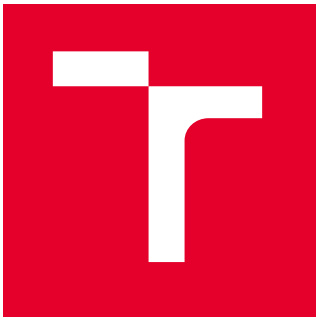
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2022 Bc. Michael Švejcar



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM PRO NALEZENÍ DUPLIKÁTŮ NAHRÁVEK NA ZÁKLADĚ AUDIO INFORMACE

SYSTEM FOR FINDING DUPLICATE RECORDINGS BASED ON AUDIO INFORMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michael Švejcar**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Matěj Ištvánek**
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Bc. Michael Švejcar ***ID:*** 203749
Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Systém pro nalezení duplikátů nahrávek na základě audio informace

POKYNY PRO VYPRACOVÁNÍ:

Popište možnosti a metody nalezení duplikátů hudebních nahrávek. Následně vytvořte systém, který vyhledá a označí duplikátní nahrávky v rámci datasetu různých interpretací stejné skladby. Implementace by měla dokázat označit stejné nahrávky, přestože se liší délkou (nahrávka pochází z různých zdrojů), potleskem, přidaným tichem apod. Nakonec bude výsledná implementovaná metoda podrobena testování na velké databázi nahrávek smyčcových kvartetů a vhodně vyhodnocena. Skripty implementujte v programovacím jazyce Python. Cílem práce je shrnutí metod nalezení duplikátů hudebních nahrávek pro následnou analýzu interpretačního výkonu a implementace skriptů pro nalezení duplikátů v datasetu včetně automatického vyhodnocení a vhodného exportu dat.

DOPORUČENÁ LITERATURA:

[1] MÜLLER, Meinard. Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications. Cham: Springer International Publishing, 2015. ISBN 978-3-319-21945-5.
[2] PRÄTZLICH, Thomas, DRIEGER, Jonathan a MÜLLER, Meinard. Memory-restricted multiscale dynamic time warping. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, s. 569–573, DOI: 10.1109/ICASSP.2016.7471739.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: Ing. Matěj Ištvánek

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato diplomová práce se zabývá metodami pro rozpoznání hudebních duplikátů v databázi souborů. Problém tkví v tom, že soubory hudebních duplikátů nemusí být vždy zcela shodné – mohou se lišit například v kvalitě či obsaženém potlesku na konci jedné z nahrávek. Cílem bylo navrhnout a implementovat systém, který shodné nahrávky identifikuje, vzájemně k sobě přiřadí a zapíše do výstupního souboru. Systém by měl být dostatečně robustní vůči již zmíněným drobným rozdílům mezi duplikáty, zároveň by měl být natolik přesný, aby nedocházelo k chybnému přiřazení vzájemně neshodných nahrávek. K těmto účelům byl použit programovací jazyk Python společně s dostupnými knihovnami pro výpočet chromagramů, techniky Image Hashing a různých variant algoritmu dynamického borcení časové osy. V rámci výsledného systému byly implementované tři různé metody, lišící se v jejich přesnosti a výpočetní náročnosti. Metody byly následně otestované na předem připraveném datasetu a na základě získaných výsledků byly vytvořeny čtyři různé úrovně přednastavené přesnosti výsledného systému. Výsledný systém se jeví jako vysoce přesný a zároveň robustní vůči nahrávkám, které jsou si velmi podobné, nikoli však shodné, jako je tomu u různých interpretací stejné skladby.

KLÍČOVÁ SLOVA

Music Information Retrival, hudební duplikáty, chromagram, dynamické borcení časové osy, Image Hashing, databáze, podobnost

ABSTRACT

This diploma thesis discusses different methods of detecting duplicates in a music file database. The problem at hand is that files containing the same recording may differ in sound quality, applause at the end of a performance and other such parameters. The aim of this thesis is to design and implement a system that identifies duplicate recordings and provides an output file for the comparison. The system needs to not be affected by the mentioned parameters but precise enough to prevent matching non-identical recordings. The system is realized using the Python programming language, freely available libraries for computing chroma features, Image Hashing technique and multiple variants of the dynamic time warping algorithm. Three comparison methods were implemented in the system, differing in precision and computation complexity. The methods were then tested on a prepared dataset and four preset precision options were created. The final system seems very precise and insusceptible to detecting recordings that are very similar but not identical as duplicates, for example in case of different interpretations of the same musical piece.

KEYWORDS

Music Information Retrieval, music duplicates, chroma features, Dynamic Time Warping, Image Hashing, dataset, similarity

ŠVEJCAR, Michael. *Systém pro nalezení duplikátů nahrávek na základě audio informace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 74 s. Diplomová práce. Vedoucí práce: Ing. Matěj Ištvánek

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Michael Švejcar
VUT ID autora:	203749
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Systém pro nalezení duplikátů nahrávek na základě audio informace

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Matěji Ištvánkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Dva zvukové soubory mezi sebou mohou vykazovat různou míru shody – od shody nejmenší, kdy se může jednat například o stejný žánr či náladu skladby, až přes absolutní shodu, kdy jsou soubory zcela totožné. Tato práce se zabývá problematikou identifikace duplikátů se shodným hudebním obsahem.

Cílem praktické části práce je vytvoření systému pro nalezení duplikátů hudebních nahrávek v jazyce Python. Systém by měl být schopný odhalit stejné nahrávky v databázi i pokud se duplikáty liší délkou, potleskem na konci, kvalitou apod. Zároveň by měl být dostatečně robustní vůči hudebním souborům, které jsou si velmi podobné, nikoli však shodné, jako je tomu u různých interpretací jedné skladby. V takovém případě se o hudební duplikáty nejedná. Algoritmus výsledné metody by zároveň měl být co nejlépe optimalizovaný z hlediska výpočetní náročnosti.

První kapitola teoretické části práce se věnuje oblasti Music Information Retrieval zabývající se získáváním a zpracováním relevantních informací z hudby. Zde je nejprve definovaná hudba společně s jejími funkcemi ve společnosti, na což je navázáno způsoby využití oblasti Music Information Retrieval. V další části práce jsou popsané některé metody a parametry používané při zpracování zvukového signálu, na které je v dalších částech práce odkazováno. Třetí kapitola se věnuje metodě dynamického borcení časové osy. Kromě základního algoritmu jsou zde uvedené jeho různé variace, které se liší v jejich charakteristických vlastnostech, zejména v přesnosti a výpočetním výkonu. Dále následuje kapitola zabývající se technikou Image Hashing, která je ve výsledném systému použita zejména k účelům jeho optimalizace. V páté kapitole jsou popsané již existující metody, které jsou k vyhledávání duplicit či podobných nahrávek v současnosti používané. Poslední kapitola se zabývá implementací výsledného systému pro vyhledávání duplicitních audio souborů. Zde jsou vysvětleny algoritmy jednotlivých variant systému společně se získanými výsledky testováním systému na datasetu.

1 Music Information Retrieval

1.1 Hudba

Hudba je typ umění, ve kterém jsou jednotlivé zvukové události systematicky uspořádány a strukturovány v jednotný a časově kontinuální celek, vyjadřující určité emoce či myšlenky pomocí různých hudebních prostředků [1]. Výběr zvuků a jejich rytmické uspořádání má vliv na estetické působení, kvalitu i funkci výsledného díla. Na základě těchto rozdílů jsou také definovány různé hudební žánry (např. jazz, rock, techno), kategorizující jednotlivá hudební díla do skupin na základě určitých podobností. Věda, která se zabývá zkoumáním hudby a s ní souvisejících jevů, se nazývá *muzikologie* [2].

Hudba doprovází člověka již odpradávna. Její historie sahá až do doby kamenné – nejstaršími dochovalými pozůstatky jsou flétny vyrobené z ptačích kostí a mamutích klů, které byly nalezeny roku 2009 v jeskyni *Hohle Fels* nacházející se v jižní části Německa [3, 4]. Podle vědeckých odhadů jsou tyto předměty staré přibližně čtyřicet tři tisíc let. Nejstarší nalezený fragment hudební notace byl nalezen na čtyři tisíce let staré sumerské tabulce, za nejstarší dochovalou kompletní píseň je ale označováno dílo *Hymnus pro Nikkalu* [5]. Píseň byla zaznamenána, společně s desítkami dalších skladeb, na hliněných tabulkách objevených v minulém století v oblasti severní Sýrie. Dle odhadů bylo dílo vytvořeno kolem roku 1400 před naším letopočtem a sloužilo jako chvalozpěv oslavující semitskou bohyni úrody. Společně s notací byly u skladby zaznamenány i instrukce o způsobu její hry na devítistrunnou lyru.

1.1.1 Funkce hudby

Ve společnosti může hudba plnit různé funkce, dle Alana P. Merriama jich existuje těchto deset základních, přičemž jejich hranice nejsou jednoznačně definované, nýbrž spíše abstraktní – v některých případech se mezi sebou prolínají a jejich definice mohou být odlišné napříč různými kulturami. Následující odstavce vychází z publikací [6, 7].

1. **Vyjadřování emocí** – Při analýze hudebních textů si lze všimnout, že hudba bývá velmi často využívána jako nástroj pro stimulaci emocí interpreta, jejich následnému vyjádření a předání posluchačům. Tímto způsobem mohou být předány různé typy emocí a nálad, ať už pozitivní či negativní – kupříkladu vyjádření smutku pomocí žalozpěvu či vyjádření lásky milostnou písní. Hudbou lze ovšem evokovat i další typy emocí a pocitů, jako je nostalgie, vyrovnanost, patriotismus či energičnost. Typickým příkladem je použití ukolébavek za účelem uklidnění a následného uspání dětí. Není ovšem neobvyklé, že hudby jako nástroje ke stimulaci emocí v posluchačích bývá i zneužíváno, například pro podněcování nenávisti vůči různým etnickým skupinám či šíření radikálních ideologií.
2. **Estetický požitek** – Asociace hudby s estetikou je nepochybně pozorovatelná u hudby západní, ale například i u hudby arabské, japonské či indonéské. Není ovšem zcela jasné, zda hudba plní funkci estetickou celosvětově, neboť samotná definice estetiky je problematická a liší se napříč jednotlivými kulturami. Dalším problémem je, že estetika je vysoce subjektivní pojem i mezi zástupci jedné kulturní skupiny – to, co jednomu připadá estetické, se může druhému jedinci jevit jako silně nevkusné.
3. **Funkce zábavní** – Funkce zábavní je v dnešní době hojně využívanou funkcí hudby. Pojí se s ní také určitý sociální aspekt, neboť hudba reprodukovaná na akcích pořádaných zábavním průmyslem bývá obvykle konzumovaná skupinově. V tomto případě se jedná o koncerty či festivaly, hudba ale plní funkci zábavní např. i během divadelních představení, v rámci filmů či u skupinových cvičení ve sportovních centrech.
4. **Komunikační funkce** – Není pochyb o tom, že lidé z odlišných kultur hovořící různými jazyky mezi sebou do jisté míry dokáží komunikovat pomocí hudby. Jako problematické se ovšem jeví to, že nemalá část sdělení, které je hudbou předávané, bývá spjaté s kulturními konvencemi, tradicemi, jazykem či estetickými standardy skupiny, ze které daná hudba pochází. Kvůli tomuto faktu může být složité či zcela nemožné určitá hudební díla interpretovat v jiných jazycích tak, aby se zachoval původní význam jejich sdělení.

5. **Symbolické vyjádření** – Majoritní část současné kognitivní psychologie hudby se zaobírá tím, jakým způsobem si lidé vnitřně vyjadřují základní hudební parametry, jako je výška tónu, témbr či harmonie. Symbolické vyjádření v sobě tedy zahrnuje určitý způsob přenášení informací, kvůli kterému se tato funkce do jisté míry prolíná s funkcí č.4. Rozdíl je ovšem v tom, že v případě symbolického vyjádření se jedná spíše o přenos nehudebních informací, jako jsou různé společenské ideály, hodnoty apod., které mohou být vyjádřeny více či méně abstraktním způsobem za pomocí textu, melodie či pomocí celkového „obrazu“, který je spojením těchto jednotlivých prostředků tvořen. Tato funkce také bývá často závislá na kulturních konvencích dané skupiny.
6. **Fyzická odezva** – Další funkcí hudby je fyzická odezva – v naší společnosti je zcela běžné, že lidé na hudbu reagují různými pohyby, zejména tancem. Díky tomu se tato funkce do jisté míry prolíná s funkcí komunikační i s funkcí vyjadřování emocí, neboť tanec obvykle bývá společenskou záležitostí. Stejně jako u předešlých funkcí je i tato funkce velmi silně závislá na kulturních aspektech dané společnosti. Za zmínku také stojí, že poslech hudby má vliv na srdeční tepovou frekvenci – například rocková hudba tepovou frekvenci zvedá, klasická hudba jí naopak snižuje [8].
7. **Prosazování společenských norem** – Písně jakožto nástroje pro ovládání společnosti bývají používané napříč mnoha kulturami. Význam takovýchto skladeb může být adresován přímo, v podobě varování před špatným chováním, jednat se ale může i o formu nepřímého stanovení – v takovém případě z významu písní vyplývá, co je v dané společnosti považováno za správné chování. Tento jev je možný pozorovat například u zahajování různých náboženských rituálů, při kterých jsou mladší členové komunity vzděláváni ke správnému chování skrze písně.
8. **Validace společenských norem a náboženských rituálů** – Písně mohou být použity k validaci a upevňování určitých společenských norem – příkladem je validace náboženských systémů skrze texty pojednávající o církevní tématice. Na stejném principu bývá upevňován i folklor, vyprávěním mýtů a bájí texty lidových písní.
9. **Podpora kontinuity a stability kultury** – Pokud lze hudbu použít k vyjadřování emocí, ke zprostředkování estetického požitku, k zábavě i ke komunikaci, dále pomocí ní lze vyjadřovat symboliku, vyvolávat fyzickou odezvu jedinců či prosazovat i upevňovat různé společenské normy a konvence – pak z tohoto všeho vyplývá, že hudba napomáhá kontinuitě a stabilitě kultury. Po této stránce hudba možná nepřispívá o nic více či méně nežli ostatní kulturní aspekty, ovšem málokterý jiný kulturní aspekt plní ve společnosti funkcí tolik, jako právě hudba.

10. **Podpora sociální integrace** – Jednou z klíčových funkcí hudby je podpora sociální integrace. Již odnepaměti se lidé díky hudbě shromažďují. Díky tomu mají účastníci pocit sounáležitosti s ostatními a zároveň pocit uspokojení ze zapadnutí do kolektivu, se kterým sdílí podobné životní hodnoty, formy umění i životní styl.

1.1.2 Hudební interpretace

Zahrají-li různí hudebníci stejnou skladbu dle identického notového zápisu, jejich přednes nikdy nebude identický – tyto jednotlivé verze se nazývají hudebními interpretacemi. Rozdíly v jednotlivých interpretacích bývají zejména v tempu, v dynamice přednesu, v délkách trvání jednotlivých not, ale i v celkové délce trvání skladby či v témbru použitých nástrojů. U některých hudebních žánrů je navíc zcela běžné, že umělec v určitých částech interpretované skladby improvizuje – v takovém případě dojde k vynechání některých not, popřípadě k jejich záměně za jiné či naopak k přidání not zcela nových. Typickým příkladem je použití techniky *basso continuo*, při které hráč dle číselných údajů uvedených v notaci improvizuje harmonii nad basovou linkou dle svého uměleckého cítění [9].

Při pořízení zvukových záznamů těchto jednotlivých interpretací se rozdíly mezi nahrávkami mohou ještě o to více prohloubit, pokud je každá nahrávka pořízena pomocí jiného mikrofonu, v prostoru s odlišnou akustikou, zakódovaná do odlišného zvukového formátu apod.

1.2 Využití MIR

Music Information Retrieval (zkráceně MIR) je oblast výzkumu, která se zabývá získáváním, analýzou a zpracováním relevantních informací z hudby v různých formách – může se jednat například o hudební nahrávku, notový zápis či soubor MIDI.

Jedná se o poměrně mladou oblast, která se začala formovat teprve během 90. let minulého století. Od té doby zažívá rozmach, zejména díky vzniku formátů využívajících ztrátovou kompresi (např. mp3, ogg), vzrůstajícímu výkonu osobních počítačů či dostupnosti přenosných zařízeních umožňujících přehrávání hudby. V poslední době měl na rozmach oblasti MIR dopad i celosvětový nárůst používání hudebních streamovacích služeb jako je *Spotify*[1], *Tidal*[2] či *SoundCloud*[3] [10]. Výrazným milníkem pro rozmach MIR bylo také založení neziskové organizace ISMIR (The International Society for Music Information Retrieval) v roce 2008 [11]. Význam této organizace

[1] `https://www.spotify.com`
[2] `https://tidal.com`
[3] `https://soundcloud.com`

spočívá v podporování výzkumu a vývoje nových metod pro získávání hudebních informací, vytvoření komunity pro možnost spolupráce i podpora vzdělávání v dané oblasti. Organizace každý rok pořádá konference, kde jsou představovány a diskutovány nové metody získávání hudebních informací.

1.2.1 Kategorizace hudby

Klasickým problémem, který bývá řešen pomocí technik MIR, je automatická kategorizace hudebních skladeb do předem definovaných skupin na základě určitých podobností. Skladby lze kategorizovat například dle jejich žánrů, ale i dle jejich celkové nálady či dalších parametrů [12, 13].

1.2.2 Navrhování skladeb

Další oblastí, na kterou lze techniky MIR uplatnit, jsou systémy pro automatické navrhování skladeb, které jsou v současnosti používané zejména u hudebních streamovacích služeb. Jejich princip spočívá v tom, že pokud posluchači jedna skladba dohraje, platforma ho vyzve k výběru následující skladby ze seznamu. Skladby v seznamu jsou navrženy na základě podobností s předchozí skladbou, čímž se zvyšuje pravděpodobnost, že uživatele nahrávka zaujme a v používání dané služby bude pokračovat. U některých služeb bývá další skladba vybrána a spuštěna zcela automaticky, aniž by byl uživatel k jejímu vybrání ze seznamu vyzván. Metody pro navrhování skladeb lze dělit do tří základních kategorií – navrhování na základě metadat, kolaborativní filtrování a doporučování na základě obsahu [14, 15].

Navrhování na základě metadat

Nejstarším a také nejprimitivnějším způsobem je vytváření návrhů na základě metadat. V takovém případě je nadcházející skladba zvolena např. dle informací o žánru, autorovi či zpěvákovi. Výsledky dosažené touto metodou ovšem nebývají v porovnání s jinými metodami velmi relevantní.

Kolaborativní filtrování

Metoda kolaborativního filtrování pracuje na základě kombinace dvou základních principů. První princip funguje na základě preferencí a výsledků hodnocení skladeb jinými uživateli s podobným hudebním vkusem. Druhým principem je tvorba návrhu dle skupin skladeb, ve kterých se aktuálně přehrávaná skladba nachází (např. album či playlist). Předpokladem je, že pokud se uživateli líbí skladba A z dané kategorie, je velmi pravděpodobné, že se mu ze stejné skupiny bude líbit i skladba B [16].

Doporučování na základě obsahu

V případě doporučování na základě obsahu jsou skladby nejprve analyzovány za účelem získání vhodných parametrů – jedná se o parametry popisující například rytmus nahrávek, vývoj melodie, celkovou náladu či témbr. Všechny tyto informace jsou ukládány ve strukturovaném formátu. Výsledný návrh je vyhodnocen na základě podobností těchto informací mezi různými skladbami [14, 17].

Největším problémem této metody v současnosti je velký počet skladeb v databázích. Tím je kladen velký nárok na výpočetní výkon i čas, který je k získání parametrů a jejich porovnání potřeba. Díky tomuto faktu jsou v současnosti vyvíjené a používané různé techniky optimalizované z hlediska výpočetní náročnosti, jako např. *audio fingerprinting* [18].

1.2.3 Identifikace jednotlivých hudebních nástrojů a separace stop

Dalším problémem, který lze pomocí MIR technik řešit, je automatické rozpoznání hudebních nástrojů, případně lidského hlasu v nahrávce a jejich následné izolování do jednotlivých stop [19]. Tuto funkcionalitu nabízí například open-source projekt *Spleeter*[4], který funguje na bázi umělé inteligence. Nástroj nabízí předpřipravené modely pro separaci stop, uživateli ale také umožňuje tvorbu modelů z vlastních datasetů. Zde je však nutno podotknout, že i přes to, že jsou tyto nástroje postupně čím dál preciznější, v jednotlivých extrahovaných stopách jsou obvykle slyšet určité spektrální pozůstatky z jiných nástrojů.

Procesu oddělení jednotlivých stop bývá dále využíváno například v rámci programů pro automatický převod mluvené řeči do textu [20], neboť ve zpracovávané nahrávce jsou vždy zastoupeny i okolní ruchy a šumy daného prostředí, které je před samotným zpracováním řeči třeba potlačit. Zatímco lidé nemají s psychologickým oddělením vnímaného hlasu, na který se soustředí, od okolních ruchů a ostatních hlasů příliš velký problém (tzv. *fenomén koktejlové párty* [21]), v případě zpracování akustického signálu počítačem se jedná o poměrně náročný úkol, který bývá často řešen pomocí neuronových sítí.

1.2.4 Automatický přepis hudby do not

Automatickým přepisem hudby do not se myslí proces, během kterého jsou ze vstupního hudebního signálu získány informace o tónovém i rytmickém vývoji skladby v čase [22]. Na základě těchto informací je posléze automaticky vygenerován notový

[4] `https://research.deezer.com/projects/spleeter.html`

zápis, soubor `.midi` či jiný formát souboru reprezentující tento typ hudebních informací. Celý proces se skládá z několika dílčích úkolů – zjištění tónového vývoje nahrávky v čase, detekce počátečních i koncových bodů jednotlivých tónů, výpočtu hlasitosti, rozpoznání hudebních nástrojů a extrakce rytmických informací s časovou kvantizací. Problémové mohou být polyfonní mixtury, ve kterých hraje více hudebních nástrojů zároveň. To může mít za následek neblahý vliv na přesnost výstupního materiálu. Tyto problémy mohou být redukovány tím, že se nástroje nejprve separují do jednotlivých hudebních stop (viz kapitola 1.2.2) a poté se analyzují individuálně.

1.2.5 Automatické generování hudby

Počátky automatického generování hudby sahají až do 60. let minulého století, kdy ruský výzkumník R. Kh. Zaripov zveřejnil první publikaci o algoritmické hudební kompozici [23].

Metody používané v současnosti se dělí do dvou základních kategorií – v první kategorii je algoritmus použitý ke generování hudby zadefinovaný člověkem (např. fraktálová hudba nebo translační modely, při kterých je výsledná posloupnost hudebních událostí vygenerovaná na základě určitých vstupních dat, například fotografie krajiny ve formátu .png), ve druhé kategorii se používá kombinace vstupního datasetu se strojovým učením, které v těchto datech vyhledává určité vzory a dle zjištěných vzorů přizpůsobí algoritmus určený ke generování hudebního díla. Pokud je například umělá inteligence natrénovaná na datasetu Mozartových kompozicí, styl výsledných vygenerovaných skladeb bude velmi podobný. Tento postup v současnosti zažívá velký rozmach, za zmínku jistě stojí například to, že pomocí umělé inteligence byla dokončena Beethovenova 10. symfonie [24, 25]

1.3 Základní aspekty MIR

Hudební informace, které lze z hudebních nahrávek získat, se skládají ze sedmi základních aspektů [29]. Těmi jsou výška tónu, časový aspekt, harmonie, témbr, vydavatelský aspekt, textový aspekt a bibliografický aspekt. Tyto aspekty se navzájem nevylučují, naopak se mezi sebou často prolínají – například hudební pojem *adagio* lze považovat za časový i vydavatelský aspekt, záleží na kontextu příslušného díla.

1.3.1 Výška tónu

Výška tónu je aspekt, který závisí především na fundamentální frekvenci daného tónu (fundamentální frekvence neboli první harmonická složka, je rovna nejnižší frekvenci daného periodického signálu). Nejedná se ovšem o čistě objektivní fyzikální

vlastnost, nýbrž o subjektivní atribut zvuku [26]. Věda, která se věnuje subjektivnímu vnímání zvuku člověkem, se nazývá psychoakustika [27]. Jednou z nejčastěji používaných reprezentací vyjadřující výšku tónu je vertikální umístění noty v notové osnově. Výšku tónu lze ovšem zapsat mnoha způsoby, například názvem noty či pomocí frekvence.

Rozdíl mezi dvěma výškami tónu se nazývá interval. Velikost intervalu může být vyjádřena počtem půltónů či pomocí názvů intervalů používaných v rámci hudební teorie (např. čistá oktáva, malá tercie).

1.3.2 Časový aspekt

Informace vztahující se k délce trvání různých hudebních událostí spadají pod časový aspekt. Jedná se o tempová označení, metrum, délky tónů, délky harmonií a akcenty. Časové informace mohou být absolutní (např. 180 BPM[5]), relativní (accelerando, ritardando) či obecné (andante, allegro).

Vzhledem k tomu, že celková rytmická struktura skladby je dána kombinací výše popsaných parametrů, je možné ji vyjádřit různými způsoby. Při interpretaci některých hudebních žánrů (např. baroko, jazz) je navíc běžné, že se umělec v rámci sebevyjádření odchyluje od přesně daných rytmických hodnot uvedených v notovém zápise. Kvůli těmto faktům může být získávání časových informací z hudby nejednoznačnou úlohou.

1.3.3 Harmonie

Pokud v hudebním díle zní dva či více tónů o různých výškách současně, jedná se o harmonii, někdy také nazývanou jako polyfonie. Klasickým příkladem harmonie je využití akordů neboli současného souzvuku tří tónů. Opakem polyfonie je monofonie, při níž je využíváno pouze jediného hlasu.

1.3.4 Témbr

Jako témbr se označují všechny vlastnosti udávající zvukovou barvu. Díky témbru je od sebe možné rozeznat například kytaru od flétny, i přes to, že na oba nástroje je zahraný stejný tón. Informace o instrumentaci skladby tudíž spadá pod tento aspekt. Výsledný témbr ovšem může ovlivnit například i použitá technika hry na daný hudební nástroj (např. pizzicato, použití dusítka). V takovém případě dochází k prolínání hranic témbru a vydavatelského aspektu, neboť informace o použití těchto technik mohou být uvedeny přímo v notovém zápise.

[5]BPM = Beats Per Minute, česky počet úderů za minutu

1.3.5 Vydavatelský aspekt

Pod vydavatelský aspekt spadají zejména instrukce vyjadřující například vhodný prstoklad, artikulaci či techniku hry v daném místě notového zápisu (např. *pianissimo* nebo *forte*). Tyto informace nemusí být v hudebním díle uvedeny vůbec, nebo mohou být vyjádřeny symboly, číselně či oběma způsoby najednou. Není výjimkou, že v různých verzích stejného hudebního díla se tyto informace liší, což komplikuje výběr vhodné verze dané práce pro použití v rámci MIR systému.

1.3.6 Textový aspekt

Textový aspekt se skládá z textů písní, chorálů, árií, symfonií a obdobných hudebních forem. Není však závislý na použité melodii ani aranži – ze samotné části textu skladby obvykle není možné odvodit související melodii a naopak pouze z melodie nelze odvodit text písně. Písně navíc bývají překládány do vícero jazyků za použití stejné melodie, v některých případech bývá i jedna melodie použita s různými texty ve stejném jazyce.

1.3.7 Bibliografický aspekt

Informace týkající se například názvu skladby, jména jejího autora, autora textu, vydavatelství i data vydání spadají pod bibliografický aspekt. Jedná se o jediný aspekt hudebních informací, které nebývají získávány ze samotného díla – jde spíše o typ informací, které jsou použity k jeho popisu. Jinými slovy, jedná se o hudební metadata.

2 Zpracování zvukového signálu

V této kapitole jsou popsané některé metody a parametry používané při zpracování zvukového signálu společně se způsoby jejich reprezentace, na které je odkazováno v následujících kapitolách práce.

2.1 Efektivní hodnota

Efektivní hodnota (zkráceně RMS) je veličina popisující velikost signálu [28]. V případě diskrétního signálu pro RMS série n vzorků platí:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{i=n} x_i^2} \quad (-), \tag{2.1}$$

kde x_i je hodnota i-tého vzorku časové posloupnosti.

2.2 Fourierova transformace

Fourierova transformace (zkráceně FT) je metoda umožňující převod spojitého signálu x_t z časové oblasti do oblasti frekvenční [30]. Výstupem FT je tzv. frekvenční spektrum X_ω, které signál reprezentuje sumou harmonických funkcí sinus, resp. kosinus. Spektrum signálu lze získat aplikací transformace pomocí vztahu:

$$X_\omega = \int_{-\infty}^{\infty} x_t \cdot \mathrm{e}^{-\mathrm{j}\omega t} dt. \tag{2.2}$$

Transformaci lze provádět oběma směry, spektrum X_ω je možné převést zpět na signál x_t aplikací tzv. Inverzní Fourierovy transformace (zkráceně IFT). Nevýhodou transformace je však fakt, že spektrální šířka odpovídající jednotlivým výstupním koeficientům je konstantní, zatímco vnímání zvukových frekvencí člověkem je téměř logaritmické. Kvůli tomuto faktu je přesnost metody v závislosti na lidském vnímání zvuku nízká na spodních kmitočtech, zatímco na frekvencích vysokých je přesnost metody naopak často zbytečně vysoká.

V případě zpracování diskrétního signálu x_n vyjádřeného konečným počtem vzorků n v čase, se k získání frekvenčního spektra X používá tzv. Diskrétní Fourierova transformace (zkráceně DFT), viz vzorec:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \mathrm{e}^{-\mathrm{j}\cdot 2\pi \cdot \frac{k}{N} n}, \tag{2.3}$$

kde X_k je k-tá harmonická frekvence ve frekvenční oblasti, x_n je n-tý vzorek signálu v časové oblasti, k je pořadí výstupního vzorku, n je pořadí vstupního vzorku a N reprezentuje počet vzorků vstupního signálu.

Výpočet DFT je možné optimalizovat použitím Rychlé Fourierovy transformace (zkráceně FFT). Výsledky získané oběma metodami jsou shodné, FFT je však výhodnější z hlediska její výpočetní náročnosti $O(N \log(N))$, která je narozdíl od DFT o výpočetní náročnosti $O(N^2)$ podstatně nižší. Způsobů pro výpočet FFT existuje několik, přičemž nejčastěji používaným je tzv. Cooley-Tukey algoritmus vynalezený v šedesátých letech minulého století [32]. Většina algoritmů pro výpočet FFT vyžaduje délku bloku vstupního signálu o počtu 2^n vzorků, kde $n \in \mathbb{N}^+$.

Pro zjištění vývoje spektrálního obsahu nahrávky v čase se vstupní signál rozdělí na bloky o konstantní velikosti definované délkou časového okna. Výpočet FT se poté provede pro každý blok zvlášť, přičemž jednotlivé bloky se mezi sebou obvykle do jisté míry překrývají. Delší časové okno znamená lepší rozlišení v kmitočtové oblasti, zatímco použití kratšího časového okna vede k lepšímu rozlišení v oblasti časové [28]. Tento postup je nazýván jako Krátkodobá Fourierova transformace (anglicky Short-time Fourier transform, zkráceně STFT).

2.3 Diskrétní kosinová transformace

Diskrétní kosinová transformace (anglicky Discrete Cosine Transform, zkráceně DCT) je metoda podobná Fourierově transformaci, pomocí které lze spektrum signálu vyjádřit sumou harmonických funkcí kosinus [33]. Metodu využívá například kompresní formát `.jpeg` určený ke ztrátovému kódování obrazových souborů [31]. Pro výpočet DCT signálu $x(n), n \in \{0, 1, 2, ..., N\}$ platí vztah:

$$X(k) = \sum_{n=0}^{N-1} \alpha(n) x(n) \cos \frac{(2k+1)n\pi}{2N}, k = 0, 1, ..., N, \tag{2.4}$$

kde $\alpha(n) = \begin{cases} \sqrt{\frac{1}{N}} & \text{pro } n = 0, \\ \sqrt{\frac{2}{N}} & \text{v opačném případě.} \end{cases}$.

2.4 Lineární frekvenční kepstrální koeficienty

Lineární frekvenční kepstrální koeficienty (anglicky linear frequency cepstral coefficients, zkráceně LFCC) popisují témbr vstupního zvukového signálu [34]. Koeficienty se získají aplikací FFT a následnou filtrací kmitočtového spektra pomocí banky filtrů. Posledním krokem je použití DCT k převedení signálu do časové domény.

2.5 Konstantní Q transformace

Konstantní Q transformace (anglicky Constant-Q transform, zkráceně CQT) byla navržena k překonání nedostatečného spektrálního rozlišení STFT na nízkých kmitočtech při použití časového okna běžné délky [35]. Výstupem metody jsou podobně jako u DFT spektrální koeficienty, rozdílem je však jejich logaritmické rozložení na frekvenční ose. Z těchto důvodů je metoda často používaná ke spektrální analýze hudebních nahrávek. Pro výpočet CQT diskrétního signálu $x(n)$ platí vztah:

$$X^{CQ}(k,n) = \sum_{j=n-\lceil N_k/2 \rceil}^{n+\lceil N_k/2 \rceil} x(j) a_k^*(j - n + N_k/2), \tag{2.5}$$

kde $k \in \{1, 2, ..., K\}$ udává pozici spektrálního koeficientu, $\lceil \cdot \rceil$ označuje zaokrouhlení směrem k zápornému nekonečnu a $a_k^*(n)$ vyjadřuje komplexně sdružené číslo k $a_k(n)$, pro které platí:

$$a_k(n) = \frac{1}{N_k} \omega(\frac{n}{N_k}) \mathrm{e}^{-\mathrm{i}2\pi n \frac{f_k}{f_s}}, \tag{2.6}$$

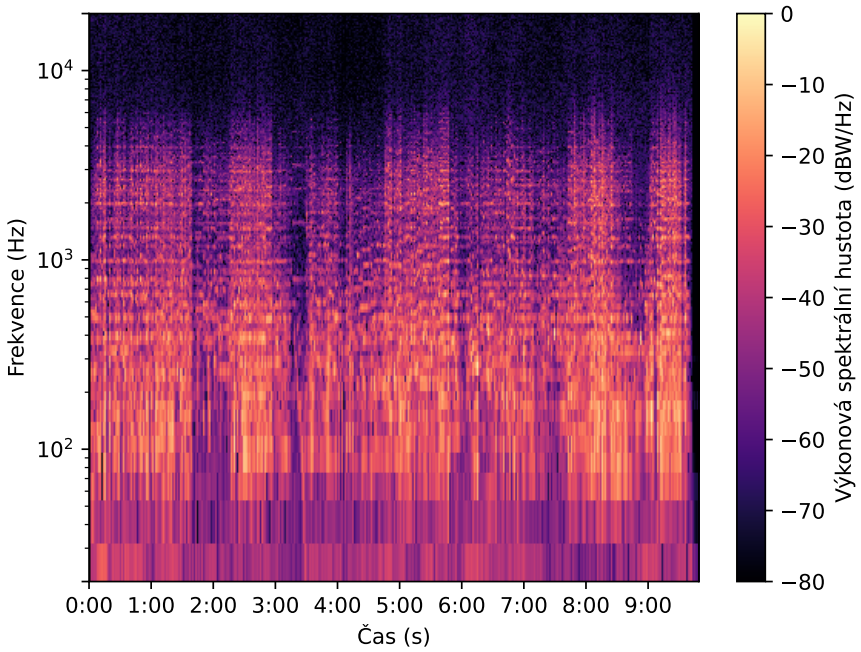
kde f_k je střední kmitočet k-tého koeficientu, f_s označuje vzorkovací kmitočet a ω je funkce váhovacího okna (např. Hannovo okno či Blackmanovo okno).

2.6 Spektrogram

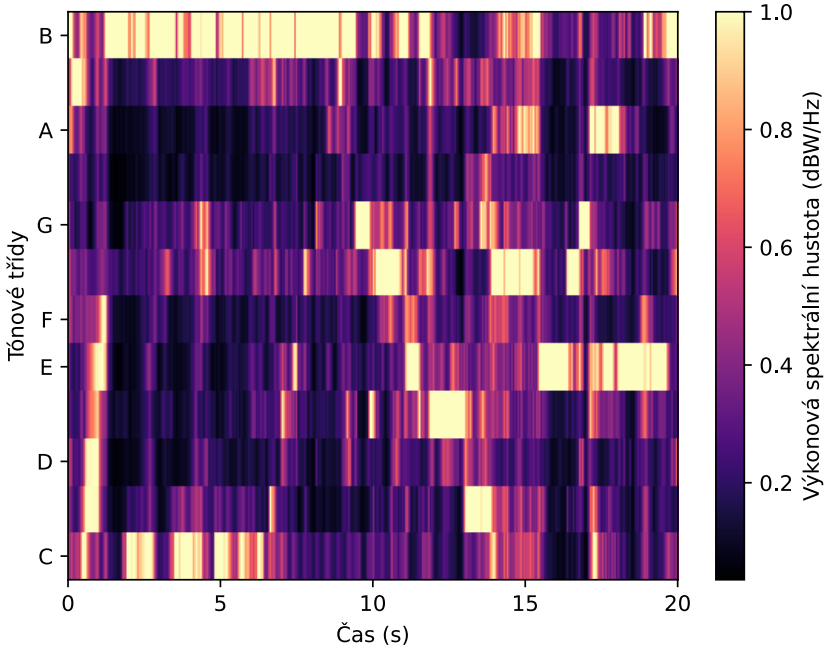
Informace o frekvenčním vývoji zvukové nahrávky v čase získaných pomocí STFT či CQT je možné graficky vizualizovat. K těmto účelům se používá tzv. spektrogram, ve kterém je na jedné ose promítnut čas, ve druhé ose frekvence a výkonová spektrální hustota je obvykle vyjádřena barevným odstínem (viz obr. 2.1).

2.7 Chromagram

Pomocí chromagramu lze popsat tónový vývoj zvukové nahrávky v čase [28]. Jedná se o posloupnost chroma vektorů (anglicky pitch-class profiles), které popisují spektrální energii daného úseku zvukové nahrávky v tónových třídách. V případě této práce jsou chromagramy tvořeny dvanácti-dimenzionálními chroma parametry, přičemž každá dimenze vektoru vyjadřuje jednu tónovou třídu. Jednotlivé tónové třídy jsou určeny dle západní chromatické stupnice, ve které je každá oktáva rozdělena na dvanáct půltónů (viz obr. 2.2)



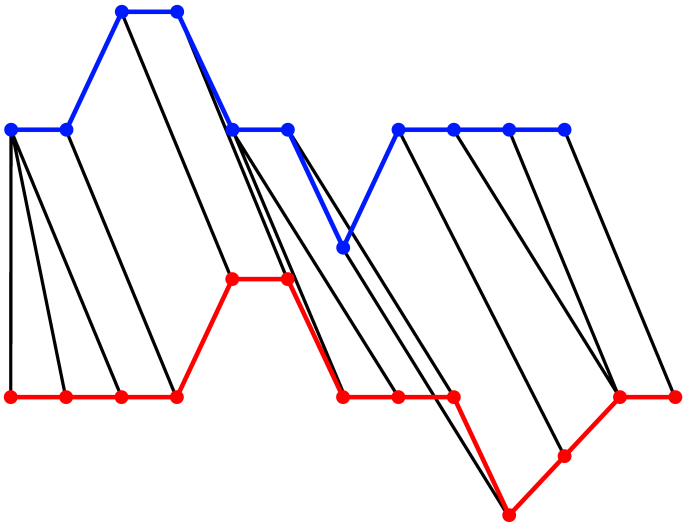
Obr. 2.1: Spektrogram



Obr. 2.2: Chromagram

3 Dynamické borcení časové osy

Dynamické borcení časové osy (anglicky *dynamic time warping*, zkráceně DTW) je metoda, pomocí které lze zjistit podobnost časových posloupností dvou vstupních signálů. Výstupem DTW je tzv. optimální cesta p, která k sobě přiřazuje korespondující body vstupních časových posloupností (viz obr. 3.1). Výhoda metody spočívá zejména v její robustnosti vůči vzájemným změnám časových vzdáleností korespondujících bodů během doby trvání signálů. V současnosti je metoda DTW využívaná v celé řadě odvětví, původně však byla používaná k automatickému rozpoznávání slov ve zvukových nahrávkách [36].



Obr. 3.1: Přiřazení korespondujících prvků časových posloupností pomocí DTW

3.1 Popis algoritmu

Definujme dvě posloupnosti X a Y jako:

$$X = (x_1, x_2, ..., x_N), N \in \mathbb{N}, \tag{3.1}$$

$$Y = (y_1, y_2, ..., y_M), M \in \mathbb{N}. \tag{3.2}$$

Tyto posloupnosti mohou být časovými posloupnostmi diskrétních signálů, obecně se však může jednat o vyjádření jakékoli sekvence zaznamenané s konstantně velkými časovými úseky mezi jednotlivými body.

Prvním krokem algoritmu je inicializace tzv. vzdálenostní matice (anglicky *cost matrix*) $D = (d_{\mathrm{i,\,j}})_{\mathrm{M+1,\,N+1}}$, viz:

$$D = \begin{pmatrix} d_{\mathrm{M,0}} & \ldots & \ldots & d_{\mathrm{M,N}} \\ \ldots & \ldots & \ldots & \ldots \\ d_{1,0} & \ldots & \ldots & \ldots \\ d_{0,0} & d_{0,1} & \ldots & d_{0,\mathrm{N}} \end{pmatrix},$$

přičemž platí, že:

$$d_{0,\,0} = 0, \tag{3.3}$$

$$d_{\mathrm{i},\,0} = \infty, i \in \langle 1, M \rangle \subset \mathbb{N}, \tag{3.4}$$

$$d_{0,\,\mathrm{j}} = \infty, j \in \langle 1, N \rangle \subset \mathbb{N}. \tag{3.5}$$

Hodnoty zbylých prvků $d_{\mathrm{i,j}}$ matice D se vypočítají dle vztahu:

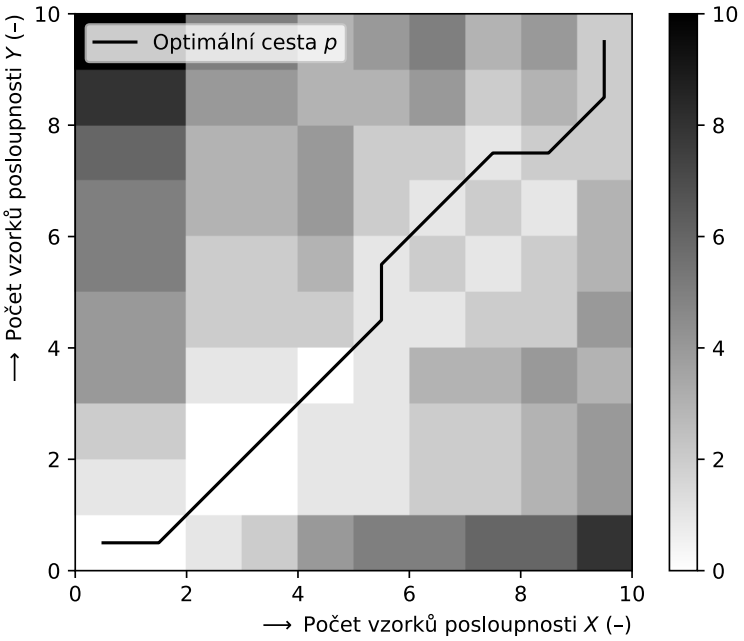
$$d_{\mathrm{i,j}} = |x_i - x_j| + \min \begin{cases} d_{\mathrm{i\text{-}1,\,j\text{-}1}} \\ d_{\mathrm{i\text{-}1,\,j}} \\ d_{\mathrm{i,\,j\text{-}1}} \end{cases} \tag{3.6}$$

Po vypočtení celé vzdálenostní matice D zbývá určit, kterými prvky matice prochází optimální cesta p (viz obr. 3.2). Její výpočet se provádí zpětně, od pravého horního rohu matice (prvek $d_{\mathrm{M,N}}$) až po roh vlevo dole (prvek $d_{0,0}$). Podmínkou při výpočtu je, že dílčí část cesty mezi každými dvěma prvky matice může jít pouze jedním ze tří směrů – vertikálně, horizontálně či po diagonále ve směru vpravo nahoru. Pokud je tedy na začátku výpočtu znám pouze jediný bod optimální cesty p – prvek matice $d_{\mathrm{M,N}}$, dle těchto podmínek může být jako následující bod vyhodnocen pouze prvek $d_{\mathrm{M\text{-}1,N}}$, $d_{\mathrm{M,N\text{-}1}}$ či $d_{\mathrm{M\text{-}1,N\text{-}1}}$. Následujícím bodem optimální cesty se stane právě ten prvek z těchto tří, jehož hodnota je nejmenší. Tento proces se opakuje tak dlouho, dokud se optimální cesta zpětně nedopočítá až do prvku $d_{0,0}$.

Po zjištění, které prvky matice náleží optimální cestě, lze také vypočítat celkovou vzdálenost C dvou vstupních posloupností, což je parametr vyjadřující míru podobnosti daných signálů. Pro výpočet celkové vzdálenosti C platí vztah:

$$C = \frac{\sum_{i=1}^{k} p(i)}{k}, \tag{3.7}$$

kde *p(i)* jsou hodnoty jednotlivých prvků matice, které odpovídají optimální cestě p a parametr k je celkový počet těchto prvků [37, 38, 39].



Obr. 3.2: Vzdálenostní matice D a optimální cesta p

Z definice metody DTW vyplývá, že při její aplikaci je nutné vypočítat jednotlivé prvky matice D pro každou dvojici bodů vstupního signálu X se vstupním signálem Y. To znamená, že výpočetní náročnost metody je $O(N \cdot M)$. S rostoucí velikostí vstupních vektorů tedy exponenciálně narůstá i čas potřebný k výpočtu cesty p, zároveň jsou v takovém případě kladeny vyšší nároky na paměť výpočetního systému, neboť počet prvků matice D je roven $N \cdot M$. Z těchto důvodů byly vyvinuty různé metody, pomocí kterých lze proces výpočtu DTW optimalizovat, což vede k nižší časové náročnosti. Tyto metody jsou popsány v následujících podkapitolách.

3.2 Podvzorkování vstupních signálů

Jedním ze základních způsobů, jak lze výpočet DTW urychlit, je podvzorkování vstupních signálů X a Y předem definovaným činitelem $T \in \mathbb{N}$. V případě, že signál X má délku N a signál Y délku M, velikost vstupních vektorů po podvzorkování bude N/T, resp. M/T. Počet prvků matice D tedy bude $\frac{N}{T} \cdot \frac{M}{T} = \frac{N \cdot M}{T^2}$, z čehož vyplývá, že s rostoucím činitelem podvzorkování T exponenciálně klesá čas potřebný k výpočtu DTW. Samotné podvzorkování signálu lze provést jeho filtrací pomocí filtru typu dolní propust a následným vynecháním určitých vzorků, či pomocí lineární aproximace.

Tento způsob optimalizace DTW je vhodné použít v případech, kdy není třeba znát optimální cestu p s přesností na jeden vzorek vstupních signálů. Její nevýhodou je fakt, že podvzorkováním signálů nevyhnutelně dochází ke ztrátám informací, což od určité velikosti činitele T vede k získání velmi nepřesných, až zcela irelevantních průběhů takto získané optimální cesty p [40].

3.3 Omezení oblasti výpočtu optimální cesty pomocí metod Itakura, Sakoe-Chiba

Dalším způsobem, jak lze algoritmus DTW urychlit, je omezení oblasti pro výpočet optimální cesty p. Postup vychází z předpokladu, že pro některé prvky matice D je velmi nepravděpodobné, že jimi bude optimální cesta procházet – například rohové prvky $d_{0,\mathrm{M}}$ či $d_{\mathrm{N},0}$. Metody *Itakura* a *Sakoe-Chiba* (viz obr 3.3) tedy definují množiny bodů, pro které se výpočet DTW provádí (bílé regiony) a zároveň množiny bodů, u kterých se výpočet neprovádí (šedé regiony), protože hodnoty vzdálenostní funkce (viz vzorec 3.6) pro tyto prvky velmi pravděpodobně není třeba znát.

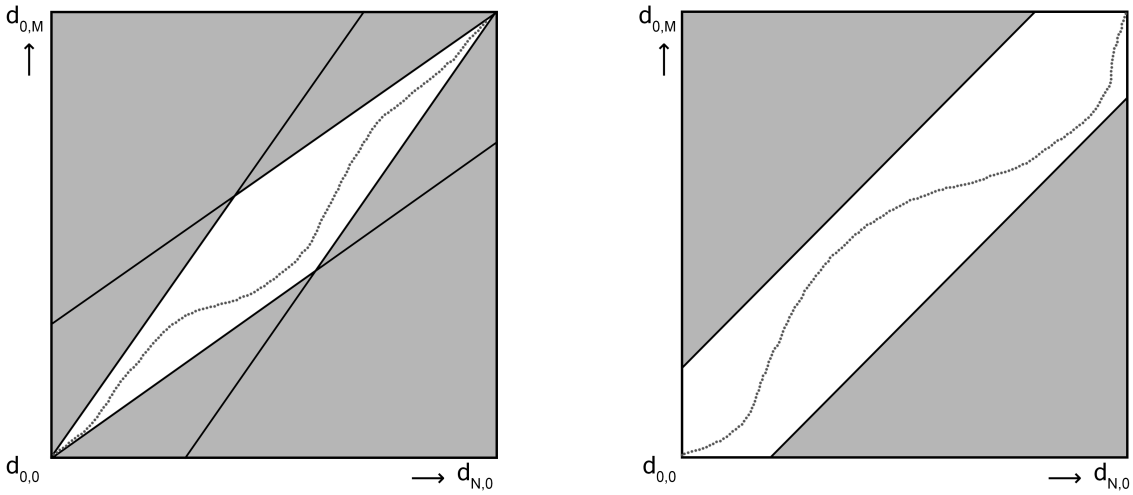
Oblast výpočtu u *Sakoe-Chiba* je vymezená pásmem, které se skládá ze dvou úseček rovnoběžných s hlavní diagonálou $i = j$. Šířka pásma je konstantní a předem definovaná jako $R \in \mathbb{N}$. Parametr R specifikuje délku vertikálního vektoru, který určuje vzdálenost horní hranice *Sakoe-Chiba* pásma od hlavní diagonály a zároveň vzdálenost spodní hranice pásma k hlavní diagonále, přičemž platí, že:

$$|i(k) - m| \leq R, |j(k) - m| \leq R, \tag{3.8}$$

kde m je korespondující bod na hlavní diagonále a k vyjadřuje k-tý bod optimální cesty.

U metody *Itakura* není šířka pásma konstantní – největší je v prostředku diagonály $i = j$, nejmenší naopak v rohových pozicích matice.

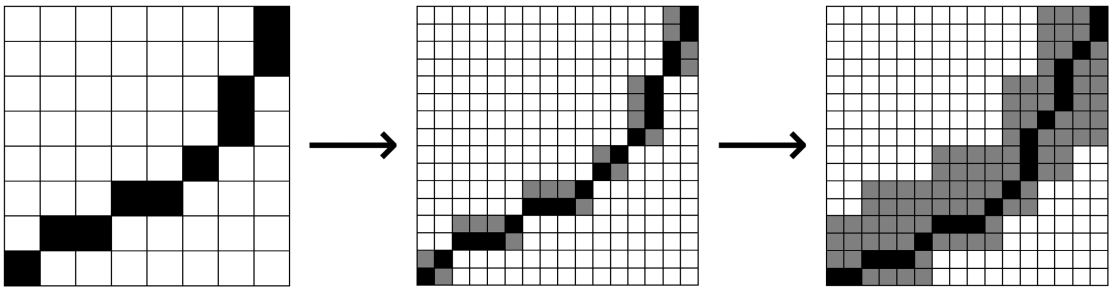
Výhodou těchto metod je jejich schopnost urychlit výpočet DTW, protože redukují počet prvků, pro které se počítá vzdálenostní funkce (viz vzorec 3.6). Zde je ovšem nutné podotknout, že doba výpočtu DTW se stále bude s rostoucí velikostí vstupních signálů zvětšovat exponenciálně. Poměrně velkou nevýhodou je ovšem fakt, že kdyby optimální cesta mimo vymezenou oblast doopravdy ležela, získané výsledky budou nepřesné – průběh optimální cesty vyjde jiný, než kdyby se výpočet DTW provedl bez těchto omezení, či pokud by velikost vymezené oblasti byla dostatečně velká, aby jí všechny body optimální cesty náležely [40].



Obr. 3.3: Omezení oblasti výpočtu optimální cesty pomocí metod Itakura (vlevo) a Sakoe-Chiba (vpravo)

3.4 Multiscale DTW

Multiscale DTW (zkráceně MsDTW) je iterační metoda kombinující princip podvzorkování vstupních signálů s omezením oblasti výpočtu cesty (viz kapitoly 3.2 a 3.3) v závislosti na mezivýsledcích získaných v předchozích iteracích algoritmu (viz obr. 3.4). V prvním kroce se vstupní signály X_1 a Y_1 o délkách N_1 a M_1 podvzorkují činitelem $T_2 \in \mathbb{N}$, čímž se získají signály X_2 a Y_2 o délkách N_2 a M_2. V následujícím kroce se v tomto rozlišení vypočte optimální cesta p_2^* (úroveň 2). Získaná cesta je poté promítnuta do matice s vyšším rozlišením (úroveň 1), kde svým průběhem vymezuje oblast R pro výpočet DTW na nižší úrovni. Takto získaná oblast má velikost $L_2 \cdot T_2^2$ prvků, přičemž L_2 udává velikost p_2^*. Následně se vypočte optimální cesta p_R^* náležící oblasti R. Celkový počet prvků, pro které se v tomto případě počítá vzdálenostní funkce (viz vzorec 3.6), je $N_2 \cdot M_2 + L_2 \cdot T_2^2$, narozdíl od $N \cdot M$ prvků v případě použití klasické jednoúrovňové DTW. Tento postup může být opakován vícekrát s tím, že se vždy začíná na úrovni nejvyšší a rozlišení výpočtu se zvyšuje postupnou rekurzí algoritmu – toho využívá např. metoda FastDTW, která je popsaná v následující kapitole. Vzhledem k tomu, že vlivem podvzorkování a s ním spojených ztrát informací mohou být výsledky získané na vyšších úrovních nepřesné, při výpočtu na nižší úrovni je vhodné rozšířit vymezující oblast R přidáním $\delta \in \mathbb{N}$ buněk vedle každého prvku R ve všech směrech. Při nastavení dostatečně velké hodnoty δ má cesta $p_{R_\delta}^*$ vypočtená v této oblasti shodný průběh s cestou p, kterou lze získat aplikací jednoúrovňové DTW [41].

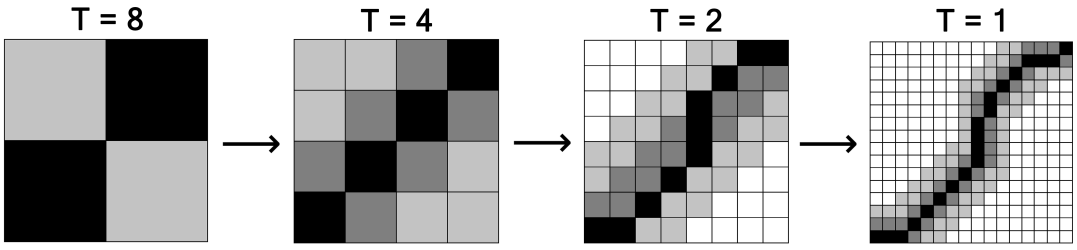


Obr. 3.4: Víceúrovňové DTW – úroveň 2 a optimální cesta p_2^* (vlevo), úroveň 1 a optimální cesta p_R^* náležící vymezené oblasti R (uprostřed), úroveň 1 a optimální cesta $p_{R\delta}^*$ náležící vymezené oblasti R rozšířené parametrem δ (vpravo)

3.5 FastDTW

Metoda FastDTW využívá princip víceúrovňového DTW, přičemž nejprve se vypočte optimální cesta v nejnižším možném rozlišení a přesnost výpočtu se postupnou rekurzí algoritmu zvyšuje do té doby, dokud nemá získaná optimální cesta rozlišení ekvivalentní vstupním signálům X a Y (úroveň 1, viz obr. 3.5) .Prvním krokem algoritmu je podvzorkování signálů činitelem $T = 2 \cdot n, n \in \mathbb{N}$, při kterém je vždy ze dvou sousedních prvků získán jeden prvek průměrováním hodnot. Proces se opakuje několikrát pro různé hodnoty n, čímž se získají signály o různých rozlišeních. Činitel podvzorkování pro každou nejbližší vyšší úroveň je dvojnásobný. Dle zjištěného průběhu optimální cesty na vyšší úrovni se vždy vymezí oblast pro výpočet cesty na úrovni nižší – jeden prvek matice o vyšší úrovni se tedy promítne do minimálně čtyřech prvků matice nižší úrovně (jejich počet může být odlišný, pokud nejsou velikosti vstupních signálů shodné). Počet prvků takto vzniklé matice je v porovnání s maticí odpovídající předchozí úrovní čtyřnásobný. Obdobně jako u Multiscale DTW je i zde zaveden parametr r (radius), pomocí kterého lze vymezenou oblast pro výpočet cesty na další úrovni rozšířit o určitý počet prvků ve všech směrech, čímž dojde ke snížení rizika špatně vyhodnoceného průběhu optimální cesty kvůli ztrátám informací vzniklých během procesu podvzorkování.

Výhoda metody FastDTW spočívá v její lineární výpočetní náročnosti $O(N)$, která je daná tím, že šířka vymezené oblasti je ekvivalentní na všech úrovních výpočtu. Počet prvků matice, pro které algoritmus počítá vzdálenostní funkci (viz vzorec 3.6), se tedy s rostoucí délkou zpracovávaných signálů zvětšuje lineárně, nikoli exponenciálně, jako v případě klasické DTW [42].



Obr. 3.5: FastDTW – výpočet optimální cesty p na jednotlivých úrovních

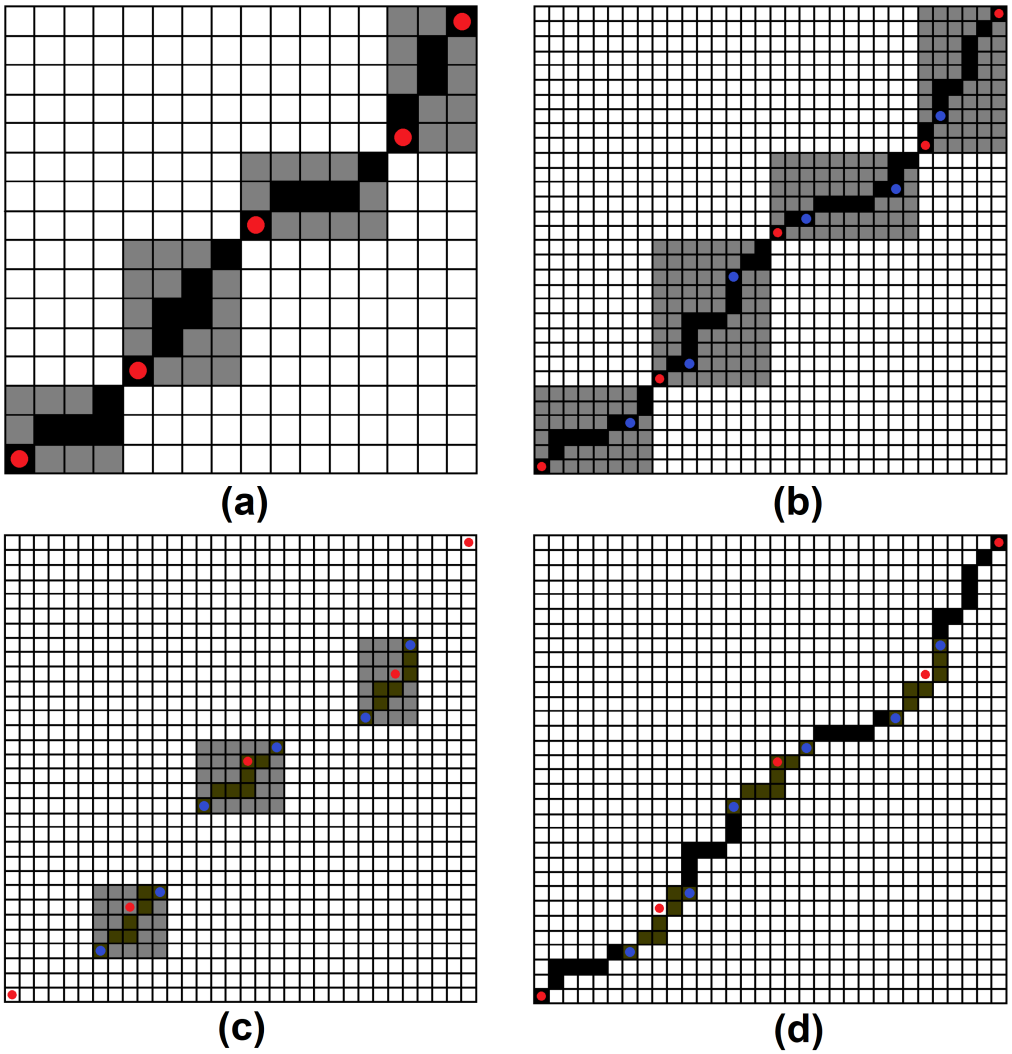
3.6 Memory-restricted multiscale DTW

Memory-restricted multiscale DTW (zkráceně MrMsDTW, viz obr. 3.6) je metoda kombinující principy Multiscale DTW s dynamickým vymezením oblasti pro výpočet optimální cesty [43]. Narozdíl od FastDTW či MsDTW, u kterých vymezená oblast kopíruje průběh optimální cesty a má trubkovitý tvar, je v tomto případě matice D rozdělena na několik dílčích oblastí čtyřúhelníkového tvaru, pro které se výpočet cesty provádí lokálně a na sobě navzájem nezávisle. Maximální počet prvků, které mohou těmto dílčím oblastem náležet, je definován hodnotou parametru τ.

Prvním krokem algoritmu je podvzorkování vstupních signálů a následný výpočet cesty p_2^* náležící úrovni 2, pomocí klasické DTW. Oblast výpočtu cesty na této úrovni tedy není vymezená, dochází zde k vzájemnému porovnávání každé dvojice prvků podvzorkovaných vstupních signálů. Na základě takto vzniklé cesty je dále definovaná sekvence tzv. kotevních bodů $A = \{a_1, ..., a_K\}$ (viz červené body na obr. 3.6). Pomocí každé po sobě jdoucí dvojice kotevních bodů jsou určeny lokální dílčí oblasti pro výpočet cesty na úrovni 1. Počet kotevních bodů a jejich pozice jsou stanoveny rekurzivním dělením cesty p_2^* do té doby, dokud není splněna podmínka, že počet prvků náležící jednotlivým dílčím oblastem může být roven maximálně τ. Tyto oblasti jsou následně promítnuty do matice náležící úrovni 1 a pomocí klasické DTW je pro každou oblast vypočtena lokální optimální cesta $p_{L_k}^*$, kde $k \in \{1, 2, ..., K-1\}$. Vzhledem k tomu, že začátky a konce vymezených oblastí jsou definovány pozicemi kotevních bodů, musí i dílčí cesty $p_{L_k}^*$ kotevními body procházet. Optimální cesta, kterou by bylo možné vypočítat spojením takto získaných cest, by z tohoto důvodu nemusela být zcela přesná, neboť kotevní body jsou určeny na základě cesty vypočtené z podvzorkovaných signálů. Z důvodu zamezení vzniku těchto chyb se dále definuje sekvence kotevních bodů $A' = \{a'_1, ..., a'_{K-2}\}$ (viz modré body na obr. 3.6). Pozice těchto bodů se nastaví takovým způsobem, aby vymezujícím oblastem, které tyto body nově definují, náležely původní kotevní body a_k. I v tomto případě musí být splněna podmínka omezující počet jejich prvků

dle parametru τ. V dalším kroku se vypočtou lokální cesty p'^*_{Lk} náležící oblastem, které jsou dané po sobě jdoucími dvojicemi bodů a'_k. Výsledná optimální cesta p_1 se získá spojením dílčích cest p^*_{Lk} s cestami p'^*_{Lk}.

V případě potřeby nalezení optimální cesty mezi signály o velkém počtu prvků je možné výpočet algoritmu provádět na více než dvou úrovních. Velká výhoda metody spočívá v její konstantní výpočetní náročnosti $O(\tau)$, neboť počet prvků, pro které se vzdálenostní funkce počítá, je definovaný parametrem τ a není závislý na velikosti vstupních vektorů. Přesnost metody je závislá na nastavené hodnotě tohoto parametru.



Obr. 3.6: Memory-restricted multiscale DTW – cesta p^*_2 a sekvence kotevních bodů A (a), lokální cesty p^*_{Lk} se sekvencemi bodů A a A' (b), lokální cesty p'^*_{Lk} náležící oblastem definovaným body A' (c), výsledná cesta p_1 získaná spojením dílčích cest p^*_{Lk} a p'^*_{Lk} (d)

4 Image hashing

S postupně rostoucí popularitou digitálních technologií roste i počet vytvářených a ukládaných obrazových souborů [44]. V případě velké databáze takových souborů vzniká problém, pokud je potřeba vyhodnotit, zda-li se v databázi nevyskytují některé obrázky vícekrát. Základní překážku je zde velký datový objem jednotlivých souborů, kvůli kterému by vzájemné algoritmické porovnávání obrazových dat bez použití redukčních technik zapříčiňovalo vysokou výpočetní náročnost. V případě databáze o počtu milionů obrazových souborů by bylo použití takového systému silně neefektivní. Dalším požadavkem je robustnost algoritmu vůči drobným rozdílům mezi obrazovými soubory – pokud jsou například dvě porovnávané fotografie shodné, ale na jedné z nich je upravená úroveň jasu, vložený vodoznak, zmenšené rozlišení či transformovaný poměr stran. V takovém případě by měl algoritmus umět určit míru podobnosti, na základě které posléze vyhodnotí, zda se jedná o shodu či nikoli. K těmto účelům slouží algoritmy nazývané jako *image hashing*, které pomocí hashovacích funkcí redukují obrazová data na řetězce o předem definované velikosti, tzv. hashe. V případě, že jsou si dva vstupní obrazové soubory podobné, měly by být podobné i hashe, které jsou z těchto souborů extrahované.

4.1 Perceptual hashing

V případě systému navrženém a zrealizovaném v praktické části této práce je využit algoritmus nazývaný jako *Perceptual Hashing* (zkráceně *pHash*) využívající DCT [45]. Jednotlivé kroky tohoto algoritmu jsou popsány níže.

1. **Redukce velikosti** – Prvním krokem algoritmu je redukce velikosti obrázku na konstantní poměr 32×32 pixelů. Tím je zaručeno, že hashe dvou shodných obrázků si budou odpovídat bez ohledu na jejich rozlišení či poměr stran.

2. **Redukce barevného obsahu** – Dalším krokem je převedení barevné informace jednotlivých pixelů do stupňů šedi. Tím dochází k další redukci datového objemu, neboť namísto použití hodnot RGB k vyjádření barvy každého pixelu je každý pixel vyjádřený pouze jednou hodnotou popisující úroveň jasu.

3. **Výpočet DCT** – V tomto kroce se na získaná černobílá data aplikuje DCT. Výstupem je dvourozměrná matice o velikosti 32×32 vyjadřující frekvenční obsah vstupního obrázku.

4. **Redukce velikosti DCT matice** – Z DCT matice o velikosti 32×32 se zachová pouze 8×8 hodnot nacházejících se v levém horním rohu. Tyto hodnoty reprezentují nejnižší frekvence obsažené v obrazovém souboru.

5. **Výpočet průměrné hodnoty** – Pátým krokem je výpočet průměrné hodnoty buněk 8×8 DCT matice. Do výpočtu průměru se nezahrnuje hodnota první buňky, která reprezentuje stejnosměrnou složku obrazového signálu, neboť tato hodnota se může podstatně lišit od hodnot ostatních buněk a mohla by negativně ovlivnit výsledek výpočtu. Tím je zároveň zaručena i robustnost algoritmu vůči rozdílným úrovním jasu u jinak zcela shodných obrázků.

6. **Redukce hodnot DCT matice** – Předposledním krokem je nastavení hodnoty každé buňky DCT matice na hodnotu 0 nebo 1 podle toho, jestli je původní hodnota dané buňky nad nebo pod průměrnou hodnotou vypočítanou v předešlém kroce. Tímto způsobem jsou spektrální informace o vstupních obrazových datech redukovány na velmi hrubý popis jejich relativního zastoupení vůči průměru. Výstupem je vektor 64 bitů.

7. **Konstrukce řetězce hashe** – Posledním krokem je převedení získaného vektoru o velikosti 64 bitů na 64bitový `int`.

4.2 Vyhodnocení podobnosti dvou hashů

Podobnost dvou hashů vypočítaných z obrazových souborů se vyhodnocuje pomocí *Hammingovy vzdálenosti* [46]. Na každé bitové pozici hashů se porovnávají jejich hodnoty a výstupem je celočíselná hodnota udávající počet rozdílných bitů. Obrázky jsou následně vyhodnoceny za shodné, pokud je rozdíl jejich hashů menší než předem definovaná prahová hodnota.

5 Algoritmy používané k vyhledávání duplicitních nahrávek

5.1 Identifikace duplikátů na základě metadat a vlastností souborů

Jedním ze základních způsobů identifikace duplicitních souborů v rámci hudební databáze je jejich vyhodnocení na základě metadat. V současnosti existuje několik programů [47, 48], které na tomto principu fungují. Metadata jsou informace uložené v hudebním souboru společně se zvukovou informací. Bývají v nich zapsané údaje o názvu interpreta, názvu skladby, data jejího vydání, názvu alba, ze kterého skladba pochází či informace o žánru. K těmto účelům lze však použít i některé vlastnosti samotných souborů, jako je délka trvání stopy či její datový objem. U různých formátů mohou být metadata zapsané různými způsoby, v rámci formátu `.mp3` se pro jejich zápis používá kontejner `ID3 tag`, pro formát `.ogg` je standardizovaný `Vorbis comment`.

Při použití tohoto typu dat systém u testované dvojice nahrávek vyhodnocuje míru shody textových řetězců informací, které jsou v metadatech zapsané. Pokud je podobnost dostatečně velká, hudební soubory jsou označeny za shodné. Výsledky získané touto metodou mohou být ovšem velmi nepřesné, neboť samotná analýza není postavena na zvukovém obsahu souborů. Problémem je zde zejména fakt, že u některých nahrávek mohou metadata zcela chybět, či informace v nich obsažené mohou být nepřesné či nepravdivé. V takovém případě dojde u dvojice hudebních souborů k jejich chybnému vyhodnocení.

5.2 Acoustic fingerprinting

V roce 2000 byla založena společnost *Shazam Entertainment, Ltd.* přicházející s nápadem vytvoření služby umožňující identifikaci hudby pomocí mobilního telefonu [49, 50]. Požadavky na vyvíjený algoritmus byly poměrně vysoké – metoda by měla umět rozpoznat skladbu velmi rychle, navíc musí být dostatečně robustní vůči okolním ruchům, dozvukům prostoru či vůči artefaktům vzniklých v rámci digitalizace či síťového přenosu zvukového signálu – např. zkreslení způsobené ztrátovou kompresí či výpadky sítě. Algoritmus navíc musí být velmi přesný, neboť nahrávku je nutné identifikovat v rámci databáze o velikosti několika milionů skladeb.

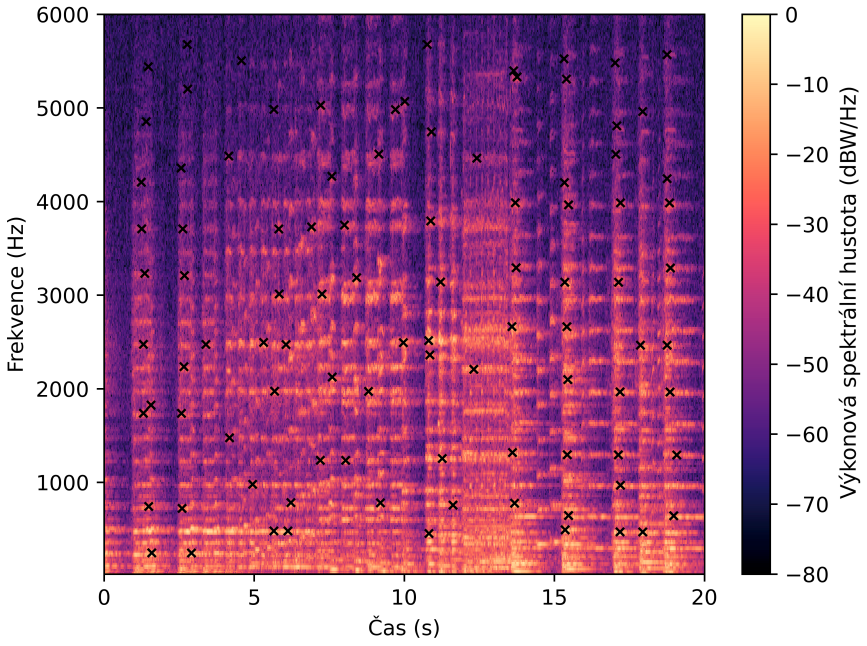
Vyvinutá metoda nazvaná jako *acoustic fingerprinting* extrahuje z každého hudebního souboru tzv. „otisk“ – unikátní informaci, která daný zvukový soubor popisuje. Během procesu identifikace nahrávky je otisk analyzované skladby porovnáván s otisky nahrávek v databázi. Databáze otisků byla v minulosti vytvořena z velkého množství nahrávek, kdy byl otisk každé nahrávky získán použitím shodné analýzy. Rozpoznávaná skladba je identifikovaná na základě největší shody jejího otisku s otiskem jedné z nahrávek z databáze.

Během procesu získávání otisku je ze zpracovávané skladby nejprve vypočten spektrogram. V dalším kroku se vypočítají pozice lokálních maxim výkonové spektrální hustoty spektrogramu v oblastech o definované velikosti (viz obr. 5.1). Předpoklad spočívá v tom, že v těchto bodech je vstupní zvukový signál nejintenzivnější, čímž je redukováno riziko zamaskování těchto oblastí okolními ruchy. Pozice takto získaných bodů lze nyní vyjádřit pouze jejich souřadnicemi na frekvenční a časové ose. Pro zápis těchto informací do databáze a zefektivnění jejich vyhledávání se ze série bodů některé body zadefinují jako tzv. „kotevní body“ společně s jejich referenčními zónami (anglicky anchor points, target zones, viz obr. 5.2). Pro každý bod náležící referenční zóně se vypočítá hodnota hašovací funkce dle vzorce:

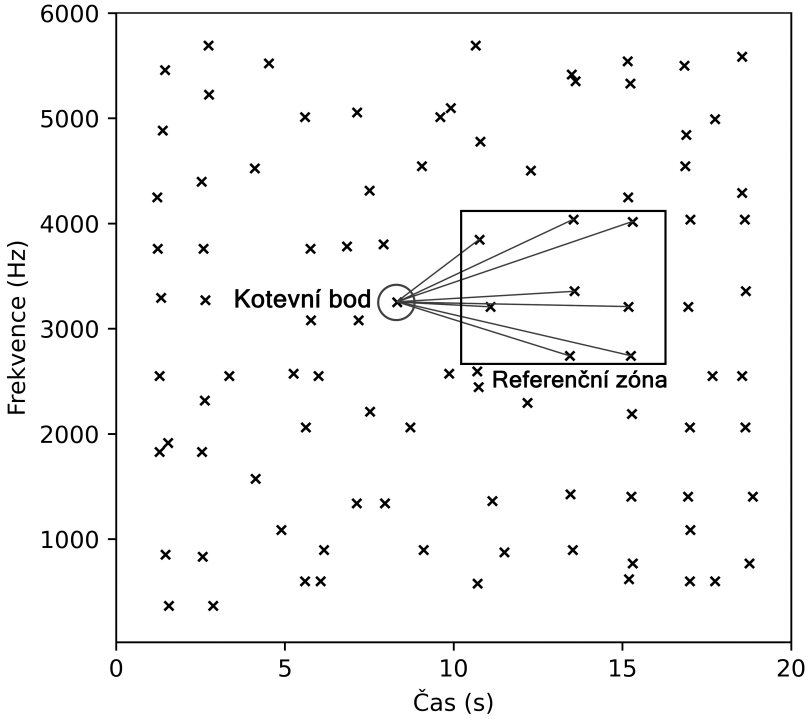
$$h = \{f_1 + f_2 + (t_2 - t_1)\} + t_1, \tag{5.1}$$

kde f_1 a t_1 je frekvence a čas kotevního bodu a f_2 s t_2 odpovídá frekvenci a času bodu náležícímu referenční zóně. Takto získané hodnoty hašovací funkce jsou vyjádřeny 32 bity a uloženy do databáze společně s identifikátorem skladby a hodnotou vyjadřující pozici kotevního bodu na časové ose vůči začátku dané nahrávky, pro které je vymezeno dalších 32 bitů. Během procesu rozpoznávání skladby je každá takováto struktura porovnávaná s daty v databázi za účelem nalezení odpovídající hašovací hodnoty. K tomu je vytvořen tzv. korelační graf, do kterého je v případě nalezení shody zakreslen bod, jehož pozice na ose x určuje časovou pozici $th2$ daného hašovacího bodu ve skladbě z databáze, zatímco pozice na ose y definuje jeho časovou pozici $th1$ v rámci analyzované nahrávky. Pokud dojde ke spárování mnoha

bodů ze stejné nahrávky, tvar posloupnosti bodů na korelačním diagramu začne připomínat diagonální úsečku, neboť míra výskytu lokálních maxim výkonové spektrální hustoty v čase, ze kterých jsou hašovací hodnoty vypočteny, bude identická u analyzované skladby i u korespondující skladby z databáze. Posledním krokem je vytvoření histogramu, do kterého jsou zakresleny hodnoty $dth = th2 - th1$ jednotlivých korespondujících hašovacích bodů. Pokud jsou tedy porovnávané nahrávky shodné, výsledný histogram bude obsahovat velké množství stejných hodnot dth, neboť jednotlivé hodnoty dth vyjadřují rozdíl časové pozice hašovacího bodu v rámci rozpoznávané nahrávky vůči časové pozici korespondujícího bodu nahrávky z databáze. Vlivem shodné frekvence výskytu těchto bodů v obou nahrávkách se na histogramu v dané pozici vytvoří vrchol, na základě kterého je míra shody dvou porovnávaných nahrávek vyhodnocena.



Obr. 5.1: Spektrogram s body vyjadřujícími pozice lokálních maxim výkonové spektrální hustoty



Obr. 5.2: Kotevní bod a referenční zóna definovaná na základě pozic lokálních maxim výkonové spektrální hustoty spektrogramu

5.3 Audio shingling

Zatímco metoda *acoustic fingerprinting* popsaná v předešlé kapitole byla vyvinuta k rozpoznání a vzájemnému přiřazení zcela shodných nahrávek či jejich pasáží (nebereme-li v potaz rozdíly vzniklé zkreslením v důsledku pořízení nahrávky mobilním telefonem a následným síťovým přenosem), metodu *audio shingles* je vhodné použít v případech, kdy dvě testované nahrávky shodné nejsou a je třeba určit míru podobnosti mezi nimi. Algoritmus se tedy používá k vyhledávání různých verzí dané nahrávky či skladeb, které jsou z ní odvozené – může se jednat například o cover verze[1], remixy[2], různé formáty té stejné skladby (např. radio edit[3] nebo club edit[4]) či verze přezpívané v jiném jazyce. Hudební obsah těchto jednotlivých verzí tedy

[1]cover verze: nová verze stejné skladby, obvykle přezpívaná jiným interpretem

[2]remix: verze skladby, ve které je použit klíčový motiv originálního díla obohacený o zcela nové prvky; typické je remixování tanečních skladeb do odlišných žánrů

[3]radio edit: zkrácená verze skladby pro vysílání v rádiu

[4]club edit: prodloužená verze skladby umožňující její plynulý přechod do jiné skladby během míchání diskžokejem

není zcela identický, některé jeho aspekty (např. melodie vokálů či basová linka) však shodné či velmi podobné jsou [51].

Následující popis metody vychází z publikací [52, 53], ve kterých jsou audio shingles extrahovány ze dvou různých parametrů nahrávek – konkrétně se jedná o vektory LFCC a Chroma parametrů, které jsou vypočteny s časovým oknem 100 ms. Parametr LFCC popisuje tvar frekvenčního spektra, čímž se získá informace o témbru daného úseku nahrávky, zatímco Chroma parametr popisuje tónový obsah daného úseku. Vytvoření shingles je docíleno spojením třiceti po sobě jdoucích vektorů těchto parametrů o časových oknech 100 ms, čímž jsou získány shingles o délkách 3 sekund. To vychází z poznatku, že tato délka je z hlediska přesnosti vyhodnocování optimální [52]. V případě Chroma shingles se tímto tedy získá posloupnost vícerozměrných vektorů o počtech prvků $10\ \text{Hz} \cdot 3\ \text{s} \cdot 12$ rozměrů $= 360$, zatímco v případě LFCC shingles budou mít vícerozměrné vektory $10\ \text{Hz} \cdot 3\ \text{s} \cdot 20$ rozměrů $= 600$ prvků. Aby shoda mezi dvěma porovnávanými skladbami nebyla určena na základě počátečních či koncových pasážích skladeb obsahujících pouze ticho či šum, odstraní se zvukové úseky, jejichž efektivní hodnota je menší než čtvrtina průměrné efektivní hodnoty shingles. Energie zbylých shingles je normalizována, aby se eliminoval vliv rozdílných hlasitostí analyzovaných skladeb na jejich vyhodnocení, neboť různé verze stejné skladby mohou být vlivem odlišného zpracování rozdílně hlasité.

Dalším krokem algoritmu je výpočet vzdáleností mezi jednotlivými shingles porovnávaných nahrávek, na základě kterých se vyhodnotí jejich podobnost. Definujme tedy $\{Q\}$ jako M-rozměrnou řadu shingles náležících analyzované nahrávce a $\{A^{(n)}\}$ jako řadu shingles porovnávané skladby n z databáze o počtu N skladeb. Jednotlivé shingles náležící těmto řadám budou označeny jako $q_i \in \{Q\}$ a ${a_j}^{(n)} \in \{A^{(n)}\}$. Vzdálenost mezi shingles se vypočte dle euklidovské metriky:

$$x = d^2(q_i, a_j^{(n)}) = 2 - 2\sum_{m=1}^{M} q_{im} a_{jm}^{(n)}, \tag{5.2}$$

na základě které se podobnost testovaných nahrávek vyhodnotí dle počtu shingles $q_i \in \{Q\}$, jejichž vzdálenost vůči ${a_j}^{(n)}$ je nižší než předem definovaná prahová hodnota x_{tresh}. K tomu je použita charakteristická funkce:

$$I_i(q_i, A^{(n)}) = \begin{cases} 1, & \text{pokud } \exists\ j, \text{ pro které } d^2(q_i, a_j^{(n)}) \leq x_{tresh} \text{ pro } i \in \{1...|Q|\}, \\ 0, & \text{v opačném případě.} \end{cases} \tag{5.3}$$

ve které $|Q|$ vyjadřuje počet prvků řady $\{Q\}$.

Posledním krokem je výpočet parametru $C(Q, A^{(k)})$ vyjadřujícího počet shingles, které mezi sebou dle rovnice 4.3 vykazují dostatečně velkou shodu. K tomu je použit vztah:

$$C(Q, A^{(k)}) = \sum_i I_i(q_i, A^{(n)}). \tag{5.4}$$

Parametr $C(Q, A^{(k)})$ tedy může nabývat hodnot v rozsahu $\langle 0, |Q| \rangle$, přičemž nejvyšší shodu vykazuje analyzovaná nahrávka právě s tou skladbou z databáze, pro kterou je hodnota tohoto parametru nejvyšší. Klíčová je zde zejména hodnota parametru x_{tresh} – pokud je příliš nízká, počet rozpoznaných podobných nahrávek bude malý, v opačném případě systém jako podobné nahrávky označí i nahrávky odlišné.

Výsledky testování z [51] poukazují na to, že použitím Chroma parametrů pro tvorbu shingles vykazuje výsledná metoda vyšší přesnost než v případě použití LFCC parametrů, nejpřesnější je však použití váhované kombinace vzdáleností z těchto parametrů vypočtených.

6 Implementace systému

V předchozí kapitole bylo popsáno několik metod používaných k nalezení shody mezi dvěma audio nahrávkami. Základním způsobem je určení podobnosti dle informací uložených v metadatech souborů, problémem je ale fakt, že informace v nich obsažené jsou často nepřesné či chybějící. Metoda *acoustic fingerprinting* slouží k rychlé identifikaci skladby pomocí tzv. otisku. Vzhledem k tomu, že míra shody je v jejím případě vyhodnocena na základě krátkého nahraného úseku rozpoznávané skladby, metoda k sobě může přiřadit nahrávky, u kterých se shodují jen některé části. Poslední uvedenou metodou v předchozí kapitole je *audio shingling*, která namísto přesné shody mezi dvěma nahrávkami určuje míru jejich podobnosti. Metodu je tedy vhodné použít spíše k identifikaci cover verzí či remixů, nikoli k rozpoznání duplikátů.

Cílem praktické části této práce je vytvoření systému, který v rámci databáze hudebních souborů dokáže identifikovat duplikáty, jejichž hudební obsah je zcela shodný. K těmto účelům byl použit programovací jazyk `Python`. Problémem je fakt, že soubory se shodným hudebním obsahem nemusí být vždy zcela stejné – rozdíly mezi nimi mohou být v hlasitosti, v úrovni obsaženého šumu či zvukové kvalitě – jeden soubor může být například v CD kvalitě, zatímco nahrávka ve druhém souboru může být zkomprimovaná ztrátovým kodekem jako je `mp3` či `vorbis`. Odlišnosti mohou vzniknout i v případě, kdy jedna nahrávka pochází z CD, zatímco druhá nahrávka je zaznamenaná z gramofonové desky. Rozdíly mezi těmito nahrávkami budou způsobené zkreslením zvuku jeho záznamem na vinylovou desku, zvukový charakter může být závislý např. i na typu použité přenosky. Nejen v tomto případě se také může stát, že délky nahrávek nebudou zcela stejné; např. při digitalizaci skladeb z vinylu je běžné, že ve výsledných souborech se na začátcích či koncích vyskytuje pouze ticho či šum. Obdobným rozdílem mezi duplikáty může být i potlesk obsažený na konci jedné z nahrávek. Navržený systém musí být vůči těmto rozdílům dostatečně robustní, zároveň ale musí být natolik přesný, aby shodné hudební nahrávky skutečně identifikoval a nedocházelo k chybnému přiřazení nahrávek, které duplicitní nejsou. To platí zejména v případě aplikace systému na databázi obsahující různé interpretace té stejné skladby, neboť přesto, že takové nahrávky jsou si velmi podobné, o hudební duplikáty se nejedná.

Výsledný zdrojový kód systému obsahující funkce, které budou v následujících podkapitolách uvedené, je odevzdaný v příloze práce. Zároveň je možné ho stáhnout z repozitáře na Githubu [54].

6.1 Použité knihovny, balíčky a moduly

FFmpeg [55] je open-source software umožňující kódování, dekódování, streamování, zpracovávání a přehrávání i nahrávání digitálního multimediálního obsahu v různých kodecích a souborových formátech. V rámci implementovaného systému je FFmpeg použit pro načítání vstupních audio souborů, neboť samotná rychlost načítání je vyšší, než v případě použití funkcí z jiných Python knihoven. Další výhodou je již zmíněná podpora velkého množství audio formátů společně se snadnou implementací jejich dekódování.

Numpy [56] je open-source knihovna určená pro realizaci matematických výpočtů a operací. Knihovna je použita z toho důvodu, že zahrnuje objekty pro vektory a matice různých rozměrů, společně s implementovanými a optimalizovanými funkcemi pro práci s těmito datovými typy a strukturami.

Scipy [57] je open-source knihovna zaměřená na technické výpočty. Poskytuje např. algoritmy pro optimalizaci, integrování, derivování, interpolaci či výpočty algebraických i diferenciálních rovnic. Knihovna zapouzdřuje optimalizované funkce implementované v nízkoúrovňových jazycích, jako je `C`, `Fortran` či `C++`. Díky tomu disponuje nízkými výpočetními nároky v kombinaci s jednoduchou syntaxí.

Matplotlib [58] je knihovna umožňující grafické vykreslování průběhů funkcí a datových řad či vytváření různých typů grafů. Vygenerované diagramy mohou být nejen statické, ale i animované. Další výhodou je, že mohou být interaktivní – lze je přibližovat, zmenšovat, odečítat z nich funkční hodnoty apod. Knihovna dále poskytuje širokou škálu přizpůsobení vygenerovaných figur společně s možnostmi exportu do různých souborových formátů. Použita byla z důvodu možnosti vizualizace výstupu systému.

Librosa [59] je balíček obsahující funkce pro analýzu a zpracování zvukových souborů včetně řady funkcí implementující metody z oblasti Music Information Retrieval (viz kapitola 2). Knihovna obsahuje např. funkce pro načítání a převzorkování audio souborů, výpočet chromagramů či různých spektrálních transformací. Balíček je použitý pro výpočet chromagramů.

Dtw [60] je open-source implementace klasické DTW metody pro Python, kterou lze ve výsledném systému zvolit jako jednu z variant DTW pro výpočet optimální cesty.

Fastdtw [61] je balíček pod MIT licencí implementující FastDTW metodu v Pythonu . FastDTW je ve výsledném systému možné použít jako jednu z variant DTW pro výpočet optimální cesty.

Sync Toolbox [62] je balíček funkcí umožňujících robustní, přesnou a výpočetně efektivní synchronizaci audio nahrávek. Balíček obsahuje například funkce pro

výpočet chromagramů či DTW, použit byl však kvůli jeho implementaci metody MrMsDTW, které je využito v rámci systému pro nalezení duplikátů nahrávek.

PyTorch [63] je open-source knihovna implementující strojové učení na bázi tenzorů a hlubokého učení (anglicky deep learning). Knihovna umožňuje provádění výpočtů procesorem i jejich hardwarovou akceleraci grafickou kartou s použitím technologie `Nvidia CUDA®`[1] [64]. Použita je pro zrychlení výpočtu CQT z audio souborů za použití neuronových sítí. Vektor CQT je následně předán funkci pro výpočet chromagramů z knihovny Librosa.

nnAudio [65] je sada nástrojů poskytující funkce zvukového zpracování ve spektrální oblasti. Balíček využívá konvoluční neuronové sítě knihovny PyTorch, díky kterým umožňuje rychlý výpočet spektrogramů na bázi strojového učení s použitím hardwarové akcelerace grafickou kartou. Použita je funkce pro výpočet CQT.

Csv [66] je modul obsahující třídy funkcí, které implementují snadný zápis a čtení do, resp. ze souborového formátu `.csv` (CSV je zkratkou pro Comma Separated Values, v překladu hodnoty oddělené čárkou). Samotný formát se využívá pro jednoduchý zápis tabulek, kde každý řádek definuje jeden řádek tabulky a hodnoty náležící jednotlivým sloupcům jsou od sebe odlišeny pomocí tzv. oddělovače [67]. Jako oddělovač je většinou použit symbol čárky, formát však není plně standardizován, jako oddělovač lze použít např. i středník či mezeru. První řádek souboru často slouží jako hlavička, kde jeho buňky textovým řetězcem definují názvy jednotlivých sloupců. Modul byl použit pro zapisování výstupu systému do souboru.

Openpyxl [68] je knihovna umožňující načítání a zápis dat ze, resp. do souboru formátu `.xlsx`, se kterým pracuje tabulkový procesor Microsoft Excel. Knihovna byla použita pro zapisování výstupu systému do souboru v tomto formátu.

ImageHash [69] je knihovna implementující různé Image Hashing metody, jako je *average hashing*, *difference hashing*, *wavelet hashing* či *perceptual hashing*, která je použita v rámci implementace výsledného systému praktické části této práce.

Pillow [70] je knihovna poskytující řadu funkcí umožňujících práci s digitálními obrazovými soubory a daty. V rámci systému je použita pro ukládání chromagramů ve formátu `.png`.

[1] `Nvidia CUDA®` je softwarová a hardwarová architektura pro realizaci obecných výpočtů s použitím výpočetního výkonu podporovaných grafických karet

6.2 Dataset pro testování

Pro účely testování přesnosti, výpočetní rychlosti a robustnosti systému byl vytvořen základní dataset o počtu patnácti hudebních nahrávek. Z důvodu vytvoření co nejvíce univerzálních testovacích dat bylo použito pět nahrávek elektronické hudby, pět nahrávek populární hudby a pět nahrávek vážné hudby. Všechny tyto nahrávky byly zcela unikátní, tudíž nevyskytovaly se mezi nimi žádné duplikáty. Pro testování robustnosti systému vůči jednotlivým typům rozdílů mezi nahrávkami byly z tohoto základního datasetu vybrány a zkopírovány tři nahrávky, každá z jiné skupiny (elektronická hudba, populární hudba, vážná hudba). Následně bylo vytvořeno několik subsetů, přičemž v každém subsetu byly vybrané duplikáty zpracované odlišným způsobem. Jednotlivé subsety jsou popsané níže.

Subset bez duplikátů je základní verzí datasetu popsaného výše. Neobsahuje žádné duplikáty, pouze patnáct unikátních nahrávek.

Subset obsahující duplikáty je tvořen patnácti unikátními nahrávkami a třemi duplikáty, které jsou zcela identickou kopií původních souborů.

Subset obsahující duplikáty nízké kvality je tvořen patnácti unikátními nahrávkami a třemi duplikáty, které byly vyexportované do formátu `.mp3` s přenosovou rychlostí 64 kbps.

Subset s duplikáty obsahujícími šum na začátku či konci je tvořen patnácti unikátními nahrávkami a třemi duplikáty, přičemž do jednoho duplikátu byly vloženy dvě sekundy šumu před začátek samotné nahrávky, do druhého duplikátu byly vloženy dvě sekundy šumu po skončení nahrávky a do třetího duplikátu byly vloženy dvě sekundy šumu před začátek i po konci nahrávky.

Subset s duplikáty obsahujícími dlouhý šum na začátku je tvořen patnácti unikátními nahrávkami a třemi duplikáty, do kterých byl před začátek samotných nahrávek vložen šum o délce shodné s dobou trvání nahrávek. Jinými slovy, polovinu doby délky trvání těchto duplikátů tvoří šum.

Dataset vytvořený z interpretací jako jediný obsahuje jiné nahrávky, než subsety výše uvedené. Je tvořen osmnácti různými interpretacemi té stejné skladby žánru vážné hudby. Účelem je otestovat robustnost systému vůči nahrávkám, které jsou si velmi podobné, nikoli však shodné. V takovém případě by nemělo dojít k vyhodnocení žádné dvojice systémem za duplicitní.

6.3 Metoda vyhodnocující shodu pomocí DTW

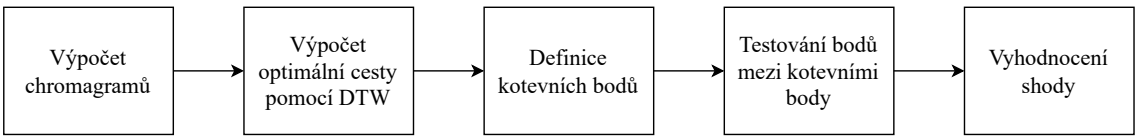
V rámci praktické části práce byly navrženy tři různé metody pro vyhodnocení shody mezi dvěma hudebními soubory. Algoritmus první metody je založen na poznatku, že tvar průběhu optimální cesty, kterou lze získat aplikací DTW na chromagramy dvou duplicitních nahrávek, by měl být rovný. To platí z toho důvodu, že hudební obsah těchto dvou nahrávek je shodný a jejich relativní rychlosti vůči sobě se v čase nemění. Intervaly mezi přiřazenými body těchto nahrávek budou tedy konstantní. V případě, že na začátku či konci jedné (či obou) z nahrávek je ticho, šum či potlesk, optimální cesta bude rovná ve většině jejího průběhu, tj. kromě jejího začátku či konce (viz obr. 6.2).

Metodu lze spustit pomocí funkce `find_duplicates_dtw(...)`. Diagram algoritmu metody je znázorněný na obr. 6.1. První fází systému je postupné načítání jednotlivých hudebních souborů z předem definovaného adresáře včetně všech jeho podsložek. Z důvodu optimalizace výpočetního času jsou hudební soubory načítány (resp. načítány a podvzorkovány) se vzorkovacím kmitočtem $f_{\text{vz}} = 11025$ Hz. Z každého načteného souboru je vypočten chromagram, přičemž samotný proces výpočtu lze rozdělit na dva kroky. Prvním krokem je výpočet CQT spektrogramu pomocí knihovny nnAudio. Ta využívá strojové učení a výpočetní výkon grafické karty za pomocí technologie `CUDA`, díky čemuž je výpočet rychlejší. Z takto vzniklého spektrogramu se pomocí knihovny Librosa vypočte chromagram. V případě, že uživatel nemá k dispozici počítač s grafickou kartou podporující technologii `CUDA` či nainstalovaný potřebný software, systém umožňuje nastavení výpočtu chromagramů pomocí funkcí z balíčku Sync Toolbox [62]. Vypočtený chromagram se následně uloží do složky v předem definovaném výstupním adresáři. Ukládání do souboru se děje z toho důvodu, že velikost operační paměti počítače nemusí být dostatečná, pokud by byl systém použit k analýze velkého množství vstupních souborů. Další výhodou je, že pokud již byl chromagram daného souboru v minulosti uložen, dojde k jeho načtení namísto opětovného výpočtu. Tento proces se opakuje do té doby, dokud nejsou vypočteny a uloženy (příp. načteny) chromagramy náležící všem hudebním souborům ze vstupní hudební databáze.

Dalším krokem po vypočtení chromagramů všech vstupních nahrávek je ověření, které dvojice nahrávek jsou shodné. Systém mezi sebou postupně testuje dvojice chromagramů náležící všem kombinacím audio souborů, přičemž celkový počet těchto kombinací je roven $\frac{n(n-1)}{2}$, kde n vyjadřuje počet vstupních nahrávek. Pro každou testovanou dvojici chromagramů je vypočtena optimální cesta pomocí DTW. Systém umí použít klasickou variantu DTW, FastDTW a MrMsDTW. Jakmile je získán průběh optimální cesty, dalším krokem je definice dvou kotevních bodů náležících optimální cestě. Pozice kotevních bodů jsou definované parametrem

`segmentdivider` – pokud je například hodnota `segmentdivider` = 4, pozice kotevních bodů budou odpovídat 1/4 a 3/4 rozsahu cesty. Pomocí kotevních bodů je následně určena přímka, která těmito body prochází. Na optimální cestu je poté promítnuta série testovacích bodů, jejichž počet je definován parametrem `testpointsnum`. Jednotlivé testovací body jsou mezi kotevními body rovnoměrně rozložené. V případě, že je počet testovacích bodů větší než počet vzorků cesty v určeném úseku, dojde k nastavení parametru `testpointsnum` na hodnotu shodnou s počtem vzorků cesty v daném úseku. Testovací body jsou promítnuty pouze do prostřední části optimální cesty právě z toho důvodu, že okraje cesty nemusí být rovné, pokud některá z nahrávek v těchto místech obsahuje např. šum či potlesk.

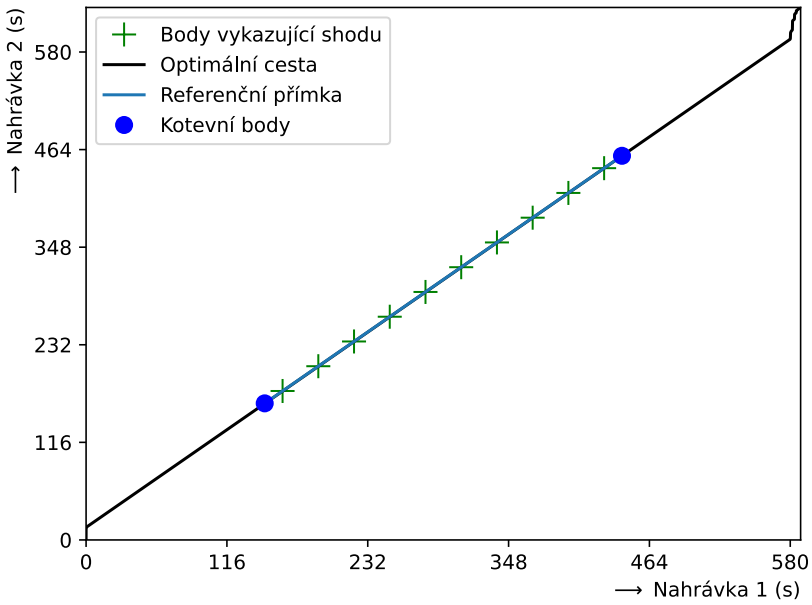
Následně je pro každý testovací bod vyhodnoceno, jestli náleží referenční přímce. Aby daný testovací bod tuto podmínku splnil, jeho vzdálenost od přímky na ose y musí být menší nebo rovna hodnotě 1,2. Hodnota 1 odpovídá rozlišení vstupních chromagramů, respektive velikosti jednoho vzorku optimální cesty, hodnota 1,2 byla zvolena z důvodu vytvoření tolerance vůči zaokrouhlovacím chybám. V případě, že tuto podmínku splňuje počet testovaných bodů percentuálně definovaný pomocí vstupního parametru `diffpointstolerance`, testované nahrávky jsou vyhodnoceny za shodné. Vizualizaci výsledků ověřování shody mezi stejnými i odlišnými nahrávkami za použití této metody lze vidět na obr. 6.2 a 6.3. Z důvodu lepší přehlednosti byl při vytváření vizualizací nastaven počet testovaných bodů `testpointsnum` = 10, výchozí hodnota parametru je však `testpointsnum` = 100.



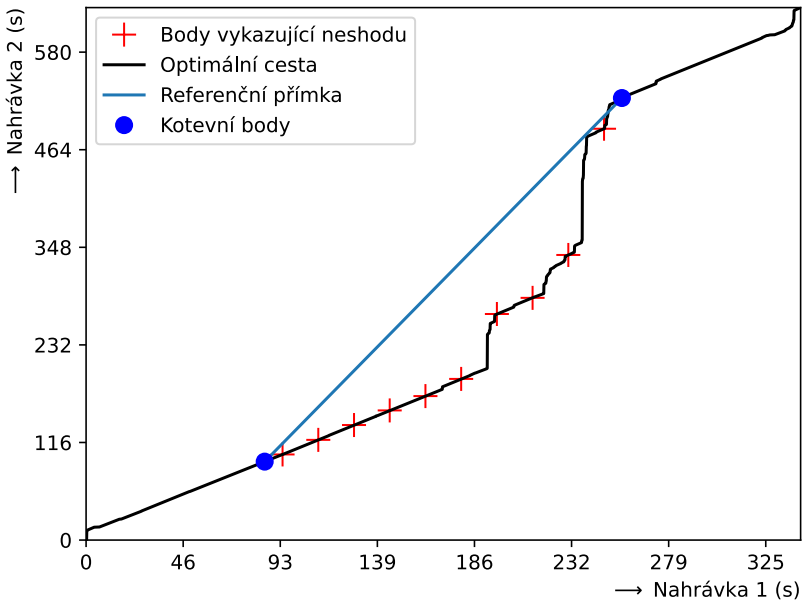
Obr. 6.1: Diagram algoritmu metody vyhodnocující shodu pomocí DTW

Pro případy, kdy uživatel v databázi očekává duplikáty obsahující velmi dlouhé pasáže ticha, šumu či potlesku na jejich začátku či konci, byl do funkce přidán parametr `verify_extremes`. U těchto případů bývá průběh optimální cesty rovný v části náležící prostředku jedné ze dvou os podle toho, v jakém pořadí jsou vůči sobě nahrávky testované (viz obr. 6.4). Pokud systém dvojici nahrávek vyhodnotí za neshodnou, při parametru `verify_extremes` nastaveném na hodnotu `True` systém následně prohodí osy cesty a vyhodnocení rovnosti jejího průběhu proběhne ještě jednou. Tento výsledek vyhodnocení je pak rozhodující. Vzhledem k tomu, že v tomto případě může docházet k výpočtu vyhodnocení každé dvojice nahrávek dvakrát, je vhodné tuto možnost zapínat pouze v případech, kdy uživatel očekává výskyt tohoto typu duplikátů, neboť tím dochází ke zvýšení celkové výpočetní náročnosti systému.

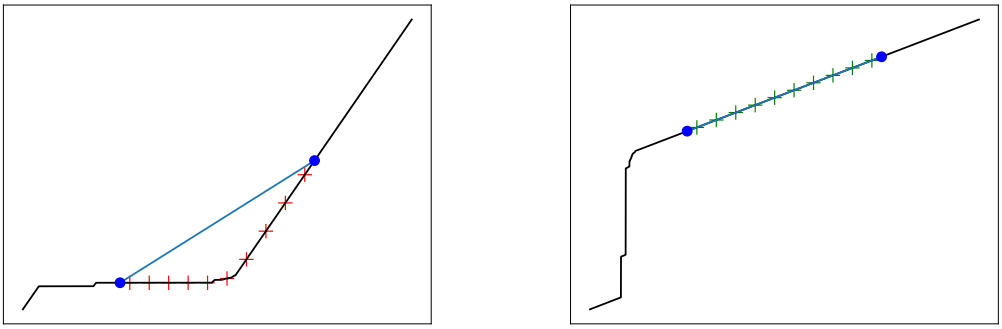
Jakmile systém provede výpočet pro všechny možné kombinace souborů, vzájemně přiřazené dvojice vyhodnocené za shodné jsou následně zapsány do souboru v předem definovaném výstupním adresáři.



Obr. 6.2: Vyhodnocení shodných nahrávek



Obr. 6.3: Vyhodnocení rozdílných nahrávek



(a) První fáze vyhodnocení

(b) Druhá fáze vyhodnocení po prohození os cesty

Obr. 6.4: Vyhodnocení extrémního případu duplikátu obsahujícího velmi dlouhou pasáž ticha pomocí parametru `verify_extremes = True`

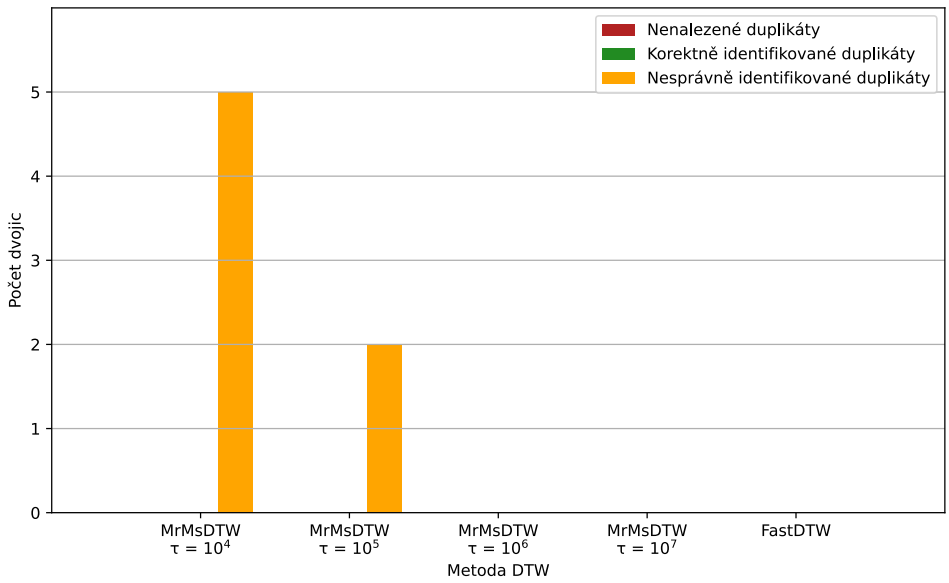
6.3.1 Vyhodnocení metody na datasetu

Metoda byla následně otestovaná na jednotlivých verzích datasetu popsaných v kapitole 6.2. Účelem testování bylo zjistit, jakou variantu algoritmu DTW je vhodné pro výpočet duplikátů použít. Otestované byly varianty metody používající FastDTW a MrMsDTW o různě nastavené hodnotě parametru τ. Vyšší hodnoty τ mohou znamenat lepší přesnost synchronizace, ale s rostoucí hodnotou τ roste i čas výpočtu výsledné optimální cesty. Cílem bylo najít nejlepší kompromis mezi výpočetní dobou a výslednou přesností algoritmu. Z hlediska výpočetního času je dle tab. 6.1 optimální použití algoritmu MrMsDTW. Jedná se o průměrné časy výpočtu všech fází dané metody, tj. výpočet chromagramů i následné vyhodnocení shody pro všechny kombinace nahrávek pomocí jedné z variant DTW.

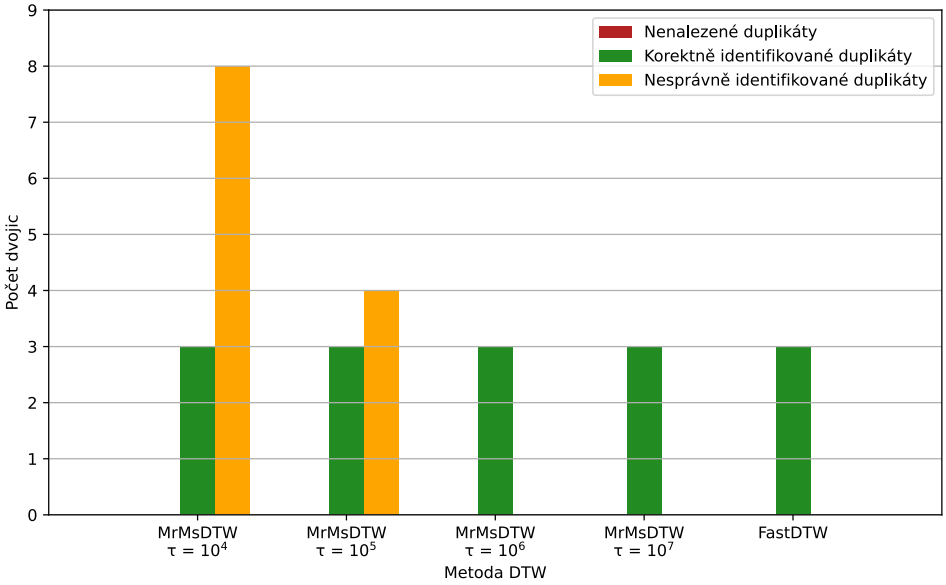
Jak lze vidět na obr. 6.5 až obr. 6.9, při použití MrMsDTW se jako ideální jeví hodnota $\tau = 10^6$. Tato hodnota byla již dostatečně vysoká na to, aby byly u všech subsetů korektně identifikované všechny duplikáty a zároveň nedocházelo k žádnému chybnému vyhodnocení dvojic, které duplicitní nejsou, jako tomu bylo při použití hodnoty $\tau = 10^5$ či $\tau = 10^4$. V případě datasetu neobsahujícího žádné duplikáty, pouze různé interpretace stejné skladby (viz obr. 6.10) je vidět, že k chybnému vyhodnocení nedocházelo ani u jedné z testovaných variant metody. Systém je tedy dostatečně robustní i vůči nahrávkám, které jsou si velmi podobné, nikoli však shodné.

Tab. 6.1: Průměrná doba výpočtu subsetů metodou vyhodnocující shodu pomocí DTW

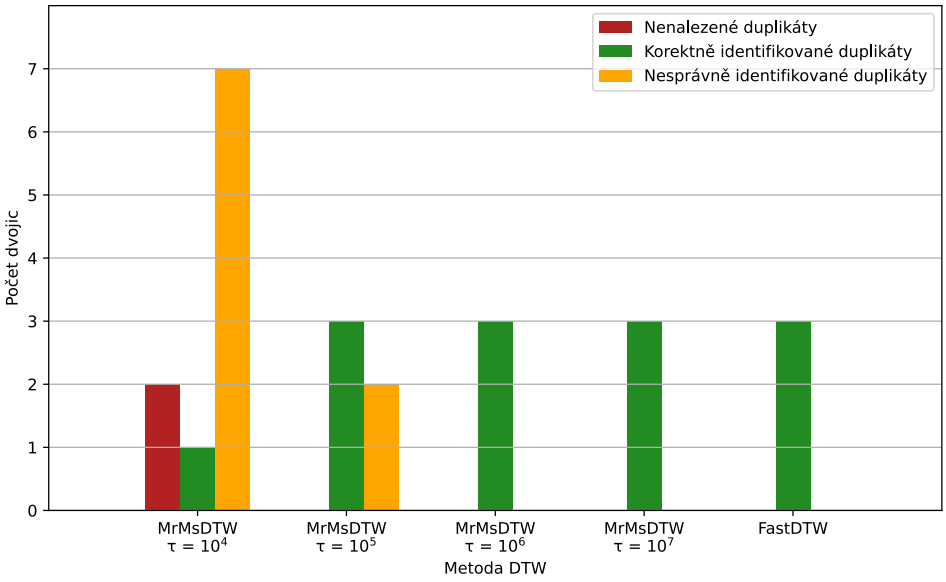
Metoda DTW	MrMsDTW, $\tau = 10^4$	MrMsDTW, $\tau = 10^5$	MrMsDTW, $\tau = 10^6$	MrMsDTW, $\tau = 10^7$	FastDTW
Čas výpočtu (s)	63	94	142	342	1737



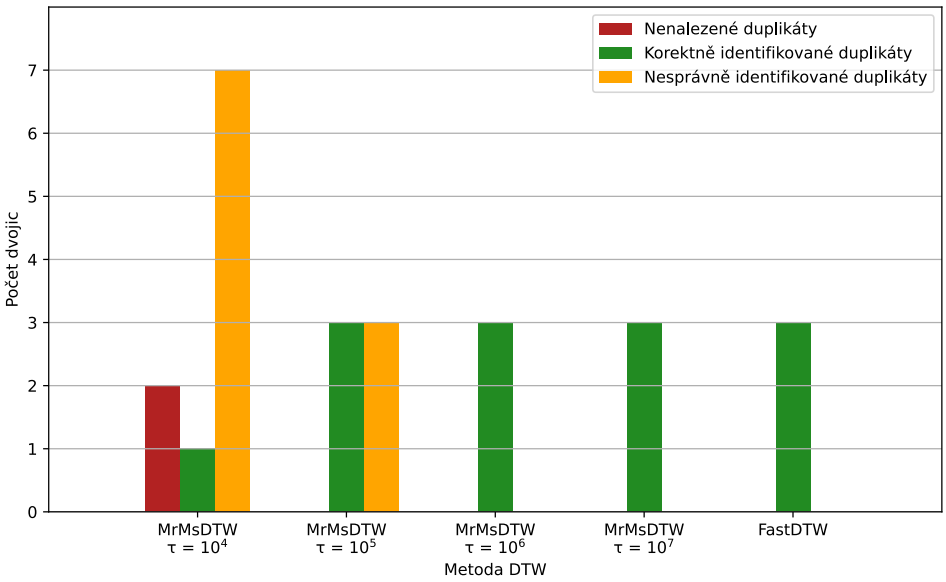
Obr. 6.5: Subset bez duplikátů – výsledky metody vyhodnocující shodu pomocí DTW



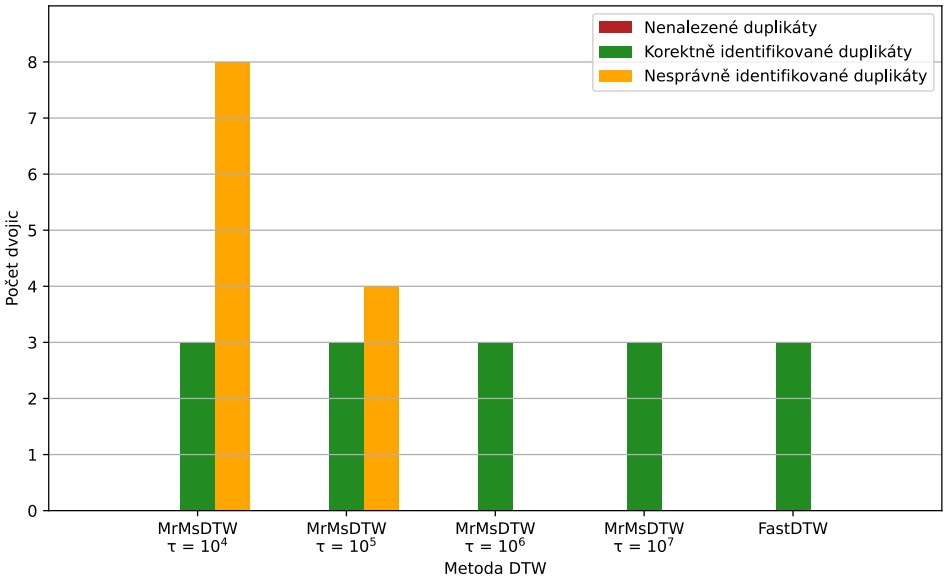
Obr. 6.6: Subset obsahující duplikáty – výsledky metody vyhodnocující shodu pomocí DTW



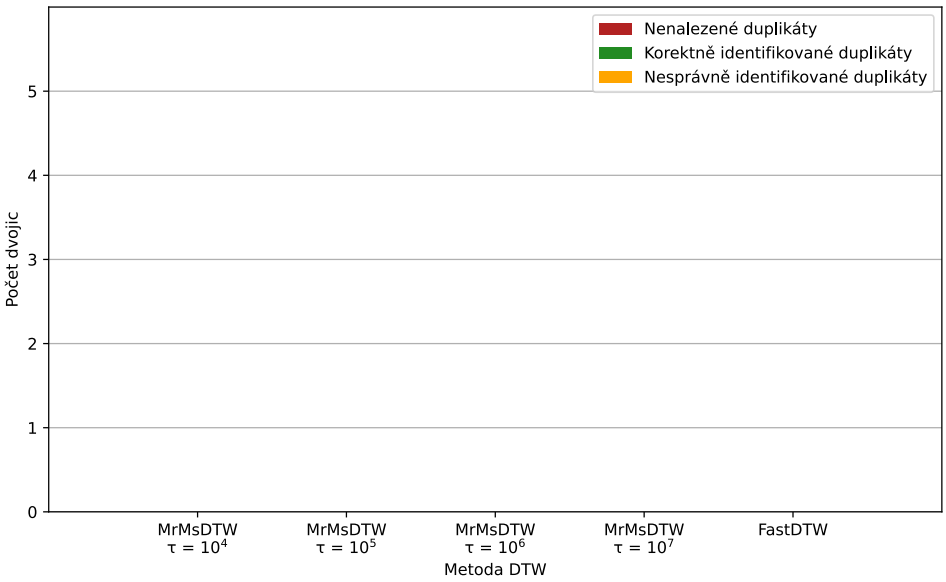
Obr. 6.7: Subset s duplikáty obsahujícími šum na začátku či konci – výsledky metody vyhodnocující shodu pomocí DTW



Obr. 6.8: Subset s duplikáty obsahujícími dlouhý šum na začátku – výsledky metody vyhodnocující shodu pomocí DTW



Obr. 6.9: Subset obsahující duplikáty nízké kvality – výsledky metody vyhodnocující shodu pomocí DTW



Obr. 6.10: Dataset vytvořený z interpretací – výsledky metody vyhodnocující shodu pomocí DTW

6.4 Metoda vyhodnocující shodu pomocí techniky Image Hashing

Algoritmus této metody vychází z myšlenky, že pokud jsou chromagramy dvou duplicitních nahrávek shodné či téměř shodné, po jejich uložení do digitálního obrazového formátu by mělo být možné určit shodu mezi nimi pomocí metody Image Hashing.

Metodu lze spustit pomocí funkce `find_duplicates_img_hashing(...)`. Diagram algoritmu je znázorněný na obr. 6.11. První fází systému je výpočet chromagramů všech vstupních nahrávek, stejně jako u předešlé metody používající techniku DTW. Chromagramy jsou následně vyexportované do obrazového formátu `.png`, ve kterém je každý prvek dvourozměrného vektoru chromagramu vyjádřený jedním pixelem. Hodnota výkonové spektrální hustoty chromagramu v daném bodě je převedena na 8 bitovou hodnotu stupňů šedi v rozsahu $\langle 0, 255 \rangle \subset \mathbb{N}$. Výsledné obrazové soubory jsou tedy černobílé a jejich výška je vždy 12 pixelů, šířka závisí na délce vstupních chromagramů, resp. zvukových souborů.

Jakmile jsou všechny chromagramy uloženy do formátu `.png`, systém následně porovnává každou dvojici pomocí metody Image Hashing (konkrétně typ *p-hash*). Z obrazových dat každé dvojice chromagramů jsou nejprve vypočteny hashe, ze kterých je následně určena jejich podobnost pomocí *Hammingovy vzdálenosti* (viz kapitola 4.2). Pokud je hodnota rozdílu hashů menší nebo rovna předem definované hodnotě parametru `hashdiff_tresh`, nahrávky jsou vyhodnoceny za shodné.

Po provedení výpočtu pro všechny možné kombinace systém zapíše nalezené duplicitní dvojice do souboru, jako tomu bylo u předešlé metody používající techniku DTW.

Obr. 6.11: Diagram algoritmu metody vyhodnocující shodu pomocí techniky Image Hashing

6.4.1 Vyhodnocení metody na datasetu

Stejně jako předešlá metoda používající techniku DTW byla i tato metoda otestovaná na vytvořeném datasetu. Vzhledem k tomu, že metoda dvě nahrávky vyhodnotí za shodné, pokud je rozdíl jejich hashů menší nebo roven hodnotě parametru `hashdiff_tresh`, cílem bylo zjistit, jak velkých hodnot budou rozdíly hashů u dvojic jednotlivých typů duplikátů dosahovat. Pokud je `hashdiff_tresh` nastavená na příliš vysokou hodnotu, systém k sobě přiřadí i nahrávky, které shodné nejsou, pokud je naopak hodnota parametru příliš nízká, systém nemusí všechny duplikáty odhalit.

Jak lze vidět na obr. 6.12 až obr. 6.16, u testovaných subsetů vytvořených z datasetu unikátních nahrávek začal systém vracet neshodné dvojice při hodnotách `hashdiff_tresh` ≥ 18. V případě datasetu obsahujícího pouze různé interpretace té stejné skladby (viz obr. 6.17) k tomu začalo docházet již při hodnotě `hashdiff_tresh` ≥ 12, což napovídá tomu, že mezi různými interpretacemi té stejné skladby jsou menší rozdíly, než mezi nahrávkami zcela odlišnými.

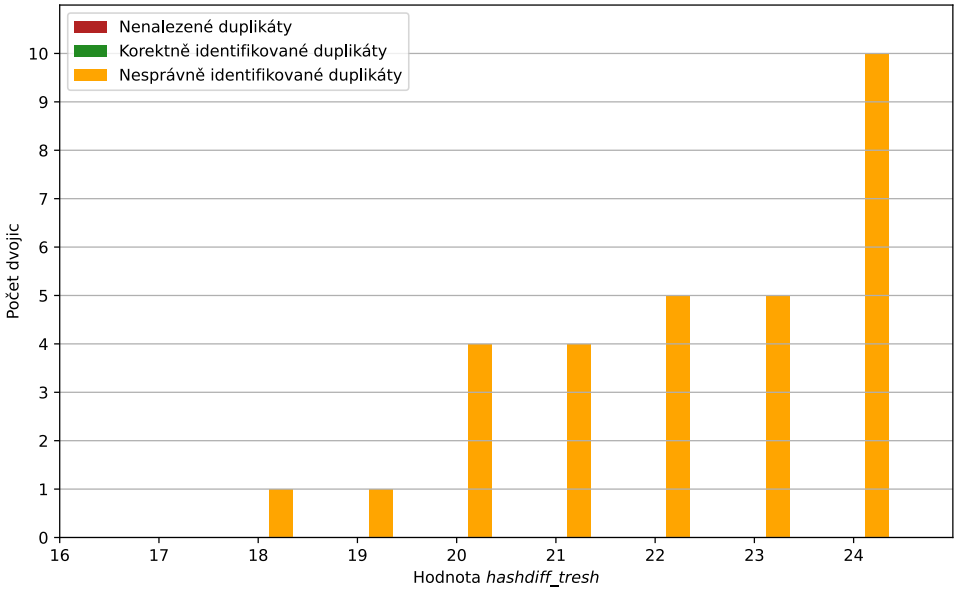
V případě subsetu obsahujícího zcela identické duplikáty (viz obr. 6.12) došlo ke korektnímu rozpoznání všech duplikátů již při hodnotě `hashdiff_tresh` $= 0$. Vstupní hudební soubory jsou v tomto případě zcela identické, tudíž jsou shodné i jejich hashe. Pokud uživatel v databázi očekává pouze zcela shodné duplikáty, lze použít tuto metodu s nastavenou hodnotou `hashdiff_tresh` $= 0$.

U subsetu obsahujícího duplikáty s krátkým šumem na začátku či konci nahrávek (viz obr. 6.14) byly všechny duplikáty odhaleny při hodnotě `hashdiff_tresh` ≥ 6.

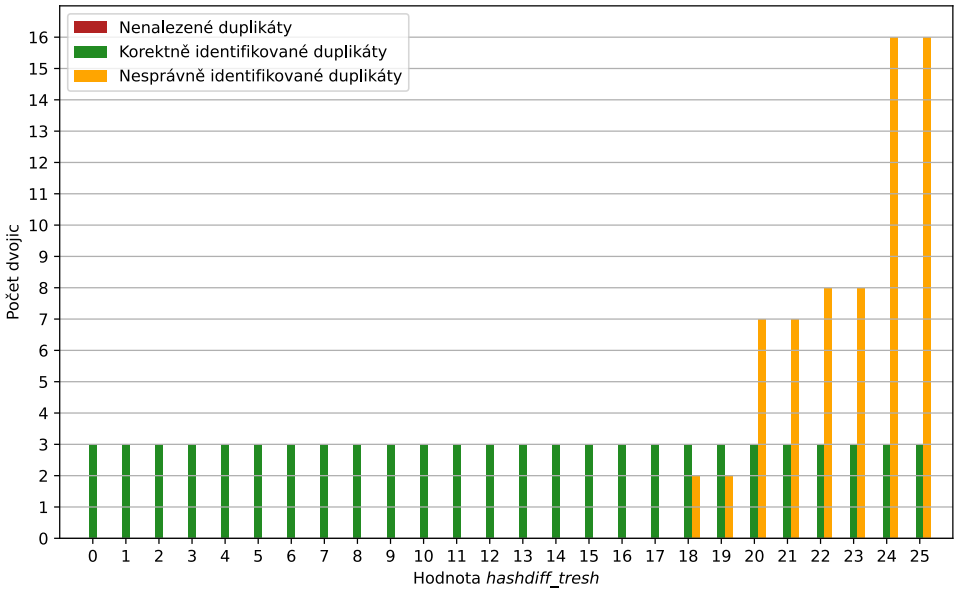
V případě subsetu s duplikáty obsahujícími šum o délce poloviny celkové doby jejich trvání (viz obr. 6.15) lze vidět, že metoda používající Image Hashing není příliš efektivní. Všechny shodné nahrávky byly identifikované až při `hashdiff_tresh` ≥ 34, což už je hodnota tak vysoká, že při ní docházelo k přiřazení velkého počtu neshodných dvojic.

U subsetu obsahujícího duplikáty nízké kvality (viz obr. 6.16) byly všechny shodné dvojice nalezeny při `hashdiff_tresh` ≥ 4.

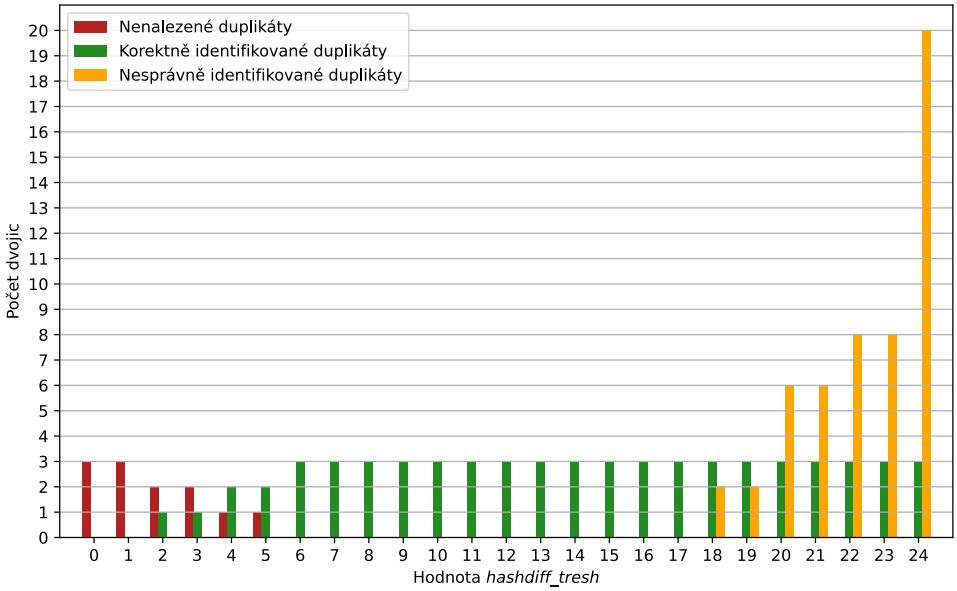
Průměrná výpočetní doba metody (tj. čas výpočtu chromagramů i vyhodnocení shody pro všechny dvojice) činila 16 sekund.



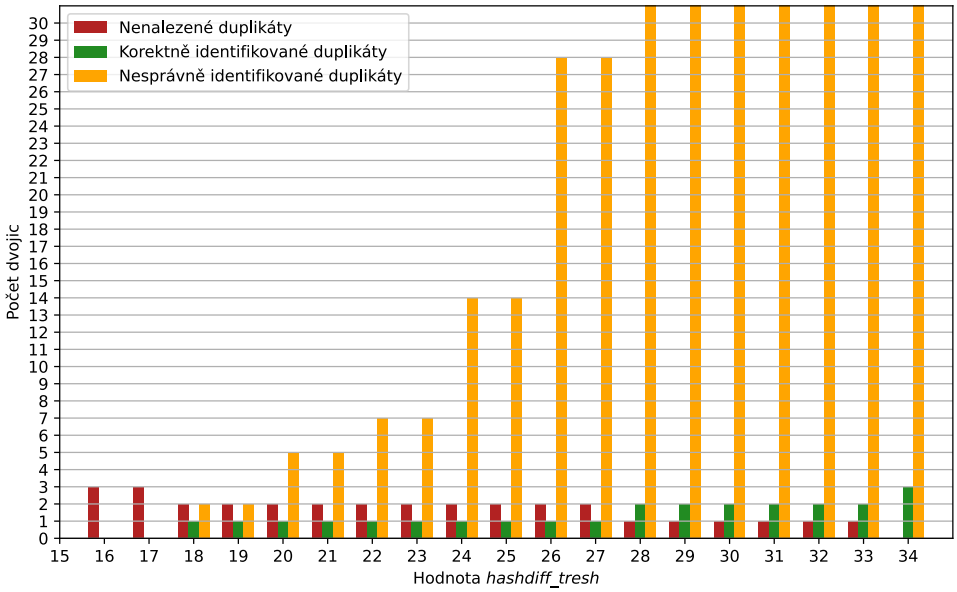
Obr. 6.12: Subset bez duplikátů – výsledky metody vyhodnocující shodu pomocí techniky Image Hashing



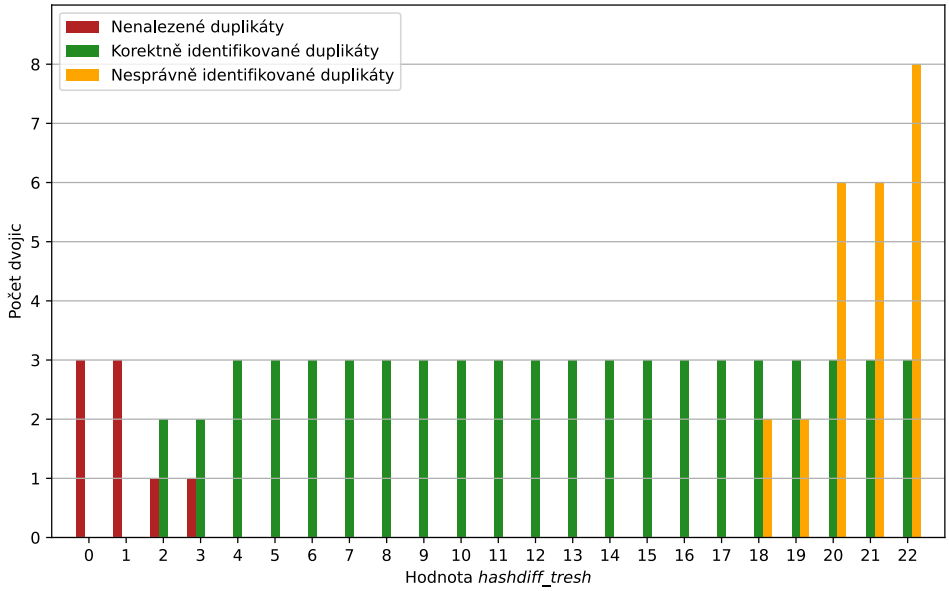
Obr. 6.13: Subset obsahující duplikáty – výsledky metody vyhodnocující shodu pomocí techniky Image Hashing



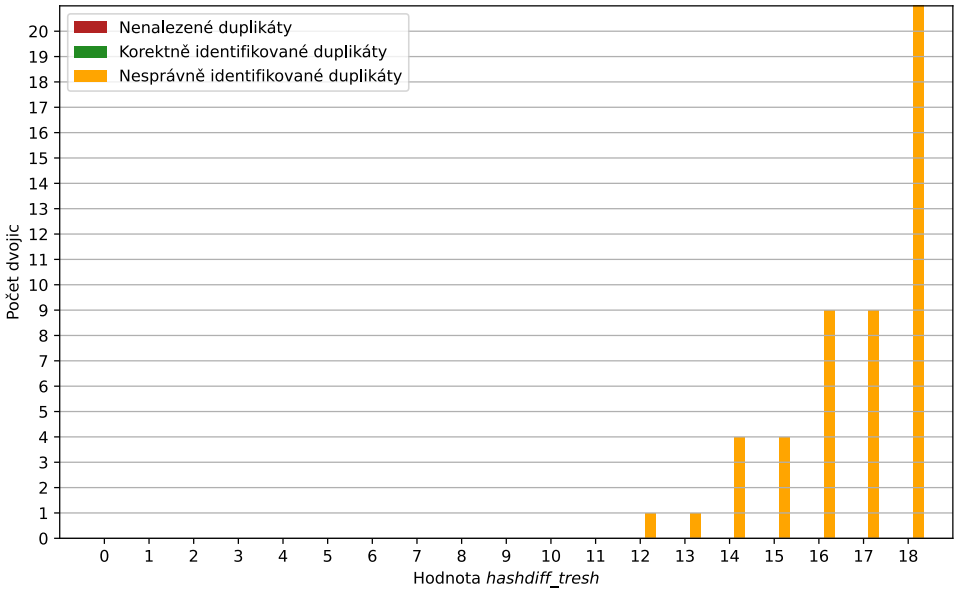
Obr. 6.14: Subset s duplikáty obsahujícími šum na začátku či konci – výsledky metody vyhodnocující shodu pomocí techniky Hashing



Obr. 6.15: Subset s duplikáty obsahujícími dlouhý šum na začátku – výsledky metody vyhodnocující shodu pomocí techniky Image Hashing



Obr. 6.16: Subset obsahující duplikáty nízké kvality – výsledky metody vyhodnocující shodu pomocí techniky Image Hashing



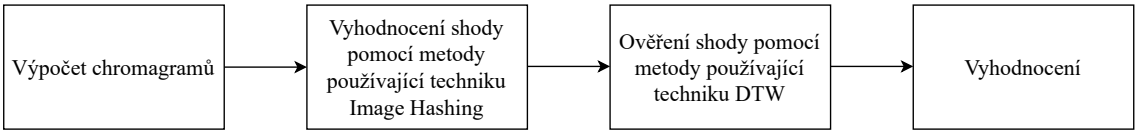
Obr. 6.17: Dataset vytvořený z interpretací – výsledky metody vyhodnocující shodu pomocí techniky Image Hashing

6.5 Metoda vyhodnocující shodu pomocí kombinace Image Hashing a DTW

Z výsledků testování dvou předešlých metod bylo zjištěno, že při použití metody vyhodnocující shodu pomocí DTW se jako ideální jeví varianta MrMsDTW s nastavenou hodnotou parametru $\tau = 10^6$. V takovém případě je přesnost vyhodnocení dostatečně vysoká, výpočetní čas je ale v porovnání s metodou vyhodnocující shodu pomocí techniky Image Hashing podstatně vyšší – průměrná doba výpočtu zde činila 142 sekund, zatímco při aplikaci metody používající Image Hashing byla průměrná doba výpočtu pouze 16 sekund. Metoda používající Image Hashing je sice výpočetně rychlá, ale nastává zde problém, na jakou hodnotu nastavit parametr `hashdiff_tresh`, aby systém korektně identifikoval všechny duplikáty a zároveň nedošlo k vzájemnému přiřazení nahrávek, které shodné nejsou.

Z těchto poznatků byla navržena třetí metoda, která kombinuje principy obou výše zmíněných metod. Tuto metodu lze spustit pomocí funkce `find_duplicates_combined(...)`. Diagram metody je znázorněný na obr. 6.17. Ze vstupních nahrávek jsou nejprve vypočteny chromagramy, na které je následně aplikovaná metoda používající techniku Image Hashing. Parametr `hashdiff_tresh` je zde nastaven na dostatečně vysokou hodnotu, aby bylo zajištěno, že metoda korektně odhalí všechny duplikáty. Cílem této fáze algoritmu je nalezení dvojic, které vykazují podobnost a tudíž mohou být shodné.

Další fází algoritmu je ověření dvojic, které k sobě byly přiřazené v předchozí fázi, pomocí metody vyhodnocující shodu pomocí DTW. Dvojice, které k sobě byly chybně přiřazené vlivem příliš vysoké hodnoty parametru `hashdiff_tresh`, jsou v této fázi eliminované, neboť systém používající DTW je dostatečně přesný a v této fázi algoritmu tudíž dojde k jejich vyhodnocení za neshodné. Výstupem systému je seznam dvojic, které byly vyhodnocené za shodné oběma metodami.



Obr. 6.18: Diagram algoritmu metody vyhodnocující shodu pomocí kombinace technik Image Hashing a DTW

6.5.1 Vyhodnocení metody na datasetu

Pro účely vyhodnocení přesnosti a výpočetní náročnosti této metody v porovnání s metodou používající pouze techniku DTW byl vytvořen dataset většího rozměru. V něm bylo obsaženo celkem šedesát unikátních nahrávek – dvacet nahrávek vážné hudby, dvacet hudebních souborů populární hudby a dvacet nahrávek elektronické hudby. Z každé této skupiny bylo následně vybráno a zkopírováno deset souborů, dataset tedy obsahoval šedesát unikátních nahrávek a třicet duplikátů. Na začátek deseti duplikátů byly vloženy čtyři sekundy ticha, dalším deseti duplikátům byla snížena kvalita jejich vyexportováním do `.mp3` o přenosové rychlosti 64 kbps.

V případě metody používající techniku DTW (konkrétně MrMsDTW s parametrem $\tau = 10^6$) trval výpočet 1872 sekund, zatímco v případě použití metody používající kombinaci Image Hashing a DTW (s parametry `hashdiff_tresh` = 10 a $\tau = 10^6$) trval výpočet pouze 108 sekund. Výsledky získané oběma metodami byly shodné – v obou případech bylo všech třicet duplikátů korektně identifikováno, aniž by došlo k chybnému vyhodnocení neshodných dvojic systémem za shodné. Zatímco v případě použití metody využívající DTW byl celkový počet testovaných dvojic roven 4005, v případě druhé metody bylo v druhé fázi výpočtu pomocí DTW ověřeno pouze 30 dvojic, přičemž všechny byly duplicitní. Tato metoda kombinující techniky Image Hashing a DTW se tedy jeví jako ideální z hlediska přesnosti i výpočetní náročnosti.

6.6 Navržená přednastavení systému

Pro jednoduché spuštění systému byly na základě získaných výsledků navrženy čtyři různé přednastavení parametrů odpovídající čtyřem úrovním přesnosti evaluace (viz tab. 6.2). Přednastavené varianty systému lze spustit pomocí funkce `find_duplicates(...)`, přičemž úroveň přesnosti se nastaví pomocí vstupního parametru `accuracy`. Jednotlivé úrovně přesnosti jsou popsané níže.

accuracy = low je přednastavená přesnost systému pro případy, kdy uživatel v databázi očekává pouze zcela shodné duplikáty.

accuracy = normal odpovídá výchozímu nastavení systému navrženému pro univerzální využití. Při tomto nastavení dokáže systém rozpoznat duplikáty obsahující šum, ticho či potlesk. Úroveň je zároveň robustní i vůči případům, při kterých je jedna z nahrávek duplicitní dvojice uložena ve velmi nízké kvalitě.

accuracy = high implementuje vyšší toleranci vůči rozdílům nahrávek během první fáze výpočtu používající techniku Image Hashing.

accuracy = extreme je navržena pro extrémní případy, při kterých uživatel očekává, že se v začátcích či koncích nahrávek duplikátů mohou vyskytovat velmi dlouhé úseky ticha, potlesku atp. Toto nastavení jako jediné nepoužívá techniku Image Hashing, výpočetní náročnost této varianty je tedy nejvyšší.

Tab. 6.2: Přednastavené úrovně přesnosti výsledného systému

Accuracy	**Technika použitá k vyhodnocení shody**	**Hodnota hashdiff_tresh**	**Hodnota** τ	**Verify_extremes**
low	Image Hashing	0	–	False
normal	Image Hashing + MrMsDTW	10	10^6	False
high	Image Hashing + MrMsDTW	20	10^6	False
extreme	MrMsDTW	–	10^7	True

Závěr

Tato práce se zabývala problematikou identifikace hudebních duplikátů. Jejím prvním cílem bylo shrnutí již existujících metod, které jsou k těmto účelům využívané. Ty byly popsané v teoretické části práce. První kapitola teoretické části se věnovala oblasti Music Information Retrieval. Zde byla definovaná hudba společně s jejími funkcemi ve společnosti a dále zde byly popsané způsoby využití oblasti MIR. Další kapitola se zabývala některými metodami a parametry používaných při zpracování zvukového signálu, na které bylo v práci dále odkazováno. Ve třetí kapitole byl popsaný algoritmus dynamického borcení časové osy společně s různými variantami optimalizace jeho výpočetní náročnosti. Následovala kapitola popisující techniku Image Hashing, která byla použita ve dvou navržených metodách výsledného systému. V páté kapitole byly vysvětlené již existující algoritmy používané k vyhledávání duplicitních nahrávek. Poslední kapitola se věnovala implementaci výsledného systému navrženého v rámci praktické části práce. Zde byly popsané algoritmy jednotlivých variant systému společně s výsledky jejich testování na předem připraveném datasetu.

Druhým cílem práce bylo navržení a realizace systému pro identifikaci hudebních duplikátů. Implementovány byly tři různé metody. První metoda k vyhodnocení shody používá DTW, druhá metoda vyhodnocuje podobnost pomocí techniky Image Hashing a třetí metoda využívá kombinace obou těchto technik. Bylo zjištěno, že z hlediska přesnosti se jako ideální jeví použití metody vyhodnocující shodu pomocí DTW (konkrétně variantu MrMsDTW o nastavené hodnotě parametru $\tau = 10^6$). V případě použití této metody byly korektně identifikované všechny duplikáty obsažené v testovacím datasetu. Metoda je dostatečně robustní i vůči menším rozdílům mezi hudebními soubory duplikátů, jako je jejich kvalita, obsažený potlesk na konci atp. Při nastavení parametru `verify_extremes` na hodnotu `True` je pomocí metody možné odhalit i extrémní případy duplikátů, ve kterých půlku doby jejich trvání tvoří ticho či ruch. Při použití metody na dataset obsahující různé interpretace stejné skladby nedocházelo k žádnému chybnému přiřazení neshodných dvojic. Nevýhoda této metody však spočívá v její vysoké výpočetní náročnosti. Metoda vyhodnocující shodu pomocí techniky Image Hashing disponuje podstatně nižší výpočetní náročností, její nevýhodou je ale nižší přesnost, která klesá s rostoucím výskytem rozdílů mezi hudebními duplikáty. Problematické je zde určení, na jakou hodnotu nastavit parametr `hashdiff_tresh` určující maximální rozdíl mezi hashi, aby došlo ke korektní identifikaci všech duplikátů obsažených v databázi a zároveň nedocházelo k chybnému přiřazení neshodných dvojic. Jako ideální se jeví použití třetí navržené metody, která shodu vyhodnocuje pomocí kombinace technik Image Hashing a DTW. V jejím případě dojde v první fázi výpočtu k předurčení dvojic vy-

kazujících podobnost, které jsou následně znovu ověřeny systémem vyhodnocujícím shodu technikou DTW. Tato metoda vykazuje podstatně nižší výpočetní náročnost než metoda používající pouze DTW, její přesnost je však pro většinu typů duplikátů shodná.

Z hlediska dalšího rozšíření systému se zde nabízí například možnost implementace varianty, která by dokázala identifikovat a vzájemně k sobě přiřadit různé interpretace stejné skladby. Takové nahrávky jsou si velmi podobné, nikoli však shodné. K těmto účelům by mělo být možné použít techniku Image Hashing, problematické je zde však určení odpovídajího rozsahu rozdílů vypočítaných hashů.

Literatura

[1] SYROVÝ, Václav. *Hudební zvuk: příspěvek k teorii zvukové tvorby*. 2., dopl. vyd. V Praze: Akademie múzických umění, 2014. Akustická knihovna Zvukového studia Hudební fakulty AMU. ISBN 978-80-7331-323-4.

[2] BADER, Rolf. *Springer Handbook of Systematic Musicology*. Springer, 2018. ISBN 9783662550021.

[3] OWEN, James. Bone Flute Is Oldest Instrument, Study Says. *National Geographic* [online]. June 24, 2009 [cit. 2021-10-9]. Dostupné z: `https://www.nationalgeographic.com/culture/article/bone-flute-is-oldest-instrument--study-says`

[4] HIGHAM, Thomas, Laura BASELL, Roger JACOBI, Rachel WOOD, Christopher Bronk RAMSEY a Nicholas J. CONARD. esting models for the beginnings of the Aurignacian and the advent of figurative art and music: The radiocarbon chronology of Geißenklösterle. In: *Journal of Human Evolution* [online]. 2012, s. 664-676 [cit. 2021-10-9]. ISSN 00472484. Dostupné z: doi:10.1016/j.jhevol.2012.03.003

[5] LUNDIN, Elizabeth. Musical History: The World's Oldest Melody. *History Things* [online]. March 14, 2021 [cit. 2021-10-9]. Dostupné z: `https://historythings.com/musical-history-worlds-oldest-melody/`

[6] Uses and functions. MERRIAM, Alan P. *The Anthropology Of Music*. Evanston, Illinois: Northwestern University Press, 1964, s. 209-227. ISBN 0-8101-0607- 8.

[7] HARGREAVES, David J. a Adrian C. NORTH. The Functions of Music in Everyday Life: Redefining the Social in Music Psychology. In: *Psychology of Music* [online]. 1999, s. 71-83 [cit. 2021-11-02]. ISSN 0305-7356. Dostupné z: doi:10.1177/0305735699271007

[8] SILLS, David a Amber TODD. Does Music Directly Affect a Person's Heart Rate? In: *Journal of Emerging Investigators* [online]. Boston (Massachusetts), 2015 [cit. 2021-11-02]. Dostupné z: `https://www.researchgate.net/publication/275521402_Does_Music_Directly_Affect_a_Person's_Heart_Rate`

[9] What is...Basso Continuo? *BBC Music Magazine* [online]. September 2013 [cit. 2021-11-07]. Dostupné z: `https://www.classical-music.com/features/articles/what-isbasso-continuo/`

[10] SCHEDL, Markus. *Automatically extracting, analyzing, and visualizing information on music artists from the world wide web* [online]. Linz, 2008 [cit. 2021-9-26]. Dostupné z: `http://www.cp.jku.at/research/papers/schedl_phd_2008.pdf`. Disertace. Johannes Kepler Universität Linz, Institut für Computational Perception.

[11] *ISMIR* [online]. c2000-2021 [cit. 2021-9-26]. Dostupné z: `https://www.ismir.net/`

[12] KIKUCHI, Yudai, Naofumi AOKI a Yoshinori DOBASHI. *A Study on Automatic Music Genre Classification Based on the Summarization of Music Data.* In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) [online]. IEEE, 2020, 2020, s. 705-708 [cit. 2021-9-27]. ISBN 978-1-7281-4985-1. Dostupné z: doi:10.1109/ICAIIC48513.2020.9065046

[13] ULFA WIDOWATI, Fika, Fitriyanto SANTI NUGROHO a Guruh FAJAR SHIDIK. *Classification of Music Moods Based on CNN.* In: 2018 International Seminar on Application for Technology of Information and Communication [online]. IEEE, 2018, 2018, s. 318-321 [cit. 2021-9-27]. ISBN 978-1-5386-7486-4. Dostupné z: doi:10.1109/ISEMANTIC.2018.8549829

[14] HAN, Huihui, Xin LUO, Tao YANG a Youqun SHI. Music Recommendation Based on Feature Similarity. In: *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)* [online]. IEEE, 2018, 2018, s. 650-654 [cit. 2021-9-30]. ISBN 978-1-5386-5514-6. Dostupné z: doi:10.1109/IICSPI.2018.8690510

[15] JIANG, Miao, Ziyi YANG a Chen ZHAO. What to play next? A RNN-based music recommendation system. In: *2017 51st Asilomar Conference on Signals, Systems, and Computers* [online]. IEEE, 2017, 2017, s. 356-358 [cit. 2021-9-30]. ISBN 978-1-5386-1823-3. Dostupné z: doi:10.1109/ACSSC.2017.8335200

[16] SHAKIROVA, Elena. Collaborative filtering for music recommender system. In: *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* [online]. IEEE, 2017, 2017, s. 548-550 [cit. 2021-9-30]. ISBN 978-1-5090-4865-6. Dostupné z: doi:10.1109/EIConRus.2017.7910613

[17] SUNITHA, M. a T. ADILAKSHMI. Mobile based music recommendation system. In: *2016 International Conference on Inventive Computation Technologies (ICICT)* [online]. IEEE,

2016, 2016, s. 1-4 [cit. 2021-9-30]. ISBN 978-1-5090-1285-5. Dostupné z: doi:10.1109/INVENTIVE.2016.7830183

[18] WANG, Wengen, Xiaoqing YU, Yun Hui WANG a Ram SWAMINATHAN. Audio fingerprint based on Spectral Flux for audio retrieval. In: *2012 International Conference on Audio, Language and Image Processing* [online]. IEEE, 2012, 2012, s. 1104-1107 [cit. 2021-9-30]. ISBN 978-1-4673-0174-9. Dostupné z: doi:10.1109/ICALIP.2012.6376781

[19] YOUSSEF, Khalid a PENG-YUNG WOO. Instrument sound separation in songs. In: *2008 IEEE International Conference on Electro/Information Technology* [online]. IEEE, 2008, 2008, s. 442-447 [cit. 2021-9-30]. ISBN 978-1-4244-2029-2. Dostupné z: doi:10.1109/EIT.2008.4554343

[20] PRODEUS, Arkadiy a Kateryna KUKHARICHEVA. Automatic speech recognition performance for training on noised speech. In: *2017 2nd International Conference on Advanced Information and Communication Technologies (AICT)* [online]. IEEE, 2017, 2017, s. 71-74 [cit. 2021-9-30]. ISBN 978-1-5386-0637-7. Dostupné z: doi:10.1109/AIACT.2017.8020068

[21] YUCHANG CAO, Arkadiy, S. SRIDHARAN a M. MOODY. Speech separation by simulating the cocktail party effect with a neural network controlled Wiener filter. In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing* [online]. IEEE Comput. Soc. Press, 1997, 2017, s. 3261-3264 [cit. 2021-9-30]. ISBN 0-8186-7919-0. Dostupné z: doi:10.1109/ICASSP.1997.595489

[22] GOWRISHANKAR, B S a Nagappa U BHAJANTRI. An exhaustive review of automatic music transcription techniques: Survey of music transcription techniques. In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)* [online]. IEEE, 2016, 2016, s. 140-152 [cit. 2021-9-30]. ISBN 978-1-5090-4620-1. Dostupné z: doi:10.1109/SCOPES.2016.7955698

[23] ZARIPOV, R. Kh. An algorithmic description of a process of musical composition. In: *Doklady Akademii Nauk SSSR*. 1960, s. 1283-1286. ISSN 0869-5652.

[24] CHIU, Shih-Chuan a Man-Kwan SHAN. Computer Music Composition Based on Discovered Music Patterns. In: *2006 IEEE International Conference on Systems, Man and Cybernetics* [online]. IEEE, 2006, 2006, s. 4401-4406 [cit. 2021-11-07]. ISBN 1-4244-0099-6. Dostupné z: doi:10.1109/ICSMC.2006.384827

[25] GOODYER, Jason. How an AI finished Beethoven's last symphony and what that means for the future of music. *Science Focus* [online]. 2021 [cit. 2021-11-07]. Dostupné z: https://www.sciencefocus.com/news/ai-beethovens-symphony/

[26] ERREDE, Steven. Pitch vs. Frequency. *Acoustical Physics of Music: Lecture Notes and Other Hand-Outs* [online]. Illinois: Department of Physics, University of Illinois at Urbana-Champaign, 2017 [cit. 2021-9-26]. Dostupné z: https://courses.physics.illinois.edu/phys406/sp2017/Lecture_Notes/P406POM_Lecture_Notes/P406POM_Lect7.pdf

[27] HOWARD, David M. a Jamie ANGUS. *Acoustics and psychoacoustics*. Fifth edition. New York, 2017. ISBN 978-113-8859-876.

[28] LERCH, Alexander. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Hoboken, New Jersey: John Wiley & Sons, 2012, s. 73-74. ISBN 9781118266823.

[29] DOWNIE, S., J.: *Music Information Retrieval*. [online]. Annual Review of Information Science and Technology 37, Medford, 2003. [cit. 19. 11. 2018]. Dostupné z: http://www.music.mcgill.ca/~ich/classes/mumt611_06/downie_mir_arist37.pdf

[30] DVOŘÁK, V. *Implementace výpočtu FFT v obvodech FPGA a ASIC*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 71 s. Vedoucí diplomové práce doc. Ing. Lukáš Fujcik, Ph.D.

[31] HUDSON, Graham, Alain LEGER, Birger NISS a Istvan SEBESTYEN. JPEG at 25: Still Going Strong. In: *IEEE MultiMedia* [online]. 2017, s. 96-103 [cit. 2021-12-04]. ISSN 1070-986X. Dostupné z: doi:10.1109/MMUL.2017.38

[32] COOLEY, James W. a John W. TUKEY. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*. 1965, 19(90). DOI: 10.2307/2003354. ISSN 00255718. Dostupné také z: https://www.jstor.org/stable/2003354?origin=crossref

[33] ZHOU, Jianqin a Ping CHEN. Generalized Discrete Cosine Transform. In: *2009 Pacific-Asia Conference on Circuits, Communications and Systems* [online]. IEEE, 2009, 2009, s. 449-452 [cit. 2021-12-04]. ISBN 978-0-7695-3614-9. Dostupné z: doi:10.1109/PACCS.2009.62

[34] DEWI, Sita Purnama, Anggunmeka Luhur PRASASTI a Budhi IRAWAN. Analysis of LFCC Feature Extraction in Baby Crying Classification using KNN.

In: *2019 IEEE International Conference on Internet of Things and Intelligence System* (IoTaIS) [online]. IEEE, 2019, 2019, s. 86-91 [cit. 2021-12-04]. ISBN 978-1-7281-2516-9. Dostupné z: doi:10.1109/IoTaIS47347.2019.8980389

[35] SCHÖRKHUBER, Christian a Anssi KLAPURI. Constant-Q transform toolbox for music processing. In: Proc. 7th Sound and Music Computing Conf. [online]. January 2010 [cit. 2021-12-04]. Dostupné z:`https://www.researchgate.net/publication/228523955_Constant-Q_transform_toolbox_for_music_processing`

[36] RABINER, Lawrence a Biing-Hwang JUANG. *Fundamentals of speech recognition*. Upper Saddle River: Prentice Hall, 1993. ISBN 01-301-5157-2.

[37] MISHRA, Abhishek. Time Series Similarity Using Dynamic Time Warping -Explained. *Medium* [online]. Dec 11, 2020 [cit. 2021-11-10]. Dostupné z: `https://medium.com/walmartglobaltech/time-series-similarity-using-dynamic-time-warping-explained-9d09119e48ec`

[38] ZHANG, Jeremy. Dynamic Time Warping. *Towards data science* [online]. Feb1, 2020 [cit. 2021-11-10]. Dostupné z: `https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd`

[39] MÜLLER, Meinard. Dynamic Time Warping. MÜLLER, Meinard. *Information Retrieval for Music and Motion*. Berlin: Springer, 2007, s. 69–82. ISBN 978-3-540-74047-6.

[40] SENIN, Pavel. *Dynamic Time Warping Algorithm Review* [online]. Honolulu, USA, 2008 [cit. 2021-11-15]. Dostupné z: `https://www.researchgate.net/publication/228785661_Dynamic_Time_Warping_Algorithm_Review`. University of Hawaii at Manoa, Information and Computer Science Department.

[41] MÜLLER, Meinard, Henning MATTES a Frank KURTH. *An Efficient Multiscale Approach to Audio Synchronization* [online]. Römerstraße 164, 53117 Bonn, Germany, 2006 [cit. 2021-11-16]. Dostupné z: `https://www.researchgate.net/publication/200806245_An_Efficient_Multiscale_Approach_to_Audio_Synchronization`. University of Bonn,.

[42] SALVADOR, Stan a Philip K. CHAN. *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time* [online]. Melbourne, FL 32901, 2004 [cit. 2021-11-18]. Dostupné z: `https://www.researchgate.net/publication/235709979_Toward_Accurate_Dynamic_Time_Warping_in_Linear_Time_and_Space`. Florida Institute of Technology, Dept. of Computer Sciences.

[43] 2016 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* [online]. IEEE, 2016 [cit. 2021-11-20]. ISBN 978-1-4799-9988-0. Dostupné z: http://ieeexplore.ieee.org/document/7471739/

[44] VENKATESAN, R., S.-M. KOON, M.H. JAKUBOWSKI a P. MOULIN. Robust image hashing. *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)* [online]. IEEE, 2000, 664-666 [cit. 2022-05-20]. ISBN 0-7803-6297-7. Dostupné z: doi:10.1109/ICIP.2000.899541

[45] Looks Like It. *The Hacker Factor Blog: Make your own noise* [online]. Thursday, 26 May 2011 [cit. 2022-05-20]. Dostupné z: https://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html

[46] BOOKSTEIN, Abraham, Vladimir A. KULYUKIN a Timo RAITA. Generalized Hamming Distance. *Information Retrieval* [online]. **5**(4), 353-375 [cit. 2022-05-20]. ISSN 13864564. Dostupné z: doi:10.1023/A:1020499411651

[47] AllDup: Find and remove duplicate files. *MTSD* [online]. Chonburi, Thailand: MTSD, c2000-2021 [cit. 2021-11-27]. Dostupné z: https://www.alldup.info/alldup_help/alldup.php

[48] Music Duplicate Remover. *ManiacTools: the best media tools* [online]. Vancouver, Canada: ManiacTools, c2000-2021 [cit. 2021-11-27]. Dostupné z: https://www.maniactools.com/soft/music-duplicate-remover/

[49] WANG, Avery. An Industrial Strength Audio Search Algorithm. In: *ISMIR 2003: 4th International Conference on Music Information Retrieval* [online]. Baltimore, Maryland, USA, January 2003 [cit. 2021-11-26]. Dostupné z: https://www.researchgate.net/publication/220723446_An_Industrial_Strength_Audio_Search_Algorithm

[50] SURDU, Nicolae. How does Shazam work to recognize a song ? *Wayback Machine* [online]. Internet Archive, January 20, 2011 [cit. 2021-11-26]. Dostupné z: https://web.archive.org/web/20161024115723/http://www.soyoucode.com/2011/how-does-shazam-recognize-song

[51] CASEY, Michael a Malcolm SLANEY. *Audio Shingling for Measuring Musical Similarity* [online]. 2006 [cit. 2021-11-28]. Dostupné z: https://engineering.purdue.edu/~malcolm/yahoo/CaseySlaney2006(AudioShinglesForACMMM).pdf

[52] CASEY, Michael a Malcolm SLANEY. Song Intersection by Approximate Nearest Neighbor Search. In: *ISMIR 2006: 7th International Conference on Music Information Retrieval* [online]. 2006, January 2006 [cit. 2021-11-27]. Dostupné z: https://www.researchgate.net/publication/220723687_Song_Intersection_by_Approximate_Nearest_Neighbor_Search

[53] CASEY, Michael, Christophe RHODES a Malcolm SLANEY. Analysis of Minimum Distances in High-Dimensional Musical Spaces. In: *IEEE Transactions on Audio, Speech, and Language Processing* [online]. 2008, s. 1015-1028 [cit. 2021-11-28]. ISSN 1558-7916. Dostupné z: doi:10.1109/TASL.2008.925883

[54] ŠVEJCAR, Michael. Duplicate Finder. *Github* [online]. [cit. 2022-05-22]. Dostupné z: https://github.com/MichaelSvejcar/duplicate_finder

[55] *FFMpeg: A complete, cross-platform solution to record, convert and stream audio and video.* [online]. [cit. 2021-11-19]. Dostupné z: https://www.ffmpeg.org/

[56] *NumPy: The fundamental package for scientific computing with Python* [online]. 2021 [cit. 2021-11-19]. Dostupné z: https://numpy.org/

[57] *SciPy: Fundamental algorithms for scientific computing in Python* [online]. The SciPy community, c2008-2021 [cit. 2021-11-19]. Dostupné z: https://scipy.org/

[58] *Matplotlib: Visualization with Python* [online]. The Matplotlib development team, 2021 [cit. 2021-11-19]. Dostupné z: https://matplotlib.org/

[59] MCFEE, Brian, Colin RAFFEL, Dawen LIANG, Daniel ELLIS, Matt MCVICAR, Eric BATTENBERG a Oriol NIETO. *Librosa: Audio and Music Signal Analysis in Python* [online]. In: . 2015, s. 18-24 [cit. 2021-11-21]. Dostupné z: doi:10.25080/Majora-7b98e3ed-003

[60] ROUANET, Pierre. DTW (Dynamic Time Warping) python module. Github [online]. San Francisco, California, U.S., 2008 [cit. 2022-05-20]. Dostupné z: https://github.com/pollen-robotics/dtw

[61] TANIDA, Kazuaki. Fastdtw 0.3.4. *PyPi: The Python Package Index* [online]. Python Software Foundation, c2021, Oct 7, 2019 [cit. 2021-12-05]. Dostupné z: https://pypi.org/project/fastdtw/

[62] MÜLLER, Meinard, Yigitcan ÖZER, Michael KRAUSE, Thomas PRÄTZLICH a Jonathan DRIEDGER. Sync Toolbox: A Python Package for

Efficient, Robust, and Accurate Music Synchronization. In: *Journal of Open Source Software* [online]. 2021 [cit. 2021-11-21]. ISSN 2475-9066. Dostupné z: doi:10.21105/joss.03434

[63] PASZKE, Adam, Sam GROSS, Francisco MASSA, et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library* [online]. December 2019 [cit. 2021-11-28]. Dostupné z: `https://www.researchgate.net/publication/337756689_PyTorch_An_Imperative_Style_High-Performance_Deep_Learning_Library`

[64] NICKOLLS, John. Scalable parallel programming with CUDA introduction. In: *2008 IEEE Hot Chips 20 Symposium (HCS)* [online]. IEEE, 2008, 2008, s. 1-9 [cit. 2021-11-28]. ISBN 978-1-4673-8871-9. Dostupné z: doi:10.1109/HOTCHIPS.2008.7476518

[65] CHEUK, Kin Wai, Hans ANDERSON, Kat AGRES a Dorien HERREMANS. NnAudio: An on-the-Fly GPU Audio to Spectrogram Conversion Toolbox Using 1D Convolutional Neural Networks. *IEEE Access* [online]. 2020, 8, 161981-162003 [cit. 2021-11-28]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.3019084

[66] Csv – CSV File Reading and Writing. *Python* [online]. Python Software Foundation, c2001-2021 [cit. 2021-11-21]. Dostupné z: `https://docs.python.org/3/library/csv.html`

[67] SHAFRANOVICH, Yakov. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [online]. 2005 [cit. 2021-11-21]. Dostupné z: `https://www.rfc-editor.org/rfc/pdfrfc/rfc4180.txt.pdf`

[68] GAZONI, Eric a Charlie CLARK. *Openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files* [online]. Sep 23, 2021 [cit. 2022-05-20]. Dostupné z: `https://openpyxl.readthedocs.io`

[69] BUCHNER, Johannes. ImageHash 4.2.1. *Python Package Index* [online]. Python Software Foundation, c2022 [cit. 2022-05-20]. Dostupné z: `https://pypi.org/project/ImageHash/`

[70] LUNDH, Fredrik a Alex CLARK. *Pillow* [online]. c1995-2022 [cit. 2022-05-20]. Dostupné z: `https://pillow.readthedocs.io/`

Seznam symbolů a zkratek

BPM	Beats Per Minute – Počet úderů za minutu
CQT	Constant-Q Transform – Konstantní Q transformace
DCT	Discrete Cosine Transform – Diskrétní kosinová transformace
DFT	Discrete Fourier Transform – Diskrétní Fourierova transformace
DTW	Dynamic Time Warping – Dynamické borcení časové osy
FFT	Fast Fourier Transform – Rychlá Fourierova transformace
FT	Fourier Transform – Fourierova transformace
IFT	Inverse Fourier Transform - Inverzní Fourierova transformace
ISMIR	The International Society for Music Retrieval
LFCC	Linear Frequency Cepstral Coefficients - Lineární frekvenční kepstrální koeficienty
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MrMsDTW	Memory-Restricted Multiscale Dynamic Time Warping
MsDTW	Multiscale Dynamic Time Warping
RMS	Root Mean Square – Efektivní hodnota
STFT	Short-time Fourier transform – Krátkodobá Fourierova transformace

Seznam příloh

A Přiložené soubory

Přiložený soubor s názvem `duplicate_finder.zip` je archiv obsahující implementaci výsledného systému na rozpoznání hudebních duplikátů. Soubory `readme.txt` a `readme.md` dokumentují, jakými způsoby je systém možné spustit. Soubor `duplicate_finder.py` obsahuje zdrojový kód výsledného systému včetně všech funkcí, které byly v textu práce uvedeny. Soubory `config.ini` a `run.py` slouží k jednoduchému spuštění systému, které je popsané v souboru readme.md. Výsledný systém je zároveň možné stáhnout z repozitáře na Githubu [54].

Obsah archivu duplicate_finder.zip:

- config.ini
- duplicate_finder.py
- readme.md
- requirements.txt
- run.py