VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ V HUDEBNÍCH SIGNÁLECH

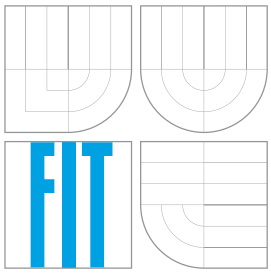
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. FRANTIŠEK SKÁLA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ V HUDEBNÍCH SIGNÁLECH

SEARCH IN MUSIC SIGNALS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. FRANTIŠEK SKÁLA
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. JAN ČERNOCKÝ
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce obsahuje přehled metod používaných v oblasti získávání informací z hudby, zejména pro účely vyhledávání hudebních nahrávek. Představeno je několik již existujících služeb, které se vyhledáváním a identifikací nahrávek zabývají, a jsou popsány jejich metody pro identifikaci nahrávky. Práce se dále zabývá možnými úpravami těchto postupů pro vyhledávání cover verzí písniček a pro možnost hledání na základě hlasem zadávaných vzorků.

Abstract

This work contains overview of methods used in the area of Music Information Retrieval, mainly for purposes of searching of musical recordings. Several existing services in the areas of music identification and searching are presented and their methods for unique song identification are described. This work also focuses on possible modifications of these algorithms for searching of cover versions of songs and for the possibility of searching based on voice created examples.

Klíčová slova

MIR, Hudba, Získávání informací z hudby, Vyhledávání v hudbě, Hledání dle vzoru, Otisk hudby, Shazam, Vyhledávání hlasem, Vyhledávání cover verzí

Keywords

MIR, Music, Music Information Retrieval, Searching in music, Query by example, Music fingerprint, Shazam, Query by humming, Song cover searching

Citace

František Skála: Vyhledávání v hudebních signálech, diplomová práce, Brno, FIT VUT v Brně, 2012

Vyhledávání v hudebních signálech

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením docenta Černockého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
František Skála
21. května 2012

Poděkování

Chtěl bych poděkovat docentu Janu Černockému za odbornou asistenci při tvorbě práce. Také bych chtěl poděkovat své přítelkyni Janě Staurovské za psychickou podporu a naslouchání.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Tato práce se zabývá vyhledáváním v hudbě a s tím souvisejícím získáváním informací z hudby, jejich klasifikací a možnostmi jejich indexace. Vzhledem k nepřebernému množství hudby, která je nám dnes k dispozici a se kterou se denně setkáváme, je v oblasti zpracování signálů stoupající potřeba hudbu vyhodnocovat. Poslech hudby z rádií je dnes spíše na ústupu, většina uživatelů si chce přesněji zvolit, jakou hudbu chtějí poslouchat. Ať už se jedná o příznivce určitého žánru (rock, metal, hip-hop a jiné), nebo posluchače, kteří volí rozmanitou hudbu dle aktuální nálady, či si chtějí zrovna pustit svojí oblíbenou písničku. Zpracování signálu může napomoci právě s klasifikací hudby a usnadnit uživateli výběr. Existují techniky pro stanovení žánru nahrávky, stejně jako porovnání vztahu skladby k náladě. Tato práce se však zabývá situací, kdy uživatel ví, kterou skladbu chce poslouchat, jen ji nemá k dispozici a neví, jak ji identifikovat.

Typickým scénářem je například situace, kdy je uživatel v restauraci a v rádiu slyší písničku, která se mu zalíbí. Dané rádio nenabízí veřejně seznam skladeb, aby si ji mohl posluchač zpětně dohledat, a tak je uživatel odkázán pouze na jediný zdroj informací – tím je samotná hudba. Například mobilním telefonem pořídí nahrávku, kterou odešle na specializovaný server, jenž mu vzápětí odpoví, o kterou skladbu se jedná, ideálně s nabídkou na její stažení. V takovém případě se jedná o vyhledávání dle vzoru, viz sekce 3.2.2.

Další běžnou situací pro vyhledávání hudby je, když uživateli „zní v hlavě“ nějaká písnička. Zná melodii, případně i slova, avšak neví, o jakou konkrétně skladbu se jedná. Je tedy schopen úsek skladby, který si pamatuje, zazpívat či jinak zreprodukovat. Pak už je to podobné, jako v předchozím případě – pořídí nahrávku své interpretace, odešle a obdrží výsledky. Liší se však struktura vstupních dat, stejně jako algoritmy na vyhledání skladby – jedná se o „Query by humming“ (volně přeložitelné jako vyhledávání brumendem), viz sekce 3.2.2.

Poslední situací, kterou se tato práce zabývá, již předpokládá, že má uživatel nějakou kompletní nahrávku k dispozici. Jedná se o obdobnou skladbu, jakou hledá, ale například od jiného interpreta či v jiné verzi. Zde se ve vyhledávání mísí oba předchozí přístupy – vstupem je kompletní skladba, ne jen významná notová linka, je však třeba na nahrávku nahlížet jako na hudbu složenou z tónů hraných na nástroje, nikoli jako na samotný audiosignál, viz sekce 3.2.3.

1.1 Členění práce

V první části práce jsou uvedeny atributy, na základě kterých lze hudbu z hlediska MIR[1] zkoumat (kapitola 2). Pro uvedení čtenáře do problematiky MIR slouží kapitola 3. Jsou zde popsány možnosti využití MIR v praxi, dostupné služby a jejich metody. Hlubší rozbor algoritmů, které jsou implementovány jako součást této práce je v kapitole 4). V další části (kapitola 5) je zběžný přehled o použitých datech, jejich získání a zhodnocení vztahu k implementaci.

Tato práce se zabývá moderní západní hudbou, zejména pro její rozšíření a popularitu v Evropě. I přesto by většina metod použitých v této práci měla bez nutnosti větších modifikací fungovat i na ostatních formách hudby, jako je například hudba orientální, testování se na ně však nezaměřovalo.

Očekává se, že čtenář má základní přehled v oblasti teorie signálů[2]. Stejně tak se předpokládá alespoň základní znalost hudebních pojmů.

[1] Music Information Retrieval – Získávání informací z hudby

[2] Většina z potřebných pojmů je vysvětlena například v materiálech k předmětu Signály a systémy na FIT VUT: https://www.fit.vutbr.cz/study/courses/ISS/public/.cs

Kapitola 2

Vlastnosti hudebních děl a jejich komponent

Dnešní populární hudba je většinou tvořena zpívanou části, melodiemi hranými na nejrůznější nástroje (např. kytara, klavír, ale i syntetizéry) s množstvím doprovodných nástrojů a efektů. Chceme-li hudbu vyhodnocovat strojově, je vhodné se zabývat vlastnostmi nejen hudby jako celku, ale i vlastnostmi jejích jednotlivých složek.

2.1 Vlastnosti hudebních děl

Na hudební díla a jejich konkrétní nahrávky lze nahlížet v několika směrech, dle kterých se liší zkoumané vlastnosti.

Nahlížíme-li na nahrávku jako na signál, můžeme zkoumat například hlasitost nahrávky, její spektrum v čase, směrodatnou odchylku energie a podobně.

Při pohledu na nahrávku z hudebního hlediska nás však zajímají zcela jiné vlastnosti. Následující atributy jsou důležité pro účely získávání informací z hudby (čerpáno z [5]), zejména pro metody použité v tomto textu:

Tónina Náležitost k určité hudební stupnici, tedy z jaké stupnice převládají jednotlivé tóny ve skladbě

Rytmus Střídání přízvučných a nepřízvučných různě dlouhých not, v populární hudbě je většinou vztažen zejména k časové složce bicího doprovodu

Tempo Rychlost skladby, tedy vztah hudebních délek k času, v populární hudbě obvykle měřeno v jednotkách BPM[1]

Dynamika Relativní hlasitost skladby a změny hlasitosti v čase

Textura Vztah a množství jednotlivých nástrojů, z hlediska tohoto textu je nejvýznamnější dělení na monofonní[2] a polyfonní[3] hudbu

Struktura Členění jednotlivých části skladby – například. refrén, úvod, sólo apod.

[1] BPM – Beats Per Minute – počet úderů za minutu, v hudbě obvykle počet čtvrťových not (tedy dob) za minutu

[2] Monofonní hudba – je obsažen pouze jeden jediný nástroj, není hráno více not současně

[3] Polyfonní hudba – je zastoupeno více nástrojů, které hrají souběžně

Tyto atributy lze objektivně určit, v notovém zápisu skladeb jsou většinou všechny zadány (byť nepřímo). V klasifikaci pro posluchače se často používají i subjektivněji hodnotitelné atributy:

Žánr Styl, druh hudby – kategorizace do žánrů vychází z určitých atributů hudby, jako jsou použité nástroje, tempo, rytmus a jiné

Nálada Vztah hudby k náladě posluchače (veselé, smutné, ale i jiné skladby)

Pro vyhledávání konkrétní skladby však nemají zásadní význam, posluchač je využije spíše při třídění své existující kolekce.

2.2 Atributy hudebních složek

Zejména pro vyhledávání, při kterém nepracujeme s hudebním signálem jako celkem, ale je ho třeba rozdělit na konkrétní složky, je třeba i tyto klasifikovat. Typickým příkladem je vyhledávání cover verze skladby – může obsahovat stejnou ústřední melodii, avšak zahranou na jiný nástroj, mít podobný doprovod, ale celkově být v rychlejším tempu.

Mezi zásadní atributy patří:

Melodie Posloupnost jednotlivých not v čase, tedy jejich délky, výšky, ale i pauzy mezi nimi

Barva Spektrum konkrétního nástroje, tedy vztah hlasitosti jednotlivých harmonických frekvencí, spolu s časovým průběhem této charakteristiky

Hlasitost Odpovídá energii signálu, její vnímání člověkem však není na energii lineárně závislé

Výška Výška zvuku nástroje v konkrétním čase odpovídá frekvenci jeho nejnižší harmonické frekvence

Tóny Vyjádření výšky v konkrétním čase pomocí hudební stupnice (tedy pomocí not), vztah mezi tóny a frekvencí zvuku je logaritmický

Harmonie Souznění několika tónů, například při hraní akordů

Kapitola 3

Existující služby využívající Music Information Retrieval

Music Information Retrieval (do češtiny přeložitelné jako získávání informací z hudby) je obor, který se zabývá získáváním informací obsažených v hudbě. Zasahuje do něj nejen zpracování signálů, ale i jiné vědní obory, zejména z oblasti hudební teorie. Tato práce se však zaměřuje právě na možnosti automatického vyhodnocování metodami zpracování signálů.

V dnešní době existuje mnoho služeb a nástrojů, které usnadňují posluchači výběr hudby, třídění kolekce a vyhledávání. Využívají různé způsoby získávání informací o hudbě, se kterou pracují, což většinou odpovídá i jejich zaměření a nabízeným možnostem.

3.1 Služby založené na lidském vstupu

Za jednu kategorii služeb využívajících MIR jsou služby bez automatického zpracování hudby jako takové, které využívají metody založené na vstupu od uživatelů.

Jednou z těchto služeb je americká *Pandora*[1]. Cílem této služby je nabídnout posluchači pro něj neznámou hudbu, která je však podobná jeho oblíbené. Po vstupu na stránku je uživatel dotázán na jeho oblíbenou kapelu či písničku. Na základě jeho volby jsou mu poté přehrávány písničky, které jsou považovány za podobné. Pandora vybírá tyto skladby na základě experty zadaných metadat, tudíž její databáze nedosahuje takových rozměrů, jako jiné služby. Vytvoření metadat o jedné písničce totiž zabere 20–30 minut[2] [5].

Podobnou službou je i *last.fm*[3]. To však metadata nezískává na základě expertů, nicméně je také závislé na lidském vstupu. Vztahy mezi jednotlivými skladbami jsou budovány na základě toho, jak si je jednotliví uživatelé přehrávají. Služba *last.fm* využívá předpokladu, že uživatelé si většinou sami budují seznamy skladeb na základě podobnosti hudby, tyto vztahy jsou tedy uloženy a používány dále, když posluchač zvolí poslech automaticky vytvořené „radiostanice“. Jelikož je *last.fm* postaveno i jako sociální síť, má již v dnešní době velmi mnoho posluchačů, tudíž má poměrně bohatou databázi vztahů mezi skladbami [5].

[1] http://www.pandora.com, dostupná pouze uživatelům z USA
[2] Údaj je z roku 2008
[3] http://www.last.fm, přehrávání je zpoplatněno

3.2 Služby založené na strojovém zpracování

V této kategorii služeb jsou dva základní přístupy. První z nich, podobně jako služby založené na lidském vstupu, slouží převážně k volbě hudby, kterou chce uživatel zrovna poslouchat. Nepracují však s nějakou centralizovanou databází, ale přímo s kolekcí, kterou má uživatel na svém počítači. Druhá kategorie naopak centrální databázi používá, slouží však převážně k vyhledávání konkrétních skladeb na základě dotazu od uživatele.

3.2.1 Nástroje na získávání metadat

Víceméně každý hudební přehrávač nabízí alespoň nějaké možnosti pro filtraci hudby – výběr dle interpreta, žánru a podobně. Jednoduché nástroje si vystačí s metadaty, která jsou v hudebních souborech mimo hudby samotné již obsažena. Pokročilejší nástroje pak umožňují tato metadata automaticky získat. Služby pro získávání metadat využívají databáze uložené na centrálním serveru, ze kterého jsou metadata získávána na základě unikátního identifikátoru konkrétní písničky. Metadata na centrálním serveru pochází ve většině případů od komunity uživatelů. Jednotlivé služby se však liší zejména způsobem vytvoření identifikátoru (otisku) konkrétní nahrávky, na jehož základě jsou poté metadata získávána.

Služba *Gracenote*[4] využívá pro identifikaci písničky vlastnosti jejího spektra získaného diskrétní kosinovou transformací. Používá okno o velikosti tří sekund s krokem po polovině vteřiny, v daném okně si pak extrahuje ze spektra informace jako amplituda dané frekvence, průměrná hodnota, směrodatná odchylka a další. Tyto informace jsou pro každou skladbu uchovávány v databázi. Při vyhledávání je pak stejným způsobem analyzována celá vyhledávaná nahrávka. Na základě otisku spektra je pak hledán záznam v databázi. Při hledání se využívá i hodnot směrodatné odchylky, neboť se metoda musí umět vypořádat se změnami ve spektru nahrávky, které jsou způsobeny její možnou ztrátovou kompresí (ať už podvzorkováním, nebo využitím pokročilejší komprese typu mp3) [17]. Tyto změny se projevují buď jen v části spektra, nebo jsou některé frekvence pouze utlumeny. Služba se však nevyrovná se zašumělou nahrávkou, neboť v takové dochází ke změnám průměrné energie i její směrodatné odchylky napříč celým spektrem. Pro vyhledání je také potřeba kompletní skladba.[5] Na druhou stranu, pro nahrávky získané pouze kompresí originálu získaného například z CD funguje služba velmi dobře. Za zmínku stojí i fakt, že pro zrychlení a zpřesnění hledání se používají již známá metadata (například název souboru a cesta k němu, případně i ID3 tagy, jsou-li přítomny) [3].

Identifikaci písničky nabízí i služba *MusicBrainz*[6]. Pro identifikaci nahrávky slouží unikátní otisk, který je pro konkrétní nahrávku uložen v databázi služby. Pro jeho získání je využita časově–frekvenční matice získaná pomocí FFT[7], ze které je následně získaná matice $\mathbf{V}$ algoritmem SVD[8]. Z této matice $\mathbf{V}$ je pak vytvořen výsledný identifikátor, kterým je nahrávka označována ve výsledcích vyhledávání. Samotné vyhledávání však využívá přímo spektrum signálu – jsou v něm vyhledávány špičky, kterým jsou následně dle frekvence přiřazovány příslušné čísla not odpovídající MIDI formátu. Nejvýznamnější z těchto not

[4] http://www.gracenote.com/

[5] V roce 2011 ohlásila služba Gracenote rozšíření služby MusicID o možnosti integrace do domácích kin – služba tedy nově nabízí i vyhledávání na základě krátkého vzorku písničky i s potencionálním šumem prostředí (hudba ve filmech či televizních přenosech) [2].

[6] http://musicbrainz.org/

[7] FFT – Fast Fourier Transform – rychlá Fourierova transformace

[8] SVD – Singular Value Decomposition – viz například http://www.math.muni.cz/~zelinka/cjh/svd.pdf

jsou pak použity jako indexy písničky pro její vyhledávání [8].

Služba *MusicBrainz* má za cíl postupně přejít na kompletně opensource řešení, snaží se tedy začlenit algoritmus *AcoustID*[9]. Ten využívá podobně jako původní algoritmus spektrogram namapovaný na MIDI noty. Na výsledný spektrogram, na který lze nahlížet jako na bitmapový obrázek, je posuvným oknem o velikosti 16x12 pixelů aplikováno 16 filtrů, z nichž každý produkuje dvoubitovou odezvu. V každém posuvu okna se tedy získá 32 bitů, jejich skládáním za sebe pro zkoumaný úsek nahrávky získáme kompletní otisk. Při porovnávání skladeb pak lze použít jako hodnotící funkci pro dva otisky počet bitů, které jsou shodné, viz grafické znázornění na obrázku 3.1 [10].

(a) AcoustID pro skladbu Heaven - FLAC verze

(b) AcoustID pro skladbu Heaven - mp3 verze

(c) XOR předchozích 2 otisků

(d) Ukázka rozdílu dvou odlišných skladeb (Heaven a Under The Ice)

Obrázek 3.1: Ukázky grafického znázornění AcoustID konkrétních nahrávek a porovnání dvou rozdílných souborů. Obrázky jsou přejaty z [10].

3.2.2 Nástroje na identifikaci nahrávky dle uživatelského vzorku

Předchozí kategorie služeb je již integrovaná do mnoha moderních hudebních přehrávačů, nebo jsou alespoň dostupné pomocí specializovaných nástrojů. Pro většinu posluchačů jsou tak již samozřejmostí. Poněkud méně rozšířenou kategorií jsou pak nástroje, které nahrávku identifikují dle krátkého vzorku získaného z reálného prostředí. Tyto služby slouží uživatelům pro identifikaci konkrétní nahrávky, kterou však uživatel nemá k dispozici. Dělí se do dvou kategorií na základě druhu vstupních dat. První kategorie identifikuje na základě nahrávky úseku skladby, kterou uživatel slyší reprodukovanou někde v reálném prostředí. Služby se tak musí vypořádat se šumem a rušivými vlivy. Druhá kategorie pak používá pro vyhledávání pouze melodii zadanou uživatelem například pomocí zpěvu či pískání. Oproti všem předchozím diskutovaným metodám se nemohou spolehnout na fingerprinting na základě spektra signálu, neboť uživatel zadává vstup pouze svým hlasem, kterým je schopen podat pouze základní melodii, a i tu jen značně nepřesně.

Služby se vstupem z reálného prostředí Tuto kategorii služeb využijí posluchači, kteří chtějí zjistit, jakou skladbu jim rádio či televize hraje právě v daném okamžiku. Ty-

[9] http://acoustid.org/

picky pomocí mobilního telefonu nahrají krátký úsek, který je zrovna přehráván, a odešlou ho na server služby, který jim vzápětí vrátí odpověď – název skladby, interpreta a případně i odkaz na možnost nákupu hudebního souboru. Jelikož se typicky vstup zadává mobilním telefonem a je následně přenášen datovým přenosem na centrální server ke zpracování, využívají se k omezení potřebného datového přenosu a doby odezvy nahrávky dlouhé pouze několik vteřin. Dalším důvodem pro práci pouze s několikasekundovým úsekem skladby je fakt, že nahrávání a tedy i zpracování nemůže probíhat rychleji, než v reálném čase.

Jednou z nejrozšířenějších služeb tohoto druhu je *Shazam*, který funguje od roku 2002. Své služby začal poskytovat uživatelům mobilních telefonů ve Velké Británii přes GSM síť. Použití bylo velmi jednoduché – zákazník zavolal na telefonní číslo služby, nasměroval mobilní telefon na zdroj hudby a služba začala s identifikací. Po pár vteřinách byl hovor ukončen a poté dostal klient v textové zprávě informaci o názvu interpreta a skladby. Dnes už služba nevyžaduje použití placeného volání, nabízí ke stažení za jednorázový poplatek aplikaci, která umožňuje identifikaci nahrávky za použití připojení přes internet. Navíc ale nabízí rovnou i odkaz na zakoupení skladby, například pomocí služby iTunes [18].

Shazam je tedy zaměřený na krátké nahrávky, které jsou nahrávány mobilním telefonem v reálném prostředí a poté projdou GSM sítí. Pro otisky tak nelze zvolit přímo parametry celého spektra, jak tomu bylo u algoritmů v sekci 3.2.1, ale je nutno volit takové parametry, které uvedené zkreslení neovlivní. Shazam tedy ve spektru vyhledává pouze nejvýraznější špičky. Nejprve vyhledá špičky (lokální maxima) průběhu energie celého signálu, kterých je vzhledem ke struktuře hudby obvykle dostatečné množství – obvykle několik v každé sekundě nahrávky. V okamžiky těchto špiček v energii pak hledá maximum na ose frekvenční. Z výsledných hodnot je pak pro Shazam klíčový pouze čas energetické špičky a frekvence s maximální energií. Samotné špičky jsou většinou dostatečně výrazné, takže jsou zachovány i v případě přítomného šumu z prostředí. Zkreslení způsobené reprodukcí, nahráním na mobilní telefon a přenosem po GSM síti jsou pak kompenzovány právě použitím pouze času a frekvence, nikoli amplitudy.

Pro zvýšení entropie otisků jsou pak používány dvojice těchto špiček spolu s informací o jejich rozestupu. Spektrogram s rozlišením 1024 frekvenčních pásem totiž poskytuje pouze devět bitů informace o pozici špičky. Přidáním druhé špičky z určité oblasti následující po první špičce pak získáme dalších devět bitů, spolu s informací o časovém rozestupu těchto dvou špiček má pak každý otisk v rámci nahrávky 26 bitů.

Výhoda vytváření jednotlivých nezávislých otisků je zejména dosažitelná rychlost při vyhledávání – v případě použití hashovací tabulky je časová složitost vyhledání jednoho otisku konstantní. S velikostí databáze pak tedy narůstá jen počet písniček, které obsahují stejné otisky, a ze kterých je tedy třeba sestavovat hodnotící histogramy.

Přesnější popis jednotlivých fází tohoto algoritmu je detailně uveden v sekci 4.1

Vyhledávače na základě uživatelovy interpretace Další kategorie služeb vyhledává na základě uživatelova podání skladby, což může být například zpěv, pískání nebo brumendo. V případě zpěvu lze extrahovat i slova a využít je ke zpřesnění hledání, ale ve zbylých scénářích se musí vyhledávač omezit pouze na melodii a rytmus.

Komerčně úspěšná je služba *SoundHound*[10], která je přístupná pomocí aplikace do chytrých mobilních telefonů. K vyhledávání využívá databáze služby *Midomi*[11], která obsahuje přes dva miliony písniček nazpívaných uživateli [1]. Vyhledávání porovnává interpretaci

[10] http://www.soundhound.com
[11] http://www.midomi.com/

uživatele s interpretacemi uloženými v databázi. Na základě nalezené interpretace je pak jako výsledek hledání zvolena k ní přiřazená předloha. Služba tedy pro vyhledávání nezískává žádná data z jednotlivých skladeb, omezuje se pouze na k nim přiřazené uživatelské interpretace. Vytvoření takto rozsáhlé databáze se službě Midomi podařilo díky tomu, že sloužila jako server sdružující amatérské zpěváky a získávala jejich karaoke nahrávky.

3.2.3 Nástroje na vyhledávání cover verzí

Možností vyhledat cover verzi nahrávky jsou v současné době značně omezené. Existující služby, jako například *Second Hand Songs*[12], totiž nenabízí automatizované hledání cover verzí. Využívají ručně anotované databáze a hledání na základě textového vstupu, tedy názvu skladby či autora, nikoli na základě automatické analýzy hudebního díla. Ani v oblasti hledání cover verzí v rámci uživatelovy vlastní kolekce neexistuje žádný úspěšný nástroj.

Problematika hledání cover verzí je podobná vyhledávání na základě zpěvu – je třeba nalézt skladby s podobnou melodií. Ovšem při hledání cover verzí je třeba analyzovat vždy několik melodických linek a hledat možnosti shody mezi různými nástroji. Vzhledem k tomu, že ani nejúspěšnější služba z oblasti vyhledávání na základě zpěvu nevyužívá hudební charakteristiky originálních skladeb, pouze jejich uživatelských podání, je zřejmé, že úloha vyhledávání cover verzí není triviální. Spolu se zkvalitňováním nástrojů na extrakci not z polyfonních melodií má však budoucnost.

[12] http://www.secondhandsongs.com/

Kapitola 4

Vlastní implementace vyhledávání v hudbě

V této kapitole jsou popsány zvolené systémy, shrnuty jejich možné výhody a nevýhody a obsahuje přesný popis použitých algoritmů.

Pro vyhledávání na základě vzorku z reálného prostředí byl zvolen algoritmus služby Shazam, který již byl popsán v sekci 3.2.2. Pro vyhledávání na základě zpěvu se dostupné služby nehodily (vzhledem k nutnosti existence rozsáhle databáze identifikovaných nazpívaných nahrávek), proto byl zvolen postup vyhledávání pomocí extrakce melodie. Modifikace obou těchto metod pak byly použity pro vyhledávání cover verzí.

4.1 Shazam

Služba Shazam již byla zběžně popsána v sekci 3.2.2. Jednotlivé charakteristiky algoritmu byly však pouze zevrubné, zde je uveden přesný postup získání otisků ze signálu:

1. Vstupní audiosignál je převzorkována na 8000 Hz a převeden na jednokanálový (mono).

2. Je vytvořena časově-frekvenční matice **A** (spektrogram s lineární reprezentací amplitud) pomocí FFT s použitím Hannova okna o velikosti 1024 vzorků a krokem 64 vzorků, viz algoritmus 1[1]. Funkce $\mathrm{cplx}(x)$ v uvedeném algoritmu vrací komplexní číslo s reálnou složkou x a nulovou imaginární složkou, funkce $\mathrm{hann}(k, size)$ vrací váhu k-tého prvku Hannova okna velikosti $size$ a funkce $\mathrm{FFT}(arr, size)$ vrací pole komplexních koeficientů rychlé Fourierovy transformace pole arr velikosti $size$. Příklad výsledného spektrogramu je na obrázku 4.1a.

3. Pro každý časový okamžik získané matice **A** je spočítána L4-norma ($\|X\|_4$):

$$\|X\|_4 = \left(\sum_{n=0}^{N-1} |X_n|^4 \right)^{1/4} \tag{4.1}$$

   Shazam se při výpočtu L4-normy omezuje pouze na frekvence od 500 Hz do 2500 Hz. V získané posloupnosti L4-norem (obrázek 4.1b) jsou nalezena lokální maxima pomocí algoritmu 2. Jejich výskyty v čase jsou poté použity jako *landmarky* **l** (obrázek 4.1c).

[1] Postup vytvoření časově-frekvenční matice by měl být čtenáři znám, je však uveden pro ozřejmení výsledné struktury pole **A**, které je využíváno v dalších krocích algoritmu.

4. V časové okamžiky odpovídající landmarkům jsou v matici **A** nalezena ve frekvenční ose frekvence maxim **m** absolutních hodnot koeficientů (vždy frekvence jednoho maxima na jeden landmark, opět pomocí algoritmu 2; obrázek 4.2a). Tato maxima jsou opět vybírána jen ve frekvenčním rozsahu 500-2500 Hz.

5. Ke každému takto získanému bodu je přiřazen každý další, který následuje nejméně 1 a nejvíce 3 vteřiny poté. Ze získaných dvojic je vytvořeno jediné binární číslo, obsahující po řadě 9 bitů informace o frekvenci první špičky, 8 bitů pro informaci o rozdílu času T těchto dvou špiček a dalších 9 bitů informace o frekvenci druhé špičky. Toto párování je znázorněno algoritmem 3 a obrázkem 4.2b.

6. Získané otisky **h** jsou uloženy do databáze spolu s označením skladby a času, ve kterém se otisk v nahrávce nachází.

Algoritmus 1 Postup získání časově-frekvenční matice **A** ze vstupního signálu **s** (Hannovo okno velikosti 1024, krok 64 vzorků).

for $i = 1 \rightarrow \text{size}(s)/64$ **do**
  for $j = 0 \rightarrow 1024$ **do**
    if $i{\times}64 + j > \text{size}(s)$ **then**
      $tmp[j] \leftarrow \text{cplx}(0)$
    else
      $tmp[j] \leftarrow \text{cplx}(s[i{\times}64 + j]{\times}\text{hann}(j, 1024))$
    end if
  end for
  $A[i] \leftarrow \text{FFT}(tmp, 1024)$
end for

Algoritmus 2 Postup získání lokálních maxim ze signálu **s** za využití filtrů – formy zápisu polí a uvedené funkce odpovídají prostředkům dostupným v Matlabu.

$tmp \leftarrow \text{filter}([1\ -1], 1, \mathbf{s})$
$tmp \leftarrow \text{sign}(tmp)$
$tmp \leftarrow \text{filter}([-1\ 1]/2, 1, tmp)$
$tmp \leftarrow [tmp(2 : \text{size}(S)); \text{zeros}(1, 1)]$
$localmaxs \leftarrow \text{find}(tmp == 1)$

Postup získání otisků je znázorněn vývojovým diagramem na obrázku 4.3.

Při vyhledávání jsou pak stejným způsobem získány otisky ze vstupního vzorku a poté jsou v databázi vyhledány odpovídající skladby (které obsahují libovolné množství z těchto otisků). Jako hodnotící funkce pro výběr nejvhodnější skladby se pak používá maximum v histogramu, kam jsou zaneseny hodnoty $T_{database} - T_{sample}$. $T_{database}$ je čas výskytu zkoumaného otisku v celé skladbě uložené v databázi a T_{sample} je čas, kdy se daný otisk vyskytuje ve vzorku (viz obrázek 4.5). Nejvyšší skóre tak má nahrávka, ve které se jednotlivé otisky vyskytují se stejnými rozestupy, jako v databázová nahrávce, přičemž se může jednat o libovolný úsek. Postup vyhledávání je znázorněn vývojovým diagramem 4.4.

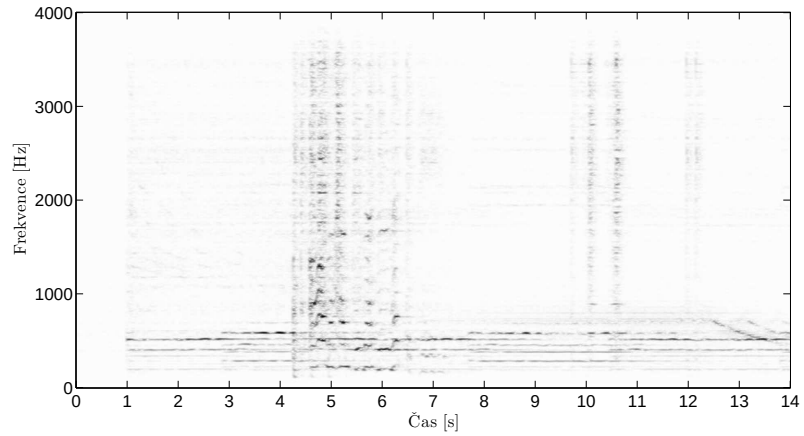
Aby bylo možné Shazam v praxi otestovat a experimentovat s jeho modifikacemi, byl implementován jako součást této práce dle výše uvedených algoritmů. Rozpoznávač byl im-

Algoritmus 3 Algoritmus na vytvoření párů **h** špiček z časově-frekvenční matice **A** na základě časů landmarků **l** a hodnot špiček **m** v těchto landmarcích

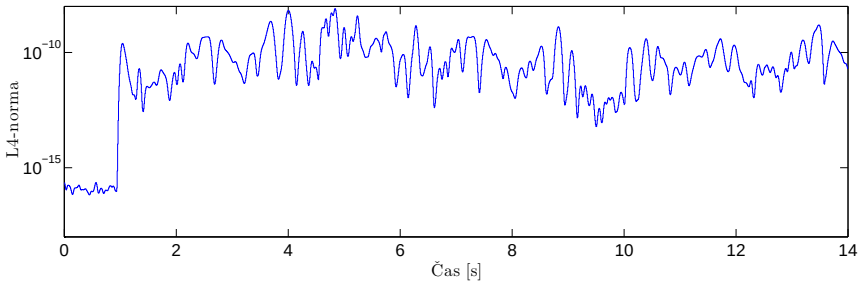
```
for all i = 1 → size(l) do
    for j = i → size(l) do
        t1 ← li
        t2 ← lj
        M1 ← mt1
        M2 ← mt2
        if t2 − t1 ≥ 1 & t2 − t1 < 3 then
            hash ← ((M1 ≪ 8) + (t2 − t1)) ≪ 9 + M2
            h.append(hash)
        end if
    end for
end for
```

plementován v jazyce C s využitím knihovny `libsndfile`[2] a `libsamplerate`[3]. Pro uchování dat a dotazování na ně byla použita databáze MySQL[4] a na její obsluhu posloužily shell scripty v jazyce PHP.

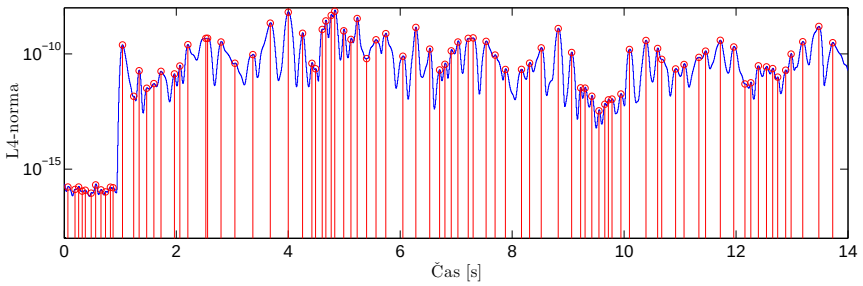
[2] http://www.mega-nerd.com/libsndfile/
[3] http://www.mega-nerd.com/SRC/
[4] http://www.mysql.com/



(a) Spektrogram získaný ze skladby Undone skupiny Artemis. Skladba je obsažena v datasetu Magnatune.

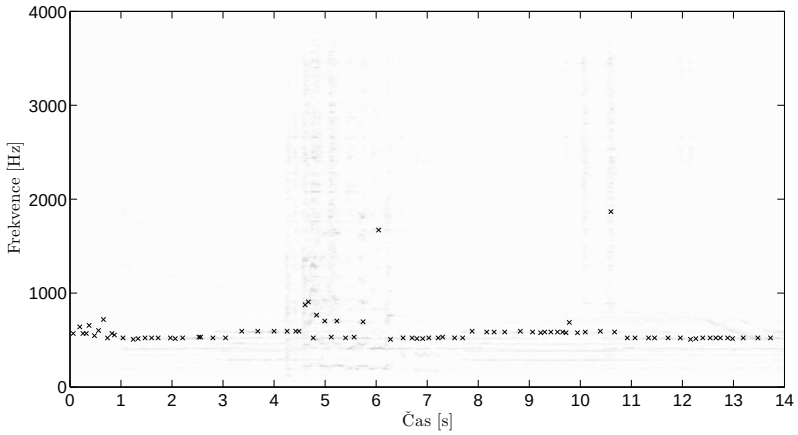


(b) Průběh L4-normy v získaném spektrogramu

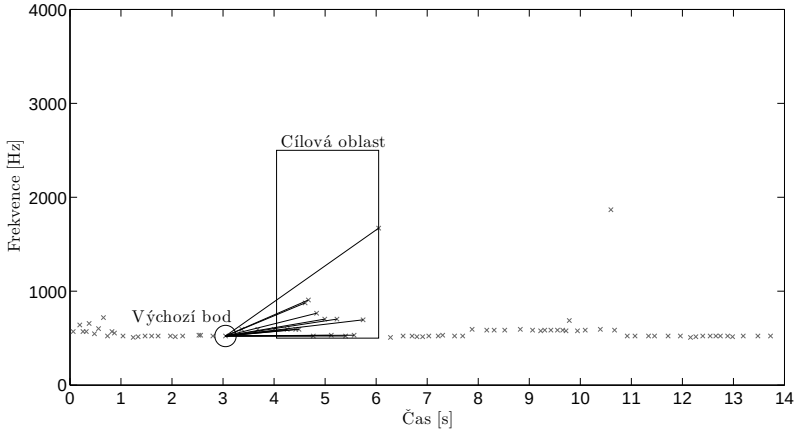


(c) Landmarky získané jako špičky průběhu L4-normy

Obrázek 4.1: Způsob nalezení landmarků algoritmem Shazam

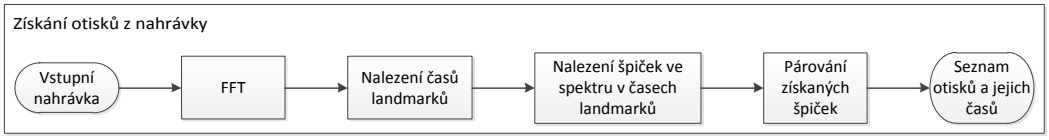


(a) Špičky získané v příslušných landmarcích spektrogramu

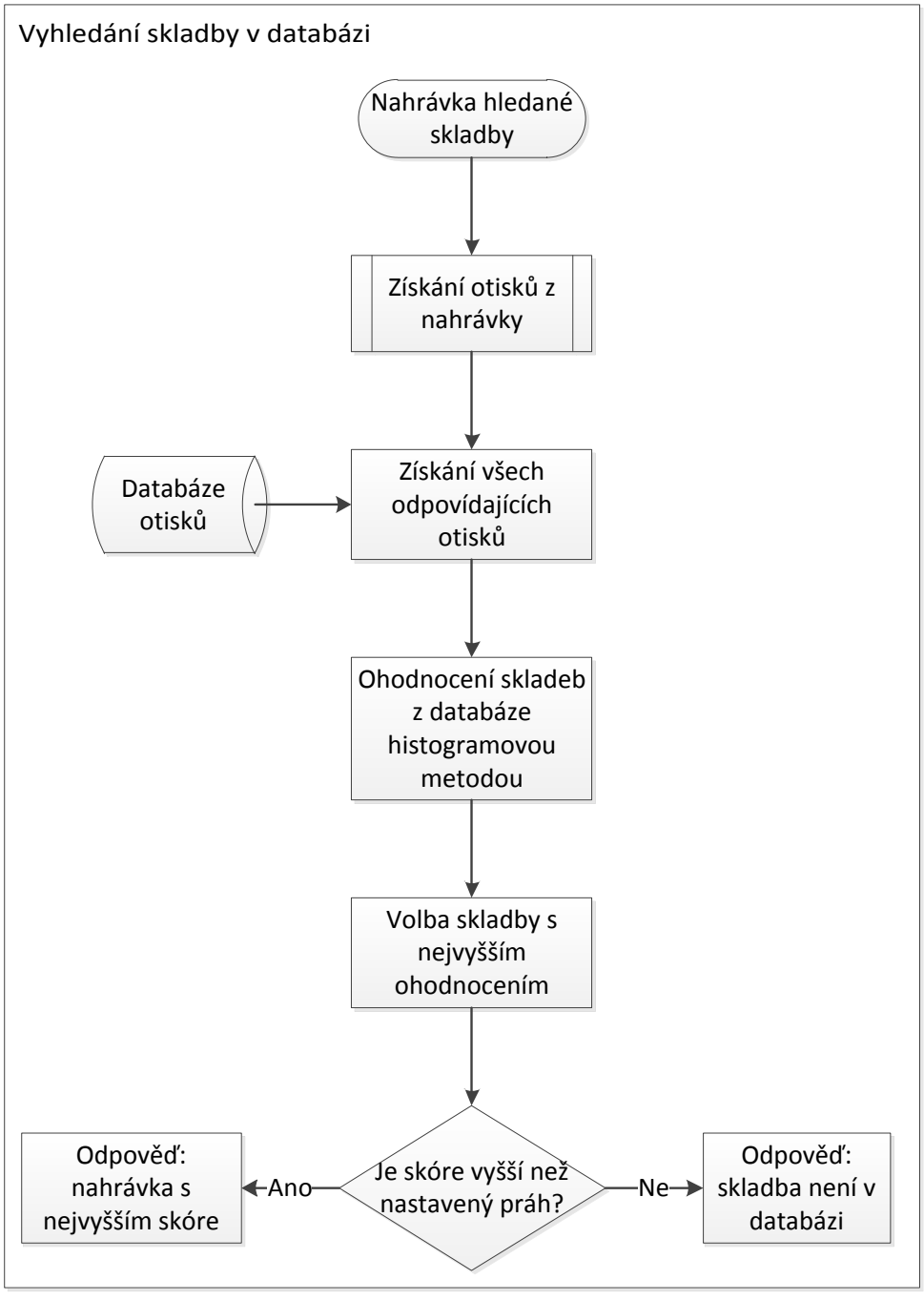


(b) Princip výběru špiček tvořících páry

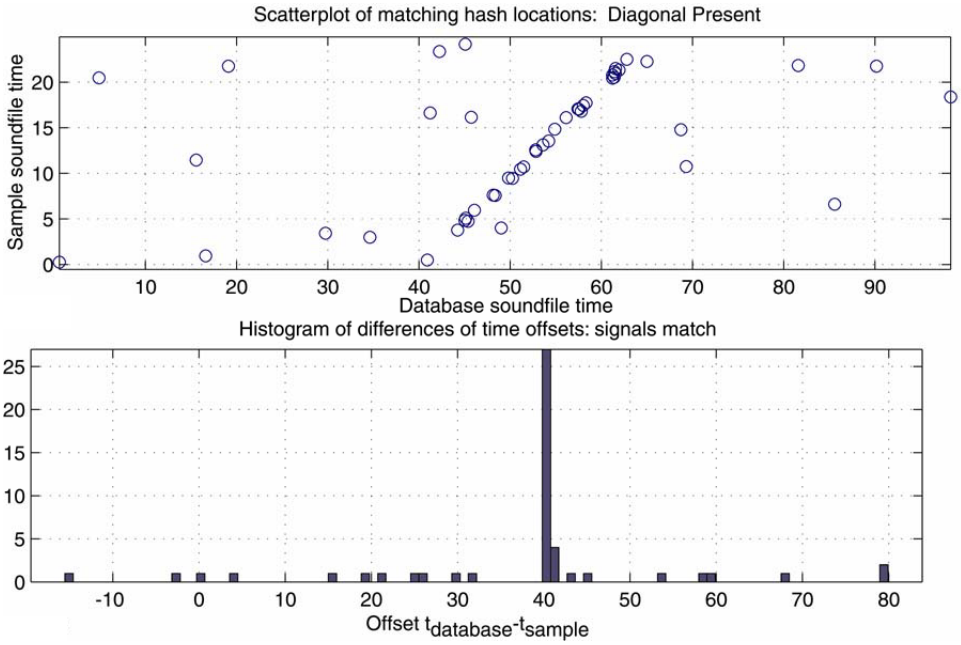
Obrázek 4.2: Postup služby Shazam na získání otisků ve spektru na základě nalezených landmarků



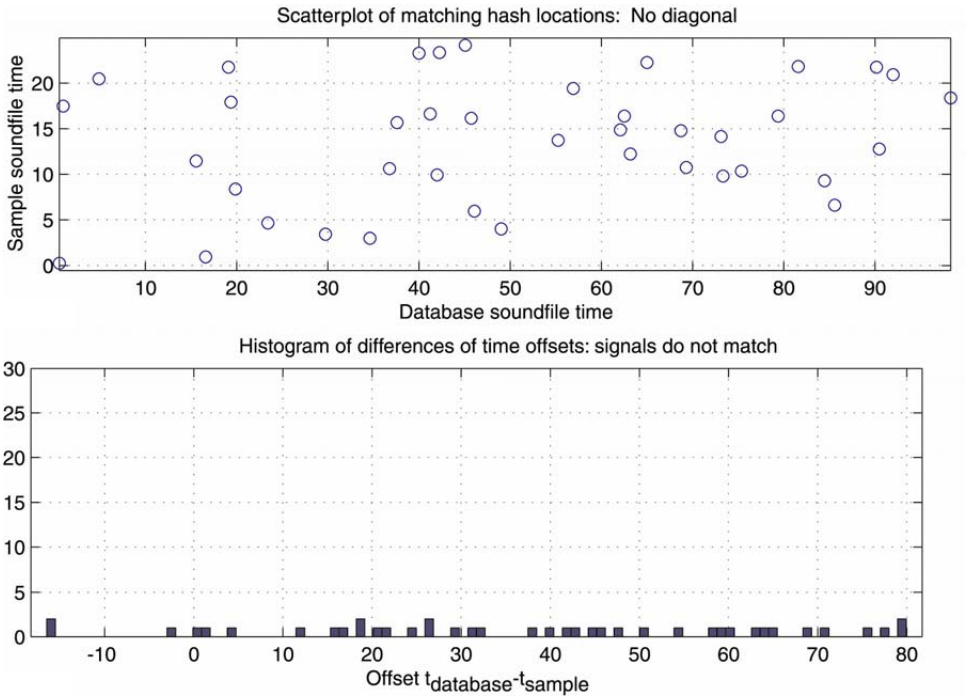
Obrázek 4.3: Vývojový diagram postupu získání otisku nahrávky službou Shazam



Obrázek 4.4: Vývojový diagram postupu vyhledávání skladby na základě nahrávky službou Shazam



(a) Ukázka korelačního diagramu a histogramu v případě, že se jedná o shodu



(b) Ukázka korelačního diagramu a histogramu v případě, že ke shodě nedošlo

Obrázek 4.5: Znázornění principu hodnotící funkce při porovnávání vzorku a databázové nahrávky službou Shazam. Obrázky jsou převzaty z [16].

4.2 Vyhledávání cover verzí pomocí Shazamu

Pro vyhledávání cover verzí je třeba vzít v úvahu to, že nová interpretace není tvořena jako věrné podání originální nahrávky, ale variací na ní. Obvykle tedy dochází ke změně tempa, doprovodných nástrojů, ale často i k posunu hlavní melodické linky do jiné tóniny. Tyto změny ale nejsou pravidlem, proto lze u využití algoritmu Shazamu uvažovat. Vzhledem k možnosti, že cover verze písničky i při stejné melodii a tempu může mít jinou strukturu skladby (jiné pořadí jednotlivých částí, jiné prodlevy mezi nimi apod.), není vhodné využívat jako hodnotící funkci maximum v histogramu rozdílu časových výskytů otisků. Místo toho lze využít například pouze samotný počet shodných otisků. Vstupem je, na rozdíl od hledání dle vzoru, vždy celá nahrávka, což zvyšuje množství použitelných hashů, ovšem navyšuje výpočetní náročnost. Vzhledem k výše uvedeným problémům (zejména jiná frekvenční charakteristika skladby) lze předpokládat, že tato metoda nebude příliš úspěšná.

4.3 Vyhledávání na základě uživatelovy interpretace

Při vyhledávání na základě uživatelovy interpretace (tedy zpěvu, pískání atp.) naráží algoritmus používaný službou Shazam hned na několik úskalí:

1. Neshoduje se spektrum jednotlivých úseků skladeb – uživatel nenapodobuje zvuk jednotlivých nástrojů, interpretuje pouze melodii
2. Neshoduje se časová charakteristika – uživatel není schopen přesně napodobit rychlost původní nahrávky, rychlost jeho interpretace obvykle kolísá

První problém je zdánlivě vyřešen tím, že se neanalyzuje celé spektrum, ale pouze jeho špičky, které by v melodii podané uživatelem měly být obdobné. Ovšem vzhledem k tomu, že mnohé ze špiček nejsou pouze v hlavní melodické lince, ale i v doprovodných nástrojích a efektech, lze očekávat minimální shodu. Uživatel navíc může skladby zpívat v jiné tónině, případně falešně – charakteristika skladby na základě spektra je tedy nedostačující.

Problém s časovou složkou by byl řešitelný, pokud by nebyla obsažena v jednotlivých otiscích. Při získávání skóre nahrávky by se nepoužila metoda histogramu, ale bylo by možné využít například algoritmus Dynamic Time Warping. Jelikož je ale v jednotlivých otiscích informace o časovém rozestupu dvou špiček, nedojde k nalezení shodných otisků.

Pro tuto problematiku byl tedy zvolen jiný postup – detekce základního tónu nahrávek (jak uživatelovy, tak originálních nahrávek v databázi) a jejich porovnávání. Je třeba zvolit vhodnou reprezentaci získané melodie, ve které bude možné zohledňovat jak kolísání rytmu, tak nepřesnosti v melodii. Při ohodnocování je třeba brát v úvahu nejen nepřesnosti uživatelské interpretace, ale i nepřesnosti samotného stanovení základní frekvence F_0.

Postup vyhledávání Prvním krokem je stanovení průběhu základního tónu nahrávky v čase. K tomuto účelu byl použit toolkit *CMU Sphinx*[5], který využívá algoritmus **Yin**. Ten využívá metodu autokorelace obohacenou o několik zpřesňujících modifikací [6]. Hledání F_0 je prováděno nad nahrávkami převedenými na vzorkovací frekvenci 8000 Hz pomocí okna o velikosti 200 vzorků a krokem 40 vzorků. Výstupem této extrakce je tedy odhadovaná frekvence základního tónu s časovým rozlišením 5 ms.

Získaná sekvence odhadů F_0 je dále filtrována, nejprve mediánovým filtrem s rozsahem 0,3 sekundy pro odstranění „šumu“ a poté histogramovým filtrem se stejným rozsahem

[5] http://cmusphinx.sourceforge.net/

pro výběr pouze lokálně nejdelších souvislých sekvencí (histogramový filtr je realizován algoritmem 4). Pro vyhlazení výsledků průběhu histogramového filtru v lokalitách, kde byly časté změny odhadu F_0, je opět aplikován stejný mediánový filtr. Ve výsledném odhadu jsou hledány změny F_0 v čase, přičemž za změnu je považován pouze případ, kdy se jednotlivé frekvence liší alespoň o půltón. Výpočet tohoto rozdílu je následující [12]:

$$\Delta f = 12 \times \log_2 f_1 - 12 \times \log_2 f_2 \tag{4.2}$$

Výsledkem je rozdíl Δf tónů f_1 a f_2, kde jednotlivé tóny jsou v Hertzích a výsledek v půltónech. Okamžiky, ve kterých je odhad F_0 menší než 60 Hz jsou považovány za chybu detekce a jsou přeskakovány. Produktem tohoto algoritmu je tedy sekvence čísel reprezentujících změny v melodii.

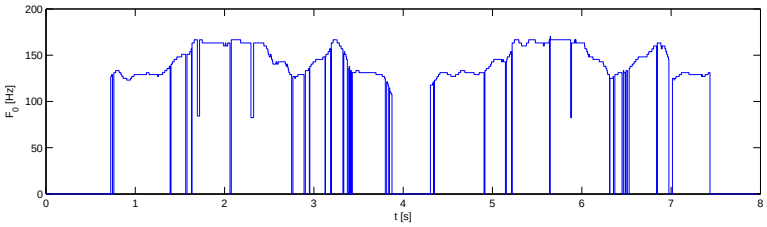
Postup jednotlivých kroků na ukázkových datech je znázorněn na obrázku 4.6.

Algoritmus 4 Algoritmus histogramového filtru délky `len` (vstupem je jednorozměrné pole `in`, výstupem jednorozměrné pole filtrovaných hodnot `out`)

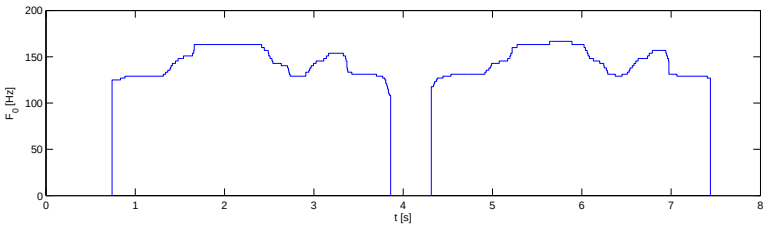
```
mid ← floor(len/2)
for i = 1 → size(in) do
    tmp ← newHashMap()
    for j = 1 → len do
        s ← i − mid + j
        if s ≤ 1 then
            val ← in[1]
        else if s > size(in) then
            val ← in.last()
        else
            val ← in[s]
        end if
        val ← floor(val×100)          ▷ Zachováme pouze dvě desetinná místa
        tmp[val]+ = 1
    end for
    tmp ← sortByValue(tmp)
    out[i] ← tmp.first().key()
end for
```

Vyhledávání pak probíhá nad databází tvořenou sekvencemi získanými z jednotlivých uložených originálních nahrávek. Sekvence získaná ze vstupního dotazu je porovnávána s každou ze sekvencí uložených v databázi. Jako výsledek porovnávání je zvolena nahrávka, jejíž ohodnocení je nejvyšší (je tedy nejpodobnější sekvenci hledané).

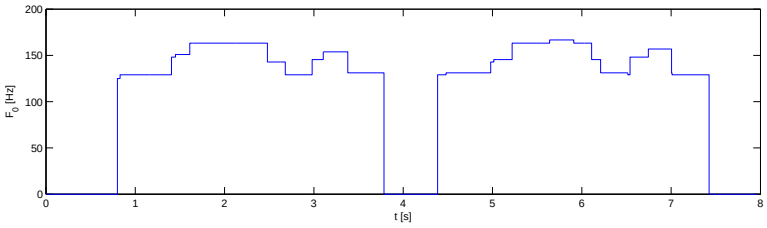
Ohodnocení sekvencí Ohodnocení sekvencí dat je úloha paralelní k úlohám typu porovnávání sekvencí DNA. Touto problematikou se podrobně zabývá například [15], odkud čerpá tato část práce. Algoritmy jsou většinou přizpůsobeny práci s řetězci, ale s minimálními úpravami je lze použít i na sekvence čísel. Jednou z metod je ohodnocení vstupního řetězce součtem cen operací, pomocí kterých jej lze transformovat na porovnávaný. Jednotlivé hodnotící funkce se pak liší jednak možnými operacemi, ale také jejich cenami. Mezi běžné operace patří následující (*v uvedené notaci A a B značí libovolné řetězce, x a y značí libovolné znaky.*):



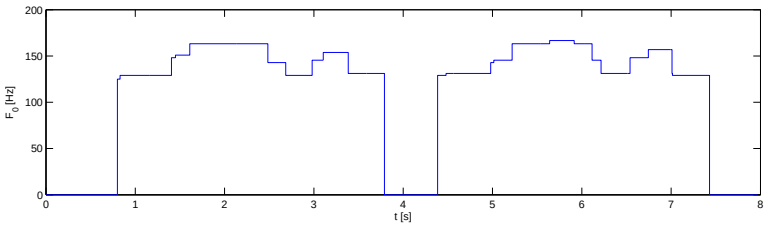
(a) Získaný odhad F_0 algoritmem Yin



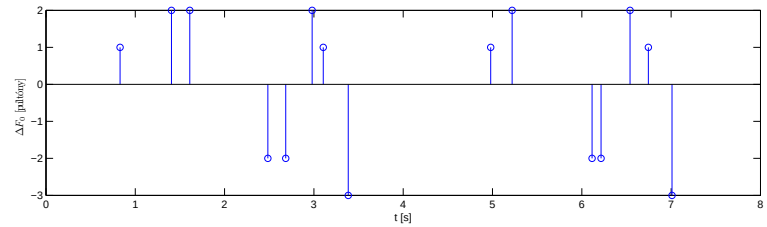
(b) Zarovnání průběhu F_0 pomocí mediánového filtru s délkou 0.3 s



(c) Další zarovnání F_0 - histogramovým filtrem s délkou 0.3 s



(d) Poslední zarovnání F_0 - opět mediánový filtr s délkou 0.3 s



(e) Detekce změn F_0 při zaokrouhlování na půltóny a nalezené hodnoty v čase

Obrázek 4.6: Postup získání informací o změnách melodie z dat získaných z průběhu odhadu frekvence základního tónu F_0.

- **Edit** (substitute) – úprava znaku – transformace řetězce AxB na AyB
- **Delete** – odstranění znaku – transformace řetězce AxB na AB
- **Insert** – vložení znaku – transformace řetězce AB na AxB

Cena transformace řetězce *abcd* na *acc* by tak byl součet ceny operace edit a delete. Jednou z nejběžněji používaných funkcí je *Levenshtein distance* (Levenshteinova vzdálenost), kde je každá operace ohodnocena cenou jedna.

Vytvoření tohoto ohodnocení je úloha vhodná pro dynamické programování. Na základě hodnotící funkce je průběžně sestavována matice C, kde $C_{i,j}$ odpovídá nejnižší možné ceně transformací pro převod prvních i znaků vstupního řetězce na prvních j znaků hodnoceného. Pro Levenshteinovu vzdálenost je sestavení matice uvedeno v algoritmu 5.

Algoritmus 5 Sestavení ohodnocující matice C Levenshteinovy vzdálenosti pro řetězce s a t. Výsledné hodnocení je poslední prvek této matice.

```
C ← newZeroArray(length(s), length(t))   ▷ Vytvoření dvourozměrného pole vyplněného nulami
for i = 1 → length(s) do
    C_{i,0} ← i                          ▷ Cena odstranění prvních i znaků řetězce s je i
end for
for j = 1 → length(t) do
    C_{0,j} ← j                          ▷ Cena vložení prvních j znaků z t do s je j
end for
for i = 1 → length(s) do
    for j = 1 → length(t) do
        if s_i = t_j then
            C_{i,j} ← C_{i-1,j-1}        ▷ Shoda – nulová cena
        else
            C_{i,j} ← 1+ min(
                C_{i-1,j},               ▷ Delete
                C_{i,j-1},               ▷ Insert
                C_{i-1,j-1}              ▷ Substitute
            )
        end if
    end for
end for
```

Při porovnávání sekvence S získané z uživatelovy melodie a sekvence D uložené v databázi je třeba zohlednit, že uživatel zpívá obvykle pouze část. Možnost vynechat libovolně dlouhou subsekvenci na začátku a na konci D lze zohlednit ohodnocením operace „insert“ cenou nula, pokud se jedná o přechod na začátku nebo konci D. Další vhodnou úpravou je ohodnocení operace „substitute“ na základě rozdílu modifikované hodnoty v řetězci S a D. Vzhledem k různé ceně operace substituce je vhodné zvýšit i cenu ostatních operací, neboť se uživatel při zpěvu mnohdy odchýlí i o několik půltónů. Výsledkem těchto úprav je algoritmus 6.

Při vyhledávání je jako skóre použita negace ceny získané výše uvedeným algoritmem. Příklad vytvořené ohodnocující matice **C** je v tabulce 4.1.

Algoritmus 6 Sestavení ohodnocující matice C modifikací Levenshteinovy vzdálenosti.

$C \leftarrow \text{newZeroArray}(\text{length}(s), \text{length}(t))$ ▷ Vytvoření dvourozměrného pole vyplněného nulami

for $i = 1 \rightarrow \text{length}(s)$ **do**
  $C_{i,0} \leftarrow i$ ▷ Cena odstranění prvních i znaků řetězce s je i
end for
for $j = 1 \rightarrow \text{length}(t)$ **do**
  $C_{0,j} \leftarrow 0$ ▷ **Modifikace**: Cena vložení prvních j znaků z t do s je 0
end for
for $i = 1 \rightarrow \text{length}(s)$ **do**
  for $j = 1 \rightarrow \text{length}(t)$ **do**
    $C_{i,j} \leftarrow \min($
      $1 + C_{i-1,j},$ ▷ **Modifikace**: Cena operace delete zvýšena na 2
      $1 + C_{i,j-1},$ ▷ **Modifikace**: Cena operace insert zvýšena na 2
      $C_{i-1,j-1} + abs(s_i - t_j)$ ▷ **Modifikace**: Cena operace substituce je absolutní hodnota rozdílu hodnot s_i a t_j
    $)$
    if $i = \text{length}(s)$ **then**
      $C_{i,j} \leftarrow \min(C_{i,j}, C_{i,j-1})$ ▷ **Modifikace**: Jsme-li na konci řetězce s, umožníme operaci insert s nulovou cenou
    end if
  end for
end for

	2	-2	5	-1	-4	2	-2	7	-2	-5	12	-3	-4	-1	-2	8	-1	-4	2	-2	
-3	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	2	**1**	2	2	1	2	1	2	1	2	2	0	1	2	1	2	2	1	2	1
-1	8	4	3	**1**	3	3	4	3	3	3	4	4	2	3	4	3	4	4	3	4	3
-6	12	6	5	3	**1**	3	5	5	5	4	6	6	4	5	3	5	6	4	5	6	5
3	16	8	7	5	3	**3**	5	7	7	6	5	7	6	6	5	7	8	6	6	8	7
-2	20	10	9	7	5	5	**4**	6	8	8	7	9	8	8	7	9	10	8	8	7	9
7	24	12	10	9	7	7	6	**4**	6	8	9	11	10	10	9	7	9	10	10	9	7
-2	28	14	12	11	9	9	8	6	**4**	6	8	10	12	12	11	9	8	10	12	11	9
	32	16	14	13	11	11	10	8	6	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Tabulka 4.1: Ukázka ohodnocující matice **C** získané při výpočtu modifikované Levenshteinovy vzdálenosti. Úvodní sloupec obsahuje hodnoty získané z nahrávky od uživatele, úvodní řádek je získán z melodie uložené v databázi. Tučně jsou vyznačeny hodnoty, které byly voleny pro získání výsledné ceny.

Kapitola 5

Volba testovacích dat a hodnocení výsledků

Součástí práce je implementace nejen některého z existujících algoritmů, ale i experimentování s modifikacemi a hodnocení jejich přínosu. Pro testování různých scénářů jsou potřeba data splňující různé parametry:

1. Vyhledávání dle vzoru – Kolekce celých písniček a jejich úryvků pořízených mikrofonem
2. Vyhledávání cover verzí – Databáze obsahující různé podání stejných písniček (tedy od různých interpretů)
3. Vyhledávání dle zpívaných vzorků – Soubor písniček a k nim odpovídajícím nápěvům od uživatelů

5.1 Testovací data

5.1.1 Vyhledávání dle vzoru

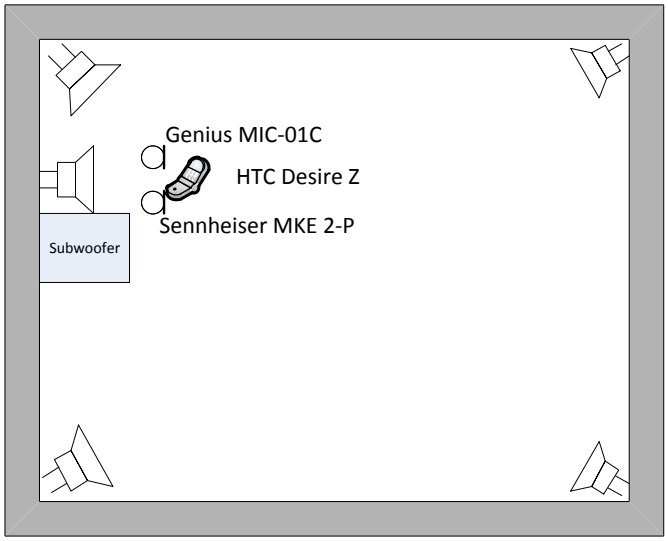
Pro vyhledávání dle vzoru jsem zvolil dataset Magnatune[1] [11]. Tento dataset obsahuje úryvky skladeb nejrůznějších žánrů. Formát dat uveden v tabulce 5.1.

Formát dat	MP3 (mono, 32 kbps)
Zařazení	Různé žánry
Počet nahrávek	25 863
Délka jedné nahrávky	30 s
Vzorkovací frekvence	16 kHz

Tabulka 5.1: Parametry datasetu Magnatune

Pro samotné dotazy pak byly z části tohoto datasetu pořízeny nahrávky pomocí mikrofonu. K reprodukci posloužily reproduktory Genius SW-HF5.1 5050 V2, nahrávky pak byly pořizovány souběžně třemi mikrofony, viz tabulka 5.2. Mikrofony byly umístěny blízko sebe a nasměrované na centrální reproduktor, viz schéma 5.1. Formát získaných dat je uveden v tabulce 5.3.

[1] http://tagatune.org/Magnatagatune.html



Obrázek 5.1: Schéma rozmístění mikrofonů a reproduktorů při pořizování testovacích nahrávek

Mikrofon	Popis	Účel
Sennheiser MKE 2-P	Kvalitní mikrofon	Věrné zachycení reprodukované hudby
Genius MIC-01C	Levný mikrofon k PC	Zachycení v kvalitě odpovídající běžnému vybavení PC či notebooku
HTC Desire Z	Mobilní telefon	Kvalita odpovídající vstupu do aplikace na chytrém mobilním telefonu

Tabulka 5.2: Použité mikrofony pro záznam reprodukované hudby

Pro simulaci krátkého vstupu od uživatele používá vyhledávací script pouze 10 vteřin z každé nahrávky (konkrétně byl zvolen časový úsek od páté do patnácté vteřiny).

5.1.2 Vyhledávání cover verzí

Pro úlohu hledání cover verzí je nezbytné, aby dataset obsahoval anotované cover verze některých písniček. Pro tuto část byl zvolen dataset Covers80[2] [7], neboť obsahuje množství písniček a k nim příslušných cover verzí. Podrobnosti o těchto datech jsou v tabulce 5.4.

5.1.3 Vyhledávání dle uživatelské interpretace

Na poslední část, tedy vyhledávání na základě uživatelovy vlastní interpretace (zpěv, brumendo atp.) byl zvolen dataset *MIR-QBSH: A Corpus for Designing QBSH (Query by Sin-*

[2] http://labrosa.ee.columbia.edu/projects/coversongs/covers80/

Formát dat	RAW (mono, PCM 16, little endian)
Zařazení	Populární hudba (západní)
Počet nahrávek	907 (×3)
Délka jedné nahrávky	30 s
Vzorkovací frekvence	44 100 Hz

Tabulka 5.3: Parametry nahrávek z mikrofonů

Formát dat	MP3 (mono, 32 kbps)
Zařazení	Různé žánry
Počet nahrávek	80 párů originál–cover
Délka jedné nahrávky	Obvykle 3–5 minut
Vzorkovací frekvence	16 kHz

Tabulka 5.4: Parametry datasetu Covers80

ging/Humming) Systems[3] [9] (dále jen *MIR-QBSH*), díky jeho zaměření přímo na danou problematiku. Jeho přesnější parametry jsou v tabulce 5.5. Obsahuje množství uživatelsky nazpívaných skladeb dle předloh, které jsou ve formátu MIDI. Díky tomu byly algoritmy testovány se zjednodušeným scénářem, ve kterém odpadají chyby způsobené extrakcí melodie z originálních skladeb.

Pro testování algoritmu Shazam na této úloze byly jednotlivé MIDI soubory převedeny do formátu WAV nástrojem TiMidity++[4].

	Originální skladby	Uživatelská podání
Formát dat	MIDI	WAV (mono, PCM 8)
Zařazení	Populární hudba (západní)	
Počet nahrávek	48	4431
Délka jedné nahrávky	∓20 s	∓10 s
Vzorkovací frekvence	–	8 kHz

Tabulka 5.5: Parametry datasetu MIR-QBSH

5.2 Použité metriky

V testovaných scénářích se jedná o úlohy identifikace a rozhodování. V databázi je třeba nalézt odpovídající nahrávku a na základ skóre rozhodnout, jestli systém bude tuto odpověď považovat za správnou a vrátí ji jako výsledek hledání. V takovýchto systémech lze tedy nastavit hodnotu prahu v závislosti na tom, jakých hodnot jednotlivých metrik chceme dosáhnout, pokud nám to kvalita systému umožňuje.

V úloze identifikace je klíčové, zda byla hledaná hodnota nalezena (označována jako *Retrieved*) či nikoli (*Not Retrieved*) a zda byla nalezená hodnota relevantní (*Relevant*/*Not Relevant*), tedy odpovídala hledanému dotazu. Poměry těchto kritérií vyjadřují hodnoty *Precision* a *Recall*[4].

[3] http://www.music-ir.org/mirex/wiki/2009:Query_by_Singing/Humming
[4] http://timidity.sourceforge.net/

Precision Počet nalezených relevantních výsledků v poměru k celkovému počtu nalezených výsledků

Recall Počet nalezených relevantních výsledků v poměru k celkovému počtu relevantních, tedy i nenalezených, výsledků

Většina zde používaných metod získává pouze jeden nalezený záznam s nejvyšším skóre, na základě hodnoty prahu je pak tento jeden záznam považován za výsledek hledání, nebo je výsledkem hledání prázdná množina. Hovoříme o přijetí (*accept*) a odmítnutí (*reject*) výsledku. Rozhodnutí o této skutečnosti může být buď správné (*true*), nebo špatné (*false*). Možné výsledky rozhodnutí jsou následující [13]:

True Accept/True Positive Přijetí výsledku A pro hledaný prvek A

True Reject/True Negative Odmítnutí výsledku B pro hledaný prvek A

False Accept/False Positive Přijetí výsledku B pro hledaný prvek A

False Reject/False Negative Odmítnutí výsledku A pro hledaný prvek A

Na základě těchto hodnot lze rozpoznávač charakterizovat mírou chybného přijetí (FAR) a mírou chybného odmítnutí (FRR) [13]

FAR False Accept Rate – Poměr počtu chybných přijetí k celkovému počtu výsledků, které měly být hodnoceny jako odmítnutí – $FAR = \frac{FP}{FP+TN}$

FRR False Reject Rate – Poměr počtu chybných odmítnutí k celkovému počtu výsledků, které měly být hodnoceny jako přijetí – $FRR = \frac{FN}{TP+FN}$

Hodnoty těchto měr v závislosti na nastavení prahu lze vyjádřit pomocí ROC[5] křivky, případně DET[6] křivky. Pomocí nich je možné porovnávat kvalitu jednotlivých algoritmů. DET křivka pak oproti ROC křivce přináší lepší rozlišení v oblasti zájmu díky nelineárnímu zobrazení os X a Y. Celkově lze systémy charakterizovat hodnotou EER[7], která vyjadřuje nejlepší dosažitelnou chybovost obou hodnot, tedy kdy je FAR rovno FRR. V praxi je možné systém přizpůsobit například nižší míře FRR, což odpovídá posunu po DET křivce.

Při testování hledání cover verzí algoritmem Shazam bylo hodnoceno kritérium Top10, tedy kolik procent dotazů obsahovalo správně identifikovanou cover verzi mezi prvními deseti nejlépe hodnocenými výsledky. Toto kritérium je vhodné pro systémy, které nejsou samy o sobě schopny dodat přesný výsledek, ale dokáží posloužit alespoň pro rychlejší získání pravděpodobných výsledků, ze kterých je ten správný zvolen pomocí nějaké další, případně i výpočetně náročnější metody.

5.3 Testování jednotlivých algoritmů

Pro testování byly využívány automatizované skripty, jejichž výsledkem byly skóre jednotlivých nejlepších výsledků hledání spolu s informací, zda je výsledek správný. Na tyto hodnoty byl využit *BioSecure Tool*[8][14], jehož výstupem jsou právě ROC a DET křivky a hodnota EER.

[5] ROC – Receiver operating characteristic

[6] DET – Detection Error Tradeoff

[7] EER – Equal Error Rate – míra stejné chybovosti

[8] Sada scriptů do Matlabu – http://svnext.it-sudparis.eu/svnview2-eph/ref_syst/Tools/PerformanceEvaluation/

5.3.1 Shazam

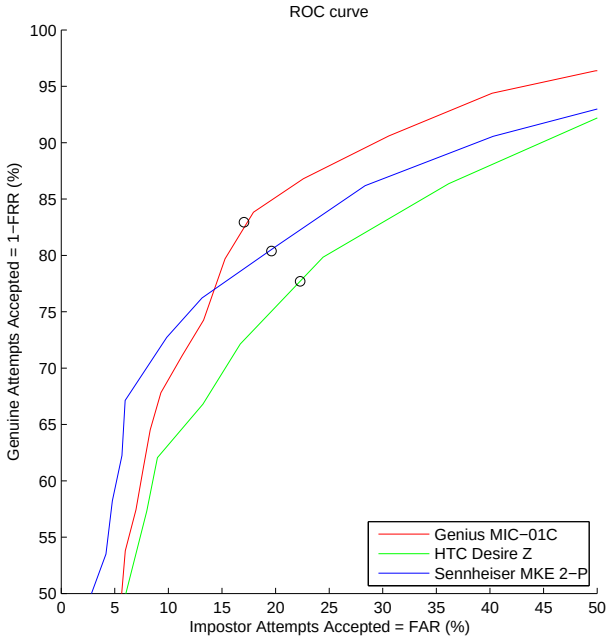
Vyhledávání dle vzoru Implementace služby Shazam dosáhla při testování vyhledávání vzoru poměrně nízké hodnoty EER (kolem 20%) při vstupech ze všech tří testovaných mikrofonů. Výsledky pro jednotlivé mikrofony zachycují grafy v obrázku 5.2. Na obrázku 5.3 je pak vidět průběh P/R křivky pro data netřízená dle mikrofonů. Je zřejmé, že lze dosáhnout hodnoty Recall maximálně okolo 70%, což je způsobeno tím, že dochází vždy k vyhledání maximálně jednoho výsledku, neboť jde o úlohu identifikace. Vzhledem k velikosti databáze (2419 písniček), naměřeným hodnotám a cílovému zaměření aplikace lze tento algoritmus hodnotit jako poměrně spolehlivý. Navíc dosahuje relativně vysoké rychlosti – čas na získání odpovědi na dotaz je okolo jedné vteřiny[9]. Tato hodnota byla zjištěna při použití databázového indexu typu binární vyhledávací strom, který má časovou složitost vyhledávání logaritmickou. Při použití hashtabulky by bylo možné dosáhnout vyhledávání s konstantní časovou složitostí, což by ovšem vedlo k mnohem větším paměťovým nárokům. Nezanedbatelnou časovou složkou ohodnocení písničky je také sestavení histogramů vzdáleností konkrétních otisků v rámci jednotlivých písniček.

Algoritmus si tedy poměrně dobře poradí se šumem vzniklým při nahrávání skladeb, stejně jako se změnou spektrální charakteristiky způsobené přehráváním pomocí reproduktorů a případným použitím ekvalizéru. Na obrázku 5.4 je ilustrováno, jaký vliv na detekci špiček měly jednotlivé použité mikrofony při záznamu z reproduktorů. Z jednotlivých grafů je vidět, že se získané špičky liší, ovšem je mezi nimi možno nalézt špičky shodné. Toho metoda Shazam využívá – detekuje velké množství špiček, aby zvýšila pravděpodobnost shody. Tím získané riziko nalezení týchž špiček v jiných skladbách je řešeno použitím právě dvojic těchto špiček. Z obrázků 5.4a a 5.4e je zřejmé, že získané špičky ze dvou různých skladeb mohou být značně odlišné (a tak tomu i ve většině případů je). Díky tomu je možné dosáhnout nízké míry chyb typu False Positive.

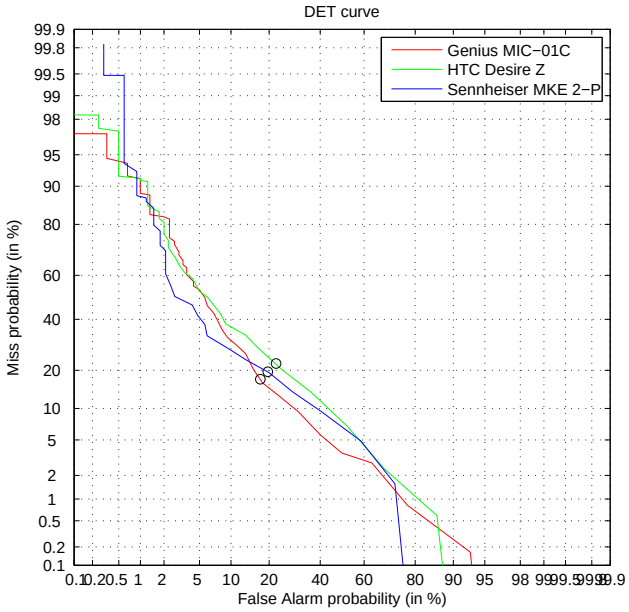
Vyhledávání cover verzí Algoritmus upravený na vyhledávání cover verzí dle očekávání nepřinesl pozitivní výsledky – ze 164 dotazů na cover verze v databázi obsažené byly pouze čtyři výsledky hledání správné, což odpovídá pouhým přibližně dvěma procentům. Na základě těchto dat je zbytečné sestavovat ROC křivku, případně hledat vhodný práh pro dosažení EER, neboť je zřejmé, že je systém této kvality v praxi nepoužitelný. Tím se potvrdil odhad ze sekce 4.2. Při testování na umístění v Top10 dosáhla tato metoda pouhých 14% úspěšnosti (24 ze 164 dotazů obsahovalo správnou nahrávku mezi deseti nejlépe hodnocenými výsledky), což jen potvrzuje, že tato metoda je pro hledání cover verzí nevhodná.

Vyhledávání na základě uživatelské interpretace Obdobně byl algoritmus služby Shazam testován i na úloze vyhledávání na základě uživatelského podání. Ohodnocování nalezených výsledků bylo stejné, jako při metodě hledání cover verzí. Z 829 dotazů tvořených uživatelskými nahrávkami bylo pouze 18 identifikováno správně, tedy přibližně 2%. Při počtu 48 nahrávek v databázi lze považovat výsledky algoritmu Shazam za zcela náhodné, z čehož plyne, že pro danou úlohu není vhodný.

[9] Testováno na virtuálním stroji s přiřazeným jedním jádrem procesoru taktovaného na 4 GHz.

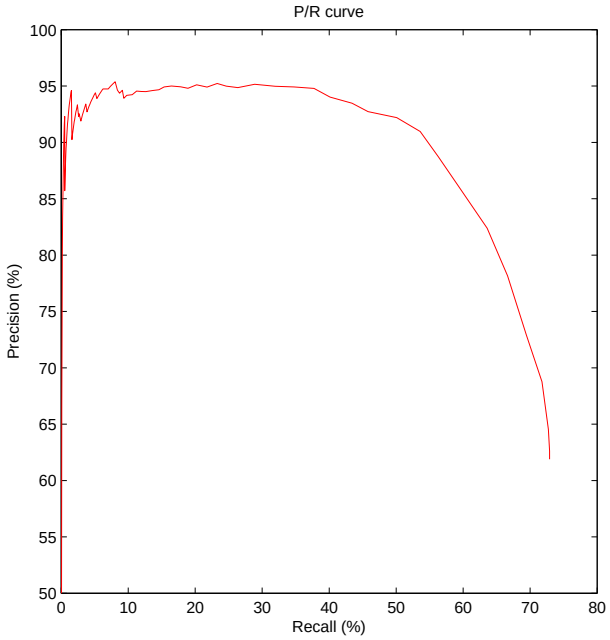


(a) ROC křivka



(b) DET křivka

Obrázek 5.2: Výsledky testování algoritmu služby Shazam v úloze vyhledávání dle vzoru s nahrávkami z různých mikrofonů.



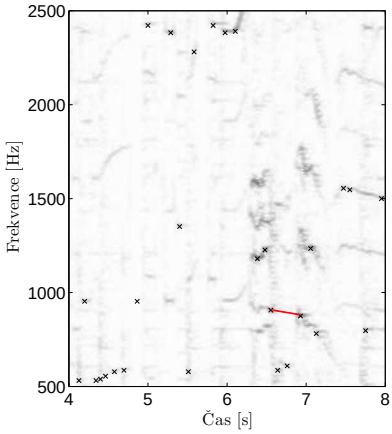
Obrázek 5.3: Výsledná Precision/Recall křivka zjištěná testováním algoritmu Shazam v úloze vyhledávání hledání dle vzoru s nahrávkami z různých mikrofonů.

5.3.2 Porovnávání melodií

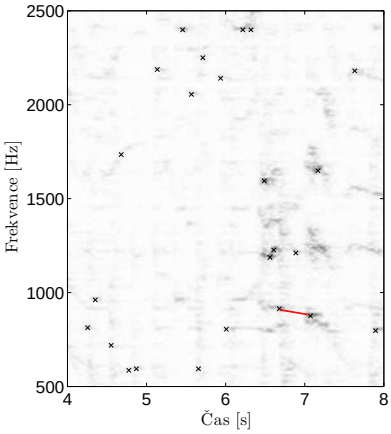
Vyhledávání na základě uživatelské interpretace Metoda porovnávání změn v melodii byla použita zejména kvůli hledání na základě zpěvu, proto byla testována nejprve na této úloze. Testování probíhalo nad datasetem MIR-QBSH, který obsahuje originální melodie v MIDI formátu. To umožnilo ověřit funkčnost na ideálním scénáři, ve kterém by byla extrakce melodie bezchybná. Při testu na 48 originálních nahrávkách a 827 uživatelských podáních bylo dosaženo EER přibližně 40%. Na obrázku 5.5 je zachycena výsledná DET křivka tohoto měření spolu s grafem průběhu hodnot FAR a FRR v závislosti na nastavení prahu. Zjištěné hodnoty precision a recall jsou zachyceny na obrázku 5.6.

Vzhledem k malé velikosti databáze není hodnota EER příliš uspokojující, na hodnotách precision a recall je zřejmé, že je míra identifikace správné nahrávky velmi nízká. Problémem je zejména nízká entropie jednotlivých sekvencí, neboť zachycují pouze změny ve výšce tónů melodie. Možným zlepšením by tak bylo zakomponování časové informace do výsledné sekvence čísel a vhodná úprava hodnotící funkce. Přitom je třeba zohlednit nepřesnosti v rytmu při uživatelské interpretaci. Jedním z možných přístupů je použití relativní informace o délce tónu vzhledem k délce předchozího tónu, jak je navrženo v [12]. K tomu je ovšem nezbytná detekce okamžiků počátku a konce tónu, o kterou by bylo nutné implementovaný systém také rozšířit.

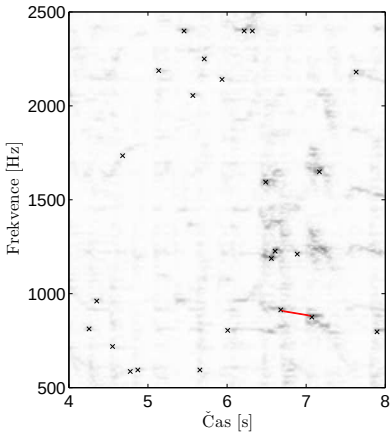
Hledání cover verzí Výše uvedená metoda byla testována i na úloze hledání cover verzí nad datasetem Covers80. Ze 164 dotazů na cover verze v databázi obsažené byly pouze tři výsledky hledání správné. Je tedy zřejmé, že výsledky hledání lze považovat za náhodné. Mezi příčiny neúspěchu lze řadit:



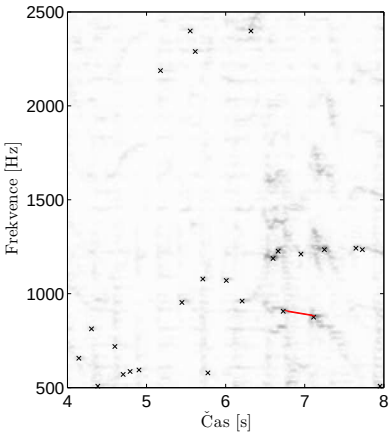
(a) Originální nahrávka (William Brooks - Karma Dogs)



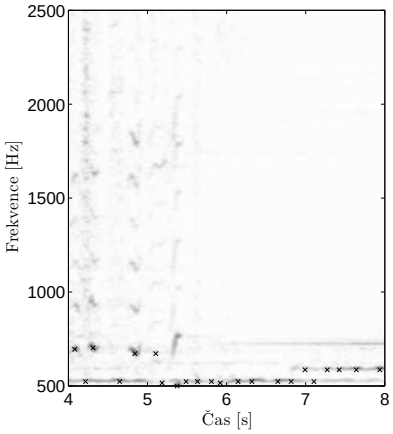
(b) Nahrávka pořízená mobilním telefonem



(c) Nahrávka pořízená kvalitním mikrofonem

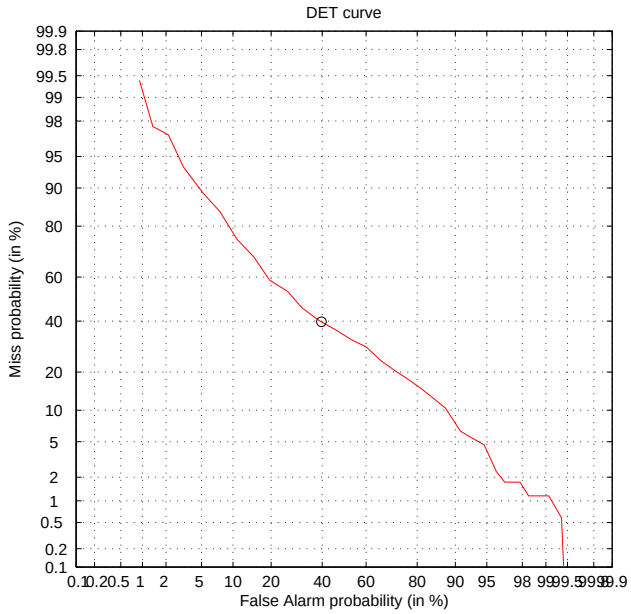


(d) Nahrávka pořízená levným mikrofonem

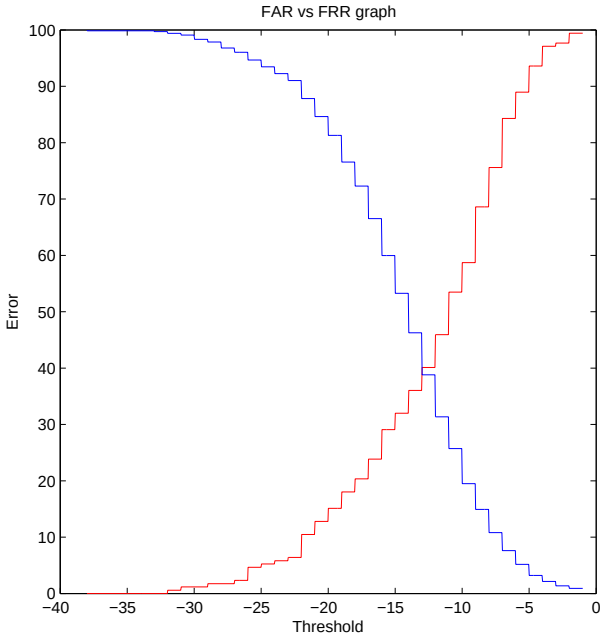


(e) Ukázka jiné skladby (Artemis - Undone)

Obrázek 5.4: Ukázka špiček detekovaných v různých záznamech téže nahrávky. Pro porovnání jsou ukázány i špičky získané z jiné skladby. Červeně je naznačen jeden z možných otisků, který je detekován ve všech nahrávkách zkoumané skladby.

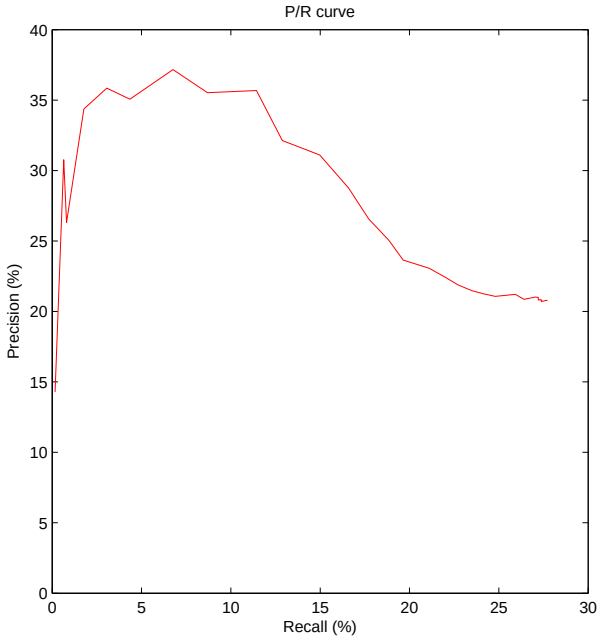


(a) DET křivka



(b) Hodnoty FAR (modrá) a FRR (červená) v závislosti na nastavení hodnoty prahu

Obrázek 5.5: Výsledky testování metody porovnávání změn v melodii na datasetu MIR-QBSH.



Obrázek 5.6: Výsledná Precision/Recall křivka z testování porovnávání melodií nad datasetem MIR-QBSH.

- Nepřesnost odhadu melodie – při extrakci frekvence základního tónu z polyfonní melodií dochází oproti monofonní k větším chybám, neboť je spektrum zkreslené dalšími nástroji
- Nerespektování polyfonního charakteru – použitá extrakce nezískává informace o dalších nástrojích, pouze o tom, který je v daný okamžik nejvýraznější
- Nerespektování hudební struktury – při porovnávání sekvencí nejsou nijak odlišeny jednotlivé části hudby, jako jsou refrén či sólo, takže hledaná cover verze musí mít stejnou melodii jako originální nahrávka po celou dobu trvání skladby
- Ignorování délek tónů – formát sekvence změn v melodii neobsahuje žádné časové informace o trvání tónů

Možným řešením je využití rozkladu celé polyfonní skladby na MIDI, což je však úloha zatím spolehlivě nevyřešená a svou složitostí překračující rozsah této práce. V hudbě by také bylo vhodné anotovat jednotlivé části, odpovídající slokám, refrénu apod. a porovnávání provádět jenom v rámci nich. Lze totiž předpokládat, že melodie refrénu cover verze bude stejná, ovšem například sóla se budou lišit. Výsledné ohodnocení sekvencí pak musí tuto skutečnost respektovat. Informace o délce tónů lze do sekvence zakomponovat obdobně, jako bylo uvedeno u vyhledávání na základě uživatelské interpretace.

Kapitola 6

Závěr

6.1 Souhrn

Práce slouží jako úvod do získávání informací z hudby se zaměřením na vyhledávání hudebních nahrávek. Pro hudebně méně vzdělané proto obsahuje i úvod do vlastností hudby, kterými se MIR zabývá. Jelikož cílem je výběr vhodného algoritmu pro vyhledávání hudebních nahrávek, je zde představeno několik algoritmů, které identifikaci nahrávek již komerčně nabízí. Vzhledem k zaměření práce na vyhledávání na základě krátkých vstupů, nahraných například mobilním telefonem, je detailně popsán algoritmus služby Shazam, která tuto funkci nabízí. Na základě testování jsou pak navrženy možné úpravy tohoto algoritmu pro možnost vyhledávání cover verzí a vyhledávání na základě uživatelského vstupu. Kvůli zjištěné nízké úspěšnosti algoritmu Shazam na těchto scénářích je probrán přístup k řešení těchto úloh pomocí extrakce melodie. Testování prokázalo, že je možné tuto metodu využít pro hledání na základě uživatelského podání písničky, byť tato metoda vyžaduje ještě další vývoj. Testování ukázalo, že vyhledávání cover verzí není dostatečně řešitelné žádným z těchto postupů. Jsou však diskutovány další možnosti úprav použitého postupu, které by mohly k řešení daného problému vést.

6.2 Výhled do budoucna

Na základě použitých algoritmů a zavedení diskutovaných vylepšení může být v budoucnu vytvořena například webová služba, která by zprostředkovávala vyhledávání hudebních nahrávek široké veřejnosti. Dostupné služby se zaměřují pouze na jeden z možných vstupů vyhledávání, bylo by však možné vytvořit jednu univerzální službu, která by automaticky rozpoznala, o jaký druh vstupu se jedná a nabídla širší rozsah výsledků, tedy nejen hledanou nahrávku, ale případně i možné cover verze a podobně.

Podobné služby by se mohly stát pevnou součástí mobilních telefonů, s tím že by telefon neustále nahrával zvuk do posuvného zásobníku a v případě uživatelova zájmu o identifikaci by mohl rovnou začít vyhledávat na základě paměti předchozího vstupu, jako to dělají například některé videokamery. To by uživatelům ušetřilo potíže v případě, že se rozhodnou identifikovat písničku na samém konci, tudíž již nemohou nahrát dostatečně dlouhý úsek. Navíc by bylo se tím zkrátila doba odezvy vyhledávání, neboť by uživatel nemusel čekat na pořízení nahrávky.

Literatura

[1] Midomi – About us. online.
URL http://www.midomi.com/index.php?action=main.about_us

[2] Gracenote Connected TV Technologies Ship with Leading Home Entertainment Products. Press release, Leden 6 2011.
URL
http://www.gracenote.com/Gracenote%20Connected%20TV%20Technologies%20Ship%20with%20Leading%20Home%20Entertainment%20Products

[3] Gracenote MusicID. online, 2011.
URL http://www.gracenote.com/products/musicid/

[4] Buckland, M.; Gey, F.: The relationship between recall and precision. *Journal of the American society for information science*, ročník 45, č. 1, 1994: s. 12–19.

[5] Casey, M.; Veltkamp, R.; Goto, M.; aj.: Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, ročník 96, č. 4, 03 2008: s. 668–696.
URL http://dx.doi.org/10.1109/JPROC.2008.916370

[6] De Cheveigné, A.; Kawahara, H.: YIN, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am*, ročník 111, č. 4, 2002: s. 1917–1930.

[7] Ellis, D. P. W.: The "covers80"cover song data set. 2007.
URL http://labrosa.ee.columbia.edu/projects/coversongs/covers80/

[8] Holm, F.; Hicken, W. T.: Audio fingerprinting system and method. 03 2006.
URL http://www.patentlens.net/patentlens/patent/US_7013301/en/

[9] Jang, J.: MIR-QBSH Corpus. *MIR Lab, CS Dept, Tsing Hua Univ, Taiwan. Available at the "MIR-QBSH Corpus"link at http://www.cs.nthu.edu.tw/˜jang.*

[10] Lalinský, L.: How does Chromaprint work? Leden 13 2011.
URL http://oxygene.sk/lukas/2011/01/how-does-chromaprint-work/

[11] Law, E.; Von Ahn, L.: Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009, s. 1197–1206.

[12] Liu, T.; Huang, X.; Yang, L.; aj.: Query by Humming: Comparing Voices to Voices. In *Management and Service Science, 2009. MASS'09. International Conference on*, IEEE, 2009, s. 1–4.

[13] Mansfield, A.; Wayman, J.: Best Practices in Testing and Reporting Performance of Biometric Devices. 2002.

[14] Mayoue, A.: Biosecure Tool: Performance Evaluation of A Biometric Verification System. 2007.
URL http://svnext.it-sudparis.eu/svnview2-eph/ref_syst/Tools/PerformanceEvaluation/

[15] Navarro, G.: A guided tour to approximate string matching. *ACM Comput. Surv.*, ročník 33, č. 1, Březen 2001: s. 31–88, ISSN 0360-0300, doi:10.1145/375360.375365.
URL http://doi.acm.org/10.1145/375360.375365

[16] Wang, A.: An Industrial-Strength Audio Search Algorithm. http://www.ismir.net: ISMIR, Baltimore, MD, USA, 10 2003, s. 7–13.
URL http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

[17] Wells, M.; Venkatachalam, V.; Cazzanti, L.; aj.: Automatic identification of sound recordings. 02 2008.
URL http://www.patentlens.net/patentlens/patent/US_7328153/en/

[18] Woodward, D.: Shazam names that tune. online, Prosinec 2009.
URL
http://www.director.co.uk/magazine/2009/11%20December/shazam_63_04.html

Příloha A

Obsah CD

```
/
├── text ..... Technická zpráva spolu se zdrojovými kódy v LaTeXu
├── implementace ..... Zdrojové kódy implementace této práce
│   └── datasets ..... Datasety použité v rámci práce
└── literatura ..... Použitá literatura, která byla dostupná v elektronické formě
```

Příloha B

Použití

Jelikož měla práce experimentální charakter, není její součástí celistvá aplikace, ale množství scriptů/programů, které slouží pouze jako dílčí krok některého problému.
Programy jsou spustitelné na operačním systému Linux (testováno na distribuci Debian Squeeze). Jednotlivé programy vyžaduje ke kompilaci či spuštění následující nástroje:

- **G+ 4.3 a vyšší**
  - Kompilátor jazyka C++
  - Použito pro `extractor`
  - Balíček v Debianu: `g++`
- **GCC 4.3 a vyšší**
  - Kompilátor jazyka C
  - Použito pro `pitchextractor` a `pitchextractor_covers`
  - Balíček v Debianu: `gcc`
- **Libsndfile**
  - Knihovna pro čtení zvukových souborů
  - Použito pro `extractor`, `pitchextractor` a `pitchextractor_covers`
  - http://www.mega-nerd.com/libsndfile/
  - Balíček v Debianu: `libsndfile1-dev`
- **Secret Rabbit Code** (Libsamplerate)
  - Knihovna pro převod signálu na jinou vzorkovací frekvenci
  - Použito pro `extractor`, `pitchextractor` a `pitchextractor_covers`
  - http://www.mega-nerd.com/SRC/
  - Balíček v Debianu: `libsamplerate0-dev`
- **MySQL 5.1 a vyšší**
  - Databázový systém
  - Využíváno scripty na bázi algoritmu služby Shazam

  - http://mysql.com
  - Balíček v Debianu: `mysql-server`

- **PHP 5.3 a vyšší**
  - Interpret skriptovacího jazyka PHP
  - Použito pro většinu implementovaných skriptů
  - http://php.net
  - Balíčky v Debianu: `php5`, `php5-cli` a `php5-mysql`

- **LAME**
  - Knihovna pro práci s mp3 soubory
  - Použito pro automatickou konverzi datasetů z formátu MP3 do formátu WAV (není třeba, pokud se pracuje pouze s WAV soubory)
  - http://lame.sourceforge.net/
  - Balíček v Debianu: `lame` (dostupný v repozitáři *Backports* či *Sid*)

- **TiMidity++**
  - Knihovna pro převod MIDI souborů do formátu WAV
  - Použito pro vyhledávání nad datasetem MIR-QBSH algoritmem služby Shazam (script `fill_db_qbh.php`)
  - http://timidity.sourceforge.net/
  - Balíček v Debianu: `timidity`

Pomocné nástroje pro pořízení nahrávek z mikrofonu pak ještě vyžadují ke kompilaci Qt SDK ve verzi alespoň 4.7 (http://qt.nokia.com/products/).

B.1 Implementace služby Shazam

B.1.1 Příprava

Implementace služby Shazam byla využita pro tři scénáře:

1. Vyhledávání dle vzoru
2. Hledání cover verzí
3. Vyhledávání na základě zpěvu

Pro všechny tři je třeba mít zkompilovaný nástroj `Extractor`, viz sekce B.1.3.
Ve všech případech používá databázi MySQL, kterou je třeba vytvořit. Přístupové údaje se nastavují v souboru `implementace/db_conn.ini`. Vytvoření uživatele a databáze s výchozími přístupovými údaji se provede následovně (řádky začínající znakem `$` se zadávají do terminálu, řádky začínající `mysql>` do konzole MySQL, vždy bez této úvodní sekvence):

```
$ mysql -u root -p
```

MySQL vyzve k zadání hesla uživatele root, které je třeba zadat.

```
mysql> create database dp;
mysql> grant usage on *.* to DP@localhost identified by 'dp';
mysql> grant all privileges on dp.* to DP@localhost ;
mysql> flush privileges;
mysql> exit;
```

Následně je třeba vytvořit příslušné tabulky:

```
$ mysql -u DP -pdp dp < implementace/db.sql
```

Dalším krokem je indexace písniček do databáze. Pro jednotlivé úlohy provádí indexaci následující scripty umístěné ve složce `implementace`:

- Vyhledávání dle vzoru – `fill_db.php` a `fill_db_all.php` (první zmiňovaný vloží pouze písničky, z nichž existují verze pořízené mikrofonem, druhý vkládá všechny nahrávky)
- Hledání cover verzí – `fill_db_covers.php`
- Vyhledávání na základě zpěvu – `fill_db_qbh.php`

Všechny scripty vyžadují **právo zápisu do složky s datasetem**, neboť převádějí mp3 soubory do formátu WAV, proto je vhodné celou složku implementace přemístit mimo CD. Scripty pro vyhledávání dle vzoru používají pouze první ze 16 složek datasetu Magnatune, zbylé jsou pro úsporu místa v přiloženém archivu ve složce `implementace/datasets/Magnatune`.
Je také nezbytné zkompilovat program `extractor` ve složce `implementace/extractor`, a to použitém příkazu `make`.

B.1.2 Spuštění

Příkazy pro pro vyhledávání se spouští vždy ve tvaru `vyhledávací_script.php cesta/k/nahrávce/z/mikrofonu.raw`. Nahrávky z mikrofonů jsou ve složce `implementace/datasets/custom`, je třeba je převést z formátu mp3 do formátu RAW v dané složce umístěným scriptem `convert.php`.
Jednotlivé vyhledávače jsou následující:

- Vyhledávání dle vzoru – `search.php` (používá ze zadané nahrávky pouze desetivteřinový úsek z důvodu simulace časově omezeného vstupu, konkrétně se jedná o úsek od páté vteřiny po patnáctou)
- Hledání cover verzí – `search_covers_shazam.php`
- Vyhledávání na základě zpěvu – `search_qbh_shazam.php`

Výstupem je vždy deset nejpodobnějších nahrávek spolu s příslušným skóre. Ukázka výstupu vyhledávače na konkrétní dotaz (databáze pro vyhledávání dle vzoru obsahovala část datasetu Magnatune a celý dataset Covers80):

```
$ ./search.php ./datasets/custom/magnatune_0/bad_mic/american_bach_soloists-
    joseph_haydn__masses-01-kyrie__allegro_moderato-88-117.mp3.raw
score: 16               path: datasets/Magnatune/mp3/0/
    american_bach_soloists-joseph_haydn__masses-01-kyrie__allegro_moderato
    -88-117.wav
```

```
score: 6                path: datasets/covers80/coversongs/covers32k/
    A_Whiter_Shade_Of_Pale/annie_lennox+Medusa+03-A_Whiter_Shade_Of_Pale.wav
score: 6                path: datasets/Magnatune/mp3/0/jeffrey_luck_lucas-
    what_we_whisper-03-the_pills-59-88.wav
score: 6                path: datasets/Magnatune/mp3/1/dac_crowell-sferica
    -02-murata-552-581.wav
score: 5                path: datasets/Magnatune/mp3/1/ambient_teknology-
    phoenix-09-afterburner-0-29.wav
score: 5                path: datasets/covers80/coversongs/covers32k/
    Never_Let_Me_Down_Again/smashing_pumpkins+
    For_The_Masses_A_Tribute_To_Depeche_Mode+01-Never_Let_Me_Down_Again.wav
score: 5                path: datasets/covers80/coversongs/covers32k/New_Age
    /tori_amos+Strange_Little_Girls+01-New_Age.wav
score: 5                path: datasets/covers80/coversongs/covers32k/
    My_Heart_Will_Go_On/celine_dion+Au_Coer_Du_Stade+11-My_Heart_Will_Go_On.
    wav
score: 5                path: datasets/Magnatune/mp3/0/
    american_bach_soloists-joseph_haydn__masses-16-sanctus__allegro-0-29.wav
score: 5                path: datasets/Magnatune/mp3/1/etherine-24_days-02-
    i_remembered-0-29.wav
```

B.1.3 Dílčí nástroje

Extractor Pro získání špiček v histogramu metodou Shazam slouží nástroj `Extractor` (`extractor/extractor`). Kompiluje se použitím příkazu make ve složce `extractor`. Jako parametr přijímá soubor k analýze a na výstup vrací seznam všech nalezených špiček ve tvaru `čas frekvence`, kde jednotlivé hodnoty v seznamu jsou odděleny znakem nového řádku. Čas je uveden ve počtu posuvů plovoucího okna od začátku zvukového souboru, frekvence pak odpovídá indexu pole získaného z Fourierovy transformace. Výchozí posuv okna je 64 vzorků, velikost okna je pak 1024 vzorků. Extractor podporuje zvukové soubory, jejichž čtení umožňuje knihovna `libsndfile`, tedy většinu otevřených zvukových formátů (mezi které se formát MP3 neřadí).

Hasher Pro vytvoření otisků (tedy dvojic špiček) slouží script `hasher/hasher.php`, který na vstupu očekává výstup z programu Extractor a na výstup vypíše na každý řádek vždy jeden nalezený otisk ve tvaru `čas otisk`, kde čas odpovídá času vypisovaném nástrojem Extractor.

Inserter Vkládání do databáze provádí nástroj `inserter/inserter.php`, který přijímá jako parametr cestu ke zvukovému souboru, který má být vložen. Druhým (volitelným) argumentem je jedna z hodnot `covers` a `qbh`, pokud nemá být vkládáno do databáze pro vyhledávání dle vzoru, ale pro vyhledávání cover verzí (resp. vyhledávání na základě zpěvu). Inserter automaticky spouští nástroje Extractor a Hasher.

B.2 Porovnávání melodií

B.2.1 Příprava

Podobně jako pro metodu Shazam, i při použití této metody je třeba nejprve naindexovat data. Nepoužívá se však databáze, pouze textové soubory obsahující extrahovanou melodii. Je tedy nezbytné, aby při práci s daty bylo možno do složky, která je obsahuje, zapisovat.

Pro automatické oindexování datasetů slouží scripty `tag_midi.php` a `tag_covers.php`. K jejich běhu je třeba mít zkompilované nástroje `pitchextractor` a `pitchextractor_covers`, viz sekce B.2.3.

B.2.2 Spuštění

Spuštění nástrojů je obdobné, jako nástrojů pro algoritmus Shazam. Na vyhledávání metodou porovnávání melodií slouží scripty `search_qbh.php` a `search_covers_midi.php`. Vyhledávače vrací seznam výsledků, které dosáhly shodně nejlepšího ohodnocení, spolu s ohodnocením. Příklad výstupu:

```
Best match: 00043.txt, 00023.txt
Best match score: -10
```

B.2.3 Dílčí nástroje

Pitchextractor Pro získání odhadu frekvence základního tónu slouží nástroj `Pitchextractor` (`pitchextractor/pitchextractor`, resp. `pitchextractor_covers/pitchextractor_covers`). Kompilují se příkazem `make`, nejprve je však třeba zkompilovat a nainstalovat pomocné soubory pro algoritmus Yin z knihovny Sphinxbase. To se provede ve složce `knihovny/sphinxbase` pomocí následujících příkazů:

```
./configure
make
make install
```

Použití je následující:

```
./pitchextractor/pitchextractor cesta/k/souboru.wav prah velikost_okna krok
```

Hodnota prahu určuje, jaké minimální ohodnocení jistoty získané hodnoty odhadu F_0 musí vrátit algoritmus Yin, aby byl tento odhad použit. Velikost okna a krok pak určují odpovídající vlastnosti, jednotkami je v obou případech počet vzorků při vzorkovací frekvenci 8000 Hz, na kterou je analyzovaná nahrávka převedena. Výstupem je pak mezerami oddělená sekvence čísel odpovídajících odhadu frekvence $F0$ s příslušným krokem (hodnoty jsou v Hertzích).

Pitchencoder Pro vyhlazení průběhu odhadu F_0 a převodu na sekvenci změn o půltóny slouží script `pitchencoder/pitchencoder.php` (resp. `pitchencoder_covers/pitchencoder_covers.php`) Spouští příslušný Pitchextractor s hodnotou prahu 0.1, velikostí okna 200 vzorků a krokem 40 vzorků. Výstupem je pak mezerami oddělená sekvence čísel, které udávají postupné změny v melodii v půltónech.

B.3 Automatické testování

Pro automatické testování jsou připraveny scripty ve složce `implementace/tests`, které jsou rozlišeny dle použité metody (podsložky `method_shazam` a `method_midi`). Zde jsou umístěny testovací scripty, které stačí spustit, a ony automaticky projdou dostupná data pro danou úlohy, provedou vyhledávání pro každý dotaz a vyhodnotí výsledek. Výsledky jsou vždy zobrazeny tabulkovou formou, kde jsou hodnoty False Accept, False Reject, True Accept a True Reject vždy v závislosti na nastavení různých hodnot prahu. Je-li to v dané úloze

vhodné, jsou zobrazeny i hodnoty Recall a Precision. Zobrazování tabulky vyžaduje dostatečně velké okno terminálu (alespoň 100×60 znaků) a kompatibilitu s escape sekvencemi terminálu VT100. Všechny tyto scripty akceptují parametr `--mscores=cesta_k_vystupu.m`, které způsobí, že po doběhnutí testu budou do zadaného souboru zapsána pole ohodnocení všech správně vyhledaných a všech špatně vyhledaných nahrávek (kontroluje se vždy nejlépe hodnocený výsledek, pokud jich vyhledávač vrací několik). Tato pole je možné využít v Matlabu pomocí nástroje BioSecure Tool [14].