VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

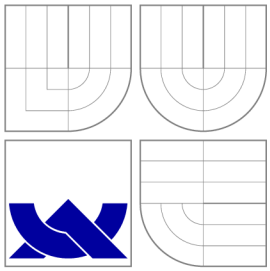
ZMĚNA RYCHLOSTI (SYNCHRONIZACE) PŘEHRÁVÁNÍ VIDEA V ZÁVISLOSTI NA RYCHLOSTI ŘEČI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

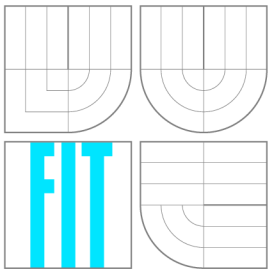
AUTOR PRÁCE
AUTHOR

MICHAL HROMÁDKO

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZMĚNA RYCHLOSTI (SYNCHRONIZACE) PŘEHRÁVÁNÍ VIDEA V ZÁVISLOSTI NA RYCHLOSTI ŘEČI

PLAYING OF VIDEO DEPENDING ON SPEED OF SPEECH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL HROMÁDKO
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZÖKE
SUPERVISOR

BRNO 2008

Licenční smlouva je uvedena v archivním výtisku uloženém v knihovně FIT VUT v Brně.

Abstrakt

Tato bakalářská práce se zabývá rozšířením přehrávače VLC o metodu PSOLA. Tato metoda umožňuje měnit rychlost přehrávání videa při zachování základního tónu a srozumitelnosti řeči.

Klíčová slova

VLC, modul, PSOLA, řeč, rychlost

Abstract

This bachelor's thesis discusses adding the PSOLA method into the VLC Media Player. PSOLA method is used for playing rate modification. It doesn't change base tone and understandableness of the speech.

Keywords

VLC, plugin, PSOLA, speech, speed, rate

Citace

Michal Hromádko: Změna rychlosti (synchronizace) přehrávání videa v závislosti na rychlosti řeči, bakalářská práce, Brno, FIT VUT v Brně, 2008

Změna rychlosti (synchronizace) přehrávání videa v závislosti na rychlosti řeči

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Hromádko
14. května 2008

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Igoru Szökemu, za vedení, ochotu při řešení problému, canné rady a připomínky při zpracování a řešení bakalářské práce.

Obsah

Kapitola 1

Úvod

Přehrávání videa je jednou z nejrozšířenějších a nejoblíbenějších činností, jaké lze na počítači provozovat. Ze začátku šlo jen o zábavu několika nadšenců, kteří s úžasem v očích sledovali několik měnících-se pixelů na obrazovce. Vývoj se však nedá zastavit a brzy se na monitory počítačů dostala i videa již sledovatelné kvality. Bohužel to mělo za důsledek velký rozvoj počítačového pirátství. Objevili se pokoutní prodejci vypálených CD s různými filmy. Dalším milníkem byla určitě možnost přenášet (streamovat) video přes Internet. V dnešní době není problém se na počítači podívat na film ve vysokém rozlišení, nebo živě sledovat přenos z konference na druhé straně planety.

Díky těmto technickým vymoženostem se objevila další úžasná možnost - sledovat živě přenos z přednášky pěkně v teple domova (koleje). Sledování přednášky na dálku samozřejmě nenahradí její návštěvu, ale to studentům očividně nevadí a účast na přednáškách ze pomalu začala zmenšovat.

Po tomto kroku chyběla k dokonalosti pouze jedna věc. Záznamy přednášek. Ty na sebe nenechaly dlouho čekat. Ke stažení je k dispozici několik stovek přednášek z různých kurzů. Pro studenty je to skvělá možnost jak si před zkouškou zopakovat probranou látku. Má to jeden drobný nedostatek. Přednášky jsou dlouhé a přednášející občas mluví pomalu nebo vůbec. Každý jistě zkusil přehrávání přednášky zrychlit a zjistil, že se dostavil velmi nepříjemný efekt. Přednášející se změnil v „nesrozumitelně drmolícího kačera Donalda“. Kromně toho, že je to celkem vtipné, se však žádaný efekt nedostavil. Video je nutné opět zpomalit a sledovat v normální rychlosti.

Cílem této bakalářské práce je upravit některý z existujících přehrávačů videa tak, aby bylo možné dynamicky měnit rychlost přehrávání videa se zachováním srozumitelnosti řeči. Vycházím v ní z diplomové práce Ing. Aleše Kováříka s názvem Změna rychlosti řeči [7]. Jeho diplomová práce se zabývá právě zrychlováním a zpomalováním řeči při zachování srozumitelnosti. Pro změnu rychlosti používá metodu PSOLA[1] s rozšířením o fonémový rozpoznávač.

1.1 Průvodce prací

Ve 2. kapitole nazvané Přehrávače videa se zabývám výběrem vhodného přehrávače. Popisuji zde kritéria pro výběr a několik zkoumaných přehrávačů.

Kapitola číslo 3 popisuje mnou vybraný přehrávač VLC. Od základního popisu postupně přecházím k vlastnostem, vnitřnímu uspořádání a tvorbě modulů. Při tvorbě této kapitoly

[1] Pitch Synchronous OverLap Add

jsem vycházel částečně z dokumentace VLC [2], ale především ze studia zdrojových kódů.

V kapitole nazvané Metoda PSOLA a fonémový rozpoznávač(4) vycházím z diplomové práce Ing. Kováříka [7] a z knihy pana Tieriho Dutoida [6]. Jak její název napovídá, zabývá se popisem metody PSOLA, fonémového rozpoznávače a jejich vzájemného propojení.

Poslední kapitola (5) se zabývá vlastní úpravou přehrávače VLC. Popisuji zde začlenění upravené metody PSOLA do VLC a problémy se kterými jsem se během implementace setkal.

Kapitola 2

Přehrávače videa

Prvním krokem při tvorbě této bakalářské práce bylo najít vhodný přehrávač videa. Hledaný přehrávač musí být pokud možno multiplatformní a otevřený. Otevřeností myslím šíření přehrávače pod jednou z open-source licencí. Nejrozšířenější z těchto licencí je GP - GNU Generel Lublic Licence [3]. Tato licence umožňuje volnou editaci zdrojových kódů programu. Dalším hlediskem byla rozšířenost přehrávače. Lidé většinou nemají rádi věci, které neznají. Důležitý je také celkový dojem z přehrávače. Bral jsem v potaz funkčnost, možnost rozšíření o přídavné moduly a v neposlední ředě také vzhled. Mnozí mohou namítat, že na vzhledu nezáleží, ale z vlastní zkušenosti vím, že vzhled je důležitým faktorem.

Při bližším zkoumání přehrávačů jsem se zaměřil na další jejich vlastnosti. Důležitou vlastností je například počet různých multimediálních formátů, které je přehrávač schopen přehrát. Dříve bylo nutné mít několik různých přehrávačů v závislosti na typu přehrávaného materiálu. V dnešní době většinou postačí jeden kvalitní přehrávač a sada kodeků (chodec pacek). Proto jsem se zaměřil spíše na možnost přehrávání různých multimediálních streamů vysílaných jak po lokálních sítích, tak po Internetu. Největší váhu jsem přiřadil protokolu RESP [1] s jehož pomocí je možné sledovat záznamy přednášek z naší fakulty. Dalším důležitým faktorem byla možnost sledovat přednášky přímo živě. To sice nemá na mou práci vliv, protože živý přenos nejde zrychlit, ale proč mít více přehrávačů když stačí jeden.

U přehrávačů jsem zkoumal také možnost záznamu přehrávaného streamu. Mnoho studentů jistě využije možnost zaznamenat si sledovanou přednášku. Záznamy je samozřejmě možné stahovat přímo z video serverů FIT, ale u některých přednášek tato možnost není, nebo jsou zveřejněny až po delším čase. Zajímavá je také možnost naplánovat si nahrávání dopředu. Osobně velmi často využívám takzvaný time shifting. Jde o možnost pozastavit právě sledovaný živý přenos a později pokračovat ve sledování.

Nejdůležitější zkoumanou vlastností byla snadná rozšiřitelnost přehrávače. Sudoval jsem možnost jednoduše začlenit metodu PSOLA pro změnu rychlosti zvuku. To vyžaduje modularitu přehrávače a také kvalitní dokumentaci.

2.1 Media Player Classic

Tento přehrávač se vyznačuje svou jednoduchostí. Jak je patrné z jeho jména, vzhledem vychází z přehrávače Windows Media Player verze 6. Zvládá přehrávat většinu známých formátů videa a zvuku. Bohužel postrádá jakoukoliv možnost přehrávat streamované video.

[1] Regal Štime Streaming Protocol

Media Player Classic je vyvíjen v rámci projektu Guliverkli pod licencí GPL. Je distribuován s většinou sad kodeků. Díky tomu je poměrně dobře rozšířen. Bohužel je závislý na platformě Microsoft Windows.

2.2 Miro

Miro je poměrně nový přehrávač. V mnoha směrech je revoluční. Podle vývojářů [4] mění počítač v internetovou televizi. V podstatě jde spíše o stahovač videa s možností jeho přehrávání. Umožňuje stahovat videa z mnoha internetových serverů. Z těch nejznámějších uvedu například YouTube.

Podporuje přehrávání mnoha formátů videa. Všechna tato videa však musí být stažena v počítači. Neobjevil jsem možnost jak přehrát multimediální stream. Z hlediska uživatelského rozhraní je to velmi povedený program. Kombinuje jednoduchost s moderními grafickými prvky. Je platformově nezávislý a je šířen pod licencí GPL.

2.3 MPlayer

MPlayer je jedním z nejrozšířenějších open-source přehrávačů. Jde spustit téměř na všech známých architekturách. Je to výborný přehrávač, který zvládá přehrát doslova cokoliv. Pokud je to možné, nabízí pěkné a jednoduché uživatelské rozhraní. Toto rozhraní je možné měnit pomocí motivů vzhledu (skinů). Sám jsem tento přehrávač nějakou dobu používal a byl jsem s ním naprosto spokojen. Jediná věc, která mi v tomto přehrávači chybí je možnost zaznamenávání streamovaného videa na disk.

2.4 xine

Xine je sama o sobě knihovna pro přehrávání multimédií. Podporuje většinu známých formátů. Vývojový tým k této knihovně vytvořil i celkem pěkné uživatelské rozhraní xine-ui. Několik známých multimediálních přehrávačů, například Totem, je postaveno právě na xine.

2.5 GStreamer

GStreamer je knihovna pro práci s multimédii. Kromě přehrávání podporuje i editaci. Obsahuje mnoho nástrojů pro vývojáře multimediálního software. Samotná knihovna je snadno rozšiřitelná a kdokoliv může přispět novým modulem či kodekem. Na rozdíl od všech předchozích přehrávačů je tato knihovna šířena pod licencí LGPL, což umožňuje její nasazení i v komerčních přehrávačích. Mezi nejznámější přehrávače založené na této knihovně patří například Kaffeine, Rhythmbox a Totem[2].

2.6 VLC

VLC je naprosto univerzální přehrávač. Kromně samotného přehrávání všech možných typů médií dokáže také pracovat jako streamovací server. Má zabudovanou podporu různých zařízení jako jsou například TV karty a kamery. Jejich obraz dokáže nejen zobrazit, ale

[2]Tento přehrávač je založen jak na knihovně xine tak na knihovně GStreamer.

také uložit na disk, nebo vysílat po internetu. Z hlediska uživatelského rozhraní jde o velmi pěkný a povedený přehrávač.

Tento přehrávač jsem si vybral pro svou bakalářskou práci a jeho podrobnějšímu popisu věnuji celou následující kapitolu (3).

Kapitola 3

Přehrávač VLC

VLC media player [5] je multiplatformní multimediální přehrávač a streamovací server vyvíjený v rámci projektu VideoLAN. Díky tomu, že má v sobě zakomponovány veškeré potřebné kodeky se z něj stává naprosto univerzální a soběstačný přehrávač. Poradí si s většinou známých typů multimédií. Uživatelé si mohou vybrat z mnoha různých způsobů ovládání. Díky tomu se dá VLC ovládat přes klasické uživatelské rozhraní (wxWidgets nebo skins2), konzoli (příkazy nebo rozhraní ncurses), webové rozhraní, Telnet nebo gesta myši. Na obrázku 3.1 je zobrazeno klasické uživatelské rozhraní VLC.

Obrázek 3.1: Základní uživatelské rozhraní přehrávače VLC (wxWidgets).

Kromě klasického přehrávání videa nabízí VLC také integraci s webovým prohlížečem, možnost vysílat (streamovat) po síti, ukládat stream na disk a funkci time shifting. To z něj dělá opravdu univerzální nástroj pro práci s videem a zvukem. Příkladem zajímavého využití VLC budiž vysílání televizních programů, přijímaných pomocí televizní karty, po kolejní síti.

Na sledování přednášek je VLC naprosto ideální. Umožňuje jak sledování živého přenosu z přednášky tak sledování jejího záznamu přímo ze serveru bez nutnosti stažení do počítače.

Aktuální verzi přehrávače je možné stáhnout na stránkách http://www.videolan.org/vlc/. K dispozici je instalátor pro Windows, předkompilované balíčky pro různé distribuce systému Linux a zdrojové kódy. O velké rozšířenosti a oblíbenosti VLC svědčí počet stažení aktuální verze blížící se k 79 000 000 (ze dne 8.5.2008).

3.1 Vnitřní uspořádání

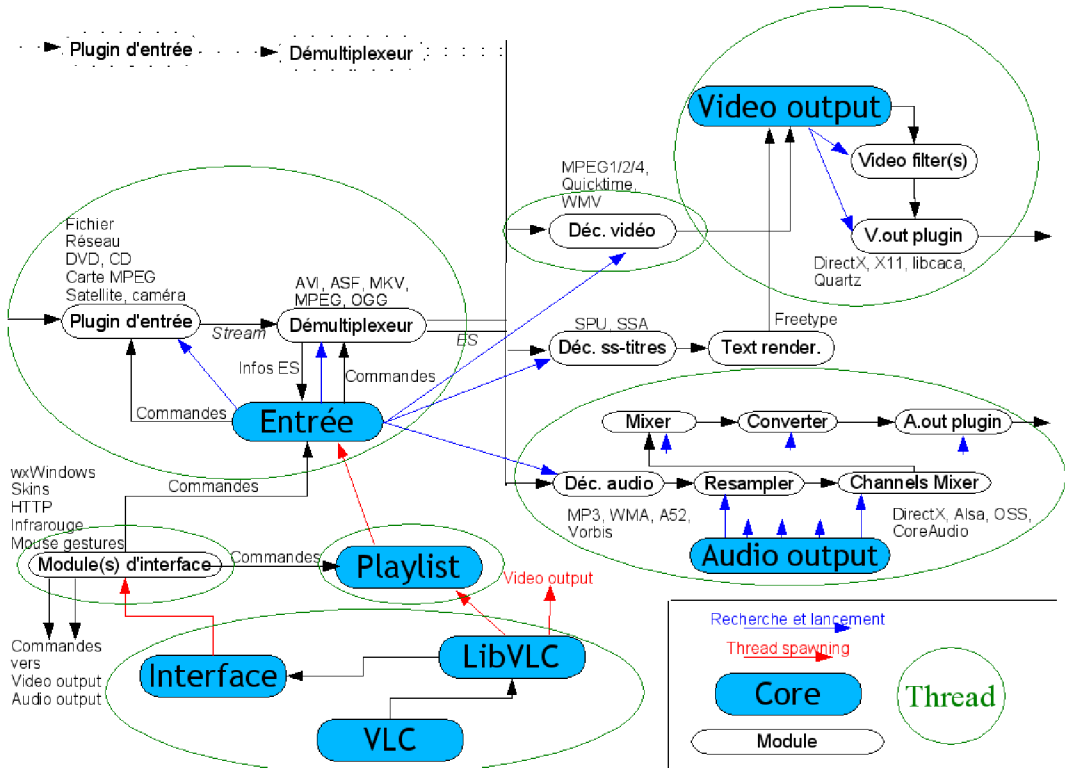
Celý přehrávač VLC je složen z modulů. Tyto moduly jsou na sobě téměř nezávislé. Díky tomu je možné upravit jednu část bez vlivu na zbytek přehrávače. Je také velmi jednoduché

přidat modul zcela nový. Přidáváním nového modulu se budu zabývat v další části této kapitoly (3.2).

Všechny moduly jsou sdruženy do kategorií a subkategorií. Dále je jim přiřazena jasná funkce (capability) a priorita (score). Při zavádění modulů se určí potřebné funkce a podle nich jsou moduly vybrány. Pokud pro určitou funkci existuje více modulů, vybere se ten s nejvyšší prioritou. V případě chyby při zavádění modulu (například neznámý formát vstupních dat) se zavádění přeruší a je vybrán modul s prioritou nižší. Díky tomuto postupu je vždy vybrán modul, který nejlépe odpovídá všem požadavkům. Dalo by se říci že přehrávač za běhu dokáže změnit svou vnitřní strukturu tak, aby co nejlépe odpovídala typu přehrávaného média.

Z programátorského hlediska jsou všechny moduly reprezentovány jako objekty. Mají své metody, konstruktor a destruktor, datové položky (atributy) a díky důmyslné sadě maker je implementována i dědičnost. Velmi elegantně je také vyřešena možnost lokalizace.

Za běhu je VLC rozdělen do několika vláken (threads), která spolu vzájemně komunikují. To umožňuje spravedlivé vytížení všech částí přehrávače a dosažení plynulého přehrávání. Dekódování a přehrávání se děje naprosto nezávisle (asynchronně). Tím je docíleno toho, že všechny části zvuku i videa jsou přehrány přesně v určeném čase bez nutnosti čekání na dekodéry. Na obrázku 3.2 je znázorněna vnitřní struktura VLC při přehrávání videa s titulky s vyznačením jednotlivých vláken a komunikačních cest. Tento obrázek jsem převzal od jednoho z vývojářů VLC [1].



Obrázek 3.2: Vnitřní struktura VLC

Seznam všech kategorií modulů je uveden v Průvodci úpravami VLC [2]. Já ve své práci uvedu jen několik základních.

3.1.1 Moduly jádra

Jádro přehrávače VLC se skládá z několika modulů. Ty se starají o běh celého přehrávače.

- **VLC** Spustitelný program. Zpracovává parametry příkazové řádky a zavádí modul LibVLC. Při použití VLC jako knihovny nemá tento modul smysl a o zavedení LibVLC se stará aplikace která tuto knihovnu používá.
- **LibVLC** Zapouzdření VLC do knihovny. Umožňuje použít VLC jako knihovnu pro přehrávání multimediálního obsahu v cizích aplikacích. Zavádí všechny potřebné moduly jádra. Poskytuje rozhraní pro řízení běhu přehrávání a přístup k vnitřním proměnným VLC.
- **Interface** Zavádí moduly uživatelského rozhraní.
- **Playlist** Zavádí modul Input. Je ovládán příkazy od uživatelským rozhraním a na jejich základě řídí celé přehrávání.
- **Input** Podle typu přehrávaného média vybírá a zavádí potřebné vstupní moduly, demultiplexor a dekodéry. Dále se stará o nahrávání a ukládání nahraného videa na disk.
- **Video output** Zavádí a řídí filtry videa a výstupní modul.
- **Audio output** Zavádí a řídí všechny moduly potřebné pro konečnou úpravu a výstup zvuku.
- **Stream output** Zavádí a řídí všechny moduly potřebné pro sestavení streamu a jeho vysílání po síti. V případě potřeby duplikuje data z dekodérů pro moduly výstupu videa a zvuku.

3.1.2 Vstupní moduly

Moduly vstupu se starají o otevření média, zjištění základních informací a zprostředkování dat v nějaké jednotné formě dále do přehrávače. Mezi základní patří například typ média, časování a možnost přesouvat se (seek) v obsahu. Tyto informace jsou velmi důležité pro běh celého přehrávače. Pokud není možné se v obsahu přesouvat, nemá smysl zobrazovat ukazatel průběhu a hlavně nelze měnit rychlost přehrávání. Médiem bez možnosti přesouvat se v obsahu je například živý přenos z přednášky.

3.1.3 Demultiplexory

Tyto moduly mají na starost rozdělení vstupního proudu na video, zvuk a titulky. Téměř pro každý typ média je potřeba jiný demultiplexor. V místě demultiplexoru jsou data naposledy v jednom vláknu programu. Proto musí demultiplexor zprostředkovat informace o časování jednotlivých částí.

3.1.4 Dekodéry

Dekodérů jsou tři typy. Dekodéry videa, dekodéry zvuku a dekodéry titulků. Jejich prací je převést vstupní data do jednotné, nekomprimované (raw) formy. Po dekódování už téměř nezáleží na tom, jaký formát média je přehráván. Jedinou neměnitelnou vlastností je již zmíněná možnost přesouvat se v obsahu.

3.1.5 Filtry videa

Mezi tyto moduly patří různé efekty, odstranění prokládání, rozdělení videa na více částí a také zakomponování OSD[1] informací do obrazu. Mezi tento druh informací patří i titulky.

3.1.6 Filtry zvuku

Zpracování zvuku se skládá z několika kroků.

- Úprava rychlosti zvuku za pomoci převzorkování (resampling). Mezi tyto moduly přidávám v této bakalářské práci svůj vlastní, který za pomoci metody PSOLA zachovává srozumitelnost řeči při změně rychlosti.
- Úprava počtu zvukových kanálů. Tento krok je důležitý například při přehrávání prostorového zvuku na stereo reproduktorech, nebo naopak při přehrávání stereo nahrávky na domácím kinu.
- Úprava zvuku pomocí ekvalizéru a dalších filtrů. Ve VLC je implementován desetikanálový ekvalizér s několika předvolbami a možností dvouprůchodového zpracování. Mezi zvukové filtry patří například normalizace hlasitosti.
- Převod do formátu vhodného pro výstup. Zvuk je ve VLC zpracováván převážně ve formátu s plovoucí řádovou čárkou. V některých případech je však tento formát odlišný.

3.1.7 Výstupní moduly

Moduly pro výstup videa a zvuku. Tyto moduly se liší v závislosti na architektuře cílového operačního systému. Ve Windows je výstup realizován pomocí rozhraní DirectX. V Linuxu jsou to například X11[2] pro video a ALSA[3] pro zvuk.

3.2 Rozšiřitelnost

Díky své modularitě je VLC snadno rozšiřitelný. Bohužel, dokumentace je v době psaní této práce nekompletní a dosti obecná. Vývojáři slibují, že se vše v budoucnosti zlepší. I přes tuto komplikaci je psaní modulů pro VLC celkem snadné. Moduly je možné psát buď v jazyku C, nebo v jazyku C++. O přeložení a provázání všech modulů se stará sada automatizovaných konfiguračních a překladových skriptů. V podstatě stačí pouze napsat modul, přidat seznam zdrojových kódů potřebných pro přeložení modulu do seznamu zdrojových kódů, přidat modul do hlavního konfiguračního skriptu a přeložit.

Během překladu je modul zakomponován do přehrávače jako dynamická knihovna. Je také možné modul vytvořit naprosto samostatně a potom jej do VLC přidat jako již zkompilovanou knihovnu. O tuto možnost jsem se příliš nezajímal.

Při psaní modulu je kvůli nekompletní dokumentaci vhodné vycházet z již hotového a funkčního modulu. Důležité části jsou většinou velmi podobné a neliší se ani v různých kategoriích. Většina datových struktur je popsaná pouze v hlavičkových souborech. To velmi znesnadňuje pochopení VLC, protože komentáře občas chybí, nebo přesně nevystihují podstatu části kódu.

[1] On-Screen Display
[2] X Window System
[3] Advanced Linux Sound Architecture

3.2.1 Struktura modulu

Všechny moduly ve VLC mají společnou základní strukturu. Obsahují hlavičku modulu a základní funkce potřebné pro svou funkci. V hlavičce modulu se definují všechny potřebné parametry, například o jaký modul se jedná a jaké má konfigurační hodnoty (zobrazené v nastavení VLC). Je také možné dodefinovat submoduly. To je užitečné zejména u rozsáhlejších modulů.

Pro začátek bych rád upozornil na dvě důležitá makra. Jsou to `N_( string )` a `_( string )`. První z nich vytvoří přeložitelný (lokalizovatelný) řetězec. Druhé navíc řetězec přímo překládá. Při psaní modulu doporučuji používat první z maker. Jeho použití je naprosto bezpečné. Druhé z maker může způsobit ošklivé problémy pokud nebude řetězec správně přeložen. Při lokalizaci je samozřejmě nutné přejít na druhé makro aby se změny projevily.

Hlavička modulu

Hlavička modulu se vytváří pomocí vkládání několika maker. Tato makra jsou definována v hlavičkovém souboru `vlc_modules_macros.h`. Já popíši jen ty nejdůležitější.

- `vlc_module_begin( )` Začátek definice modulu.
- `vlc_module_end( )` Konec definice modulu.
- `add_shortcut( shortcut )` Definice vnitřního jména modulu.
- `set_shortname( shortname )` Definice jména modulu, které se zobrazuje například v nastavení.
- `set_capability( cap, score )` Definice funkčnosti modulu. Položka `cap` určuje kategorii modulu. Položka `score` určuje jeho prioritu. Tato položka má zřejmě i jiný význam, který se mi nepodařilo zcela objasnit. Zdá se že hodnota nějakým způsobem zpřesňuje funkci modulu. Například u resamplerů se hodnota této položky pohybuje mezi 1 a 99. Hodnota 0 je přiřazena obecným filtrům.
- `set_callbacks( activate, deactivate )` Definice konstruktoru a destruktoru, neboli funkcí, které se zavolají při aktivaci a deaktivaci modulu.

Další makra, definovaná v hlavičkovém souboru `vlc_configuration.h`, umožňují k modulu definovat konfigurační hodnoty. Tyto hodnoty je po spuštění možné měnit v nastavení přehrávače. Opět uvedu příklady těch nejdůležitějších.

- `set_category( i_id )` Definice kategorie. V tomto případě jde čistě o kategorii pod kterou se bude nastavení modulu zobrazovat v menu.
- `set_subcategory( i_id )` Definice subkategorie. Platí to samé co pro kategorii.
- `add_integer( name, value, p_callback, text, longtext, advc )` Definice číselné konfigurační hodnoty. Parametr `name` určuje vnitřní název hodnoty. Parametr `value` určuje výchozí hodnotu. Parametr `p_callback` určuje ukazatel na funkci, která se provede po změně hodnoty. Parametr `text` určuje zobrazovaný název položky. Parametr `longtext` určuje popis hodnoty. Parametr `advc` určuje, zda bude hodnota zobrazena v normálním, nebo pouze v pokročilém nastavení.

- `add_string( name, value, p_callback, text, longtext, advc )` Definice textové konfigurační hodnoty. Význam parametrů je stejný jako u makra pro přidání číselné hodnoty.
- `change_string_list( list, list_text, list_update_func )` Změna textové konfigurační hodnoty na seznam s výběrem. Parametr `list` určuje seznam všech možných konfiguračních hodnot. Parametr `list_text` určuje názvy těchto hodnot. Poslední parametr `list_update_func` není nikde použit. Zřejmě jde o pozůstatek z minulosti, nebo plánované rozšíření do budoucna.

Konstruktor a destruktor

Tyto funkce mají jediný parametr. Tím je ukazatel na některého z potomků struktury `vlc_object_t`. V podstatě je to ukazatel na objekt. Každý modul má ve svém objektu atribut `p_sys`. Do tohoto atributu si modul ukládá vlastní strukturu s potřebnými daty. Úkolem konstruktoru je tuto strukturu vytvořit a inicializovat. Dále kontroluje, zda je vůbec možné modul použít. Destruktor se stará o uvolnění alokované paměti, popřípadě odstranění všech dynamicky vytvořených objektů.

3.2.2 Zvukové filtry

Zvukové filtry jsou potomky typu `vlc_object_t`. Rozšiřují tento typ o informace o zpracovávaném zvukovém proudu. Dále přibývá ukazatel `pf_do_work`. Do tohoto ukazatele je při inicializaci modulu potřeba přiřadit ukazatel na funkci provádějící vlastní zpracování (výkonnou funkci). Dalším novým atributem je `b_in_place`. Ten určuje, zda se bude alokovat nový výstupní buffer, nebo zda postačí buffer vstupní, který se přepíše.

Výkonná funkce

Výkonná funkce má čtyři parametry. Prvním parametrem je ukazatel na instanci celého vlákna zvukového výstupu. Je to vlastně ukazatel na modul Audio output z jádra VLC. Dalším parametrem je ukazatel na instanci modulu. Pro tento parametr budu v následujícím popisu používat označení `p_filter`. Poslední dva parametry obsahují ukazatele na vstupní a výstupní buffery.

Buffery jsou implementovány jako struktury obsahující pole s daty a další potřebné informace.

- `p_buffer` Pole s daty. Podle hlavičkového souboru je typu `uint8_t *`, ale většinou obsahuje položky typu `float`.
- `i_alloc_type` Určuje typ alokace: 0 - žádná, 1 - alokace na zásobníku (stack), 2 - alokace na haldě (heap).
- `i_size` Reálná velikost bufferu. Používá se pouze k ladícím účelům.
- `i_nb_bytes` Skutečný počet využitých bytů v bufferu.
- `i_nb_samples` Počet vzorků v bufferu. Tato hodnota určuje počet vzorků v jednom kanálu. V bufferu je uložen celkový počet vzorků rovnající se této hodnotě krát počet kanálů.
- `start_date` Časová značka určující začátek přehrávání této části zvukového proudu.

- `end_date` Časová značka určující konec přehrávání této části zvukového proudu.
- `b_discontinuity` U proudů, které nevyužívají PCM[4], označuje nespojitost.
- `p_next` Ukazatel na následující buffer.
- `p_sys` Tato položka se již nepoužívá a bude odstraněna.
- `pf_release` Díky tomuto ukazateli je možné dodefinovat vlastní funkci, která bude buffer uvolňovat.

Typ dat uložených v bufferu se dá zjistit z atributu `p_filter->input.i_format`. Pokud se tato hodnota rovná výstupu makra `VLC_FOURCC('f','l','3','2')`, jde o typ `float`. Pokud se rovná `VLC_FOURCC('f','i','3','2')`, jde o typ `unsigned int`. Já jsem se osobně setkal pouze s první možností a většina novějších filtrů s touto možností počítá.

Počet kanálů je možné získat pomocí funkce `aout_FormatNbChannels( &p_filter->input )`. Vzorky z jednotlivých kanálů jsou v bufferu uloženy po skupinách. Po vzorcích číslo 1 ze všech kanálů následují vzorky číslo 2 a tak dále.

[4]Pulse-Code Modulation

Kapitola 4

Metoda PSOLA a fonémový rozpoznávač

Ve své bakalářské práci používám zdrojové kódy pro výpočet metody PSOLA z diplomové práce Ing. Kovářika. Mým úkolem je použít tyto zdrojové kódy v přehrávači VLC. Text této kapitoly je zjednodušeným výtahem z diplomové práce Ing. Kovářika [7], do kterého nepřidávám žádné nové myšlenky. Použité obrázky rovněž pochází z uvedené diplomové práce.

Metoda PSOLA (Pitch Synchronous OverLap Add) byla vyvinuta společností France Telecom (CNET). Slouží pro snadnou úpravu délky a základního tónu signálu. Syntetizéry používající tuto metodu se vyznačují vysokou kvalitou výstupu a nízkou výpočetní náročností. Pitch synchronous znamená, že segmenty hlasové nahrávky jsou napojovány tak, aby se překryly periody základního tónu. Tím je zamezeno vzniku nepříjemných zvukových artefaktů.

PSOLA má několik variant. Varianta, kterou Ing. Kovářík používá ve své diplomové práci pracuje v časové oblasti. Proto je nazývána TD-PSOLA[1] (Time Domain PSOLA). Ta je velmi jednoduchá a výpočetně nenáročná. Dalšími variantami jsou FD-PSOLA (Frequency Domain) pracující ve frekvenční oblasti, LP-PSOLA (Linear Prediction) pracující s lineární predikcí a WD-PSOLA (Wawelet Domain) pracující s vlnkovou transformací.

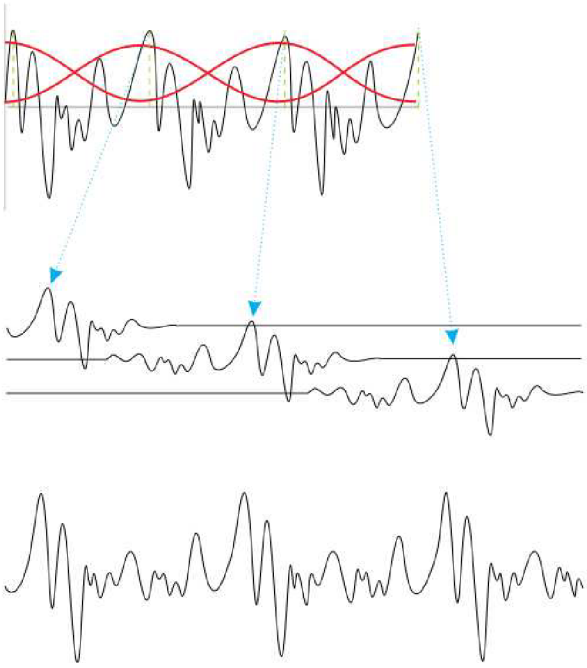
Algoritmus PSOLA pracuje následujícím způsobem: V hlasovém signálu se určí základní tón. Přesné určení period základního tónu je klíčem ke kvalitnímu výsledku. Začátky period se označí Pitch Marky (Pitch Marks). Spojování segmentů metodou překryj a sečti (OverLap Add) probíhá na úrovni period základního tónu.

Aby byl přechod mezi periodami co nejhladší, je nutné tyto periody „(“vyseknout) pomocí okénkovací funkce. Často používanou okénkovací funkcí je Hammingova funkce. Okénkovací funkce se aplikuje v rozsahu dvou až čtyř period základního tónu v okolí pitch marku na jehož pozici je okénkovací funkce vystředěna.

Následuje výpočet nových pozic pitch marků na něž jsou namapována získaná okénka. Ta jsou potom pomocí metody OLA sečtena a je získán syntetizovaný výstupní signál. Pomocí manipulace s polohou pitch marků lze docílit změny základního tónu. Pomocí zdvojování nebo vynechávání pitch marků lze docílit změny délky syntetizovaného signálu. Na obrázku 4.1 je znázorněna metoda PSOLA př snižování základního tónu.

Ve své diplomové práci Ing. Kovařík mění rychlost řeči právě díky manipulaci s pitch marky. Při klasické změně rychlosti je koeficient změny β konstantní. Při zapojení fonémového

[1] PSOLA/TD® je registrovanou obchodní značkou společnosti France Telecom.



Obrázek 4.1: Funkce metody PSOLA při snížení základního tónu

rozpoznávače se však tento koeficient v mění v čase podle flexibility zjištěných fonémů. Tím je dosaženo větší srozumitelnosti syntetizovaného signálu.

4.1 Detekce základního tónu

Detekce základního tónu a správné umístění pitch marků má ohromný význam na kvalitu syntetizovaného signálu. Při potřebě co nejpřesnější detekce je tento úkol svěřen lidem, kteří ručně zadávají pozice pitch marků. To je však zdlouhavé. Proto se tento postup používá například při tvorbě fonémové databáze pro tvorbu řeči. Během přehrávání videa se musíme spokojit se strojovou detekcí základního tónu.

V následující části práce uvedu stručný popis některých metod strojové detekce základního tónu. Všechny uvedené metody pracují s rámci. To znamená že nezpracovávají celý signál najednou, ale pouze jeho jednotlivé části. To je velmi důležitá vlastnost pro použití těchto metod při úpravě rychlosti za běhu, kdy není možné prozkoumat předem celý signál.

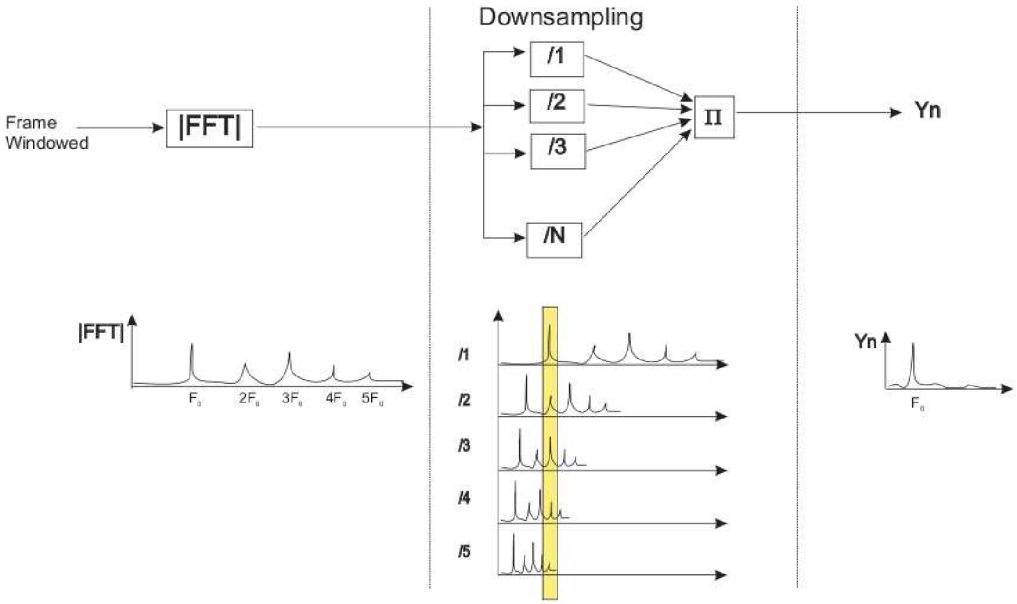
4.1.1 Autokorelační funkce

Nejjednodušším způsobem získání základního tónu je CCF (Cross-Correlation Function). Tato funkce pracuje v časové oblasti. Princip této funkce spočívá v posouvání signálu vůči sobě samotnému a hledání co největší shody. Podobnost se určuje vynásobením posunutých vzorků a následným sečtením těchto násobků. Posunutí s nejvyšším výsledkem autokorelační funkce odpovídá periodě základního tónu.

Podobnou funkcí je AMDF (Average Magnitude Difference Function). Ta je méně výpočetně náročná, protože místo součinu používá rozdíl a místo maxima hledá minimum.

4.1.2 Harmonic Product Spectrum (HPS)

Harmonic Product Spectrum pracuje ve frekvenční oblasti. Vychází z předpokladu že řečový signál je výsledkem filtrování základního tónu v artikulačním ústrojí. Řečový signál se tedy skládá ze základního tónu a jeho celočíselných násobků (harmonických složek). Algoritmus tedy převede řečový signál pomocí rychlé Fourierovy transformace (FFT) do frekvenční oblasti. Následně tento převedený signál (spektrum) několikrát podvzorkuje. Výsledkem se součin spektra signálu a jeho podvzorkovaných podob. V místě maxima tohoto součinu se nachází základní tón. Princip algoritmu HPS je znázorněn na obrázku 4.2



Obrázek 4.2: Harmonic Product Spectrum

4.1.3 Harmonic and Noise Model (HNM)

Trasovač spojitého základního tónu na základě modelu harmonických a šumu myl vyvinut na naší fakultě. Jde o velmi sofistikovaný a přesný algoritmus. Na rozdíl od předchozích metod dokáže tento algoritmus doplnit pitch marky i v neznělých částech řečového signálu.

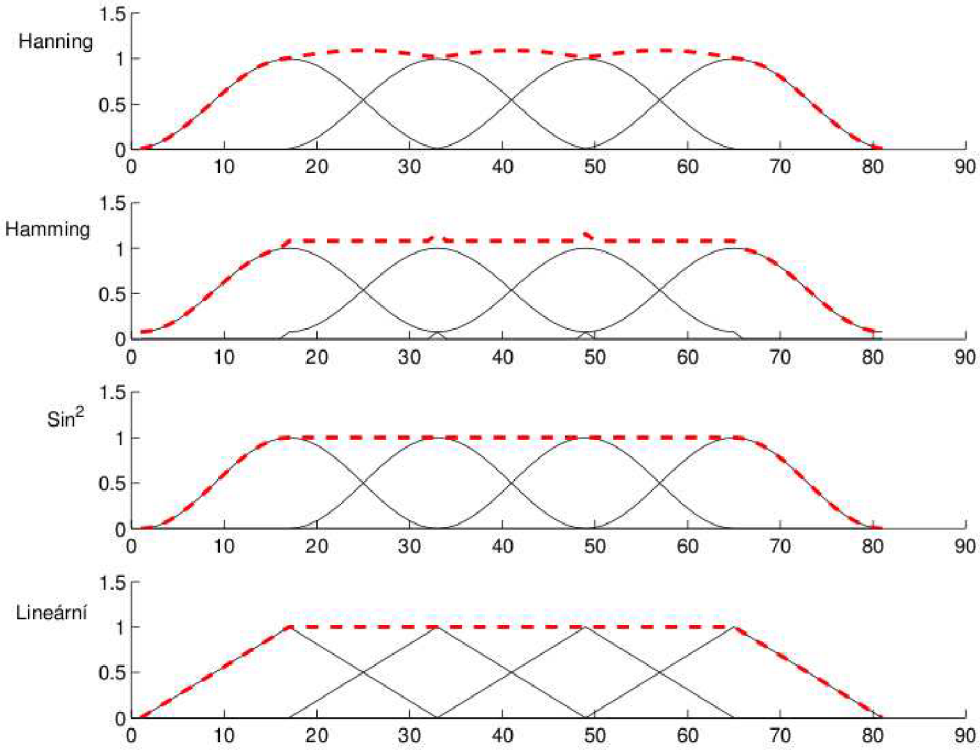
Principem tohoto algoritmu je, že se snaží vyrobit signál co nejpodobnější zkoumanému řečovému signálu. Podobně jako HPS vychází z předpokladu, že řečový signál je složen převážně ze základního tónu a harmonických složek. Pro získání modulů a fází frekvencí, ze kterých se řečový signál skládá je použita rychlá Fourierova transformace. Na základě těchto informací jsou potom pro určitý rozsah frekvencí základního tónu generovány signály a je zkoumána jejich podobnost se signálem řečovým. Podobnost se určuje na základě výpočtu chyb. Tyto chyby se ukládají do chybové matice.

Pomocí metody hledání cesty s nejmenší cenou (chybou) je v chybové matici nalezena základní frekvence pro každou znělou část signálu. Pro určení znělosti a neznělosti je použita metoda prahování. Ve znělých částech se pitch marky umísťují do míst předpokládaného vrcholu excitace periody základního signálu. V neznělé části se pitch mark umístí na místo součtu pozice předchozího pitch marku a periody základní frekvence.

4.2 Okénkovací funkce

Volba správné okénkovací funkce má vliv na celkovou kvalitu syntetizovaného signálu. Při volbě špatné funkce se v signálu mohou vyskytovat nechtěné artefakty. Nejčastěji se používá Hanningova nebo Hammingova funkce. Ing. Kovářík ve své práci vysvětluje, že tyto funkce při použití metody OLA nepříznivě ovlivňují výstupní signál. Kromě žádané změny rychlosti vstupního signálu také zvyšují jeho energii, protože součet sousedních segmentů těchto okénkovacích funkcí je větší než 1.

Jako nejlépe použitelná funkce se jeví sin^2. Díky tomu, že funkce sin při posunu o čtvrt peroidy odpovídá funkci cos, dochází při okénkování přes dvě periody základního tónu k překrytí funkce sin^2 a cos^2. Součet těchto 2 funkcí je 1. Tato okénkovací funkce také preferuje část kolem středu pitch marku a zmírňuje tak možné interference mezi sousedními segmenty. Vlastnost součtu rovného 1 má i lineární funkce. Ta však způsobuje větší promísení souseních segmentů. Základní okénkovací funkce jsou znázorněny na obrázku 4.3.



Obrázek 4.3: Okénkovací funkce

4.3 Výběr period základního tónu

Výběr period základního tónu určuje, zda bude měněna rychlost celého signálu nebobude měněn jeho základní tón. Možná je samozřejmě i kombinace. Záleží na tom, zda se bude měnit vzdálenost mezi pitch marky, nebo budou některé z nich zdvojeny popřípadě vynechány.

Při úpravě vzdálenosti mezi pitch marky dochází ke změně základního tónu. Pro snížení základního tónu se tato vzdálenost zvětšuje, naopak pro zvýšení základního tónu se zmenšuje.

Pro zachování délky signálu je nutné některé pitch marky vynechat, nebo naopak zdvojit.

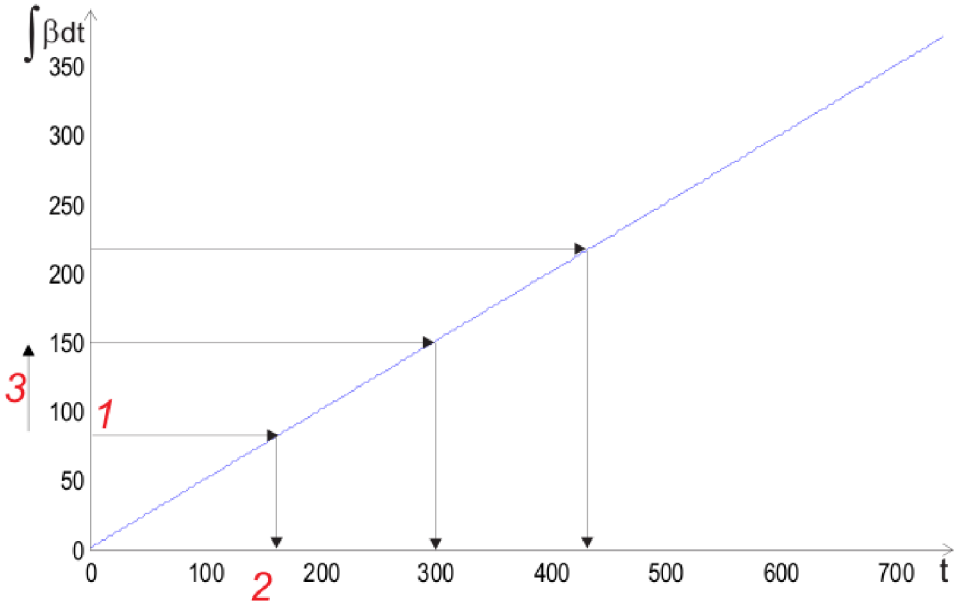
Při změně rychlosti signálu je nutné určit, které periody základního tónu se použijí a které ne. Nové pitch marky se nazývají syntetizované. Jejich výběr se provádí pomocí integrace funkce koeficientu změny rychlosti β. Při běžné změně rychlosti je tato funkce lineární a integrace je konstantní. Na obrázku 4.4 je znázorněn výběr period základního tónu pro koeficient změny $0,5$.

Na začátku výběru se první pitch mark ze zdrojového signálu označí jako syntetizovaný. V dalších iteracích se pokračuje podle následujícího algoritmu:

1. Na grafu integrálu se vyhledá hodnota syntetizovaného pitch marku.
2. Vzorek z osy x, který tuto hodnotu nesl spadá do nějaké periody. Tento vzorek je označen jako virtuální pitch mark.
3. Délka této periody se přičte k syntetizovanému pitch marku a tím je nalezen nový.

Kroky 1 - 3 se opakují až do dosažení konce vstupního signálu.

Virtuální pitch marky určují, které periody základního tónu se použijí při syntéze nového signálu. Nové segmenty se na konec syntetizovaného signálu připojují tak, že jejich levý okraj je zarovnán na střed předchozího segmentu.



Obrázek 4.4: Určení syntetizovaných pitch marků

4.4 Rozšíření o fonémový rozpoznávač

Aby byl výstup metody PSOLA co možná nejsrozumitelnější, je potřeba zohlednit flexibilitu jednotlivých fonémů. Fonémy jsou nejmenší součásti zvukové stránky řeči, které mají rozlišovací funkci. Záměna fonému má schopnost změnit význam slova [8]. Každý foném se od ostatních liší alespoň jednou fonologickou distinktivní vlastností. Jednou z těchto vlastností je například kvantita — délka trvání fonému.

Spisovná čeština obsahuje 36 fonémů. Okolo tohoto počtu a seznamu všech fonémů se vedou debaty a pře. Ing. Kovářík používá ve své diplomové práci českou fonémovou sadu z knihovny BSAPI vyvíjené na naší fakultě. Tato fonémová sada obsahuje celkem 46 fonémů včetně některých, které zastupují různé chyby signálu a ruchy v pozadí.

4.4.1 Fonémový rozpoznávač

Fonémový rozpoznávač detekuje fonémy v řečovém signálu. Ve zmíněné knihovně BSAPI je jako jeden z modulů implementován fonémový rozpoznávač založený na hierarchických neuronových sítích. Vstupem tohoto rozpoznávače je řečový signál a výstupem sekvence fonémů.

4.4.2 Rozšíření PSOLY

Pro každý foném byla určena jeho flexibilita γ. Ta říká, do jaké míry je možné změnit délku fonému se zachováním jeho srozumitelnosti. Flexibilitou jednotlivých fonémů je následovně modifikována funkce β. Vzhledem k tomu, že v běžné řeči je dosti velké zastoupení méně flexibilních fonémů, je výsledná změna rychlosti menší, než jaká byla požadována.

Kapitola 5

Implementace

Metotu PSOLA jsem do přehrávače VLC začlenil jako zvukový filtr. Jak jsem již popsal v kapitole Přehrávač VLC (3), existuje více druhů zvukových filtrů. Z vlastností metody PSOLA vyplývá, že mění počet vzorků a tím upravuje délku (rychlost) signálu. Provádí vlastně převzorkování (resampling).

Zvukových filtrů typu resampler je ve VLC implementováno několik. Liší se výpočetními nároky a hlavně kvalitou. Já jsem při začlenění metody PSOLA použil jako základ resampler s označením `trivial`.

5.1 Hlavička modulu

V hlavičce modulu bylo nutné dodefinovat konfigurační hodnoty. Textová hodnota `psola_method` určuje algoritmus použitý pro detekci základního tónu. V diplomové práci Ing. Kovářka jsou pro tento účel implementovány tři různé algoritmy.

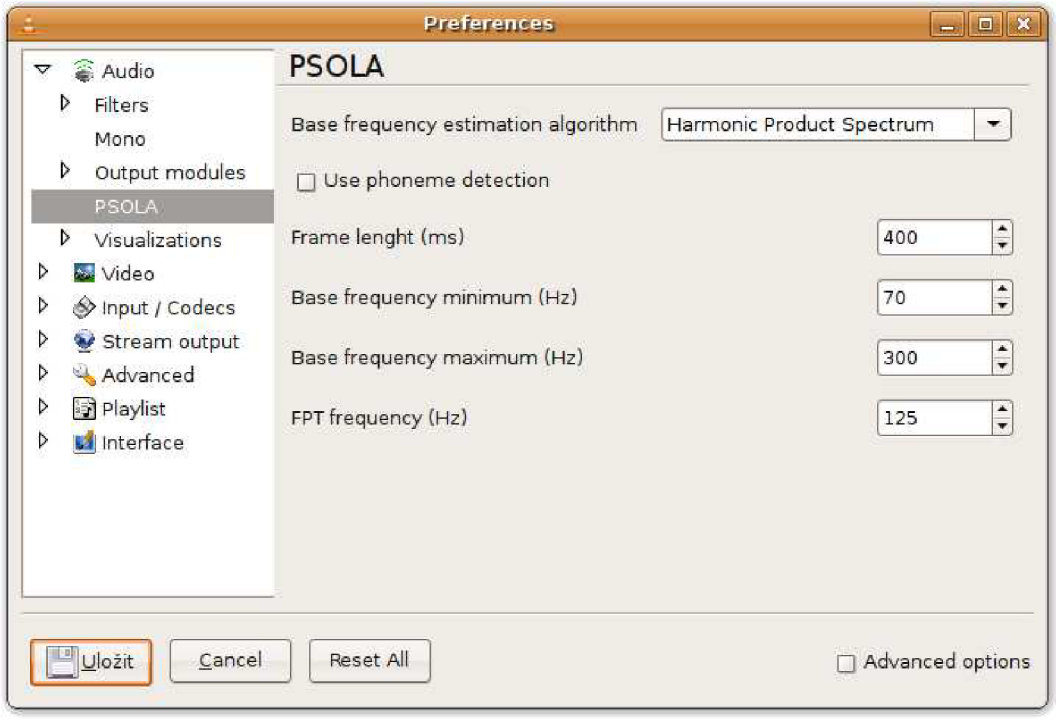
- `Fixed Pitch Tracker` Tento algoritmus neprovádí žádnou analýzu. Pitch marky rozmístí v určených rozestupech. Degraduje metodu PSOLA na obdobu metody OLA.
- `Harmonic Product Spectrum` Jde o implementaci algoritmu HPS. Ing. Kovářík tuto metodu rozšířil o překlenutí neznělých částí signálu pomocí dynamického programování a interpolace. Tento postup je podobný jako u algoritmu HNM.
- `Harmonic and Noise Model` Tento vysoce přesný a také výpočetně náročný algoritmus využívá postupy detekce popsané v sekci 4.1.3.

Pro zjednodušení výběru algoritmu jsem se rozhodl použít seznam s výběrem těchto tří algoritmů.

Další konfigurační hodnotou je zapnutí fonémového detektoru. Tuto možnost jsem bohužel musel zneplatnit z důvodů popsaných dále. Toto zaškrtávací políčko tedy už nemá smysl, ale rozhodl jsem se ho ponechat pro možné budoucí zprovoznění.

Následuje několik číselných hodnot pro určení parametrů filtru. První z nich je délka jedné dávky signálu v milisekundách. Určuje jak velké časové úseky signálu se budou zpracovávat najednou. Další z konfiguračních hodnot jsou minimální a maximální frekvence hledaného základního tónu. Poslední hodnotou je frekvence základního tónu pro Fixed Pitch Tracker.

Na obrázku 5.1 je zobrazeno dialogové okno s nastavením modulu PSOLA.



Obrázek 5.1: Nastavení modulu PSOLA

Dále jsem musel upravit definici schopností modulu. Kategorie modulu zůstala podle vzorového modulu nastavena na `audio filter`, prioritu jsem upravil na 50. Modul PSOLA se proto vždy použije jako výchozí.

5.2 Konstruktor a destruktor

Pro zavedení modulu slouží funkce `Open( vlc_object_t *p_this )`. Na začátku zkontroluje, zda jsou splněny všechny podmínky pro zavedení. Vstupní počet vzorků se musí odlišovat od výstupního, musí být shodný počet vstupních a výstupních kanálů a formát vzorků musí odpovídat formátu `VLC_FOURCC('f','l','3','2')` což je 32 bitové desetinné číslo s plovoucí řádovou čárkou (typ `float`). Pokud nejsou všechny podmínky splněny, zavádění modulů končí.

Následuje alokace struktury pro uložení lokálních proměnných modulu. Součásti těchto lokálních proměnných jsou následující:

- Vstupní a výstupní buffery metody PSOLA. Velikost těchto bufferů jsem zvolil pro uložení pěti sekund zvukového signálu. Tato velikost je více než dostačující. Využívá se velikost odpovídající délce jedné dávky zpracovávaného signálu. Počet vstupních a výstupních bufferů je závislý na počtu zvukových kanálů. Pro každý kanál je vytvořen jeden vstupní a jeden výstupní buffer.
- Instance objektu `psola` z diplomové práce Ing. Kováříka. Tento objekt zprostředkovává výpočet metody PSOLA.

- Identifikátor zvolené metody detekce základního tónu. Tento identifikátor je implementován jako výčtový typ. Při zavádění modulu se rozpozná konfigurační hodnota se zvolenou metodou a nastaví se patřičná hodnota.
- Minimální frekvence základního tónu.
- Maximální frekvence základního tónu.
- Frekvence základního tónu metody OLA.
- Časová značka začátku dat uložených na vstupním bufferu metody PSOLA.

Poslední částí zavádění modulu je nastavení ukazatele na výkonnou funkci a nastavení hodnoty `p_filter->b_in_place`. Tuto hodnotu jsem nastavil na `true`. Toto nastavení zamezuje alokaci výstupního bufferu modulu. Vstupní i výstupní buffer je potom jednotný. Ve svém modulu všechna vstupní okamžitě kopíruji do vstupního bufferu metody PSOLA. Proto je možné je následně přepsat daty výstupními.

Funkce `Close( vlc_object_t *p_this )` se stará o uvolnění všech alokovaných prostředků a zrušení instance objektu `psola`.

5.3 Výkonná funkce

Na začátku je vypočten koeficient změny rychlosti. Pokud se tento koeficient rovná jedné, není potřeba měnit rychlost. Příznak konzistence vstupních dat je nastaven na hodnotu `false` a provádění funkce je ukončeno.

V opačném případě je provedena kontrola příznaku konzistence. Pokud je zaznamenána nekonzistence, jsou smazány vstupní i výstupní buffery metody PSOLA. Tato možnost nastane například pokud uživatel změní požadovanou rychlost přehrávání, nebo se přesune na jiné místo v přehrávaném médiu.

Následně jsou vstupní data převedena do formátu, kde jsou jednotlivé zvukové kanály odděleny a tato data jsou uložena do vstupních bufferů metody PSOLA. Pokud jsou před vložením tyto vstupní buffery prázdné, je časová značka začátku vstupních dat uložena. Tato informace je později použita pro synchronizaci.

Pokud je na vstupních bufferech dostačující počet dat ke spuštění dávky, dojde k výpočtu metody PSOLA pro všechny kanály. Postup výpočtu je následující:

- Požadovaný počet dat je vyjmut ze vstupního bufferu a předán instanci objektu `psola`.
- Na tomto místě by mělo následovat volání detektoru fonémů.
- V závislosti na identifikátoru zvoleného algoritmu pro detekci základního tónu je zavolána metoda `analyzeSignal`, které je jako parametr předána reference na příslušnou instanci detektoru základního tónu.
- Následuje volání metody `synthesizeSignal`, která provede vlastní výpočet metody PSOLA.
- Pomocí metody `get_syntSigLength` je získán počet výstupních vzorků.
- Výstupní data jsou uložena přidána do výstupního bufferu.

Po provedení výpočtu metody PSOLA je podle počtu výstupních vzorků vypočten skutečný koeficient změny rychlosti. Z tohoto koeficientu je vypočtena vzorkovací frekvence výstupního signálu. Vypočtená frakvence je použita k inicializaci vnitřní synchronizační funkce VLC. Dále je podle koeficientu změny rychlosti vypočten počet výstupních vzorků modulu.

Pokud je na výstupních bufferech dostatečný počet dat pro výstup, jsou výstupní data převedena do správného formátu a uložena na výstupní buffer. Synchronizační funkce poskytne potřebné časové značky pro správné určení začátku a konce výstupu. V opačném případě je výstupní buffer modulu zneplatněn tím, že se jeho velikost nastaví na nulu.

5.4 Fonémový detektor

Při pokusu o začlenění fonémového detektoru do VLC jsem narazil na chybu, která znemožňovala jeho použití. Z neznámých důvodů fonémový detektor při inicializaci hlásí chybu porušeného souboru s váhovými koeficienty. Tento soubor je ovšem naprosto v pořádku a při použití fonémového detektoru mimo VLC je načten korektně. Různými pokusy o odhalení a opravu chyby jsem se zabýval poměrně dlouhou dobu, bohužel bez jakéhokoliv výsledku.

V praxi by detekce fonémů pro každé přehrání přednášky nepřipadala v úvahu, protože je to velmi zdlouhavá metoda. Detekce fonémů by se provedla pouze jednou a výstup této detekce by se uložil. Studenti by si potom se záznamem přednášek stáhli i soubor obsahující detekovené fonémy. Tento soubor by se společně se záznamam přednášky načetl do přehrávače. Podobnou metodu jsem použil při testování. Pomocí doprovodného programu k diplomové práci Ing. Kováříka jsem získal posloupnost fonémů. Tuto posloupnost jsem posléze zadal metodě PSOLA.

5.5 Uživatelské rozhraní

Ve svém základu umožňuje VLC měnit rychlost po poměrně velkých krocích. Proto jsem do uživatelského rozhraní wxWidgets přidal možnost měnit rychlost po mnohem menších krocích. Tato možnost je součástí rozšířeného rozhraní, které je třeba povolit v menu `Settings->Extended GUI`. Na obrázku 5.2 je toto rozhraní znázorněno.

Obrázek 5.2: Rozšířené rozhraní s možností změny rychlosti přehrávání

Kapitola 6

Testy

K testování výsledné aplikace jsem přizval několik známých. Zkoušeli jsme přehrávat různé filmy, zvukové soubory a streamy. Zkoumali jsme kvalitu výstupu, přesnost synchronizace a vytížení systému.

6.1 Kvalita výstupu

Při použití Fixed Pitch Trackeru (degradace na metodu OLA) byl výstup poměrně zkreslený. Zvuk zněl nepřirozeně, dalo by se říci že kovově. Srozumitelnost výstupu však byla mnohem lepší, než při použití klasických resamplerů. Až na malé odchylky byl zachován základní tón řeči a proto bylo možné jednotlivé mluvčí velmi dobře rozeznat.

Použití ostatních dvou metod detekce základního tónu je dosti výpočetně náročné. U metody HPS jsme při přehrávání zvukového souboru s malou vzorkovací frekvencí a jedním zvukovým kanálem dosáhli téměř plynulého výstupu. Museli jsme ale zmenšit velikost dávek zpracovávaného signálu na desetinu výchozího nastavení, neboli na $40ms$. To mělo za následek občasné rušivé artefakty ve výstupu. Ozývalo se praskání a občasný šum na pozadí. Kvalita výstupu byla větší než u předchozí metody, ale za cenu vysokých nároků na systém. Metodu HNM není možné pro přehrávání použít vůbec.

Při testování jsem také narazil na nepříjemný problém. Aktuální verze VLC ve spojení s mým systémem nedokáže korektně přehrávat záznamy z přednášek. Obraz je v pořádku, ale zvuk je téměř neposlouchatelný. Místo výkladu přednášejícího se ozývá pouze jakési bzučení.

6.2 Přesnost synchronizace

U většiny testovaných filmů bylo po zrychlení synchronizováno dobře. Ke zpoždění docházelo u videí, jejichž zvuk má vzorkovací frekvenci $48kHz$. Je to zřejmě způsobeno tím, že při zrychlení vzorkovací frekvence občas přesáhne hranici $48kHz$, která je ve VLC maximální. VLC potom takový úsek zvuku přehraje pomaleji. Řešením by bylo podvzorkovat tento signál na nižší vzorkovací frekvenci.

6.3 Vytížení systému

V tabulce 6.1 uvádím průměrné vytížení procesoru při přehrávání různých druhů médií. Při pokusech s videem byl obraz vždy plynulý. Jedinou výjimkou byl formát WMV u kterého

obraz vři vyšším vytížení procesoru zamrzal. VLC kvůli synchronizaci vynechává zpožděné zvukové rámce a proto při použití složitějších metod detekce základního tónu přestal zvuk fungovat.

Druh média	**Metoda detekce základního tónu**	**Průměrná zátěž**
MP3	Žádná	3%
MP3	OLA	8%
MP3	HPS	87%
MP3	HNM	90%
WMV	Žádná	19%
WMV	OLA	33%
WMV	HPS	60%
WMV	HNM	89%
DivX	Žádná	11%
DivX	OLA	16%
DivX	HPS	73%
DivX	HNM	87%
XviD	Žádná	9%
XviD	OLA	17%
XviD	HPS	78%
XviD	HNM	82%

Tabulka 6.1: Vytížení procesoru při přehrávání

Kapitola 7

Závěr

V této bakalářské práci jsem navázal na diplomovou práci Ing. Kováříka nazvanou Změna rychlosti řeči [7]. Použil jsem jím vytvořený ukázkový program jako knihovnu pro změnu rychlosti přehrávání v přehrávači VLC. Toho jsem dosáhl vytvořením nového modulu kategorie resampler. Mé řešení pracuje celkem spolehlivě. Z důvodu vysokých výpočetních nároků však složitější metody detekce základního tónu (HPS a HNM) způsobují zpoždění zvuku. Zpožděný zvuk není kvůli synchronizaci přehrávačem občas přehrán. Proto se vyskytují časté mezery, nebo zvuk nefunguje vůbec.

Použití metody psola jako resampleru zřejmě není nejideálnější řešení. Ideální by zřejmě bylo zapojit tuto metodu jako součást dekodéru zvuku. To by ovšem znamenalo upravit všechny dekodéry pro použití této metody. Výhodou zakomponování metody PSOLA přímo do dekodérů by byla možnost použití kvalitnějších metod detekce základního tónu bez vlivu na konečnou plynulost přehrávání. Veškerá potřebná analýza by se provedla na začátku přehrávání. To by sice znamenalo prodlevu před začátkem přehrávání, ale dosáhlo by se kvalitnějšího výstupu.

Jako další rozšíření VLC by bylo možné implementovat zpětnou vazbu mezi moduly pro zpracování zvuku a moduly pro zpracování obrazu. Díky detekci změn ve videu by bylo možné určit úseky ve kterých se nic neděje. Pokud by v té chvíli nebyl detekován ani žádný zvuk, dal by se celý úsek zrychlit nebo přeskočit.

Literatura

[1] VLC diagram. [online].
URL http://clement.stenac.org/projects/videolan/vlc.png

[2] Documentation:Hacker's Guide. [online], 2008.
URL http://wiki.videolan.org/Documentation:Hacker%27s_Guide

[3] GNU General Public License. [online], 2008.
URL http://www.gnu.org/licenses/gpl.html

[4] Miro - free, open source internet tv and video player. [online], 2008.
URL http://www.getmiro.com/

[5] VLC media player. [online], 2008.
URL http://www.videolan.org/vlc/

[6] Dutoid, T.: *An Introduction to Text-to-Speech Synthesis*. Kluwer academic Publishers, 1997, ISBN 0-7923-4498-7.

[7] Kovářík, A.: *Změna rychlosti řeči*. Diplomová práce, FIT VUT v Brně, Brno, 2007.

[8] Wikipedie: Foném. [online].
URL http://cs.wikipedia.org/wiki/Fon%C3%A9m

Dodatek A

Obsah CD

Na kompaktním disku přiloženém k práci jsou k dispozici všechny mnou upravené zdrojové kódy přehrávače VLC.

Dodatek B

Překlad VLC

Seznam všech potřebných knihoven je uveden na adrese http://wiki.videolan.org/Contrib_Status.

Pro překlad stačí pouze spustit příkazy `./configure` a `./compile`.

Mnou používaná konfigurace je následující:

```
--prefix=/usr/local
--enable-libtool
--enable-dvdread
--enable-vcdx
--enable-faad
--enable-real
--enable-flac
--enable-realrtsp
--enable-snapshot
--enable-live555
--enable-dv
--enable-theora
--enable-esd
--enable-wxwidgets
```