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

ZVUKOVÝ EFEKT PRO ZMĚNU VÝŠKY TÓNU

AUDIO EFFECT FOR PITCH MODIFICATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Bc. Václav Čermák

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jiří Schimmel, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

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

Student: Bc. Václav Čermák ***ID:*** 221462
Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Zvukový efekt pro změnu výšky tónu

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte princip zvukových efektů pro detekci a změny výšky zvukového signálu v reálném čase. V obou případech zvolte vhodný algoritmus, ověřte jej v prostředí Matlab pro off-line zpracování dat. Odladěné algoritmy následně implementujte jako zásuvný modul technologie VST pomocí prostředí JUCE či jiného.

DOPORUČENÁ LITERATURA:

[1] PIRKLE, William C. Designing audio effect plug-ins in C++ with digital audio signal processing theory. Abingdon, Oxon: Focal Press, 2013. ISBN 978-0-240-82515-1.
[2] BOULANGER, Richard Charles a Victor LAZZARINI. The audio programming book. Cambridge, Mass.: MIT Press, c2011. ISBN 978-0-262-01446-5.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: doc. Ing. Jiří Schimmel, Ph.D.

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

ABSTRAKT

Tato práce se zabývá realizací efektu pro detekci výšky zvukového signálu a její přelaďování v reálném čase. Nejprve analyzuje zvukové signály, jejich vlastnosti a základní operace pro časovou a frekvenční analýzu. Důraz je kladen na převod signálu z časové do frekvenční oblasti. Dále jsou rozebrány metody přelaďování zvukového signálu, detekce základního kmitočtu a určení znělosti. Praktická část se věnuje návrhu příslušných algoritmů, jejich implementaci v prostředí Matlab a hodnocení výsledků. Na to navazuje vytvořením zásuvného modulu technologie VST pro přelaďování zvukového signálu v reálném čase, který obsahuje vybrané algoritmy z předchozí části a je výstupem této práce.

KLÍČOVÁ SLOVA

detekce základní frekvence, Matlab, PSOLA, Fázový Vokodér, určení znělosti řečového signálu, váhovací okno, změna výšky tónu, zásuvný modul VST

ABSTRACT

This thesis deals with the realisation of an effect for real-time pitch detection and pitch shifting. At first, properties and basic operations of audio signals are analyzed for time and frequency analysis. An emphasis is placed on signal conversion from the time domain to the frequency domain. Methods for audio pitch shifting, fundamental frequency detection and voiced parts determination are discussed next. The practical part describes design of relevant algorithms, their implementation in Matlab environment and evaluation of the results. Evaluation is followed by the creation of a VST plug-in for real-time pitch shifting, which contains chosen algorithms from the previous section and is the final output of this thesis.

KEYWORDS

pitch detection, Matlab, PSOLA, Phase Vocoder, estimation of voiced speech signal, window function, pitch shifting, VST plugin

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

ČERMÁK, Václav. *Zvukový efekt pro změnu výšky tónu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 97 s. Bakalářská práce. Vedoucí práce: doc. Ing. Jiří Schimmel, Ph.D

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Václav Čermák
VUT ID autora:	221462
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Zvukový efekt pro změnu výšky tónu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Jiřímu Schimmelovi, Ph.D. a konzultantu Ing. Matouši Vrbíkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Transpozice zvukového signálu, tj. změna jeho výšky je důležitou funkcí při studiové práci se zvukovými signály. Díky němu jsme schopni například sjednotit ladění hudebních nástrojů v nahrávkách nebo opravit intonační chyby. Tato práce se zabývá principy změny výšky zvukového signálu v reálném čase a detekce výšky tónu, jejich porovnáním v prostředí Matlab a následnou realizací ve frameworku JUCE.

V teoretické části jsou rozebrány vlastnosti zvukových signálů, metody pro detekci a změny výšky tónu. Důraz je kladen na nejčastěji používané algoritmy, jejich omezení, silné stránky a slabiny. Některé poskytují spolehlivé a přesné výsledky za cenu vyšší výpočetní náročnosti, jiné naopak. Každý zvukový signál má jiné vlastnosti, které též vymezují vhodnost použití konkrétních algoritmů. Na základě nich jsou analyzovány i algoritmy nutné pro získání atributů klíčových pro některé metody přelaďování zvukového signálu, jako je například detekce jeho znělých částí.

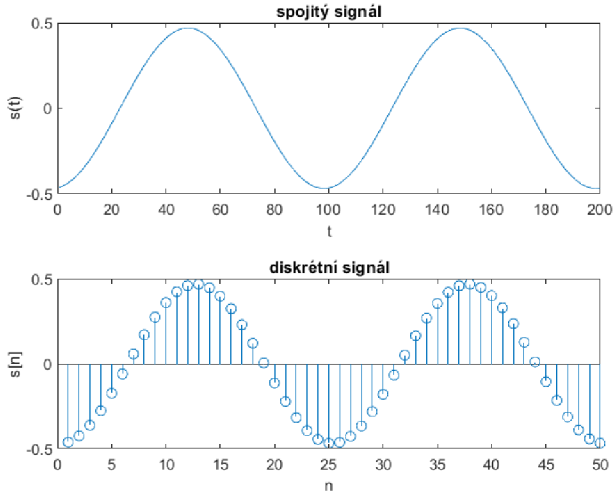
Praktická část se skládá ze dvou sekcí. První se zaměřuje na návrh, implementaci a porovnání všech zkoumaných algoritmů v prostředí Matlab. Druhá je věnována realizaci vybraných metod v jazyce C++ s využitím frameworku JUCE a vytvoření zásuvného modulu technologie VST, který je výstupem této práce.

1 Teoretický úvod

Tato kapitola se věnuje základním principům zvukových signálů. Nejprve je definován zvukový signál a jeho vlastnosti, podrobněji jsou zkoumány principy a algoritmy pro přelaďování zvukových signálů ve frekvenční a časové doméně. Dále jsou popsány metody detekce výšky tónu a metody určení znělosti segmentu, které mohou být součástí některých algoritmů přelaďování. Závěrem jsou shrnuty základy teorie stupnic a ladění.

1.1 Zvuk a zvukový signál

Signál je veličina, která slouží k přenosu a uchování informace. Hodnota signálu je v čase proměnná a podle spojitosti v čase dělíme signálové funkce na spojité $s(t)$ a diskrétní $s[n]$ [1, online].

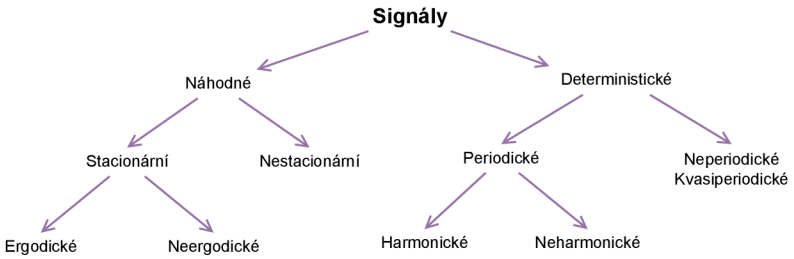


Obr. 1.1: Spojitý a diskrétní signál

Před digitálním zpracováním zvuku je nejprve analogový (spojitý) signál převeden pomocí procesu A/D převodu na digitální (diskrétní). Převod je realizován vzorkováním, následným kvantováním a kódováním (pro účely této práce bude vzorkování více rozebráno v části 1.1.3).

1.1.1 Dělení signálů

Podle různých aspektů lze signály dělit do kategorií. Základní dělení je podle náhodnosti jejich průběhu na deterministické a náhodné [1, online].



Obr. 1.2: Základní dělení signálů (podle [1])

Deterministické signály mají známou hodnotu závislé proměnné pro každou hodnotu nezávislé proměnné. Jejich průběh je definován známou funkcí nebo posloupností, např. $\cos x$. Dělí se dále na periodické a neperiodické.

Náhodné signály jsou takové, kterým nelze přesně stanovit jejich hodnotu, ale lze pouze odhadnout jejich vlastnosti s určitou pravděpodobností (pomocí distribuční funkce) [2, str. 16].

Periodické signály

Signál, pro který existuje takové $T > 0$, že platí $x[t] = x[t+T]$, nazýváme periodický s periodou T. Nejmenší T, které splňuje tuto podmínku, se pak nazývá základní periodou a značí se T_0 [2, str. 16].

Většina hudebních signálů je kvaziperiodická a jejich periodičnost se tedy v čase mění. Hudební signál má proto smysl analyzovat z hlediska jeho krátkodobých segmentů (více o segmentaci v sekci 1.3). Periodickým segmentům lze určit jejich základní kmitočet, který je stěžejní pro algoritmy přeladění výšky tónu pracující v časové doméně.

Harmonické signály

Jedná se o podskupinu periodických signálů definovaných pomocí funkcí sinus nebo kosinus.

$$u(t) = U_m \cdot \cos\left(\frac{2\pi}{T_0}t + \phi_1\right) \tag{1.1}$$

1.1.2 Spektrum

Každý periodický signál lze rozložit na dílčí harmonické složky s kmitočtem rovným celočíselnému násobku základního kmitočtu. Pro k-tou harmonickou složku periodického signálu a základní kmitočet platí:

$$f_k = k \cdot f_0, \text{ kde } f_0 = \frac{1}{T_0} \tag{1.2}$$

Zobrazení všech harmonických složek signálu v kmitočtové oblasti nazýváme jeho frekvenčním spektrem [2, str. 38]. Základní kmitočet definuje výšku tónu a označujeme jej jako první harmonickou složku. Zvýšením základního kmitočtu tedy roste i výška tónu.

1.1.3 Vzorkování signálu

Vzorkování je první částí analogově-číslicového převodu, díky kterému ze signálu se spojitým časem získáme signál s diskrétním časem [2, str. 90]. Počet vzorků získaných ze vstupu za jednu vteřinu nazýváme vzorkovacím kmitočtem a značíme jej f_{vz}. Z něj lze odvodit vzorkovací periodu, která vyjadřuje časovou vzdálenost mezi dvěma po sobě jdoucími vzorky.

$$T_{\text{vz}} = \frac{1}{f_{\text{vz}}} \tag{1.3}$$

Vzorkování signálu je patrné na obrázku 1.1 (z horního spojitého signálu je získán spodní diskrétní).

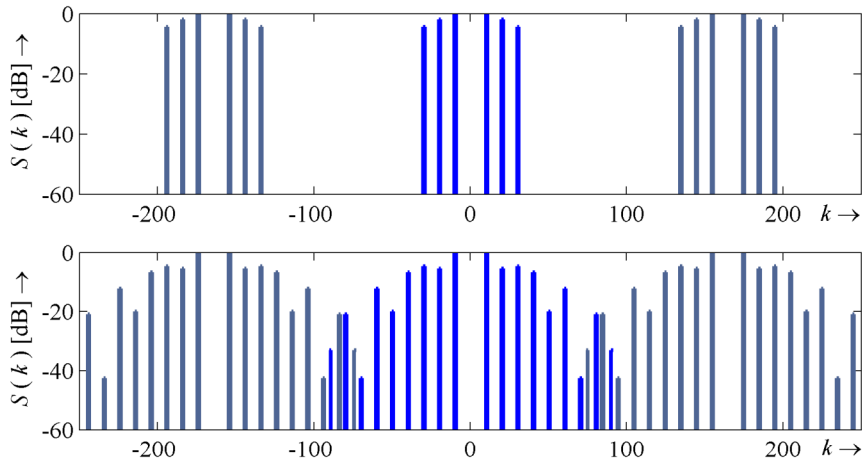
Nyquistův teorém

Při vzorkování signálu vzorkovacím kmitočtem f_{vz} dochází na kmitočtu $\frac{f_{\text{vz}}}{2}$ k zrcadlení spektra. Na kmitočtech rovných celočíselným násobkům vzorkovacího kmitočtu se celé spektrum periodicky opakuje. Z toho plyne omezující vztah vzorkovacího kmitočtu a maximálního kmitočtu, který lze vzorkovacím kmitočtem věrohodně reprezentovat. Toto omezení popisuje vzorkovací neboli Nyquistův teorém:

$$f_s > 2 \cdot f_{\max} \tag{1.4}$$

Vzorkovací kmitočet musí být dvakrát vyšší, než maximální kmitočet, který má být v signálu obsažen. Při nedodržení vzorkovacího teorému dochází kolem f_{vz} k prolnutí spekter neboli aliasingu. To je zobrazeno na obrázku 1.3. Dolní graf znázorňuje překrytí spekter kolem f_{vz}.

Kmitočtový rozsah lidského sluchu se pro většinu populace pohybuje v rozmezí 20 Hz − 20 kHz. Pro záznam zvukových signálů je proto potřeba volit kmitočet vyšší, než $2 \cdot 20 = 40$ kHz. Pro hudební signály byly standardizovány vzorkovací kmitočty 44,1 kHz, 48 kHz a jejich celočíselné násobky. První kmitočet má původ v použití pro páskové videorekordéry kvůli synchronizaci s obrazovými snímky. Druhý vznikl jako jednoduchý poměr ke vzorkovacímu kmitočtu 32 kHz pro FM stereo [3, str.8].



Obr. 1.3: Oboustranné modulové kmitočtové spektrum, dochází k aliasingu [3, str. 9].

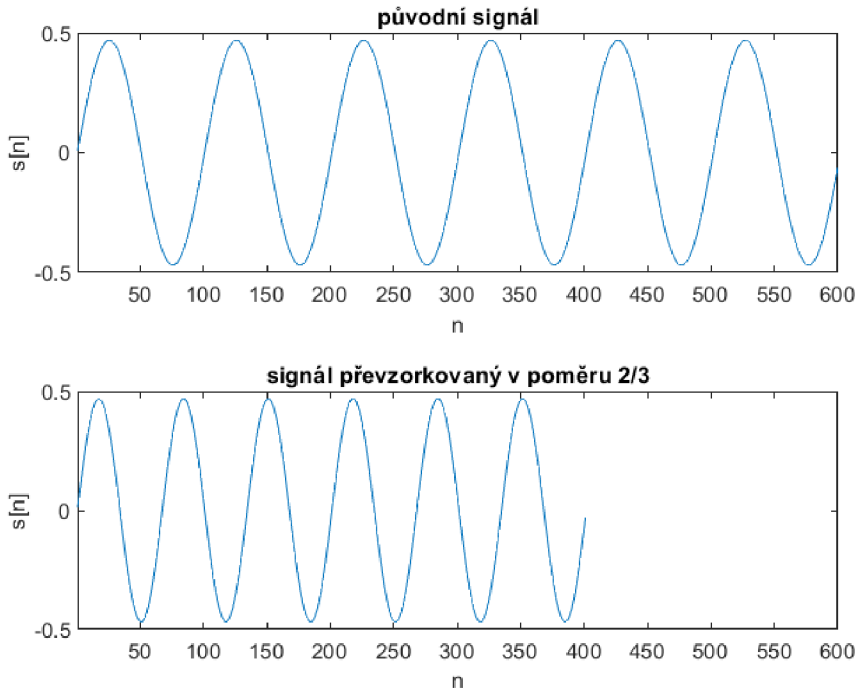
Převzorkování signálu

Převzorkování signálu je proces při kterém dochází ke změně jeho vzorkovacího kmitočtu. Převzorkování se skládá z operací nadvzorkování a podvzorkování prováděných v celočíselném poměru. Nadvzorkováním dochází ke zvýšení vzorkovací frekvence f_{vz} a tedy i počtu vzorků. Mezi staré vzorky se nejprve vloží sekvence nulových vzorků, jejichž hodnota se následně dopočte interpolací původního signálu. Podvzorkováním naopak f_{vz} klesá a některé vzorky se vypouští [4, online].

Pokud má být signál převzorkován z $f_{\mathrm{vz1}} = 10$ kHz na $f_{\mathrm{vz2}} = 15$ kHz je potřeba tak učinit v poměru $k = \frac{f_{\mathrm{vz2}}}{f_{\mathrm{vz1}}} = \frac{3}{2}$. Signál se nadvzorkuje třikrát, tedy mezi každé dva vzorky jsou vloženy tři nulové, které jsou poté dopočteny interpolací. Signál je následně dvakrát podvzorkován, a tedy každý druhý vzorek je zahozen.

Převzorkováním dochází ke změně výšky tónu, pokud je nový signál přehrán s původním vzorkovacím kmitočtem. Současně se však mění i jeho délka a spektrální obálka. Převzorkováním v poměru $k > 1$ se zvýší základní kmitočet f_0, prodlouží se délka signálu a formantové oblasti se posunou k nižším kmitočtům. S poměrem $k < 1$ naopak f_0 roste , délka signálu se zkracuje a formantové oblasti se posouvají směrem nahoru.

Samotné převzorkování nelze použít pro změnu výšky tónu v reálném čase, neboť mění délku zpracovaného signálu. Je ale součástí mnoha algoritmů pro změnu výšky tónu (více v 1.4).



Obr. 1.4: Převzorkování signálu v poměru N_1/N_2

1.2 Fourierova transformace

K vyjádření spektra signálu využíváme Fourierovu transformaci, která je definována následujícím předpisem:

$$S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-\mathrm{j}t\omega} \mathrm{d}t, S(\omega) \in \mathbb{C} \tag{1.5}$$

K převedení spektra zpět do časové oblasti signálu slouží zpětná Fourierova transformace, definovaná vztahem:

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{\mathrm{j}t\omega} \mathrm{d}\omega, s(t) \in \mathbb{C} \tag{1.6}$$

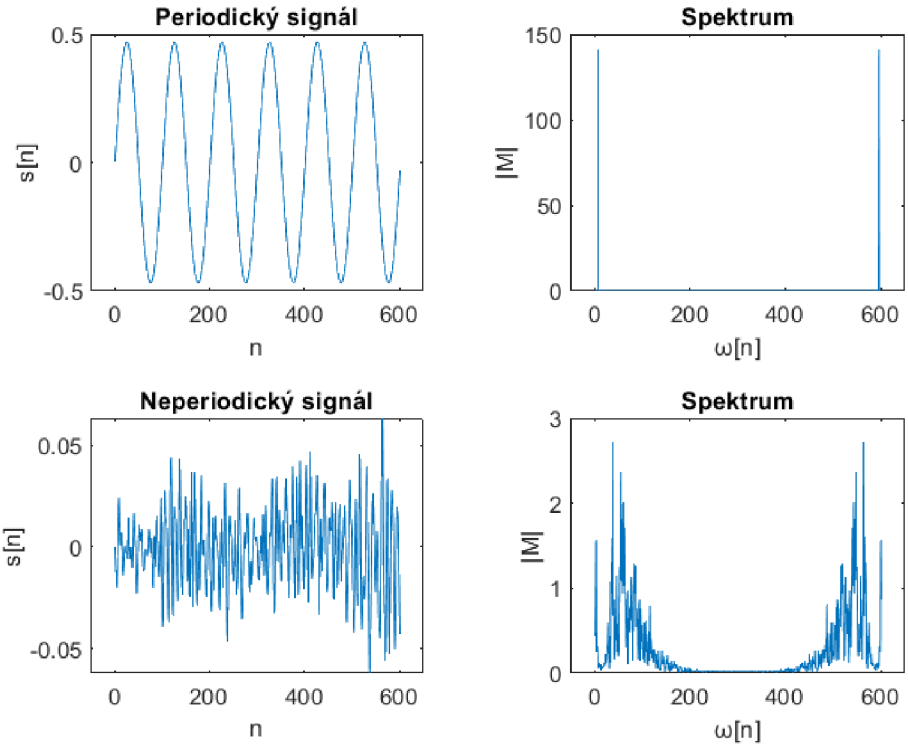
Tyto obecné vztahy platí pro signály se spojitým časem, ze kterých vychází i úpravy Fourierovy transformace pro signály s diskrétním časem. Dalším omezením je platnost vzorců pouze na signály s nekonečným počtem vzorků. V praxi digitálního zpracování signálů se ve většině případů pracuje s konečnými signály [2, str. 54].

1.2.1 Fourierova transformace časově diskrétních signálů

DTFT (*discrete-time Fourier transform*) je metodou analýzy časově diskrétních signálů a dává možnost pohlížet na ně jako na součet dílčích harmonických složek. Používá se k výpočtu spektra a je definována vztahem:

$$S(e^{\mathrm{j}\omega}) = \sum_{-\infty}^{\infty} s[n] e^{-\mathrm{j}\omega n} \tag{1.7}$$

Aplikací DTFT na diskrétní periodický signál vzniká spojité spektrum. Periodický diskrétní signál má též spojité spektrum, avšak jednotlivé harmonické složky se v něm projeví v podobě impulzů, viz obrázek 1.5 (lze si všimnout symetričnosti spektra podle poloviny vzorkovacího kmitočtu). Spojitost spektra je pro účely digitálního zpracování signálu nepraktická. Z aplikace vzorkování v kmitočtové oblasti s pevným krokem $\Delta\omega = \frac{2\pi}{N}$ lze odvodit diskrétní Fourierovu transformaci, jejímž výstupem je diskrétní reprezentace spektra [5, str. 120].



Obr. 1.5: Diskrétní Fourierova transformace periodického a neperiodického signálu

1.2.2 Diskrétní Fourierova transformace

DFT (discrete Fourier transform) je variantou Fourierovy transformace využívanou v praxi pro diskrétní signály konečné délky a definuje ji předpis

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-\mathrm{j}2\pi k \frac{n}{N}} = \sum_{n=0}^{N-1} s[n] \left[\cos\left(\frac{2\pi}{N} kn\right) - \mathrm{j} \sin\left(\frac{2\pi}{N} kn\right) \right], \tag{1.8}$$

kde N je délka signálu a k určuje pořadí vzorku spektra. Výstupem DFT je soubor komplexních čísel, ve kterých je uložena informace o amplitudě a fázi jednotlivých spektrálních složek. Tento soubor nazýváme komplexním spektrem. Frekvenci dané složky v souboru určuje její pořadí a rozlišení spektra, které je dáno vztahem:

$$f_\mathrm{r} = \frac{f_\mathrm{vz}}{N} \tag{1.9}$$

Například pro pátou složku spektra jsou získaného pomocí DFT jsou informace o spektru uloženy v souboru na pozici 5 a odpovídající frekvence složky je $f = 5 \cdot f_r$.

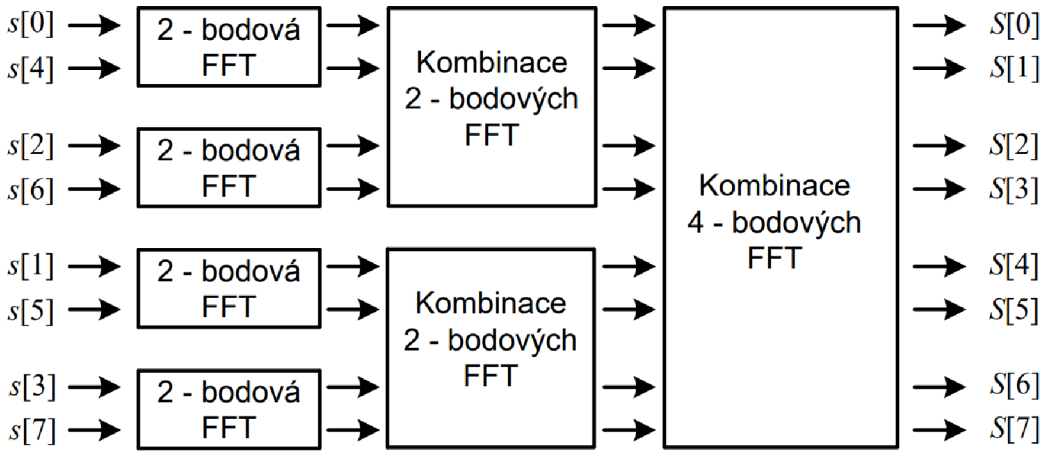
Je-li délka signálu N, musí DFT pro každý vzorek spektra $S[k]$ provést N^2 operací komplexního násobení a $N \cdot (N-1)$ komplexních součtů, což je dohromady $2 \cdot N^2 - N$ operací [5, str. 137]. S délkou signálu tedy významně roste počet operací a výpočet koeficientů spektra může být neefektivní. Úpravou algoritmu lze však časovou složitost podstatně snížit.

1.2.3 Rychlá Fourierova Transformace

Pod označením FFT jsou sdruženy algoritmy vylepšující DFT z hlediska výpočetní náročnosti. Algoritmy jsou založeny na rozložení výpočtu N-bodového DFT do několika menších M-bodových DFT a následné kombinaci jejich dílčích výsledků (viz obr. 1.6). Rozkládání signálu postupně na sudé a liché poloviny se říká decimace v čase. Největší výpočetní úspory dosahuje algoritmus pro bloky dlouhé $N = 2^m, m \in N$ vzorků [5, str. 137].
DFT i FFT mají své inverzní verze, které provadí stejné operace jako inverzní Fourierova transformace, ovšem s diskrétními signály.

Krátkodobá Fourierova transformace

Pokud je potřeba sledovat změny spektra v čase, je možné využít algoritmus krátkodobé Fourierovy transformace neboli STFT (angl. *short-time Fourier transform*). Zakládá se na rozkladu signálu do překrývajících se segmentů (viz 1.3), na které je aplikován algoritmus FFT. Toho lze využít například při detekci základního kmitočtu (více v kapitole 1.5).



Obr. 1.6: FFT – Blokový diagram [5, str. 139]

Rozmazávání spektra

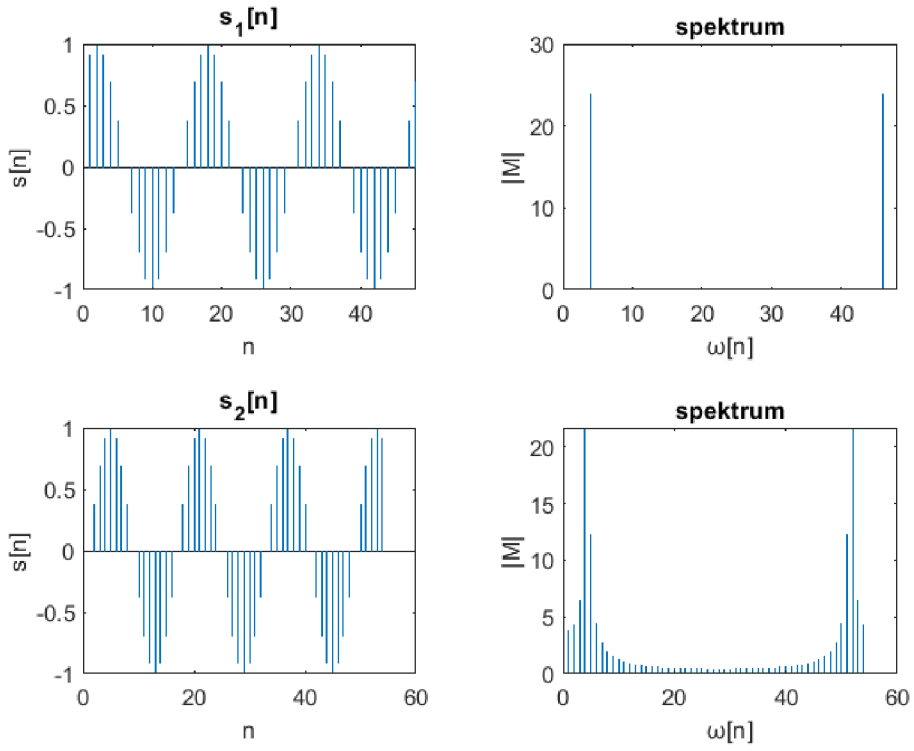
Rozmazávání resp. prosakování spektra (angl. *spectral leakage*) je jev, při kterém dochází k vykreslení více spektrálních složek, než těch, které jsou obsaženy v signálu. Dochází k němu, při zvolení špatného rozlišení FFT v poměru k základnímu tónu signálu. Nechť je signál $s[n]$ periodický (viz sekce 1.1.1), N je jeho délkou a T_0 základní periodou udávanou jako počet vzorků. Pokud platí vztah

$$N = k \cdot T_0, \tag{1.10}$$

pak nedochází k prosakování spektra [5, str. 143]. Situaci, která nastává v opačném případě ilustruje obrázek 1.7. Signály s_1 a s_2 mají stejnou základní periodu $T_{1,1} = T_{1,2} = 16$. Liší se však jejich délky $N_1 = 48$ a $N_2 = 54$ vzorků.
U obecného signálu je základní perioda často neznámá a rozlišení FFT je v praxi pevně dané. Proto je potřeba vliv rozmazávání spektra buď snížit (například použitím okénkových funkcí viz. 1.3) nebo s ním počítat a extrahovat přesnou hodnotu frekvence jinak (například z předchozí fáze viz 1.4.2).

1.3 Segmentace signálu

Signál je při zpracování v reálném čase podrobován metodám krátkodobé analýzy. Kvůli určení krátkodobých časových a spektrálních vlastností je nutné jej rozdělit do menších částí. Tento proces se nazývá segmentace a využívá se při určování základního tónu i při přeladění výšky zvukového signálu. Důvodem použití může být požadavek na průběžné zobrazení zpracovaného výstupu nebo jiné informace



Obr. 1.7: Prosakování spektra. Signál s_1 splňuje podmínku 1.10, s_2 ne.

získané ze signálu. Při segmentaci lze použít i různou míru překrytí sousedních segmentů (angl. *overlap*), kterým lze docílit například zvýšení rozlišení FFT.

1.3.1 Okénková funkce

Okénková funkce nabývá hodnoty 0 mimo definovaný, většinou symetrický interval a uvnitř tohoto intervalu nabývá maximálně hodnoty 1. Při zpracování diskrétního signálu pomocí STFT dochází k prosakování spektra. Okénková funkce tento jev do určité potlačuje. Obecně lze popsat předpisem:

$$w(n) = \begin{cases} f(n) & n \in \langle 1; N \rangle \\ 0 & \text{pro ostatní } n \end{cases}$$

Podle tvaru funkce $f(n)$ rozlišujeme několik základních typů okénkových funkcí.

Pravoúhlé okno je nejjednodušší implementací. Nezabraňuje však rozmazávání spektra.

$$f(n) = 1 \tag{1.11}$$

Barlettovo okno jiným názvem trojúhelníkové okno.

$$f(n) = \begin{cases} \frac{2n}{N} & n \in \left\langle 0; \frac{N}{2} \right\rangle \\ 2 - \frac{2n}{N} & n \in \left\langle \frac{N}{2}; N \right\rangle \end{cases}$$

Hammingovo okno se používá při zpracování řeči.

$$f(n) = 0{,}54 - 0{,}46 \cos 2\pi \frac{n}{N} \tag{1.12}$$

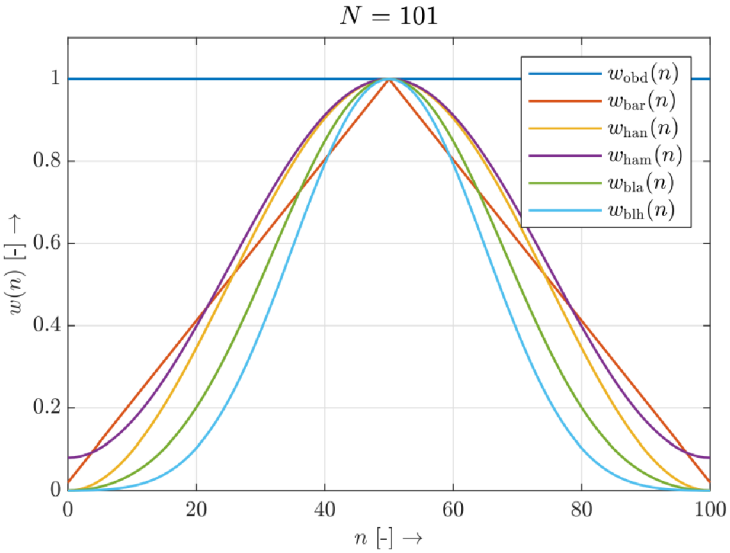
Hannovo okno se též používá při zpracování řeči.

$$f(n) = 0{,}5 \cdot \left(1 - \cos 2\pi \frac{n}{N}\right) \tag{1.13}$$

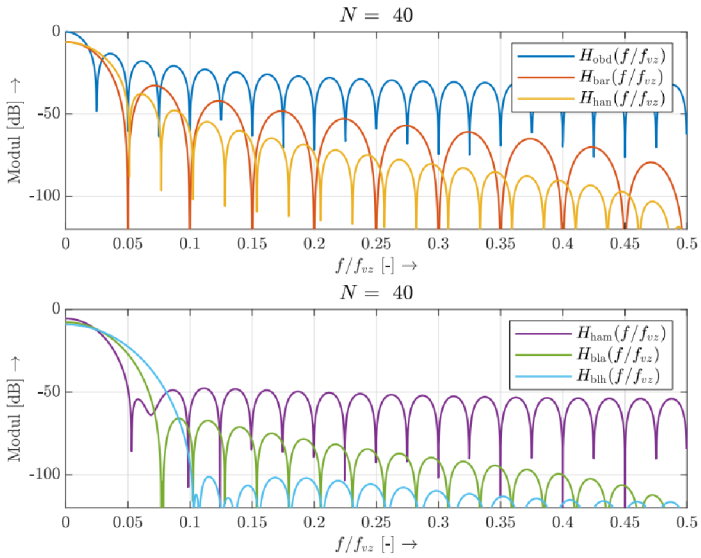
Blackmanovo okno se používá při návrhu FIR filtrů.

$$f(n) = 0{,}42 - 0{,}5 \cos\left(2\pi \frac{n}{N-1}\right) + 0{,}08 \cos 4\pi \frac{n}{N-1} \tag{1.14}$$

Na obrázku 1.8 jsou zobrazeny tvary známých okénkových funkcí a následně jejich frekvenční charakteristiky na obrázku 1.9.



Obr. 1.8: Srovnání časových průběhů oken dlouhých 101 vzorků (převzato z [6, str. 28])



Obr. 1.9: Srovnání kmitočtových charakteristik oken o délce 40 vzorků (převzato z [6, str. 28])

1.3.2 Overlap-Add

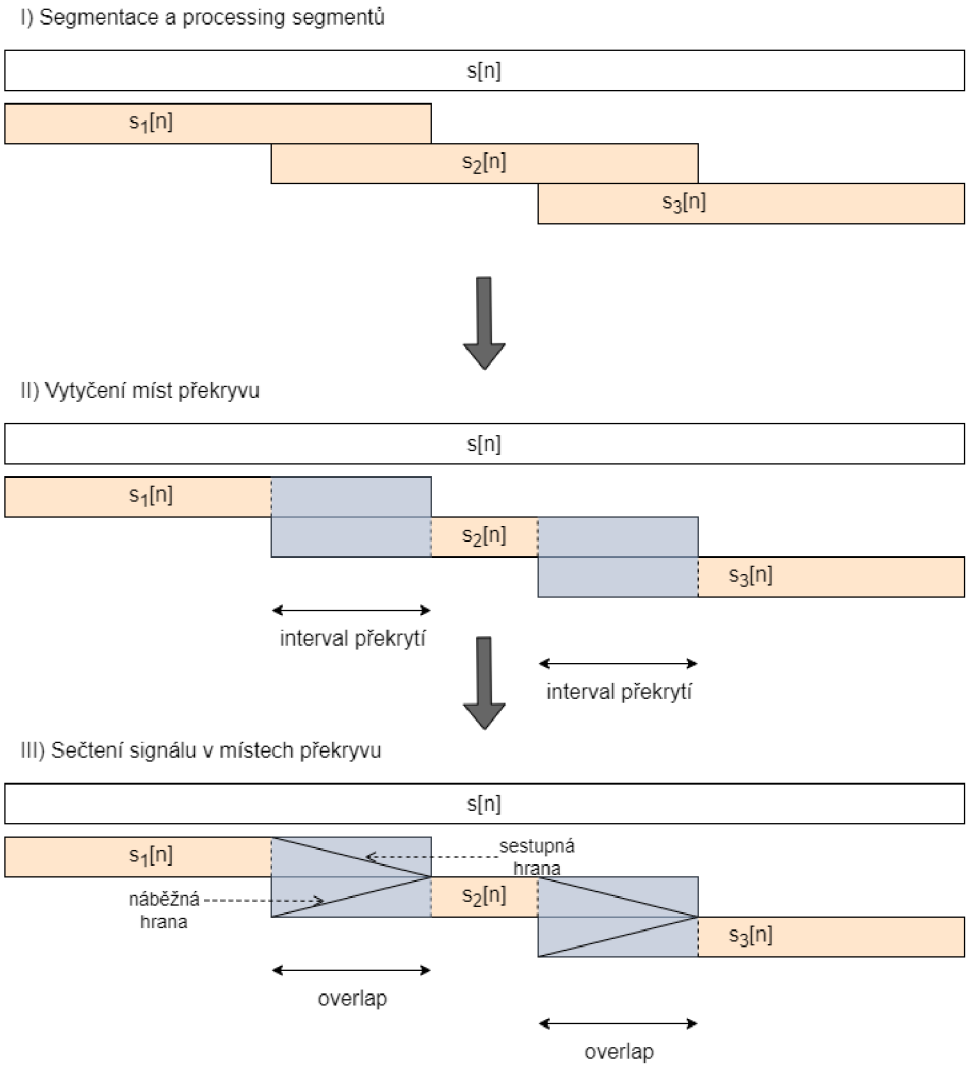
Overlap-Add neboli OLA je metoda používaná při segmentaci a opětovné rekonstrukci signálu. Skládá se ze tří kroků:

1. Rozdělení signálu na překrývající se segmenty
2. Zpracování každého segmentu
3. Sečtení zpracovaných komponent ve výstupní signál

Na obrázku 1.10 je znázorněn princip metody. V místě překrytí segmentů jsou vytvořeny náběžné a sestupné hrany například váhováním vhodným oknem, po němž jsou zpracované segmenty sečteny ve výstupní signál [7, str. 311].

Kritérium konstantnosti OLA

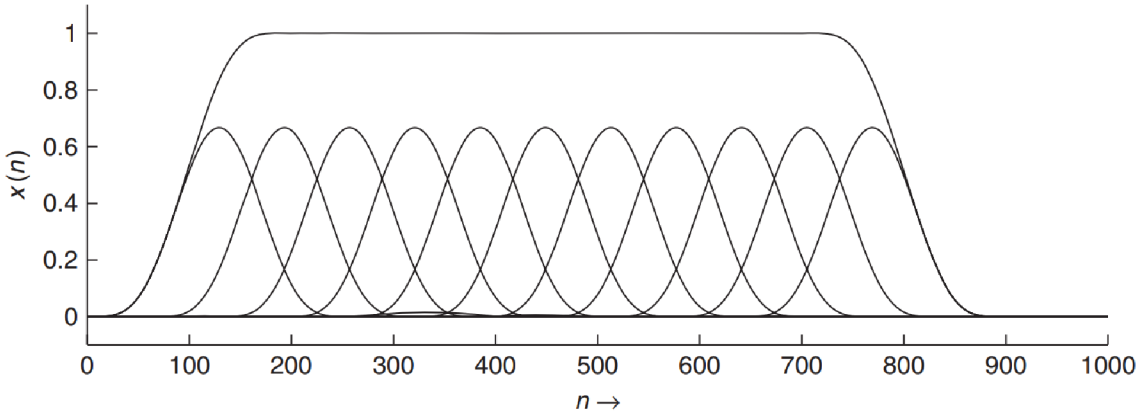
Jedná se o podmínku zachování amplitud ve spektru při sčítání překrývajících se segmentů. Neboli pro zachování amplitudy ve spektru musí být součet překrývajících se oken konstantní [8, str. 19]. Situace je znázorněna na obrázku 1.11.



Obr. 1.10: Schéma metody *Overlap-Add* (podle [9, str. 51])

1.4 Principy přeladění zvukového signálu

Cílem algoritmů pro přeladění výšky zvukového signálu, neboli transpozice, je změnit jeho základní kmitočet. Toho lze docílit přímo manipulací se vzorky signálu a o příslušných algoritmech přeladění pak říkáme, že pracují v časové oblasti. Druhou možností je získat krátkodobé spektrum pomocí Fourierovy transformace, provést požadované změny v něm a zpětnou Fourierovou transformací získat signál s přeladěnou výškou tónu. O takových algoritmech říkáme, že pracují ve frekvenční oblasti [10, str. 199]. Přímočarým řešením pro přeladění výšky by mohlo být i prosté převzorkování signálu. S ním je však spojena i změna jeho délky, což je pro zpracování v reálném čase nežádoucí.



Obr. 1.11: Kritérium zachování konstantnosti (převzato z [10, str. 233])

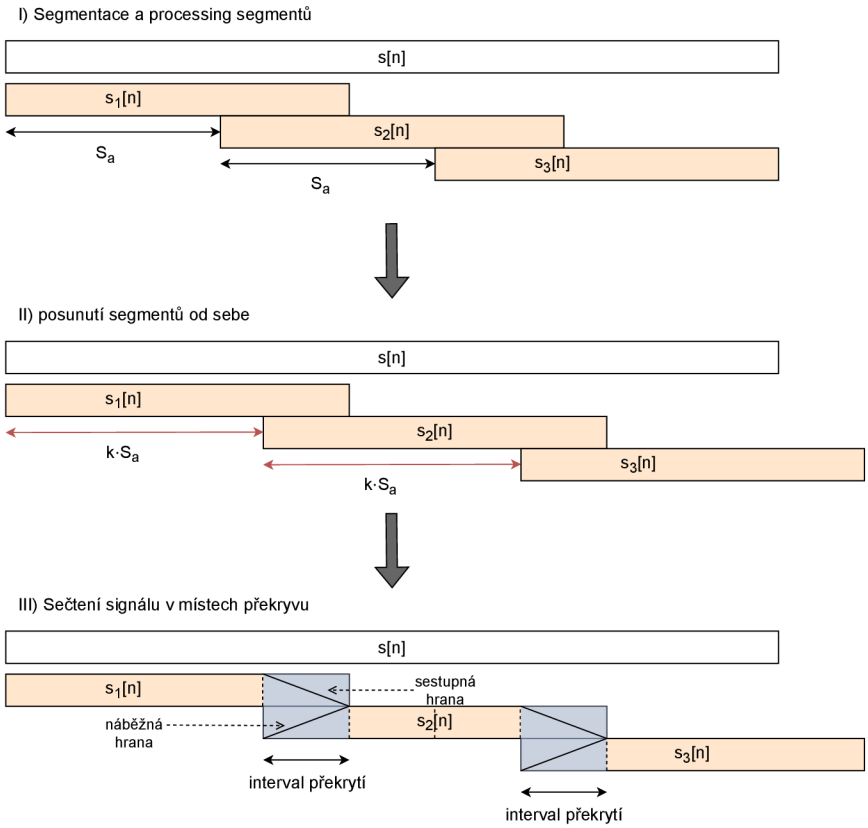
1.4.1 SOLA

Celým názvem *Synchronous Overlap and Add* je algoritmus, primárně určený pro změnu délky zvukového signálu při zachování jeho výšky. Pro svou funkci využívá metodu *overlap-add*, při které jsou segmenty od sebe posunuty blíže resp. dále, čímž je dosaženo prodloužení resp. zkrácení signálu. Při zpětném sčítání segmentů může docházet ke vzniku nežádoucích artefaktů jako například k fázovému zkreslení (viz obr. 1.12), protože sousední segmenty na sebe po posunutí nemusí plynule navazovat.

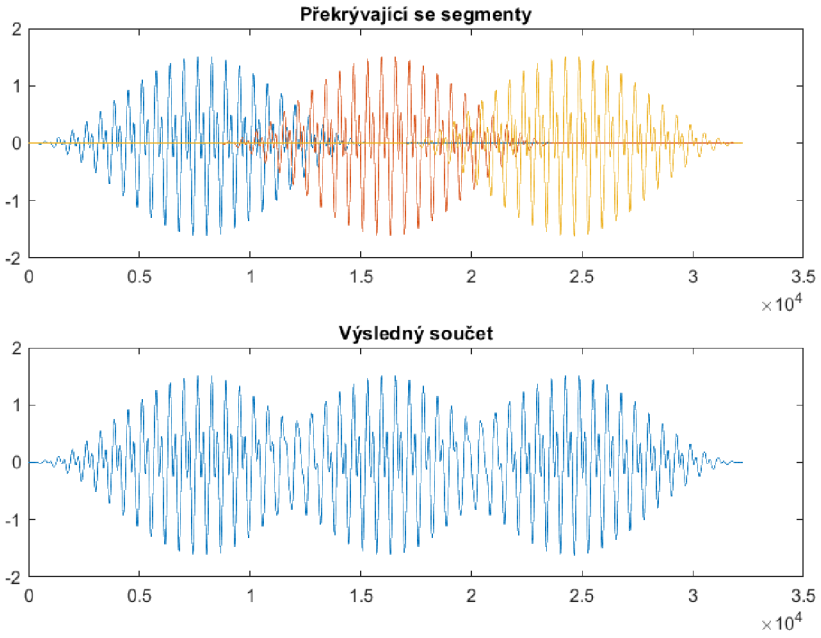
V kombinaci s následným převzorkováním výstupního signálu lze algoritmus SOLA využít i k přeladění výšky tónu, pokud je výstupní signál po prodloužení v poměru N_1/N_2 převzorkován poměrem N_2/N_1. Změní se základní frekvence i formantové oblasti.

1.4.2 PSOLA

Algoritmus PSOLA, celým názvem *Pitch Synchronous Overlap and Add* vychází z algoritmu SOLA. Má několik aplikací a lze využít i pro změnu výšky tónu při zachování délky signálu. Oproti SOLA však provádí segmentaci v souladu se základní periodou jednotlivých segmentů. Jeho výhodou je i to, že dokáže zachovat původní formantové oblasti [10, str. 194].



Obr. 1.12: Schéma změny délky signálu pomocí algoritmu SOLA



Obr. 1.13: Fázové zkreslení způsobené nenavazujícími segmenty

Existuje několik variant algoritmu podle toho, ve které oblasti se signálem pracuje. Mezi nejběžnější z nich patří:

TD-PSOLA neboli Time Domain PSOLA pracující v časové oblasti signálu pracuje přímo se vzorky signálu.

FD-PSOLA neboli Frequency Domain PSOLA, která k analýze a zpracování signálu využívá různých modelů jako je krátkodobé spektrum.

LP-PSOLA neboli Linear Predictive PSOLA, která je založena na lineární predikci.

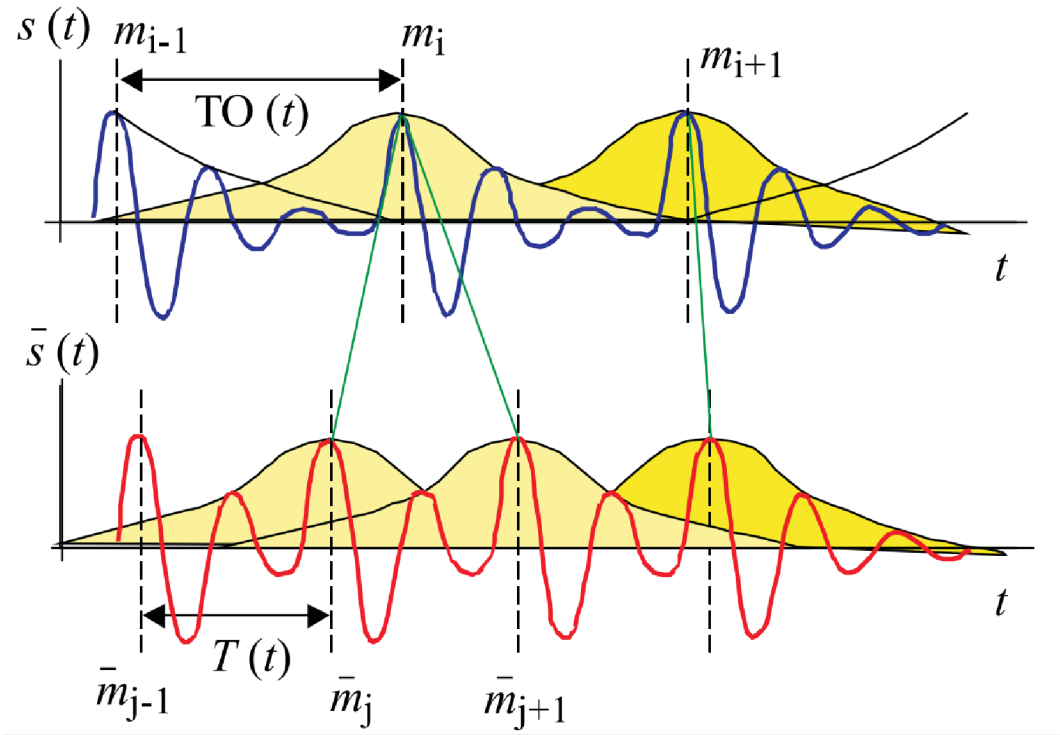
Nejpoužívanější variantou je TD-PSOLA díky své nízké výpočetní náročnosti. Ta je při přelaďování v reálném čase klíčová, neboť lze v reálném prostředí předpokládat nasazení algoritmu ve VST pluginu na několika sběrnicích v DAW zároveň, při současném použití dalších pluginů, které budou vytěžovat procesor. Zbývající dvě metody jsou výpočetně náročnější, ale disponují větší variabilitou kontroly nad spektrální obálkou [11, str. 58]. Tato práce se dále zabývá variantou algoritmu pracující v časové oblasti.

TD-PSOLA

Time-Domain PSOLA je variantou algoritmu pracující se vzorky signálu přímo v časové oblasti a skládá ze 4 kroků [11, str. 58]:

1. Identifikace špiček základní periody
2. Segmentace signálu $s[n]$ na krátkodobé překrývající se úseky $s_{\mathrm{m}}[n]$, dlouhé dvě základní periody T_0 se středem ve špičkách základní periody (viz obr. 1.14).
3. Váhování každého segmentu vhodným oknem $w[n]$, které je umístěno v jeho středu, tedy místě jedné ze špiček základní periody. Nejčastěji se k váhování používá Hannovo okno a překrytí segmentů 50 %, tedy délka jedné základní periody.
4. Časová komprese či expanze a sečtení po sobě jdoucích segmentů v místech překrytí pomocí metody *overlap-add*. Při zkracování resp. prodlužování signálů jsou na vhodných místech segmenty duplikovány resp. vynechány, aby došlo k zachování délky.

Modifikací algoritmu lze docílit i posunutí formantových oblastí při zachování výšky tónu. Algoritmus PSOLA potřebuje k rozkladu signálu a časové modifikaci znát základní periodu (kmitočet), aby dokázal efektivně detekovat její špičky. K tomu lze použít některý z algoritmů detekce základní periody, kterým se věnuje kapitola 1.5.



Obr. 1.14: Ukázka posunutí jednotlivých *pitch* segmentů (převzato z [11, str. 58])

1.4.3 Fázový vokodér

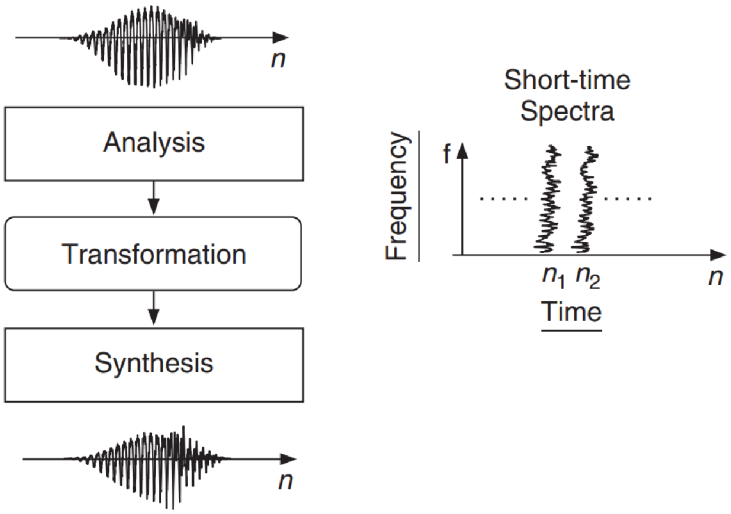
Fázový vokodér (angl. *Phase Vocoder*) patří do kategorie algoritmů pracujících ve frekvenční doméně. Existuje několik různých implementací, ale skládají se vždy ze tří fází:

1. Spektrální analýza
2. Transformace
3. Resyntéza

Fázový vokodér může být využit pro časovou kompresi resp. expanzi zvukového signálu, v časově proměnných filtrech, pro změnu výšky zvukového signálu nebo například odšumění signálu [10, str. 219]. Existuje několik variant algoritmu [12, str. 147].

1. Analýza vstupního signálu pomocí banky filtrů, kterou jsou získány moduly a fáze. Ty jsou modifikovány transformačním algoritmem a syntetizovány zpět pomocí součtu digitálních oscilátorů s nastavitelnou amplitudou a fází. Čím vyšší je počet pásem banky filtrů, tím kvalitnější je zpětná syntéza zvukového signálu.
2. Analýza signálu pomocí FFT, modifikace spektra transformačním algoritmem (přeladění) a zpětná syntéza skrze IFFT a *overlap-add*. Algoritmus využívá segmentace signálu s překryvem a váhovacím oknem (viz 1.16).

3. Analýza pomocí FFT, interpolace modulů a fází s následnou aplikací transformačního algoritmu a zpětná syntéza pomocí součtu harmonických signálů.
4. Analýza a syntéza pomocí Gaborovy transformace, založené na krátkodobé FFT s Gaussovým oknem.

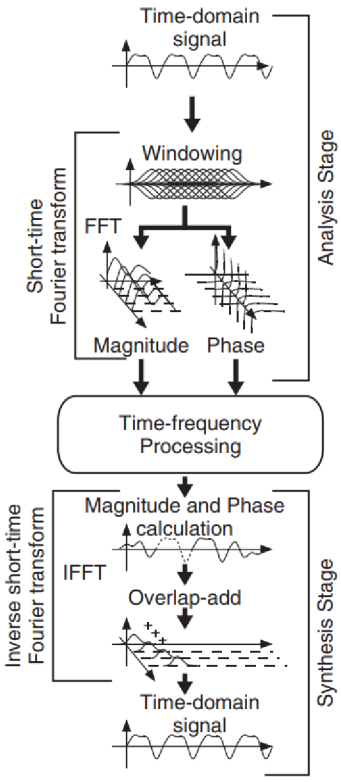


Obr. 1.15: Fázový vokodér a jeho fáze zpracování signálu (převzato z [10, str. 220])

Fázový vokodér má využití v mnoha reálných aplikacích a lze pomocí něj dosáhnout rozdílných efektů v závislosti na použitém transformačního algoritmu. Jádro vycházející ze zvolené varianty je stejné a efekty mohou být i kombinovány. Nejběžnějšími variantami fázového vokodéru jsou 2 a 3 díky rychlosti zpracování a snadné adaptibilitě [10, str. 243]. Tato práce se dále zabývá variantou 2 využívající dopřednou a zpětnou Fourierovu transformaci.

Rozbalení fáze

Rozbalení fáze (anglicky *phase unwrapping*) je klíčovou metodou k získání přesnější informace o vyšších harmonických složkách signálu. Pevné rozlišení FFT omezuje to, které frekvence mohou být v komplexním spektru zaznamenány přesně. Při rozlišení $N = 1024$ vzorků a vzorkovacím kmitočtu $f_{vz} = 44100$ Hz lze popsat přesně pouze celočíselné násobky rozlišení spektra $f_r = \frac{f_{vz}}{N} = 43{,}07$ Hz. Ty budou ve frekvenčním spektru zobrazeny jako špičky (viz obr. 1.7, signál s_1). Pokud nějaká frekvenční složka vstupního signálu neodpovídá přesnému celočíselnému násobku, například $f_k = 80$ Hz, zobrazí se jako špička v nejbližším bloku FFT (kterému odpovídá hodnota 86,14 Hz). Dojde však k rozmazání spektra a její energie se rozprostře i do sousedních bloků (viz obr. 1.7, signál s_2). Rozbalení fáze je proces, při kterém se



Obr. 1.16: Fázový vokodér varianta s FFT/IFFT (převzato z [10, str. 220])

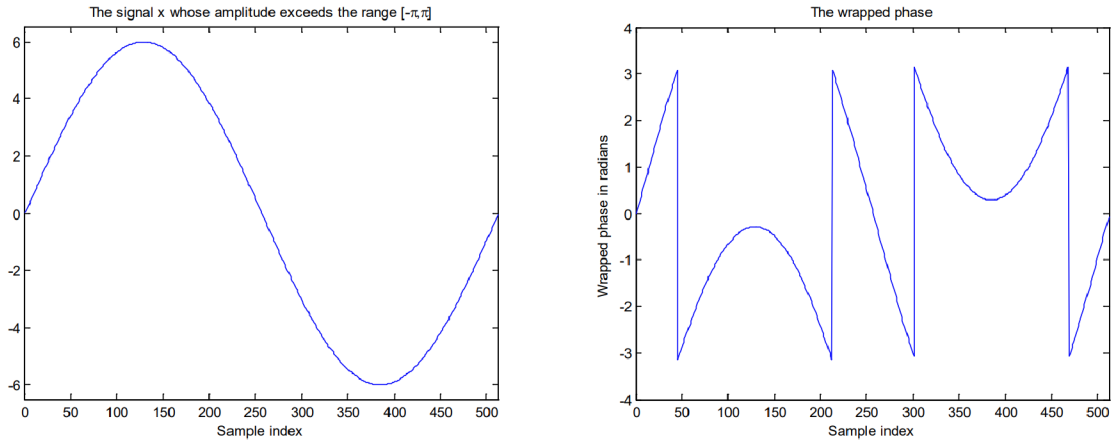
pomocí současné a předchozí hodnoty fáze vypočítané z bloku komplexního spektra aproximuje její přesnější okamžitá hodnota [10, str. 241]. Zabalení fáze představuje pravý opak. Obecná hodnota fáze, která může nabývat libovolného reálného čísla je mapována tak, aby vždy nabývala hodnot z intervalu $\langle -\pi; \pi \rangle$.

Problematika rozbalení fáze vychází z definice funkce atan2, která je rozšířením arkus tangens na výpočet ve všech čtyřech kvadrantech a používá se k výpočtu fáze z komplexního spektra. Obor hodnot čtyřkvadrantové funkce atan2 je $\langle -\pi; \pi \rangle$. Fáze však může nabývat i hodnot, které leží mimo tento interval, a proto může při výpočtu docházet k tomu, že hodnoty spadající mimo něj budou ořezány a ve fázovém spektru vzniknou skoky [13, online]. V některých případech je naopak žádoucí provést zabalení fáze. Jedním z nich je výpočet fázové diference, která je definována předpisem

$$\Delta\phi(H,k) = \phi_m(H,k) - \phi_{m-1}(H,k) - \omega_k H, \tag{1.15}$$

kde m je pořadí segmentu ve vstupním signálu, H vychází z překryvu segmentů a udává velikost kroku ve vzorcích a k je pořadí bloku harmonické složky v komplexním spektru FFT. Fázová diference udává vzdálenost, o kterou se liší skutečná

fáze od teoretické fáze příslušného bloku komplexního spektra. První část rovnice 1.4.3 představuje rozdíl současné a předchozí fáze této složky. Nese informaci o fázovém posunu v bloku k spektra, který následoval po skoku o H vzorků ve vstupním signálu. Poslední člen $\omega_k H$ představuje očekávaný fázový posun, který by spektrální složka vykonala po H vzorcích, kdyby odpovídala celočíselnému násobku rozlišení FFT. Fázová diference jinými slovy popisuje rozdíl skutečného a očekávaného fázového posunu [10, str. 243]. S jeho znalostí lze určit přesnou hodnotu frekvenčních složek spektra.



Obr. 1.17: Obrázek a) ukazuje rozbalenou fázi, na obrázku b) je signál se zabalenou fází (převzato z [13, online])

Transformační algoritmus pro změnu výšky

Transformační algoritmy slouží k modifikaci signálu ve frekvenční oblasti pro dosažení požadovaného efektu. Jednou z těchto transformací může být i změna výšky tónu, které lze dosáhnout několika způsoby. Jedním z nich je, obdobně jako u algoritmu PSOLA, časová komprese resp. expanze v poměru N_2/N_1 a převzorkování v opačném poměru N_1/N_2, čímž je zachována délka výstupního signálu. I u fázového vokodéru je proto možné vycházet z transformačního algoritmu pro změnu délky (angl. *time stretching*), převzorkovat každý segment ihned po syntéze a aplikovat na sousední segmenty metodu OLA [10, str. 259]. Jinou možností je přelaďovat zvukový signál přímo posunutím frekvenčních složek v krátkodobém spektru Fourierovy transformace [14, online]. Jádro metody se skládá z následujících kroků:

1. zisk krátkodobého modulového a fázového spektra Fourierovou transformací
2. výpočet přesné frekvence složek z bloků komplexního spektra pomocí fáze předchozího segmentu
3. roztažení všech frekvenčních složek daným poměrem
4. dosazení nově vzniklých frekvencí do nejbližších bloků spektra (obrácený postup bodu č. 2)
5. syntéza z nově vzniklých amplitud a fází spektrálních složek
6. převedení syntetizovaného signálu zpět do časové oblasti zpětnou Fourierovou transformací

Pro každý blok FFT je potřeba provést rozbalení fáze. Pokud by přeladění probíhalo přímo na blocích FFT, vznikly by ve výsledném signálu nežádoucí neharmonické poměry mezi jeho vyššími harmonickými složkami [14, online]. Jedním ze způsobů vycházejících z rozbalení fáze, je využití fázové diference. Z ní lze pro každý blok FFT určit skutečný kmitočet frekvenčních složek a škálovat jej, čímž bude dosaženo přeladění výšky tónu. Vychází se přitom z definice úhlového kmitočtu jakožto časové derivace fáze

$$\omega = \frac{\mathrm{d}\phi}{\mathrm{d}t} \approx \frac{\Delta\phi}{H} f_{\mathrm{vz}}, \tag{1.16}$$

$$f = \frac{1}{2\pi}\frac{\mathrm{d}\phi}{\mathrm{d}t} \approx \frac{1}{2\pi}\frac{\Delta\phi}{H} f_{\mathrm{vz}}, \tag{1.17}$$

kde H je délka skoku ve vzorcích a f_{vz} je vzorkovací kmitočet. Po každém výpočtu fázové diference je vhodné následovat zabalením fáze (viz 1.4.3), aby bylo zamezeno fázovým skokům složek mezi jednotlivými segmenty [14, online].

1.4.4 AutoTune

Algoritmy přelaďování lze snadno adaptovat pro realizaci různých efektů, jako je například *auto-tune* [10, str. 322]. Jeho princip spočívá v detekci intonace a její následné korekci. Ta může být prováděna buď chromaticky k nejbližšímu půltónu nebo k nejbližšímu tónu zvolené stupnice. Transpozice se mění v čase podle detekovaného základního kmitočtu. K realizaci tohoto efektu je potřeba spolehlivý a rychlý algoritmus detekce základního kmitočtu. Pokud se detekovaný tón mírně odchyluje od ladění definované stupnice, lze pak dopočítat nejbližší tón ze stupnice, ve které by se měla intonace zvukového signálu pohybovat. Se znalostí změřené a ideální výšky tónu lze určit koeficient přeladění, pomocí něhož lze provést korekci intonace některým z algoritmů pro přelaďování.

1.5 Detekce základního kmitočtu

Základní frekvence hudebního signálu určuje jeho výšku. Znalost výšky je klíčová k určení intonace a rozlišení, zda se daný tón nachází v tónině. Využívá se též v algoritmu PSOLA pro detekci špiček ve zvukovém segmentu. Detekce zvukového signálu může obdobně jako přelaďování výšky tónu pracovat v časové a frekvenční oblasti. Jelikož se intonace může s časem měnit, je stejně jako u metod přelaďování zvukového signálu vhodné rozdělit vstupní signál do překrývajících se segmentů a detekovat výšku tónu pro jednotlivé z nich. Pokud jsou v této práci zmíněny algoritmy pro detekci základní periody je jimi myšlen tentýž soubor algoritmů, neboť základní perioda vychází z reciprokého vztahu $T = \frac{1}{f}$.

1.5.1 Autokorelační metoda

Autokorelační metoda pracuje v časové oblasti a k určení základní periody používá autokorelační funkci. Při existenci mnoha variant a vylepšení této metody se jedná o nejpoužívanější a zároveň nejjednodušší metodu pro odhad základního tónu. Autokorelační funkce určuje míru podobnosti v rámci signálu a je definována předpisem [15, str. 3]:

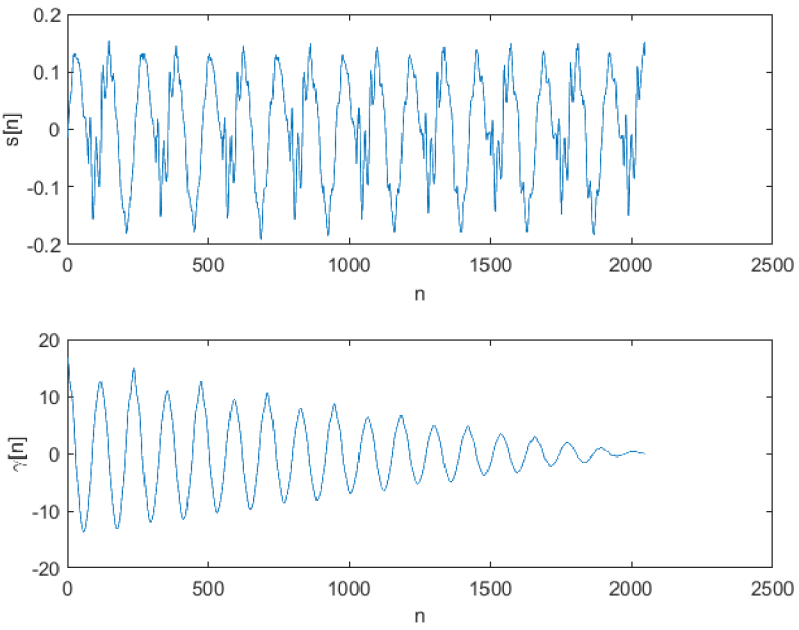
$$\gamma[n] = \sum_{\mathrm{n=0}}^{N-m-1} x[n] \cdot x[n+m] \tag{1.18}$$

Tvar autokorelační funkce závisí na periodičnosti zvukového segmentu. Na obrázku 1.18 je zobrazena obálka znělého segmentu zpěvu hlásky a jeho jednostranná autokorelační funkce. Na jejím tvaru lze vypozorovat výskyt lokálních maxim v místech vzdálených od sebe o délku T_0. Globální maximum autokorelační funkce pro analyzovaný segment se bude vždy vyskytovat v počátku segmentu $s[0]$. V něm dojde v první iteraci k překrytí signálu se sebou samým (nulové posunutí). V místě druhého nejvyššího maxima $s[k]$ vzdáleného k vzorků od počátku se pak nachází *peak* základní periody T_0 pro daný segment [15, str. 3]. Z jeho pozice lze určit základní kmitočet a tedy výšku tónu pomocí vztahu:

$$f_0 = \frac{f_{\mathrm{vz}}}{\mathrm{k}} \tag{1.19}$$

Obrázek 1.18 oproti tomu zobrazuje časový průběh neznělého segmentu hlásky „s“. Z rozdílného tvaru funkcí plyne, že autokorelace poskytuje relevantní odhad základní periody pouze pro periodické neboli znělé segmenty. Pro poskytnutí relevantních výsledků autokorelační metody je nejprve potřeba určit periodičnost zpracovávaného segmentu a v případě, že se jedná o neznělý jej přeskočit. Určení znělosti segmentu podrobněji zkoumá sekce 1.5.4. Dalším omezením autokorelace je nevhodnost jejího použití pro signály s potlačenou první harmonickou složkou. Příkladem může být

zvukový záznam violoncella, které má výraznější druhou harmonickou složku. Tu autokorelace chybně prohlásí za základní kmitočet (nastává *octave-error*).



Obr. 1.18: Autokorelační funkce znělého segmentu

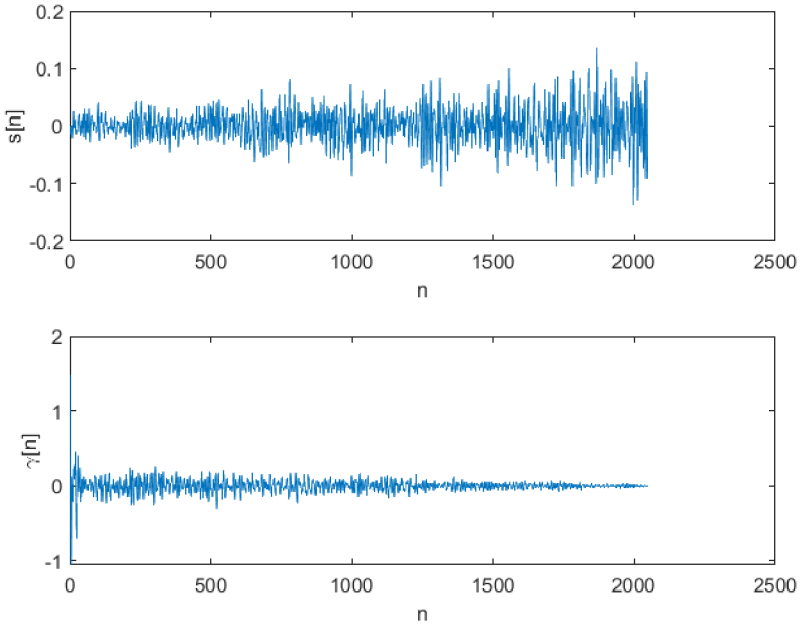
Výhodou autokorelační metody je její implementační jednoduchost a přijatelná výpočetní náročnost. Tu lze zefektivnit přidáním mezí, ve kterých se má odhad základní periody pohybovat. Pro příklad omezením odhadu na interval frekvence od 80 Hz do 800 Hz, které odpovídají při vzorkovacím kmitočtu 44100 Hz periodám o délkách 552 a 56 vzorků. U segmentu delšího než $552 - 56 = 496$ vzorků není potřeba počítat autokorelační funkci v celé délce segmentu, ale ušetřit výpočetní výkon.

Rychlá autokorelační metoda

Autokorelační funkci lze získat i jiným způsobem, než přímým výpočtem rovnice 1.5.1. Rychlejší variantou je její získání pomocí FFT, které vychází z rozšíření předchozího vztahu [16, str. 32]:

$$\gamma[n] = \sum_{\mathrm{n=0}}^{N-m-1} x[n] \cdot x[n+m] = \mathrm{IFFT}\{\mathrm{FFT}\{x\} \cdot \mathrm{FFT}\{x\}^*\}, \tag{1.20}$$

kde * značí komplexně sdružené číslo. Při dodržení podmínek délky segmentu 2^N pro FFT je výpočet autokorelační funkce podstatně rychlejší, než výpočet pomocí definice 1.5.1. Určení základní frekvence je pak obdobné jako v autokorelační metodě.



Obr. 1.19: Autokorelační funkce neznělého segmentu

1.5.2 Metoda centrálního klipování

Jedná se o vylepšení autokorelační funkce vycházející z předpokladu, že k určení základního kmitočtu vstupního signálu stačí znát pouze polohu jeho špiček. K jejich přesnějšímu rozlišení se využívá prahování signálu nastaveným prahem P [15, str. 4]. Princip algoritmu lze rozdělit do několika kroků:

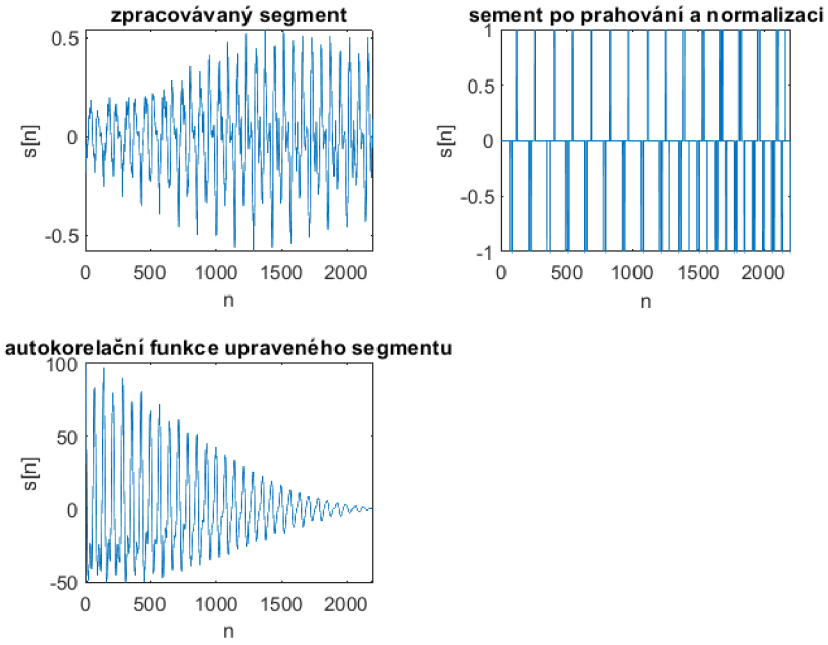
1. Rozdělení signálu na segmenty
2. Určení prahové hodnoty P pro každý segment. Amplituda signálu se v čase výrazně mění, a proto nelze určit jednotný práh, pro celý signál. Práh je proto stanoven pomocí maxim amplitud sousedních segmentů pomocí vzorce:

$$T_i = k \cdot \min(\max(\text{segment}_{i-1}), \max(\text{segment}_{i+1})) \tag{1.21}$$

   Hodnota parametru $k = 0{,}8$ je určena empiricky [15, str. 4].
3. Prahovaný signál je dále normalizován tak, aby nabýval pouze 3 hodnot. Vzorky, jejichž hodnota leží v intervalu $\langle -P; P \rangle$ jsou prahovány na hodnotu 0. Vzorkům větším, než P je přiřazena hodnota 1 a analogicky vzorkům menším, než $-P$ hodnota -1.
4. Na normalizovaném signálu je vypočtena autokorelační funkce, z níž je určena základní perioda T_0.

Nevýhodou centrálního klipování je nutnost znát hodnotu maximální špičky nadcházejícího segmentu, což komplikuje použití metody pro zpracování v reálném čase.

Jedním z řešení může být zpoždění vstupního signálu o délku jednoho segmentu. Metoda se díky prahování stává citlivější na šum, který ale pro potřeby této práce není tak podstatným aspektem.



Obr. 1.20: Metoda centrálního klipování aplikovaná na zpěv

1.5.3 Spektrální metoda

Jednoduchou a přímočarou metodou detekce základního kmitočtu je jeho určení přímo z diskrétního spektra. Postup metody je následující:

1. Výpočet spektrogramu pomocí FFT
2. Frekvenční omezení oblasti zpracování
3. Výběr maximální hodnoty modulu

Tato metoda má však několik omezení. Mezi nejvýznamnější z nich patří rozlišení FFT, které vychází z délky analyzovaného segmentu. Při délce 2048 vzorků a vzorkovací frekvenci $f_{\mathrm{vz}} = 44100$ Hz je rozlišení spektra pouhých $f_{\mathrm{r}} = \frac{44100}{2048} = 21{,}53$ Hz. Lze předpokládat, že většina aplikací algoritmu přeladování bude omezena na zdroje zvuku se základním kmitočtem pohybujícím se v rozmezí 50 Hz – 1000 Hz, ve kterém je takové rozlišení nedostatečné. Existuje však několik metod, jak její přesnost zvýšit. Kvůli výběru maximální špičky ve spektru mohou též nastat *octave-errory*.

1.5.4 Kepstrální metoda

Základní tón se určuje pomocí reálného kepstra $c[n]$, které je definováno jako reálná část inverzní Fourierovy transformace přirozeného logaritmu modulu FFT vstupního signálu [15, str. 7].

$$c_R[n] = \Re\{\text{IFFT}(\ln|\text{FFT}(s[n])|)\} \tag{1.22}$$

Postup metody:

1. segmentace signálu
2. pro každý segment se vypočítá modulové spektrum pomocí FFT
3. na každou jeho složku se aplikuje přirozený logaritmus
4. získání kepstra $c[n]$ z IFFT
5. reálná část kepstra $c_R[n]$ se vynásobí pravoúhlým oknem $w[n]$, které je definováno vztahem

$$w[n] = 0 \text{ pro } n \notin \left\langle \frac{f_{vz}}{50}; \frac{f_{vz}}{1000} \right\rangle \tag{1.23}$$

Okno se používá k oddělení vlastností hlasového traktu od informací o excitaci [15, str. 8]. Čísla 50 a 1000 udávají interval, ve kterém bude hledán základní kmitočet f_0. V kepstru se po vynásobení oknem hledá špička, z jejíž polohy se vyjádří základní kmitočet podle vztahu 1.5.1. U řečových signálů popisuje spodní část kepstra vlastnosti hlasového traktu a horní část jeho excitační vlastnosti [5, str. 169].

1.6 Určení znělosti segmentu

Metody detekce základního kmitočtu předpokládají pro správnou detekci s periodickým signálem na vstupu. Většina hudebních signálů je kvaziperiodická, tedy jejich periodičnost se v čase mění. Například záznam zpěvu může obsahovat jak periodické (znělé), tak neperiodické (neznělé) úseky. Za znělé úseky lze považovat všechny samohlásky. Pro souhlásky jako například „s“ nebo „z“ však základní frekvenci nelze přesně určit a jejich spektrum vykazuje spíše šumový charakter [17, str. 118]. Výsledky poskytnuté metodami pro detekci základního kmitočtu proto nebudou validní pro neznělé segmenty. Chybně určený základní kmitočet se následně projeví i při přeladování neznělého segmentu, především u algoritmu PSOLA, pro který je spolehlivé určení základního kmitočtu klíčové. Algoritmy přeladování výšky, především časové domény, by proto měly nejprve určit znělost segmentů a následně přeladit pouze ty, které jsou znělé.

Krátkodobá energie

Parametr, který lze kromě znělosti použít i pro detekci řečové aktivity. Energii segmentu lze určit ze vztahu

$$E = \frac{1}{N} \sum_{n=0}^{N-1} s[n]^2, \tag{1.24}$$

kde N je délka segmentu a $s[n]$ jeho n-tý vzorek. Znělé hlásky disponují vysokou krátkodobou energií, neznělé naopak nízkou [17, str. 118].

Zero-crossing

Počet průchodů nulovou úrovní určuje počet změn polarity signálu a je definován vztahem

$$Z = \frac{1}{N} \sum_{n=0}^{N-1} |\text{sign}(s[n]) - \text{sign}(s[n-1])|^2, \tag{1.25}$$

kde funkce signum je definována jako

$$\text{sign}(n) = \begin{cases} -1 & n < 0 \\ 0 & n = 0 \\ 1 & n > 0. \end{cases}$$

Parametr je citlivý na přídavný šum [17, str. 119].

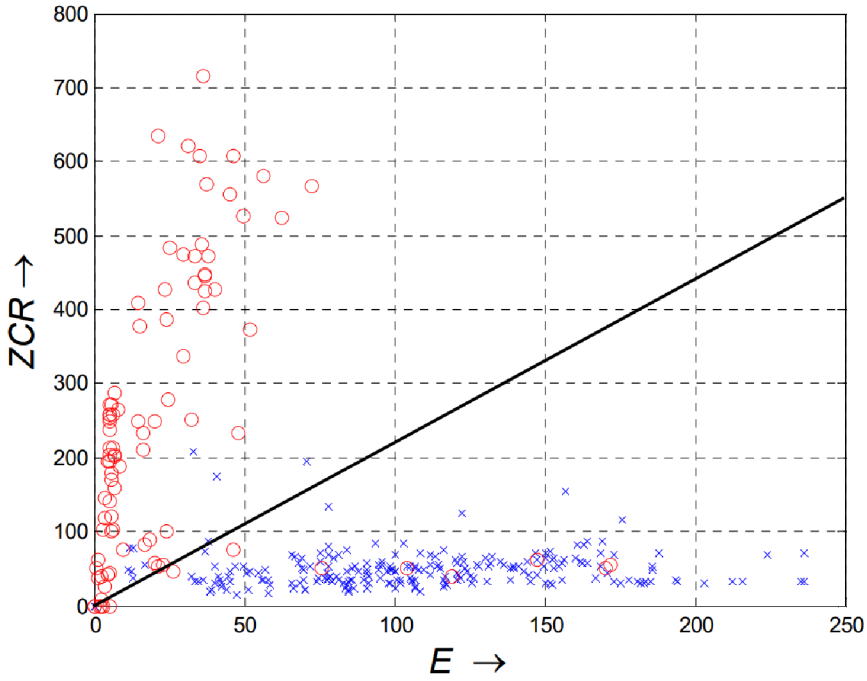
Klasifikace znělosti metodou neuronu

S využitím krátkodobé energie a počtu průchodů nulou lze stanovit znělost segmentu metodou jednoho neuronu. Signál znělého úseku má v porovnání s neznělým úsekem větší krátkodobou energii a menší počet průchodů nulou, viz obrázek 1.21 [15, str. 9]. Příznaky parametrů znělosti se téměř nepřekrývají a lze je proto oddělit přímkou. Problematická je pouze hodnota krátkodobé energie, která závisí i na časové proměnlivosti signálu. Hlasitější části signálu budou disponovat vyšší energií, tišší pasáže naopak slabší.

$$y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + \Theta) \tag{1.26}$$

$$y = f(2{,}068 \cdot E(\text{i}) - 1{,}065 \cdot Z(\text{i}) + 0{,}013) \tag{1.27}$$

Rozhodovací funkce f vrací hodnoty 0 pro neznělý a 1 pro znělý segment. Nejprve je spočtena hodnota $Y = 2{,}068 \cdot E(\text{i}) - 1{,}065 \cdot Z(\text{i})$. Tato hodnota se porovná s empiricky nastaveným prahem $P = 0{,}013$. Za znělý je označen segment, pro který Y má vyšší hodnotu, než práh P [15, str. 9].



Obr. 1.21: Příznaky parametrů znělosti [15, str. 9]

Klasifikace znělosti poměrem energií v pásmech

Metoda se využívá pro klasifikaci znělosti řeči pomocí rozložení krátkodobé energie ve spektru. U znělých hlásek se většina energie koncentruje v pásmu přibližně do 1 kHz oproti neznělým hláskám, které mají energii rozloženou ve spektru rovnoměrně. Pomocí filtrů se signál rozdělí na čtyři kmitočtová subpásma podle tabulky 1.6:

Tab. 1.1: Rozsahy kmitočtových pásem pro $f_{\mathrm{vz}} = 8000$ Hz

	kmitočtové pásmo [Hz]
$x_1[n]$	0–500
$x_2[n]$	500–1000
$x_3[n]$	1000–2000
$x_4[n]$	2000–4000

Pro jednotlivá subpásma rozdělená do 4 segmetů je vypočítána krátkodobá energie. Poměr energií v pásmech je stanoven podle vztahu

$$R = \frac{E_1 + E_2}{E_3 + E_4}, \tag{1.28}$$

kde E_i je energie signálu $x[i]$ v subpásmu pro $i = 1, 2, 3, 4$. Dále je spočten empirický práh podle vzorce

$$T = \mu - 0.9 \cdot \sigma, \tag{1.29}$$

kde σ je směrodatná odchylka poměrů energií pro jednotlivé segmenty a μ jejich střední hodnota [15, str. 11]. Tato metoda používá k vyhodnocení znělosti celé délky signálu, zejména při určení střední hodnoty a směrodatné odchylky. Její použití je v tomto provedení pro zpracování signálu v reálném čase nevhodné.

Spectral Flatness Measure

Metoda určení spektrální plochosti (SFM) vychází z předpokladu, že neznělé segmenty mají výkon ve spektru rovnoměrně rozložen. Nejprve vypočte výkonové spektrum (PSD) jako druhou mocninu amplitudy všech spektrálních složek. Spektrální plochost je určena vztahem

$$\mathrm{SFM} = 10 \log \frac{G_\mathrm{m}}{A_\mathrm{m}}, \tag{1.30}$$

kde G_m je geometrický průměr výkonového spektra a A_m je aritmetický průměr výkonového spektra. Pomocí SFM rozhoduje o znělosti segmentu index tonality

$$\delta = \min\left(\frac{\mathrm{SFM}}{-60}, 1\right). \tag{1.31}$$

Spektrální složku považujeme za tónovou v případě, že $\delta > 0{,}5$ [18, str. 20].

1.7 Stupnice a ladění

Jedním z výstupů algoritmu pro přeladňování výšky tónu je možnost opravit intonaci hudebního signálu (viz 1.4.3), tedy změnit výšku tónu tak, aby odpovídal zvolené stupnici. Hudební teorie rozlišuje různé stupnice a ladění.

1.7.1 Stupnice

Definuje řadu tónu uspořádanou podle definovaných pravidel jako je vzdálenost a jejich počet (nejčastěji v rozmezí oktávy). Podle počtu stupňů (tónů) dělíme stupnice například na pětistupňové, šestistupňové nebo sedmistupňové. Vzdálenosti mezi stupni mohou být stejné nebo různé [19, str. 55]. Stupnice, ve kterých se vyskytují celé tóny i půltóny se nazývají diatonické a obvykle obsahují základní tónovou řadu ($c - d - e - f - g - a - h$). V evropské moderní hudbě mají největší význam stupnice molová a durová. Liší se od sebe umístěním půltónových vzdáleností v tónové řadě. Dále rozlišujeme například stupnici chromatickou (dělí oktávu na dvanáct rovnocenných půltónů), cikánskou, pentatonickou nebo celotónovou.

Přirozené ladění

Ladění definuje frekvence tónů ve stupnici a jejich poměry. Mezi nejpoužívanější typy ladění patří přirozené a temperované. V přirozeném ladění mají mezi sebou tóny definován čistý poměr od tóniky v podobě přirozeného čísla. Přestože je sluchový vjem přirozeného ladění příjemný, v praxi je u soudobé hudby zřídka využitelný, neboť omezuje použití nástroje do jedné stupnice. Čistě naladěné intervaly definují jinou frekvenci pro tón C ve stupnici *G dur*, než ve stupnici *A dur*. To znemožňuje aplikaci enharmonické záměny a při modulaci způsobuje rozladění některých intervalů [20, str. 268].

1.7.2 Temperované ladění

Vývoj nástrojů, jejichž konstrukce omezovala nebo vylučovala změnu ladění vedl k poptávce po kompromisním systému ladění. Po systému, který by umožňoval nástroji hrát i v jiných tóninách, než ve kterých je naladěn. Tím vznikl systém rovnoměrného temperování, který označujeme jako temperované ladění [20, str. 269]. To používá oktávu jako čistý interval, kterou dělí na 12 stejných půltónů. Frekvenční poměr pro sousední půltóny lze vyjádřit jako

$$x = \sqrt[12]{2} = 1{,}05946309. \tag{1.32}$$

K vyjádření jemnějších odchylek slouží Ellisovo ladění, které dělí oktávu na 1200 centů. Jeden temperovaný půltón je složen ze 100 centů a jeden cent jakožto jednotka vzdálenosti mezi tóny v temperovaném ladění lze vyjádřit vztahem

$$x = \sqrt[100]{\sqrt[12]{2}} = \sqrt[1200]{2} = 1{,}00057779. \tag{1.33}$$

1.7.3 Normál ladění

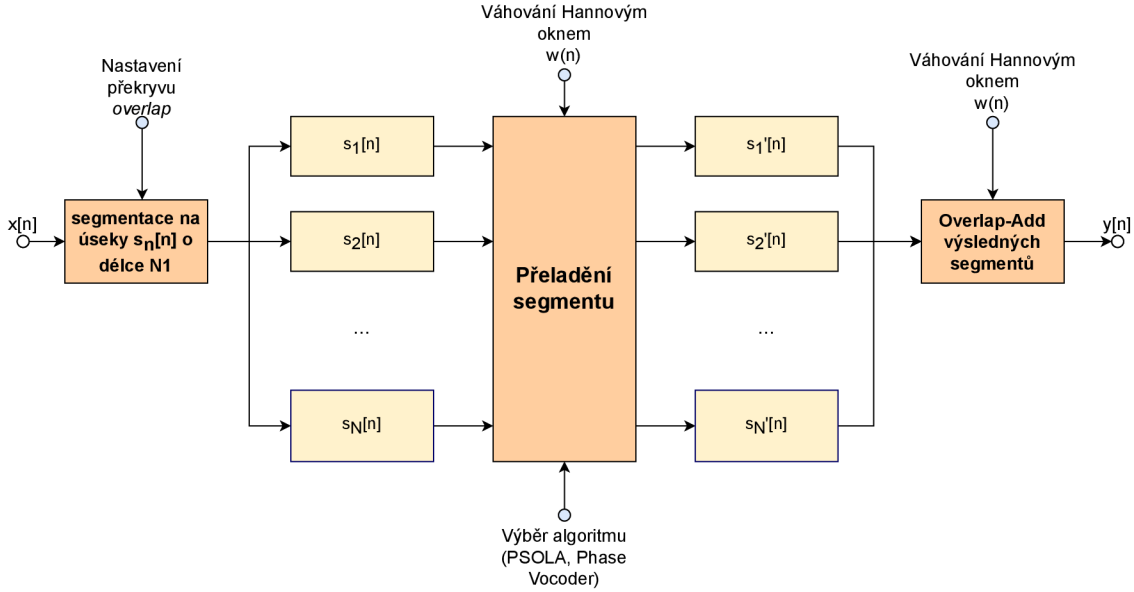
Výškový normál ladění slouží k jednotnému ladění nástrojů a u mnoha nástrojů ovlivňuje už samotnou konstrukci. Je vztahován k tónu a^1 díky jeho výhodné střední poloze v jednočárkované oktávě [20, str. 279]. Výškový normál se historicky měnil, než byl ustanoven na $a^1 = 440\text{Hz}$. Přesto se lze často setkat s laděními, která tento standard nedodržují.

2 Návrh a implementace testovací aplikace

Tato kapitola se zabývá návrhem algoritmů pro přelaďování výšky tónu, jejich realizaci pro platformu Matlab. Jejím výstupem je testovací aplikace sloužící k jejich porovnání. Algoritmy jsou realizovány v režimu *off-line* zpracování dat. To znamená, že je nejprve načten celý zvukový soubor a následně je provedeno zpracování signálu.

2.1 Návrh algoritmu

Schéma navrženého algoritmu znázorňuje obrázek 2.1. Jako metody přelaďování zvukového signálu byly vybrány PSOLA pracující v časové doméně a fázový vokodér založený na STFT a manipulaci se spektrálními složkami. Varianta vokodéru využívající změnu délky signálu a interpolaci je pro zpracování v reálném čase méně vhodná, neboť při přelaďování signálu může být délka vstupního a výstupního segmentu různá (což je kompenzováno různým skokem ve vstupním a výstupním signálu). Metoda pracující přímo v diskrétním spektru poskytuje pro každý vstupní segment stejně dlouhý přeladěný výstupní, a proto byla vybrána k porovnání s algoritmem PSOLA. Signál se nejprve rozdělí na stejně dlouhé segmenty $s_k[n]$ nastavitelné délky a překryvu, které jsou zpracovány algoritmem přeladění výšky. Upravené segmenty $s'_k[n]$ jsou následně sečteny metodou *overlap-add* ve výstupní signál. Zpracování segmentů je prováděno sekvenčně, tedy jednotlivé segmenty jsou zpracovávány jeden po druhém a postupně přičítány k výstupnímu signálu.

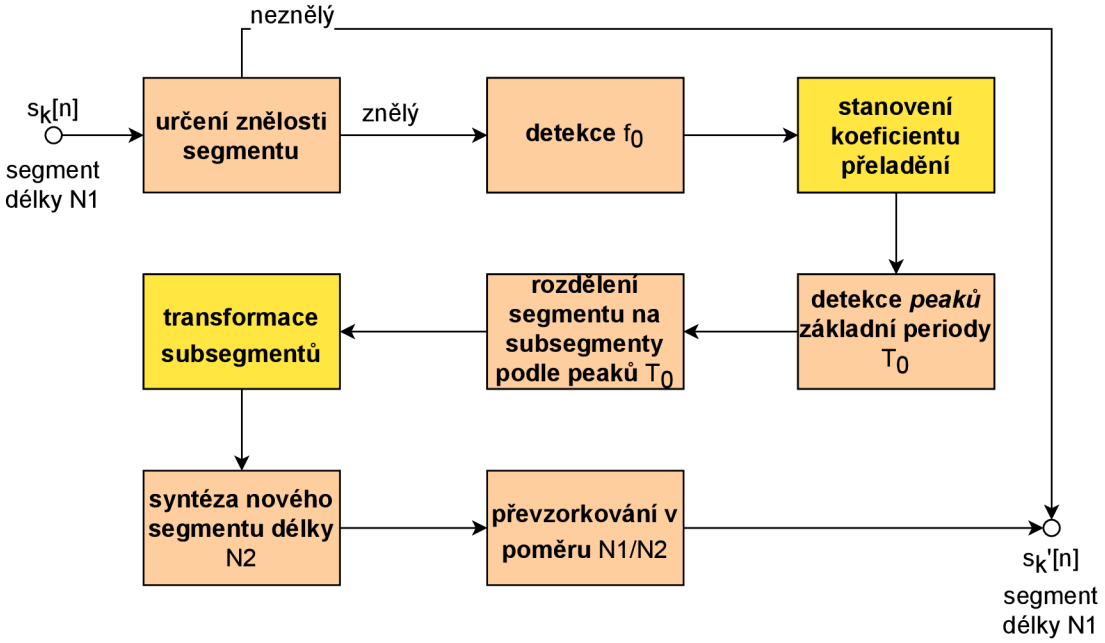


Obr. 2.1: Schéma algoritmu přelaďování

Blok s názvem *přeladění segmentu* sdružuje metody PSOLA a fázový vokodér, jejichž vnitřní struktura se liší. V následujících kapitolách proto budou rozebrány jejich implementační detaily z pohledu přeladění jednoho segmentu. V obou případech je nejprve určena jeho znělost, na základě které se rozhoduje o aplikaci přeladění. U neznělých segmentů nemá smysl, tudíž jsou pouze překopírovány na výstup. Znělé segmenty algoritmus přeladí. PSOLA k němu navíc oproti fázovému vokodéru potřebuje znát základní kmitočet, který lze určovat pomocí některé z metod popsaných v kapitole 1.5. Pro účely práce byly implementovány všechny metody zde popsané. K přeladění nelze použít pouze metodu centrálního klipování, neboť potřebuje znát maximum z předcházejícího a následujícího segmentu. Tato skutečnost lze v reálném čase kompenzovat zpožděním, které však při použití metody zbytečně narůstá o velikost dvou segmentů. Více o zpoždění v kapitole 4.2.

2.1.1 PSOLA

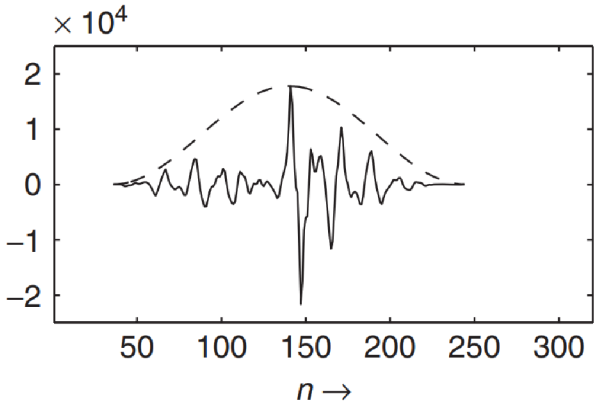
Návrh přeladění segmentu pomocí algoritmu PSOLA popisuje schéma 2.2.



Obr. 2.2: Schéma algoritmu PSOLA

Kvalitu přeladění v reálném čase ovlivňuje přesnost a rychlost detekce základního kmitočtu pomocí vhodné metody. Na jejím základě jsou detekovány pozice špiček základní periody. Jádro algoritmu je stejné a při implementaci různých efektů se mění pouze žlutě zvýrazněné bloky na obrázku 2.2). Koeficient přeladění může být

dán staticky nebo určován dynamicky podle základního kmitočtu signálu (efekt *auto-tune*). Další efekty lze získat rozdílnou transformací subsegmentů (viz obr. 2.3).

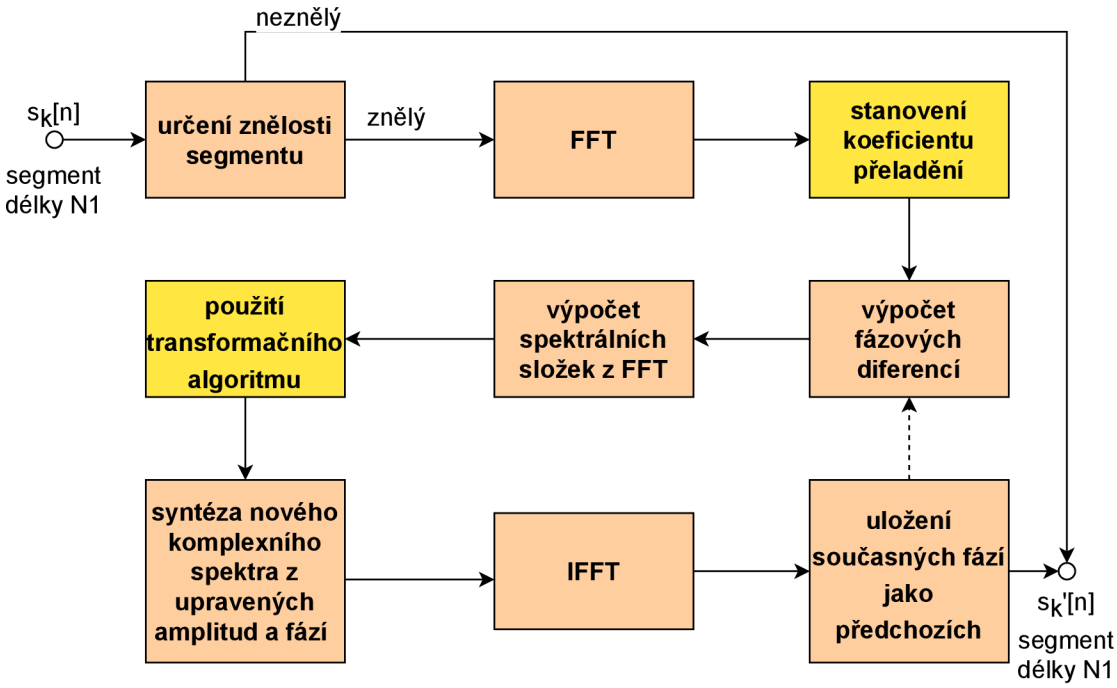


Obr. 2.3: Subsegment o délce dvou základních period s centrem ve špičce základní periody váhovaný Hannovým oknem (převzato z [10, str. 208])

Škálováním pozic špiček základní periody, které se nachází ve středu každého subsegmentu, následovaného převzorkováním bude dosaženo posunutí formantových oblastí při zachování původního základního kmitočtu. Prodloužení signálu zopakováním některých subsegmentů následované převzorkováním způsobí posunutí základního tónu. Pokud bude časová komprese resp. expanze prováděna pevným poměrem, dojde k rovnoměrnému posunutí základního kmitočtu napříč celým signálem. Další možností je implementovat efekt *auto-tune*, tedy řídit transpozici signálu v čase podle poměru detekovaného základního kmitočtu a kmitočtu nejbližšího nalezeného půltónu.

2.1.2 Fázový vokodér

Návrh přeladění segmentu pomocí fázového vokodéru popisuje schéma 2.4. K dosažení různých efektů slouží stejné bloky jako v algoritmu PSOLA. I efekt *auto-tune* je realizován obdobně. Transformační algoritmy tentokrát nemanipulují se vzorky segmentu, ale s fázemi složek ve spektru. Například nastavením fází všech složek na nulu vznikne efekt robotizace. Škálování kmitočtů všech složek spektra způsobí přeladění výšky tónu. Rozlišení spektra však výjimečně odpovídá přesným hodnotám složek v signálu. Protože je před tím potřeba z komplexního spektra získat informaci o skutečných hodnotách kmitočtů vyšších harmonických složek (viz 1.4.2).



Obr. 2.4: Schéma algoritmu fázový vokodér

2.1.3 Nalezení nejbližšího půltónu

Pro korekci intonační křivky je potřeba rozhodnout, jestli se detekovaný tón nachází ve zvolené tónině nebo ne. K odvození funkce, která by byla schopna najít nejbližší půltón v tónině je možné vyjít ze vztahu poměrů intervalů definovaných pro temperované ladění (viz. 1.7.2). Nechť f_{ref} je frekvenční normál a f_{k} frekvence vzdálená od normálu o k půltónů, pak platí

$$f_{\mathrm{k}} = f_{\mathrm{ref}} \cdot (\sqrt[12]{2})^{\mathrm{k}} = f_{\mathrm{ref}} \cdot 2^{\frac{\mathrm{k}}{12}}. \tag{2.1}$$

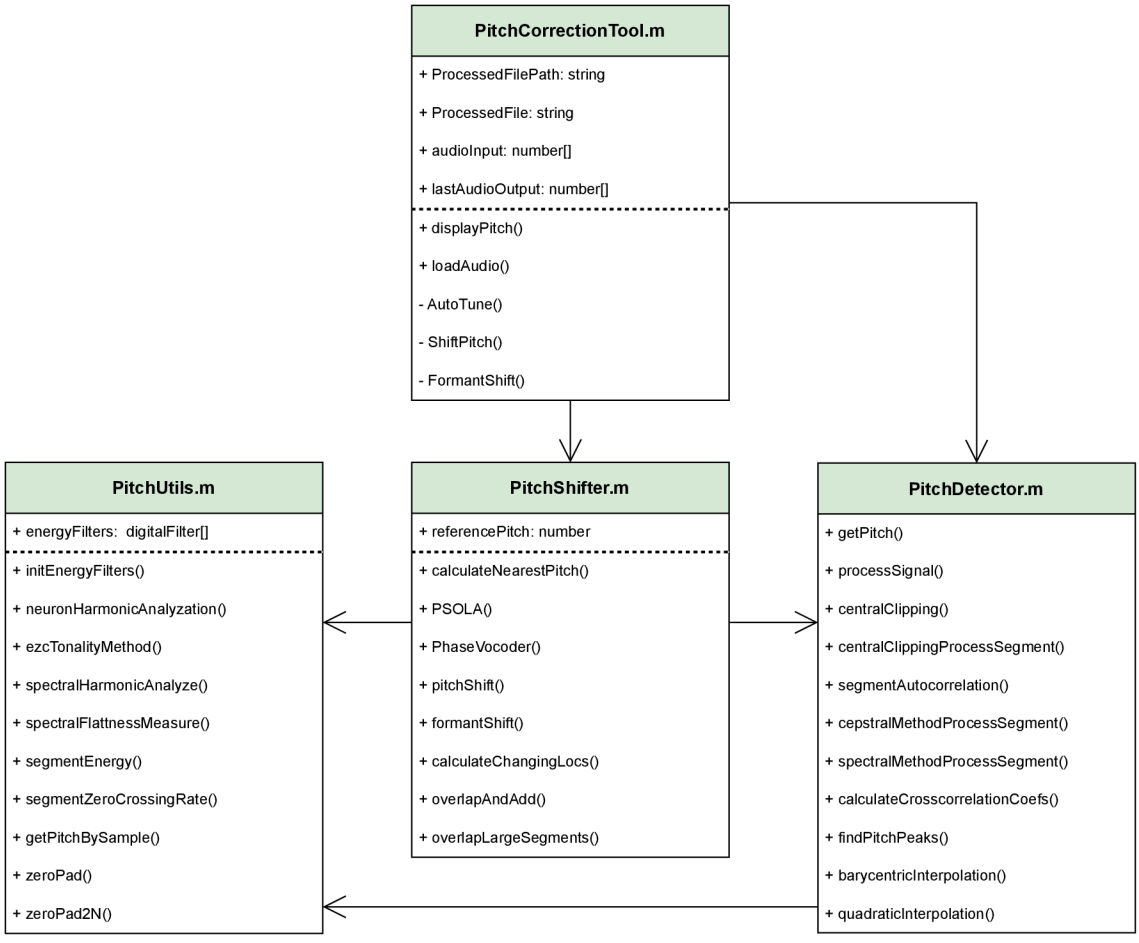
Pokud by f_k byla detekovaná frekvence, lze spočítat počet půltónů, o které je tato vzdálena od frekvenčního normálu úpravou předchozího vztahu

$$k = \log_2 \left(\frac{f_{\mathrm{k}}}{f_{\mathrm{ref}}} \right) \cdot 12. \tag{2.2}$$

Pokud je k celé číslo, pak se jedná o čistý interval podle temperovaného ladění a není třeba měnit intonaci segmentu. Pokud by hodnota k byla neceločíselná, pak se nejbližší půltón z tóniny dopočítá zaokrouhlením k na nejbližší celočíselnou hodnotu. Následným dosazením zpět do rovnice 2.1.3 je získána frekvence nejbližšího půltónu v chromatické stupnici. Omezením hodnot, kterých může k nabývat, lze definovat mnoho dalších odvozených stupnic, ve které bude algoritmus intonační korekce pracovat.

2.2 Implementace algoritmů pro platformu Matlab

Pro implementaci algoritmu pro platformu Matlab bylo využito interaktivního vývojového prostředí App Designer, které disponuje funkcemi pro návrh uživatelského rozhraní. Nabízí i možnost exportovat aplikaci jako standardní spustitelný Matlab skript. Jedním z výstupů této práce je aplikace `PitchCorrectionTool`, která demonstruje popisované algoritmy a metody pro přelaďování zvukového signálu. Třídy, které zobrazuje diagram na obrázku 2.5 tvoří jádro této aplikace a budou proto podrobněji rozebrány.



Obr. 2.5: Diagram tříd

2.2.1 PitchUtils

`PitchUtils` sdružuje funkce pro určení znělosti segmentu a další podpůrné funkce, jako například doplnění segmentu nulami. K určení znělosti je možné využít čtyři funkce, které implementují algoritmy popsané v 1.5.4. `neuronHarmonicAnalyzation` je obdobou metody neuronu, `spectralFlatnessMeasure` implementací metody spektrální plochosti a `spectralHarmonicAnalyzer` je metoda založená na rozložení energie v pásmech. Poslední jmenovaná metoda je stavová, neboť ke své práci používá banku digitálních filtrů (atribut `energyFilters`). Před použitím této metody je proto nejprve potřeba inicializovat filtry funkcí `initEnergyFilters`.

2.2.2 PitchDetector

Třída `PitchDetector` zapouzdřuje metody detekce základního kmitočtu. Ke zpracování signálu slouží obecná metoda `getPitch`, která provádí segmentaci vstupního signálu, váhování oknem a překryv segmentů. Jednotlivé segmenty jsou následně zpracovány konkrétními metodami detekce, které odpovídají metodám nastíněným v kapitole 1.5. Výjimku tvoří pouze metoda centrálního klipování. Ta používá k analýze vlastní segmentaci kvůli zpožděnému zpracování vstupu, které je zapříčiněno nutností analyzovat maxima předchozího a nadcházejícího segmentu. Analýza signálu tedy neprobíhá, dokud nejsou do paměti načteny alespoň dva po sobě jdoucí segmenty vstupního signálu.

Vedle metod detekce výšky tónu nabízí `PitchDetector` funkci `findPitchPeaks`. Jedná se o funkci, kterou používá PSOLA, avšak díky své obecnosti a provázanosti s metodami detekce výšky tónu byla přesunuta do třídy `PitchDetector`. Funkce nejprve detekuje základní frekvenci, a pak hledá pomocí autokorelační funkce špičky vzdálené od sebe o délku základní periody s určitou mírou tolerance.

2.2.3 PitchShifter

`PitchShifter` představuje třídu, která spojuje atributy a metody pro přeladění výšky tónu. Implementuje například metodu `calculateNearestPitch` pro nalezení nejbližšího půltónu v tónině pro zadanou vstupní frekvenci, popsanou v sekci 2.1.2. K ní se váže atribut `referencePitch` reprezentující výškový normál ladění. V základu je nastaven na evropský standard 440 Hz.

K manipulaci se spektrem vstupního signálu jako například přeladění výšky tónu nebo posunutí formantů slouží metoda `pitchShift`. Ta pracuje ve dvou módech, které se volí vstupním argumentem `mode`. Prvním z nich je *pitch-shifting* k přeladění výšky celého signálu pevným poměrem. Druhý mód s názvem *auto-tune* detekuje pro každý segment jeho základní frekvenci a v případě chybné intonace ležící

mimo chromatickou stupnici s normálem nastaveným v atributu `referencePitch` se jej snaží přeladit k nejbližšímu půltónu. Posledním módem funkce je *formant-shifting* neboli přeladění formantů se zachováním původního základního kmitočtu, která však funguje pouze s algoritmem PSOLA. Výstup funkce `pitchShift` tedy může být různý, samotná však funguje spíše jako *wrapper* pro segmentaci a syntézu pomocí *overlap-add*. O přeladění dílčích segmentů se starají metody `PSOLA` a `formantShift`. Metoda `PhaseVocoder` představuje celou implementaci algoritmu od segmentace až po výsledný výstupní signál. Nakonec nebyla implementována do společného jádra kvůli snazšímu ukládání stavových parametrů při výpočtu, jako je informace o předchozí fázi.

Dalším klíčovým argumentem funkce je řetězec `PDMethod` nesoucí název metody pro detekci základního kmitočtu. Jedná se o některou z metod, která je součástí třídy `PitchDetector`, a na kterou má třída `PitchShifter` referenci. Výstupem metody `pitchShift` je přeladěný signál.

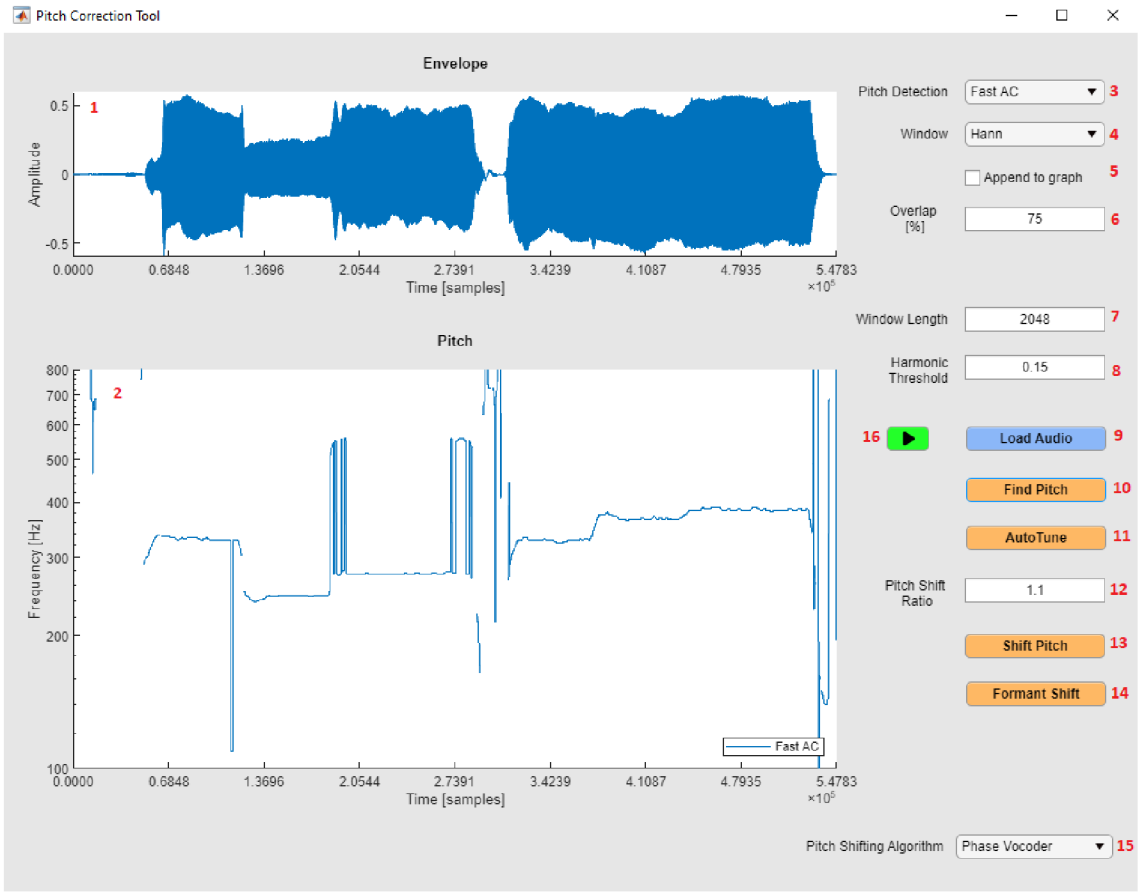
Významnou funkcí je i `calculateChangingLocs`, součást metody PSOLA. Analyzuje špičky základní periody předané jako vstupní argument a vypočte pozice těch, které je potřeba duplikovat resp. zahazovat při časové expanzi segmentu. Jejím výstupem je tedy vektor pozičních značek, na kterých je potřeba duplikovat resp. zahazovat. Je-li nutné některé subsegmenty zduplikovat vícekrát, bude výstupní vektor pozic obsahovat stejný počet duplicit odpovídající počtu duplikací subsegmentu. Pokud je potřeba dvakrát duplikovat subsegment, jehož střed leží ve vstupním segmentu na vzorku 225, do výstupního vektoru se dvakrát za sebou zapíše hodnota 225.

2.2.4 PitchCorrectionTool

Jedná se o hlavní třídu, kterou je aplikace spouštěna. Definuje komponenty uživatelského rozhraní, funkce pro načítání a přehrávání hudebních souborů, vykreslení grafů na obrazovku a *callback* funkce komponent. Pamatuje si poslední výstup a umožňuje tak přehrát poslední zpracovaný hudební signál.

2.3 Ovládání aplikace

Cílem této sekce je popsat uživatelské rozhranní a vysvětlit ovládání testovací aplikace vzniklé v prostředí Matlab. Po spuštění skriptu `PitchCorrectionTool.m` se zobrazí hlavní obrazovka aplikace, která se skládá z grafů vykreslujících obálku a ve stejném měřítku i průběh základní frekvence v levé části. Pravá část obsahuje tlačítka menu a nastavitelné hodnoty parametrů. Následující popisky odkazují na číselné indikátory komponent v obrázku 2.6.



Obr. 2.6: Náhled uživatelského rozhraní s identifikátory

1 Graf pro vykreslení amplitudové obálky signálu. Rozlišení os je v počtu vzorků.

2 Graf pro vykreslení průběhu základního kmitočtu. Rozsahy časových os grafů jsou vzájemně synchronizované.

3 Rozbalovací okno, které slouží k výberů metod detekce základního kmitočtu.

4 Typ váhovacího okna, který bude použit při segmentaci signálu, jak při detekci základního tónu, tak při přeladování signálu.

5 Zaškrtávací parametr *Append to graph* přepne režim vykreslování grafu průběhu základního kmitočtu z přepisování na přidávání. Jakákoliv akce, která povede k vykreslení hodnot do tohoto grafu nepřepíše původní vykreslené křivky. Parametr lze použít například pro porovnání metod detekce základního tónu. Po zaškrtnutí se budou jednotlivé funkce vykreslovat přes sebe. Legenda se nepřepisuje a parametry se vykreslují na konec.

6 Nastavení procentuálního překrytí segmentů. Musí být v rozsahu 0–85.

7 Délka okna společná i pro délku analyzovaného segmentu.

8 Číselný práh pro určení znělosti segmentu. Jeho překročením je segment vyhodnocen jako znělý.

9 Načtení zvukového souboru do paměti.

10 Detekuje průběh základního kmitočtu pro načtený signál a vykreslí ho do příslušného grafu.

11 Provede korekci intonace vstupního signálu a vykreslí průběh základního kmitočtu výstupního signálu.

12 Koeficient přeladění výšky tónu. Koeficient $k > 1$ naznačuje zvýšení základního kmitočtu, koeficient $k < 1$ naopak snížení.

13 Posunutí základního kmitočtu v poměru *Pitch Shift Ratio* v celém vstupním signálu.

14 Posunutí formantů v poměru *Pitch Shift Ratio*. Kofeicient $k > 1$ značí posunutí formantů k vyšším kmitočtům, pro $k < 1$ naopak.

15 Výběr algoritmu pro přeladění.

16 Přehrávání naposled zpracovaného výstupního signálu. Pokud nebyl dosud zpracován žádný signál, přehraje se vstup.

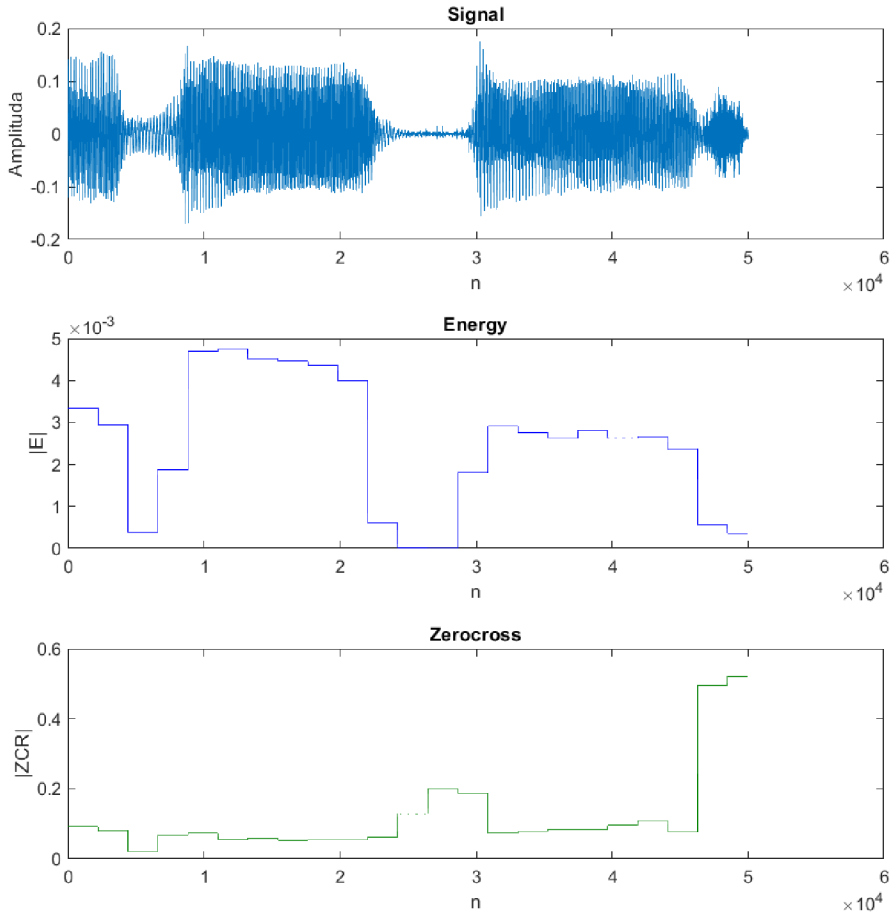
Nastavení hodnot některých parametrů jako například *Harmonic Threshold* (2.3) má dopad na kvalitu přeladění v časové doméně, ale jeho nastavení může být neintuitivní. V kapitole 3 je proto sekce věnovaná nastavení optimálních prahových hodnot pro detekci znělosti. Přestože algoritmy umožňují pracovat se stereo zvukovými soubory, zvukový signál se interně převádí na mono a jejich výstupem je tedy též mono signál.

3 Výběr algoritmů

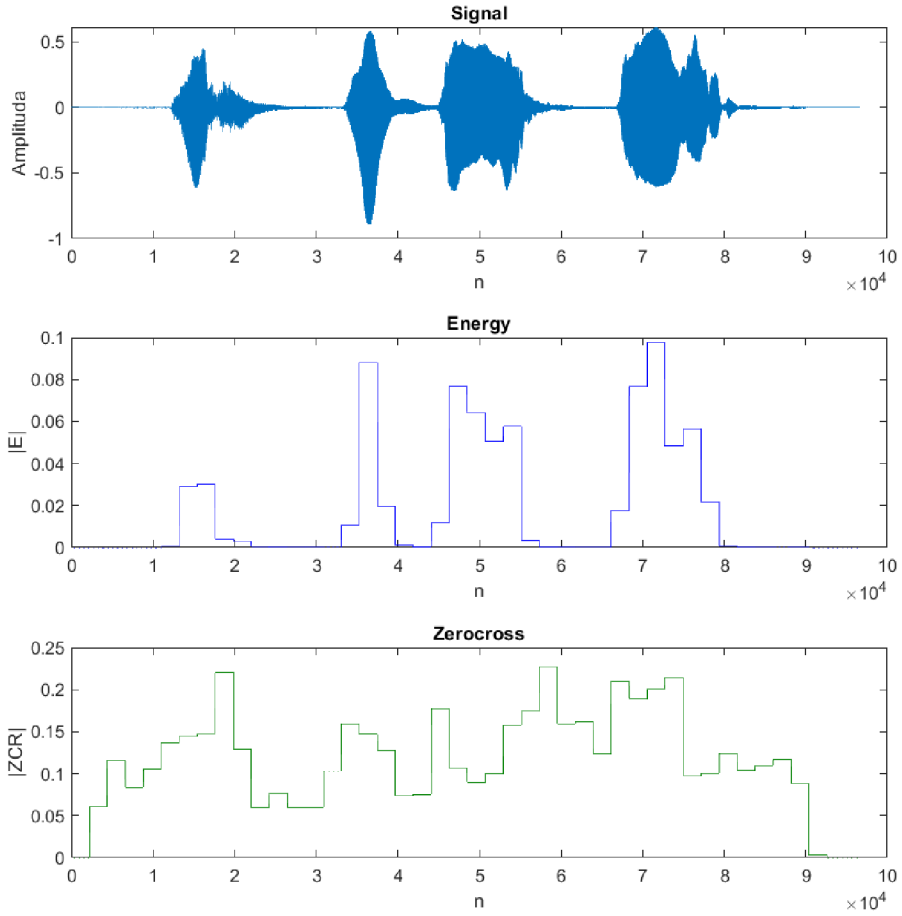
Tato kapitola obsahuje subjektivní hodnocení implementovaných metod pro přeladování, detekci výšky tónu a znělosti. Porovnává jejich přesnost a omezení.

3.1 Znělost

Pro správnou detekci znělosti je potřeba empiricky určit prahové hodnoty, po kterých je segment považován za znělý. Tato část se proto věnuje jejich odhadu. Pro účely testování detekce znělých segmentů v signálu byly vybrány soubory *WhatsernameHarmonicTest.wav*, *Violin_F#m.wav* a *Guitar_F#min.wav*, které jsou součástí příloh.



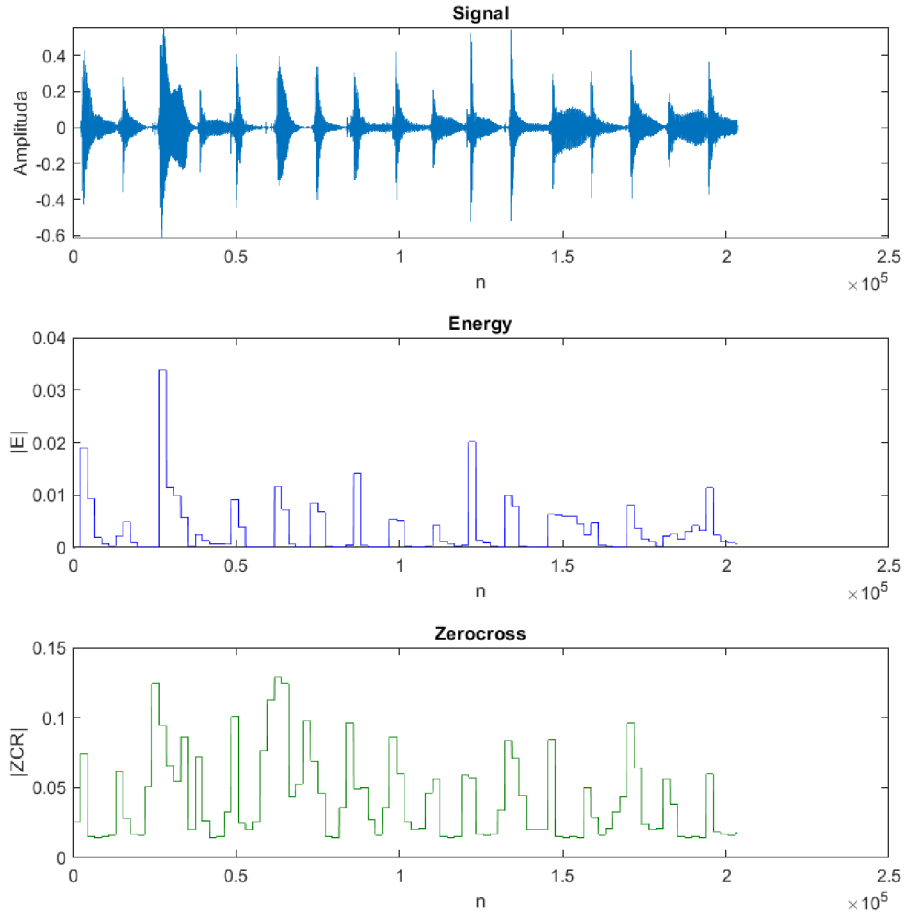
Obr. 3.1: Porovnání hodnot krátkodobé energie a ZCR – zvukový záznam zpěvu



Obr. 3.2: Porovnání hodnot krátkodobé energie a ZCR – zvukový záznam houslí

Na obrázcích 3.1, 3.2 a 3.3 jsou znázorněny průběhy jejich zvukových signálů, ve stejném měřítku i krátkodobá energie (E) a počet průchodů nulou úrovní (dále jen ZCR). Průběhy krátkodobé energie i ZCR odpovídají ve všech třech případech očekávání. Energie je pro ticho a neznělé části nízká, hodnota ZCR naopak vysoká. Nejlépe je to vidět na obrázku 3.1 v poslední části obálky. Jedná se o souhlásku „s“. V ostatních nahrávkách lze pozorovat zvýšenou hodnotu ZCR v místech, kde dochází k ruchům či harmonickému zkreslení. V průběhu elektrické kytary lze spatřit zvýšenou hodnotu ZCR v místech, kde došlo ke zkreslení.

Rozsahy hodnot, ve kterých se veličiny pohybují se však liší. V řečovém signálu je průměrná hodnota energie nižší, než v nahrávce houslí a kytary. Porovnání průměrných a maximálních hodnot zobrazuje tabulka 3.1. Hodnota krátkodobé energie je závislá na hlasitosti vzorku. Čím hlasitější segment, tedy vyšší číselné hodnoty vzorků, tím vyšší bude i krátkodobá energie.



Obr. 3.3: Porovnání hodnot krátkodobé energie a ZCR – zvukový záznam elektrické kytary

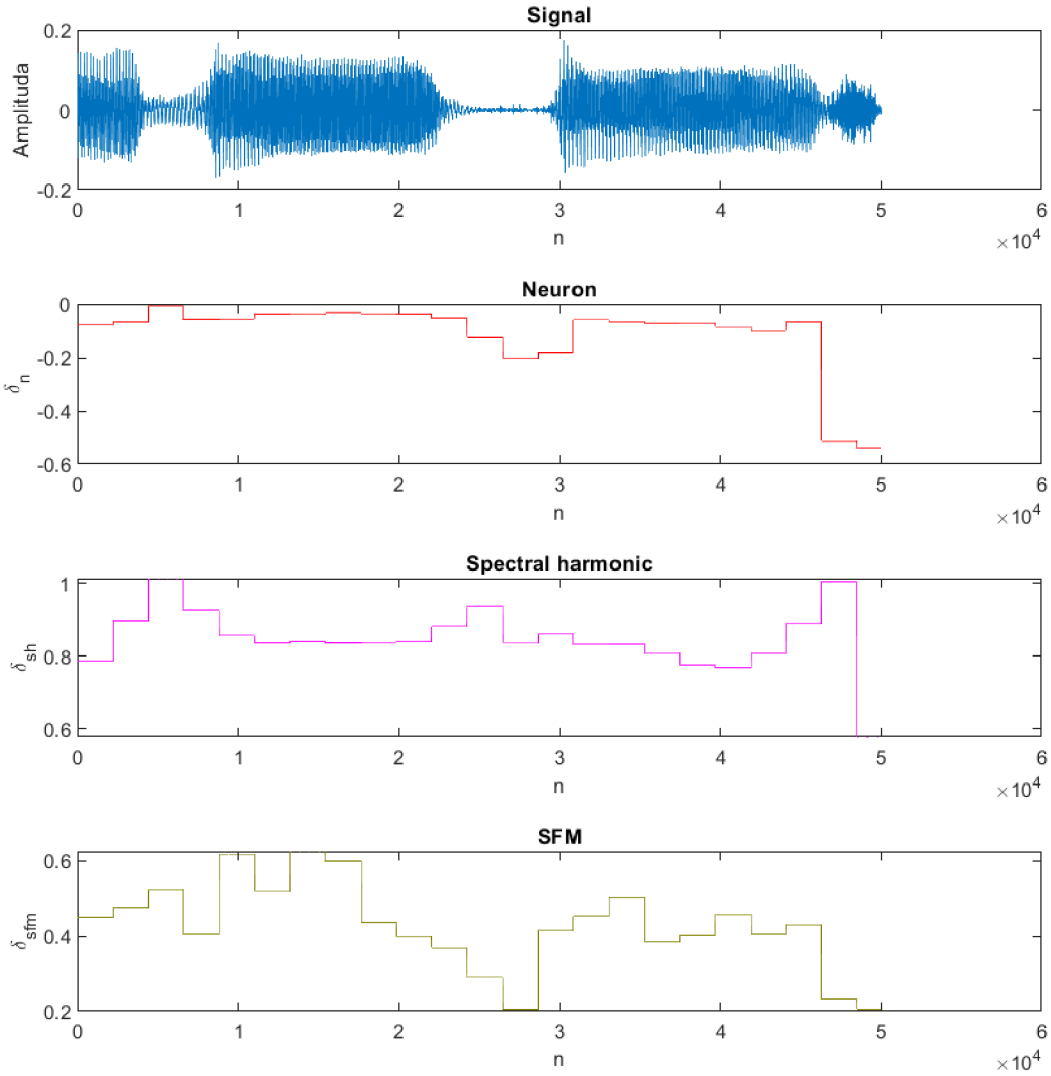
Počet průchodů nulou je na hlasitosti nezávislý, ale citlivý na přídavný šum. Čím nižší odstup od šumu bude zvukový signál mít, tím vyšší bude hodnota ZCR. Hodnotu krátkodobé energie je tedy vhodné normalizovat, například podle maximální špičky.

Tab. 3.1: Porovnání hodnot energií a ZCR pro nahrávky

zdroj zvuku	mean(E)	max(E)	mean(ZCR)	max(ZCR)
zpěv	0,0025	0,0047	0,1159	0,5205
housle	0,0176	0,0976	0,1169	0,2277
kytara	0,0034	0,0338	0,0418	0,1288

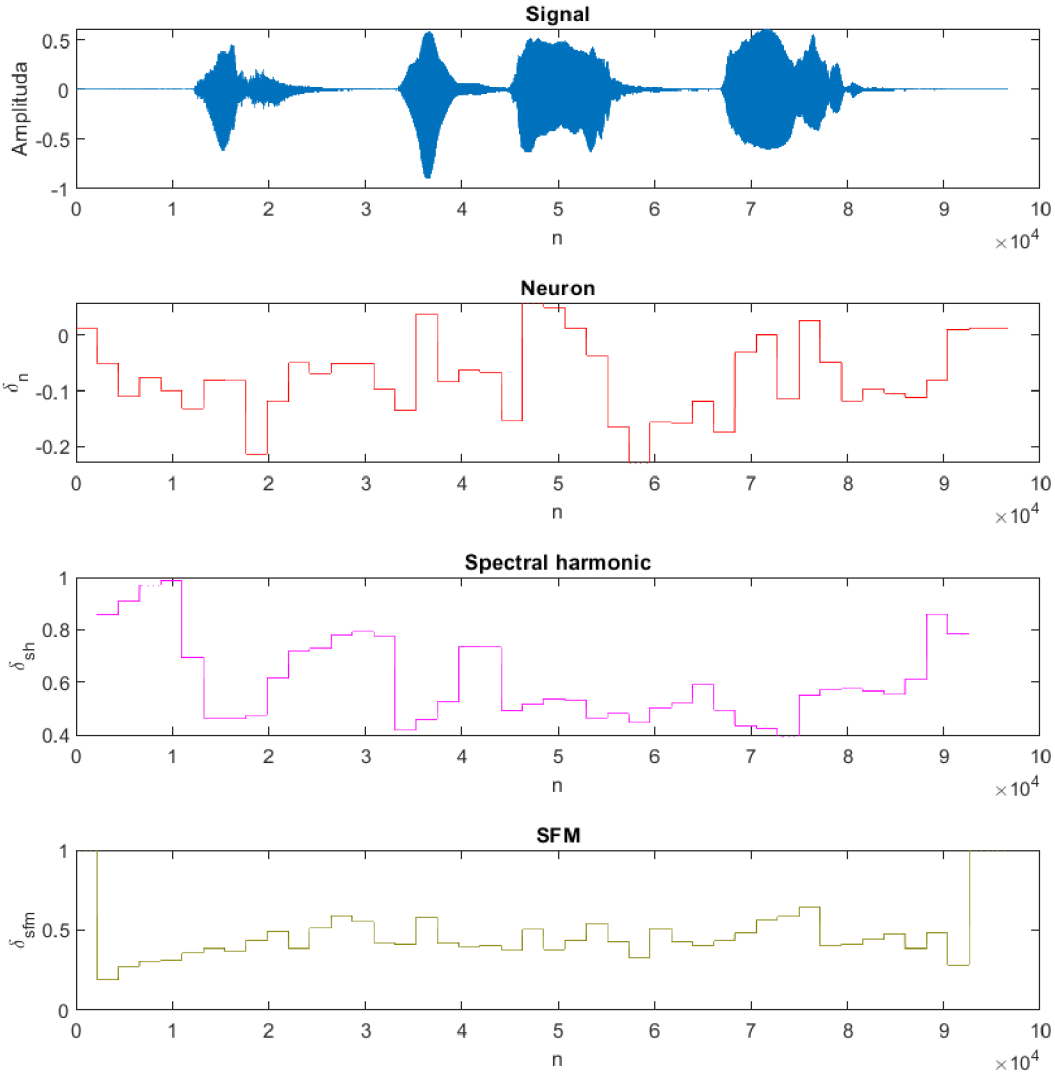
3.1.1 Srovnání metod znělosti

Tato část se věnuje porovnání tří metod určení znělosti – metoda neuronu, poměr energií ve spektru (v grafech *Spectral harmonic*) a spektrální plochost (SFM). První dvě jmenované vycházejí z výpočtu energie a ZCR. Pro testování byly použity stejné zvukové vzorky v předchozí části.



Obr. 3.4: Testování metod na určení znělosti segmentu – zvukový záznam zpěvu

Metody popsané v této práci vypočítají jedinečnou hodnotu (dále index tonality), která při porovnání s prahovou hodnotou rozhoduje o znělosti segmentu.

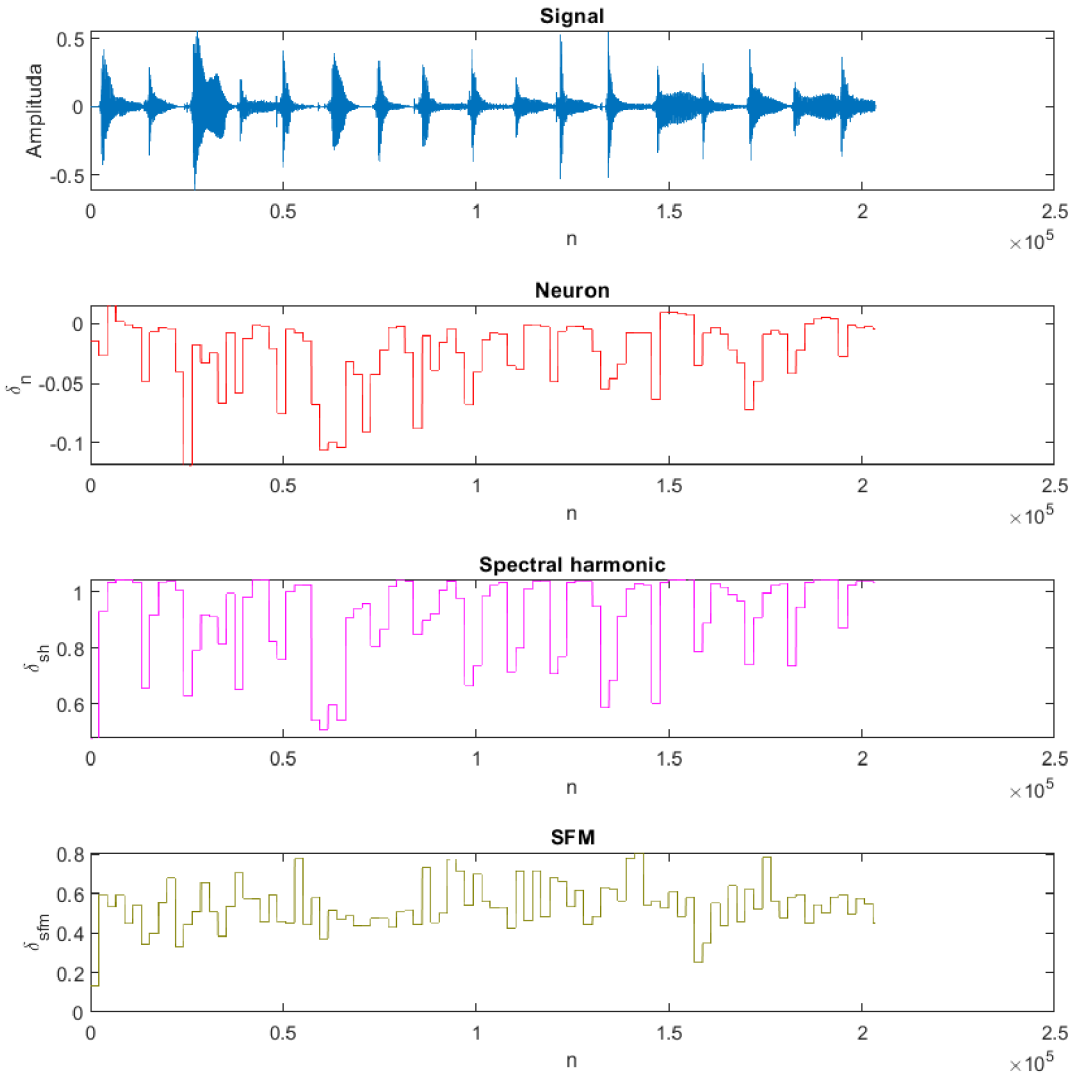


Obr. 3.5: Testování metod na určení znělosti segmentu – zvukový záznam houslí

K výpočtu znělosti a vykreslení grafů byl použit skript `voicedEstimationPlot.m`. Z grafů na obrázcích 3.4, 3.5 a 3.6 vyplývá, že index tonality je pro metodu neuronu a SFM vyšší pro znělé segmenty. Pro metodu poměru energií ve spektru platí opačný vztah, tedy index tonality je pro znělé segmenty menší než pro neznělé.

Tab. 3.2: Porovnání indexu tonality pro metody znělosti, kde δ_{n} je index tonality pro metodu neuronu, δ_{sh} index tonality pro metodu poměru energií ve spektru a δ_{sfm} index tonality pro metodu spektrální plochosti

zvuk	mean(δ_{n})	max(δ_{n})	mean(δ_{sh})	max(δ_{sh})	mean(δ_{sfm})	max(δ_{sfm})
zpěv	-0,1052	-0,0094	0,8508	1,0137	0,4294	0,6246
housle	-0,0752	0,0574	0,6095	0,9876	0,4700	1
kytara	-0,0244	0,0157	0,9123	1,0455	0,5363	0,8066



Obr. 3.6: Testování metod na určení znělosti segmentu – zvukový záznam kytary

V tabulce 3.1.1 jsou znázorněny průměrné a maximální hodnoty indexů tonality pro jednotlivé nahrávky. Z testování nejlépe vychází metoda neuronu, která vykazuje největší odstup indexu tonality pro znělé a neznělé úseky. Její nevýhodou je však závislost prahu na hlasitosti signálu, což lze řešit normalizací. Nejhůře pak metoda SFM, jejíž malý odstup tonality znělých a neznělých segmentů činí nastavení prahu obtížným. V praxi se jako optimální použitelná hodnota pro určení znělosti segmentu metodou neuronu nejlépe uplatnil práh:

$$\delta_{\text{thr}} = 0{,}2$$

3.2 Srovnání metod detekce výšky tónu

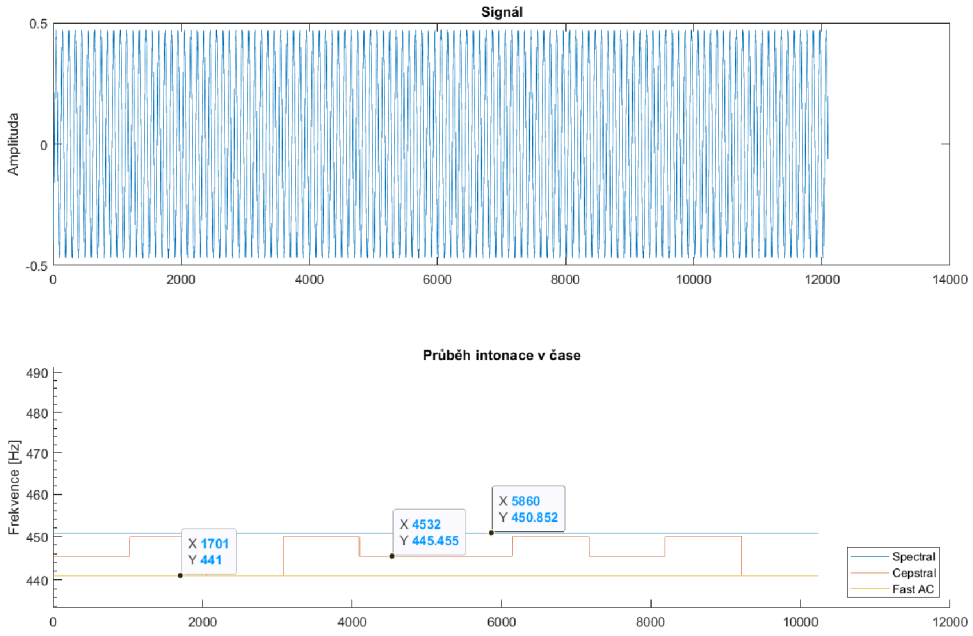
Pro algoritmy detekce výšky tónu je klíčová přesnost odhadu a rychlost výpočtu. K měření rychlosti byl použit skript `pitchDetectionCompare.m`, který vykresluje průběhy intonace pro zvolený vstupní soubor a měří čas výpočtu základní frekvence pro vstupní signál. V této práci bylo implementováno celkem 5 metod, které jsou spolu s časy zobrazeny v tabulce 3.2. Pro detekci byla nastavena délka okna 2048 vzorků a překrytí 50 %.

Tab. 3.3: Porovnání rychlosti metod detekce výšky tónu

Metoda detekce	rychlost výpočtu [s]
Autokorelace	0,950369
Centrální klipování	0,891108
Spektrální metoda	0,685546
Kepstrální metoda	0,743416
Rychlá autokorelace	0,713661

Nejrychlejší jsou **spektrální metoda** a **rychlá autokorelace**. Dalším parametrem je přesnost výpočtu. Ta se testuje obtížně, neboť určovat základní kmitočet ve zvukovém signálu má smysl pouze pro jeho znělé části a tudíž přesnost závisí na robustní metodě určení znělosti signálu. Bez ní se ve spektru objeví chybně detekované frekvence pro neznělé úseky a grafy budou špatně čitelné. Díky závislosti na znělosti nelze spolehlivě měřit ani odchylku od referenční hodnoty. Pro demonstraci omezení některých algoritmů byl proto zvolen jednoduchý sinusový signál s kmitočtem 440 Hz, který je zaznamenán v testovacím souboru *a1-440(short).wav*.

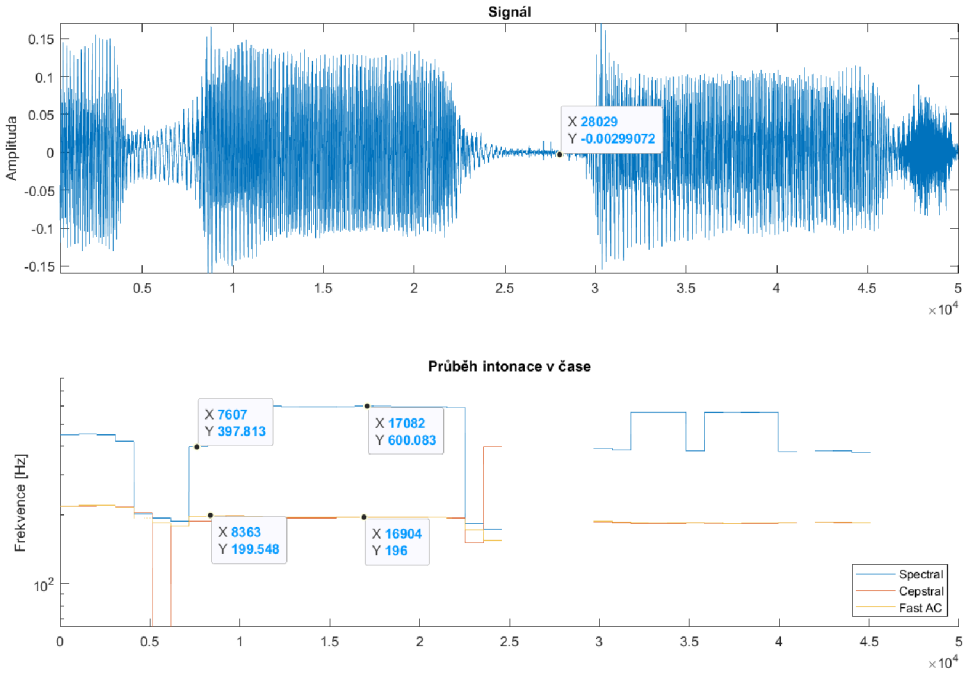
Obrázek 3.7 porovnává tři metody detekce výšky tónu – rychlá autokorelace, spektrální metoda, kepstrální metoda a demonstruje jedno ze zmíněných omezení spektrální metody. Její přesnost se odvíjí z rozlišení FFT, tedy z délky okna.



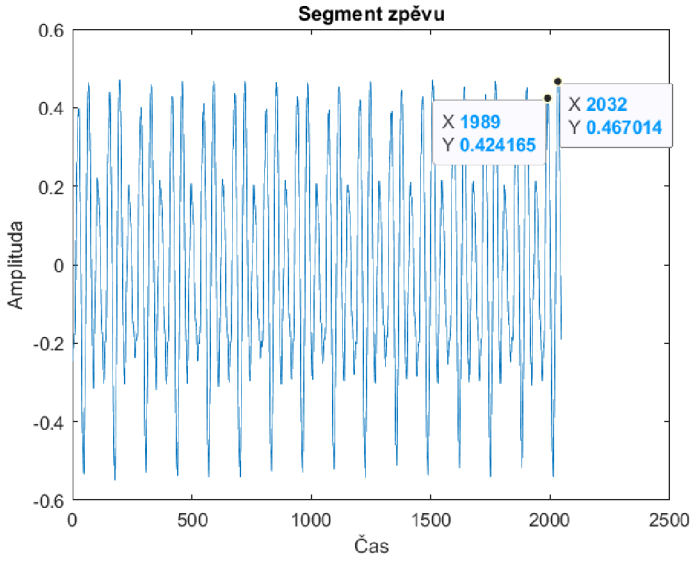
Obr. 3.7: Porovnání metod detekce výšky na harmonickém signálu. Spektrální metoda detekovala základní kmitočet, který se liší o 11 Hz od reference. Rychlá autokorelace má v porovnání odchylku 1 Hz.

Při rozlišení 2048 vzorků je tedy spektrum rozděleno na bloky, jejichž vzdálenost při vzorkovacím kmitočtu 48 kHz odpovídá rozlišení$f_r = \frac{48000}{2048} = 23{,}44$ Hz.

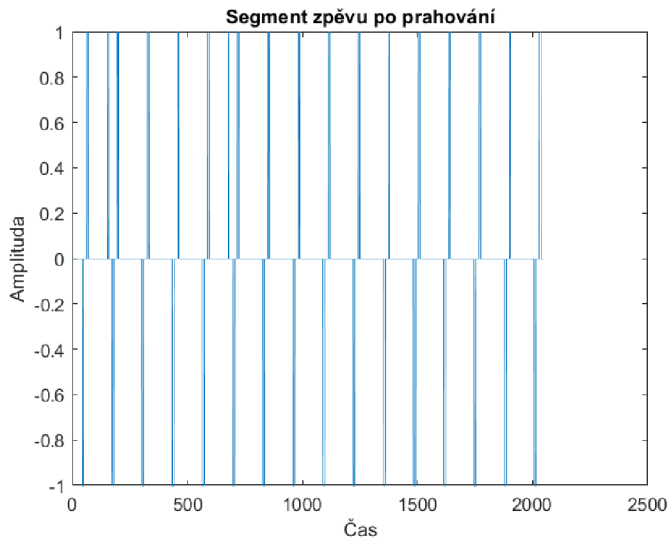
Rozlišení spektra FFT je lineární, ale vnímání výšky tónu lidským sluchem je logaritmické. Čím nižší kmitočet, tím nepřesnější metoda bude. V této práci byl zvolen postup pomocí hledání maxima ve spektru. V případě, že však fundament nemá nejvyšší modul ve spektru, dojde k chybné detekci základního kmitočtu (*octave-error*, viz obr. 3.8). Podobný problém se projevuje i v metodě centrálního klipování, kde je způsoben prahováním a normalizací segmentu. První vrchol na obrázku 3.9 patří vyšší harmonické složce, která je velikostí amplitudy velmi blízko fundamentu. Tato složka projde prahováním a je normalizována spolu s fundamentem (viz obrázek 3.10). Nejvyšší vrchol autokorelační funkce segmentu se nachází na druhé pozici místo první, kvůli čemuž je detekována dvojnásobná základní perioda, ze které je vypočten o oktávu nižší kmitočet, než je skutečný (viz. 3.11). Kepstrální metoda dosahuje dobrých výsledků pro řečové signály. U hudebních, nástrojů jako jsou například housle, však dochází k podobným potížím jako u frekvenční metody a centrálního klipování. Nejpřesnějších a nejrychlejších výsledků v čase dosahuje **metoda rychlé autokorelace**, která bude použita ve výsledném VST modulu.



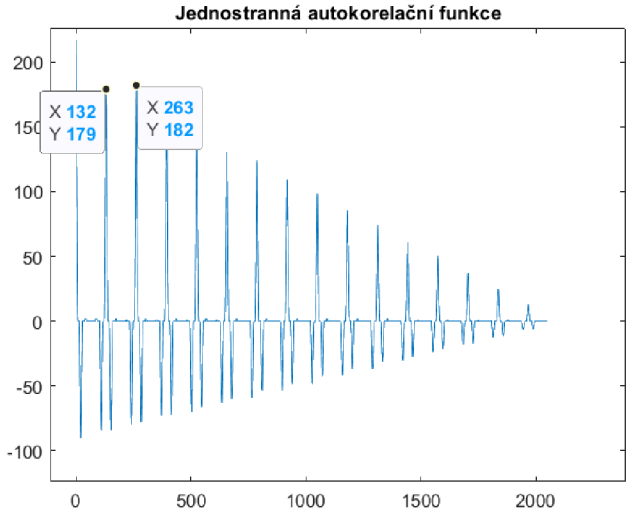
Obr. 3.8: Porovnání metod detekce výšky na záznamu zpěvu. Spektrální metoda detekovala základní kmitočet, který je vzdálen dvě oktávy od fundamentu.



Obr. 3.9: Detekce metodou centrálního klipování. Ukázka segmentu zpěvu ze souboru *backingVocal(outtake).wav*



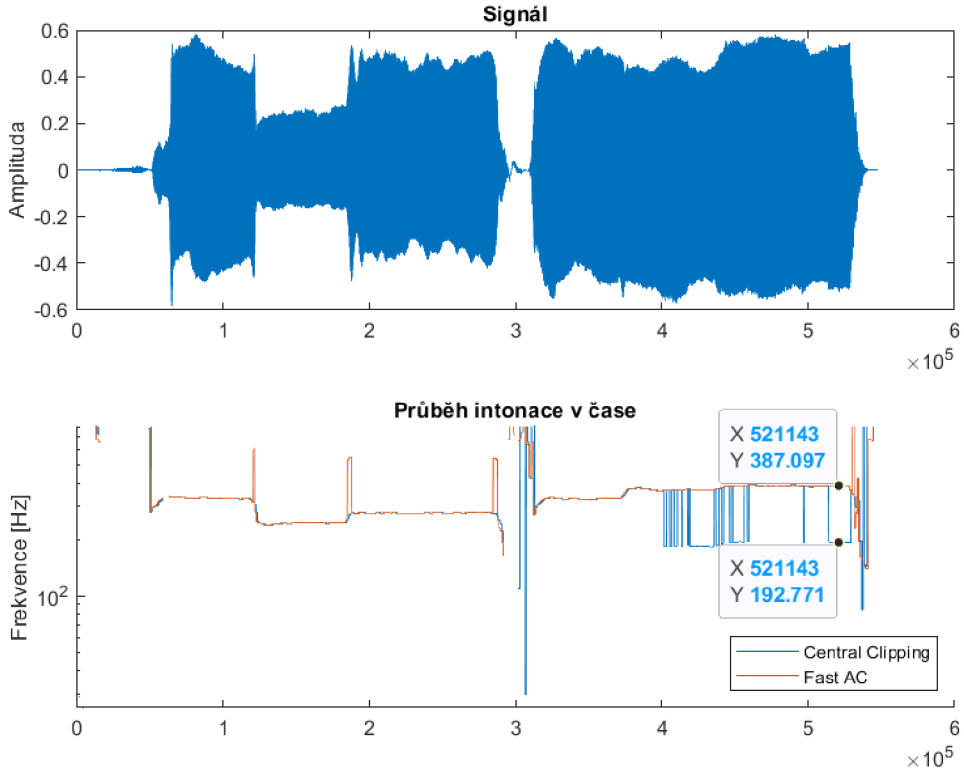
Obr. 3.10: Segment z obrázku 3.9 po normalizaci a prahování.



Obr. 3.11: Jednostranná autokorelační funkce vypočtená pro segment z obrázku 3.10.

3.3 Přeladďování signálu

Poslední část hodnocení se věnuje metodám přeladďování a jejich parametrům. V rámci bakalářské práce byly implementovány metody PSOLA a fázový vokodér. Obě umožňují přeladění vstupního signálu škálováním základního kmitočtu a efekt *auto-tune*. PSOLA nabízí i variantu přeladění formantů při zachování výšky tónu. Změny v časové a frekvenční oblasti byly demonstrovány na souboru *backingVocal(outtake).wav*.

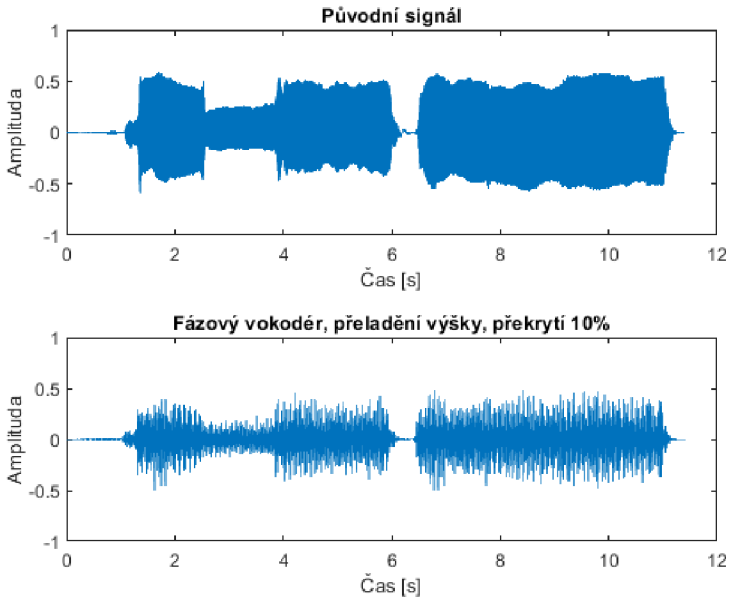


Obr. 3.12: Porovnání přesnosti metody centrálního klipování a rychlé autokorelace. Centrální klipování má v bodech znázorněných na obrázku odchylku jedné oktávy od referenční výšky tónu.

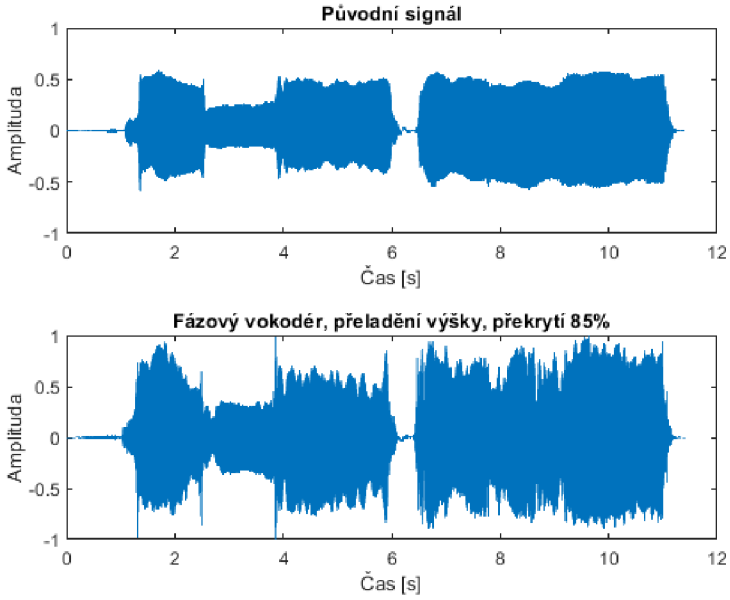
3.3.1 Délka okna

V testovací aplikaci pro platformu Matlab byla implementována možnost variabilního nastavení některých parametrů. Jde především o délku časového okna a míru překrytí segmentů, které mají značný podíl nejen na kvalitě, ale i na rychlosti zpracování. Vyšší překrytí segmentů zanáší méně artefaktů do výstupního signálu, ale snižuje výpočetní výkon. Podobné je to s délkou časového okna. Čím delší, tím vyšší rozlišení pro FFT, ale i vyšší výpočetní náročnost. Míra překrytí segmentů je kritická pro dodržení kritéria konstantnosti OLA (viz 1.3.2).

V praxi se pro minimalizaci artefaktů, především vlivu váhovacího okna, osvědčila délka segmentu 2048 vzorků s 85% překrytím. Při hodnotách menších než 75 % se již projevuje modulace způsobená váhovacím oknem, místo pro optimalizaci je proto vhodné hledat spíše v délce segmentu a z něj plynoucího rozlišení pro FFT. Finální nastavení parametrů proto proběhne až při testu v reálném čase v C++ na platformě JUCE.



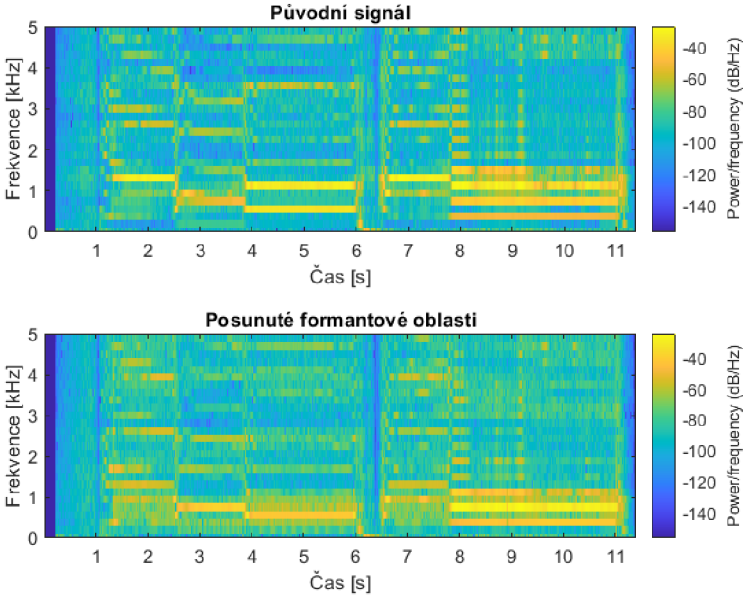
Obr. 3.13: Přeladění výšky tónu pro koeficient 1,4 a překrytí segmentů 10%. V signálu je patrný parazitní jev amplitudové modulace vzniklé nedodržením kritéria konstantnosti.



Obr. 3.14: Přeladění výšky tónu pro koeficient 1,4 a překrytí segmentů 10%. Vliv modulace je menší (výstup byl normalizován)

3.3.2 Přeladďování formantů

Při malých koeficientech přeladění nezanáší algoritmus do výstupního signálu výrazné artefakty.

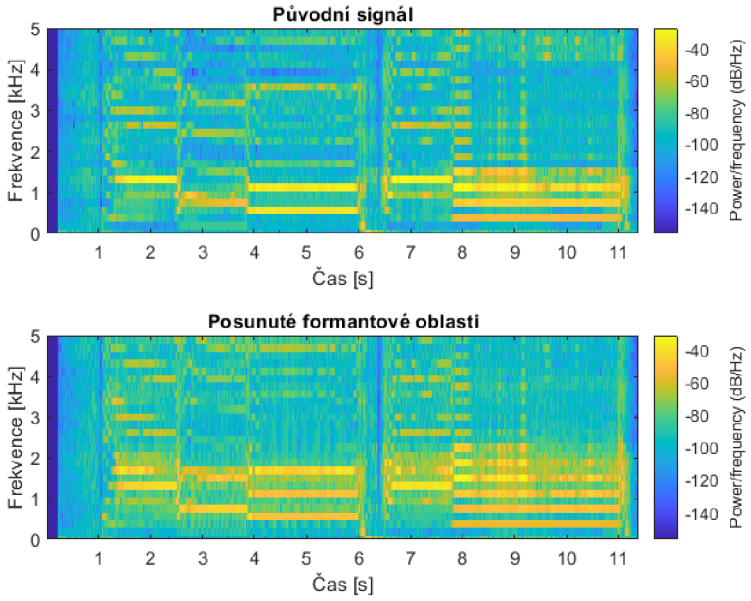


Obr. 3.15: Přeladění formantových oblastí při zachování výšky tónu pro koeficient přeladění 0,7. Formantové oblasti se posunuly směrem dolů a základní kmitočet zůstal zachován.

Posunutí formantů vychází z již zmiňovaného škálování podle pozic subsegmentů ve vstupním segmentu. V kódu se tato operace provádí násobením původních pozic jejich špiček základní periody následovaným syntézou subsegmentů na nové pozice.

```
[T0, locs] = class.pitchDetector.findPitchPeaks(segment, fs
    ,60,900);
newLocs = round(locs * shiftRatio);
```

Po převzorkování se špičky základní periody vrátí na původní pozici. Vyšší harmonické, které byly nedotčeny, se však vlivem převzorkování posunou.



Obr. 3.16: Přeladění formantových oblastí při zachování výšky tónu pro koeficient přeladění 1,4. Formantové oblasti se posunuly směrem nahoru a základní kmitočet zůstal zachován.

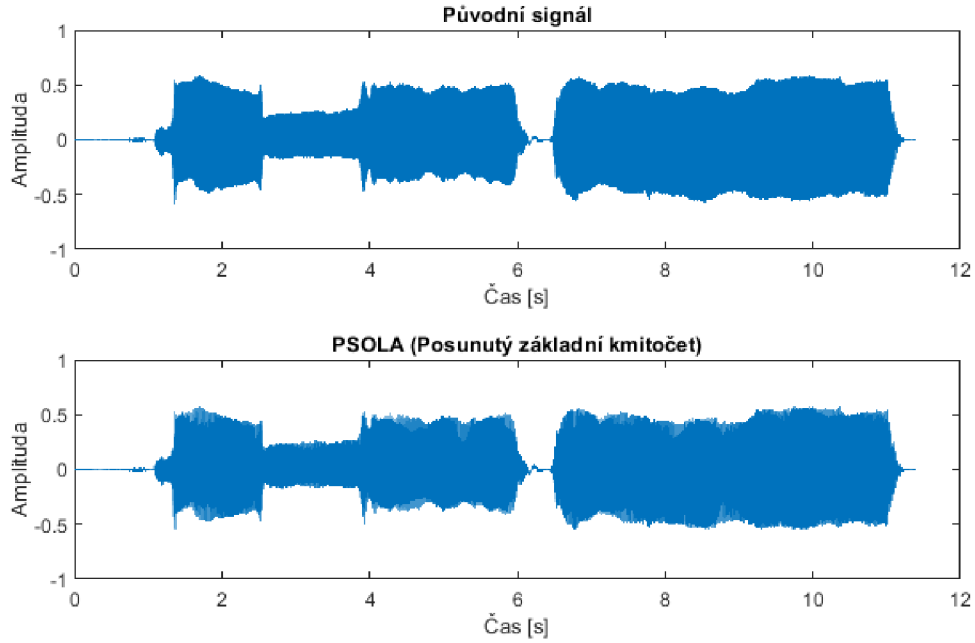
3.3.3 Přelaďování výšky tónu

PSOLA

Jádro algoritmu PSOLA je stejné jako v přeladění formantů. Změna výšky se provádí pomocí časové komprese resp. expanze v číselném poměru následované převzorkováním v poměru opačném. Přístup k syntéze špiček subsegmentů je oproti formantům jiný. Místo posunutí jsou ponechány na původním místě a jsou vyhledány špičky subsegmentů, které budou duplikovány.

```
locsToChange = calculateChangingLocs(class, segment, locs,
    pitchShiftCoef, T0);
```

Vložením těchto kopií je dosaženo nové délky segmentu. Po jeho převzorkování se změní výška tónu a délka se vrátí na původní velikost.
V aplikaci, která je výstupem této práce, byly implementovány dva módy, ve kterých přelaďování výšky funguje (viz kapitola 2.2.2). Jedná se o stejnou implementaci algoritmu PSOLA a mód *auto-tune* pouze počítá poměr přeladění pro každý segment zvlášť. Oba módy však doprovází přítomnost nežádoucích artefaktů, které jsou praveděpodobně způsobeny špatnou návazností segmentů a modulací časové obálky signálu.

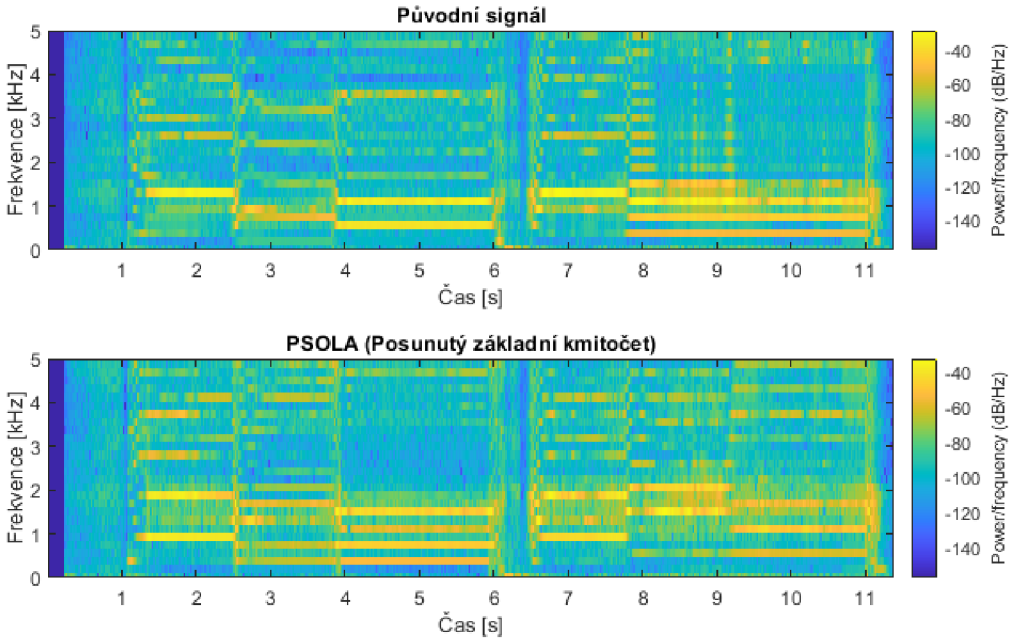


Obr. 3.17: Přeladění výšky tónu algoritmem PSOLA poměrem 1,4. Na obálce jsou patrné změny způsobené špatnou návazností segmentů váhovaných oknem.

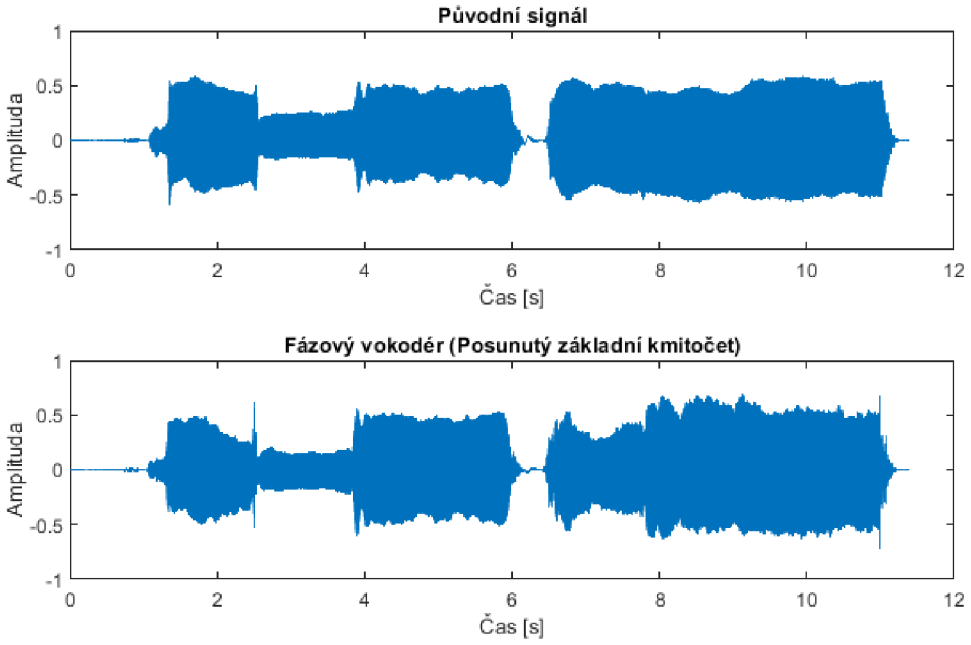
Obrázky 3.17 a 3.18 zachycují vznik parazitních složek ve spektru, které nepříznivě ovlivňují zvukový vjem. Jsou patrné při libovolném nastavení váhovacího okna i překryvu.

Fázový vokodér

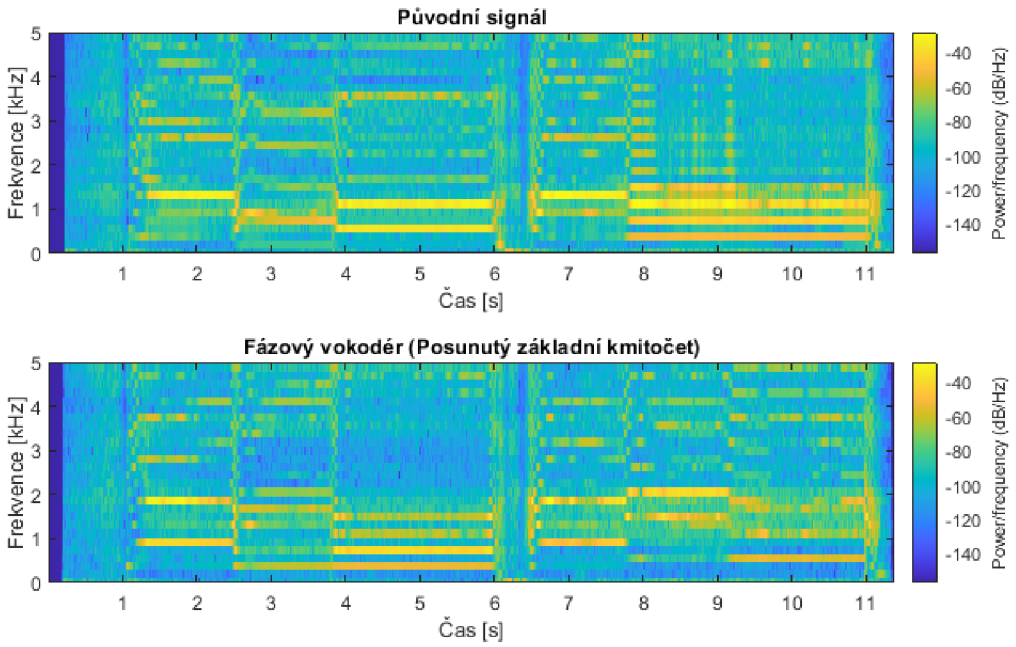
Obrázek 3.19 zachycuje změnu obálky signálu přeladěného fázovým vokodérem a ve spektrogramu 3.20 je patrné přeladění složek signálu. Fázový vokodér poskytuje lepší výsledky, než PSOLA hlavně díky absenci parazitních harmonických složek ve spektru výstupního signálu. Artefakty vytváří též, ale v pásmu vysokých kmitočtů a oproti těm, které způsobuje algoritmus PSOLA jsou zanedbatelné. Při větším koeficientu přeladění je znatelnější i posunutí formantů, které ovlivňují výsledný vjem. V efektu *auto-tune* se ale nijak významně neprojeví, neboť intonační posun není větší než celý tón a koeficient přeladění tak zůstává malý. Proto byl fázový vokodér vybrán jako výsledný algoritmus přeladění pro VST modul.



Obr. 3.18: Přeladění výšky tónu algoritmem PSOLA poměrem 1,4. Ve spektru je patrný vznik parazitních harmonických složek, které odpovídají modulační frekvenci obálky.



Obr. 3.19: Přeladění výšky tónu poměrem 1,4. Změny v obálce jsou méně patrné, než u algoritmu PSOLA.



Obr. 3.20: Přeladění výšky fázovým vokodérem s poměrem 1,4.

4 Implementace v prostředí JUCE

Testovací aplikace implementovaná pro platformu Matlab je postavena na *off-line* zpracování dat. Zvukové soubory jsou nejprve celé načteny do paměti a následně zpracovány. Tento přístup dovoluje provádět komplexnější analýzy a výpočetně náročnější operace. Ve VST modulu, který pracuje v reálném čase, není tento přístup možný, protože kompletní vstupní data nejsou předem známa. Místo toho jsou dávkována po kratších časových oknech, která mohou mít různou délku. Ta se odvíjí od délky vyrovnávací paměti nastavené v ovladači, avšak nemusí její velikost přesně kopírovat. V obecném případě, se kterým by měl modul počítat, mohou mít po sobě jdoucí okna libovolnou délkou. Pro efektivní výpočet DFT je však vhodné volit délku okna v mocninách čísla 2 a udržovat tuto pevnou délku po celou dobu výpočtu. To přináší další implementační výzvy, jejichž řešením se bude tato kapitola zabývat.

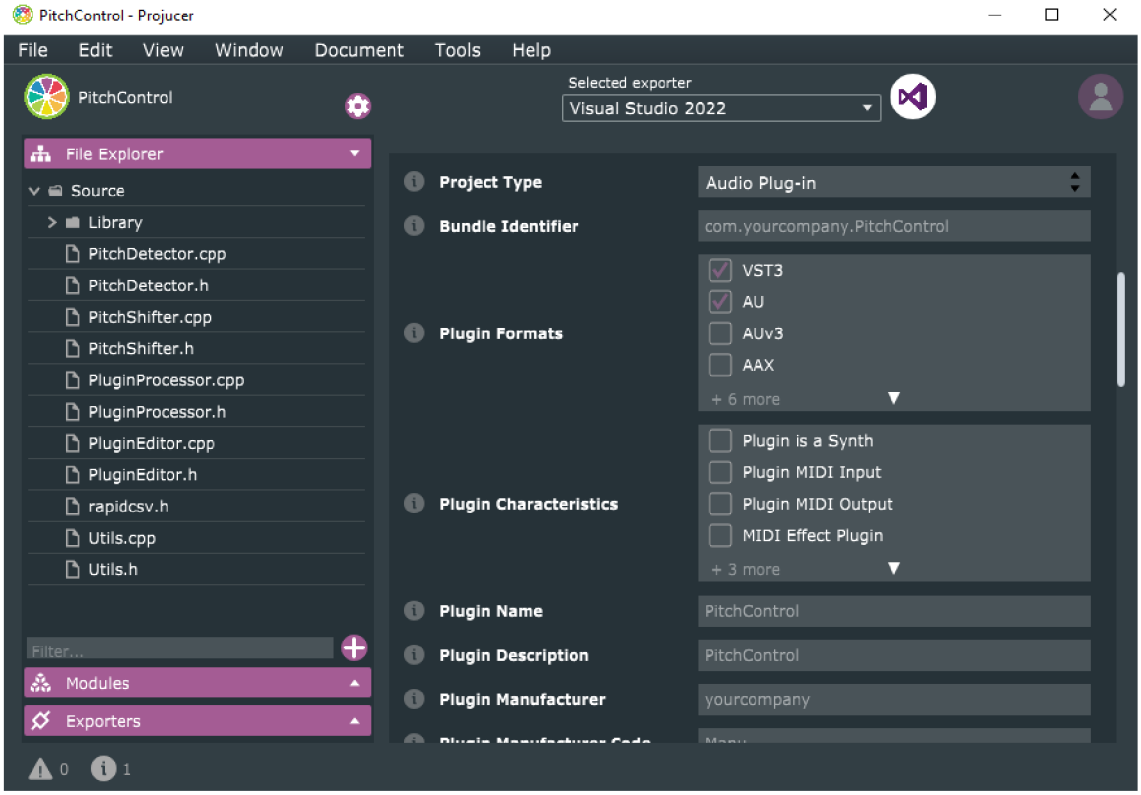
4.1 Vývojové prostředí

Pro výslednou realizaci VST modulu bylo zvoleno vývojové prostředí, tzv. framework, JUCE určený pro více platforem a využívající programovacího jazyka C++, dostupné z https://juce.com/, které umožňuje psát jediný kód, který lze poté zkompilovat a sestavit pro mnoho platforem (Windows, macOS, Linux, Android, iOS). Vývojáře navíc odstiňuje od požadavků rozdílných aplikačních rozhraní (VST3, AU, AAX) a umožňuje mu vyvíjet software pro všechna současně. Disponuje mnoho nástroji a knihovnami pro digitální zpracování zvuku usnadňujícími například použití Fourierovy transformace, vývoj filtrů nebo práci s MIDI. Řeší i komunikaci se zvukovým rozhraním a vývojář se tak může soustředit čistě na implementaci algoritmů pro zpracování signálu.

Ze společného kódu dokáže sestavit VST modul i *standalone* aplikaci a obsahuje mnoho podpůrných nástrojů pro sestavování, ladění a testování projektů. Nejvýznamnějšími z nich jsou Projucer a AudioPluginHost.

4.1.1 Projucer

Hlavní funkcí aplikace Projucer je především management projektů. Spravuje jejich nastavení, atributy pro kompilaci, zdrojové kódy, závislosti na externích knihovnách a mnoho dalších. Generuje projekty pro vývojové prostředí, obsahuje i nástroje k drobným úpravám ve zdrojovém kódu, nicméně pro ladění a vývoj není sám o sobě dostatečný. Zdrojové kódy je vhodné spravovat ve standardním vývojovém prostředí. V této práci bylo použito MS Visual Studio 22.

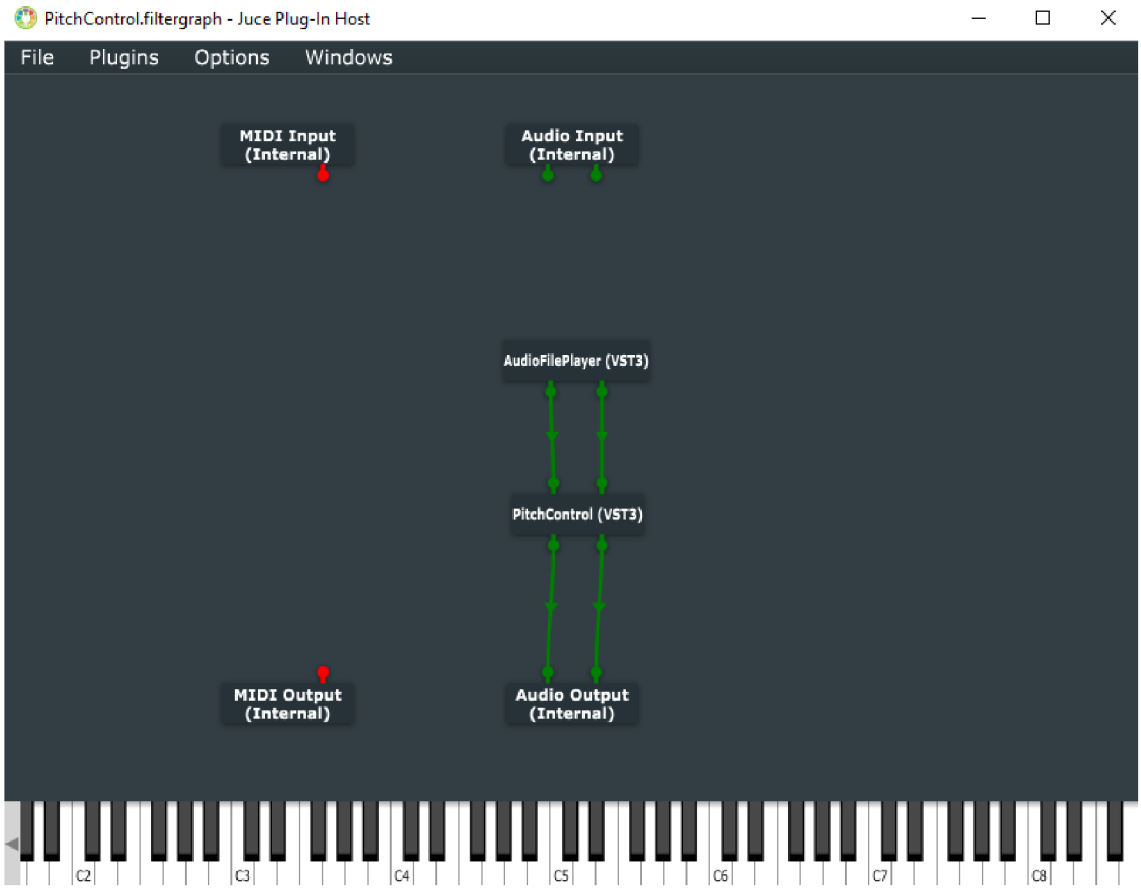


Obr. 4.1: Uživatelské rozhraní aplikace Projucer

4.1.2 AudioPluginHost

Jedná se o podpůrnou aplikaci sloužící k testování, která je stejně jako Projucer součástí JUCE SDK. Poskytuje uživateli uzavřené testovací prostředí se standardními vstupy a výstupy zvukové karty. Kromě nich lze využít i MIDI vstupy a výstupy nebo otevřít VST přehrávač audio souborů. Tímto způsobem lze porovnat výstupy navržených algoritmů v Matlabu s jejich implementací v reálném čase. V prostředí lze otevřít libovolný VST modul, směrovat jej na výstupní porty či vstupy jiného VST a otestovat tak jeho chování.

Moduly lze testovat též ve standardním DAW. AudioPluginHost lze chápat jako jeho odlehčenou náhradu vhodnou pro testování konkrétního VST. Výhodou AudioPluginHostu je možnost nastavit pevnou délku vstupního bloku dat, který bude předán modulu, což může být vhodné pro ladění. Některá DAW tuto možnost nabízet nemusí a plugin obdrží v každém volání ukazatel na vyrovnávací paměť různé délky, pro kterou je systémová hodnota pouze horním limitem.



Obr. 4.2: Uživatelské rozhraní aplikace AudioPluginhost

4.2 Šablony a třídy

Součástí JUCE SDK je několik šablon pro návrh klasických typů programů. Každá obsahuje jiné bázové třídy s různými vzájemnými závislostmi. Součástí jsou šablony vhodné pro *standalone* aplikace (samostatně spustitelné, nepotřebují nadřazený proces), konzolové aplikace či audio pluginy (moduly). Poslední jmenovaná byla použita v této práci a její součástí jsou následující zdrojové a hlavičkové soubory:

PluginEditor obsahující hlavní třídu pro ovládání uživatelského rozhraní. Zde by se mělo odehrávat veškeré vykreslování a zobrazování.

PluginProcessor obsahuje třídu, která je potomkem třídy `juce::AudioProcessor`. Slouží především k manipulaci se vzorky signálu, jejich ukládání do mezipaměti a předávání na výstup. Její součástí jsou metody pro inicializaci pluginu, komunikaci s DAW, uložení aktuálního stavu, nastavení zpoždění a zpracování zvukového signálu.

Ve výsledném VST moduluu, který je výstupem této práce, se analogické třídy jmenují:

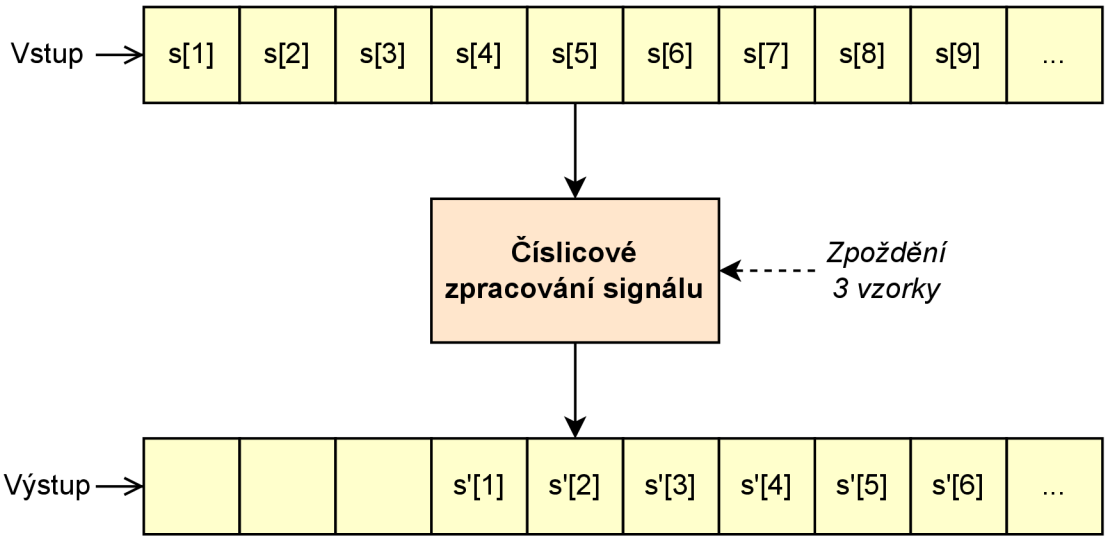
- `PitchControlAudioProcessor`
- `PitchControlAudioProcessorEditor`

Ke čtení vzorků ze vstupu a jejich předávání na výstup slouží jediná metoda `processBlock`. Jejím vstupním argumentem je ukazatel na vyrovnávací paměť se zvukovými vzorky, které byly předány na vstup audio pluginu. Tato paměť může mít různou velikost, která se odvíjí od velikosti vyrovnávací paměti zvukové karty a zároveň je i výstupním argumentem funkce. To znamená, že data v něm uložená, budou po zpracování předána nadřazenému procesu (DAW). V rámci funkce `processBlock` jsou z této vyrovnávací paměti čteny vstupní vzorky, které jsou následně upraveny a na konci funkčního bloku do ní zapsány zpět. Funkce `processBlock` je volána asynchronně podle toho, kdy jsou od hostitelského procesu obdržena vstupní data. Zpracování signálu proto nesmí trvat příliš dlouho a vzorky musí být předány na výstup do té doby, než jsou obdržena nová vstupní data. Jinak by mohlo dojít k zanesení nežádoucích artefaktů do výstupního signálu.

4.2.1 Zpoždění

Vstupně-výstupní vyrovnávací paměť (angl. *buffer*) požaduje v rámci každého volání funkce `processBlocku` bezprostřední obdržení výstupních dat. Fázový vokodér ale využívá k přeladění signálu Fourierovu transformaci o pevné délce bloku, aby bylo použito dostatečné rozlišení a výpočet komplexního spektra mohl probíhat efektivně. Pro bloky kratší, než je rozlišení Fourierovy transformace, tedy nelze zaručit bezprostřední obdržení výstupních dat. Řešením tohoto problému je zavedení další mezipaměti pro zvukové vzorky a zpoždění výstupního signálu. Nadřezený proces (DAW) obdrží informaci o tom, kolik vzorků plug-in modul zpožďuje, k čemuž v aplikačním rozhraní frameworku JUCE slouží metoda `setLatencySamples`. Nadřazený proces si zpoždění všech VST modulů řeší vnitřně sám.

Obrázek 4.3 znázorňuje zavedení zpoždění výstupního signálu. Ve VST modulu pro přeladování výšky, který je výstupem této práce, je potřeba docílit téhož schématu. Výstup nelze obdržet před načtením vzorků počtu délky jednoho časového segmentu Fourierovy transformace, které je nastaveno na délku 1024 vzorků. Díky váhování těchto segmentů váhovacím oken a vzájemnému překrytí segmentů lze po provedení zpětné transformace předat na výstup pouze počet vzorků odpovídající délce jednoho skoku. Ta byla nastavena empiricky na 128 vzorků na základě testování algoritmů v prostředí Matlab. Tyto hodnoty se v praxi ukázaly být nejlepším balancem mezi časovou složitostí a výskytem nežádoucích artefaktů ve výstupním signálu. Celková hodnota zpoždění výstupu je rovna 1024 vzorků.



Obr. 4.3: Zpoždění výstupního signálu

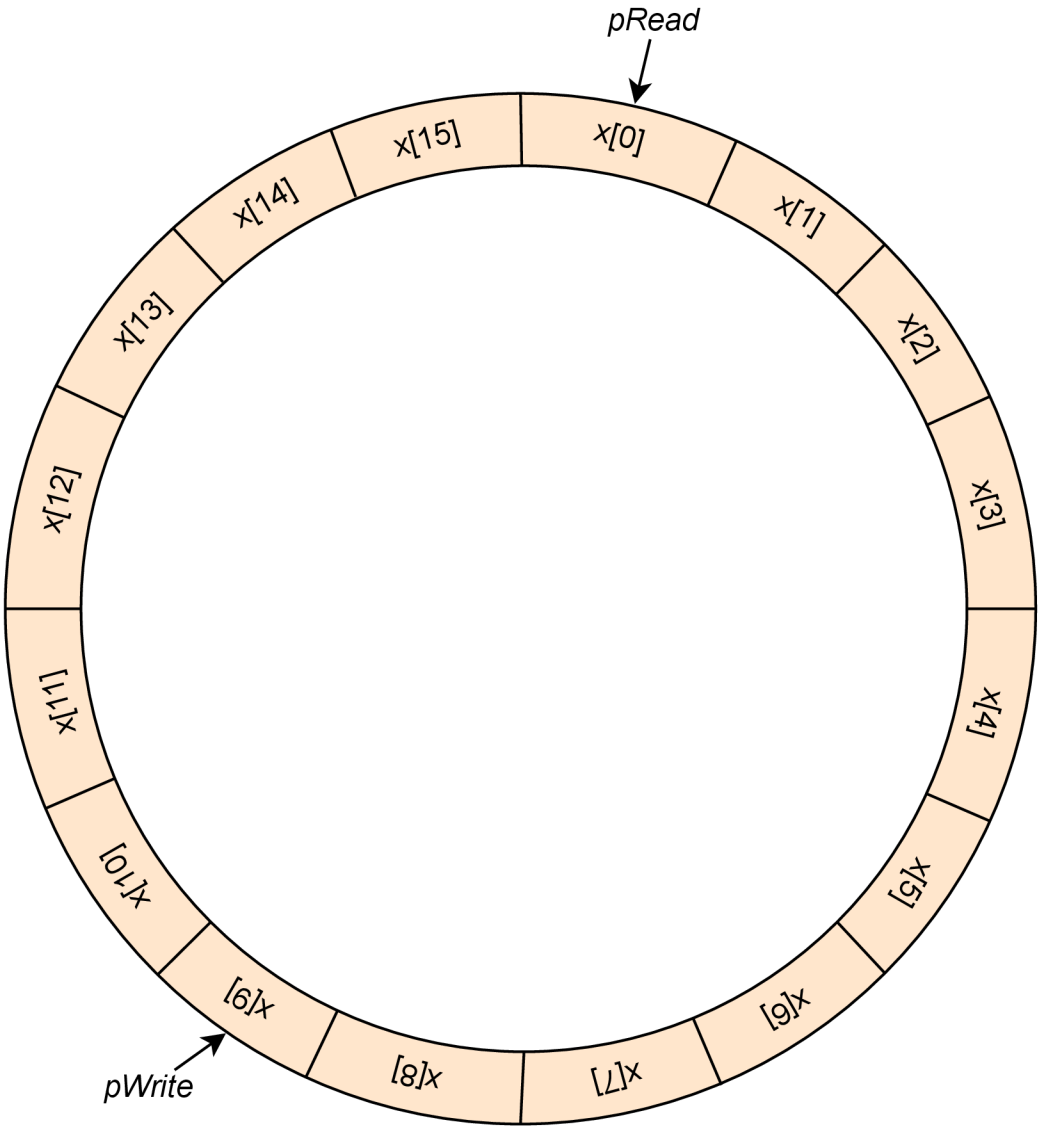
4.2.2 Kruhová vyrovnávací paměť

Vstupní vzorky mohou od nadřazeného procesu přicházet po různě velkých úsecích. Úkolem mezipaměti je ukládat tyto vzorky, dokud nebude k dispozici jejich dostatečný počet (1024) pro provedení Fourierovy transformace.

K realizaci vyrovnávací paměti je nejvhodnější využít kruhovou strukturu (viz obr. 4.4). Ta obsahuje ukazatele pro čtení (*pRead*) a pro zápis (*pWrite*). Oba se nejprve nachází v počátku. Po obdržení N vstupních vzorků dojde k jejich zápisu od místa ukazatele pro zápis, který se následně posouvá o N vzorků v paměti vpřed. Ukazatel pro čtení setrvává v původní pozici do doby čtení a poté se též posouvá vpřed. Pokud by mělo nastat přetečení, pokračuje se zápisem vzorků od začátku. Pokud bylo do paměti často zapisováno, ale málo vzorků přečteno, může v obecném případě dojít k tomu, že ukazatel pro zápis po přetečení předběhne ukazatel pro čtení. Proto je vhodné volit dostatečnou velikost mezipaměti. Ve výsledné aplikaci byly využity dvě kruhové mezipaměti o délce 8192 vzorků, což odpovídá osmi segmentům Fourierovy transformace, jeden pro načítání vstupního signálu a druhý pro výstupní signál.

4.2.3 Podpůrné třídy

Realizace tříd v C++ vychází z návrhu vytvořeném v prostředí Matlab (viz sekce 2.1.3). Rozdílem zůstává pouze přístup ke zpracování dat, který zde probíhá v reálném čase. Následující výčet shrnuje hlavní třídy a jejich účel v rámci pluginu.



Obr. 4.4: Realizace kruhové vyrovnávací mezipaměti o délce 16 vzorků

PitchControlAudioProcessor je jádrem pluginu. Stará se o práci s mezipamětí, což zahrnuje ukládání vstupních a výstupních vzorků, segmentaci, aplikaci váhovacího okna a čtení parametrů uživatelského rozhraní. Implementuje metodu `processBlock`, která kromě výše zmíněného obsahuje i logiku pro výběr operačního módu pluginu. Těmi jsou *pitch shifting* a *auto-tune*.

PitchShifter sdružuje metody týkající se přeladění výšky zvukového signálu. Implementuje algoritmus fázového vokodéru, udržuje informaci o aktuální zvolené tónině a hodnoty fází předchozího zpracovaného segmentu pro každý kanál.

PitchDetector slouží k detekci výšky tónu, ke které používá metodu rychlé autokorelace. Dále poskytuje metody pro nalezení nejbližšího tónu podle detekované základní frekvence ve dvou módech. Prvním je nalezení nejbližšího chromatického půltónu podle temperovaného ladění. Druhý mód umožňuje nalézt nejbližší tón ze zvolené tóniny.

Utils obsahuje podpůrné metody a konstanty využívané napříč všemi třídami, jako například zápis do souboru, vlastní výpočet funkce modulo či konstantu π.

K výpočtům Fourierovy transformace byla použita implementace z modulu `juce::dsp`, který je součástí frameworku JUCE. V rámci celého pluginu je pomocí ukazatelů sdílena jedna instance třídy `juce::dsp::FFT`, aby bylo omezeno zbytečné kopírování a vytváření nových instancí.

4.3 Uživatelské rozhraní

Výsledný plugin byl pojmenován jako PitchControl. Jeho součástí je i jednoduché uživatelské rozhraní (viz obr. 4.5), které vychází ze standardního ovládacího panelu *pitch-shifteru* s následujícími ovládacími prvky.

Reference Pitch nastavuje výškový normál. Výchozí hodnota 440 Hz odpovídá standardnímu ladění ISO [21, online]. Aplikuje se v režimu *Auto-Tune*

Coarse slouží ke skokovým změnám výšky v rámci půltónů s rozsahem -12 až 12. Záporné hodnoty značí podladění směrem dolů, kladné nadladění směrem nahoru.

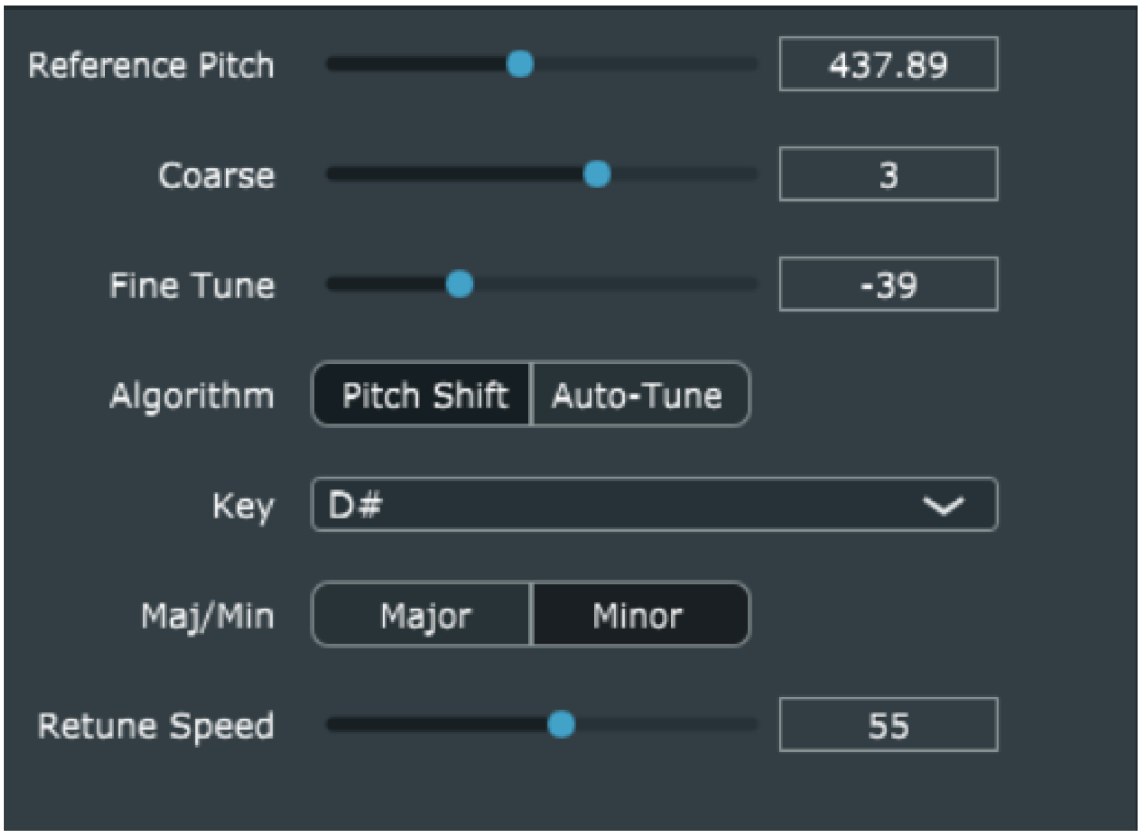
Fine Tune slouží k jemnému doladění v *centech* s rozsahem -100 až 100.

Algorithm přepíná mezi operačními módy pluginu. *Pitch Shift* provádí pevné přeladění vstupního signálu a k jeho ovládání slouží ovládací prvky *Coarse* a *Fine Tune*. *Auto-Tune* využívá i zbývající z nich.

Key nastavuje tóninu, ve které se hledá nejbližší tón a ve které by se měl pohybovat i zvukový signál. K dispozici jsou standardní tóniny podle temperovaného ladění a chromatická stupnice se všemi dvanácti půltóny.

Maj/Min mění typ tóniny z molové na durovou.

Efekt přeladění pomocí Coarse a Fine Tune se v režimu *Auto-Tune* sčítá. Nejprve je vyhlazena intonace podle nastavené tóniny a následně je signál přeladěn podle hodnot těchto parametrů. K vykreslení uživatelského rozhraní byl z časových důvodů použit `juce::GenericAudioProcessorEditor`, který vytváří jednoduché GUI a pro každou hodnotu parametru příslušný ovládací prvek. Výsledný VST modul dokáže v režimu *Auto-Tune* zastoupit režim *Pitch Shift* při nastavení parametru *Retune Speed* na nulu. Stále je však plýtván výpočetní výkon na detekci základního kmitočtu, kterou režim *Pitch Shift* obchází, a proto byl ponechán jako separátní mód.



Obr. 4.5: Uživatelské rozhraní pluginu PitchControl

4.4 Omezení VST

Výsledný VST modul dosahuje při přeladění mono signálů stejných výsledků jako testovací aplikace vyvinutá na platformě Matlab. Narozdíl od ní však umožňuje zpracovávat více kanálů, což může působit problém v režimu autotune, který může v každém kanálu detekovat základní kmitočet s nepatrným rozdílem. Při přeladění stereo signálu se proto v tomto režimu detekuje základní perioda pouze v levém kanálu a stejný koeficient přeladění se použije i pro kanál pravý.

Závěr

Tato práce zkoumá, implementuje a porovnává metody přelaďování zvukového signálu a detekce základního kmitočtu, definuje navazující problémy a navrhuje jejich řešení. Jejím výstupem je testovací aplikace pro platformu Matlab, která v režimu *off-line* zpracování signálu demonstruje jednotlivé metody a zásuvný modul technologie VST, který na základě vybraných metod provádí přelaďování výšky tónu v reálném čase.

V testovací aplikaci platformy Matlab byly implementovány a otestovány dvě metody pro přeladění výšky tónu založené na algoritmech fázový vokodér a PSOLA. Aplikace dále obsahuje 5 metod detekce základního kmitočtu a 3 metody určení znělosti segmentu. Všechny byly podrobeny testování a do zásuvného modulu technologie VST byly vybrány fázový vokodér díky menšímu výskytu artefaktů ve výstupním signálu, rychlá autokorelace díky přesnosti a rychlosti zpracování vstupních dat a metoda neuronu kvůli nejlepší rozlišovací schopnosti znělých částí signálu.

Pro implementaci zásuvného modulu technologie VST byl vybrán framework JUCE. Modul může být nasazen na několika sběrnicích v libovolném DAW, které VST podporuje. Nabízí operační módy *pitch-shifter*, který přelaďuje výšku zvukového signálu v rozsahu dvou oktáv s možností jemného doladění a *auto-tune*, který na základě zvolené tóniny provádí korekce intonace s možností kombinace obou módů.

Z výsledků, které v této práci poskytuje algoritmus PSOLA, neplyne nevhodnost tohoto algoritmu pro použití k přelaďování výšky tónu, ale spíše nutnost velké preciznosti při jeho implementaci. Kvalita přeladění zde závisí na detekci špiček základní periody, jejíž nedostatky se propisují i do kvality přelaďování. PSOLA navíc provádí dvojitou segmentaci, kde vzniká prostor pro řadu implementačních chyb. Výhodou fázového vokodéru je, že k samotnému přeladění nepotřebuje znát průběh základní periody (kmitočtu) a využívá ji pouze v režimu *auto-tune*. Určení znělosti při segmentaci se v praxi ukázalo jako méně významné, neboť neznělé segmenty lze z velké části odfiltrovat nastavením prahových hodnot autokorelační metody. Kromě toho zvolený postup pro normalizaci segmentu pomocí maximální špičkové hodnoty není v praxi účinný, protože využitím maxima krátkodobého segmentu se normalizuje i hodnota krátkodobé energie šumu a neznělých částí, což má za následek nepřesnost při výpočtu ZCR. Vhodnější by bylo měřit průměrné RMS v čase a normalizovat segmenty podle něj. Kvůli tomu byla metoda neuronu v modulu vyřazena. Její implementace však byla ponechána pro možnost budoucího přezkoumání.

Výsledný VST modul se i tak svými možnostmi přibližuje komerčním pluginům pro přelaďování a korekci intonace. Tam, kde zaostává, vzniká prostor pro další vylepšení. Tím může být především robustnější detekce základního kmitočtu a lepší uživatelské rozhraní, které by mohlo například vykreslovat průběh intonační křivky.

Literatura

[1] SMÉKAL, Zdeněk. *Analýza signálů a soustav: Signály a jejich matematické modely*. Brno: Vysoké učení technické v Brně, 2019.

[2] SMÉKAL, Zdeněk. *Analýza signálů a soustav - BASS*. 1. Brno: Vysoké učení technické v Brně, 2012. ISBN 978-80-214-4453-9.

[3] SCHIMMEL, Jiří. *Studiová technika: Digitální zvukové signály* [online]. Brno: Vysoké učení technické v Brně, 53 s. [cit. 2022-12-01].

[4] D. W. Barker, *Efficient Resampling Implementations [DSP Tips & Tricks]*,"in IEEE Signal Processing Magazine, vol. 25, no. 4, pp. 114-117, July 2008, doi: 10.1109/MSP.2008.923508.

[5] MIŠUREC, Jiří, SMÉKAL,Z. *Číslicové zpracování signálů*. Brno: Vysoké učení technické v Brně, 2011.

[6] VRBÍK, Matouš. *Modelování lineárního zkreslení zvukových zařízení* [online]. Brno, 2020 [cit. 2022-12-03]. Dostupné z: http://hdl.handle.net/11012/189402. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Jiří Schimmel.

[7] SMITH, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing*. 1. California: California Technical Publishing, 1997. ISBN 978-0-9660176-3-2.

[8] ONDERKA, Jan. Pitch Shifting of Audio Signals in Real Time Using STFT on a Digital Signal Processor. Praha, 2018. Dostupné také z: https://dspace.cvut.cz/handle/10467/77279. Bakalářská práce. České vysoké učení technické v Praze.

[9] SCHIMMEL, Jiří. *Studiová technika: Zvukové efekty* [online]. Brno: Vysoké učení technické v Brně, 53 s. [cit. 2022-12-01].

[10] ZÖLZER, Udo, ed. *DAFX - Digital Audio Effects*. 2. Hoboken, New Jersey, USA: Wiley, 2011. ISBN 978-0-470-66599-2.

[11] MOUSA, Allam. *Voice Conversion using Pitch Shifting Algorithm with PSOLA and Re-Sampling*. Journal of Electrical Engineering. 2011, 2011(61). Dostupné z: doi:10.2478/v10187-010-0008-5

[12] SCHIMMEL, Jiří. *Studiová a hudební elektronika* [online]. 2. Brno: Vysoké učení technické v Brně, 2015. ISBN 978-80-214-4452-2.

[13] GDEISAT, Munther a Francis LILLEY. One-dimensional phase unwrapping problem. Signal. 2011, (4), 6.

[14] BERNSEE, Stephan. Pitch Shifting Using the Fourier Transform. Stephan Bernsee's Blog [online]. 21.9.1999 [cit. 2023-05-13]. Dostupné z: http://blogs.zynaptiq.com/bernsee/pitch-shifting-using-the-ft/

[15] ATASSI, Hicham. *Metody detekce základního tónu řeči.* Elektrorevue [online]. 2008, 21.1.2008, 2008(4), 1-17 [cit. 2022-12-04]. ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/70/metody-detekce-zakladniho-tonu-reci/

[16] APICELLA, Barbara, Annalisa BRUNO, Xuan WANG a Nicola SPINELLI. Fast Fourier Transform and autocorrelation function for the analysis of complex mass spectra. International Journal of Mass Spectrometry. 2013, 2013(338), 30-38. ISSN 1387-3806. Dostupné z: doi:https://doi.org/10.1016/j.ijms.2013.01.003

[17] SMÉKAL, Zdeněk. *Zpracování řeči.* Brno: Vysoké učení technické v Brně, 2012. s. 1-171. ISBN: 978-80-214-4896- 4.

[18] PELOUŠEK, Tomáš. *Simulace zkreslení zvukového signálu v percepčním zvukovém kodéru* [online]. Brno, 2021 [cit. 2022-12-01]. Dostupné z: http://hdl.handle.net/11012/197098. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací.

[19] ZENKL, Luděk. *ABC hudební nauky.* 1. Praha: Editio Bärenreiter, 2003. ISBN 80-86385-21-3.

[20] SYROVÝ, Václav.*Hudební akustika.* 3. Praha: Akademie múzických umění v Praze, 2013. ISBN 978-80-7331-297-8.

[21] Acoustics — Standard tuning frequency (Standard musical pitch) [online]. 1975, 1 [cit. 2023-05-24]. Dostupné z: https://www.iso.org/standard/3601.html

Seznam symbolů a zkratek

DFT	diskrétní Fourierova transformace – Discrete Fourier Transform
DTFT	Fourierova transformace časově diskrétních signálů – Discrete Time Fourier Transform
FFT	rychlá Fourierova transformace – Fast Fourier Transform
IFFT	inverzní rychlá Fourierova transformace – Inverse Fast Fourier Transform
STFT	krátkodobá Fourierova transformace – Short Time Fourier Transform
OLA	Overlap and Add
SOLA	Synchronous Overlap and Add
PSOLA	Pitch Synchronous Overlap and Add
TD-PSOLA	Time-Domain Pitch Synchronous Overlap and Add
FD-PSOLA	Frequency-Domain Pitch Synchronous Overlap and Add
SFM	Spektrální plochost – Spectral Flattness Measure
f_{vz}	vzorkovací frekvence
T_0	základní perioda
f_0	základní frekvence
DAW	Digital Audio Workstation
VST	Virtual Studio Technology
UI	uživatelské rozhraní – User Interface
IDE	Integrované vývojové prostředí – Integrated Development Environment
SDK	Software Development Kit
RMS	Efektivní hodnota – Root Mean Square
ISO	International Organization for Standardization
GUI	Grafické uživatelské rozhraní – Graphical User Interface

Seznam příloh

A Řešení v prostředí Matlab

Testovací projekt realizovaný v prostředí Matlab App Designer a jeho zdrojové soubory vznikaly ve verzi R2021b. V příloze se nachází zdrojové kódy a pomocné skripty. Součástí zdrojových kódů testovací aplikace je i popis všech klíčových atributů a metod doplněný komentáři. Testovací aplikace se spouští z hlavního skriptu *PitchCorrectionTool.m*. Skripty, které se netýkají přímo testovací aplikace, ale byly použity například pro generování grafů či porovnávání výstupů, jsou umístěny ve složce *supportScripts*.

```
/ ..... kořenový adresář
└── Matlab ..... testovací aplikace pro platformu Matlab
    ├── supportScripts ..... pomocné skripty pro generování grafů
    │   ├── parameterComparison.m
    │   ├── pitchShiftingCompare.m
    │   ├── pitchDetectionCompare.m
    │   ├── spectrogramRetune.m
    │   └── voicedEstimationPlot.m
    ├── play_icon.png
    ├── PitchCorrectionTool.m ..... hlavní spouštěcí skript aplikace
    ├── PitchCorrectionTool.mlapp ..... projekt utility App Designer
    ├── PitchDetector.m ..... třída pro detekci základního tónu
    ├── PitchShifter.m ..... třída pro přelaďování signálu
    ├── PitchUtils.m ..... třída pro detekci znělosti a další podpůrné metody
    └── phaseWrap.m ..... funkce pro zabalení fáze
```

B Řešení v prostředí JUCE

Zásuvný modul VST byl realizován ve frameworku JUCE v jeho vývojovém prostředí Projucer verze 7.0.1. Vývoj a správa zdrojových kódů probíhala v Microsoft Visual Studio 2022. Jeho projekt je vhodné spouštět přes Projucer, protože nastavení ve VS provedená mimo Projucer se při dalším otevření přepíší. Při kompilaci je nutné nastavit konfiguraci *release*, neboť dojde ke značným optimalizacím při kompilaci a významnému snížení výpočetní náročnosti modulu. Zkompilovaný VST modul se nachází ve složce `JUCE\Builds\VisualStudio2022\x64\Release\VST3` a je potřeba jej zkopírovat do složky `C:\Program Files\Common Files\VST3`, což je standardní umístění pro VST na platformě Windows.

Součástí příloh jsou kromě zdrojových souborů i testovací template pro AudioPluginHost, který byl použit při testování aplikace a zkompilovaný VST plugin ve složce *Builds*. Pro instalaci VST modulu na platformě Windows je potřeba jej zkopírovat do výše zmíněné složky. Testování modulu probíhalo v FL Studio 12.

- `/` ..... kořenový adresář
  - `JUCE` ..... JUCE projekt
    - `JuceLibraryCode` ..... zdrojové kódy frameworku JUCE
    - `Source` ..... zdrojové soubory pluginu
      - `PluginProcessor.h`
      - `PluginProcessor.cpp` ..... výpočetní jádro pluginu
      - `PluginEditor.h`
      - `PluginEditor.cpp` ..... uživatelské rozhraní
      - `Library`
        - `PitchDetector.h`
        - `PitchDetector.cpp` ..... detekce základního kmitočtu
        - `PitchShifter.h`
        - `PitchShifter.cpp` ..... přelaďování výšky
        - `Utils.h`
        - `Utils.cpp` ..... podpůrné funkce
    - `PitchControl.jucer` ..... hlavní spouštěcí soubor
    - `PitchControl.filtergraph` ..... testovací template pro AudioPluginHost
    - `Builds`
      - `VisualStudio2022` ..... vygenerovaný projekt pro Visual Studio
      - `PitchControl.vst3` ..... výsledný VST plugin

C Testovací soubory

K testování algoritmů v Matlabu a C++ byly použity následující zvukové vzorky.

- `/` — kořenový adresář
  - `PitchCorrectionTool` — testovací aplikace pro platformu Matlab
    - `audio` — testovací zvukové soubory
      - `other` — ostatní zvukové nahrávky
        - `Guitar_F#min.wav`
      - `sinewaves` — harmonické signály
        - `a1-440(long).wav`
        - `a1-440(short).wav`
        - `a-220.wav`
        - `e1-330.wav`
      - `strings` — strunné nástroje
        - `Violin_Cm.wav`
        - `Violin_F#m.wav`
        - `Violin_Fm.wav`
      - `vocals` — řeč a zpěv
        - `backingVocal(outtake).wav`
        - `WhatsernameHarmonicTest.wav`
        - `WhatsernamePt1.wav`