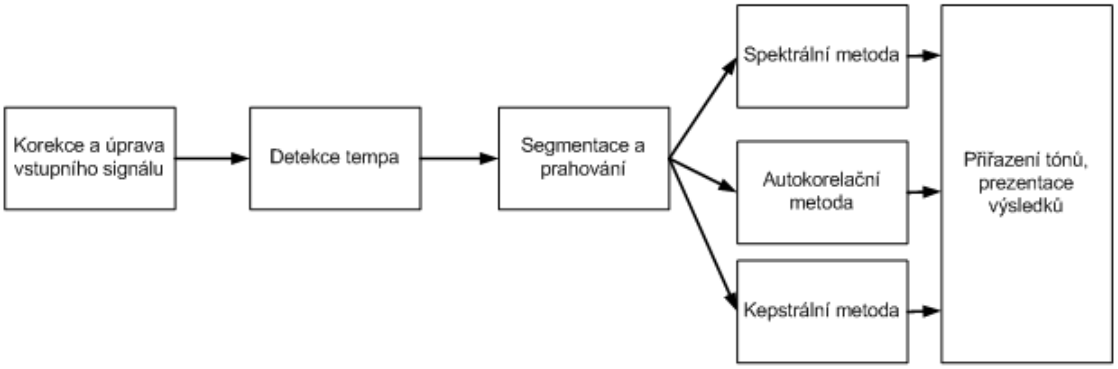
Jednohlasá a vícehlasá melodie

Jednohlasá melodie je posloupnost tónů, které hrají v čase za sebou, kdy alespoň jeden z tónů má jinou výšku. Melodie může obsahovat i tzv. **pomlky**- jedná se o technicky řečeno tón s nulovou amplitudou, tedy jejím parametrem je pouze délka.

Vícehlasá melodie má vlastnosti prakticky stejné jako jednohlasá, rozdíl je pouze v tom, že alespoň po dobu nejkratší použité délce noty hraje více tónů ve stejný čas. [11]

4 ALGORITMY K URČENÍ MELODIE

V této kapitole jsou vysvětleny algoritmy použité k určení melodie ve zvukových souborech. Hlavní ovládací skript napsaný v programovém prostředí MATLAB obsahuje bloky uvedené v blokovém schématu na obrázku 4.1.



Obrázek 4.1: Blokové schéma celého algoritmu.

V podstatě se vstupní signál rozdělí na segmenty odpovídající jednotlivým tónům. Poté je detekována perioda v jednotlivých segmentech, jsou použity 3 metody a všechny vychází z metod pro detekci základní periody řeči. U všech se detekce provádí v cyklu, kdy se zpracují jednotlivé segmenty vždy zvlášť. Následně jsou přiřazeny odpovídající tóny jednotlivým frekvencím a zobrazeny detekované výsledky. Podrobněji jsou jednotlivé dílčí bloky probrány v následujících podkapitolách.

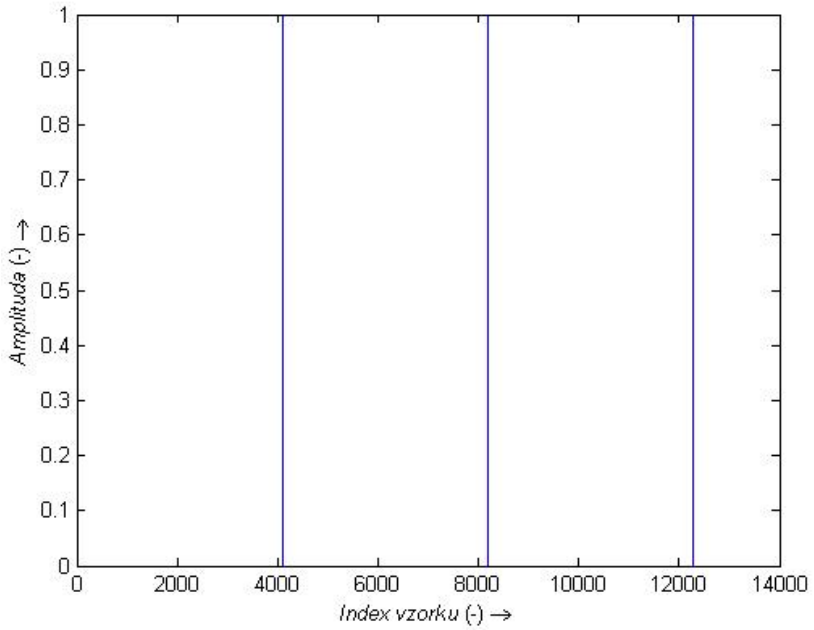
4.1 Korekce a úprava vstupního signálu

V této části se načítá vstupní wav soubor do proměnné, následně se zkontroluje, zda se jedná o monofonní, případně vícekanálový zvuk, v případě vícekanálového zvuku se provede sloučení (sečtení) kanálů do jednoho monofonního signálu, to zaručí, že se budou zpracovávat skutečně všechny tóny v nahrávce, někdy se totiž např. u stereo nahrávky (dvoukanálový zvuk-levý a pravý kanál) pro dosažení prostorového efektu, používá střídání levého a pravého kanálu. Následně se vypočítá stejnosměrná složka pomocí matlabovské funkce `mean` a ta se odečte.

4.2 Detekce tempa

Tato část má za úkol zjistit tempo analyzované skladby. Vstupní signál se nejprve rozdělí do 6 frekvenčních pásem. Tato pásma jsou 0-200Hz, 200-400Hz, 400-800Hz, 800-1600Hz, 1600-3200Hz a poslední 3200Hz-polovina vzorkovací frekvence. To je z toho důvodu, že analyzovaný signál může obsahovat různé nástroje a ty mohou mít různý frekvenční rozsah. Následně se pro jednodušší zpracování jednotlivé pásmové signály jednosměrně usměrní, což se prakticky provede funkcí pro absolutní hodnotu. Poté se extrahuje obálka, to se provádí pomocí dolní propusti, konkrétně se zde použilo násobení frekvenčního spektra jednotlivých pásem s pravou polovinou Hannova okna,

což v časové oblasti odpovídá konvoluci těchto dvou signálů. Poté se vyfiltrované pásmové signály převedou zpět do časové oblasti. Nyní se spočítá rozdíl dvou sousedních vzorků v každém pásmu, přičemž do výstupní proměnné, která se dále zpracovává, se zapíše pouze kladný rozdíl a na zbylých pozicích jsou nuly. Tempo je v podstatě pravidelný důraz zvuku, tedy by vlastně mělo odpovídat rozdílu indexů výstupního vektoru, kde je nejvyšší rozdíl dvou vzorků. V posledním kroku, který je nejvíce výpočetně náročný, se porovnává energie v jednotlivých pásmech s hřebenovým filtrem. To probíhá v cyklu, kdy se pokaždé vygeneruje speciální filtr, který má nenulové hodnoty tak, aby jejich vzdálenost odpovídala danému tempu, tímto filtrem se filtrují postupně všechna pásma rozdílového signálu. Na obrázku 4.2 je zobrazen tento filtr v časové oblasti pro tempo 120BPM.



Obrázek 4.2: Speciální filtr v časové oblasti odpovídající tempu 120BPM.

Je tedy potřeba projít v cyklu všechny možná tempa, což znamená od 60 do 240. Jako tempo se vybere ta hodnota, při které má výsledná energie nejvyšší hodnotu. Pro menší výpočetní náročnost se filtr i dané pásmo převede do frekvenční oblasti, kde se filtr i dané pásmo mezi sebou násobí (což odpovídá konvoluci)[14].

4.3 Segmentace a prahování

Na základě získané číselné hodnoty tempa je vstupní signál rozdělen na jednotlivé segmenty. Algoritmus pro určení tempa najde hodnotu tempa odpovídající nejkratšímu tónu, což znamená, že v každém segmentu je právě. Počet vzorků v každém segmentu je konstantní a lze jej určit ze vztahu:

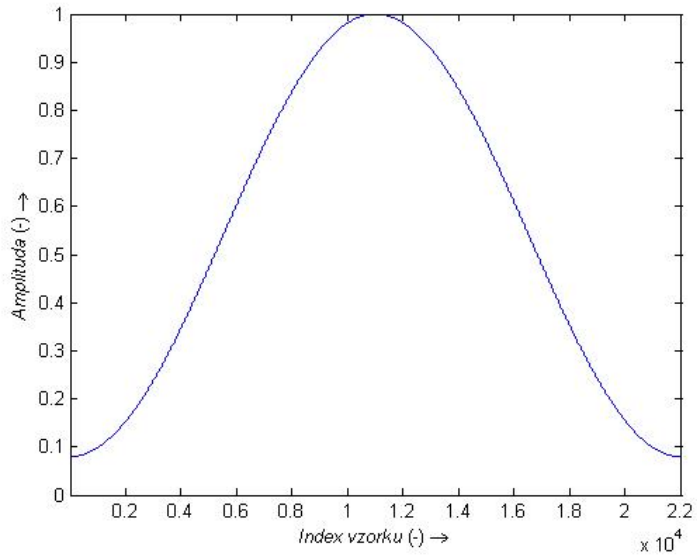
$$segment_{length} = \frac{60 \cdot f_s}{tempo}, \tag{8}$$

kde f_s je vzorkovací frekvence, jíž byl vstupní signál vzorkován a *tempo* je číselná hodnota získaná z předešlého bloku. Celkový počet segmentů daného vstupního signálu

je potom:

$$n_{seg} = \frac{s_{length}}{segment_{length}}, \tag{9}$$

kde s_{length} je počet vzorků vstupního signálu. Poté se testuje, zda délka vstupního signálu je celočíselným násobkem délky segmentu, pokud není, doplní se vstupní signál o patřičný počet nul. Jednotlivé segmenty se následně váhují Hammingovým oknem o stejné délce jako segment. Na obrázku 4.3 je zobrazeno, jak toto okno vypadá pro segment při tempu 120BPM a vzorkování 44,1kHz.



Obrázek 4.3: Hammingovo okno pro segment při tempu 120BPM.

Tento typ okna byl vybrán experimentálně jako nejvhodnější. Váhování je použito, protože hrané tóny nemusí být úplně synchronní s detekovaným tempem, může se tedy stát, že v jednom segmentu může být na začátku nebo na konci již tón s jinou výškou, nebo v závislosti na hraném nástroji může doznívat předešlý tón. Hammingovo okno tedy potlačí amplitudu těchto případných sousedních tónů. Segmenty jsou uloženy v matici *segg*, kde jednotlivé segmenty tvoří sloupce této matice, tedy počet řádků odpovídá délce segmentů a počet sloupců odpovídá počtu segmentů. Následně se vypočítá průměrná energie vzorku daného segmentu podle vztahu:

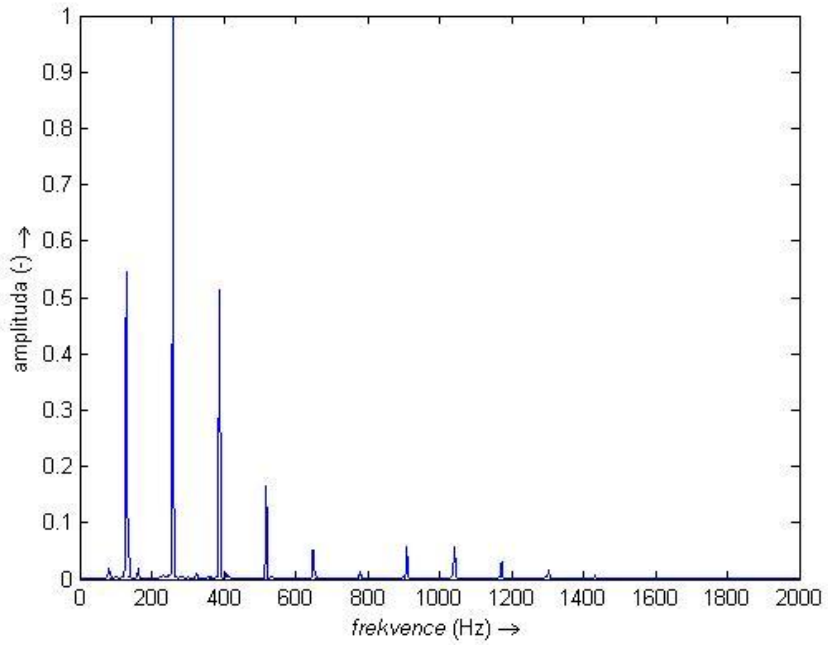
$$E = \frac{\sum_{i=1}^{segment_{length}} segg_{ij}^{\,2}}{segment_{length}}, \tag{10}$$

kde *i* je řádek matice segg a *j* je odpovídající sloupec. Tento výpočet slouží k detekci pomlk, protože i během pomlky je v signálu zaznamenán šum snímacího zařízení, případně různé rušivé zvuky z nahrávací místnosti. Pokud je průměrná energie vzorku menší než prahová hodnota, která byla experimentálně určena a je možné ji případně v aplikaci přizpůsobit, je tento segment považován za pomlku a hodnoty tohoto segmentu jsou nahrazeny nulami. Prahová hodnota byla určena za pomoci nahrávky, kde bylo nahráno ticho - tedy pouze šum a ruchy, z této nahrávky byla

zjištěna průměrná energie vzorku. Prahová hodnota byla testováním zvolena jako 200 násobek referenční hodnoty, tedy o 23dB vyšší.

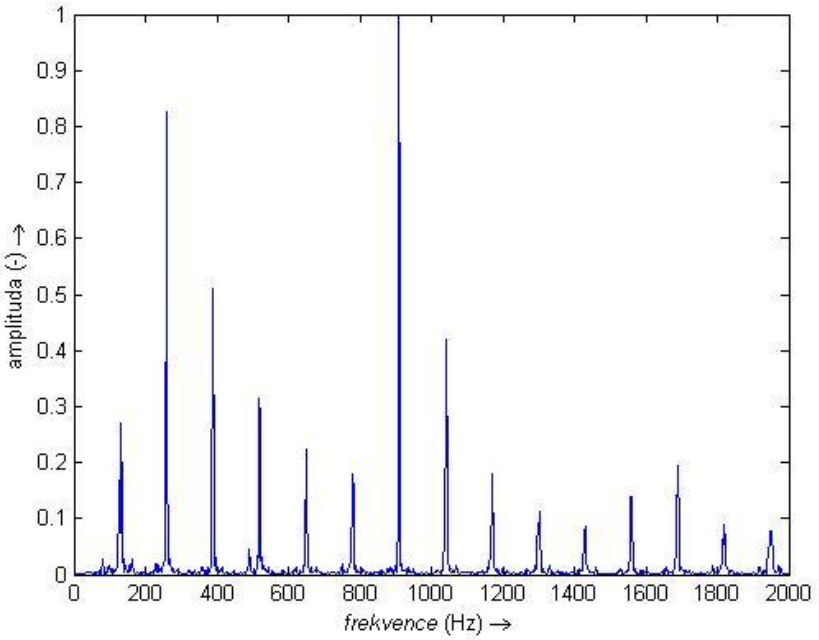
4.4 Spektrální metoda

Tato metoda vychází z DFT. U této metody jako jediné není třeba testovat, zda daný segment není pomlka, protože pro nulové vzorky v čase budou výstupní hodnoty transformace také nulové. Na každý segment se aplikuje diskrétní Fourierova transformace matlabovskou funkcí `fft`. Vypočítá se její modul. Na obrázku 4.4 je zobrazena takto získaná modulová frekvenční charakteristika jednoho segmentu, kde daný tón byl zahrán na čistou (nezkreslenou) kytaru.



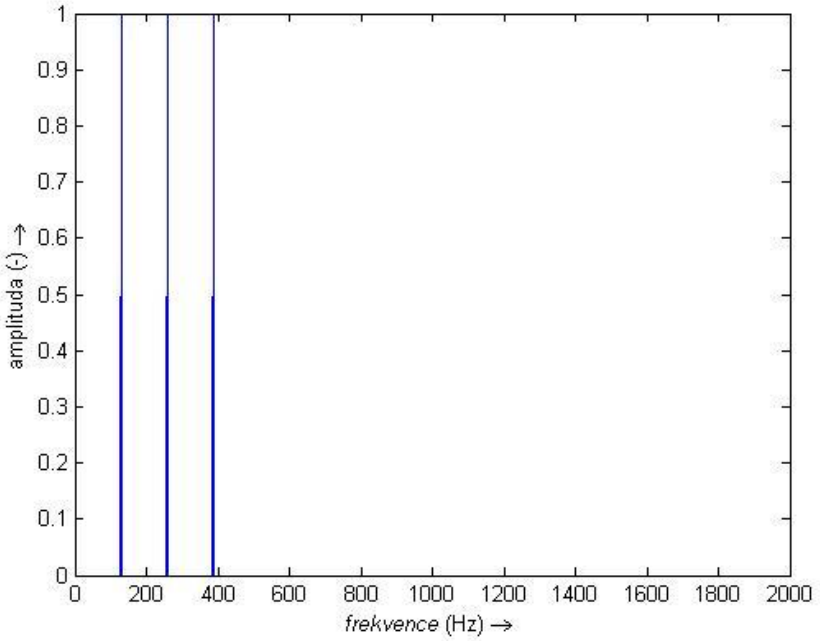
Obrázek 4.4: Frekvenční spektrum tónu čisté kytary.

Na obrázku 4.5 je zobrazena tatáž charakteristika i stejný tón, pouze byla zdrojem zkreslená kytara.



Obrázek 4.5: Frekvenční spektrum tónu zkreslené kytary.

Jak lze z obou obrázků vidět, základní tón má nižší amplitudu než nejvyšší harmonická. Proto, pokud by se detekovala jednoduše maximální hodnota, nemuselo by se jednat o základní tón. Je tedy třeba najít základní frekvenci. Nejlogičtější by bylo najít jednoduše maximální hodnotu a poté testovat, zda se na poloviční frekvenci nenachází výrazná špička a takto testovat dokud by tato podmínka byla splněna, nebo dokud by frekvence testovaného tónu byla vyšší jak frekvence nejnižšího možného tónu. Tento algoritmus byl testován, ale nebyl příliš úspěšný a také byl více výpočetně náročnější než algoritmus, který byl nakonec použit. Bylo použito prahování, kde se hodnotám spektra menším nebo rovné prahu přiřadí nulová hodnota a hodnotám nad tímto prahem se přiřadí hodnota 1. Problém ovšem je určit hodnotu prahu. Problém ovšem je určit hodnotu prahu, protože jak lze vidět, nelze zvolit univerzální hodnotu, byť by byla procentuálně určená z maximální hodnoty. Jak vypadá spektrum zobrazené na obr. 4.4 po prahování, kde práh byl zvolen 50%, je na obrázku 4.6.



Obrázek 4.6: Prahované frekvenční spektrum tónu čisté kytary.

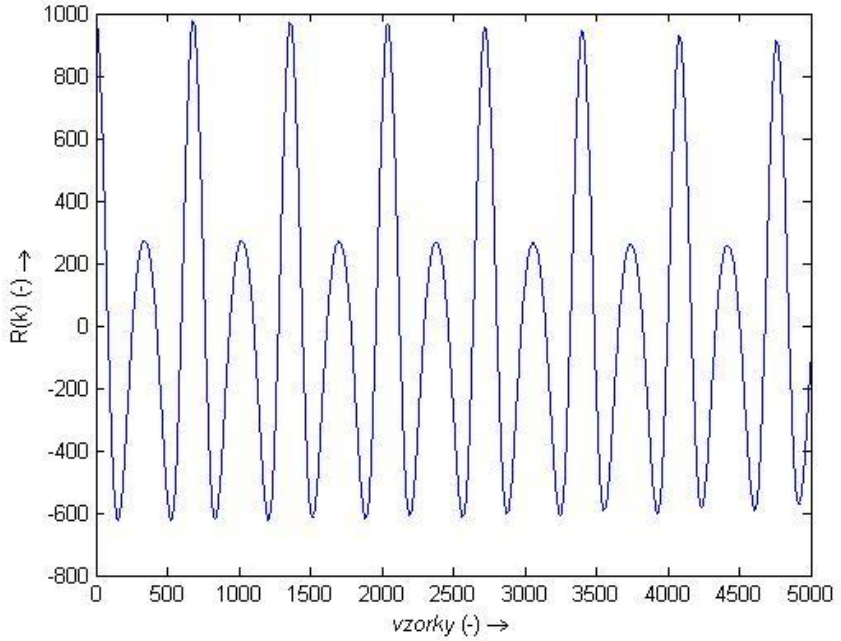
Nyní už stačí zjistit index prvního maxima funkcí `max`. Protože mají všechna maxima stejnou hodnotu, tato funkce vrátí hodnotu indexu maxima, který je nejblíže 0. Pak se odpovídajíc frekvence vypočítá dle vztahu:

$$f_t = \frac{fs}{segment_{length}}(i_{\max} - 1), \tag{11}$$

kde f_t je frekvence tónu v Hz, i_{max} je index prvního maxima. Konstanta -1 se v rovnici vyskytuje z důvodu, že MATLAB indexuje proměnné od hodnoty 1.

4.5 Autokorelační metoda

Nejprve je testováno, zda daný segment není pomlka. To se provádí zjištěním maximální hodnoty v segmentu funkcí `max`, pokud je tato hodnota 0, další výpočet se neprovádí a do výstupního vektoru se na daný index segmentu zapíše 0. Funkcí `xcorr` se provede cyklická autokorelace. Protože je cyklická autokorelace funkce sudá, stačí použít pouze pravou polovinu této funkce, což se prakticky provede tak, že se do proměnné, se kterou se poté dále pracuje, uloží pouze vzorky od indexu odpovídající polovině celkového počtu vzorků. Protože autokorelace určuje míru podobnosti, je zřejmé, že signál sám sobě bude nejvíce podobný na vzorcích odpovídající základní periodě daného signálu. Hledá se tedy maximum této funkce. Jak vypadá jednostranná autokorelační funkce jednoho segmentu tónu čisté kytary, je na obrázku 4.7.



Obrázek 4.7: Jednostranná autokorelační funkce tónu čisté kytary.

Začátek této funkce připomínající tvarem funkci cosinus vyjadřuje energii daného signálu, další vrchol poté základní periodu. Frekvenci základního tónu lze zjistit ze vztahu:

$$f_z = \frac{f_s}{L}, \tag{12}$$

kde L je index prvního vrcholu. Energie signálu je zpravidla vyšší než maximální hodnota dalších vrcholů, proto je nutné maximum hledat až od určitého indexu, vyjdeme-li ze vztahu 12, tak můžeme zjistit index maximální detekované frekvence, která se v tónu může objevit a od toho indexu hledat maximum, tedy od indexu:

$$L_{\min} = \frac{f_s}{f_m}, \tag{13}$$

kde f_m je nejvyšší detekovatelná frekvence daného tónu. Vztah pro výpočet frekvence základního tónu tedy je:

$$f_t = \frac{f_s}{L_{\min} + L_i - 2}, \tag{14}$$

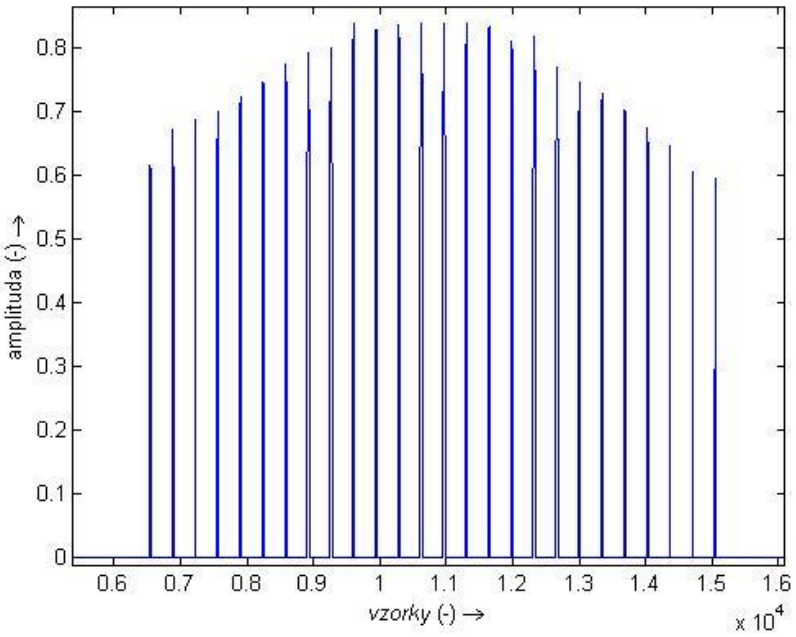
kde L_i je index prvního maxima získaný z MATLABu. Konstanta -2 se zde z důvodu, že MATLAB indexuje od 1 a tedy, je třeba provést korekci jak členu L_{min}, tak členu L_i.

Dále bylo implementováno předzpracování vstupního signálu algoritmem nazývaným **centrální klipování**, jež je definován:

$$c(i)=x(i) \text{ pro } |x(i)|=>p, \tag{15a}$$

$$c(i)=0 \text{ pro } |x(i)|<p, \tag{15b}$$

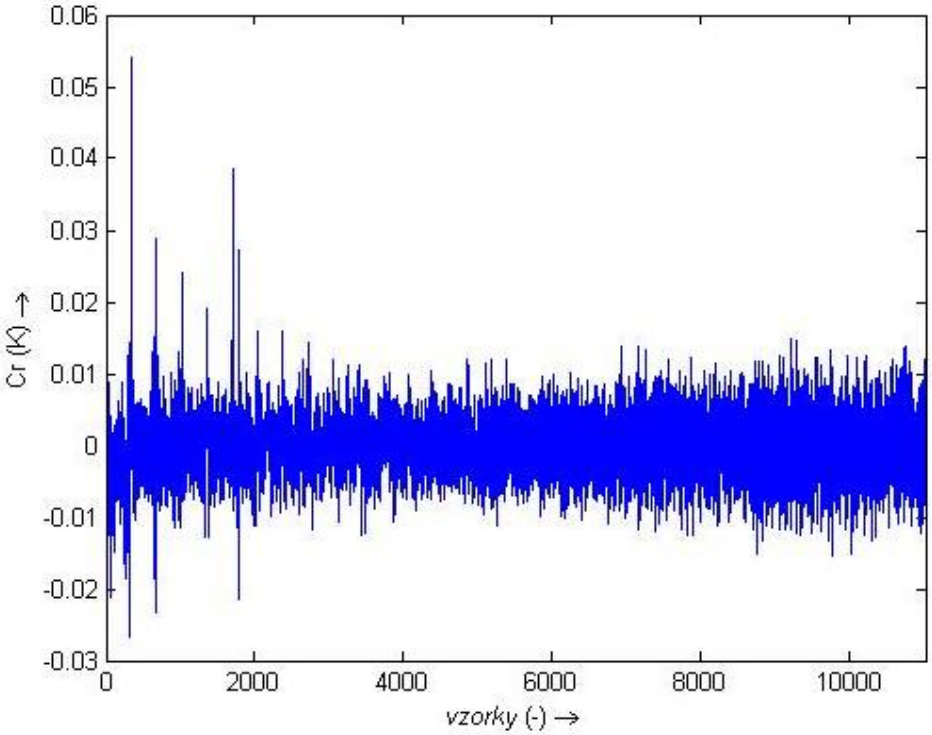
kde *c(i)* je klipovaná hodnota o indexu odpovídající indexu vstupního vektoru, *x(i)* je hodnota vektoru na daném indexu, *p* je práh, jež se nejčastěji volí okolo 70% z maxima zpracovávaného segmentu. Tento algoritmus by měl zlepšovat přesnost detekce autokorelační metodou. Jak vypadá vstupní segment odpovídající tónu čisté kytary klipovaný touto metodou je na obrázku 4.8: [12]



Obrázek 4.8: Klipovaný vstupní signál.

4.6 Kepstrální metoda

U této metody je, stejně jako u autokorelační metody, nutné nejprve detekovat pomlku, což je realizováno identickým postupem jako u autokorelační metody. Poté je vypočítáno kepstrum, tak jak bylo uvedeno v kapitole 1. Z vypočteného vektoru se dále zpracovává pouze reálná část, ta se v MATLABu vybere funkcí `real`. Frekvence základního tónu se z kepstra určí obdobně jako u autokorelační metody, je také třeba začít hledat maximum až od určité hodnoty, určené dle vztahu 13, u této metody je však nutno i určit maximální index, kde hledat maximum, to se určí také ze vztahu 13, ale namísto maximální frekvence se dosadí minimální frekvence. Základní tón se pak určí podle vztahu 14. Jak vypadá reálné kepstrum tónu čisté kytary je na obrázku. 4.9.



Obrázek 4.9: Reálné kepstrum tónu čisté kytary.

4.7 Přiřazení tónů a prezentace výsledků

V této poslední části se detekovaným frekvencím tónů přiřadí čísla, která jsou odvozena z tabulky 4.1:

oktáva	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
název tónů	**číslo tónu**									
c	0	12	24	36	48	60	72	84	96	108
cis	1	13	25	37	49	61	73	85	97	109
d	2	14	26	38	50	62	74	86	98	110
dis	3	15	27	39	51	63	75	87	99	111
e	4	16	28	40	52	64	76	88	100	112
f	5	17	29	41	53	65	77	89	101	113
fis	6	18	30	42	54	66	78	90	102	114
g	7	19	31	43	55	67	79	91	103	115
gis	8	20	32	44	56	68	80	92	104	116
a	9	21	33	45	57	69	81	93	105	117
ais	10	22	34	46	58	70	82	94	106	118
h	11	23	35	47	59	71	83	95	107	119

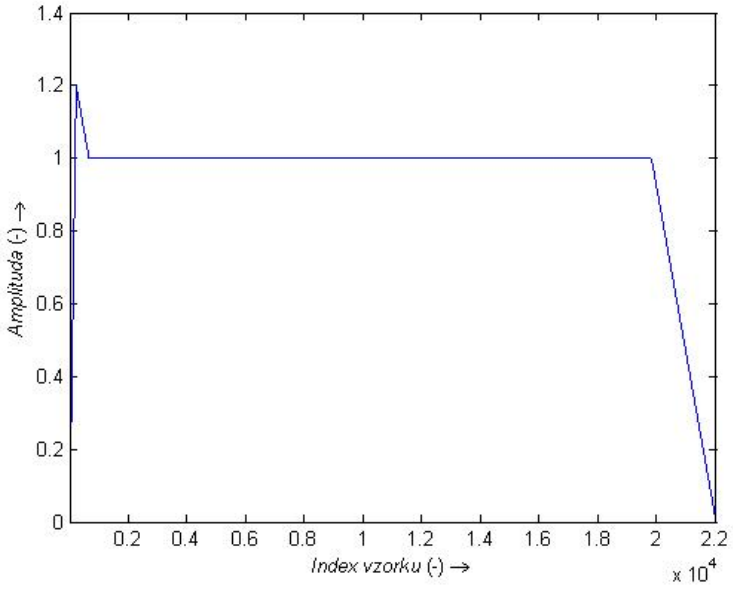
Tabulka 4.1: Přiřazení čísel tónům

Tato čísla nebyla vybrána náhodně, nýbrž jsou to čísla používaná ke kódování výšky tónů v protokolu MIDI. Tato volba tak usnadňuje možnou modifikaci aplikace, kdy by se detekovaná melodie zapisovala do MIDI souboru.[15]

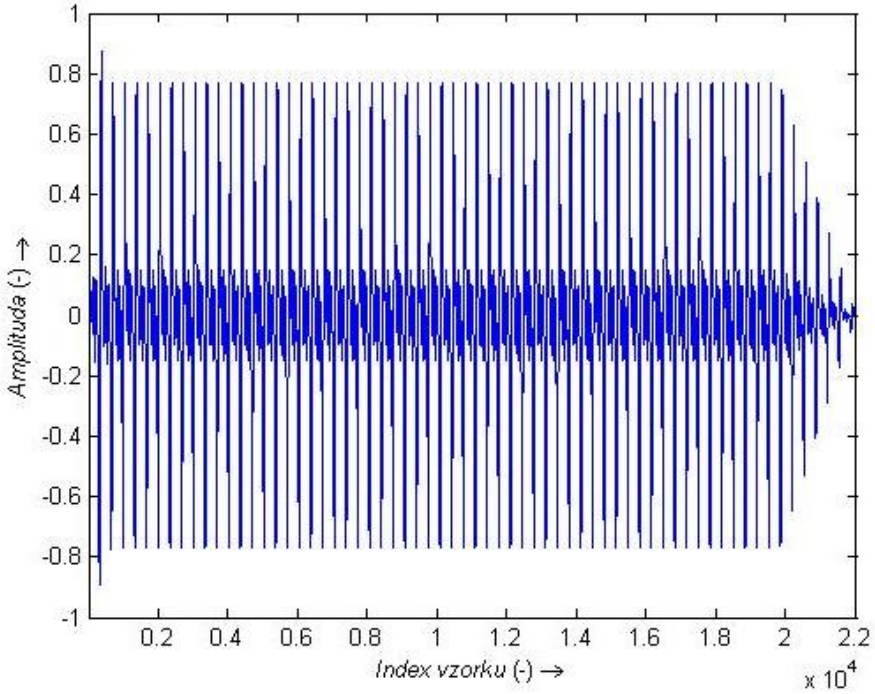
Nejprve je vygenerován vektor frekvencí odpovídají podle vztahu 5. Rozsah tónů byl ale zvolen od 1. do 9. oktávy, tedy první hodnota tohoto vektoru je 32,7Hz. Důvodem proč není zahrnuta i 0. oktáva je, že tyto tóny jsou na spodní hranici slyšitelnosti lidského ucha, stejně tak je ucho na tyto frekvence velmi málo citlivé, melodie se tedy v této oktávě zpravidla nevyskytuje. Pro větší variabilitu nebyl vektor frekvencí tónů napsán staticky do tabulky, ale je pokaždé vygenerován, toho se může využít, pokud daná skladba nebyla zahrána ve standardizovaném ladění, ale v mírně odlišném, pak stačí přepsat pouze 1. hodnotu vektoru frekvencí a zbytek se automaticky dopočítá. V cyklu se prochází vstupní vektor frekvencí a porovnává s vektorem vygenerovaných tónů. Protože reálný nástroj může být mírně rozladěn a popsané algoritmy detekce dominantní frekvence mají přesnost danou délkou segmentu, je použit interval kam může spadat frekvence vstupního vektoru, tolerance byla experimentálně zvolena na 3% na obě strany od přesné hodnoty daného tónu, při použití této tolerance je nepatrný překryv mezi tóny, to zaručuje, že je vždy vstupnímu vektoru přiřazen tón, pokud je v intervalu mezi nejnižším a nejvyšším tónem. Pokud vstupní frekvence je v toleranci daného tónu, odpovídající číslo tónu se přiřadí na danou pozici do výstupního vektoru. Jestliže je vstupní frekvence 0, tedy jedná se o pomlku, je na tuto pozici do výstupního vektoru zapsána hodnota „0“. V případě, kdy vstupní frekvence je pod tolerancí nejnižšího tónu, je zapsána hodnota „900“, pokud je frekvence nad tolerancí nejvyššího tónu, je zapsána hodnota „1000“.

Hodnoty určených tónů všemi metodami jsou poté zobrazeny v hlavním okně MATLABu, stejně tak grafické vyjádření ve formě sloupcových grafů, pro lepší možnost porovnání, pod sebou v jednom okně Figure.

Pro sluchovou kontrolu byl vytvořen generátor tónů, který vytvoří ze získaných vektorů tónů melodii. Generátor tónů vytváří sinusový průběh odpovídající frekvenci daného tónu, pro přirozenější zvuk jsou přičteny vyšší harmonické s amplitudami zvolenými tak, aby zvuk byl poslouchatelný. Doba tónů se vypočte z detekovaného tempa. Protože mezi sousedními tóny bylo slyšet lupnutí, byla upravena obálka tónu, tóny jsou tedy amplitudově modulovány, modulační signál je tvarován tak, aby lupnutí nebylo patrné. Pro tóny pod tolerancí nejnižšího tónu byla zvolena frekvence 20Hz a pro tóny nad tolerancí nejvyššího tónu 20kHz. Na obrázku 4.10 je obálka pro tón o tempu 120BPM, vzorkování 44,1kHz. Na obrázku 4.11 je tvar tónu „c“ ze třetí oktávy o stejných parametrech, jako obálka na obr. 4.10.



Obrázek 4.10: Obálka tónu.



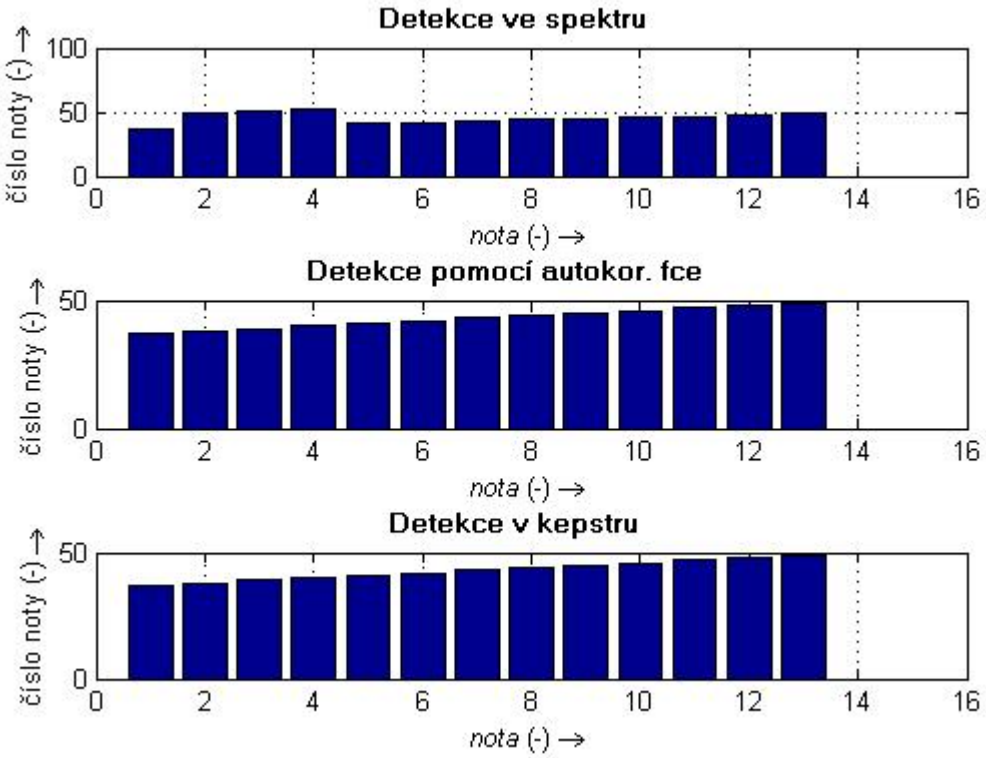
Obrázek 4.11: Vygenerovaný tón "C" ze třetí oktávy.

4.8 Porovnání metod

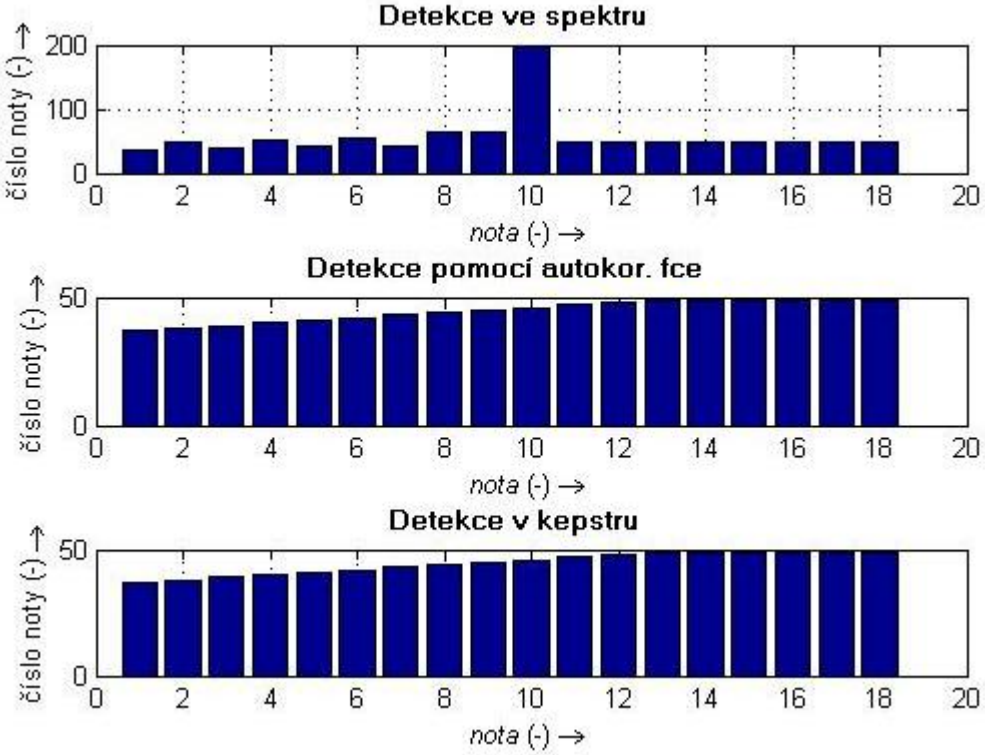
Pro porovnání jednotlivých metod detekce základního tónu byly použity celkem 3 nahrávky-půltóny zahrané vzestupně od tónu „c“ z 3. oktávy do tónu „c“ ze 4. oktávy, nahrané čistou kytarou a při použití efektu overdrive, což je v podstatě oboustranný omezovač signálu, tedy potom má tón bohatší spektrum a poslední je jednoduchá

melodie s pomlkou zahraná čistou kytarou.

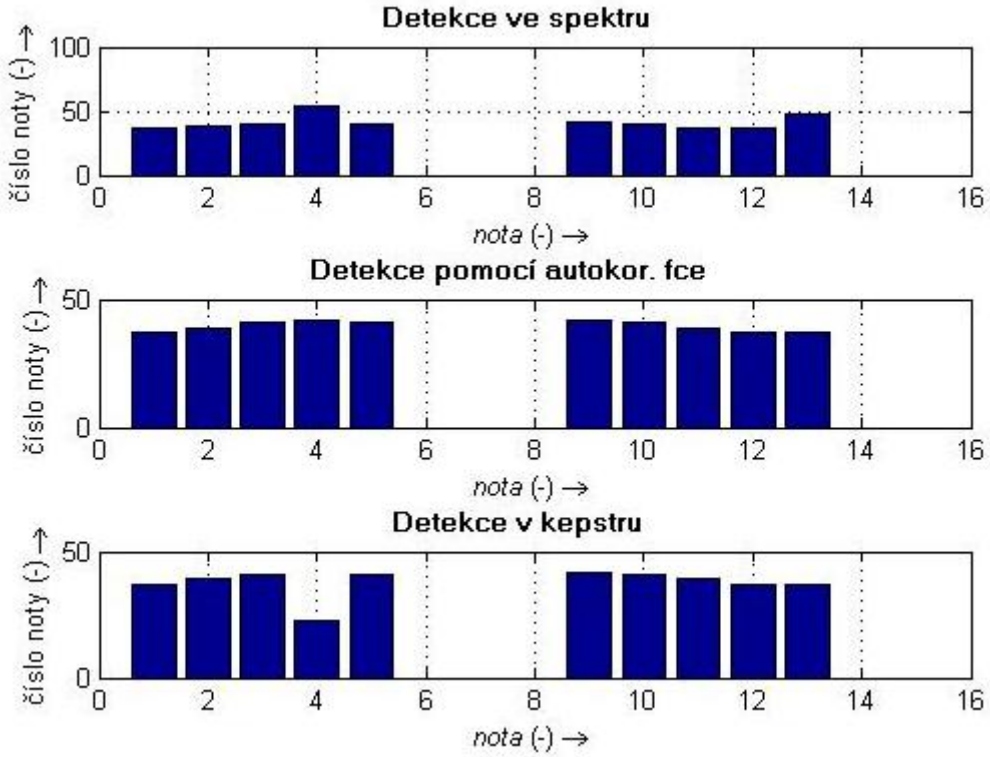
Na obrázcích 4.12 až 4.14 jsou zobrazeny grafické výstupy detekce tónu z programového prostředí MATLAB, pořadí jednotlivých obrázků koresponduje s výše uvedenými testovacími signály. Práh pro spektrální metodu byl zvolen na 35% z maxima a byl pro všechny signály stejný. U obou chromatických stupnic byl u autokorelační metody použit algoritmus centrálního klipování, bylo zjištěno, že i při jeho absenci jsou stejné výsledky. U posledního signálu při použití tohoto algoritmu byl 1 tón chybně určen, při jeho absenci byly všechny tóny správně určeny, proto se tento algoritmus u posledního testovacího signálu nepoužil. Pomlky na konci každého vektoru jsou způsobeny nepatrným doznívání tónu. U chromatické stupnice je nahrávka zkreslené kytary delší a i vlivem zkreslení má doznívající tón vyšší intenzitu, proto je detekován poslední tón delší dobu.



Obrázek 4.12: Detekované tóny pro chrom. tóny čisté kytary.



Obrázek 4.13: Detekované tóny pro chrom. tóny zkreslené kytary.



Obrázek 4.14: Detekce tónů v melodii včetně pomlky.

V tabulce 4.2 jsou uvedeny detekované tóny pro oba testovací signály chromatické stupnice, dále referenční správné tóny a úspěšnost jednotlivých metod. Úspěšnost je počítána na základě počtu správně detekovaných tónů k celkovému počtu tónu, přičemž pomlky se nezapočítávají, protože jejich detekce není závislá na těchto algoritmech. V tabulce 4.3 jsou uvedeny detekované tóny jednoduché melodie zahrané na elektrofonickou kytaru, ve které jsou obsaženy pomlky, rovněž jsou uvedeny referenční správné tóny a úspěšnost jednotlivých metod.

čistá kytara				zkreslená kytara			
č. ref. tónů	**DFT**	**autokor.**	**kepstr.**	**č. ref. tónů**	**DFT**	**autokor.**	**kepstr.**
37	37	37	37	37	37	37	37
38	50	38	38	38	50	38	38
39	51	39	39	39	39	39	39
40	52	40	40	40	52	40	40
41	41	41	41	41	41	41	41
42	41	42	42	42	54	42	42
43	43	43	43	43	43	43	43
44	44	44	44	44	63	44	44
45	45	45	45	45	64	45	45
46	46	46	46	46	199	46	46
47	46	47	47	47	47	47	47
48	47	48	48	48	48	48	48
49	49	49	49	49	49	49	49
0	0	0	0	49	49	49	49
0	0	0	0	49	49	49	49
0	0	0	0	49	49	49	49
				49	49	49	49
				49	49	49	49
				0	0	0	0
				0	0	0	0
úspěšnost	53,90%	100%	100%	-	66,70%	100%	100%

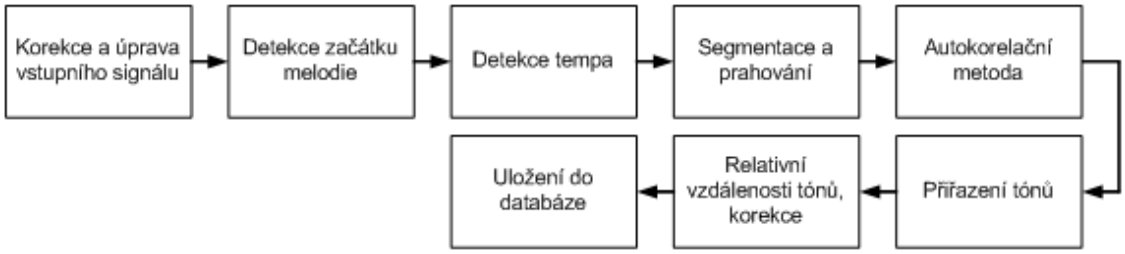
Tabulka 4.2: Detekované tóny chromatické stupnice a úspěšnost metod.

č. ref. tónů	**DFT**	**autokor.**	**kepstr.**
37	37	37	37
39	38	39	39
41	40	41	41
42	54	42	23
41	40	41	41
0	0	0	0
0	0	0	0
0	0	0	0
42	41	42	42
41	40	41	41
39	37	39	39
37	37	37	37
37	48	37	37
0	0	0	0
0	0	0	0
0	0	0	0
úspěšnost	20%	100%	90%

Tabulka 4.3: Detekované tóny melodie s pomlkou a úspěšnost metod.

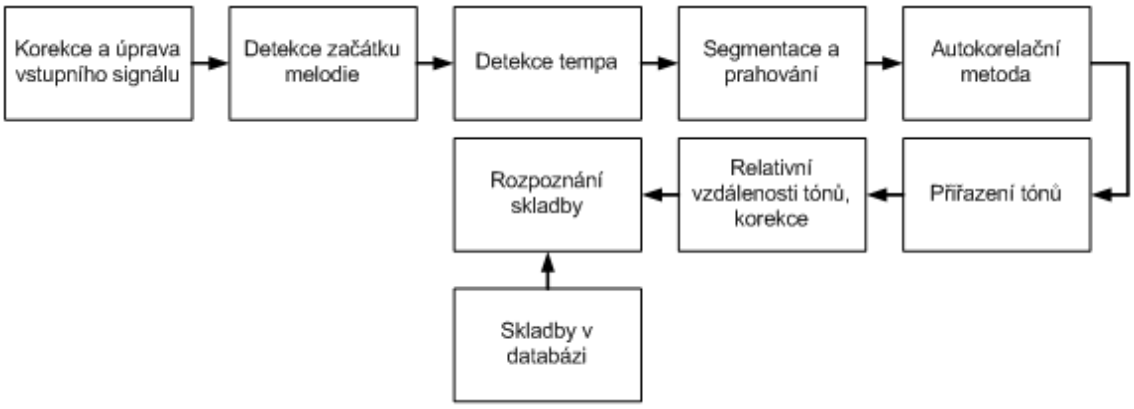
5 DATABÁZE MELODIÍ A ROZPOZNÁNÍ MELODIE

Tato kapitola popisuje způsob, jak se melodie zaznamenané v wav souboru zapisují do databáze a jak se rozpoznává úsek melodie (v této práci dále označovaná jako hledaná skladba, což je část melodie, která je obsažena v melodii v databázi) ze vstupního wav souboru v které melodii uložené v databázi je obsažen. Samotná implementace algoritmů do uživatelského rozhraní je uvedena v následující kapitole. Na obrázku 5.1 je blokové schéma algoritmu pro přidání skladby do databáze.



Obrázek 5.1: Blokové schéma algoritmu pro přidání skladby do databáze.

Je logické, že většina bloků je použita z algoritmu pro určení melodie, uvedené v předchozí kapitole, tyto bloky jsou kromě detekce tempa a přiřazení tónů nezměněny. V bloku Detekce tempa byla pouze rozšířena detekce nejvyššího možného tempa na 400BPM a krok detekce tempa byl zvýšen o 1, tj. že se detekují pouze sudé hodnoty tempa, což sníží výpočetní náročnost, ale zároveň má minimální vliv na detekci jednotlivých tónů. Blok přiřazení tónů byl změněn na základě požadavků bloků relativní vzdálenosti tónů, korekce, proto pro lepší pochopení souvislosti mezi nimi bude vysvětlen až v podkapitole, ve které je tento blok rozebrán. Na obrázku 5.2 je pak uvedeno blokové schéma algoritmu pro rozpoznání skladby.



Obrázek 5.2: Blokové schéma algoritmu pro rozpoznání melodie.

Lze si všimnout, že schéma je identické s předchozím schématem, vyjma posledního bloku, navíc je zde jeden blok navíc. Toho je využito při tvorbě uživatelského rozhraní, vice je uvedeno v další kapitole. Nové bloky uvedené v obou algoritmech jsou podrobněji rozebrány v následujících podkapitolách.

5.1 Detekce začátku melodie

Uživatel při záznamu melodie nemusí nutně začít hrát ihned při zapnutí nahrávání, takže melodie ve vstupním wav souboru nemusí začínat v čase nula daného souboru, jinými slovy na začátku může být blok nul, respektive vzorky odpovídající šumu a ruchům snímacího zařízení. Pokud by tento blok nebyl použit, mohlo by se zdát, že segmenty by neodpovídaly tónům melodie, ale byly by o určitou konstantní hodnotu posunuty, potom by v každém segmentu byly dva tóny a rozpoznání tónů by bylo chybné.

Princip algoritmu je obdobný jako detekce pomlk uvedená v předchozí kapitole. Spočítá se průměrná střední hodnota celého signálu podle vztahu

$$E = \frac{\sum_{i=1}^{N} s(i)^2}{N}, \tag{16}$$

kde *s* je vzorek vstupního signálu na indexu *i*, *N* je celkový počet hodnot vzorků ve vstupním signálu. Poté se matlabovskou funkcí `find` zjistí první nejnižší index vzorku, který má absolutní hodnotu větší než zjištěná prům. střední hodnota vynásobená experimentálně zvolenou konstantou. Vstupní signál je pak uložen do proměnné pro další zpracování od tohoto indexu.

5.2 Relativní vzdálenost tónů a korekce

Jen malé procento lidí má absolutní sluch, což znamená schopnost si pamatovat melodii v její absolutní výšce [17]. To znamená, že je vysoká pravděpodobnost, že by uživatel hledanou část melodie nahrál v jiné tónině (v jiné absolutní výšce), vyhledávací algoritmus by potom původní skladbu nebyl schopen najít. Proto je toto ošetřeno tím, že se ukládají pouze intervaly mezi tóny a ne jejich absolutní výška. Prakticky se od sebe odečtou vždy dva sousední tóny, což lze zapsat jako:

$$tonyr(i) = tony(i+1) - tony(i), \tag{17}$$

kde *tonyr* je vektor relativních tónů, *tony* je vektor tónů v absolutní výšce, *i* je index daného vektoru a *I* je počet hodnot ve vektoru *tony*. Je tedy zřejmé, že vyhrazené číselné označení "0" pro pomlku je třeba změnit, protože nyní "0" znamená, že dva tóny vedle sebe mají stejnou hodnotu (výšku). Proto ve funkci pro přiřazení tónů byla změněna pro pomlku hodnota na "700". Při nahrávání hledané melodie uživatelem je většinou po dohrání melodie prodleva než ukončí nahrávání stiskem tlačítka v daném záznamovém zařízení (nebo aplikaci), což se projeví jako pomlky na konci melodie. Rozpoznání odpovídající skladby by potom nemuselo být úspěšné, protože v odpovídající skladbě zpravidla za touto hledanou melodií není počet pomlk odpovídající prodlevě ukončení nahrávání. Tyto pomlky na konci se odstraňují algoritmem, kdy se v cyklu od posledního indexu melodie testuje, zda je hodnota "700".

Pokud by hledaná melodie byla v jiné tónině jako odpovídající skladba v databázi a

melodie by obsahovala pomlky, tón za pomlkou by měl u obou melodií rozdílnou hodnotu. Toto je ošetřeno tak, že od tónu za pomlkou se odečte tón před pomlkou. Algoritmus na tuto operaci je poněkud složitější. Nejprve je třeba tónům s hodnotou větší jak "500", což odpovídá začátkům pomlk, znovu přiřadit hodnotu "700", tónům s hodnotou menší jak "-500", což analogicky odpovídá koncům pomlk, je třeba přiřadit "-700". Dále je použita matlabovská funkce `strfind`, která vrací index (případně vektor indexů) vektoru, na kterém je zadaná hodnota. Touto funkcí se hledají konce pomlk, tedy hodnota "700" a také začátky pomlk, tedy hodnota „-700“ ve vektoru relativních tónů. Do vektoru relativních melodií se pak na indexu konce pomlky odečte hodnota odpovídající tomuto indexu ve vektoru absolutních melodií od hodnoty odpovídající indexu začátku pomlky, též v absolutních tónech.

5.3 Uložení do databáze a skladby v databázi

Jednotlivé melodie určené z wav souborů, které jsou zaznamenané formou intervalů, je nutno uložit společně s názvem daného wav souboru. Pokud by se tato data ukládala pouze do proměnné, při ukončení aplikace by se data ztratila, z toho důvodu je nutné je uložit do souboru na disk počítače. Jako optimální formát se ukázal tzv. *.mat file, což je speciální typ souboru používaný prostředím MATLAB, kam je možno ukládat jednotlivé proměnné. Do souboru se ukládá příkazem `save` za nímž následuje název souboru, pokud soubor s daným názvem existuje, automaticky se přepíše.

V práci je použit soubor databaze.mat ve kterém jsou dvě proměnné, první z nich nese název *tony* a každý řádek obsahuje melodii z jednotlivých wav souborů. Názvy souborů se ukládají do řádků, které odpovídají řádkům melodiím. Názvy se ukládají do proměnné *nazev*, jež je druhá proměnná v souboru databaze.mat. Při uložení nové melodie do databáze jsou nejprve načteny obě proměnné příkazem `load databaze`, uložení nového názvu se provede příkazem `char`, konkrétně `nazev=char(nazev,nazevn);` kde *nazevn* je proměnná, kde je uložen název nového souboru. Uložení melodie je mírně komplikovanější a to z důvodu, že obecně může být melodie různé dlouhá a může tedy mít různý počet hodnot, protože se ukládá do matice, musí být všechny vektory melodií stejně dlouhé. To je ošetřeno tak, že se porovnává délka nové melodie a délka melodií v databázi, pokud je nová melodie delší, k melodiím v databázi se doplní patřičný počet nul, pokud je nová melodie kratší, doplní se k této melodii nuly. Počet řádků matice je tedy roven počtu hodnot nejdelší melodie. Aby to takto mohlo fungovat, byl v MATLABU vytvořen skript "resetmatf", který do proměnné *nazev* uloží "prazdne" a do proměnné *tony* uloží "0", tento skript je třeba spustit před přidáním prvního záznamu do databáze, tím je zaručeno, že všechny melodie v databázi budou mít stejný počet hodnot.

5.4 Rozpoznání skladby

K vyhledání skladby byla nejprve využita funkce vzájemné korelace mezi hledanou melodií a jednotlivými melodiemi zaznamenanými v databázi, jako odpovídající skladba byla vybrána ta, která měla nejvyšší hodnotu vzájemné korelace.

Toto řešení se však ukázalo nespolehlivé. Jako nejvhodnější řešení se ukázalo použití funkce `strfind`, která mimo jiné umožňuje místo jedné hodnoty použití sekvence určených hledaných hodnot.

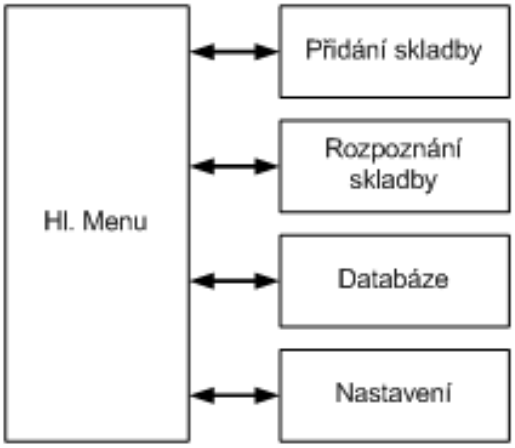
Nejprve je třeba načíst databázi a detekovat tóny v hledané melodii. Poté se provede příkaz: `ix=strfind(reshape(transpose( tony),1,[]),tonyn);` kde *tonyn* jsou tóny hledané melodie, *tony* jsou tóny melodií z databáze a *ix* je index, na kterém začíná hledaná sekvence. Protože tato funkce vyžaduje vstupní proměnnou vektor, je matice tónů, která se načte z databáze, přeskládána na jeden dlouhý vektor pomocí funkce `reshape`. Pokud je daná sekvence (což jsou tóny hledané melodie) obsažena, je třeba index(y) přepočítat zpět na číslo řádku, což se provede prostým vydělením indexu počtem sloupců se zaokrouhlením pomocí funkce `ceil`. Následně je funkcí `unique` odstraněn případný opakující se řádek (v případě, že se daný úryvek melodie ve skladbě opakuje). Hledaná melodie pak nese název uložený v proměnné *nazev* na daném řádku.

6 UŽIVATELSKÉ ROZHRANÍ

V této kapitole je popsáno uživatelské rozhraní aplikace pro rozpoznávání melodií, nejprve z pohledu jak bylo vytvořeno a následně z uživatelského hlediska, tj. jak se program obsluhuje. Nakonec jsou uvedeny výsledky při ověřování funkce programu na testovacích melodiích.

6.1 Implementace algoritmů

Uživatelské rozhraní bylo vytvořeno v MATLAB Graphical User Interfaces, zkráceně GUI. Toto rozhraní umožňuje pohodlné ovládání funkcí a skriptů pomocí interaktivních prvků v grafickém rozhraní. Na obrázku 6.1 je zobrazeno blokové schéma celé aplikace.

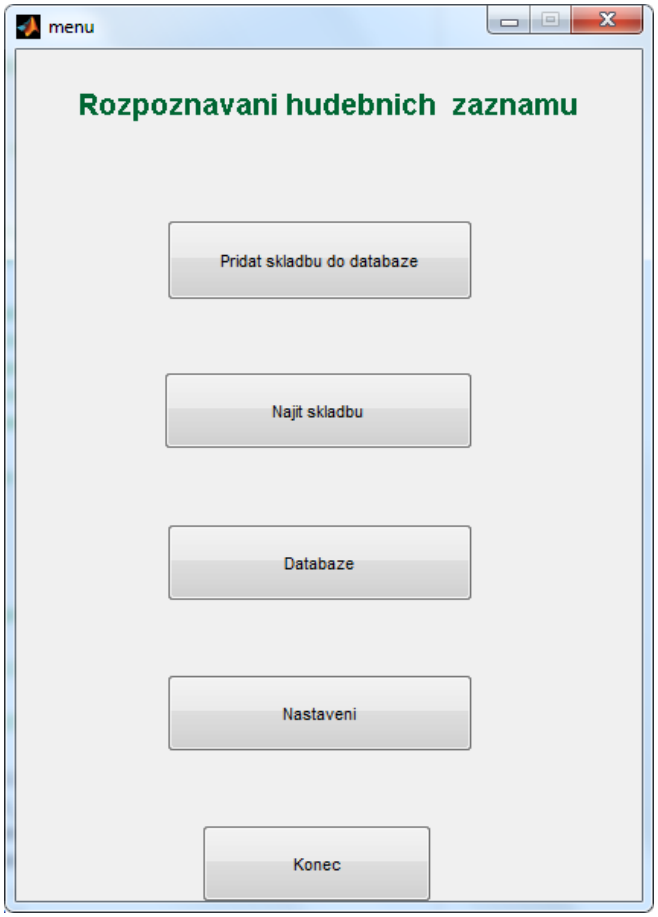


Obrázek 6.1: Blokové schéma celé aplikace.

Jednotlivé bloky jsou přímo okna, které jsou popsány v následující podkapitole. Každé okno se skládá ze dvou částí-z grafické části a obslužné funkce. Grafická část má příponu *.fig, tedy jedná se o stejný typ souboru, který je v MATLABU určen pro zobrazování grafů, v tomto souboru jsou uložena jednotlivá tlačítka a další prvky v okně, jejich parametry, tj. pozice v okně, popisek apod. V souboru obslužné funkce, který má příponu *.m, jedná se tedy o běžný typ souboru pro funkce a skripty, jsou uloženy obslužné funkce pro jednotlivé interaktivní prvky, je zde také možné měnit parametry interaktivních prvků. Program se spouští souborem menu.m, je také možné do příkazového řádku v MATLABU zadat „menu“, přičemž je nutné mít aktuální adresář nastavený na složku s tímto souborem. Soubory s melodiemi je třeba také umístit do složky s programem.

6.2 Hlavní menu

Při spuštění programu se nejprve testuje, zda existuje soubor databaze.mat a nastaveni.mat, pokud neexistuje soubor databaze.m, spustí se skript "resetmatf", pokud neexistuje soubor nastaveni.m, což je soubor pro nastavení parametrů pro určení melodie, o kterém je napsáno více v podkapitole Nastavení, vytvoří se i tento soubor s výchozími hodnotami. Poté se zobrazí okno uvedené na obrázku 6.1



Obrázek 6.2: Hlavní menu.

Z tohoto okna je možné se dále dostat na požadovanou funkci volbou tlačítek:

Tlačítko „**Pridat skladbu do databaze**“ otevře okno přidání skladby. Tato volba se vybere, pokud chce uživatel do databáze přidat novou skladbu s melodií.

Tlačítko „**Rozpoznání skladby**“ otevře okno rozpoznání skladby, slouží k nalezení hledané melodie.

Tlačítko „**Databaze**“ otevře okno databáze. V tomto okně jsou zobrazeny názvy skladeb, které byly přidány pomocí tlačítka „pridat skladbu do databaze“, je zde také možno celou databázi smazat.

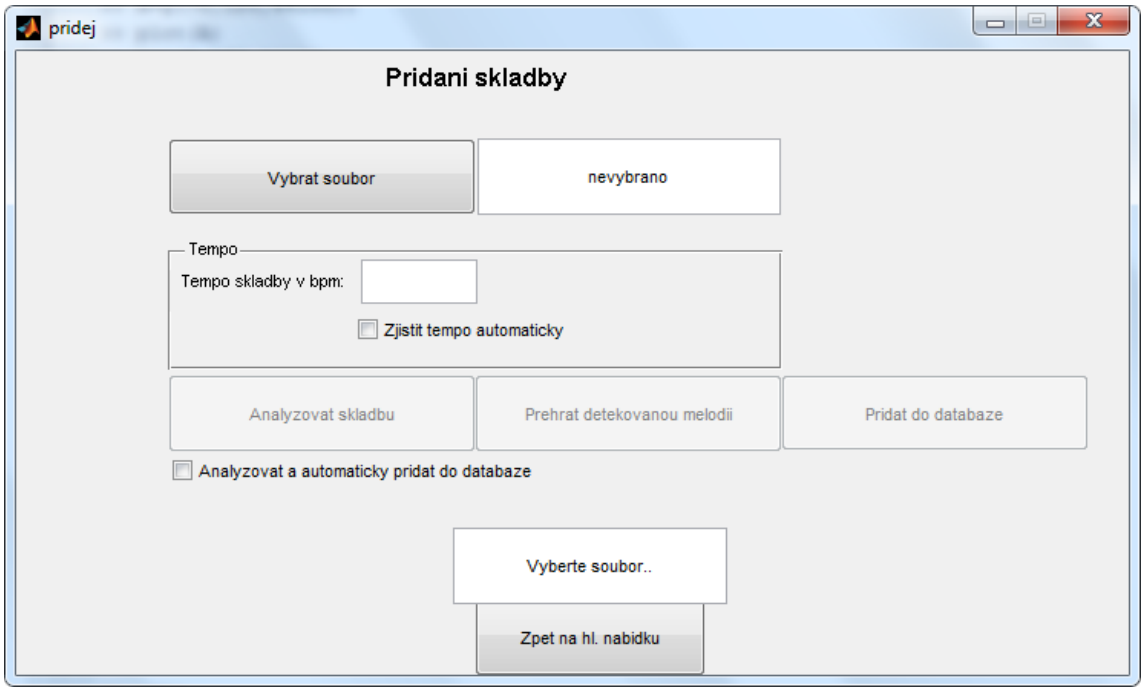
Tlačítko „**Nastaveni**“ otevře okno nastavení. Nastavit je možné parametry pro určení melodie, toto nastavení se použije jak pro přidání skladby, tak pro rozpoznání skladby.

Tlačítko „**Konec**“ ukončí program.

Při stisknutí jakéhokoliv tlačítka se otevře dané nové okno a okno hlavní menu se zavře.

6.3 Přidání skladby

Jak toto okno vypadá, je uvedeno na obrázku 6.3.



Obrázek 6.3: Přidání skladby.

Tlačítkem „Vybrat soubor“ se otevře dialogové okno, kde je třeba nejprve vybrat wav soubor, který chce uživatel přidat do databáze, napravo od tohoto tlačítka je textové pole, kde se vypíše název vybraného souboru.

Textové pole umístěné nad tlačítkem „Zpět na hl. nabidku“ slouží k zobrazování různých informací pro uživatele, dále bude toto textové pole označováno jako informační pole.

Po vybrání wav souboru s danou melodií se tlačítko „Analyzovat skladbu“ stane aktivní. Před jeho stisknutím je však třeba zadat tempo do textového pole „Tempo skladby v bpm“, nebo případně zatrhnout možnost „Zjistit tempo automaticky“.

Nyní je už možné spustit samotné určení melodie, které se provede stiskem tlačítka „Analyzovat skladbu“. Pokud je melodie úspěšně detekována, v informačním poli se vypíše „*.wav byla analyzovana“, přičemž místo symbolu „*“ se zobrazí název daného wav souboru.

Po úspěšné analýze se zpřístupní tlačítka „Prehrat detekovanou melodii“ a „Pridat do databaze“. Tlačítko „Prehrat detekovanou melodii“ přehraje detekovanou melodii pomocí generátoru popsaného v kapitole 4.7, toho je možno využít jako akustickou

kontrolu, zda melodie byla správně detekovaná. Jestliže přehraná melodie neodpovídá původní melodii z wav souboru, je možno zkusit změnit hodnotu tempa a znovu provést analýzu, například pokud generovaná melodie obsahuje pouze každý druhý tón, je možno zkusit hodnotu tempa zdvojnásobit. Dále je možné změnit nastavení detekce melodie, více je uvedeno v podkapitole „Nastavení“.

Poté je možno danou melodii uložit do databáze stiskem tlačítka „Pridat do databaze“, po jeho stisku informuje o úspěšném uložení hláška „*.wav byla pridana“ zobrazená v informačním okně.

Zatržením volby „Analyzovat a automaticky pridat do databaze“ zmizí z okna tlačítko „Pridat do databaze“ a tlačítko „Analyzovat skladbu“ se změní na „Analyzovat a pridat“, stiskem tohoto tlačítka se současně provede jak analýza, tak přidání do databáze, tím lze např. více skladeb uložit do databáze rychleji.

Do okna „Hlavní menu“ se lze vrátit stisknutím tlačítka „Zpet na hl. nabidku“, toto tlačítko je i ve všech dalších oknech, popsaných v dalších kapitolách a má stejnou funkci.

V informačním poli se mohou objevit následující chybové hlášky:

„**Vyberte wav soubor!**“ - Tato hláška se objeví, pokud uživatel v dialogovém okně vybral jiný typ souboru než wav. Všechna tlačítka kromě „Vybrat soubor“ se stanou neaktivní. Je tedy třeba vybrat soubor s příponou wav.

„**Nebyl vybran soubor!**“ - V dialogovém okně nebyl vybrán žádný soubor, tlačítka jsou zablokována stejně jako u předešlé hlášky, řešení je také stejné.

„**Nebylo zadano tempo!**“ - Tato hláška se zobrazí v případě, kdy uživatel stiskne tlačítko pro analýzu skladby, ale nebylo zadáno tempo ani nebyla zatržena možnost automatické detekce tempa. Je třeba zadat tempo nebo zvolit automatickou detekci tempa.

„**Nebylo zadano korektni tempo!**“ - Tento případ nastane, pokud se do pole tempa zadá hodnota „0“ nebo je zadán jiný znak než číslice. Je třeba zadat nenulové tempo, nepoužívat jiné znaky než číslice.

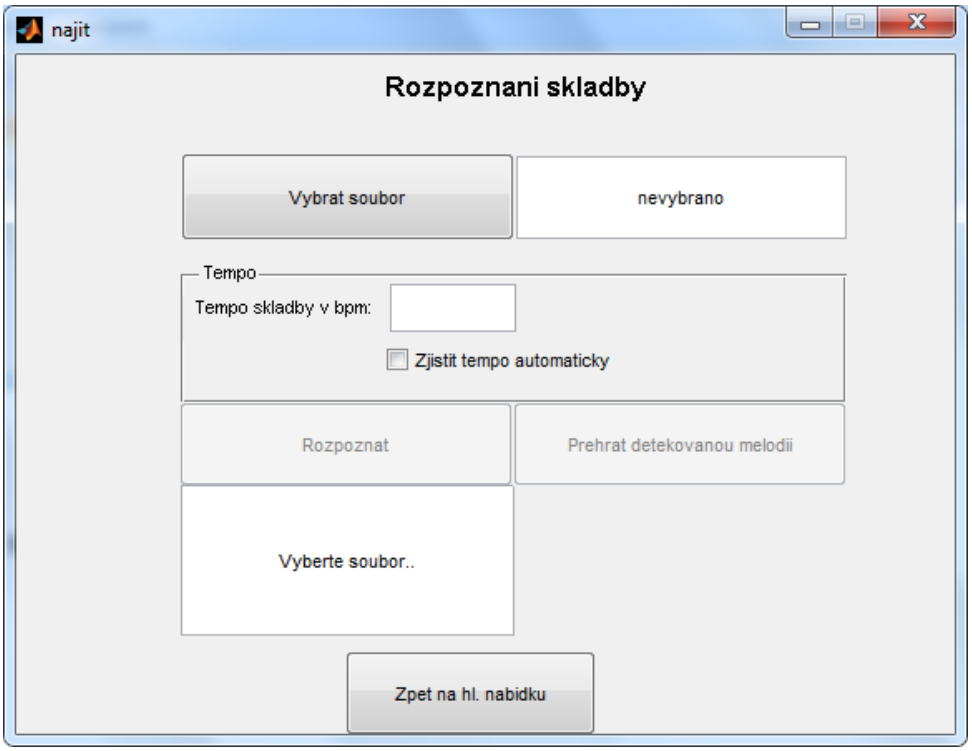
„**Zadejte vyssi hodnotu bpm!**“ - Tato hláška se objeví, jestliže počet segmentů, respektive tónů je méně než 3. Je třeba zvolit vyšší tempo.

„**Nevhodne zvoleny prah!**“ - Pokud uživatel změnil v okně „Nastaveni“ práh pro detekci pomlk o příliš velkou hodnotu, může se zobrazit tato hláška. Je třeba zvolit velikost prahu v intervalu nové a původní hodnoty prahu.

„**Vysoky prah nebo prazdna skladba!**“ - Nastane v případě, kdy je detekován začátek skladby, tj. čas kdy začíná melodie, je blízký konci skladby nebo začátek není vůbec detekovaný. Je to způsobeno příliš vysokým prahem pro pomlky. Řešením je snížit v okně „Nastaveni“ práh pro detekci melodie. Tato hláška se také zobrazí, pokud daný wav soubor je prázdný (obsahuje pouze šum).

6.4 Rozpoznání skladby

Toto okno je na obrázku 6.4.



Obrázek 6.4: Rozpoznání skladby.

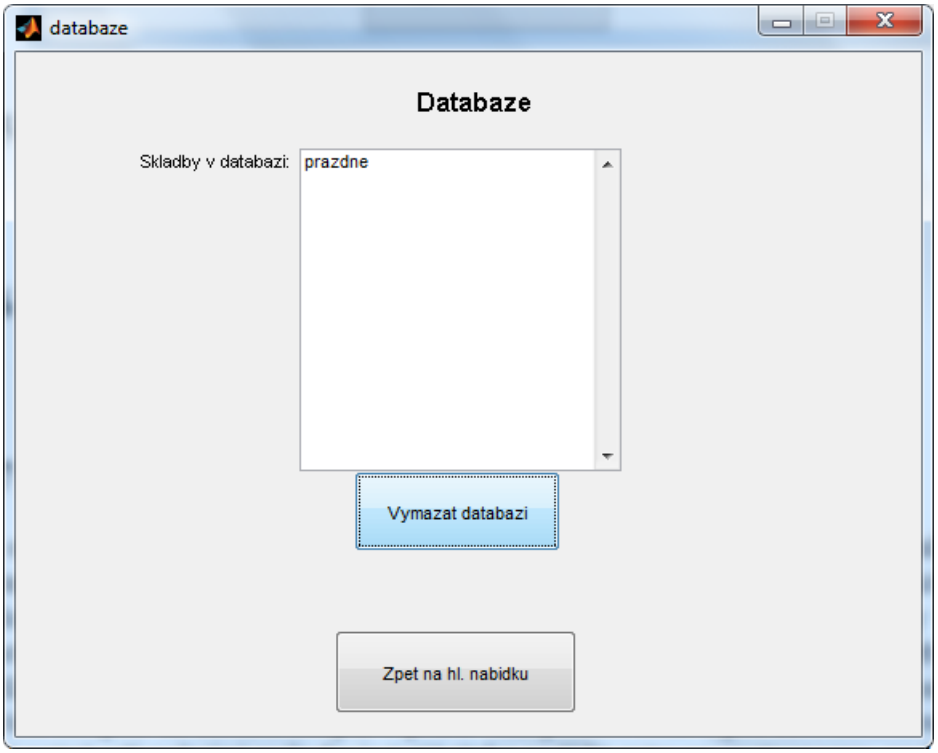
Lze si všimnout, že má některé prvky stejné jako předchozí okno. Informační pole je zde pod tlačítkem „Rozpoznat“. Ovládání je obdobné. Tlačítkem „Vybrat soubor“ se vybere wav soubor, který obsahuje část melodie, ke které chce uživatel vyhledat odpovídající název celé melodie z databáze. Poté se stejně jako v předchozím okně zadá tempo, případně nechá automaticky detekovat. Rozpoznávání se provede stisknutím tlačítka „Rozpoznat“. Pokud program najde odpovídající melodii v databázi, vypíše se v informačním poli „Hledana melodie nalezena ve skladbe: *.wav“, přičemž místo symbolu „*“ je uveden název nalezené skladby. V případě, kdy hledaná melodie nenalezena v žádné skladbě v databázi, je v informačním poli vypsáno „Hledana melodie NENALEZENA“.

V informačním poli se mohou objevit stejné chybové hlášky jako u předchozího okna.

Tlačítko „Prehrat detekovanou melodii“ přehraje vestavěným generátorem hledanou melodii, nikoliv odpovídající melodii.

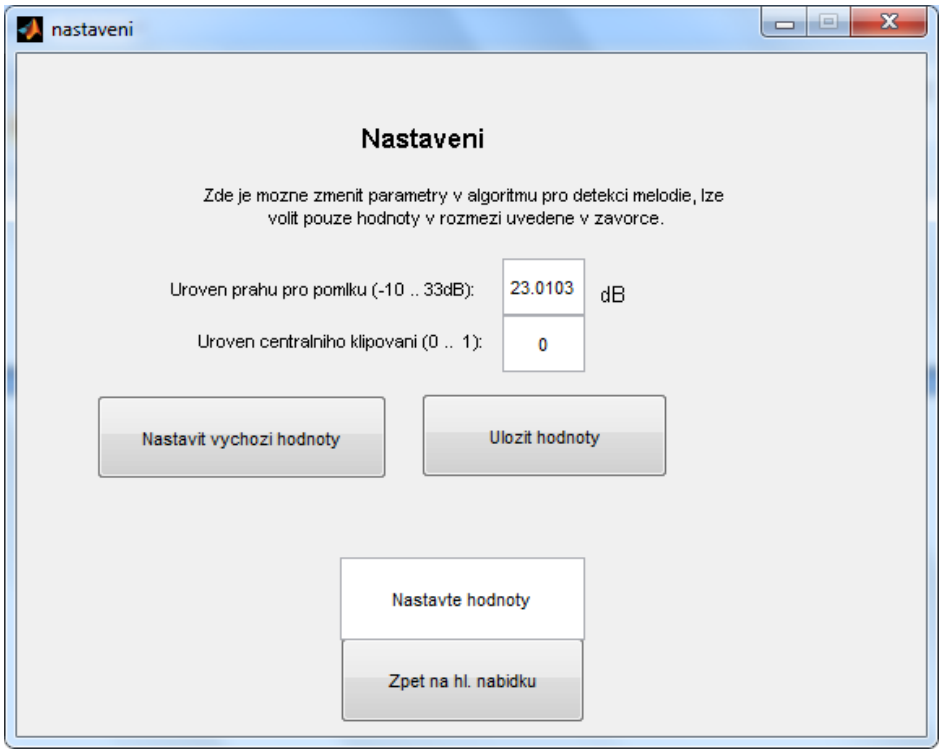
6.5 Databáze

Toto okno je zobrazeno na obrázku 6.5. V textovém poli se zobrazují názvy souborů, které byly přidány do databáze. Kromě tlačítka pro vrácení se do hlavního menu, je zde pouze jediné tlačítko „Vymazat databazi“, po jehož stisknutí se otevře okno s dotazem (okno Databáze zůstane otevřené), zda chce uživatel opravdu databázi vymazat. Při stisknutí tlačítka „Ano“ se spustí skript resetmatf.m, čímž se smaže databáze. Při stisknu tlačítka „Ne“ se okno s dotazem zavře.



Obrázek 6.5: Databáze.

6.6 Nastavení



Obrázek 6.6: Nastavení.

Toto okno je na obrázku 6.6. Parametry v tomto okně je vhodné měnit až v případě, kdy detekovaná melodie neodpovídá melodii v daném wav souboru. Parametr „Uroven prahu pro pomlku“ určuje, které segmenty budou považovány za pomlky a které za tóny a zároveň určuje detekci začátku melodie. Pokud v detekované melodii jsou místo pomlk náhodné tóny, je třeba tento parametr zvýšit, naopak při chybějících tónech v melodii je vhodné jej snížit. Do textového pole se zapisuje kolikrát je práh vyšší než referenční hodnota pro pomlku, vzhledem k možnosti jeho velké změny byla zvolena decibelová míra.

Parametr „Uroven centralniho klipovani“ slouží ke změně úrovně centrálního klipování použitého u autokorelační detekce tónů. Hodnota do textového pole se zadává v relativní míře k nejvyšší amplitudě daného segmentu. Hodnota „0“ znamená, že centrální klipování není vůbec použito. Tento parametr se doporučuje měnit pouze v případě chybné detekce tónů.

Nastavené parametry lze uložit stiskem tlačítka „Ulozit hodnoty“, v textovém poli (opět označovaném jako informační pole) nad tlačítkem „Zpet na hl. nabidku“ se vypíše „Zadane hodnoty ulozeny“, jestliže některá zapsaná hodnota, nebo oboje, není číslo, vypíše se v inf. poli „Zadejte ciselne hodnoty!“, pokud není některá z hodnot, případně oboje, v uvedených rozsazích, vypíše se v informačním poli „Zadejte hodnoty v rozsahu!“. Doporučené hodnoty je možno nastavit stiskem tlačítka „Nastavit vychozi hodnoty“, přičemž jsou i rovnou uloženy, v informačním poli se vypíše „Vychozi hodnoty ulozeny“, tyto výchozí hodnoty jsou zobrazeny na obr. 6.6.

6.7 Testování programu

Na otestování funkčnosti celé aplikace bylo použito 5 testovacích melodií. Byly zvoleny všeobecně známé lidové písně aby byla možná snadná sluchová kontrola. Tyto melodie byly zahrané na elektrofonickou kytaru a k nim byly pořízeny odpovídající krátké úryvky těchto melodií. Krátké úryvky melodií byly záměrně nahrány v jiné tónině a jiném rytmu aby se ověřily všechny vlastnosti aplikace, dále byly některé melodie nahrány s tichem na začátku, aby se ověřil algoritmus pro detekci začátku melodie. Funkce automatické detekce tempa u některých skladeb chybně určila tempo, což způsobilo, že některé melodie nebyly rozpoznány, úspěšnost je tedy závislá na správné detekci tempa. V tom případě bylo tempo zadáno ručně. V praxi však použití funkce pro automatickou detekci není nutná-při samotném nahrávání melodie je totiž třeba se řídit rytmem metronomu, na kterém je možno si zvolit tempo, tato hodnota se potom ručně zapíše do aplikace.

V tabulce 6.1 je uveden přehled úspěšnosti detekce.

	celá melodie		úryvek melodie		
název písně	**tóny (%)**	**tempo**	**tóny (%)**	**tempo**	**Detekce odpovídající skladby**
Běží liška k Táboru	100	NE	100	NE	ANO
Kdyby byl Bavorov	100	NE	100	NE	ANO
Kočka leze dírou	100	NE	100	NE	ANO
Pásli ovce valaši	100	NE	100	ANO	ANO
Skákal pes	100	ANO	100	NE	ANO

Tabulka 6.1: Úspěšnost detekce melodií.

Při testování nebylo třeba měnit nastavení prahu a centrálního klipování u žádné melodie, byly použity výchozí hodnoty. Je vidět, že program ve všech případech rozpoznal odpovídající melodie, úspěšnost je tedy 100%.

Dále bylo testováno rozpoznávání při zhoršené kvalitě nahrávek. Pro simulaci zhoršené kvality byl vytvořen skript V MATLABu s názvem ruseni.m, který umožňuje přidat do nahrávky bílý šum a síťové rušení. Míra šumu se nastavuje zápisem hodnoty do proměnné *k*, která může nabývat hodnot od 0 do 1, přičemž 1 znamená míru šumu 100%, šum pak má nejvyšší možnou amplitudu stejnou jako nejvyšší amplituda v nahrávce. Amplituda síťového rušení se zapisuje do proměnné *rus* a může nabývat hodnot ve stejném rozsahu jako proměnná pro míru šumu. Šum se do nahrávky přidává pomocí funkce `randn`, kterou se vygeneruje vektor s náhodnými hodnotami s délkou rovnou délce nahrávky, vektor šumu se pak vhodně podělí, aby měl odpovídající velikost. Poté stačí tyto dva vektory sečíst. Síťové rušení je simulováno vygenerováním harmonického průběhu o frekvenci 50Hz. K tomuto signálu bylo ještě připočteno 6 vyšších harmonických o poloviční amplitudě než je amplituda harmonického průběhu o frekvenci 50Hz. Tento výsledný signál je přičten k nahrávce.

Nejdříve byl k nahrávkám přičten pouze šum o úrovni 20% z maximální amplitudy nahrávky. V tabulce 6.2 jsou uvedeny výsledky.

	Detekce odpovídající skladby		
název písně	**rušení v celé melodii**	**rušení v úryvku melodie**	**rušení v obou melodiích**
Běží liška k Táboru	ANO	ANO	ANO
Kdyby byl Bavorov	ANO	ANO	ANO
Kočka leze dírou	ANO	ANO	ANO
Pásli ovce valaši	ANO	ANO	ANO
Skákal pes	NE	ANO	NE

Tabulka 6.2: Detekce zašuměných melodií.

Ve sloupci „rušení v celé melodii“ byly zašumělé pouze celé melodie. Ve sloupci „rušení v úryvku melodie“ byly zašumělé pouze úryvky melodií a ve sloupci „rušení v obou melodiích“ byly jak pro uložení do databáze, tak pro rozpoznávání použity zašumělé nahrávky. Použití těchto kombinací zaručuje otestování všech možných případů, které mohou nastat, např. pokud celá melodie nahraje na kvalitním nahrávacím zařízení a úryvek melodie bude nahrán někde jinde na nekvalitním zařízení.

U všech detekcí byl práh pro pomlku změněn na hodnotu 33dB, centrální klipování bylo zachováno výchozí-nulové. U úryvku melodie „Skákal pes“ musel být práh snížen na 30dB, snížení bylo provedeno na základě zobrazení příslušné chybové hlášky. Je patrné, že kromě melodie „Skákal pes“, bylo rozpoznávání u všech možných případů úspěšné. Pro správnou funkci programu tedy ve většině případů můžou být nahrávky s nízkým odstupem signál šum (SNR).

Dále bylo do testovacích nahrávek přidáno síťové rušení. Amplituda rušení byla opět 20% z maximální amplitudy signálu. v tabulce 6.3 jsou uvedeny výsledky.

	Detekce odpovídající skladby		
název písně	**rušení v celé melodii**	**rušení v úryvku melodie**	**rušení v obou melodiích**
Běží liška k Táboru	ANO	ANO	ANO
Kdyby byl Bavorov	ANO	ANO	ANO
Kočka leze dírou	ANO	ANO	ANO
Pásli ovce valaši	ANO	ANO	NE
Skákal pes	NE	ANO	NE

Tabulka 6.3: Detekce melodií se síťovým brumem.

U úryvku melodie „Skákal pes“ byl práh pro pomlku23dB, u všech ostatních byl 33dB. Úspěšnost je obdobná, jako u předchozího případu, navíc byla neúspěšná při brumu obou melodií u „Pásli ovce valaši“. I tak je ale poměrně velká úspěšnost, lze prohlásit, že u většiny melodií síťový brum nemá vliv na rozpoznávání. Síťový brum by bylo možné odstranit notch filtrem, který by musel být vysoce selektivní, protože dle tabulky 3.1 mají tóny „g“ a „gis“ z první oktávy (a samozřejmě tyto tóny z dalších oktáv na vyšších harmonických síťového brumu) frekvence velmi blízko síťového brumu.

V posledním případě byla testována úspěšnost při zašumělých i síťově rušených nahrávek. V tabulce 6.4 jsou uvedeny výsledky.

	Detekce odpovídající skladby		
název písně	**rušení v celé melodii**	**rušení v úryvku melodie**	**rušení v obou melodiích**
Běží liška k Táboru	ANO	ANO	ANO
Kdyby byl Bavorov	ANO	ANO	ANO
Kočka leze dírou	ANO	ANO	ANO
Pásli ovce valaši	ANO	ANO	NE
Skákal pes	NE	ANO	NE

Tabulka 6.4: Detekce zašumělých melodií se síťovým brumem.

U všech nahrávek byl práh pro pomlku 33dB vyjma úryvku melodie „Skákal pes“, kde byl 30dB. Lze si všimnout, že úspěšnost je stejná jako v předcházejícím případě. Větší chybovost tedy způsobuje síťové rušení než šum.

7 ZÁVĚR

Byly probrány metody analýzy hudebních signálů, které v práci budou dále použity pro detekci základní melodie vyjma psychoakustického měření. Dále byly popsány formáty hudebních souborů, jak po stránce použité modulace, tak po stránce formátů používané ve výpočetní technice. Bylo diskutováno, že nejlépe bude použít nekomprimovaný formát, např. Wave form audio format, v nichž jsou uloženy hudební signály, které se mají dále zpracovávat. Na základě popsaných algoritmů kódování ztrátových formátů bylo zjištěno, že je výhodné převádět soubory pouze z bezeztrátových do ztrátových.

Dále byly vysvětleny některé termíny z hudební teorie, které byly použity v dalších kapitolách.

V další obsáhlé kapitole byly vysvětleny, jak jednotlivé algoritmy pro detekci melodie fungují. Bylo zjištěno, že velký vliv na přesnost detekce má určení správného tempa. Největší chybovost vykazovala metoda využívající DFT, ta v některých případech detekovala jako základní tón, tón který byl vyšší harmonickou daného tónu. Tato chyba byla částečně eliminována prahováním hodnot spektra, avšak její nevýhoda je, že u každého nástroje je jiné spektrum a tedy je nutné zvolit jinou prahovací konstantu. Metoda detekce v kepstru vykazovala stejné výsledky jako autokorelační pro obě chromatické stupnice, u posledního testovacího signálu však měla nižší přesnost, její nevýhoda je také výpočetní náročnost. Jako nejlepší metoda byla proto vyhodnocena autokorelační metoda, která jako jediná měla 100% přesnost detekce, u této metody bylo také vyzkoušeno použití centrálního klipování vstupního signálu, avšak u obou chromatických stupnic byly výsledky stejné jako při absenci tohoto algoritmu vyjma 1 tónu, kdy bez centrálního prahování byla 1 hodnota chybná, u posledního testovaného signálu však paradoxně byla horší přesnost při využití centrálního klipování, bez něj i zde byla 100% přesnost. Je tedy možné konstatovat, že algoritmus centrálního klipování má malý vliv na dosažené výsledky. Nad rámec práce byl vytvořen jednoduchý generátor melodie z detekovaných tónů pro sluchovou kontrolu výsledků.

V poslední části práce byly vytvořeny funkce pro ukládání detekovaných melodií databáze a nalezení skladby. Následně bylo vytvořeno uživatelské rozhraní, které umožňuje přidávání skladeb do databáze, rozpoznávání skladeb. Úspěšnost této aplikace na testovacích skladbách byla 100%. Aplikace může nalézt uplatnění u hudebních skladatelů, kdy si můžou své nápady takto archivovat, pokud se pak bude chtít vrátit k nějaké starší melodii a bude si pamatovat jen její část, stačí tuto malou část nahrát a aplikace již najde celou skladbu. Dalším možným využitím je např. v ZUŠ, kdy žák bude mít v databázi aplikace uloženy skladby správně zahrané např. učitelem a může takto jednoduše kontrolovat, zda je zahrál dobře. Při mírné modifikaci algoritmů by se dala aplikace použít např. jako diagnostika mechanických vad materiálů, které jsou charakteristické určitým zvukovým projevem.

LITERATURA

[1] KÁŇA, L., SCHIMMEL, J. *Studiová a hudební elektronika.* 2002.

[2] NOVOTNÝ, V. *Nízkofrekvenční elektronika*: *přednášky* [online]. 1. vyd. Brno: VUT FEKT, 2002, 114 s. [cit. 2012-04-26]. ISBN 80-214-2234-3.

[3] JAN, J. *Číslicová filtrace, analýza a restaurace signálů.* vědecké monografie. vědecké monografie. Brno: VUTIUM Brno, 2002. 427 s. ISBN: 80-214-1558- 4.

[4] TUROŇ, V., JANÍK J., ŠPETÍK R., SOVKA P. a VLČEK M.. Porovnání dvou spektrálních metod pro analýzu akustických signálů. In: *Akustické listy*. Praha, 2011, s. 26-30. ISSN 1212-4702.

[5] R2012a Documentation → MATLAB: Fast Fourier Transform (FFT). MATHWORKS, Inc. [online]. [cit. 2012-04-30]. Dostupné z: http://www.mathworks.com/help/techdoc/math/brentm1-1.html

[6] ROUBAL, P. *Fotografie, hudba a video ve Windows Vista*. Vyd. 1. Brno: Computer Press, 2007, 256 s. ISBN 978-80-251-1859-7.

[7] VALOUŠEK, P. Jednoduchá konverze audio formátů: DSD na PCM. In: *Audio Technologies and Processing*: *ATP 2005 : proceedings of the 6th conference of Czech student AES, Section on Audio Technologies and Processing*. Vyd. 1. V Brně: Vysoké učení technické, 2005, s. 116-122. ISBN 80-214-2925-9.

[8] HOLAKOVSKÝ, J. Hudební formáty část 1 - obecná komprese. *Hudební formáty část 1 - obecná komprese* [online]. [cit. 2012-04-30]. Dostupné z: http://mag.repro.cz/view.php?cisloclanku=2004011704

[9] BĚHUNEK, M. *Rozpoznávání řeči při různé kvalitě vstupního signálu* [online]. Praha, 2010 [cit. 2012-05-02]. Dostupné z: http://noel.feld.cvut.cz/speechlab/publications/056_diplomka11.pdf. Diplomová práce. České vysoké učení technické v Praze.

[10] ADAM, P. *Úvod do metód spracovania zvuku v súčasnom multimediálnom prostredí* [online]. Bratislava, 2006 [cit. 2012-05-03]. Dostupné z: http://www.dcs.fmph.uniba.sk/diplomovky/obhajene/getfile.php/dipl.pdf?id=107&fid=177&type=application%2Fpdf. Diplomová práce. Univerzita Komenského, Bratislava.

[11] ZENKL, L. *ABC Hudební* nauky, Editio Bärenreiter Praha, 2003

[12] PLEVA, P. *Převod zvuku do MIDI formátu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 52 s.

[13] KRUPIČKA, J. *Převod not jednohlasé melodie ze zvukového signálu do protokolu MIDI.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologii, 2009. 52 s.

[14] CHENG, K., B. NAZER, J. UPPULURI a R. VERRET. *Beat This: A Beat Synchronization Project* [online]. 2001 [cit. 2012-12-04]. Dostupné z: http://www.clear.rice.edu/elec301/Projects01/beat_sync/beatalgo.html

[15] SCHIMMEL, J. Komunikační protokol MIDI. *Elektrorevue - Internetový časopis (http://www.elektrorevue.cz)*, 2002, roč. 2002, č. 69, s. 1 (s.)ISSN: 1213- 1539.

[16] SOVA, J. *Hlasové pole*. Praha, 2008. 83 s. Bakalařská práce. České vysoké

učení technické v Praze.

[17] ŠMIDÁK, M. *ABSOLUTNÏ SLUCH*. Praha: Hudební fakulta Akademie múzických umění v Praze, 2005.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

s	Signál v časové oblasti
s(n)	Signál v diskrétní podobě
F	Signál ve frekvenční oblasti
M	Počet segmentů
N	Počet vzorků
ADPCM	Adaptive differential pulse-code modulation, adaptivní pulsně kódová modulace
A/D	Analog to Digital Converter, analogově digitální převodník
BPM	Beats per minute, úderů za minutu
D/A	Digital to Analog Converter, digitálně analogový převodník
DFT	Discrete Fourier Transform, diskrétní Fourierova transformace
FFT	Fast Fourier Transform, rychlá Fourierova transformace
MDCT	modified discrete cosine transform, modifikovaná Kosinova transformace
PCM	pulse-code modulation, pulsně kódová modulace
SNR	poměr signál/šum.

SEZNAM PŘÍLOH

A Obsah CD 45

Adresáře:

- Melodie - Obsahuje zvukové nahrávky testovacích melodií.
- Program - Obsahuje všechny soubory potřebné k fungování programu.
- Text - Obsahuje samotný text této práce včetně obrázku apod. ve formátu pdf a také ve formátu doc.