

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2024

Bc. Viktor Slezák



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

SVĚTELNÉ ANIMACE PRO SYSTÉM SPECTODA NA ZÁKLADĚ ANALÝZY PARAMETRŮ Z HUDEBNÍCH NAHRÁVEK

LIGHT ANIMATIONS FOR THE SPECTODA SYSTEM BASED ON THE ANALYSIS OF PARAMETERS FROM
MUSIC RECORDINGS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Viktor Slezák

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Matěj Ištvanek, Ph.D.

BRNO 2024



Diplomová práce

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

Student: Bc. Viktor Slezák

ID: 203745

Ročník: 2

Akademický rok: 2023/24

NÁZEV TÉMATU:

Světelné animace pro systém Spectoda na základě analýzy parametrů z hudebních nahrávek

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte systém pro výpočet parametrů z hudební nahrávky s důrazem na dynamickou, rytmickou a akordickou strukturu. Otestujte výhody a nevýhody nejnovějších přístupů založených na metodách strojového učení pro extrakci relevantních parametrů. Získaná data analyzujte a na jejich základě navrhněte a naprogramujte algoritmus generující specifický kód „SpectodaCode“ pro následné vytváření světelných animací. Výstupem práce bude jednoduché webové rozhraní, které po nahrání hudební skladby vygeneruje unikátní světelné animace.

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] CARSAULT, Tristan, NIKA, Jérôme, ESLING, Philippe a ASSAYAG, Gérard. 2021. Combining Real-Time Extraction and Prediction of Musical Chord Progressions for Creative Applications. Electronics, vol. 10, no. 21: 2634. DOI <https://doi.org/10.3390/electronics10212634>.

Termín zadání: 5.2.2024

Termín odevzdání: 21.5.2024

Vedoucí práce: Ing. Matěj Ištvanek, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.

předseda rady studijního programu

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

V práci je prozkoumána problematika oboru Music information retrieval. Na základě získaných znalostí je navržena struktura systému pro generování animací z parametrů hudební nahrávky. Dále jsou popsány a porovnány možnosti jak tyto parametry extra-hovat.

KLÍČOVÁ SLOVA

MIR, extrakce parametrů, audio analýza, generátor animací

ABSTRACT

In this paper, is explored the field of Music Information Retrieval. Based on the knowledge acquired, a system structure for generating animations from parameters of a music recording is proposed. Furthermore, the possibilities for extracting these parameters are described and compared.

KEYWORDS

MIR, feature extraction, audio analysis, animation generator

SLEZÁK, Viktor. *Světelné animace pro systém Spectoda na základě analýzy parametrů z hudebních nahrávek*. Online: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 72 s. Diplomová práce. Vedoucí práce: Ing. Matěj Ištvánek

Obsah

Úvod	10
1 Teorie	11
1.1 MIR - Music information retrieval	11
1.1.1 Historie	11
1.1.2 Řetězec zpracování – pipeline	12
1.2 Parametrizace hudebních nahrávek	14
1.2.1 Reprezentace audio signálů	14
1.2.2 Časová oblast	14
1.2.3 Frekvenční oblast	15
1.2.4 DFT – Diskrétní Fourierova transformace	17
1.2.5 STFT – Short-time Fourier transform	19
1.2.6 Dynamika hlasitost a intenzita	21
1.2.7 Barva	22
1.3 Detekce nástupů a analýza tempa skladby	23
1.3.1 Využití energie signálu	24
1.3.2 Využití spektra signálu	26
1.3.3 Detekce periodicity	28
1.4 Analýza chroma vlastností	29
1.5 Segmentace	29
1.6 Klasifikace žánrů a nálady	29
1.7 Dostupná řešení	29
1.7.1 Librosa	29
1.7.2 Madmom	31
1.7.3 Aubio	31
1.7.4 Hodnocení extrakce informací	31
1.8 Systém Spectoda	31
1.8.1 Tvorba animace	32
2 Výsledky studentské práce	33
2.1 Návrh výsledného systému	33
2.1.1 Uživatelské rozhraní	33
2.1.2 Parametry hudební nahrávky	33
2.1.3 Databáze bloků animací	35
2.1.4 Systém pro generování animací	36
2.2 Metody extrakce parametrů z hudební nahrávky	40
2.2.1 Detekce dob a tempa	40

2.2.2	Analýza chromavektorů	42
2.2.3	Efektivní hodnota signálu	44
2.2.4	Hlasitost	45
2.2.5	Segmentace	45
2.3	Realizace	46
2.3.1	Uživatelské rozhraní	47
2.3.2	Extrakce parametrů	47
2.3.3	Paleta barev	50
2.3.4	Výběr datasetu	51
2.3.5	Výběr bloku animace	52
2.3.6	Slabé stránky systému a návrhy na zlepšení	53
Závěr		54
Literatura		55
Seznam symbolů a zkratek		59
Seznam příloh		60
A Ukázky zdrojových kódů		61
B Některé příkazy balíčku <i>thesis</i>		66
B.1	Příkazy pro sazbu veličin a jednotek	66
B.2	Příkazy pro sazbu symbolů	66
C Druhá příloha		67
D Příklad sazby zdrojových kódů		68
D.1	Balíček <i>listings</i>	68
E Obsah elektronické přílohy		72

Seznam obrázků

1.1	Řetězec procesů MIR [25]	13
1.2	Zobrazení časového průběhu signálu	15
1.3	Reprezentace tónu E zahráného na basovou kytaru. a) Časová oblast b) Frekvenční spektrum	16
1.4	Časově spojitý signál a diskrétní signál	18
1.5	Signál o délce 1 s s počáteční frekvencí 10 Hz a koncovou frekvencí 30 Hz a) Původní signál b) Signál s okénkem od 0,2 s do 0,5 s c) Signál s okénkem od 0,35 s do 0,65 s d) Signál s okénkem od 0,5 s do 0,8 s [21]	20
1.6	Nahrávka piana a) Okamžitý výchylka nahrávky b) Frekvenční spek- trum nahrávky zobrazené pomocí spektrogramu	21
1.7	Tón A5 zahráný na klavír a) Okamžitá výchylka tónu b) ADSR obálka tónu	23
1.8	Detekce nástupů perikusivního zvuku a) Okamžitá výchylka nahrávky b) Lokální energie signálu $E_{xw}(n)$ c) Derivace lokální enegrie signálu s půlvlnným usměrněním $\Delta_E(n)$	25
1.9	Výpočet spektrálního toku pro nahrávku piana a) Okamžitá výchylka nahrávky b) Spektrogram nahrávky c) Spektrální tok bez komprese d) Spektrální tok s kompresí spektra $\gamma = 1$	27
1.10	Výpočet spektrálního toku z mel spektrogramu pro nahrávku piana a) Okamžitá výchylka nahrávky b) Mel spektrogram nahrávky c) Spektrální tok	28
2.1	Blokové schéma postupu uživatele webovou aplikací	33
2.2	Blokové diagramy tříd <code>Dataset</code> , <code>AnimationBlock</code>	35
2.3	Blokový diagram výběru datasetu.	36
2.4	Blokový diagram procesu generování animací	38
2.5	Blokový diagram procesu výpočtu parametrů pro výběr animace	39
2.6	Porovnání metod detekce dob na úryvku skladby Oh-Darling!. a) Mel spektrogram b) Detekce dob pomocí Librosa c) Detekce dob pomocí Madmom d) Detekce dob pomocí Aubio	41
2.7	Porovnání přesnosti a času metod detekce dob na skladbách Oh- Darling!, Come Together a Here Comes The Sun. a) Čas výpočtu b) Cemgil skóre	42
2.8	Porovnání výpočtu chromavektorů na skladbě Oh-Darling! v časovém rozmezí 0 - 60 s.	43
2.9	Porovnání délky výpočtu chromavektorů	44
2.10	Zobrazení efektivní hodnoty skladby Belly dancer	44

2.11	Porovnání vlivu výpočtu chroma vektorů na výslednou segmentaci pro 12 segmentů.	46
2.12	Struktura třídy <i>BeatTracking</i>	48
2.13	Struktura třídy <i>ChromaFeatures</i>	48
2.14	Provádění počtu segmentů	49
2.15	Struktura třídy <i>Segment</i>	49
2.16	Struktura třídy <i>GenreClassification</i>	50
2.17	Postup generování barevné palety „a)“ Barvy základní, „b)“ Barvy zbylé	51
C.1	Alenčino zrcadlo	67

Úvod

Diplomová práce se zabývá generováním světelných animací pro systémy Spectoda, na základě analýze parametrů získaných z hudebních nahrávek. Spectoda je společnost specializující se na inteligentní řízení světel a světelných efektů. Aplikace vyvíjená v rámci této práce je primárně určena pro animace adresovatelných LED pásků. Práce je strukturovaná do třech cílů, které pokrývají celý proces vývoje – od teoretického návrhu, přes průzkum a testování, až po vývoj finálního systému generujícího funkční kód pro ovládání LED pásků.

Prvním cílem – je prozkoumat možnosti generování animací a identifikovat, které informace z nahrávek jsou pro tento proces nejrelevantnější. Na základě získaných dat bude navržena struktura systému.

Druhým cílem – je prozkoumat existující řešení v oblasti extrakce hudebních parametrů, známé pod oborem Music Information Retrieval (MIR), a vyhodnotit je z hlediska přesnosti a efektivity výpočtů. Z těchto poznatků budou vybrány nejvhodnější metody pro implementaci.

Třetím cílem – je naprogramovat a optimalizovat navržený systém.

1 Teorie

V této kapitole je shrnuta teorie a sepsány potřebné informace pro realizaci a pochopení výstupů práce. Jsou popisovány zejména problematiky z oboru MIR popsaném v bodě 1.1. Metody pro extrakci parametrů z hudebních nahrávek a principy pro pochopení jak tyto metody fungují. Poslední bod 1.8 této kapitoly je zaměřen na systém Spectoda jeho možnosti tvorby interaktivních světelných animací a ovládání světelných zdrojů.

1.1 MIR - Music information retrieval

Music information retrieval je interdisciplinární vědní obor soustředící se na získávání informací z hudebních nahrávek. Jsou zde kombinovány znalosti mnoha oborů jako jsou muzikologie, psychoakustika, strojové učení, zpracování signálů a další.

Výstupy jeho výzkumu jsou využívány v populárních technologických aplikacích. Jednou z aplikací je personalizované doporučování hudebních skladeb, které se nachází v moderních streamovacích platformách. Další využití je v programech pro mixování hudby používaných diskžokeji k plynulejší práci díky analýze tempa a klíčových částí skladby. Tyto technologie se nachází v mnoha dalších aplikacích a s rozširováním digitálního audia jejich důležitost stále roste.

1.1.1 Historie

V tomto bodě je napsán souhrn historie MIR z knihy A New Companion to Digital Humanities [25]. Výzkumy v oblasti MIR se začínají objevovat na přelomu devatenáctého a dvacátého století s příchodem moderních statistických metod. Objevují se první pokusy o aplikování statistických metod na hudební partitura. Protože ještě nebyly natolik dostupné počítače jednalo se spíše o ruční práci s partiturami a tabulaturou. Z grafických notací se analyzovaly rysy skladeb a specifikovaly charakteristiky hudebního díla. S příchodem počítačů do výzkumných laboratoří se v letech 1960 až 1970 začalo více rozvíjet zpracování signálů a s tím související možnosti analýzy hudebních nahrávek pomocí počítačů. V těchto letech se poprvé začaly objevovat nyní známé termíny jako „Computational musicology“¹ a „Music information retrieval“. První oblast výzkumů se soustředila na analýzu tempa skladby. Z důvodu malé využitelnosti v komerčním sektoru nebyly výzkumu přiděleny patřičné priority. Tento útlum pokračoval až do roku 1990 kdy výzkumům v oblasti MIR pomohly dvě změny. První důležitou změnou byly rostoucí databáze digitální hudby, která

¹Computational musicology je vědní obor zahrnující veškeré výzkumy využívající výpočetní techniku pro práci s hudbou.

se stala lehce dostupná pro výzkumné týmy. Druhým bodem který přispěl k vývoji MIR byl nárůst výpočetního výkonu počítačů a nižší náklady s nimi spojené. Díky těmto změnám se stal výzkum dostupnější a jednodužší na realizaci [25].

V říjnu roku 2000 bylo uspořádáno první mezinárodní symposium zabývající se vývojem v oblasti MIR. Z této mezinárodní konference se stala tradice a vybudovala se kolem ní velká komunita nazývaná ISMIR². Každoročním vyvrcholením ISMIR je právě výše zmíněná konference, na které vědci z celého světa prezentují pokroky v oblasti výzkumu MIR. Zanedlouho naté v roce 2005 byl v rámci této konference představen model MIREX³ sloužící jako správa zásad pro hodnocení pokroků ve výzkumu MIR [8].

1.1.2 Řetězec zpracování – pipeline

V tomto bodě je popsán postup zpracování dat v aplikaci MIR. Jedná se o systém, kterým jsou data zpracovávána a určuje standardně využívaný řetězec jak při tvorbě algoritmů postupovat.

Vstupními daty se rozumí zejména hudební informace v digitální podobě. Tyto vstupní data se rozlišují do více typů. Mohou to být obrázky představující digitální formu zápisu hudby pomocí symbolů „not“ [25]. Například digitalizovaná partitura. Dalsím možným typem je „digitální hudba“. Jedná se o hudbu čistě v „digitálních notách“ představujících sadu příkazů. Například zápis v MIDI⁴. Nejrozšířenější formou vstupních dat jsou digitální hudební nahrávky představující audio signály.

Pre-processing – předzpracování signálu: Na začátku řetězce je zařazen blok předzpracování vstupních signálů. Tento blok se postará o připravení dat do podoby vhodné pro extrakci vlastností. Jedná se například o komprimaci komplexních vstupních signálů popsaných níže. Nebo je signál převáděn z časové do frekvenční oblasti. Více o technikách předzpracování je popsaano v bodě 1.2.

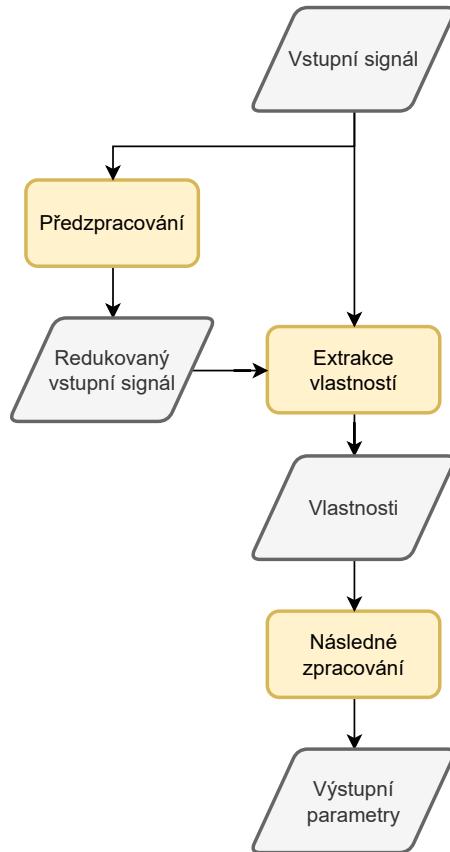
Feature extraction – extrakce vlastností signálu: Podle požadovaných vlastností pro extrakci je využíváno modelů popsaných v bodech 1.3, 1.6 a 1.4. S rostoucí popularitou strojového učení začaly při extrakci vlastností hudební nahrávky převládat kombinace hlubokých neuronových sítí. Tyto kombinace umožňují přesnější parametrizaci a menší chybovost.

²International Society of Music Information Retrieval - Mezinárodní sdružení pro MIR

³The Music Information Retrieval Evaluation eXchange - Komunitní rámec pro hodnocení pokroků výzkumu v oblasti MIR. Obhospodařovaný laboratoří International Music Information Retrieval Systems Evaluation Laboratory sídlící na University of Illinois [8].

⁴Musical Instrument Digital Interface - Volně dostupný hudební standart specifikující hardwerové a softwarové požadavky pro digitální přenos notace a komunikace mezi nástroji [36].

Post-processing – konečné zpracování: Posledním blokem v řetězci je konečné zpracování dat zajišťující jejich optimalizaci a převedení do požadované formy. V některých případech může ovlivnit přesnost zpracování.



Obr. 1.1: Řetězec procesů MIR [25]

Digitální hudební nahrávka se jako forma vstupních dat stala hlavním trendem výzkumu MIR. Je to způsobeno zejména dostupností velkých databází nahrávek ke kterým mají vědecké instituce přístup a nepotýkají se s problémy souvisejícími s autorskými právy [25].

Z důvodu velké komplexnosti vstupních dat se využívá několik technik komprimace signálů. Slučování vícekanálových nahrávek do mono signálu. Převzorkování signálu na nižší vzorkovací kmitočty, a rozložení na krátké překrývající se úseky, ze kterých mohou být nezávisle extrahovány jejich vlastnosti [14]. Výsledkem je kolekce paralelně složených sekvencí hodnot jednotlivých vlastností, které se následně zpracují na požadovaná výstupní data.

1.2 Parametrizace hudebních nahrávek

V této kapitole je popsán audio signál. Jak je reprezentován v oboru číslicového zpracování a základní principy práce s audiosignálem. V bodech 1.2.6 a 1.2.7 jsou popsány parametry získávané z audio signálu. Získané parametry slouží pro přesnější popis skladby.

1.2.1 Reprezentace audio signálů

Hudební dílo může být reprezentováno více formami. Jako tradiční médium pro její ukládání ještě před vznikem záznamu sloužily noty a další typy zápisů pomocí symbolů. Výsledné hudební dílo ale představuje mnohem více než počáteční notový zápis. Každý hudebník a hudební nástroj do skladby předává svou unikátnost. Při hře se noty začnou proměňovat v harmonické zvuky, hladké melodie a nástroje vzájemně rezonují. Každý z hudebníků do skladby přináší svou interpretaci. Jinak reagují na tempo, zvýrazňují odlišné noty a liší se jejich artikulace. Všechny tyto proměnné ve výsledku způsobují, že dílo není jen mechanické přehrání napsané partitury. Jeho součástí se stává unikátní přednes umělce a hudebního nástroje [21].

Z fyzikálního hlediska při interpretaci díla vznikají zvukové vlny šířící se vzduchem. Tyto vlny jsou reprezentovány kmítáním částic v pružném prostředí. V takovém prostředí jsou částice na sebe vázány a vytvářejí soustavu oscilátorů. Pokud dojde k vychýlení jedné částice ze své rovnovážné polohy, vlivem okolních částic dochází k působení pružných sil a vzniká její kmítání. Zároveň dochází k vzájemnému rozkmitání okolních částí a prostředím se začne šířit vlna. Jednotlivé částice kmitají pouze kolem své rovnovážné polohy. Nedochází tak k přenosu látky ale pouze energie a hybnosti [7]. Popsané kmítání je možné zachytit pomocí akustických měničů. Je získán analogový signál šířících se zvukových vln nazývaný jako audio signál. Pojmem audio je označován řetězec sloužící k záznamu, přenosu a reprodukci zvuků v mezích lidského slyšení. Avšak v audio signálu se už nenachází přesná reprezentace not a jejich paramterů jako jsou čas nástupu, tón, délka trvání, dynamika. Díky tomu je analýza hudebních signálů obtížným úkolem a je ovlivněna reprezentací interpreta, stavbou nástroje, akustikou prostoru a vnímáním posluchače. Zmíněnými problémy se zabývá samostatný vědní obor s názvem psychoakustika. Nejdůležitějšími parametry audio signálu které jsou podrobně popsány níže definujeme: frekvence, výška tónu, dynamika, intenzita, hlasitost a barva [21].

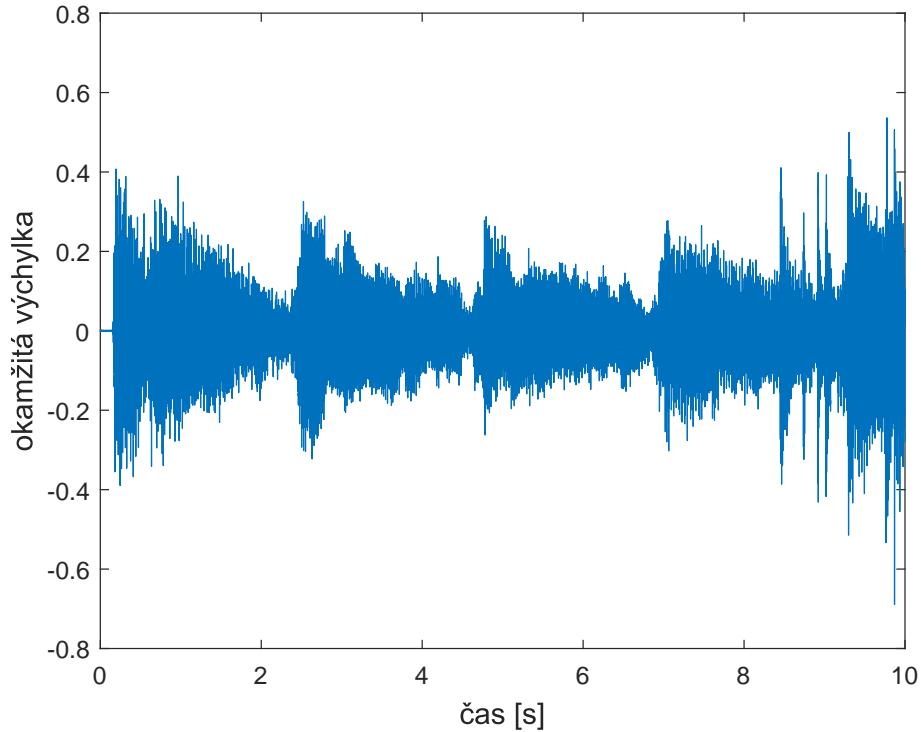
1.2.2 Časová oblast

Základní reprezentací audio signálu je tzv. zobrazení v **časové oblasti**. V časové oblasti představují číslicový signál vzorky. Jednotlivé vzorky udávají hodnotu signálu v

daném čase. Počet vzorků vztažených na jednotku času určuje vzorkovací frekvence signálu. Důležitým pravidlem pro vzorkování signálu je Shannonův-Nyquistův vzorkovací teorém

$$f_{vz} > 2f_{max} \quad (1.1)$$

kde f_{vz} je vzorkovací frekvence a f_{max} je maximální frekvence v audio signálu [3]. Pokud jednotlivé vzorky zobrazíme graficky získáme průběh signálu v čase viz obr.1.2. Tato reprezentace audio signálu poskytuje informace o průběhu amplitudy signálu. Využívá se například pro výpočet energie signálu popsaný v bodě 1.3.1.

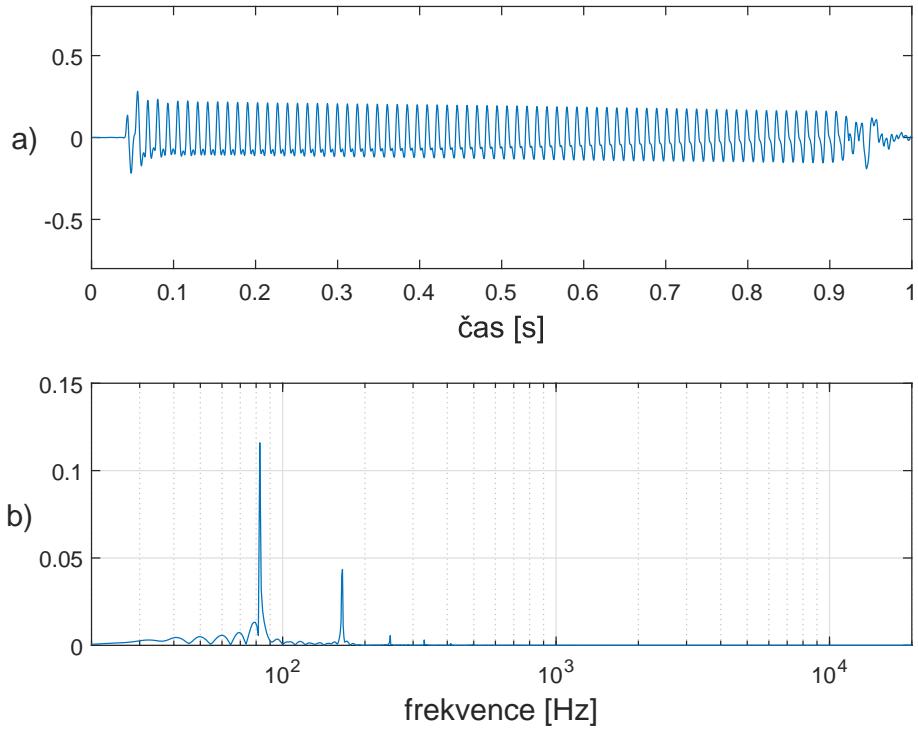


Obr. 1.2: Zobrazení časového průběhu signálu

1.2.3 Frekvenční oblast

Pro získání dalších informací o hudebním díle se využívá transformace signálu do frekvenční oblasti umožňující odlišné znázornění struktury signálu.

Ve frekvenční oblasti je signál reprezentován jeho frekvenčními složkami popsanými v komplexním tvaru. Spektrum představuje rozložení původní části signálu na jednotlivé frekvenční složky popsané funkcí sinus. Kde reálná složka obsahuje informaci o magnitudě „velikosti“ funkce sinus. Imaginární složka komplexního čísla pak udává počáteční fázi. V grafu jsou poté zobrazeny frekvenční složky se kterých se signál skládá viz obr. 1.3.



Obr. 1.3: Reprezentace tónu E zahráného na basovou kytaru. **a)** Časová oblast
b) Frekvenční spektrum

Jako názorný důvod proč je transformace do frekvenční oblasti přínosná je dán příklad. Na nástroj je zahrán tón, který je zaznamenán. V časové oblasti je možné určit délku tónu a jeho průběh podle ADSR obálky popsané v bodě 1.2.7. Pokud je ale potřeba zjistit výšku tónu a určit notu, tak se jedná o složitý proces. Díky transformaci do frekveční oblasti je patrná fundamentální frekvence tónu. Označována také první harmonická. Tato frekvence udává výšku tónu a je tak možné stanovit notu která byla zahrána.

Pro získání frekvenčního spektra signálu je třeba transformovat signál s časové oblasti. K tomu slouží několik úprav Fourierovy transformace podle vlastností vstupního signálu. Tyto metody jsou dále nazývány jako Fourierovy řady, Diskrétní časová Fourierova transformace a Diskrétní Fourierova transformace. V případě audio signálu se využívá zejména Diskrétní Fourierovy transformace popsané v bodě 1.2.4.

Fourierovy transformace zkráceně definována jako transformace převádějící signál mezi časovou a fekvenční oblastní pomocí harmonických signálů jež popisují funkce sinus a cosinus [3]. Funkce sinus a cosinus představují komplexní exponenciály.

$$e^{i\alpha t} = \cos(\alpha t) + i \sin(\alpha t) \quad (1.2)$$

Fourierova transformace pro **spojitý neperiodický signál** je pak zapsána jako

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t} dt \quad (1.3)$$

kde $\omega = 2\pi f$ a udává uhlovou frekvenci. Magnituda $|X(f)|$ je potom funkcí sudou [26].

Pro signál který je **spojitý a periodický** se definují Fourierovy řady a integrální funkce je počítaná pouze pro jednu periodu signálu

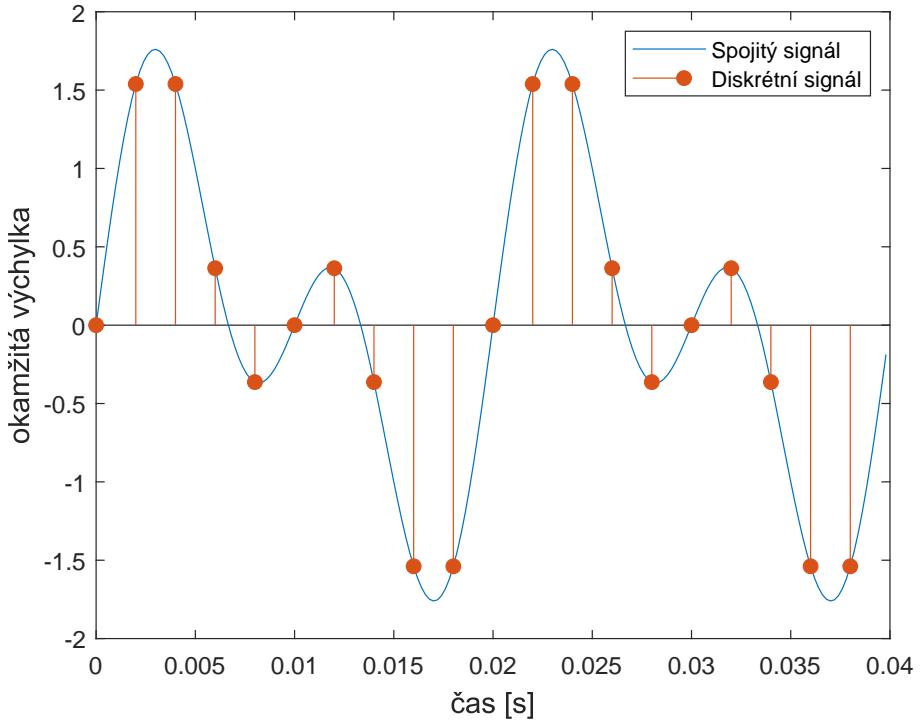
$$c[f_k] = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-ik\omega_0 t} dt \quad (1.4)$$

kde $f_k = k \times f_0$ a $k \in (\mathbb{Z}; 0, \pm 1, \pm 2, \dots)$. Vypočítané spektrum je diskrétní a ne-periodické [26]. Pokud je vstupní signál diskrétní hovoříme o Diskrétní Fourierově transformaci která je více popsána v následujícím bodě 1.2.4.

Po transformaci signálu do frekvenčního spektra jsou data signálu v komplexním tvaru a jejich magnituda $|X(f)|$ je funkce sudá, tím pádem symetrická kolem nuly a fáze $\varphi_x(f)$ je funkci lichou čili je středově symetrická. Pro analýzu audio signálů se využívá kladná část spektra.

1.2.4 DFT – Diskrétní Fourierova transformace

Pokud jsou signály zpracovávány pomocí výpočetních procesorů, tak může být uložen pouze omezený počet hodnot signálu. To znamená, že analogový signál spojitý v čase musí být převeden na signál digitální tvz. signál diskrétní, který je není spojitý v čase. Diskrétní signál je potom vhodný pro číslicové zpracování. Proto jsou pospané algoritmy DFT přizpůsobené právě pro zpracování diskrétních signálů nespojitéch v čase.



Obr. 1.4: Časově spojitý signál a diskrétní signál

Opět jsou dány odlišné definice pro signál diskrétní neperiodický a diskrétní periodický. Protože se jedná o signál diskrétní, tak zde odpadají integrální funkce. Pokud se jedná o signál **diskrétní neperiodický** jeho výsledné spektrum bude spojité a hovoříme o Fourierově transformaci diskrétní v čase. Matematicky je zapsána v následujícím tvaru

$$X(f) = \sum_{n=-\infty}^{\infty} x[n]e^{-i\Omega n} \quad (1.5)$$

kde

$$\Omega = 2\pi(f/f_s) \quad (1.6)$$

a f_s je vzorkovací frekvence signálu.

Protože v praxi signál není nikdy nekonečně dlouhý, tak je možné jej poskládat za sebe a vytvořit tak signál periodický. Pro periodické signály je výpočet DFT zapsán ve tvaru

$$X[f_k] = \sum_{n=1}^{N_0} x[n]e^{-i\Omega_k n} \quad (1.7)$$

kde

$$\Omega_k = 2\pi \frac{f_k}{f_s} \quad (1.8)$$

$$f_k = \frac{k f_s}{N_0} \quad (1.9)$$

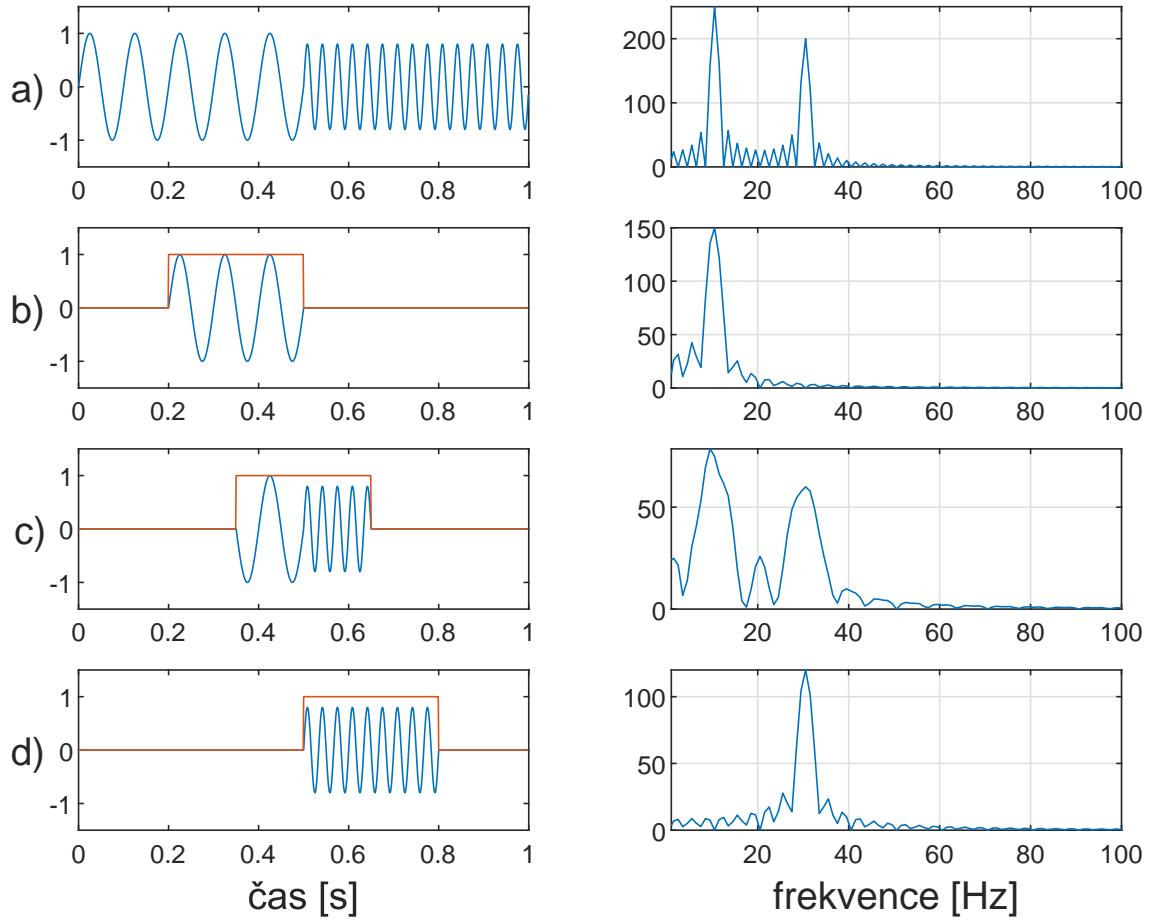
a $k \in (\mathbb{Z}; 0, N_0 - 1)$. N_0 udává počet vzorků v jedné periodě signálu. Hustota spektra K v takovém případě odpovídá $K = N_0$.

Ze strany výpočetní náročnosti je takto definovaný algoritmus neefektivní a výpočetně náročný. Pro výpočet DFT je zapotřebí velkého množství operací. Složitost algoritmu je pak zapsána jako $\mathcal{O}(N^2)$. Proto pokud počet vzorků N dosahuje většího množství je ve spoustě případů tento algoritmus příliš pomalý a neefektivní pro praktické využití.

Počet potřebných operací může být výrazně redukován. Na vývoj efektivního řešení výpočtu DFT se zasloužil Carl Friedrich Gauss a Joseph Fourier zhruba před dvěma sty lety. Tento algoritmus nazýváme Rychlou Fourierovou transformací zkráceně FFT. Počet operací pro výpočet takového algoritmu byl snížen na $\mathcal{O}(N \log_2 N)$ [21]. Například při použití vzorků $N = 2^{10} = 1024$. FFT vyžaduje $N \log_2 N = 10240$ operací namísto $N^2 = 1048576$ operací při použití DFT. Jak je vidět snížení výpočetní náročnosti je velké a exponenciálně roste s větším počtem vzorků N . Vynález FFT změnil odvětví zpracování signálů a je dnes využíván v miliardách telekomunikačních zařízeních. Stejně tak i ve zpracování a analýze zvukových signálů hraje důležitou roli [21].

1.2.5 STFT – Short-time Fourier transform

V roce 1946 Dennis Gabor představil STFT jako možnost zařazení frekvenčních složek ke konkrétnímu času signálu [30]. Fourierova transformace umožňovala převod signálu z časové oblasti do frekvenční ale nebylo zřejmé v jakém časovém úseku signálu se získané frekvenční složky nachází. Hlavní myšlenkou STFT je, že namísto analyzování celého signálu je analyzována pouze jeho malá část. Za tímto účelem je definovaná tzv. okénková funkce, která je nenulové pouze v malé části signálu. Analyzovaný signál je následně vynásoben vzniklou okénkovou funkcí a díky tomu vzniká malá nenulová část signálu dle okénkové funkce viz obr. 1.5. Chceme-li analyzovat signál v různých časech je tato funkce po signálu posouvána a následně se počítá DFT pro každý výsledný okénkový signál.



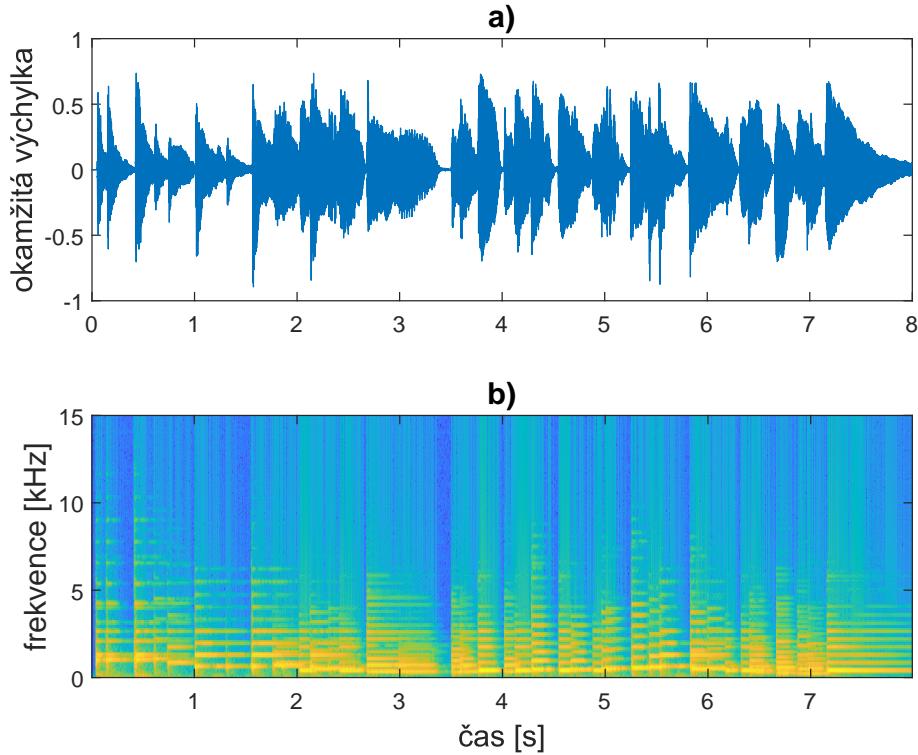
Obr. 1.5: Signál o délce 1 s s počáteční frekvencí 10 Hz a koncovou frekvencí 30 Hz
a) Původní signál **b)** Signál s okénkem od 0,2 s do 0,5 s **c)** Signál s okénkem od 0,35 s do 0,65 s **d)** Signál s okénkem od 0,5 s do 0,8 s [21]

Na obr. 1.5 je graficky znázorněna myšlenka STFT, která ukazuje princip určování frekvenčních složek v čase a jejich výhody. Signál je násoben obdélníkovou okénkovou funkcí ve třech místech. Tyto tři vzniklé signály jsou následně na sebe nezávisle transformovány do frekvenční oblasti. Z výsledků Fourierovy transformace lze vidět, že každá z těchto částí má jiné frekvenční spektrum. Pokud by bylo zapotřebí například určit přesný přechod mezi dvěma frekvencemi nacházejícími se v signálu. Lze zpřesnit časové měřítko analýzy pomocí délky okénka. Tím ale dochází ke zmenšení přesnosti ve frekvenční oblasti.

Na výsledku přesnosti analýzy pomocí STFT závisí také tvar použité okénkové funkce. V obr. 1.5 je použito obdélníkového okénka které díky svým ostrým hranám zkresluje výsledek o nechtěné frekvenční složky. Existuje více tvarů okénkových funkcí pro odstranění nežádoucích složek. Například to jsou Kaise, Chebyshev, Hann, Haming a další [6].

Pokud jsou analyzované data skrze okénka funkce STFT poskládány zpět za

sebe, tak představují matici průběhu frekvenčního spektra signálu v čase. Takové zobrazení se nazývá **spektrogram** a v případě 2D zobrazení se skládá z časové a frekvenční osy. Magnituda frekvencí je pak zobrazena barevnou škálou viz obrázek 1.6.



Obr. 1.6: Nahrávka piana **a)** Okamžitý výchylka nahrávky **b)** Frekvenční spektrum nahrávky zobrazené pomocí spektrogramu

1.2.6 Dynamika hlasitost a intenzita

V češtině se pojem hlasitost využívá pro reprezentaci subjektivního vnímání akustického tlaku definovanou například jednotkou phon⁵. Stejně tak je hlasitost využívána, hovoří li se o měřené hlasitosti vyjádřené například hladinou intenzity zvuku popsanou níže nebo efektivní hodnotou signálu. Z důvodu lepší srozumitelnosti jsou dále využívána anglické pojmy „volume“ a „loudness“.

Dynamika popisuje průběh hlasitosti „volume“ interpretovaného hudebního díla. Udává jeden z faktorů jak lze například odlišit stejnou skladbu zahranou různými muzikanty. Interpretací skladby umělec vytváří dynamiku přednášeného díla [21]. V notovém zápisu je dynamika neboli hlasitost přednesu popsána symboly jako jsou například pianissimo „pp“, piano „p“, forte „f“ a další.

⁵Phon - logaritmická jednotka vyjadřující individuální vnímání hlasitosti. Vnímání hlasitosti lidského ucha je závislé na křivce prahu slyšení a může se lišit pro každý tón [33].

V audio signálu je dynamika brána jako hlasitost „loudness“ Jedná se o změny apmlitudy signálu nebo jeho efektivní hodnoty RMS⁶ v čase.

Při měření hlasitosti „loudness“ v akustickém prostoru je pak využíváno pojmu **intenzita** zvuku a **akustický výkon**. Akustický výkon je definován jako množství energie vyzářené akustickým vysílačem ve vzduchu za jednotku času [10]. Jednotkou je W . A intenzita zvuku je pak definována jako množství energie, které projde jednotkovou plochou kolmou na směr šíření na jednotku času. Jednotkou je Wm^{-2} [37].

Z pohledu vnímání hlasitosti lidským uchem je rozsah vnímané intenzity zvuku v řádech bilionů. Práh slyšení činí $10^{-12} Wm^{-2}$ a práh bolesti je $10 Wm^{-2}$. Pro zmenšení tak velkého řádu je definována hladina intenzity zvuku v decibelech dB . Kde vztažnou hodnotou je práh slyšení $I_0 = 10^{-12} Wm^{-2}$. Hladina intenzity se vypočítá dle rovnice 1.10.

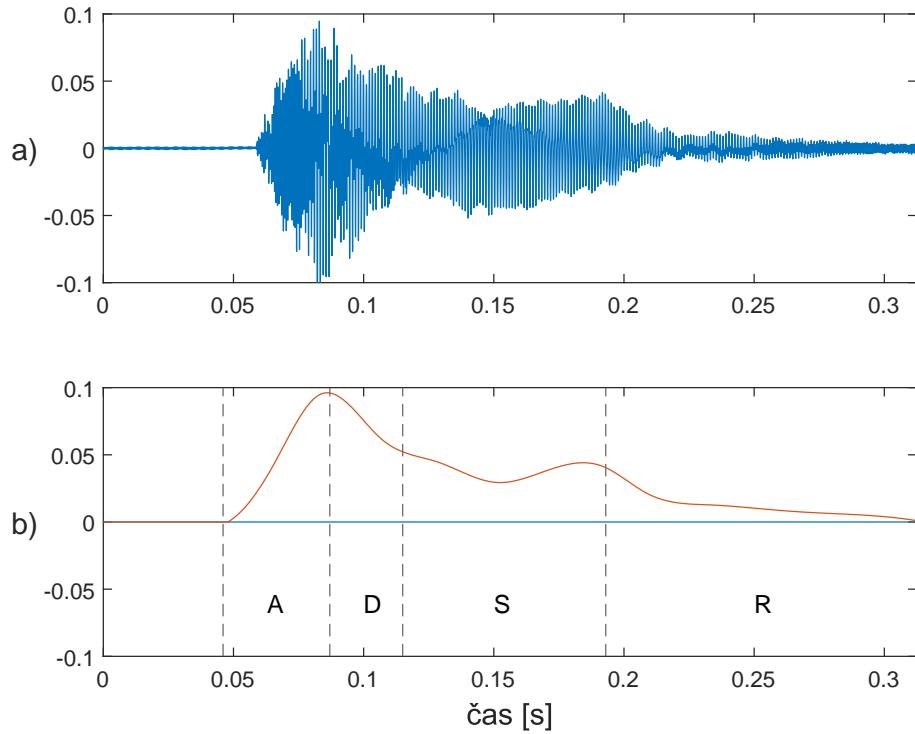
$$L_I = 10 \log\left(\frac{I}{I_0}\right) \quad (1.10)$$

1.2.7 Barva

V hudebním vyjádření se za slovem barva skrývá komplexní sdružení atributů. Jedná se jak o psychologický, tak hudební problém, který je vnímán individuálně[18]. Zjednodušeně se barva definuje jako vlastnosti, díky kterým je možné rozpoznat tón o stejné výšce a hlasitosti zahrnutý na dva různé nástroje[22]. Díky barvě je posluchač schopen rozpoznávat odlišné zvuky nástrojů a typů interpretace. Z důvodu špatné kategorizace barvy fyzikálními veličinami je většinou interpretována přídavnými jmény. Například je barva popisována jako jasná, temná, ostrá, čistá, teplá, pestrá, nazální, tonální a další.

Jedním z možných nástrojů pro analýzu barvy tónu je tzv. obálka tónu „signálu“. Obálku určuje okamžitá výchylka signálu v čase viz obr. 1.7 Je rozdělena na 4 fáze popsané z knihy Fundamentals of Music Processing [21]. **Attack „náběh“** určující začátek tónu. Například úder paličkou na blánu bubnu. V této fázi se nachází více ruchových složek z daného úderu a má velkou dynamiku. Následuje fáze s názvem **Decay „útlum“**. Po hlasitém úderu okamžitá výchylka signálu klesá a začíná převládat tonální složka. Decay udává dobu za kterou se signál z jeho maxima sníží na hodnotu sustain. **Sustain „podržení“** je fáze ve které je zřetelný tón a stálá hlasitost. Rezonující blána bubnu. Poslední fází je **Release „uvolnění“** při kterém dochází k poklesu hlasitosti zdroje zvuku až k uplnému utlumení. Například přiložení tlumítka na rezonující strunu.

⁶RMS - udává statistickou hodnotu z měření velikosti veličin. Je využívána u periodických veličin[4].



Obr. 1.7: Tón A5 zahráný na klavír a) Okamžitá výchylka tónu b) ADSR obálka tónu

Další informace o barvě signálu se nacházejí v jeho frekvenčním spektru. Tón zahráný na hudební nástroj má svou fundamentální „nosnou“ frekvenci nazývanou první harmonická udávající jeho výšku. Dle konstrukce nástroje se v signálu objevují násobky nosné frekvence. Tyto násobky představují vyšší harmonické složky tónu. Počet a amplituda vyšších harmonických složek má vliv na výslednou barvu tónu a je to hlavní důvod proč je lidské ucho schopné rozeznat stejný tón znějící na různé nástroje.

1.3 Detekce nástupů a analýza tempa skladby

S vývojem nových technologií se mění i přístupy využívané v MIR pro analýzu tempa skladby. V této kapitole je popsán postupný vývoj algoritmů. Od nejjednodušších přístupů po komplexní řešení využívající moderní metody strojového učení.

Detekce nástupů lze chápat jako detekci začátků not či dalších hudebních událostí, které se vyskytnou v průběhu skladby. Výskyt takových událostí je provázen zvýšením energie signálu zaznamenané ve fázi nástupu dle ADSR obálky popsané v bodě 1.2.7. Typickým znakem pro fázi nástupu je rychlý nárůst obálky amplitudy signálu. V této fázi se při vytváření tónu vyskytují **Tranzienty**. Tranzienty je možné

pozorovat zejména u neperkusivních nástrojů například u dechových nebo smyčcových. Jsou představovány nakmitávajícími a dokmitávajícími pochody. Jedná se o zvuk netónové podoby připomínající hluk s výraznými frekvenčními změnami. U smyčcových nástrojů takový zvuk může být ze začátku způsoben drhnutím smyčce o strunu do chvíle než se ustálí její kmitání. Délka tranzientů pak může dosahovat od jednotek milisekund až po stovky milisekund v závislosti na typu nástroje a technice hry [31]. Například v případě piana tranzient odpovídá počáteční fázi kdy byla zmáčknuta klávesa. Na základě zmáčknutí klávesy dochází ke zvednutí tlumítka, kladívko udeří do struny, struna začíná vibrovat a vibrace se přenáší do těla piana. V této fázi začíná tělo rezonovat a konečně dochází k ustálení tónové složky. Při úderu kladívka dochází k velkému přenosu energie patrném na obálce tónu. Díky tomu je lehké určit začátek tónu podle nárůstu amplitudy obálky [21].

U některých nástrojů je však energie přenášená po celou dobu znění tónu konstantně a dochází k pomalému jemnému náběhu tónu například u určitých technik hry na smyčcové nástroje nebo u dechových nástrojů. Fáze náběhu je pak pomalá a dlouhá a plynule přechází do fáze podržení. Pro tyto jemné zvuky se stává obtížné určit skutečnou pozici začátku noty.

Náročnost detekce nástupů se zvyšuje v případě, že nehráje pouze jeden nástroj. Jedná se například o polyfoní skladbu. Polyfoní skladbou rozumíme skladbu tvořenou více hlasy. Zároveň nemají určenou roli vedoucího a doprovodného hlasu [23]. Takové uspořádání hlasů může vést k překrývání a nástupy mohou být maskovány. Díky jevu maskování je obtížné zaznamenat změny energie signálu. V tomto případě přichází potřeba po komplexnějších metodách detekce nástupů [21]. Například pohled na krátkodobé změny ve spektru signálu pomocí využití STFT popsané v bodě 1.2.5.

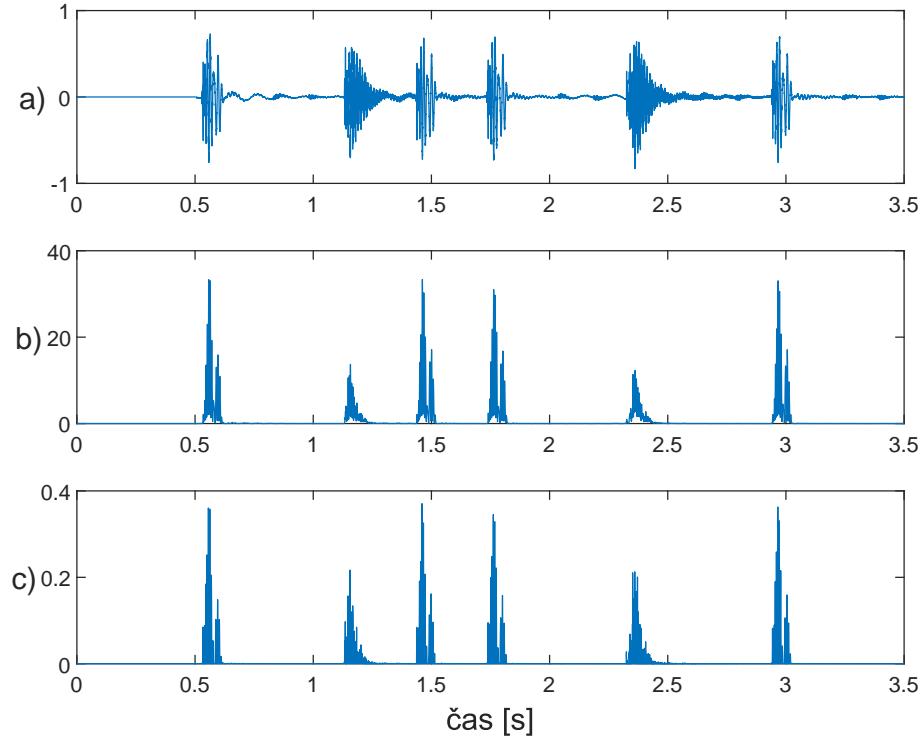
1.3.1 Využití energie signálu

Jak již bylo zmíněno při hraní dochází k přenosu energie od hráče na hudební nástroj. Tento přenos je ve značné míře provázen rychlým nárůstem amplitudy ve fázi náběhu na začátku tónu. Například při úderu kladívka piana na strunu nebo úderu paličky na blánu bubnu. Ná základě tohoto jevu je možnost detektovat nástup tónu pomocí funkce pro výpočet energie signálu v daném místě. Náhlé změny signálu v takto definované funkci ukazují potencionální místa nástupů. Matematicky pak funkci popisujeme. Stejně jako u STFT popsané v bodě 1.2.5 je definována okénková funkce $w(n)$ ve tvaru zvonu „bell-shaped function“ která je posouvaná po diskrétním signálu $x(n)$. Okénková funkce je symetrická podle počátku a potom platí že $m \in [-M : M]$

a $M \in \mathbb{N}$. Funkce lokální energie signálu je pak zapsána

$$E_{wx}(n) = \sum_{m=-M}^M |s(n+m)w(m)|^2 \quad (1.11)$$

pro $n \in \mathbb{Z}$ [21].



Obr. 1.8: Detekce nástupů perkusivního zvuku **a)** Okamžitá výchylka nahrávky **b)** Lokální energie signálu $E_{wx}(n)$ **c)** Derivace lokální energie signálu s půlvlnným usměrněním $\Delta_E(n)$

Pro názornější zobrazení se následně vypočítá derivace funkce lokální energie. V případě diskrétního signálu se derivace realizuje jako rozdíl dvou po sobě jdoucích vzorků. Protože pro detekci nástupů je důležitá zejména pozitivní změna energie nikoliv její pokles jsou ponechány pouze pozitivní rozdíly a negativní jsou zapsány jako nulové. V anglické literatuře je tento proces známý také jako půlvlnné usměrnění „half-wave rectification“. Vzorce jsou zapsány následovně

$$r = E_{wx}(n+1) - E_{wx}(n) \quad (1.12)$$

$$\Delta_E(n) = \frac{r + |r|}{2} = \begin{cases} r, & \text{if } r \geq 0 \\ 0, & \text{if } r < 0 \end{cases} \quad (1.13)$$

kde $r \in \mathbb{R}$ a $n \in \mathbb{Z}$. Na obr. 1.8 lze vidět zobrazení takto vypočítané lokální energie signálu a její derivace. Analyzovaný signál se skládá z několika perkusivních úderů. Pro výpočet bylo použito okno typu „bell-shaped function“ o velikosti 201 vzorků.

1.3.2 Využití spektra signálu

Díky rozložení signálu na jeho spektrum pomocí Fourierovy transformace popsané v bodě 1.2.3 je možné lépe rozpoznat strukturu nahrávky a zmírnit efekt maskování, který nastává při metodách založených na energii signálu popsaných v bodě 1.3.1. V polyfonní hudbě mohou interpretace o nižší hlasitosti být maskovány hlasitějšími projevy. Kdy například jeden nástroj v tónové fázi podržení ADSR obálky, popsané v bodě 1.2.7, může být hlasitější než fáza náběhu druhého nástroje. Takový nástup je pak maskován a je obtížná jeho detekce. V případě maskování některých z nástrojů v časové oblasti je možné spektrum signálu zaměřit na frekvenční oblast spektra ve které se maskovaný nástroj nachází. Díky tomu je možné nástup tónu takového nástroje detektovat pomocí spektra signálu. Každý hudební nástroj obsahuje jiné frekvenční spektrum [21]. Díky tomu se různé nástroje nachází na odlišných místech spektra. Jak spektrum daného nástroje vypadá určuje také barvu popsanou v bodě 1.2.7. S takovým jevem je důležité počítat při detekci nástupů založené na spektru signálu.

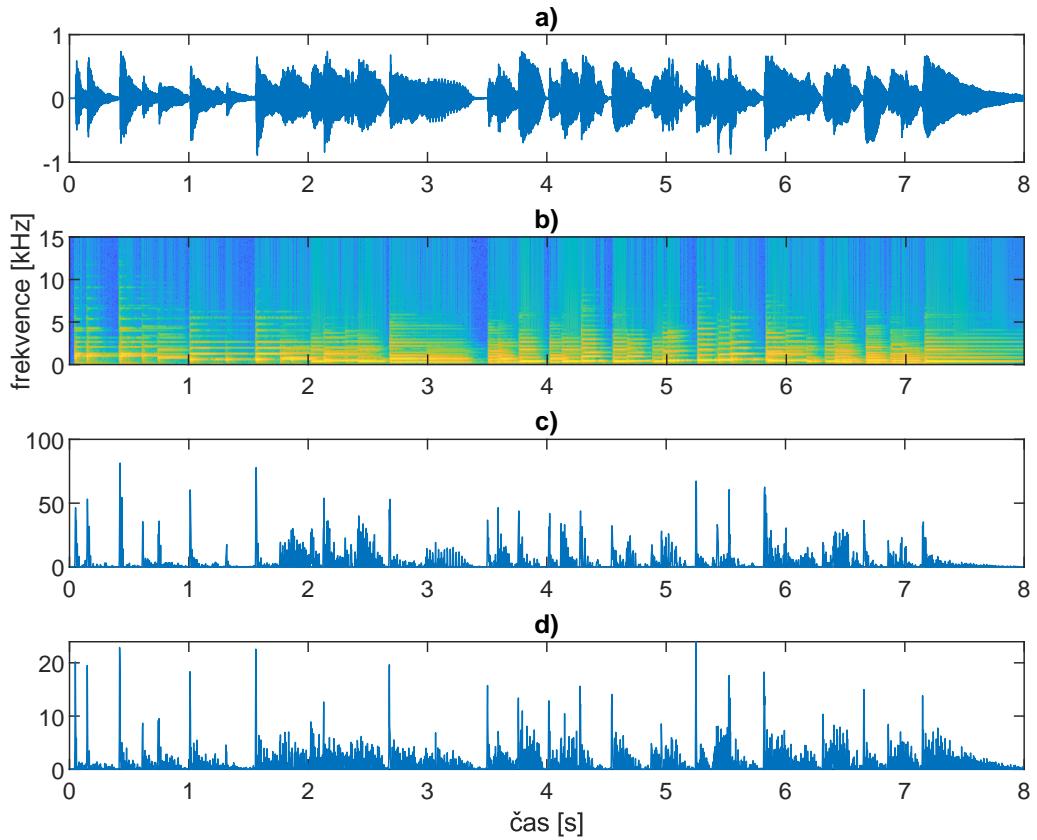
Jedním ze spůsobů analýzy nástupů pomocí spektra je detektovat změny ve spektru v průběhu času. Při zobrazení časového průběhu spektra nazývaného spektrogram popsaný v bodě 1.2.5. Jsou za sebou v čase poskládány vektory nesoucí informaci o spektru. Pokud jsou počítány rozdíly mezi dvěma po sobě jdoucimi vektory spektrogramu jsou získány informace o změně spektra v čase. Protože tranzienty se při fázi náběhu skládají z části z ruchové složky rozléhající se přes velkou část slyšitelného frekvenčního spektra je tak možné detektovat nástupy. Popsané metodě porovnávání vektorů se říká **spektrální tok** [21]. Pro výpočet spektrálního toku existuje více druhů přístupů lišících se předszpracováním dat a konečným zpracováním výsledků. Níže jsou popsány dvě metody výpočtu ze spektrogramu a následně z mel spektrogramu.

Výpočet spektrálního toku se provede derivací vstupního diskrétního signálu který zde představuje vektory spektrálních složek. Výpočet probíhá pro každou spektrální složku a výsledek je sečten. Popsáno rovnici

$$r(n) = \sum_{k=0}^K |X|(n+1, k) - |X|(n, k) \quad (1.14)$$

$$\Delta S_t(n) = \frac{r(n) + |r(n)|}{2} = \begin{cases} r, & \text{if } r \geq 0 \\ 0, & \text{if } r < 0 \end{cases} \quad (1.15)$$

kde $n \in \mathbb{Z}$ a K je počet spektrálních složek v jednom vektoru.



Obr. 1.9: Výpočet spektrálního toku pro nahrávku piana **a)** Okamžitá výchylka nahrávky **b)** Spektrogram nahrávky **c)** Spektrální tok bez komprese **d)** Spektrální tok s kompresí spektra $\gamma = 1$

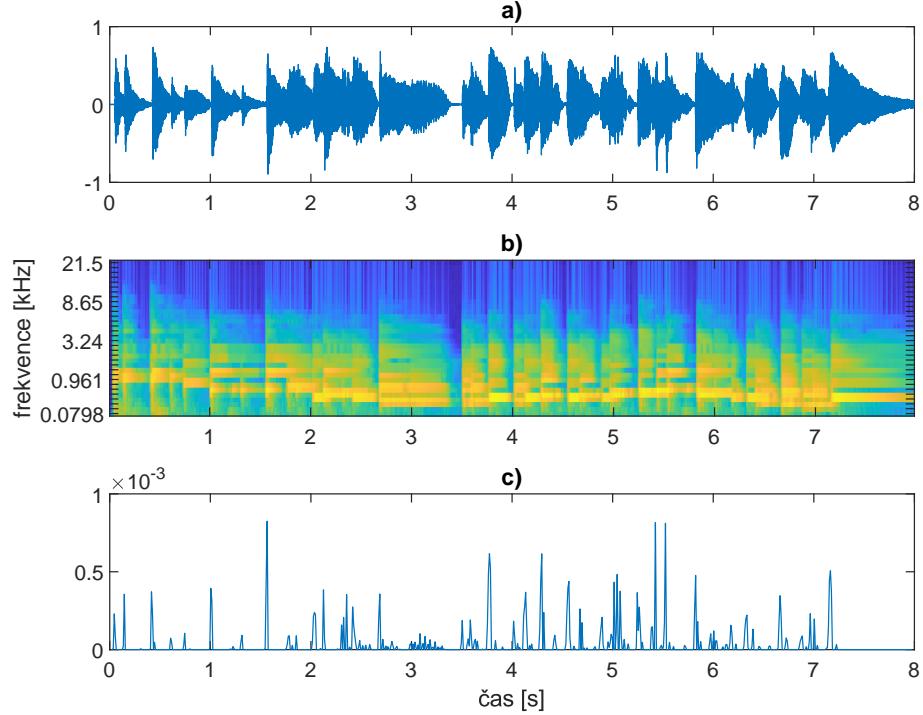
Pro dosažení přesnějších výsledků je možnost komprese spektrogramu. Funkce pro logaritmickou kompresi singálu je zapsána jako

$$X_c = \log(1 + \gamma|X|) \quad (1.16)$$

kde pro správnou kompresi je potřeba aby $\gamma \geq 1$. Míra komprese je určena velikostí γ . Nižší hodnoty jsou vhodné pro lepší detekci perkusivních zvuků. Při vyšších hodnotách γ dochází k větší kompresi spektrogramu a jsou zřetelnější zvuky o nižší intenzitě. Při velké kompresi dochází také k zvýraznění ruchových složek signálu [21].

Například v článku z ISMIR 2021 [16] je spektrální tok počítán s mel spektrogramu popsaného níže. Protože lidské ucho vnímá výšku frekvence logaritmicky je jednodužší rozoznat rozdíl mezi frekvencemi u nižší části spektra než u vyšších frekvencí. Například rozdíl mezi 500 Hz a 1 kHz je pro lidské ucho dobře detekovatelný a zřetelný. Odlišné rozlišovací schopnosti výšky tónu v různých frekvenčních pásmech jsou způsobeny biologickým rozvržením vnitřního ucha. Proto pro lepší rozlišování na základě lidského vnímání zvuku vznikla Melova stupnice. **Melova stuopnice**

je vjemová škála výšek, které posluchač posoudil jako od sebe stejně vzdálené. Referenčním bodem je pak definováno 1000 mel který odpovídá frekvenci 1000 Hz [29]. Výpočet spektrálního toku je následně stejný jak pří výpočtu ze spektrogramu. Výslednou křivku je možné vidět na obr. 1.10.



Obr. 1.10: Výpočet spektrálního toku z mel spektrogramu pro nahrávku piano a) Okamžitá výchylka nahrávky b) Mel spektrogram nahrávky c) Spektrální tok

1.3.3 Detekce periodicity

Detekce periodicity je nezbytnou součástí procesu při detekci dob skladby. Důležitou technikou pro detekci periodicity je autokorelační funkce. Autokorelační funkce slouží pro detekci periodických vzorů v datech. Tato metoda dokáže detektovat tempo, ale neumožnuje určit o jakou dobu se jedná. Základní použití je výpočet autokorelační funkce s energií signálu z bodu 1.3.1. Matematicky je pak popsána dle rovnice 1.17 [11].

$$r(n, i) = \sum_{u=-(T_w/2)+1}^{T_w/2} E(n+u)E(n+u-i) \quad (1.17)$$

Autokorelační funkce zde přináší několik výhod. Jednoduchost a efektivita. Je to jednoduchá metoda pro zjistění periodicity v signálech a vyžaduje menší množství paměti než jiné metody. Autokorelace může odhalit ne zcela zřejmé periodické vzory,

které nemusí být při poslechu nebo pozorování okamžitě zřejmé. Zároveň je robustní vůči šumu. Při analýze také nedochází ke ztrátě informace [13].

Autokorelace má také své nevýhody. Není vhodná pro analýzu v reláném čase protože potřebuje dostatečné množství dat a je potřebné aby v analyzovaném datovém okně byla obsažena celá perioda. V opačném případě nedojdete k jejímu nalezení. Autokorelace není schopná pracovat s fází signálu. Pro rozpoznání fáze je zapotřebí dalších metod. Nekonstantní tempo v analyzovaném signálu představuje pro autokorelační analýzu problém [13].

Při detekci periodicity je možné využít také banky rezonátorů hřebenového filtru s konstantním poločasem [24] nebo fázově blokující rezonátory [12].

1.4 Analýza chroma vlastností

1.5 Segmentace

1.6 Klasifikace žánrů a nálady

[1]

[15]

1.7 Dostupná řešení

Díky vědecké základně v oblasti music information retrieval vznikají volně šířitelné knihovny umožňující snadné použití v programovacím jazyce python. Tyto knihovny jsou publikovány za účelem usnadnění práce v oblasti MIR. Cílem některých knihoven je usnadnit přechod výzkumných týmů do programového jazyka Python a zakomponovat moderní praktiky softwarového vývoje. Díky tomu se staly tyto metody dostupné širší komunitě vědců [19]. V této kapitole je popsáno několik rozšířených knihoven poskytující metody z oblasti MIR.

1.7.1 Librosa

Knihovna Librosa vznikla v roce 2015 na základě potřeb vědecké komunity pro snadné použití metod z oblasti MIR v jazyce python. Publikována a popsána byla článkem Audio and Music Signal Analysis in Python [19] na konferenci SciPy 2015. Jedná se o otevřenou knihovnu jejiž další vývoj probíhá zapomoci vědecké komunity

na GitHubu kde je kladen důraz na kompatibilitu, řádnou dokumentaci obsahující popisy funkcí s příkladovými kódy, a testování funkčnosti. Knihovna obsahuje pokročilé metody pro detekci dob.

Librosa obsahuje funkce pro extrakci audio vlastností. Pro tuto práci jsou důležitými funkcemi zejména detekce tempa a dob skladby, rozpoznání začátků tónů. Následně jsou v knihovně obsaženy nástroje pro vizualizaci získaných vlastností. Například zobrazení spektrograma či grtafické zobrazení detekovaného tempa a dob. Funkce pro detekci tempa a dob skladby pomocí dynamického programování je v knihovně zapsána:

```
tempo, beats = librosa.beat.beat_track(y=y, sr=sr)
```

Kde y je analyzovaný signál a sr je vzorkovací frekvence signálu. Detekce dob je ve funkci `beat_track()` realizována pomocí dynamického programování. Proces je rozdělen na 3 etapy popsané Danielem P.W. Ellis ve článku Beat Tracking by Dynamic Programming [9]. Jedná se o tyto kroky:

1. Výpočet obálky síly nástupů
2. Globální odhad tempa
3. Porovnání obálky síly nástupů s globálním tempem

Výpočet obálky síly nástupů: Při výpočtu obálky síly nástupů je vstupní signál převzorkován na 8 kHz. Poté je vypočítána STFT s délkou okna 32 ms a 4 ms mezerou mezi okny. Vypočítaný spektrogram je převeden na mel spektrogram o 40 mel pásmech. Mel stupnice je využívána ve snaze vyrovnat důležitost každého frekvenčního pásma v návaznosti na logaritmickém vnímání frekvencí lidským uchem. Mel spektrogram je převeden na dB a následně jsou počítány diferenční rovnice pro každé mel pásmo. Principem půlvlnného usměrnění jsou výsledné negativní hotnoty změněny na nulové hodnoty kladné jsou v daném čase sečteny napříč všemi pásmi. Výsledný signál je filtrován horní propustí s mezní frekvencí 0,4 Hz a vyhlazen pomocí konvoluce s Gaussovou křivkou s šírkou okolo 20 ms. Pomocí výše popsáного procesu je získána jednorozměrná obálka síly nástupů v závislosti na čase reprezentující proporcionální nárůst energie přibližně sečtené ve mel pásmech.

Globální odhad tempa: Globální odhad tempa je určen z obálky síly signálu $O(t)$ získané v předchozím kroku. Je počítána autokorelace mezi původní obálkou signálu $O(t)$ a jejími zpožděnými verzemi. Pro zpoždění ve kterých se potká velké množství vrcholů obálky nastává velká korelace. Tato korelace může nastat v celočíselných násobcích dané hodnoty zpoždění. Díky tomuto jevu je těžké určit který čas zpozdění představuje správné tempo skladby. Avšak lidmi vnímané tempo skladeb má sklon být kolem 120 BPM, jak bylo zjištěno ve výzkumu popsaném v článku Ambiguity in Tempo Perception: What Draws

Listeners to Different Metrical Levels? [20]. Proto je na výslednou autokorelací aplikováno váhové okno snižující hodnotu autokorelace se vzdáleností od 120 BPM. Čas ve kterém autokorelace dosahuje největší hodnoty je čas hledaného tempa skladby.

Porovnání obálky síly nástupů s globálním tempem: Matematicky je tento krok formulován rovnicí 1.18 kde $\{t_i\}$ je pole nalezených dob, $O(t)$ je obálka síly signálu, α je váhový koeficient určující důležitost mezi obálkou síly signálu a detekovaným tempem. Funkce $F(\Delta t, \tau_p)$ zapsaná rovnicí 1.19 měří konzistenci mezi skutečným časem od sebe vzdálených dob Δt a ideálním časem mezi dobami τ_p určeným cílovým tempem.

$$C(\{t_i\}) = \sum_{i=1}^N O(t_i) + \alpha \sum_{i=2}^N F(t_i - t_{i-1}, \tau_p) \quad (1.18)$$

$$F(\Delta t, \tau) = -(\log \frac{\Delta t}{\tau})^2 \quad (1.19)$$

1.7.2 Madmom

1.7.3 Aubio

1.7.4 Hodnocení extrakce informací

1.8 Systém Spectoda

Systém Spectoda je vytvořen a spravován společností Light Seekres s.r.o. sídlící v Praze. Spectodu představují jako nový komunikační protokol pro bezdrátové ovládání světelncích produktů [27]. Pro převod digitálních příkazů v podobě Spectoda-kódu na světelné animace je využíváno procesorů ESP32 a jejich varianty umístěné ve vlastních obvodech speciálně navržených pro různé problematiky využití. Procesory následně ovládají připojené světelné zdroje nejčastěji pak adresovatelné led pásky osazené čipy WS2812B nebo WS2815.

Spectoda kód představující animaci je zároveň uložen na plošené desce. Proto v případě ztráty signálu se zdrojem kódu, může světelné zařízení nadále pracovat. Tím se systém stává výhodnější oproti rozšířenému protokolu DMX⁷, u kterého je nutný nepřetržitý tok příkazů.

Systém spectoda využívá pro generování animací několik základních animací. Tyto animace následně mohou být mezi sebou odčítány sčítány a skládány do bloků animací. Animace i bloky je možné posouvat a skládat zasebe či přes sebe v čase. Typickým příkladem jsou animace pro led pásky tvorící unikátní efekty v rámci

⁷Digital Multiplexer - Protokol pro digitální přenos informací a řízení scénických světel [32].

architektonického osvětlení či přenosných světel. Animace je možné synchronizovat s hudebním podkladem, takže se hodí také pro scénické osvětlení v rámci kreativních vystoupení.

1.8.1 Tvorba animace

Pro tvorbu animací je vytvořeno Spectoda studio [27] dostupné ve webovém prohlížeči. V rámci Spectoda studia jsou dostupné nástroje pro tvorbu animací, jejich následné nahrání do fyzického kontroleru a přehrání animace. Tvorba animace probíhá pomocí blokového programování. Animace vzniká skládáním základních bloků. Bloky je možné skládat do komplexních struktur a kombinovat mezi sebou. Z bloků je generován Spectoda Code, který je kompilátorem kompilován do řad hexadecimálních čísel. Základními bloky jsou:

TODO: Místo tohoto seznamu obrázek ze studia

„solid color“
„rainbow cycle“
„transition“
„shot“
„loading“
„color cycle“
„gradient“

Práce s led pásky

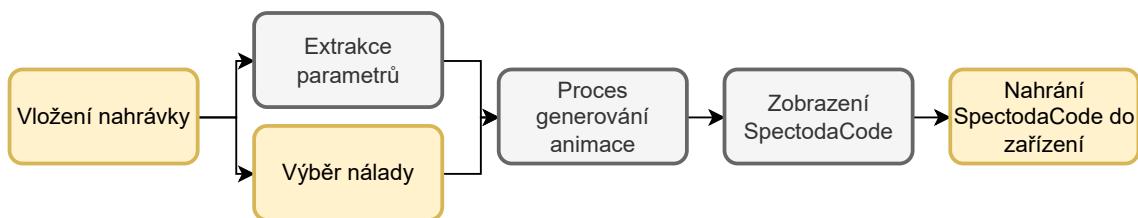
2 Výsledky studentské práce

2.1 Návrh výsledného systému

V kapitole je popsán návrh výsledného systému. Skládá se z několika částí: uživatelského rozhraní, algoritmů pro získání parametrů z hudební nahrávky, algoritmu generujícího spectoda kód na základě získaných informací a databáze bloků. Podkapitoly podrobně popisují návrh každé části systému.

2.1.1 Uživatelské rozhraní

Jedná se o jednoduché webové rozhraní, ve kterém uživatel vloží hudební skladbu ve formátu .wav nebo mp3. Rozhraní obsahuje pole pro vložení cesty ke skladbě umožňující výběr ze souborů v uživatelském úložišti. Níže se nachází posuvník se čtyřmi základními hodnotami pro výběr nálady `mood`. Jedná se hodnoty „chill“, „hang out“, „feeling happy“ a „dancing“, které jsou v tomto pořadí umístěny na posuvníku. Uživatel může pomocí posouvání posuvníku vybrat pro jakou náladu chce vytvořit animaci. Pod posuvníkem pro výběr nálady se nachází tlačítko spouštějící proces generování SpectodaCodu. Poslední částí webového rozhraní je textové pole, ve kterém se zobrazí vygenerovaný kód. Na blokovém schématu 2.1 je zobrazen proces postupu uživatele skrze webové rozhraní.



Obr. 2.1: Blokové schéma postupu uživatele webovou aplikací

2.1.2 Parametry hudební nahrávky

Systém pro generování spektoda kódu popsáný v bodě 2.1.4 vyžaduje vstupní data o hudební nahrávce. Tyto data jsou rozdělena na 7 parametrů. Každý z parametrů představuje vlastnost analyzované nahrávky. Tyto vlastnosti jsou získány pomocí technik popsaných v bodě 2.2. Jednotlivé vlastnosti a jejich datové struktury jsou shrnutы v následujících bodech.

Detekce dob – představuje pole hodnot jehož délka je závislá na době trvání nahrávky. Jednotlivé hodnoty pak udávají časy hudební nahrávky, ve kterých

se doby nacházejí. V rámci detekce dob je vypočítána i obálka síly nástupů jednotlivých not. Jedná se o pole hodnot síly nástupů pro jednotlivé snímky.

Tempo skladby – je číslo typu float s jednotkou BPM vyjadřující počet úderů za minutu. Hodnota BPM se typicky vztahuje k počtu čtvrtových not za minutu [2]. Vybraný algoritmus z knihovny Librosa neumožnuje rozeznat o jaké doby se jedná. Tempo je určeno jako počet detekovaných úderů za minutu. Postup zjištění tempa je popsán v bodě 1.7.1.

Chromavektory – jsou získány v podobě 2D pole. Počet řádků pole je dán dvacáti púltóny. Délka pole je závislá na délce nahrávky a velikosti okna při výpočtu STFT. Z chromavektorů je spočítána paleta barev dané nahrávky. Paleta barev je uložena jako pole dvacáti hodnot. Pro každý púltón jedna barva. Generování barev je popsáno více v bodech 2.3.3 a 2.3.2.

Hlasitost – představuje normalizovanou vnímanou úroveň hlasitosti skladby s ohledem na vlastnosti lidského ucha popsané psychoakustikou. Je počítána pro celou skladbu, nebo její část/segment a je značena v jednotkách LUFS Metoda výpočtu hlasitosti je popsána v bodě 1.2.6.

Segmentace – rozděluje nahranou skladbu do segmentů, ve kterých se objevují stejné hudební tvary. Segmenty jsou uloženy jako pole časů hranic segmentů. Kde hranice představuje konec jednoho segmentu a začátek druhého.

Žánr – je pro zvolenou nahrávku uložen v proměnné `genre` pomocí slovníku 10 žánrů, ke kterým je přiřazena jejich pravděpodobnost. Protože se v hudbě vyskytuje velké množství žánrů, které se mezi sebou prolínají nejde s přesnou jistotou zařadit každou nahrávku přesně k jednomu žánru. Proto je využito hodnot pravděpodobnosti. Použité žánry jsou popsány v úryvku kódu 2.1.

```
# Song genre constants
BLUES      = "blues"
CLASSICAL  = "classical"
COUNTRY    = "country"
DISCO      = "disco"
HIPHOP     = "hiphop"
JAZZ       = "jazz"
METAL      = "metal"
POP        = "pop"
REGGAE     = "reggae"
ROCK       = "rock"
```

Výpis 2.1: Hodnoty proměnné `genre`

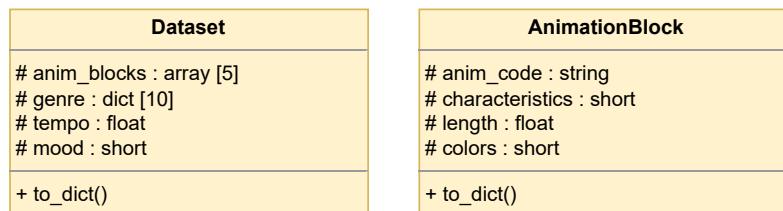
Nálada – představuje proměnnou `mood` typu float s přednastavenými statickými hodnotami zobrazenými v části kódu 2.2. Náladu zadává uživatel při nahrávání zvolené skladby.

```
# Song mood constants
CHILL      = 0
HANG_OUT   = 1
HAPPY       = 2
DANCING    = 3
```

Výpis 2.2: Hodnoty proměnné `mood`

2.1.3 Databáze bloků animací

Jak je popsáno v bodě 1.8 systém Spectoda obsahuje základní animace, které jsou skládány do bloků a tím vznikají komplexní struktury. Systém využívá již připravených bloků animací, které jsou uloženy jako objekty. Těmto objektům jsou přiřazeny parametry `code`, `characteristics`, `length` a `colors`. Jejich struktura je zapsaná datovou třídou `AnimationBlock`. Z objektů třídy jsou dále tvořeny datasety zapsané třídou `Dataset`. Tyto instance obsahují 5 bloků animací a parametry `genre`, `tempo` a `mood`. UML diagramy tříd jsou zobrazeny na obrázku 2.2.



Obr. 2.2: Blokové diagramy tříd `Dataset`, `AnimationBlock`

Jak bylo zmíněno bloky animací mají proměnnou `characteristic`, která je rozděluje do 5 kategorií. Každá kategorie má svůj specifický význam:

- BANG – Vhodné pro začátek krátkých segmentů. Velmi úderné a rychlé.
- SHOT – Rychlé animace pro kratší segmenty.
- PULL – Úderné animace, které vtáhnou do děje na začátku delších segmentů.
- THEME – Primární animace pro delší segmenty obsahuje hlavní téma.
- FLOW – Dlouhé pomalé animace pro vyplnění klidných částí skladby.

```

# Animation characteristics constants
BANG      = 0
SHOT      = 1
PULL      = 2
THEME     = 3
FLOW      = 4

```

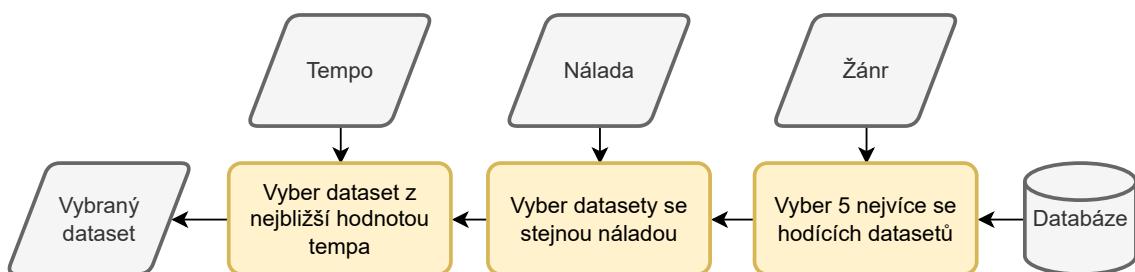
Výpis 2.3: Hodnoty proměnné `characteristics`

2.1.4 Systém pro generování animací

Systém pro generování animací je jádrem práce. Jeho struktura udává vizuální kvalitu animací a schopnost přizpůsobit se různorodým skladbám. V této kapitole je popsána jeho vnitřní logika.

První částí systému je vstupní rozhraní, ve kterém jsou přijímány data obsahující vlastnosti hudební nahrávky. Struktura přijímaných dat je popsána v bodě 2.1.2. Každý ze zmíněných parametrů plní důležitou funkci v rozhodovacím procesu skládání bloků animace. Níže jsou popsány rozhodovací funkce pro jednotlivé parametry.

Žánr a nálada – jsou používány pro výběr vhodného balíčku animací. Tyto balíčky jsou nazývány datasety a jejich datová struktura je popsána v bodě 2.1.3. Postup výběru datasetu je následující. Každý dataset obsahuje konstantní hodnotu `mood` a seznam všech deseti používaných žánrů. Ke každému žánru je přiřazeno racionální číslo udávající hodnotu jak moc je dataset pro daný žánr vhodný. Kde 1 je nejvíce vhodný a 0 nejméně. Na základě seznamu `genre` dochází k postupnému seřazení všech, datasetů dle jejich vhodnosti pro nahraný audio soubor, a výběru 5 nejvíce se hodících. V dalším kroku jsou vyřazeny všechny datasety, které neodpovídají zvolené náladě `mood`. Programové řešení výběru vhodného datasetu je popsáno v bodě 2.3.4.



Obr. 2.3: Blokový diagram výběru datasetu.

Tempo skladby – je posledním parametrem u výběru vhodného datasetu. Všechny zbývající datasety jsou porovnány s odhadovaným tempem skladby a je stanoven rozdíl jejich hodnot. Jako výsledný dataset je vybrán ten, který má hodnotu rozdílu tempa nejmenší. Celý postup je znázorněn na obrázku 2.3.

Segmentace – má důležitou roli při výběru vhodných bloků animací. U každého ze segmentů je počítána jeho délka a hlasitost tyto údaje jsou pak porovnávány vůči celé skladbě. Na základě těchto údajů jsou segmenty děleny na: hlasité, tiché, krátké a dlouhé. Práce se segmenty v programu je popsána v bodě 2.3.5

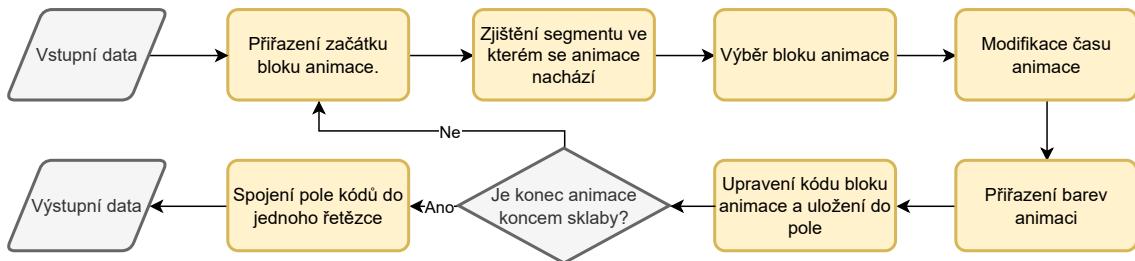
Detekce dob – udává časy, kde se v nahrávce nacházejí doby. V aplikaci je tento parametr důležitý zejména pro správné umístění začátku a konce animace, tak aby seděla do rytmu.

Obálka síly nástupů – z anglického pojmu onset strength envelope, je hodnota na základě které je určena významnost jednotlivých dob. Tato významnost je brána jako nejvyšší hodnota obálky síly nástupů v okolí dané doby.

Chroma vektory – udávají tónovou strukturu skladby v průběhu času. Tento parametr je využit pro nastavení barev animace. Chromavektory se analyzují a jsou vybrány 3 hlavní tóny nejčastěji se znějící v nahrávce. Každému z tónů je přiřazen barevný odstín tak, aby se jednalo o barvy triadické. Zbývajícím tónům jsou přiřazeny odstíny těchto barev a jsou uloženy jako barevná paleta nahrávky. U jednotlivých animací jsou na základě počtu barev v údaji `colors` z chromavektorů vypočítány nejvíce se vyskytující tóny. Na základě těchto tónů jsou animaci přiřazeny barvy z palety barev. Celý proces generování a přiřazování barev je popsán v bodě 2.3.3.

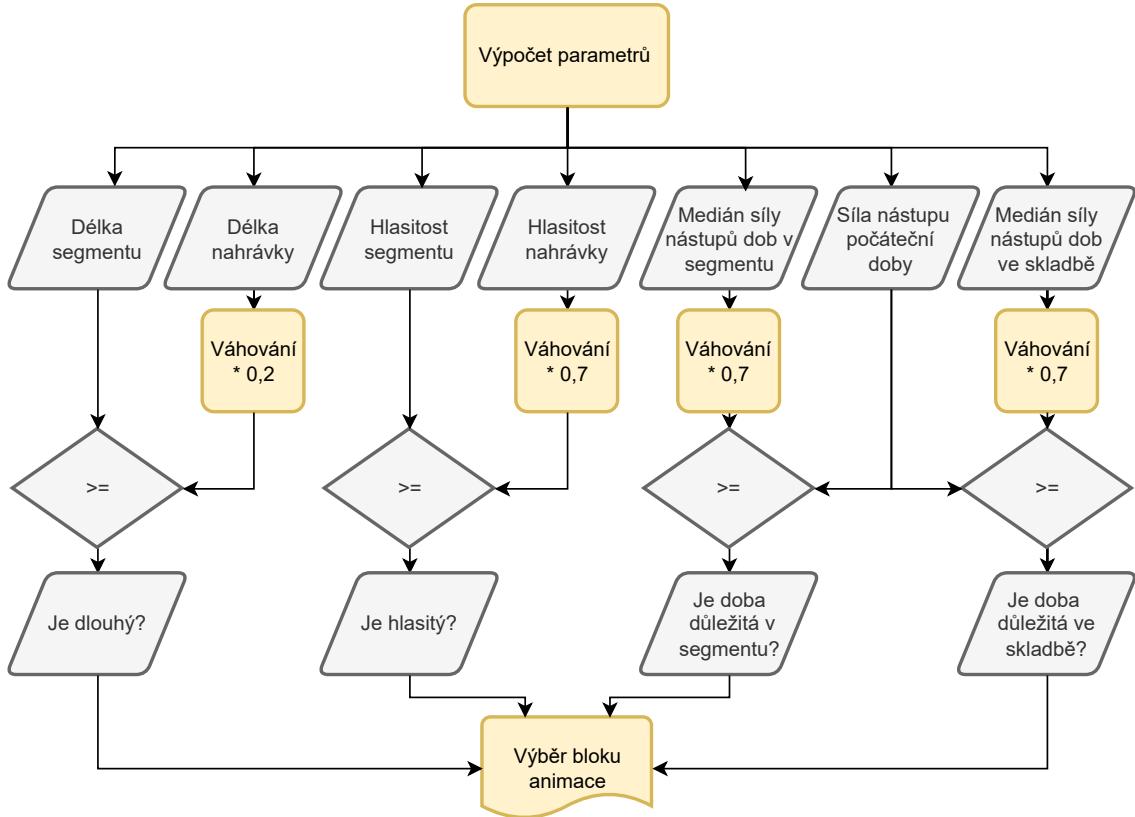
Hlasitost – pomáhá procesu segmentace. Refrén a sloka se mohou lišit hlasitostí. Hlasitost je počítána pro celou skladbu a následně pro segmenty. Při výběru vhodného bloku animace dochází k porovnání hlasitosti segmentu vůči hlasitosti nahrávky. Na základě porovnání je určeno jak výrazné mají být animace v daném segmentu. Vyšší hodnoty představují rychlejší a údernější animace. Nižší hodnoty naopak pomalejší a klidnější animace.

Druhá část systému tvoří samotnou logiku skládání bloků animací. Rozhodovací struktura lze zjednodušit na několik po sobě jdoucích kroků znázorněných blokovým diagramem na obrázku 2.4. V bodech níže jsou postupně popsány všechny kroky.



Obr. 2.4: Blokový diagram procesu generování animací

- Prvním z bloků je přiřazení času začátku animace. Pokud se jedná o počáteční animaci ve skladbě je automaticky přiřazen jako začátek čas, ve kterém se nachází první doba. Pro každou další animaci je volen čas začátku jako konec animace předchozí.
- Druhým krokem je nalezení segmentu, ve kterém se začátek animace nachází. Segment je vyjádřen jako jeho hranice začátku a konce. Tyto hranice jsou nalezeny v poli hranic segmentů jako nejbližší nižší a nejbližší vyšší čas k času začátku animace. Funkce pro vyhledání segmentu je zobrazena v úryvku kódu A.1.
- Třetím krokem v pořadí je výběr vhodného bloku animace. Výběr vhodného bloku je důležitý pro výsledný pocit z konečné animace. Ze získaných parametrů o nahrávce je vypočítáno dalších 7 hodnot. Na získané hodnoty jsou aplikovány váhy a poté jsou mezi sebou porovnány. Výsledkem procesu jsou 3 proměnné typu boolean představující otázky: Je segment dlouhý? Je segment hlasitý? Je doba začátku animace důležitá v segmentu? Je doba začátku animace důležitá v nahrávce? Pro lepší představu je struktura procesu zobrazena na blokovém diagramu 2.5. Na základě získaných logických proměnných je pomocí rozhodovací struktury vybrána vhodná charakteristika animace. Výběr charakteristiky je zobrazen v úryvku kódu A.2. Struktura je tvořena dvěma kroky a v každém kroku jsou 4 řešení. Výsledný počet řešení je tedy $2^4 = 16$. První je porovnáváno jestli je segment dlouhý a hlasitý. Kombinace těchto dvou stavů (ano/ne) tvoří první čtyři zmíněné řešení. Ve druhém kroku je porovnávána důležitost počáteční doby animace v segmentu a skladbě. Toto porovnání nabývá také 4 možných řešení.



Obr. 2.5: Blokový diagram procesu výpočtu parametrů pro výběr animace

- Poté co je vybrán vhodný blok animace je důležité upravit její délku tak, aby končila opět době. Každý blok má nastavenou doporučenou délku uloženou v proměnné `anim_length`. Nejprve je vypočítán čas konce animace jako součet začátku animace a její doporučené délky. Pomocí funkce `Find_nearest_beat` zobrazené v úryvku kódu A.3 je nalezena doba, která má nejmenší časový rozdíl od vypočítaného konce animace. Nyní je známa doba na které bude animace končit, ještě je ale potřeba vypočítat modifikátor, který upraví délku animace tak, aby na ní skutečně skončila. Modifikátor je číslo, kterým je násobena doporučená délka animace tak aby odpovídala délce mezi vybranou počáteční dobou a koncovou dobou. Výpočet je znázorněn rovnicemi 2.1, kde M je modifikátor, T_a je doporučený čas animace, T_n je nový čas animace, T_{eb} je čas ve kterém se nachází konečná doba animace a T_{sb} je čas ve kterém se nachází počáteční doba animace.

$$M = \frac{T_a}{T_n} = \frac{T_a}{T_{eb} - T_{sb}} \quad (2.1)$$

- Nyní je možné vybrat barvy s palety barev dané nahrávky. Nejprve jsou z chromavektorů vypočítány tóny, které zní nejdéle v časovém rozmezí bloku animace. Z proměnné `anim_color` uložené v bloku animace zjistíme kolik je

potřeba barev. Na základě zjištěného počtu je vybráno z palety barev vybráno n barev, které odpovídají nejvíce znělým tónům v dané části skladby.

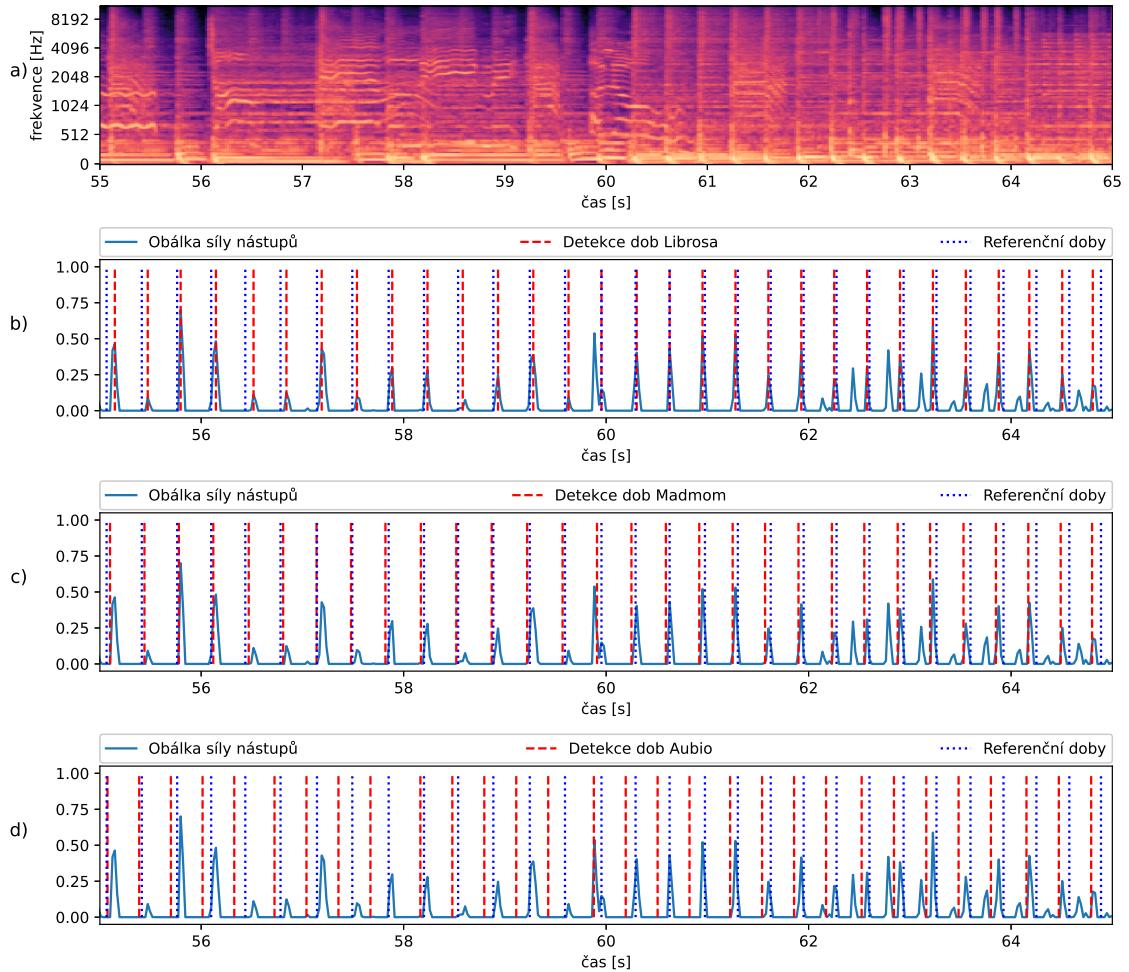
- Šestým krokem je vygenerování kódu animace. Zdrojový kód je uložen v proměnné `anim_code` vybraného bloku. Do něj je potřeba na předpřipravená místa vložit získané informace o začátku, konci, rychlosti a barvě. Po vložení těchto informací je v textovém řetězci uložen do pole jako další hotová animace.
- Posledním bodem cyklu je kontrola jestli je konec animace poslední dobou v skladbě. Pokud není cyklus se opakuje stále dokola než je vygenerován kód pro všechny části nahrávky. Po dokončení tohoto procesu je pole vygenerovaných zdrojových kódů sečteno do jednoho dlouhého řetězce jako finální kód animace a je připraven k nahrání do Spectoda zařízení.

2.2 Metody extrakce parametrů z hudební nahrávky

Díky vědecké komunitě vznikly volně dostupné knihovny obsahující techniky z oborů MIR. V této části práce jsou prozkoumány 3 knihovny zmíněné v bodě 1.7. Jsou porovnány jejich funkce pro získání parametrů z hudební nahrávky. Tyto funkce jsou mezi sebou porovnány z hlediska přesnosti výsledků, rychlosti výpočtů, jednoduchosti použití a možnosti využití pro komerční účely.

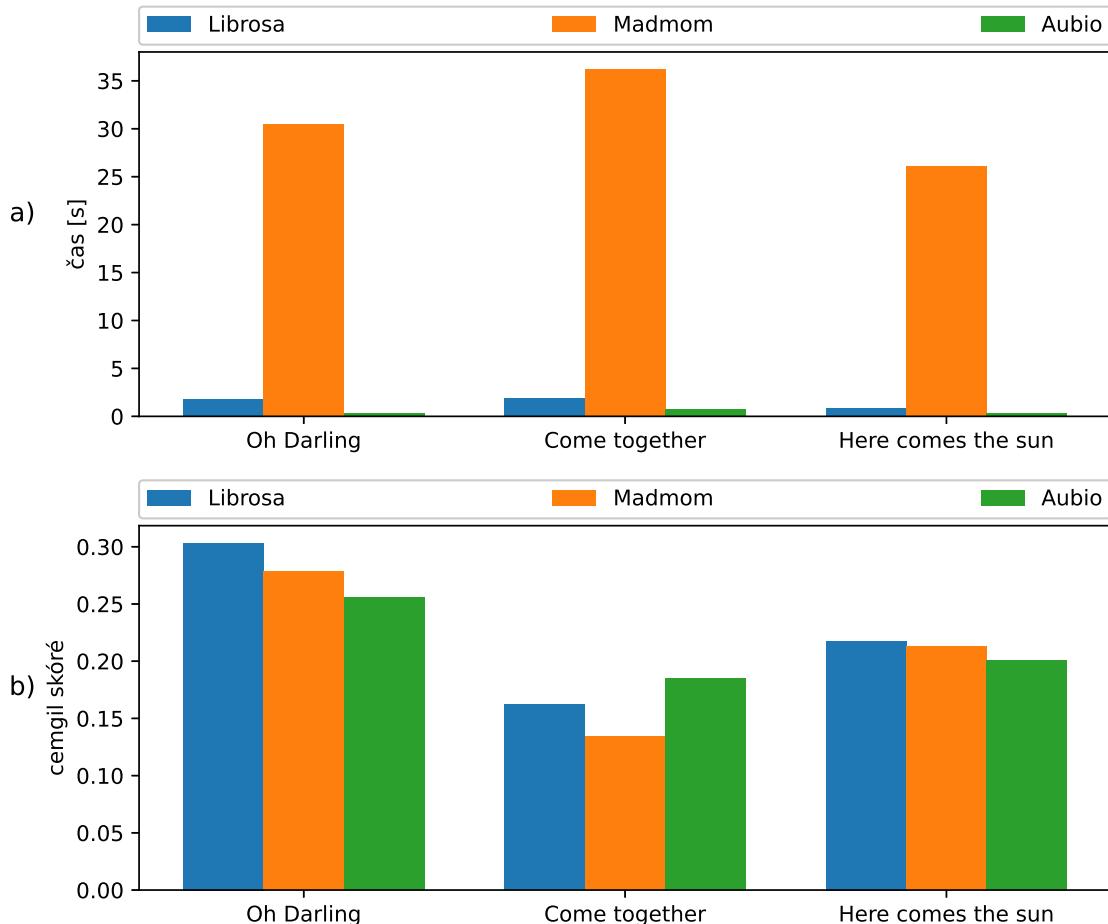
2.2.1 Detekce dob a tempa

Pro porovnání detekce dob jsou vybrány 3 funkce. Pro každou knihovnu jedna funkce. První je z knihovny Librosa funkce `beat_track`. Z knihovny Madmom je použit `BeatTrackingProcessor` a z knihovny Aubio funkce `tempo`. Funkce jsou porovnány na třech skladbách skupiny Beatles. Níže v grafu 2.6 je vidět úryvek skladby Oh-Darling!. Pro lepší zobrazení jsou osy grafu v rozsahu 55 - 65 sekund nahrávky. Na prvním z grafů je vyobrazen mel spektrogram vypočítán pomocí funkce `melspectrogram` z knihovny Librosa. Grafy b), c) a d) zobrazují v pozadí obálku sily nástupů a vertikální pruhované čáry znázorňují detekované doby dané funkce. Vertikální modré tečkované čáry jsou referenční doby zaznamenané institucí Centre for digital music na univerzitě Queen Mary, University of London [17].



Obr. 2.6: Porovnání metod detekce dob na úryvku skladby Oh-Darling!. **a)** Mel spektrogram **b)** Detekce dob pomocí Librosa **c)** Detekce dob pomocí Madmom **d)** Detekce dob pomocí Aubio

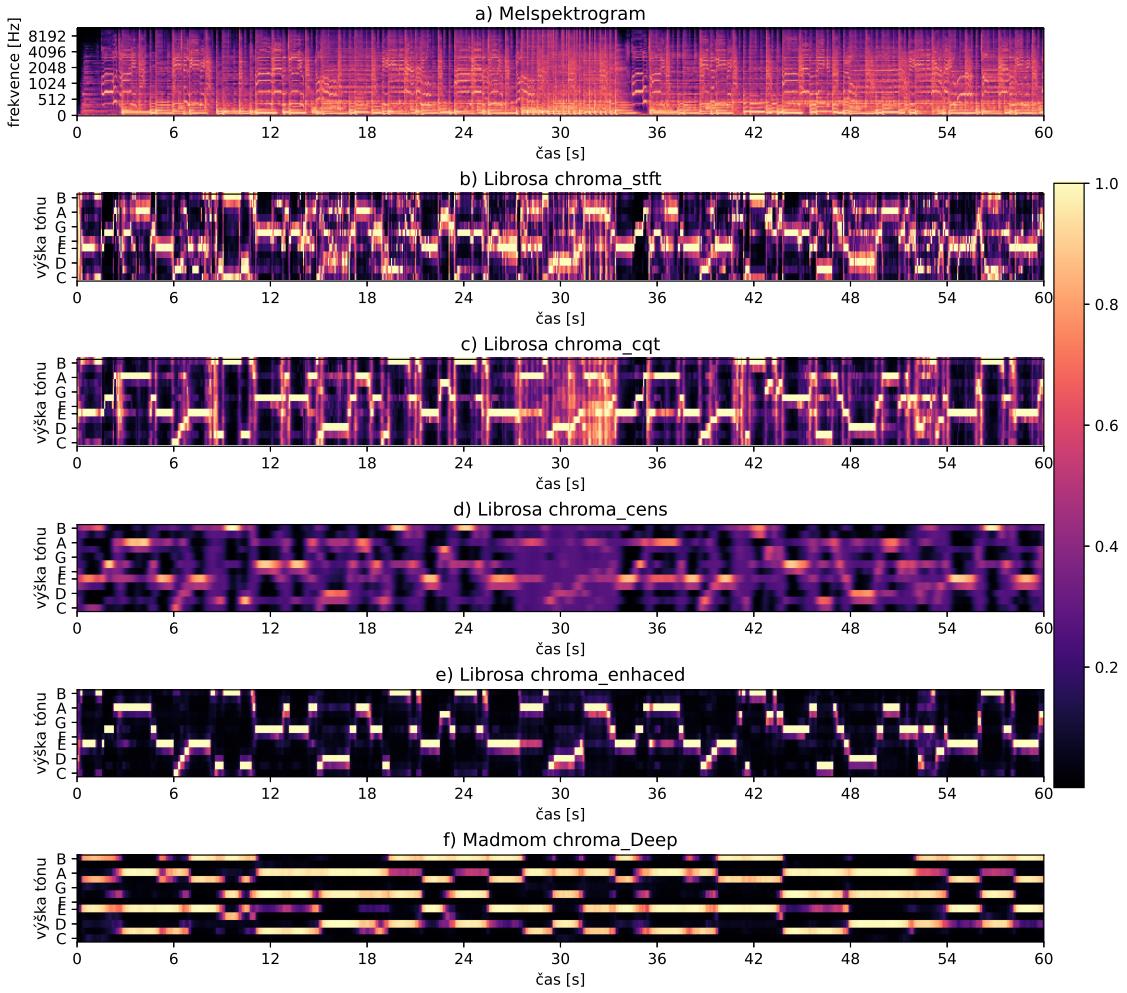
Z grafu lze vidět, že funkce z knihoven Librosa a Madmom se nejvíce přibližují referenčním dobám. Pro přesné hodnocení funkcí je využita knihovna Mir_eval poskytující funkce pro hodnocení přesnosti detekce dob. Pomocí této knihovny je počítáno Camgil skóre. Popis knihovny a výpočtu je zmíněn v bodě 1.7.4. Posledním hodnoceným parametrem je doba výpočtu. Výsledky Camgil skóre, a doby výpočtu pro jednotlivé skladby jsou zobrazeny na obrázku 2.7.



Obr. 2.7: Porovnání přesnosti a času metody detekce dob na skladbách Oh-Darling!, Come Together a Here Comes The Sun. **a)** Čas výpočtu **b)** Cemgil skóre

2.2.2 Analýza chromavektorů

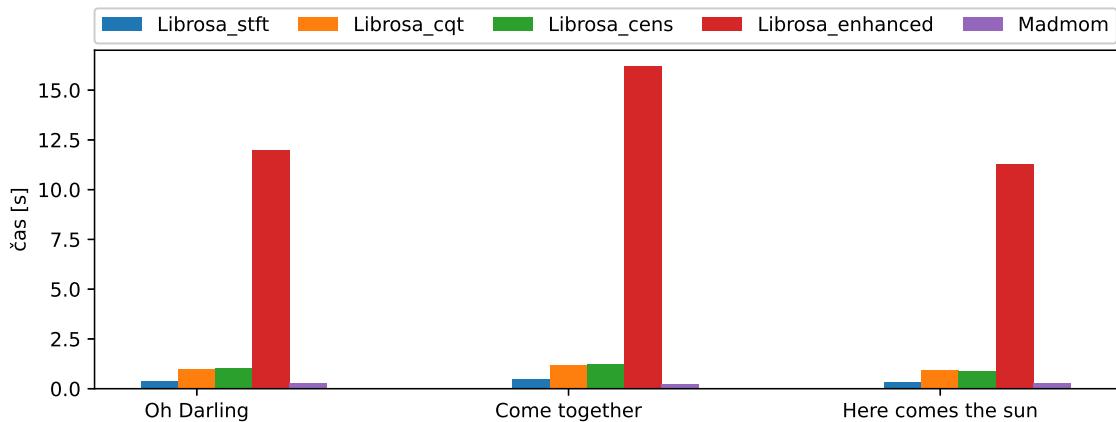
Hodnocení vhodné funkce pro výpočet chromavektorů je realizováno zejména vizuálně na zobrazených grafech referenčních skladeb. Tyto grafy lze vidět na obrázku 2.8. Porovnávány jsou 4 funkce z knihovny librosa `chroma_stft`, `chroma_cqt`, `chroma_cens` a čtvrtou je funkce `chroma_cqt` s řetězci předzpracování signálu a filtrace výsledných chromavektorů. Principy výpočtu jednotlivých funkcí jsou popsány v bodě 1.7.1. Z knihovny Madmom je použit *DeepChromaProcessor*.



Obr. 2.8: Porovnání výpočtu chromavektorů na skladbě Oh-Darling! v časovém rozmezí 0 - 60 s.

Z obrázku 2.8 lze vidět patrný rozdíl v přístupu výpočtu chromavektorů. U grafů *b), c)* a *d)* lze vidět velké množství šumu. U grafu *e)* je šum vyfiltrován přidanými metodami. Na grafu *f)* lze vidět, že knihovna Madmom a z ní použitý *DeepChromaProcessor* přistupuje k výpočtu chromavektorů odlišně. Je využito delší časové okno pro transformaci do frekvenční oblasti. Z toho je patrné, že v časovém měřítku údaj není natolik přesný jako u knihovny Librosa. Pro následné použití v algoritmu pro generování animací je však tento výsledek mnohem stálejší.

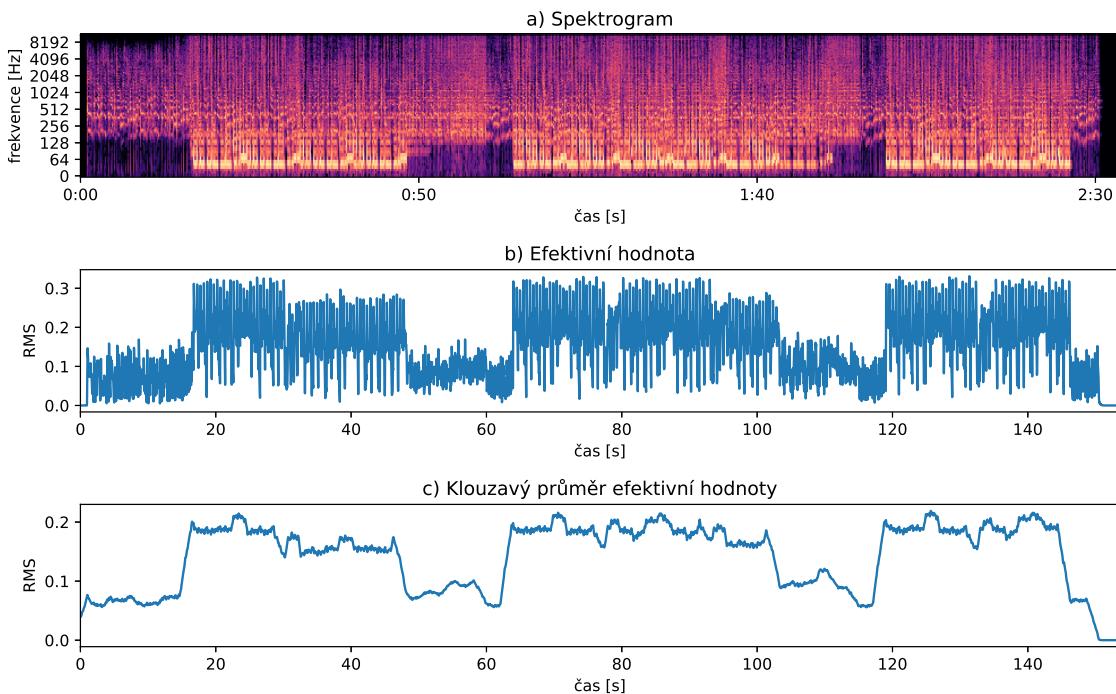
Dalším hodnoceným parametrem je délka výpočtu chromavektorů. Ta je závislá na složitosti algoritmů jež funkce využívají k výpočtu. Porovnání je zobrazeno na obrázku 2.9. Z obrázku je zřejmé, že pomocné metody pro předzpracování a filtrace chromavektorů z funkce *chroma_cqt* přidaly značné množství výpočetního času oproti samotnému výpočtu chromavektorů.



Obr. 2.9: Porovnání délky výpočtu chromavektorů

2.2.3 Efektivní hodnota signálu

Výpočet efektivní hodnoty signálu je realizován pomocí funkce z knihovny Librosa. Pro výpočet je použito okno o délce 2048 vzorků. Hodnoty jsou vycentrovány na střed rozsahu okna tedy 1024 vzorků. Získaná křivka je uhlazena pomocí výpočtu klouzavého průměru. Počet vzorků ze kterých se klouzavý průměr vypočítá odpovídá délce signálu $7,5s$ a je závislý na jeho vzorkovací frekvenci. Získané křivky jsou zobrazeny na obrázku 2.10.



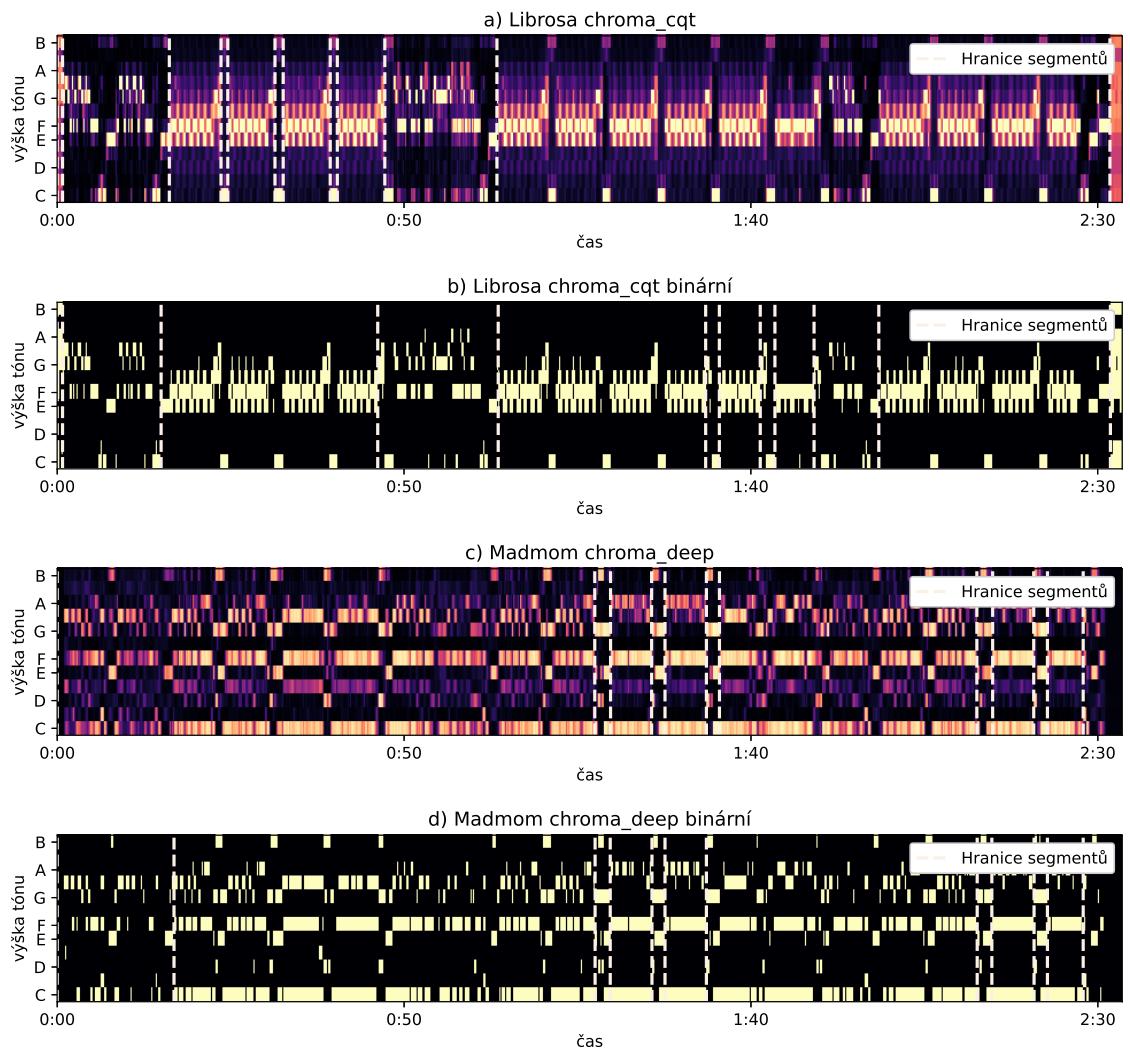
Obr. 2.10: Zobrazení efektivní hodnoty skladby Belly dancer

2.2.4 Hlasitost

Pro výpočet hlasitosti skladby nebo jejich částí je použita knihovna `Pyloudnorm` [28]. Tato knihovna implementuje standardizovaný algoritmus ITU-R BS.1770 [34]. Který je více popsáný v bodě 1.2.6. Výstupní hodnotou funkce je číselná hodnota udávající hlasitost skladby nebo jejich částí v jednotkách LUFS.

2.2.5 Segmentace

Z vybraných knihoven nabízí metody pro segmentaci nahrávky pouze knihovna librosa ve funkci `agglomerative`. Pro rozdelení na segmenty využívá jako vstupní data chroma vektory. Díky tomu je možné dosáhnout odlišných výsledků použitím různých metod pro výpočet vstupních chroma vektorů. Na obrázku 2.11 je zobrazeno využití dvou různých přístupů na vliv výpočtu dvanácti segmentů. Prvním z přístupů je porovnání chromagramů získaných zvolenými metodami z knihy librosa a madmom. Druhým přístupem je jejich přepočet na binární variantu s váhou 0,6 kde hodnoty vzorků chroma vektorů nižší nebo rovny jsou zapsány jako 0 a hodnoty vyšší jako 1. Tento proces je využit i při generování palet barev 2.3.3. Z obrázku je patrné, že největší rozdíl ve stanovení segmentů hraje právě zvolená metoda pro výpočet chroma vektorů. U binární verze pak lze vidět malé odchylky v rámci dvou až tří segmentů.



Obr. 2.11: Porovnání vlivu výpočtu chroma vektorů na výslednou segmentaci pro 12 segmentů.

2.3 Realizace

Kapitola popisuje jakým způsobem bylo dosaženo funkčního prototypu systému pro generování animací na základě parametrů hudební nahrávky. Jsou zde popsány použité programovací jazyky, metody a algoritmy. Podrobně je popsáno programové řešení jednotlivých částí systému.

2.3.1 Uživatelské rozhraní

Uživatelské rozhraní je reprezentováno webovou stránkou a je naprogramováno pomocí značkovacího jazyka HTML¹ spolu s formátováním v jazyce CSS². Aby bylo možné k webové stránce přistupovat veřejně je nasazena serveru služby PythonAnywhere. Která poskytuje cloudové řešení pro realizaci python aplikací. Funkčnost a propojení webového rozhraní s python systémem pracujícím na pozadí je řešeno pomocí frameworku Flask. Flask umožňuje jednoduchou interakci webové stránky s python aplikací běžící na pozadí pomocí protokolu HTTP³. V rámci webové aplikace je využito pouze metod GET a POST. Kde metoda POST slouží pro odeslání dat s přiřazenou nahrávkou a vybranou náladou.

2.3.2 Extrakce parametrů

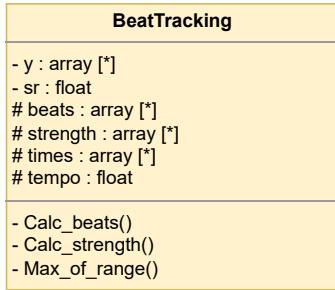
Pro získání všech potřebných parametrů z nahraného audio souboru jsou naprogramovány čtyři třídy. Tyto třídy analyzují veškeré potřebné parametry. V textu níže je postupně popsána jejich vnitřní struktura a funkce které obstarávají. Vypočítané vlastnosti jsou pak dostupné jako property dané třídy. Property v pythonu nahrazují jinde typické funkce typu Getter a Setter.

První třída BeatTracking je zaměřená na detekci síly nástupů, dob a tempa. Je tvořena s 6 parametry a obsahuje metody pro výpočet jednotlivých dob, jejich síly, tempa a časů ve kterých se nacházejí. Doby, tempo a jejich časy jsou určeny pomocí funkce `Calc_beats` a veškerá výpočty obstarává knihovna Librosa. Výpočet síly dob probíhá ve funkci `Calc_strength` kde se v okolí každé nalezené doby vezme 7 vzorků z obálky síly nástupů a z těchto vzorků je vráceno nejvyšší nalezené číslo. Díky tomu je docíleno, že je dané době přidělena největší vzorek obálky síly nástupů v jejím blízkém okolí. Zdrojový kód je zobrazen v příloze A.4.

¹HTML – Hypertext markup language. Je značkovací jazyk používaný pro tvorbu webových stránek.

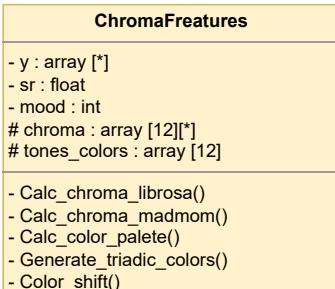
²CSS – Cascading style sheets. Je jazyk popisující styl zobrazení HTML elementů[5].

³HTTP – Hypertext transfer protocol. Je protokol navržen pro komunikaci mezi klientem a serverem.



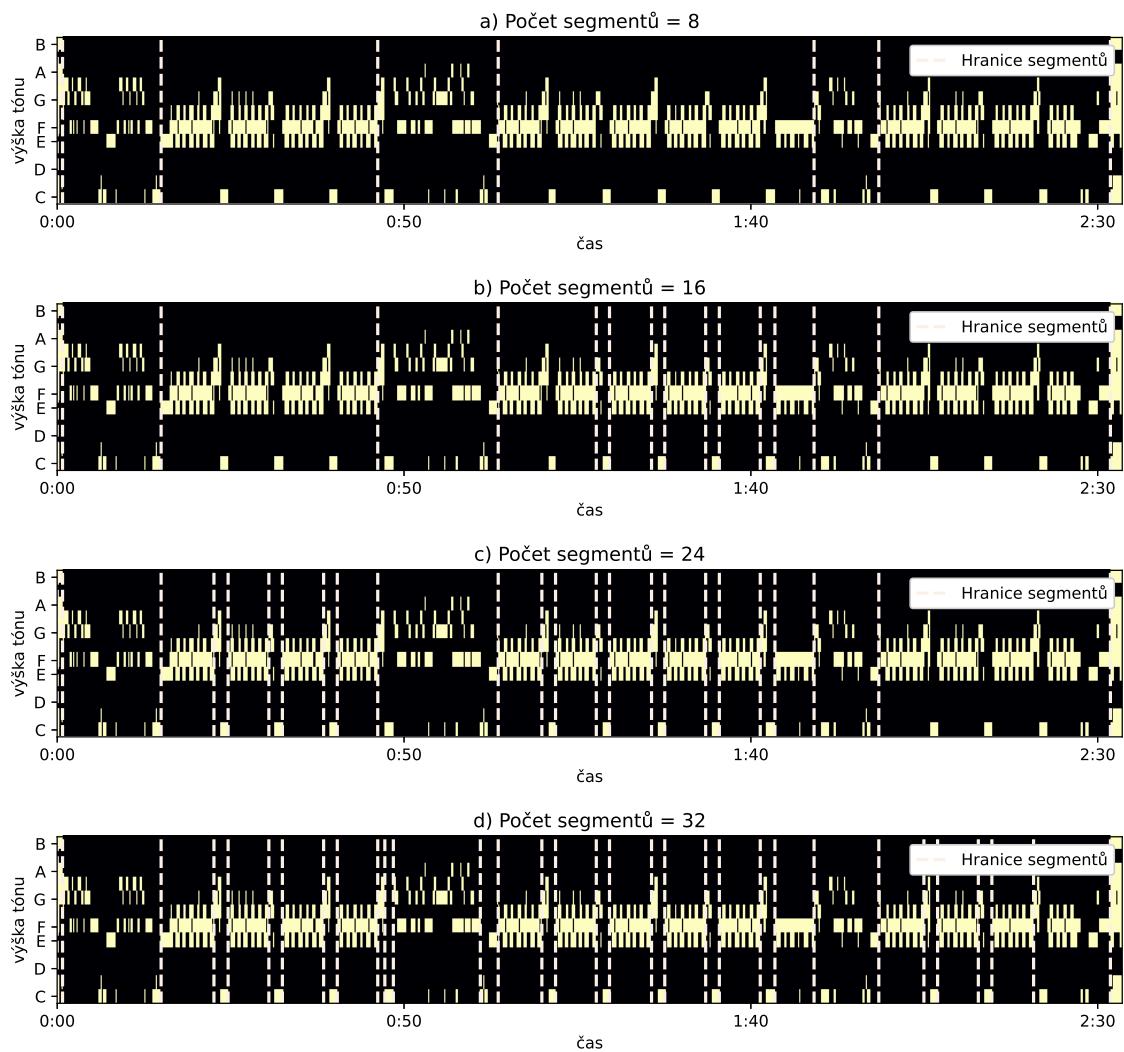
Obr. 2.12: Struktura třídy *BeatTracking*

Druhá třída s názvem *ChromaFeatures* je navržena pro získání chroma vlastností dané nahrávky. Obsahuje 4 atributy a 5 metod. Její struktura je zobrazena na diagramu 2.13. Pomocí funkcí jsou vypočteny chroma vektory a paleta barev analyzované skladby. Pro výpočet chroma vlastností jsou používány dvě funkce. První je *Calc_chroma_librosa* využívající algoritmus *chroma_cqt* s přídavným předzpracováním signálu a filtrací výsledného pole. Zmíněný výpočet je používán při zvolené náladě *mood* na hodnoty „happy“ a „dancing“. Pro hodnoty „chill“ a „hang_out“ je využíváno výpočtu pomocí knihovny madmom ve funkci *Calc_chroma_madmom*. Postup stanovení palety barev, který je naprogramován pomocí funkcí *Calc_color_palete*, *Generate_triadic_colors* a *Color_shift* je podrobně popsán v sekci 2.3.3.



Obr. 2.13: Struktura třídy *ChromaFeatures*

Pro práci se segmenty je vytvořena třída *Segmentation* a její struktura je zobrazena na obrázku 2.15. Je využíváno již vypočítaných chromavektorů, které jsou vstupními parametry funkce z knihovny Librosa. Tato funkce využívá hierarchického aglomerativního shlukování popsaného v bodě 1.7.1. Pro výpočet je nutné stanovit počet segmentů na které mají být data rozdeleny. Počet segmentů je stanoven na základně zvolené nálady. Počty jsou následující: chill – 8 segmentů, hang out – 16 segmentů, happy – 24 segmentů a dancing – 32 segmentů. Rozdíly jsou vidět na obrázku 2.14.



Obr. 2.14: Provnání počtu segmentů

Segmentation
- chroma_features : ChromaFeatures
- mood : int
bounds : array [*]
- Segments_number()
- Calc_segments()

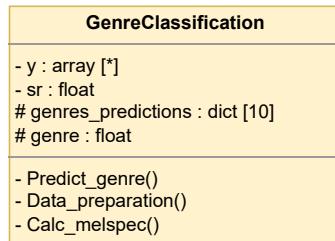
Obr. 2.15: Struktura třídy Segment

Metody pro klasifikaci žánrů jsou ve třídě `GenreClassification`. Ta se skládá ze třech parametrů a třech funkcí. Stanovení žánrů probíhá pomocí naučené neuronové sítě o struktuře Pro její vytvoření je využito balíčků TensorFlow⁴ a Keras⁵.

⁴TensorFlow je volně šířitelná knihovna pro strojové učení a umělou inteligenci.

⁵Vysokoúrovňové API rozhraní pro tvorbu neuronových sítí.

Pro neuronovou síť je nutné aby jí byly předloženy vstupní data vždy o stejné velikosti. Nejprve tedy probíhá předzpracování vstupních dat. K tomu slouží funkce `Data_preparation` a `Calc_melspec`. V prvním kroku jsou vzorky nahrávky oříznuty na délku 25 s při vzorkovací frekvenci 22050 Hz. Počet vstupních vzorků tedy je $25 * 22050 = 551250$ vzorků. Zkrácená část skladby je následně přepočtena na mel spektrogram s šírkou okna 1024 vzorků. Předzpracovaný signál je nutné převést na tenzory a poté je možné jej vložit do neuronové sítě. Výstupem modelu je tenzor o deseti hodnotách v rozmezí 0-1. Každá hodnota predikuje pravděpodobnost daného žánru.



Obr. 2.16: Struktura třídy `GenreClassification`

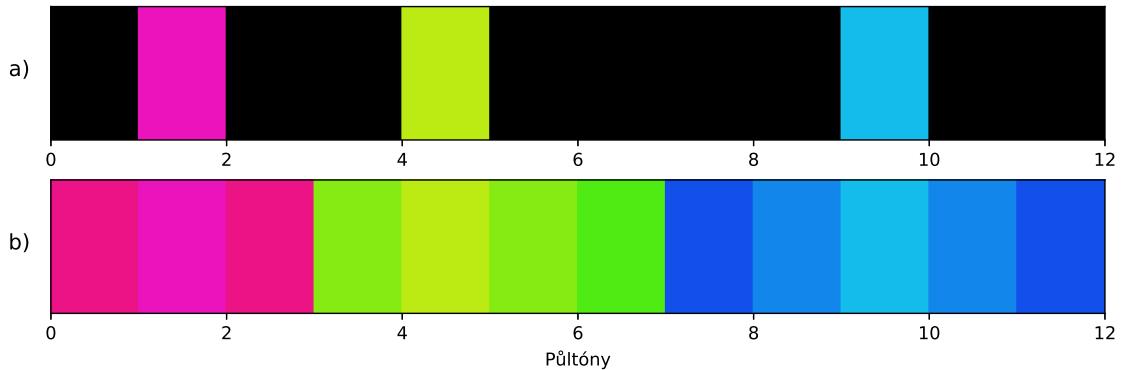
Model pro klasifikaci žánrů je vytvořen pomocí návodu „Music Classification Using Deep Learning | Python“ [1], který vychází z článku [15] a teoreticky je popsán v kapitole 1.6.

2.3.3 Paleta barev

Paleta barev znázorňuje 12 půltónů s přiřazenými barevnými hodnotami. Barvy jsou stanoveny ve dvou krocích. Jako první jsou určeny 3 základní barvy, které tvoří triadicke barevné uspořádání. Celý proces generování probíhá pomocí modelu HSV⁶. U základní barvy je náhodně vygenerovaná hodnota odstínu v rozmezí 0 až 360. Hodnoty sytosti a jasu jsou přednastavené pro všechny barvy stejně. Ve funkci `Generate_tradic_colors` jsou vygenerovány zbývající dvě barvy tak, že je hodnota odstínu posunuta vždy o 120 stupňů aby barvy odpovídaly triadickému uspořádání. Aby bylo možné základní barvy přiřadit tónům je nejprve potřeba zjistit, které tři tóny v nahrávce znějí nejčastěji. Matice chromavektorů je převedena na binární. Hodnoty větší nebo rovny 0,6 jsou zapsány jako 1 hodnoty nižší jsou zapsány jako 0. Následně jsou řádky matice sečteny. Pro každý tón je tedy získána hodnota jak často se vyskytuje. Třem tónům s nejvyšší hodnotou jsou přiřazeny základní barvy. Příklad získaných a přiřazených barev lze vidět na obrázku 2.17 a). Zbylé

⁶HSV – Hue, Saturation, Value. HSV je barevný model pro uložení informací o barvě pomocí tří hodnot tón, sytost a jas).

barvy jsou odvozeny od barvy základní pomocí posunutí hodnoty odstínu o $15 * i$ stupňů, kde i je počet půltónů od tónu se základní barvou.



Obr. 2.17: Postup generování barevné palety „a)“ Barvy základní, „b)“ Barvy zbylé

2.3.4 Výběr datasetu

Blokový diagram jak probíhá výběr datasetu je zobrazen výše na obrázku 2.3. Naprogramovaná funkce je pak k nalezení v příloze A.5. Funkce má 4 parametry, které jsou důležité pro proces nalezení datasetu. Jedná se o objekt třídy `GenreClassification`, který poskytuje slovník deseti žánrů s hodnotami pravděpodobnosti pro vloženou skladbu, objekt třídy `BeatTracking`, sloužící pro zjištění tempa skladby, a hodnotu nálady zvolenou uživatelem. Posledním parametrem je samotná databáze, která představuje list objektů typu `Dataset`. Tento list je vždy při spuštění generování načten z přiloženého textového dokumentu, kde je uložen ve formátu JSON⁷. Proces se skládá z pěti cyklů. V prvním cyklu jsou procházeny všechny datasety. Pro každý dataset je vnořen další cyklus který postupně projde všechny hodnoty slovníků s žánry a vypočítá rozdíl jejich hodnot s hodnoty získané z klasifikace žánrů. Tento rozdíl je vždy přičten k celkové hodnotě rozdílu daného datasetu a následně na konec pole `genres_difs` obsahujícího hodnoty rozdílů pro zbyvající datasety. Po dokončení prvního cyklu pole `genres_difs` obsahuje rozdíly v žánrech pro všechny datasety. Následuje druhý cyklus, který v pěti krocích ze zmíněného pole vybere 5 datasetů z nejnižší hodnotou rozdílu žánrů. V posledním cyklu je nejprve probíhá kontrola zdali dataset odpovídá zvolené náladě. Pokud podmínu splní je vypočítána hodnota rozdílu tempa mezi doporučeným tempem pro daný dataset a odhadovaným tempem pro vybranou nahrávku. Dataset s nejmenším rozdílem tempa je pak funkcí vrácen jako vybraný dataset pro vložený audio soubor.

⁷JSON - Je způsob zápisu dat nezávislý na platformě.[35]

2.3.5 Výběr bloku animace

Řešení vychází ze čtyřech získaných hodnot o audio souboru.

Délka audia je celkovou délkou audio souboru vyjádřenou v sekundách. A je vypočítána dle rovnice 2.2 kde N je počet vzorků a f_{vz} je vzorkovací frekvence.

$$t_a = \frac{N}{f_{vz}} \quad (2.2)$$

Délka segmentu – je vypočítána jako rozdíl času konce segmentu a začátku segmentu. Jednotkou jsou sekundy.

Hlasitost audia – je vypočítána z knihovny `Pyloudnorm` popsané v bodě 2.2.4. Hlasitost je dána v jednotkách LUFS.

Hlasitost segmentu – využívá stejnou funkci jako hlasitost audia akorát jsou do ní vloženy vzorky pouze z času segmentu.

Síla počáteční doby – udává hodnotu síly nástupu doby a je více popsána v bodě 2.3.2

Medián síly dob ve skladbě – je stanoven jako medián síly všech dob v nařávce.

Medián síly dob v segmentu – je stanoven jako medián síly všech dob v segmentu.

Tyto doby jsou následně mezi sebou porovnány ve čtyřech blocích. Každý blok se skládá z hodnoty váhy a funkce když. Dva bloky jsou zaměřené na daný segment a určují jak je segment dlouhý a hlasitý v porovnání s celou skladbou. Hodnoty délka a hlasitost skladby jsou před porovnáváním váhovány. Tyto váhy vytváří hranice. Následně jsou mezi sebou hodnoty porovnány pomocí dvou funkcí když. A výsledkem jsou proměnné boolean označující jestli je segment tichý/hlasitý (`is_loud`) a dlouhý/krátký (`is_long`). Zbylé dva bloky jsou zaměřeny na sílu počáteční doby animace. Tato síla je porovnána z váhovanými hodnotami mediánu síly dob ve skladbě a mediánu síly dob v segmentu. Jejich výstupy udávají označují významnost doby.

V posledním kroku jsou 4 získané boolean odpovědi porovnány mezi sebou ve funkci `Char_selection`. Logika porovnání naprogramovány funkciemi switch a case, v pythonu psané jako `match` a `case`. První primární blok porovnává proměnné `is_long` a `is_loud`. Tento blok má 4 možná řešení. V každém řešení je opět vnořena funkce `match` s dalšími čtyřmi možnými výsledky. Ve vnořeném bloku jsou porovnávány hodnoty významnosti počáteční doby animace `is_important_in_audio` a `is_important_in_segment`. Díky takto zvolené struktuře vzniká šestnáct možných řešení. Každé z řešení představuje hodnotu charakteristiky animace na základě které je vybrán blok animace.

2.3.6 Slabé stránky systému a návrhy na zlepšení

Závěr

V rámci diplomové práce byly splněny v úvodu stanovené cíle.

Literatura

- [1] Ali, M.: Music Classification Using Deep Learning | Python. *Analytics Vidhya*, 2021.
- [2] Apel, W.: *Harvard Dictionary of Music*. Harvard University Press, 1947.
URL <https://books.google.cz/books?id=Y6m0AAAAIAAJ>
- [3] Bracewell, R.: *The Fourier Transform and its Applications*. Tokyo: McGraw-Hill Kogakusha, Ltd., druh vyd n , 1978.
- [4] Cartwright, K.: Determining the effective or RMS voltage or various waveforms without calculus. ro n k 8, 01 2007.
- [5] Cederholm, D.; Marcotte, E.: *Handcrafted CSS: More Bulletproof Web Design*. USA: New Riders Publishing, prvn vyd n , 2009, ISBN 0321643380.
- [6] Cohen, L.: Time-frequency distributions-a review. *Proceedings of the IEEE*, ro n k 77, . 7, 1989: s. 941–981, doi:10.1109/5.30749.
- [7] Crocker, M.: *Handbook of Acoustics*. A Wiley-Interscience Publication, Wiley, 1998, ISBN 9780471252931.
URL https://books.google.cz/books?id=1x_RvffW-hcC
- [8] Downie, J. S.; Ehmann, A. F.; Bay, M.; aj.: *The Music Information Retrieval Evaluation eXchange: Some Observations and Insights*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ISBN 978-3-642-11674-2, s. 93–115, doi: 10.1007/978-3-642-11674-2_5.
URL https://doi.org/10.1007/978-3-642-11674-2_5
- [9] Ellis, D.: Beat Tracking by Dynamic Programming. *Journal of New Music Research*, ro n k 36, 03 2007: s. 51–60, doi:10.1080/09298210701653344.
- [10] Acoustics — Determination of sound power levels and sound energy levels of noise sources using sound pressure — Engineering methods for an essentially free field over a reflecting plane. Standard, International Organization for Standardization, B ezen 2010.
URL <https://www.iso.org/obp/ui/#iso:std:iso:3744:ed-3:v1:en>
- [11] Klapuri, A.; Davy, M.: *Signal Processing Methods for Music Transcription*. 01 2006, ISBN 978-0-387-30667-4, doi:10.1007/0-387-32845-9.
- [12] Large, E. W.; Kolen, J. F.: Resonance and the Perception of Musical Meter. *Connection Science*, ro n k 6, . 2-3, 1994: s. 177–208, doi:10.1080/

- 09540099408915723, <https://doi.org/10.1080/09540099408915723>.
URL <https://doi.org/10.1080/09540099408915723>
- [13] Lartillot, O.; Grandjean, D.: Tempo and Metrical Analysis by Tracking Multiple Metrical Levels Using Autocorrelation. *Applied Sciences*, ro n k 9, 11 2019: str. 5121, doi:10.3390/app9235121.
- [14] Lidy, T.; Rauber, A.: Music Information Retrieval. In *Handbook of Research on Digital Libraries: Design, Development, and Impact*, IGI Global, 2009, ISBN 978-1-59904-879-6, s. 448–456.
- [15] Macharla, V.; Radha Krishna, P.: Music Genre Classification using Neural Networks with Data Augmentation. *Department of Computer Science and EngineeringNational Institute of Technology Warangal, India*, 2021.
- [16] Matthew E. P. Davies, M. F., Sebastian Bock: *Tempo, Beat and Downbeat Estimation*. <https://tempobeatdownbeat.github.io/tutorial/intro.html>, 2021.
URL <https://tempobeatdownbeat.github.io/tutorial/intro.html>
- [17] Mauch, M.: Isophonic datasets. [Online; 11. 12. 2023].
URL <http://isophonics.net/>
- [18] McAdams, S.; Giordano, B. L.: 113The Perception of Musical Timbre. In *The Oxford Handbook of Music Psychology*, Oxford University Press, 01 2016, ISBN 9780198722946, doi:10.1093/oxfordhb/9780198722946.013.12, https://academic.oup.com/book/0/chapter/292611024/chapter-ag-pdf/44515461/book_34489_section_292611024.ag.pdf.
URL <https://doi.org/10.1093/oxfordhb/9780198722946.013.12>
- [19] McFee, B.; Raffel, C.; Liang, D.; aj.: librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, ro n k 8, 2015.
- [20] McKinney, M. F.; Moelants, D.: Ambiguity in Tempo Perception: What Draws Listeners to Different Metrical Levels? *Music Perception: An Interdisciplinary Journal*, ro n k 24, . 2, 2006: s. 155–166, ISSN 07307829, 15338312.
URL <http://www.jstor.org/stable/10.1525/mp.2006.24.2.155>
- [21] Müller, M.: *Fundamentals of Music Processing*. Springer International Publishing, 2015, doi:10.1007/978-3-319-21945-5.
URL <https://doi.org/10.1007%2F978-3-319-21945-5>
- [22] Müller, M.; Klapuri, A.: Chapter 27 - Music Signal Processing. In *Academic Press Library in Signal Processing: Volume 4*, Academic Press Library in Signal

- Processing*, ro n k 4, editace J. Trussell; A. Srivastava; A. K. Roy-Chowdhury; A. Srivastava; P. A. Naylor; R. Chellappa; S. Theodoridis, Elsevier, 2014, s. 713–756, doi:<https://doi.org/10.1016/B978-0-12-396501-1.00027-3>.
- URL <https://www.sciencedirect.com/science/article/pii/B9780123965011000273>
- [23] Salamon, J.; Gomez, E.: Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, ro n k 20, . 6, 2012: s. 1759–1770, doi:[10.1109/TASL.2012.2188515](https://doi.org/10.1109/TASL.2012.2188515).
- [24] Scheirer, E. D.: Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, ro n k 103, . 1, 01 1998: s. 588–601, ISSN 0001-4966, doi:[10.1121/1.421129](https://doi.org/10.1121/1.421129), https://pubs.aip.org/asa/jasa/article-pdf/103/1/588/8083614/588_1_online.pdf.
URL <https://doi.org/10.1121/1.421129>
- [25] Schreibman, S.; Siemens, R.; Unsworth, J. (edito i): *A new companion to Digital Humanities*. West Sussex, England: John Wiley & Sons Ltd, 2016, ISBN 9781118680599.
- [26] Sneddon, I.: *Fourier Transforms*. Dover books on mathematics, Dover Publications, 1995, ISBN 9780486685229.
URL <https://books.google.cz/books?id=jhpsLpRRerwC>
- [27] s.r.o, L. S.: Spectoda. [Online; 12. 12. 2023].
URL <https://spectoda.com/>
- [28] Steinmetz, C. J.; Reiss, J. D.: pyloudnorm: A simple yet flexible loudness meter in Python. In *150th AES Convention*, 2021.
- [29] Stevens, S. S.; Volkmann, J.; Newman, E. B.: A Scale for the Measurement of the Psychological Magnitude: Pitch. *The Journal of the Acoustical Society of America*, ro n k 8, . 3, 1937: s. 185–190, ISSN 0001-4966.
- [30] Strichartz, R.: *A Guide To Distribution Theory And Fourier Transforms*. World Scientific Publishing Company, 2003, ISBN 9789813102293.
URL <https://books.google.cz/books?id=YfA7DQAAQBAJ>
- [31] Syrový, V.: *Hudební akustika*. Akustická knihovna Zvukového studia Hudební fakulty AMU, Akademie múzických umění, 2013, ISBN 9788073312978.
URL <https://books.google.cz/books?id=ikrmoeEACAAJ>

- [32] United States Institute for Theatre Technology, I.: BSR E1.11, Entertainment Technology - USITT DMX512 Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories. *Entertainment Services and Technology Association*, 1999, online, 12.12.2023.
URL <http://www.jstor.org/stable/10.1525/mp.2006.24.2.155>
- [33] Tumarkin, A.: *The Decibel, The Phon and the Sone*, ro n k 64. Cambridge University Press, 1950, 178–188 s., doi:10.1017/S0022215100011919.
- [34] Union, I. T.: Recommendation ITU-R BS.1770-5. *ITU Publications*, 2023.
- [35] Wikipedie: JavaScript Object Notation — Wikipedie: Otevřená encyklopédie. 2022, [Online; navštíveno 13. 05. 2024].
URL https://cs.wikipedia.org/w/index.php?title=JavaScript_Object_Notation&oldid=20868532
- [36] Wikipedie: Musical Instrument Digital Interface — Wikipedie: Otevřená encyklopédie. 2022, [Online; navštíveno 1. 12. 2022].
URL https://cs.wikipedia.org/w/index.php?title=Musical_Instrument_Digital_Interface&oldid=21081530
- [37] WikiSkripta: Vlastnosti zvuku —. 2022, [Online; navštíveno 21. 11. 2022].
URL https://www.wikiskripta.eu/index.php?title=Vlastnosti_zvuku&oldid=458442

Seznam symbolů a zkrátek

MIR	Music information retrieval – Obor zabývající se vyhledávání informací v hudebních dílech
MIDI	Musical Instrument Digital Interface – Digitální rozhraní hudebních nástrojů
ISMIR	International Society of Music Information Retrieval – Mezinárodní sdružení pro MIR
MIREX	The Music Information Retrieval Evaluation eXchange
FT	Fourier transform – Fourierova transformace
FFT	Fast Fourier transform – Rychlá Fourierova transformace
DFT	Discrete Fourier transform – diskrétní Fourierova transformace
STFT	Short-time Fourier transform – krátkodobá Fourierova transformace
RMS	Root mean square – efektivní hodnota
BPM	Beats per minute – doby za minutu
HTML	Hypertext markup language – hypertextový značkovací jazyk
CSS	Cascading Style Sheets – kaskádové styly
DMX	Digital Multiplexer – protokol pro digitální přenos informací
LUFS	Loudness units relative to full scale
HSV	Hue, Saturation, Value – odstín, sytost, jas
HTTP	Hypertext Transfer Protocol – hypertextový přenosový protokol
JSON	JavaScript Object Notation

Seznam příloh

A	Ukázky zdrojových kódů	61
B	Některé příkazy balíčku <i>thesis</i>	66
B.1	Příkazy pro sazbu veličin a jednotek	66
B.2	Příkazy pro sazbu symbolů	66
C	Druhá příloha	67
D	Příklad sazby zdrojových kódů	68
D.1	Balíček <i>listings</i>	68
E	Obsah elektronické přílohy	72

A Ukázky zdrojových kódů

Výpis A.1: Zdrojový kód funkce `Find_segment` v jazyce Python

```
1 def Find_segment(y : list, beat_time):
2     """
3         Function that finds segment from provided list in which
4             is located the beat_time.
5
6         Parameters
7         -----
8             y : list
9                 List of times where the segments boundaries are
10                    located
11
12            beat_time : float
13                Time of the beat which is wanted to locate.
14
15        Returns
16        -----
17            start_segment : float
18                Time where the located segment starts.
19            end_segment : float
20                Time where the located segment ends.
21
22        """
23
24
25
26
27
28
29
30
31
32
y = np.asarray(y)
idx = np.abs((y - beat_time)).argmin()

if y[idx] > beat_time:
    idx -= 1

start_segment = y[idx]
try:
    end_segment = y[idx+1]
except IndexError:
    end_segment = None

return start_segment, end_segment
```

Výpis A.2: Zdrojový kód funkce Char_selection v jazyce Python

```
1 def Char_selection(is_long,
2                     is_loud,
3                     is_important_in_audion,
4                     is_important_in_segment):
5     match [is_long, is_loud]:
6         case [False, True]:
7             #Krátké úderné
8             match [is_important_in_audion, is_important_in_segment]:
9                 case [False, True]:
10                    return cns.SHOT
11                 case [False, False]:
12                    return cns.PULL
13                 case [True, True]:
14                    return cns.BANG
15                 case [True, False]:
16                    return cns.SHOT
17
18         case [False, False]:
19             #Krátké pohodové
20             match [is_important_in_audion, is_important_in_segment]:
21                 case [False, True]:
22                     return cns.PULL
23                 case [False, False]:
24                     return cns.THEMA
25                 case [True, True]:
26                     return cns.SHOT
27                 case [True, False]:
28                     return cns.PULL
29         case [True, True]:
30             #Dlouhé úderné
31             match [is_important_in_audion, is_important_in_segment]:
32                 case [False, True]:
33                     return cns.PULL
34                 case [False, False]:
35                     return cns.THEMA
```

```

36     case[True, True]:
37         return cns.SHOT
38     case[True, False]:
39         return cns.PULL
40     case[False, True]:
41         #Dlouhé tahnoucí se
42         match[is_important_in_audion, is_important_in_segment
43             ]:
44             case[False, True]:
45                 return cns.FLOW
46             case[False, False]:
47                 return cns.FLOW
48             case[True, True]:
49                 return cns.PULL
50             case[True, False]:
51                 return cns.THEMA

```

Výpis A.3: Zdrojový kód funkce `Find_nearest_beat` v jazyce Python

```

1 def Find_nearest_beat(y : list, time):
2 """
3     This function finds the nearest beat in given list.
4
5     Parameters
6     -----
7     y : list
8         List of times where the beats are located.
9     time : float
10        Time around that is searching for the nearest beat.
11
12    Returns
13    -----
14    idx : int
15        Index of finded beat in the list.
16    """
17    y = np.asarray(y)
18    idx = np.abs((y - time)).argmin()
19    return idx

```

Výpis A.4: Zdrojový kód funkce Calc_strength v jazyce Python

```
1 def __Calc_strength(self, onset_env):
2     """
3         Calculate strength of beats.
4
5         The function calculate beats strength based on onset
6             envelope in time of the beat. Function also check
7             range around the beat if there is some bigger value
8             in onset envelope.
9
10    Parameters
11    -----
12    onset_env : ndarray
13        Onset envelope
14
15    """
16    self.__strength = np.ones(len(self.__beats)) # Declaration of ones ndarray.
17    i = 0
18
19    for beat in self.__beats:
20        try:
21            index = np.where(self.__times == beat)[0] # Getting
22            a timestamp of the beat.
23            self.__strength[i] = self.__Max_of_range(int(index),
24                onset_env) # Gets a biggest onset value in
25                range around the timestamp of beat.
26        except ValueError:
27            self.__strength[i] = 0
28        i += 1
29
30    self.__strength = librosa.util.normalize(self.
31        __strength) # The beat strength normalization
32        between values 0-1.
```

Výpis A.5: Zdrojový kód funkce Dataset_selection v jazyce Python

```
1  def Dataset_selection(dataset_database : list[Dataset] ,
2                         genre_classification : GenreClassification ,
3                         beat_tracking : BeatTracking , mood : int):
4
5      # Get parameters
6      genre_predictions = genre_classification.
7          genres_predictions
8      tempo = beat_tracking.tempo
9
10     genres_difs = []
11
12     # Browsing thru all datasets
13     for i, dataset in enumerate(dataset_database):
14         genre_dif = 0
15         d_genres_prediction = dataset.genre
16         for key in d_genres_prediction:
17             genre_dif += d_genres_prediction[key] -
18                 genre_predictions[key]
19
20         genres_difs.append(np.abs(genre_dif))
21     genre_pass_datasets = []
22
23     # Get five datasets with smallest genre difference
24     for i in range(5):
25         index_of_min = int(np.argmin(genres_difs))
26         genre_pass_datasets.append( dataset_database[
27             index_of_min])
28         genres_difs[index_of_min] = 255
29
30     this_tempo_dif = 100
31     selected_dataset = Dataset
32
33     # Get dataset with same mood an smallest tempo
34     # difference
35     for dataset in genre_pass_datasets:
36         if dataset.mood == mood:
37             new_tempo_dif = abs(dataset.tempo - tempo)
38             if this_tempo_dif > new_tempo_dif:
39                 this_tempo_dif = new_tempo_dif
40                 selected_dataset = dataset
41
42     return selected_dataset
```

B Některé příkazy balíčku thesis

B.1 Příkazy pro sazbu veličin a jednotek

Tab. B.1: Přehled příkazů pro matematické prostředí

Příkaz	Příklad	Zdroj příkladu	Význam
<code>\textind{...}</code>	β_{\max}	$\$\\beta_{\\textind{max}}$$	textový index
<code>\const{...}</code>	U_{in}	$\$\\const{U}_{\\textind{in}}$$	konstantní veličina
<code>\var{...}</code>	u_{in}	$\$\\var{u}_{\\textind{in}}$$	proměnná veličina
<code>\complex{...}</code>	u_{in}	$\$\\complex{u}_{\\textind{in}}$$	komplexní veličina
<code>\vect{...}</code>	\mathbf{y}	$\$\\vect{y}$$	vektor
<code>\mat{...}</code>	\mathbf{Z}	$\$\\mat{Z}$$	matice
<code>\unit{...}</code>	kV	$\$\\unit{kV}$ či \unit{kV}$	jednotka

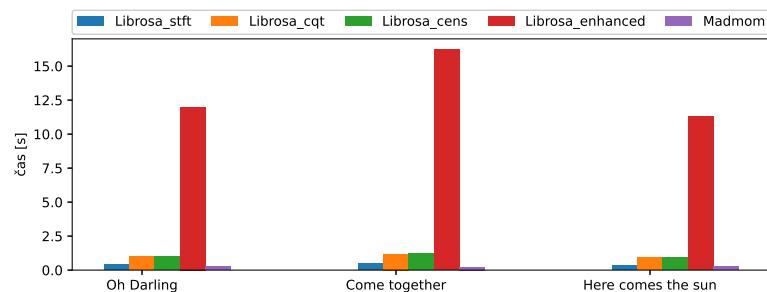
B.2 Příkazy pro sazbu symbolů

- `\E`, `\eul` – sazba Eulerova čísla: e,
- `\J`, `\jmag`, `\I`, `\imag` – sazba imaginární jednotky: j, i,
- `\dif` – sazba diferenciálu: d,
- `\sinc` – sazba funkce: sinc,
- `\mikro` – sazba symbolu mikro stojatým písmem¹: μ,
- `\uppi` – sazba symbolu π (stojaté řecké pí, na rozdíl od `\pi`, což sází π).

Všechny symboly jsou určeny pro matematický mód, vyjma `\mikro`, jenž je použitelný rovněž v textovém módu.

¹znak pochází z balíčku `textcomp`

C Druhá příloha



Obr. C.1: Zlepšené Wilsonovo proudové zrcadlo.

Pro sazbu vektorových obrázků přímo v L^AT_EXu je možné doporučit balíček TikZ. Příklady sazby je možné najít na T_EXample. Pro vyzkoušení je možné použít programy QTikz nebo TikzEdt.

D Příklad sazby zdrojových kódů

D.1 Balíček *listings*

Pro vysázení zdrojových souborů je možné použít balíček *listings*. Balíček zavádí nové prostředí *lstlisting* pro sazbu zdrojových kódů, jako například:

```
\section{Balíček lstlisting}
Pro vysázení zdrojových úsoubor je žmoné žpouit
balíček \href{https://www.ctan.org/pkg/listings}%
{\texttt{listings}}.

Balíček zavádí nové řprostředí \texttt{lstlisting} pro
sazbu zdrojových kóud.
```

Podporuje množství programovacích jazyků. Kód k vysázení může být načítán přímo ze zdrojových souborů. Umožňuje vkládat čísla řádků nebo vypisovat jen vybrané úseky kódu. Např.:

Zkratky jsou sázeny v prostředí *acronym*:

6 \begin{acronym}[KolikMista]

Šířka textu volitelného parametru *KolikMista* udává šířku prvního sloupce se zkratkami. Proto by měla být zadávána nejdelší zkratka nebo symbol. Příklad definice zkratky **symfvz!** je na výpisu D.1.

Výpis D.1: Ukázka sazby zkratky

```
1 | úžčřř
```

Ukončení seznamu je provedeno ukončením prostředí:

26 úž

```
27 {Discrete Fourier transform -- diskrétní Fourierova transformace}
28
29 \acro{STFT}
30 {Short-time Fourier transform -- krátkodobá Fourierova transformace}
31
32 \acro{RMS}
33 {Root mean square -- efektivní hodnota}
34
35 \acro{BPM}
36 {Beats per minute -- doby za minutu}
37
38 \acro{HTML}
```

```

39      {Hzpertext markup language -- hypertextový čznakovací jazyk}
40
41  \acro{CSS}
42  {Cascading Style Sheets -- kaskádové styly}
43
44  \acro{DMX}
45  {Digital Multiplexer -- protokol pro digitální řpenos informací}
46
47  \acro{LUFS}
48  {Loudness units relative to full scale}
49
50  \acro{HSV}
51  {Hue, Saturation, Value -- odstín, sytost, jas}
52
53  \acro{HTTP}
54  {Hypertext Transfer Protocol -- hypertextový řpenosový protokol}
55
56  \acro{JSON}
57  {JavaScript Object Notation}
58 \end{acronym}

```

Poznámka k výpisům s použitím volby jazyka czech nebo slovak:

Pokud Váš zdrojový kód obsahuje znak spojovníku –, pak překlad může skončit chybou. Ta je způsobená tím, že znak – je v českém nebo slovenském nastavení balíčku babel tzv. aktivním znakem. Přepněte znak – na neaktivní příkazem \shorthandoff{-} těsně před výpisem a hned za ním jej vratte na aktivní příkazem \shorthandon{-}. Podobně jako to je ukázáno ve zdrojovém kódu šablony.

Na výpisu D.2 naleznete příklad kódu pro Matlab, na výpisu D.3 zase pro jazyk C.

Výpis D.2: Příklad Schur-Cohnova testu stability v prostředí Matlab.

```
1 %% Priklad testovani stability filtru
2
3 % koeficienty polynomu ve jmenovateli
4 a = [ 5, 11.2, 5.44, -0.384, -2.3552, -1.2288];
5 disp( 'Polynom:' ); disp(poly2str( a, 'z' ))
6
7 disp('Kontrola pomocí korenu polynomu:');
8 zx = roots( a );
9 if( all( abs( zx ) < 1 ) )
10    disp('System je stabilni')
11 else
12    disp('System je nestabilni nebo na mezi stability');
13 end
14
15 disp(' '); disp('Kontrola pomocí Schur-Cohn:');
16 ma = zeros( length(a)-1,length(a));
17 ma(1,:) = a/a(1);
18 for( k = 1:length(a)-2 )
19    aa = ma(k,1:end-k+1);
20    bb = fliplr( aa );
21    ma(k+1,1:end-k+1) = (aa-aa(end)*bb)/(1-aa(end)^2);
22 end
23
24 if( all( abs( diag( ma.' ) ) ) )
25    disp('System je stabilni')
26 else
27    disp('System je nestabilni nebo na mezi stability');
28 end
```

Výpis D.3: Příklad implementace první kanonické formy v jazyce C.

```
// první kanonická forma
1
short fxdf2t( short coef[][][5], short sample)
2
{
3
    static int v1[SECTIONS] = {0,0},v2[SECTIONS] = {0,0};
4
    int x, y, accu;
5
    short k;
6
7
    x = sample;
8
    for( k = 0; k < SECTIONS; k++){
9
        accu = v1[k] >> 1;
10
        y = _sadd( accu, _smpy( coef[k][0], x));
11
        y = _ssh1(y, 1) >> 16;
12
13
        accu = v2[k] >> 1;
14
        accu = _sadd( accu, _smpy( coef[k][1], x));
15
        accu = _sadd( accu, _smpy( coef[k][2], y));
16
        v1[k] = _ssh1( accu, 1);
17
18
        accu = _smpy( coef[k][3], x);
19
        accu = _sadd( accu, _smpy( coef[k][4], y));
20
        v2[k] = _ssh1( accu, 1);
21
22
        x = y;
23
    }
24
    return( y);
25
}
```

E Obsah elektronické přílohy

Elektronická příloha je často nedílnou součástí semestrální nebo závěrečné práce. Vkládá se do informačního systému VUT v Brně ve vhodném formátu (ZIP, PDF ...).

Nezapomeňte uvést, co čtenář v této příloze najde. Je vhodné okomentovat obsah každého adresáře, specifikovat, který soubor obsahuje důležitá nastavení, který soubor je určen ke spuštění, uvést nastavení kompilátoru atd. Také je dobré napsat, v jaké verzi software byl kód testován (např. Matlab 2018b). Pokud bylo cílem práce vytvořit hardwarové zařízení, musí elektronická příloha obsahovat veškeré podklady pro výrobu (např. soubory s návrhem DPS v Eagle).

Pokud je souborů hodně a jsou organizovány ve více složkách, je možné pro výpis adresářové struktury použít balíček `dirtree`.

```
/ ..... kořenový adresář přiloženého archivu
  logo ..... loga školy a fakulty
    BUT_abbreviation_color_PANTONE_EN.pdf
    BUT_color_PANTONE_EN.pdf
    FEEC_abbreviation_color_PANTONE_EN.pdf
    FEKT_zkratka_barevne_PANTONE_CZ.pdf
    UTKO_color_PANTONE_CZ.pdf
    UTKO_color_PANTONE_EN.pdf
    VUT_barevne_PANTONE_CZ.pdf
    VUT_symbol_barevne_PANTONE_CZ.pdf
    VUT_zkratka_barevne_PANTONE_CZ.pdf
  obrazky ..... ostatní obrázky
    soucastky.png
    spoje.png
    ZlepseWilsonovoZrcadloNPN.png
    ZlepseWilsonovoZrcadloPNP.png
  pdf ..... pdf stránky generované informačním systémem
    student-desky.pdf
    student-titulka.pdf
    student-zadani.pdf
  text ..... zdrojové textové soubory
    literatura.tex
    prilohy.tex
    reseni.tex
    uvod.tex
    vysledky.tex
    zaver.tex
    zkratky.tex
  sablona-obhaj.tex ..... hlavní soubor pro sazbu prezentace k obhajobě
  sablona-prace.tex ..... hlavní soubor pro sazbu kvalifikační práce
  thesis.sty ..... balíček pro sazbu kvalifikačních prací
```