Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Návrh hashovacího algoritmu pro biometrický podpis uživatele

Plzeň 2020 Bc. František Pártl

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 27. dubna 2020

Bc. František Pártl

Poděkování

Rád bych poděkoval Ing. Kamilu Ekšteinovi, Ph.D. za odborné vedení, množství cenných a inspirativních rad, podnětů, doporučení, připomínek a zároveň značnou trpělivost a ochotu při poskytnutých konzultacích ke zpracování této diplomové práce. Současně bych chtěl poděkovat Mgr. Haně Pártlové za pomoc při gramatické kontrole práce. Také děkuji všem přispěvatelům, kteří se podíleli na sběru dat.

Abstract

One of the scientific research projects being solved at the Dept. of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, is focused on the development of a person identity verification system based upon a set of characteristics (a cryptographic imprint of an individual) extracted from a short audiovisual recording of his/her face. These characteristics are extracted from both the visual and the audio component of the recording separately. The goal of this thesis is to analyze the procedures and techniques of persons' imprints acquisition and to design, implement, and thoroughly test an imprint extraction technique operating on the audio track of an audiovisual recording.

Abstrakt

Jeden z vědeckovýzkumných projektů řešených na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni, je zaměřen na vývoj systému verifikace identity osob na základě množiny charakteristik (kryptografickém otisku osoby) extrahovaných z krátké audiovizuální nahrávky jejich obličeje. Tyto charakteristiky jsou extrahovány z obrazové i zvukové složku videozáznamu odděleně. Cílem této diplomové práce je analýza postupů a technik získávání otisků osob a návrh, implementace a důkladné testování techniky extrakce otisku operující nad zvukovou stopou audiovizuální nahrávky.

Obsah

1 Úvod

V současné době[1] probíhá na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni vývoj systému podepisování adresátů při přebírání poštovních zásilek. Primární data, získaná při aktu převzetí, představuje videozáznam obličeje adresáta en face (multimodální podpis osoby), kde je jím následně řečena předem definovaná věta

„Jmenuji se <jméno-a-příjmení> a přijímám tento balík.“,

kde *<jméno-a-příjmení>* označuje křestní jméno a příjmení podepisovaného adresáta. Tuto větu je ale nutné brát pouze jako ilustrativní, protože dle specifikace projektu je využití navržené technologie plánováno i v jiných zemích.

Kritickou částí zmíněného projektu je návrh a implementace algoritmu extrakce charakteristik z multimodálního podpisu, které jednoznačně popisují adresáta. Extrahované příznaky jsou označovány jako otisk osoby a v ideálním případě by měly mít vlastnosti kryptografického otisku, tj. otisk je pro stejného adresáta stejný, pro dva různé adresáty je různý, nehledě na adresátovi má vždy stejnou délku a nelze z něj získat původní video ani audio záznam, ze kterého by bylo možné určit identitu adresáta.

Tvorba popisovaného otisku je dle nynějšího postupu dekomponována do tří kroků, z nichž vychází tři dílčí otisky. U těchto dílčích otisků není nutné splnění vlastnosti jejich nonekvivalence pro dvě různé osoby. Nepopiratelnost převzetí zásilky adresátem je zaručena, pokud všechny tři dílčí otisky osoby souhlasí s dílčími otisky získanými při převzetí. První a druhý z dílčích otisků jsou extrahovány z vizuální komponenty audiovizuálního záznamu a vycházejí z obrazových a topologických vlastností obličeje adresáta. Poslední z otisků vzniká na základě zvukové složky, tj. řečového signálu.

Diplomovou práci lze rozdělit na dvě fáze, kterými jsou analýza postupů získávání dílčích otisků a návrh, implementace a testování algoritmů vytvářejících dílčí otisk ze zvukové stopy multimodálního podpisu. Algoritmy tvorby dílčích otisků z obrazové složky nejsou součástí této práce. Analýzu vizuální komponenty videozáznamu řešil v rámci výzkumného projektu Bc. Patrik Patera. Případné zmínky a odkazy na související postupy pro tvorbu dílčích otisků z obrazové složky záznamu se týkají jeho práce a jsou zde uváděny proto, aby čtenář nepostrádal podstatné souvislosti.

[1] Pojem současná doba označuje období od dubna 2019 do září 2020.

2 Rozpoznávání obličeje

Nejznámějším biometrických rysem, který lidé využívají každý den, je obličej. I když jsou všechny obličeje různé, jejich vnitrotřídní rozptyl je vlivem gestikulace či změny vizáže vysoký (sestřih vousů, účes, stárnutí, líčení a podobně). Na druhou stranu v případě jednovaječných dvojčat nebo dvojníků je mezitřídní rozptyl často nízký. Výhodou rozpoznávání na základě obličeje je možnost snímání osoby bez jejího vědomí, čehož se často využívá například na letištích. V dnešní době je rozpoznávání lidí prováděno na základě 2D, 3D nebo termografických snímků. Výjimkou není ani jejich kombinace. Protože běžné mobilní telefony, pomocí kterých budou obrazy podepisovaných osob získávány, obvykle nedisponují snímači vzdálenosti (3D snímky) ani infračerveného záření (termografické snímky), bude následující text popisovat pouze metodu rozpoznávání na základě 2D snímku.

Je vhodné říci, že v rámci projektu tvorby systému podepisování adresátů nebyla obrazová stránka úkolem autora této diplomové práce, proto popis metod používaných v této oblasti není nijak detailní.

2.1 Rozpoznávání na základě 2D snímku

Rozpoznávání osoby na základě 2D snímku jeho obličeje je nejrozšířenější a také nejjednodušší dostupnou technikou, která je ovšem kvůli snadnému získání fotografie ověřované osoby nejméně bezpečná.

Většina existujících algoritmů pracuje podle následujícího postupu:

1. **Detekce obličeje na snímku**
   Snímky obličejů jsou pořizovány při různých osvětleních, pozicích, rozměrech, orientacích, mimice obličeje apod. Podle scénáře snímání může snímek obsahovat i více obličejů. Detekce je často založena na klasifikátorech, které pro každou část obrazu určí, zda obsahuje či neobsahuje obličej. Klasifikátory bývají pro zvýšení výkonu vhodně uspořádány do kaskád. V případě, že klasifikátor kaskády nedokáže rozhodnout o absenci obličeje v části obrazu, je spuštěn následující klasifikátor. Obraz tedy obsahuje obličej právě, když ani jeden z klasifikátorů nerozhodne o opaku. Urychlení tohoto řešení plyne z předpokladu, že většina plochy snímku je tvořena pozadím, a vyhodnocení kaskády tak končí obvykle již po první klasifikaci.

2. **Normalizace detekovaného obličeje**
   Dalším krokem je vhodné předzpracování částí obrazu obsahujících obličej. Součástí normalizace je změna měřítka (porovnávané snímky obličejů mají tak stejné rozměry), extrakce obličeje (nahrazení pozadí jednotnou barvou), jasová kompenzace atd.

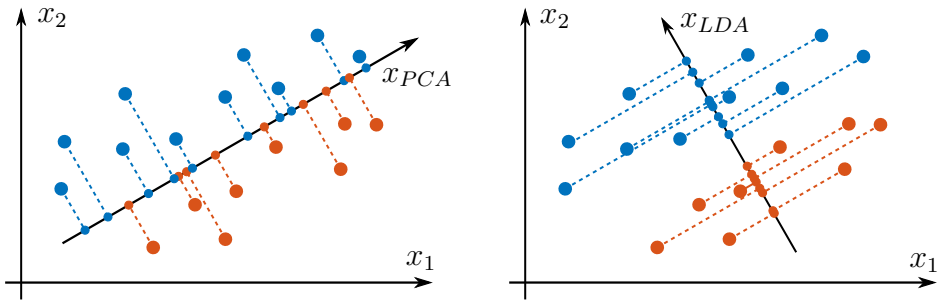
3. **Rozpoznávání osoby**
   Způsoby rozpoznávání osob podle obličejů jsou založeny na podobnosti obrazových bodů nebo na topologických vlastnostech. Způsoby jsou blíže popsány v sekcích 2.1.1 a 2.1.2.

2.1.1 Rozpoznávání podle obrazových dat

Metody rozpoznávání podle obrazových dat jsou obecně použitelné pro určení podobnosti dvou různých obrazů. Z této obecnosti vyplývá, že normalizace velikosti a odstranění pozadí je u obrazů obličejů nezbytné. Úrovně jasů jednotlivých pixelů snímku, které lze vyjádřit jako matici $S \in \mathbb{N}_0^{w \times h}$, kde w, resp. h označuje šířku, resp. výšku snímku v pixelech. Poskládáním jednotlivých řádků za sebe vzniká vektor $x \in \mathbb{N}_0^n$, kde $n = w \cdot h$. Tento vektor je možné chápat jako bod v n-dimenzionálním prostoru. Úkolem dále popsaných metod je separace bodů reprezentujících stejnou osobu od ostatních za účelem jejich rozlišení při klasifikaci. Zásadní nevýhodou metod v uvažovaném scénáři tvorby otisku je nutnost jejich trénování. Při přidání nové osoby by tak bylo nutné celý systém natrénovat na nových snímcích.

Metoda hlavních komponent (PCA)

Metoda hlavních komponent, anglicky *Principal Component Analysis (PCA)*, spočívá v transformaci n-rozměrného prostoru obličejů do jiného prostoru, jehož bázové vektory jsou seřazeny podle velikosti rozptylu dat. S minimální ztrátou informace lze tak některé dimenze zanedbat a tím prostor redukovat. Jedná se o techniku učení bez učitele, což znamená, že učení je možné bez znalosti příslušnosti snímku k osobě. U m vektorů x_i představujících snímky obličejů je nejdříve vypočten průměrný vektor $\overline{x}$, který je od každého bodu x_i odečten. Tím je zaručena nulová střední hodnota bodů x_i. Následuje výpočet vlastních vektorů SVD rozkladem kovarianční matice $\Sigma = \frac{1}{m} X^T X$, kde X je tzv. *datová matice*, jejímiž řádky je m vektorů x_i. Vlastní vektory neboli *vlastní obličeje* (*eigenfaces*) představují bázové vektory transformovaného prostoru. Zanedbáním části vektorů, tj. snížením dimenze transformovaného prostoru, vzniká *prostor vlastních obličejů*. Původní obrazy lze rekonstruovat lineární kombinací vlastních obličejů a průměrného vektoru



(a) Metoda hlavních komponent. (b) Lineární diskriminační analýza.

Obrázek 2.1: Redukce příznakového prostoru technikami PCA a LDA. Barvy odlišují řečníky, velikost pak příslušnost k prostoru. Rozložení bodů je pro PCA úmyslně voleno nevhodně. Naopak u LDA je v redukovaném prostoru x_{LDA} patrná separace bodů.

$\overline{x}$. Pro vyhodnocení podobnosti dvou obrazů je nutný jejich převod do prostoru vlastních obličejů. Vzdálenost mezi vzniklými vektory je vypočtena pomocí eukleidovské nebo Mahalanobisovy vzdálenosti. Obrázek 2.1a ukazuje aplikaci PCA na redukci $\mathbb{R}^2 \rightarrow \mathbb{R}$ při nevhodném rozložení bodů, kdy osoby jsou po projekci obrazů do prostoru obličejů špatně rozlišitelné.

Lineární diskriminační analýza (LDA)

Oproti PCA je lineární diskriminační analýza, anglicky *Linear Discriminant Analysis (LDA)*, technikou učení s učitelem, tj. u každého obrazu je známa identita osoby. Redukovaný prostor vzniká tak, aby došlo k co nejlepší separaci obrazů reprezentujících stejnou osobu. Situaci demonstruje obrázek 2.1b. Aplikace LDA na snímky obličejů se označuje jako *fisherfaces*.

2.1.2 Rozpoznávání podle topologických vlastností

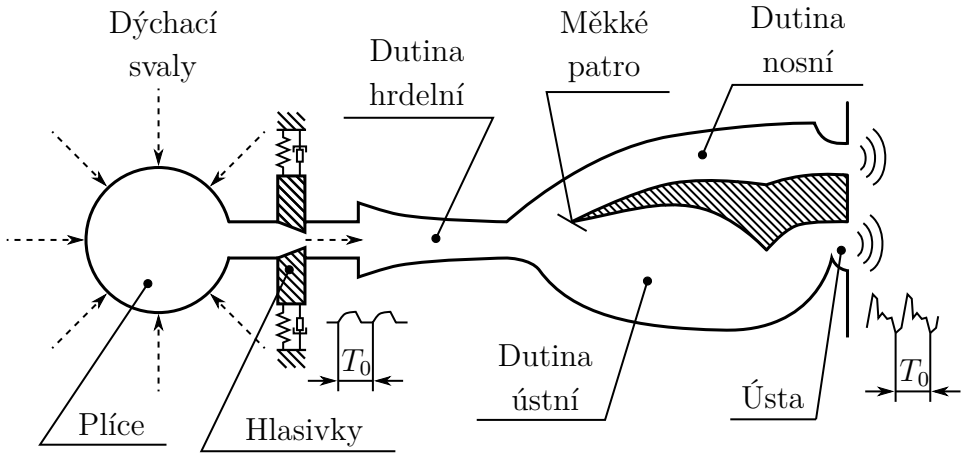
Na rozdíl od metod popisovaných v sekci 2.1.1 tyto metody snímek osoby nejprve interpretují jako obličej a pak hledají podobnosti. Základem metod jsou aktivní šablony, které se skládají obvykle z několika desítek obličejově významných bodů, tj. oči, koutky úst, nos, obočí apod. Pozice těchto bodů je ve fázi trénování nutné ručně určit. Při rozpoznávání osoby na neznámém snímku se vytvořené šablony na obraz iterativně mapují. V praxi se využívají metody *ASM (Active Shape Model)* a *AAM (Active Appearance Model)*. Zatímco šablona ASM obsahuje pouze informaci o významných bodech (tvaru šablony), AAM přidává i údaj o textuře, díky čemuž obvykle dosahuje lepších výsledků [20].

3 Biometrie hlasu

Řeč je u člověka přirozenou a také nejčastěji využívanou formou komunikace. Lidé si schopnost jejího vytváření a vnímání osvojují již během útlého dětství. Mluvená řeč je přenášena komunikačním kanálem ve formě akustických vln, tzv. akustického signálu. Ten je založen na principu vlnění elastického prostředí v oblasti slyšitelných frekvencí, které u člověka přibližně náleží intervalu od 20 do 20 000 Hz. Řečový signál kromě lingvistické informace, vyjadřující význam sdělované myšlenky, obsahuje jisté specifické charakteristiky mluvčího. Jejich původ je těsně svázán s vlastnostmi hlasového traktu řečníka a způsobu artikulace (intonace, rytmus řeči, barva hlasu apod.) [26]. Aby tedy bylo možné tyto charakteristické rysy identifikovat, je třeba popsat postup produkce řeči.

3.1 Proces vytváření řeči člověkem

Za účelem tvorby řeči existuje v lidském těle skupina orgánů tvořící tzv. *hlasový trakt*, které souhrnně nazýváme *řečové orgány* nebo také *mluvidla*. Oproti uchu, který je orgánem primárně určeným pro slyšení, není produkce řeči hlavním úkolem těchto orgánů – jedná se o dýchání, příjem potravy apod. Hlasový trakt je popsán na frontálním řezu F.1 přílohy dokumentu, zjednodušený model pak na obrázku 3.1. Trakt je rozdělen na tři základní ústrojí: dýchací, hlasové a artikulační, která jsou blíže popsána v dalších oddílech textu [26].



Obrázek 3.1: Zjednodušený fyziologický model produkce řeči.

3.1.1 Dechové ústrojí

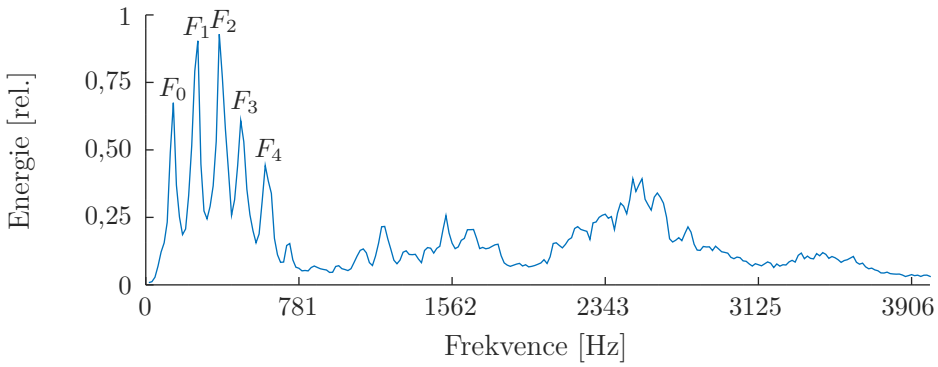
Dechové ústrojí, přesněji dolní dýchací cesty, tvoří fundamentální zdroj energie pro tvorbu hlasu. Rozumíme jím průdušnici (*trachea*), průdušinky (*bronchus principalis*), plíce (*pulmo*) a dechové svaly – vnitřní a vnější svaly mezižeberní (*musculi intercostales*) a bránici (*diaphragma*). Tyto orgány jsou uloženy v dutině hrudní (*cavum thoracis*). Mechanika dýchání, která je úzce spjata se zmíněnou tvorbou energie, je rozdělena následovně.

1. **Vdech (*inspirium*)**
   Stahem bránice a zevních svalů mezižeberních je zvětšen objem hrudní dutiny, čímž je vlivem negativního tlaku v interpleurální štěrbině způsobeno roztažení plic. V rozpjatých plicích tak vzniká pokles tlaku na hodnou nižší, než je tlak atmosferický, a do plic tak proudí vzduch. Vdech je díky činnosti dýchacích svalů označován jako aktivní děj. Tento nashromážděný vzduch (tlakový gradient) je zdrojem energie pro tvorbu řeči.

2. **Výdech (*exspirium*)**
   Ochabnutím dýchacích svalů se díky elastickým vlastnostem zmenší objem dutiny hrudní, což vede ke smrštění plic. V plicích tak dochází ke zvýšení tlaku vzduchu na hodnotu vyšší, než je tlak atmosferický, a vzduch na základě tlakového gradientu proudí z plic do vnějšího prostředí. Při výdechu tak vzniká proud vzduchu, který je fundamentálním zdrojem pro tvorbu energie. Za účelem zvýšení amplitudy řečového signálu je vzduch z plic vytlačován aktivně pomocí vnitřních mezižeberních svalů. Vitální kapacita plic, tj. objem vzduchu, který po maximálním nádechu značným úsilím vydechneme, je u dospělého člověka přibližně 5 litrů [28]. Při běžné řeči se spotřebuje až polovina tohoto objemu, zatímco při velmi hlasité řeči či zpěvu až 80 % [26].

3.1.2 Hlasové ústrojí

Hlasové ústrojí je částí hlasového traktu kde dochází ke vzniku hlasu. Je uloženo v hrtanu (*larynx*), který navazuje na průdušnici. Nejdůležitější část hlasového ústrojí představují hlasivky (*plicae vocales*). Tyto dvě ostré slizniční řasy vedou napříč hrtanem v místě jeho nejužšího průchodu. Prostor mezi hlasivkami je označován jako hlasivková štěrbina (*glottis*). Pokud člověk mlčí, hlasivkové svaly drží štěrbinu odkrytou, a vzduch určený k dýchání tak prochází bez odporu. Tento stav je označován jako klidové postavení hlasivek. Při hlasovém (fonačním) postavení hlasivky se dýchací cesty plně uzavřou.



Obrázek 3.2: Odhad výkonové spektrální hustoty znělé hlásky.

Staženými hlasivkami je z plic vytlačován vzduch, který způsobuje jejich kmitání. Hlasivky se střídavě postupně otevírají a prudce uzavírají, čímž vznikají vzduchové vlny, které vnímáme jako zvuk označovaný termínem *základní (hlasivkový) tón*. Jeho frekvence se označuje F_0 a odpovídá výšce hlasu, kterou vnímá posluchač. Rozsah F_0 náleží do intervalu $\langle 60, 400 \rangle$ Hz, přičemž u sopranistek není výjimkou F_0 vyšší než 1000 Hz.

3.1.3 Artikulační ústrojí

Posledním ústrojím, které se podílí na tvorbě řečového signálu, je artikulační ústrojí. Ve smyslu možnosti pohybu jsou jeho (i) pasivními prvky nadhrtanové dutiny (hrdelní, ústní a nosní). Při průchodu základního hlasivkového tónu dutinami dochází k soustředění energie kolem jejich rezonančních frekvencí, jejichž hodnota se liší podle řečníka. Rezonanční frekvence se nazývají *formanty* a označují se $F_1, F_2, ..., F_n$. Přibližně platí, že F_1 odpovídá rezonanční frekvenci dutiny hrdelní, F_2 dutiny ústní a F_3 dutiny nosní [20]. Ukázku formantové struktury naleznete na obrázku 3.2. (ii) Aktivními prvky jsou tzv. artikulátory, které svým pohybem mění velikost nadhrtanových dutin. Mezi ně patří jazyk (*lingua*), rty (*labia oris*), měkké patro (*palatum molle*) a také hrtan.

Podle toho, jestli zvuk vzniká v hlasovém, resp. artikulačním ústrojí, rozlišujeme tzv. znělé, resp. neznělé zvuky. Znělé zvuky, tj. samohlásky a znělé souhlásky, vznikají za fonačního postavení hlasivek a při jejich frekvenční analýze je zřetelná formantová struktura (podobně jako na obrázku 3.2). Neznělé zvuky vznikají až různým postavením artikulátorů v artikulačním ústrojí a hlasivky jsou v klidovém postavení. Jedná se o neznělé souhlásky, jejichž frekvenční spektrum má povahu šumu. Základní hlasivkový tón, či formantová struktura se zde nevyskytují [26].

4 Analýza řečového signálu

Zvukové stopy multimodálních podpisů adresátů byly zpracovávány jak v časové, tak frekvenční oblasti. Pro finální parametrizaci byla využita technika mel-frekvenčních kepstrálních koeficientů, která je popsána v sekci 4.3.

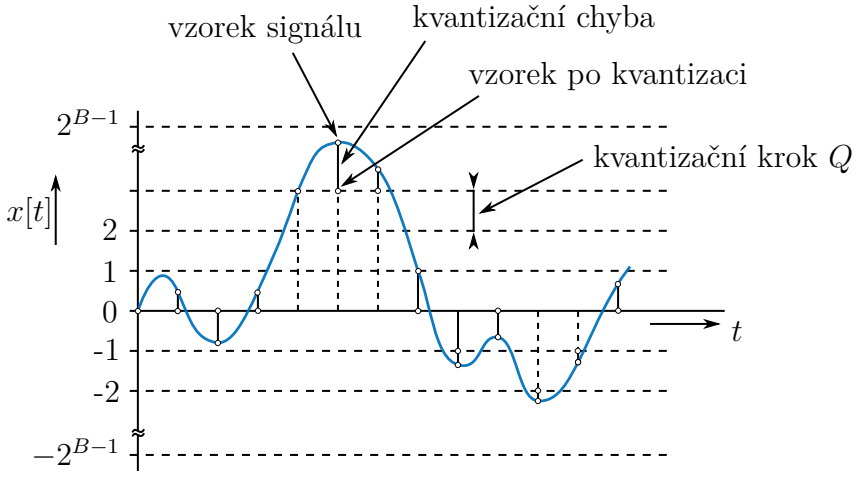
4.1 Pořízení signálu

Zvukové kmity řečového signálu podepisovaných řečníků jsou standardně snímány mikrofonem. Konkrétně byly v rámci této diplomové práce využity výhradně mikrofony mobilních telefonů. Výstupem mikrofonu je analogový spojitý signál, který je pro další zpracování nutné převést do číslicového. To znamená vytvořit takovou reprezentaci, která je posloupností čísel náležících do konečné množiny hodnot. Tato metoda je obecně nazývána jako *pulsní kódová modulace (PCM)* nebo také *digitalizace*. Je zobrazen na obrázku 4.1 a zahrnuje následující tři kroky.

- **Vzorkování**
  Představuje transformaci spojitého signálu $x(t)$ na posloupnost vzorků $x_n = x(iT)$ diskrétních v čase. iT označuje okamžiky vzorkování s periodou T a $i \in \mathbb{N}_0$. Vzorkovací frekvence $f_v = \frac{1}{T}$ je omezena zdola Shannonovým-Nyquistovým-Kotělnikovovým vzorkovacím teorémem, který říká, že rekonstrukce původního signálu z jeho vzorků je možná pouze tehdy, pokud byl vzorkován s $f_v \geq 2f_m$, kde f_m označuje maximální frekvenci obsaženou ve vzorkovaném signálu.

- **Kvantizace a kódování**
  Vzorkovaný signál je diskrétní v čase, ale hodnoty jednotlivých vzorků x_n náleží množině $\mathbb{R}$. Kvantizace je aproximací každého z těchto vzorků jednou z konečného počtu číselných hodnot. Situaci ilustruje obrázek 4.1. Kvantizér, tj. zařízení provádějící kvantizaci, podle zadané spojité úrovně signálu, počtu úrovní kvantování a kvantizačního kroku Q generuje odpovídající diskrétní kódovou reprezentaci. Počet úrovní se obvykle volí ve tvaru 2^B, kde B je počet bitů binární reprezentace čísla. Kvantizační krok Q je pak odvozen podle zvoleného počtu úrovní a maximální úrovně vzorkovaného signálu S_{max}, tedy podle [26] platí

$$2S_{max} = 2^B Q \implies Q = \frac{2S_{max}}{2^B}.$$



Obrázek 4.1: Ukázka procesu vzorkování a kvantizace.

4.2 Předzpracování

4.2.1 Normalizace střední hodnoty

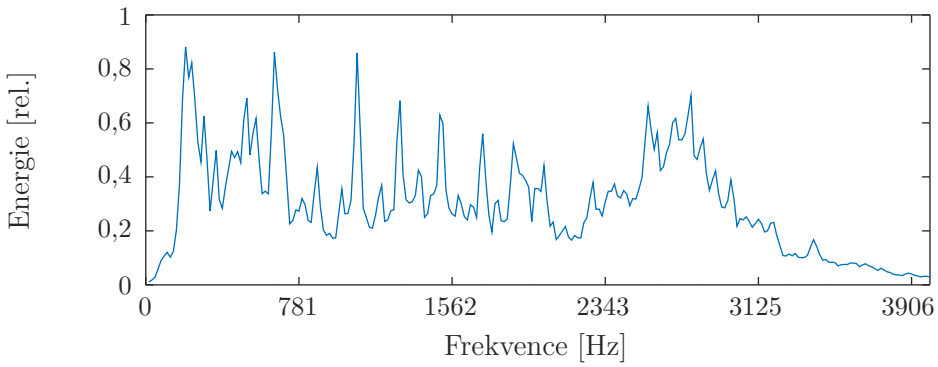
Z fyzikální podstaty akustického signálu vyplývá, že jeho dlouhodobá střední hodnota by měla být nulová. Při neplatnosti tohoto předpokladu můžeme říci, že je signál posunut v hodnotě o stejnosměrnou složku napětí, tedy obsahuje tzv. *DC ofset*. Nejčastější příčinou tohoto jevu je nekvalitní nebo špatně pracující zařízení nahrávacího řetězce [2]. Velikost ofsetu odpovídá střední hodnotě signálu $\overline{x}$. Primitivní technika normalizace střední hodnoty je popsána vztahem

$$y_i = x_i - \overline{x} \ , \qquad \text{pro } \ i \in \{0, 1, ..., I-1\},$$

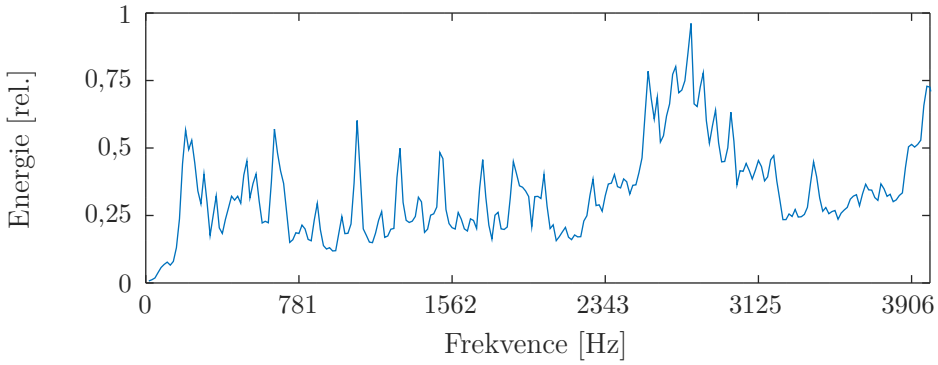
kde x_i představuje vzorek originální posloupnosti s délkou I vzorků a y_i je vzorek signálu zbavený DC ofsetu [21].

4.2.2 Preemfáze

Vlastností lidského hlasového ústrojí je potlačování amplitud vyšších spektrálních složek výstupního řečového signálu. Toto chování platí zejména u znělých fonémů. Obdobnou vlastnost má i lidský sluch, jehož citlivost se vzrůstající frekvencí klesá. Preemfáze, nebo také preakcent, je proces, který tento pokles energií kompenzuje, čímž dojde k relativnímu vyrovnání výkonového spektra řečového signálu. Efekt preemfáze na odhad výkonové spektrální hustoty je ilustrován na obrázku 4.2, kde je patrná eliminace klesající tendence energie s rostoucí frekvencí.



(a) Před aplikací preemfáze.



(b) Po aplikaci preemfáze.

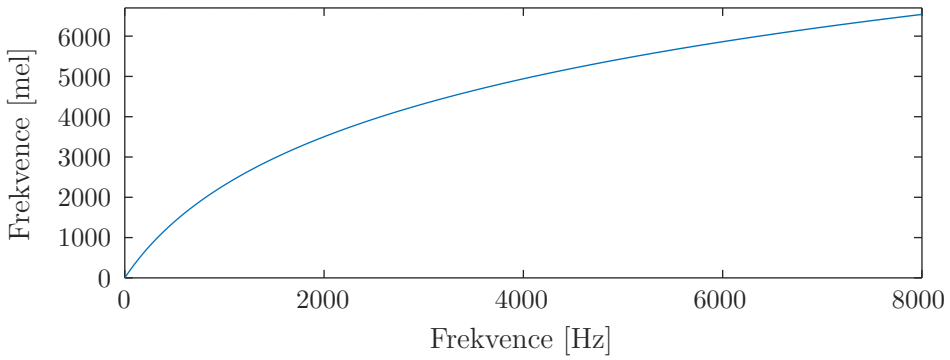
Obrázek 4.2: Efekt procesu preemfáze na odhad výkonové spektrální hustoty segmentu lidské řeči při $\alpha = 0{,}97$.

Preemfáze bývá realizována následujícími dvěma způsoby:

- **analogový filtr**, zařazený ještě před vzorkování a kvantizaci signálu,
- **číslicový filtr**, aplikovaný až na kvantizovaný signál, provádějící předzpracování podle následujícího výrazu

$$z_i = y_i - \alpha y_{i-1} \ , \qquad \text{pro } \ i \in \{1, 2, ..., I-1\},$$

kde y_i je vstupní vzorek z původního signálu s délkou I vzorků, z_i je výstupem filtru a parametr $\alpha \in \langle 0{,}9; 1\rangle$ je konstantní [26]. V implementovaných algoritmech a technikách této diplomové práce byla využita hodnota $\alpha = 0{,}97$.



Obrázek 4.3: Vyjádření nelineární závislosti melovské škály na frekvenci.

4.3 MFCC

Jednou z experimentálně potvrzených vlastností lidského sluchu je nelineární vnímání frekvencí. Parametrizace pomocí mel-frekvenčních kepstrálních koeficientů využívá banky pásmových filtrů s lineárním rozložením frekvencí v tzv. *melovské frekvenční škále*. Ta svým logaritmickým tvarem napodobuje vnímání zvuku lidských sluchem a je definována vztahem

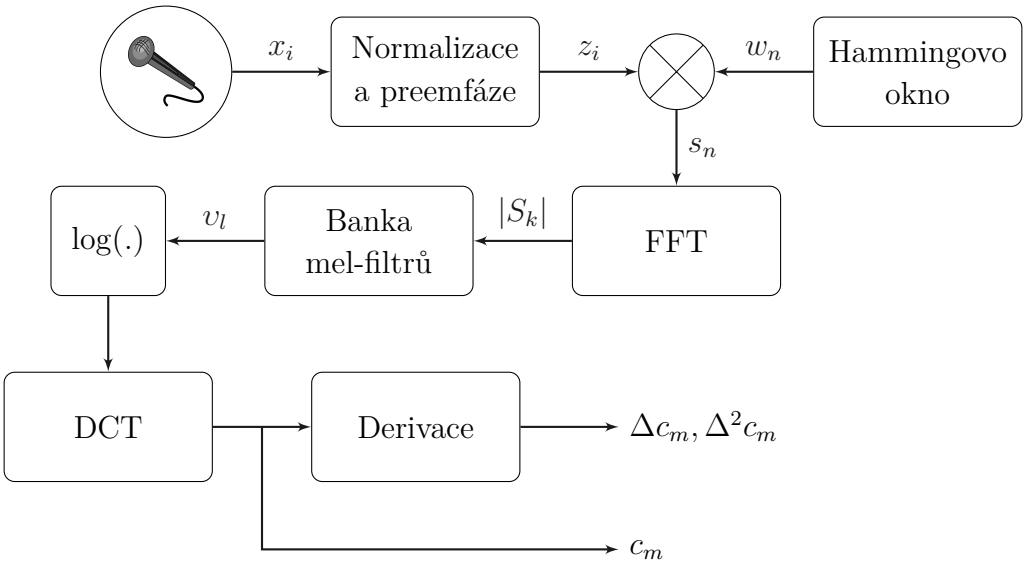
$$f_m(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right),$$

kde f je frekvence v lineární škále a f_m je odpovídající frekvence v melovské škále [26]. Logaritmická křivka definovaná tímto vztahem je zobrazena na obrázku 4.3. MFCC tedy popisuje spektrální vlastnosti parametrizovaného signálu, konkrétně odhad jeho výkonové spektrální hustoty. Celý výpočet koeficientů je popsán schématem 4.4. Konkrétní části schématu jsou popsány v následujících sekcích textu.

Implementace výpočtu MFCC koeficientů byla integrována do knihovny `libpe` a popsána v sekci 7.3.6.

4.3.1 Rozdělení signálu do segmentů

Řečový signál je v čase velice proměnlivý. Z fyziologického pohledu na jeho tvorbu ovšem vyplývá, že v krátkých segmentech se již stává kvazistacionárním a jeho spektrální analýza má smysl. Tento jev je způsoben zejména dynamickými vlastnostmi orgánů podílejících se na tvorbě řeči, které mají díky své hmotnosti omezenou rychlost změny polohy [26]. Počet vzorků jednotlivých segmentů řečového signálu je dán vztahem $N = \tau f_s$, kde f_s značí frekvenci vzorkování a τ označuje délku segmentu v sekundách. Hodnota



Obrázek 4.4: Schéma výpočtu mel-frekvenčních kepstrálních koeficientů.

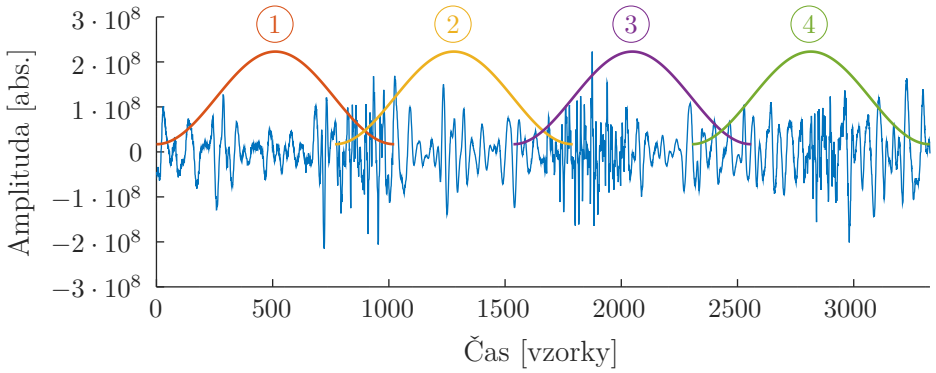
τ se pro úlohy popisu mluvčího obvykle pohybuje v rozmezí od 20 do 64 ms. Při $f_s = 8000$ Hz tedy doporučený počet vzorků segmentů náleží do intervalu $\langle 160, 512\rangle \subset \mathbb{N}$ [21]. Pro takto vytvořené segmenty probíhá další výpočet MFCC koeficientů odděleně. Obrázek 4.5 ilustruje rozdělení signálu na segmenty o velikosti $N = 1024$ vzorků a 25% překryvu.

4.3.2 Váhování segmentů

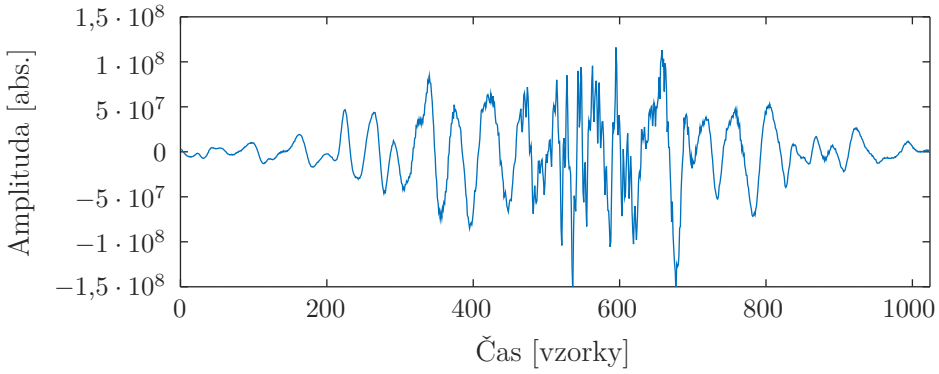
Segmenty vzniklé rozdělením původní posloupnosti vzorků $s[i]$ obecně nesplňují vlastnost nulové střední hodnoty akustického signálu. Dalším problémem je potenciální existence bodů nespojitosti prvního druhu na krajích segmentů. Díky těmto skokům se pak mohou v následující spektrální analýze objevit energie ve vysokých frekvencích, které původní signál neobsahuje. Toto chování je úzce spjato s Gibbsovým jevem [30]. Za účelem snížení váhy okrajových vzorků ve výpočtu jsou na segment aplikována tzv. *váhovací okna* dle vztahu

$$s_n = z_n w_n \ , \qquad \text{pro } \ n \in \{0, 1, ..., N-1\},$$

kde z označuje segment signálu, w použité váhovací okno a s nově vzniklý váhovaný segment.



Obrázek 4.5: Aplikace Hammingova okna na řečový signál.



Obrázek 4.6: Segment váhovaný Hammingovým oknem.

Jedním z nejčastěji využívaných druhů oken je *Hammingovo*, které je definováno vztahem

$$w_n = 0{,}54 - 0{,}46 \cos\left(\frac{2\pi n}{N-1}\right), \qquad \text{pro } n \in \{0, 1, ..., N-1\}, \tag{4.1}$$

kde N je počet vzorků segmentu, w představuje posloupnost vzorků okna délky N. Na obrázku 4.5 jsou vyobrazena čtyři Hammingova okénka. Výsledný váhovaný segment, který odpovídá 4. oknu z obrázku 4.5, je zobrazen na obrázku 4.6.

4.3.3 Diskrétní Fourierova transformace

Po získání váhovaných segmentů řečového signálu je dalším krokem výpočet odhadů jejich výkonových spektrálních hustot. K tomuto účelu se obvykle využívá rodiny matematických technik zvaných *Fourierovská analýza*, jejíž myšlenkou je rozložení průběhu signálu na součet nekonečné řady sinových průběhů o různé frekvenci, fázi a amplitudě. Jedním z členů této rodiny je

tzv. *diskrétní Frourierova transformace (DFT)*, která je definována jako

$$S_k = \sum_{n=0}^{N-1} s_n e^{-\frac{i2\pi nk}{N}} \quad , \qquad \text{pro } \; k \in \{0, 1, ..., N-1\}, \tag{4.2}$$

kde s je analyzovaný váhovaný segment a N značí počet vzorků segmentu. Výsledkem DFT je vektor komplexních čísel zvaný diskrétní odhad spektra [30]. Pokud uvažujeme vzorkovací frekvenci $f_s = 8000$ Hz, pak můžeme podle Nyquistova vzorkovacího teorému říci, že maximální možná frekvence obsažená v rekonstruovaném signálu bude $f_{max} = 4000$ Hz. Teorém je blíže diskutován v sekci 4.1. Tento spojitý rozsah frekvencí $\langle 0, 4000 \rangle$ Hz je následně popsán $N = 512$ vzorky diskrétního odhadu spektra. Každý z nich tedy popisuje energie v pásmech frekvencí s šířkou

$$\frac{f_{max}}{N} = \frac{4000}{512} = 7{,}8125 \text{ Hz}.$$

Podle jednoho z Eulerových vzorců, který je popsán vztahem (4.3) lze DFT přepsat do podoby (4.4) [17].

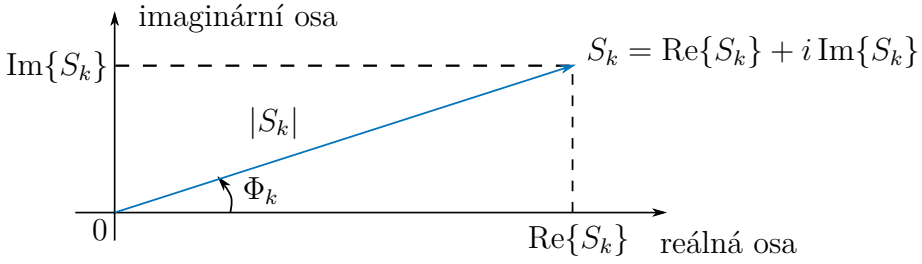
$$e^{i\varphi} = \cos\varphi + i \sin\varphi \tag{4.3}$$

$$S_k = \sum_{n=0}^{N-1} s_n \left[\cos\left(\frac{2\pi nk}{N}\right) - i \sin\left(\frac{2\pi nk}{N}\right) \right] \tag{4.4}$$

Vzorek diskrétního odhadu spektra má podobu $S_k = \mathrm{Re}\{S_k\} + i\,\mathrm{Im}\{S_k\}$, kde $\mathrm{Re}\{S_k\}$, resp. $\mathrm{Im}\{S_k\}$ je reálná, resp. imaginární složka komplexního čísla S_k. Číslo lze vizualizovat v Gaussově rovině, což ilustruje obrázek 4.7. Tato vizualizace ukazuje další možnou reprezentaci vzorku pomocí polárních souřadnic. Velikost $|S_k|$ je označována jako absolutní hodnota nebo také magnituda komplexního čísla. Při zachování původní myšlenky rozkladu signálu na sinové funkce představuje magnituda $|S_k|$ odhad amplitudy a úhel ϕ_k odhad fázového posunu sinového průběhu ve frekvenčním pásmu k. Výpočet magnitudy je možné provést pomocí Pythagorovy věty dle vztahu (4.5). Fázový posun je určen goniometrickou funkcí arctan podle výrazu (4.6). Vektor absolutních hodnot komplexních koeficientů je označován jako odhad výkonové spektrální hustoty segmentu a v textu dále zkráceně nazýván pojmem odhad spektra.

$$|S_k| = \sqrt{\mathrm{Re}\{S_k\}^2 + \mathrm{Im}\{S_k\}^2} \tag{4.5}$$

$$\phi_k = \arctan\left(\frac{\mathrm{Im}\{S_k\}}{\mathrm{Re}\{S_k\}}\right) \tag{4.6}$$



Obrázek 4.7: Ukázka magnitudy a fáze vzorku komplexního spektra S_k.

Rychlá Fourierova transformace

Ze vztahu (4.2) pro výpočet diskrétní Fourierovy transformace vyplývá složitost $\mathcal{O}(N^2)$. V roce 1965 pánové James Cooley a John Tukey představili urychlení algoritmu označené jako rychlá Fourierova transformace, anglicky *Fast Fourier Transform (FFT)*, jehož složitost je $\mathcal{O}(N \log_2 N)$ [19]. Jediným omezení algoritmu je, aby velikost analyzovaných segmentů obsahovala právě $N = 2^m$ vzorků, kde $m \in \mathbb{N}$. V dnešní době se jedná o nejvíce využívaný algoritmus v oblasti získávání komplexních spekter segmentů signálu. V této diplomové práci je využívána implementace tohoto algoritmu v podobě knihovny *KissFFT* [5], která pro zadané segmenty akustického signálu délky N vzorků vrací jejich diskrétní odhady spekter délky $\frac{N}{2} + 1$ vzorků.

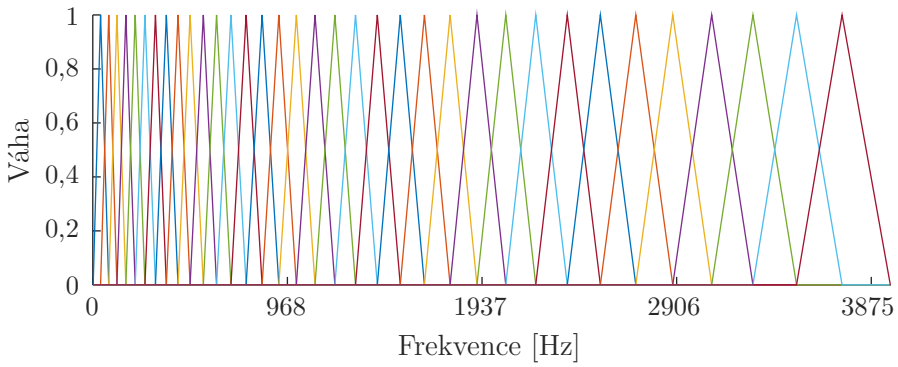
4.3.4 Výpočet melovských koeficientů

Podle diagramu 4.4 je dalším krokem výpočet tzv. melovských koeficientů. Tyto koeficienty vznikají skalárním součinem odhadu výkonové spektrální hustoty segmentu a L trojúhelníkových filtrů melovské banky, tedy

$$v_l = \sum_{k=0}^{N-1} |S_k| H_{l,k} \ , \qquad \text{pro } \ l \in \{0, 1, ..., L-1\},$$

kde $|S_k|$ je hodnota k-tého vzorku odhadu výkonové spektrální hustoty segmentu a $H_{l,k}$ je k-tý vzorek l-tého trojúhelníkového filtru melovské banky. Výsledkem je l-tý melovský koeficient v_l.

Množina trojúhelníkových filtrů je konstruována v melovské frekvenční škále, která je blíže popsána v úvodu sekce 4.3. Celá škála je rovnoměrně pokryta L trojúhelníkovými filtry jednotkové výšky, které se právě z 50 % překrývají, tj. střední frekvence filtru H_l je zároveň počátečním bodem trojúhelníku H_{l+1}. Pro výpočet odezev filtrů je třeba přepočítat všechny koeficienty DFT do melovské frekvenční škály. Jiným a výpočetně mnohem výhodnějším řešením je vyjádření filtrů banky v lineární frekvenční škále.



Obrázek 4.8: Množina 35 trojúhelníkových filtrů melovské banky.

Inverzní přepočet probíhá dle vztahu

$$f(f_m) = 700\left(\exp\left(0{,}887 \cdot 10^{-3} f_m\right) - 1\right),$$

kde f_m je frekvence v melovské škále a f v lineární [26]. Pro $L = 35$ trojúhelníkových filtrů je výsledkem banka zobrazená na obrázku 4.8, kde je na rozměrech filtrů vidět logaritmické rozložení melovské škály.

4.3.5 Převod na kepstrální koeficienty

Předpokládejme, že původní signál v časové oblasti je dán konvolucí užitečného a šumového signálu. Ve frekvenční oblasti je tedy vyjádřen součinem obrazů obou složek. Aplikací logaritmu na melovské koeficienty v_l lze pak tyto složky oddělit. Dalším efektem logaritmu, který je opět společný s lidským uchem, je omezení dynamiky signálu. Posledním krokem při výpočtu mel-frekvenčních kepstrálních koeficientů je výpočet inverze DFT (IDFT). Vzhledem k tomu, že vstupem této transformace je množina nezáporných koeficientů, lze IDFT redukovat na diskrétní kosinovou transformaci (DCT) danou výrazem

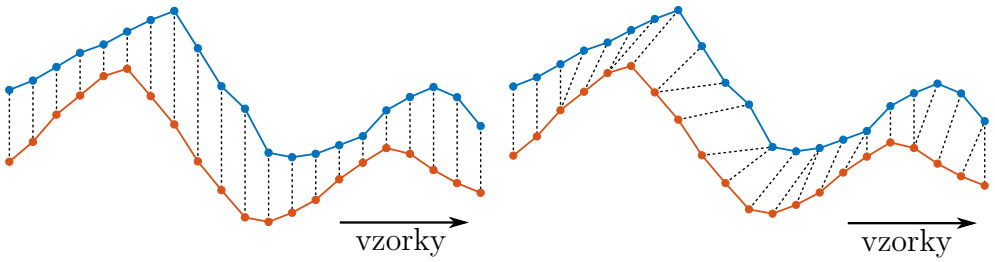
$$c_m = \sum_{l=0}^{L-1} \log(v_l) \cos\left[\frac{\pi m}{L}\left(l - \frac{1}{2}\right)\right], \qquad \text{pro } m \in \{0, 1, ..., M-1\},$$

kde L je počet melovských koeficientů v_l a $M \in \{1, 2, ..., L\}$ představuje počet výsledných kepstrálních koeficientů c_m. Výsledné koeficienty ukazuje graf 9.4, na kterém je první z koeficientů záměrně vypuštěn.

5 Použité techniky

5.1 Dynamické ohýbání času (DTW)

Algoritmus dynamického ohýbání času, anglicky *Dynamic Time Warping (DTW)*, byl poprvé představen v 60. letech minulého století a od té doby nalezl uplatnění v desítkách odvětví. Jedná se o efektivní metodu měření podobnosti časových řad různé délky, která dovoluje „elasticky“ transformovat časovou osu. Díky tomu metoda dokáže detekovat podobnosti signálů i přes jejich zkreslení v čase nebo fázový posun. Příkladem popsané situace je obrázek 5.1.



(a) Eukleidovská vzdálenost. (b) Dynamické ohýbání času.

Obrázek 5.1: Signály porovnávané pomocí eukleidovské vzdálenosti a DTW.

Vstupem algoritmu jsou porovnávané signály $X = (x_1, x_2, ..., x_N)$ a $Y = (y_1, y_2, ..., y_M)$, kde $N, M \in \mathbb{N}$. Prvním krokem je konstrukce tzv. *cenové matice* $D \in \mathbb{R}^{M \times N}$. Pro její konstrukci je vhodné označit vzdálenost dvou funkčních hodnot časových řad jako $d(x, y) = |x - y|$. Prvky matice D jsou pak definovány pro:

- první řádek: $D(1, j) = \sum_{k=1}^{j} d(x_1, y_k)$, kde $j \in \{1, 2, ..., M\}$,
- druhý řádek: $D(i, 1) = \sum_{k=1}^{i} d(x_k, y_1)$, kde $i \in \{1, 2, ..., N\}$
- a ostatní: $D(i, j) = \min \begin{Bmatrix} d(i-1, j-1) \\ d(i-1, j) \\ d(i, j-1) \end{Bmatrix} + d(x_i, y_j)$, kde $i \in \{2, ..., N\}$ a $j \in \{2, ..., M\}$.

Hodnota $D(N, M)$ je pak tzv. *cenou borcení*, která vyjadřuje podobnost signálů. Hodnotu nelze využít pro určování vzdálenosti signálů, protože by nesplňovala všechny axiomy definující metriku. Z cenové matice D je možné

extrahovat i další informace, které jsou diskutovány v článku [29].

Výpočet ceny borcení pomocí algoritmu DTW byl v rámci diplomové práce implementován v jazyce C++ a integrován do knihovny `libpe`. Popis implementace je k dispozici v sekci 7.3.9.

5.2 Shluková analýza (K-means)

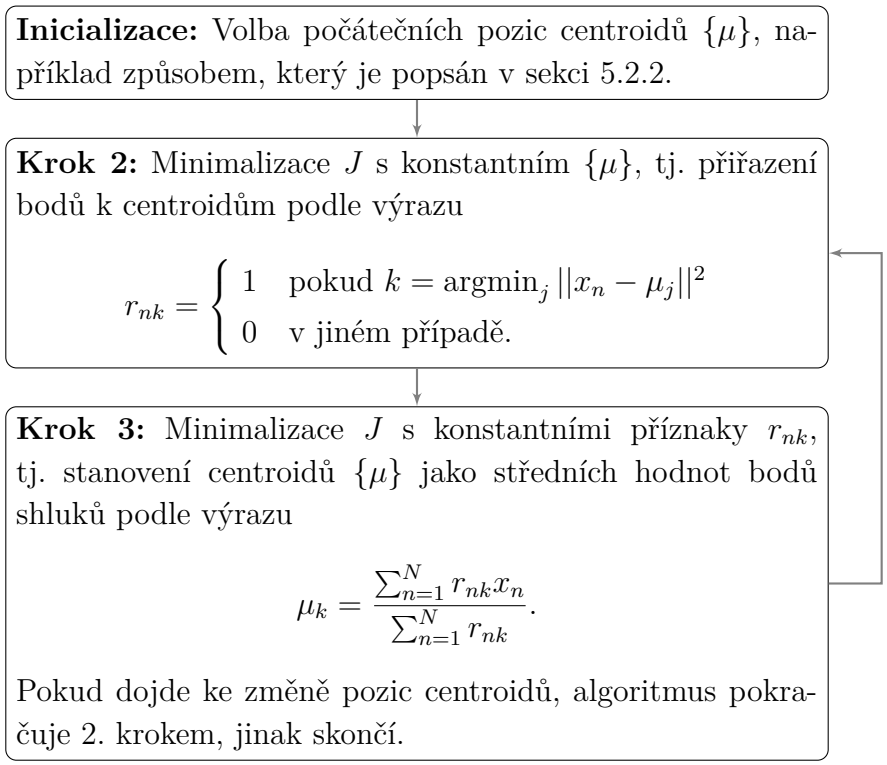
Jedno z navržených řešení tvorby otisku z promluvy řečníka je založeno na předpokladu, že příznaky extrahované z akustického signálu tvoří v příznakovém prostoru shluky, jejichž polohy odrážejí charakteristiky mluvčího, a mohou tak tvořit jeho otisk. Zaručení konstantní délky otisku je pak dáno konstantním počtem hledaných shluků. Shlukem se rozumí uskupení bodů, jejichž vzájemné vzdálenosti jsou v porovnání s jinými body malé. Rovněž platí, že jeden bod nemůže být obsažen ve více shlucích najednou.

K řešení nastíněného problému je možné využít *Lloydův algoritmus*, který je také často označován jako *K-means*. Na vstupu algoritmu je množina N D-dimenzionálních bodů $\{x_1, ..., x_N\}$ a počet hledaných shluků $K \in \{1, ..., N\}$. Body množiny $\{x\}$ mohou být v případě této diplomové práce například vektory MFCC koeficientů nebo odhadů výkonových spektrálních hustot. Výsledkem algoritmu je množina K D-dimenzionálních bodů $\{\mu_1, ..., \mu_K\}$, tzv. *centroidů*, kde každý centroid μ_k je střední hodnotou bodů k-tého shluku. Úkolem algoritmu je rozdělení bodů do shluků s centroidy $\{\mu\}$ tak, aby platilo, že suma vzdáleností bodů od jim přidruženým centroidům je minimální.

Zmíněná suma vzdáleností je často označována jako *cenová funkce* nebo *míra zkreslení*. Pro její formální definici je třeba pro každý bod z množiny $\{x\}$ stanovit binární příznak $r_{nk} \in \{0, 1\}$, který nabývá hodnoty 1, právě když bod x_n náleží k-tému shluku. Cenová funkce $J(r, \mu)$ je pak dána výrazem

$$J(r, \mu) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2.$$

Hledání minima této funkce je prováděno pomocí iterativní procedury o dvou krocích popsané schématem 5.2. Konvergence algoritmu je zaručena tím, že v každé iteraci se hodnota cenové funkce J sníží. Algoritmus ovšem nezaručuje nalezení globálního minima, což je problém, kterým se zabývá sekce 5.2.2. Dalším nezanedbatelným problémem je nutnost volby počtu shluků [18]. Řešení tohoto problému je popsáno v sekci 5.2.1.

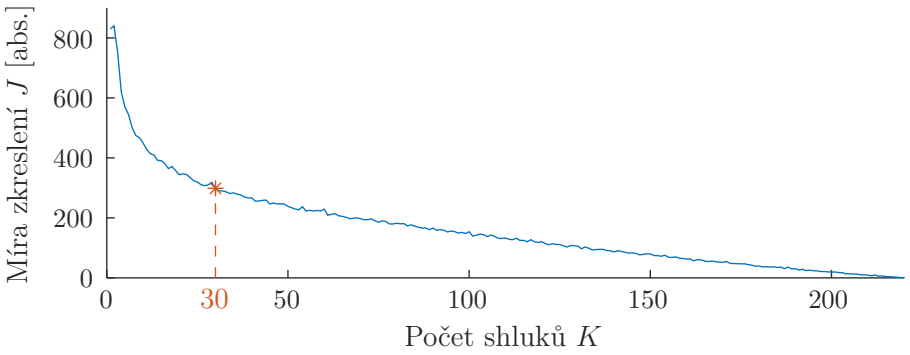


Obrázek 5.2: Schéma průběhu algoritmu K-means.

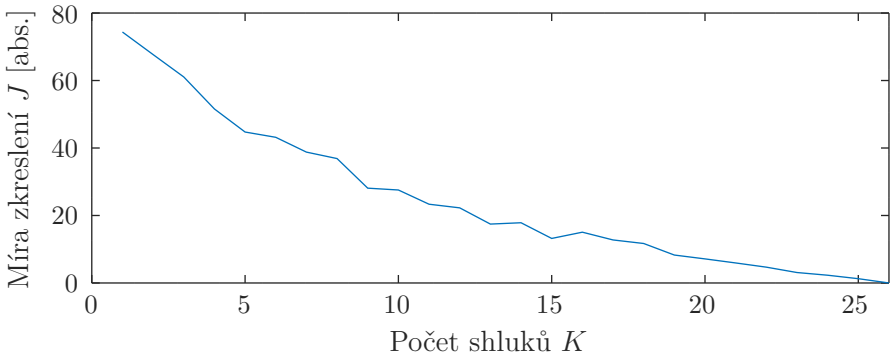
Lloydův algoritmus byl implementován v jazyce C++ a integrován do vytvořené knihovny `libpe`. Bližší popis je k dispozici v sekci 7.3.8.

5.2.1 Volba počtu shluků

Lloydův algoritmus na vstupu očekává počet shluků, do kterých budou body rozděleny. Za předpokladu, že existuje předchozí znalost o počtu shluků v datech, je volba jednoznačná. V opačném případě je možné využít tzv. *metody lokte*. Ta je založena na konstrukci grafu závislosti míry zkreslení J na zvoleném počtu shluků. Z definice algoritmu K-means vyplývá, že při $K = N$ shlucích bude $J = 0$, tj. každý z K shluků je tvořen jediným bodem, který je totožný s centroidem μ_k [1]. Na obrázku 5.3 jsou zobrazeny dvě závislosti, z nichž jedna ukazuje na existenci přibližně 30 shluků a druhá potvrzuje jejich rovnoměrné rozložení v příznakovém prostoru. Na obrázcích je rovněž vidět růst hodnoty J se zvyšujícím se K, který je způsoben konvergencí algoritmu do lokálního minima. Rovněž je důležité sledovat počet vzorků N.



(a) Existence lokte ukazující na přítomnost přibližně 30 shluků.



(b) Neexistence lokte naznačuje rovnoměrné rozložení bodů v přízn. prostoru.

Obrázek 5.3: Grafy závislosti míry zkreslení J na zvoleném počtu shluků K.

5.2.2 Volba prvotních pozic centroidů (K-means++)

Jedním z problémů Lloydova algoritmu je volba počátečních pozic centroidů $\{\mu\}$. Velice častým řešením je jejich náhodný výběr s rovnoměrným rozdělením pravděpodobnosti přímo z množiny bodů $\{x\}$. V tomto případě algoritmus konverguje k minimu, které obvykle nebývá globálním. Za účelem zpřesnění výsledku je tak algoritmus prováděn vícekrát. Další metodou výběru počátečních pozic je *K-means++*. Po definici funkce $D(x)$ jako nejmenší vzdálenosti bodu x k nejbližšímu centroidu můžeme algoritmus výběru popsat následovně:

1. Náhodný výběr centroidu μ_1 z množiny bodů $\{x\}$ s rovnoměrným rozdělením pravděpodobnosti.

2. Výběr nového centroidu μ_i, kde $i \in \{2, ..., K\}$, z bodů množiny $\{x\}$. Libovolný bod $x \in \{x\}$ bude vybrán s pravděpodobností $\frac{D(x)^2}{\sum_{n=1}^{N} D(x_n)^2}$.

3. Bod 2 je opakován, dokud nebude vybráno všech K centroidů.

S takto vybranými počátečními centroidy je proveden standardní Lloydův algoritmus [15].

5.3 Naivní detekce hlasové aktivity

Na počátku, resp. v závěru nahrávky promluvy podepisované osoby vznikají úseky ticha, které jsou způsobeny opožděným začátkem aktivity mluvčího, resp. pozdním ukončením nahrávání. Dle pozorování nashromážděných podpisů mohou tyto tiché úseky tvořit téměř polovinu délky nahrávky. Z provedených experimentů je patrné, že přesná extrakce úseků řečového signálu významně zvyšuje rozlišovací schopnosti navržených algoritmů (v některých případech i několikanásobně).

V současné době existují implementace například rekurentních neuronových sítí (RNN), které dokáží v akustickém signálu s vysokou úspěšností rozpoznat promluvu člověka a ignorovat jiné zvuky, jako jsou projíždějící auta apod. Tato řešení se hojně využívají například u chytrých zařízení, která tak de facto „slyší na zavolání“ [22].

V případě této diplomové práce, kdy spuštění záznamu je vyvoláno školenou osobou, postačí mnohem prostší řešení. Naivita navrhované techniky vychází z předpokladu, že pořízená nahrávka obsahuje kromě šumu jenom řečový signál s vysokým odstupem. V tom případě je možné segmenty řečového signálu identifikovat pouze na základě jejich energie, která je definována vztahem

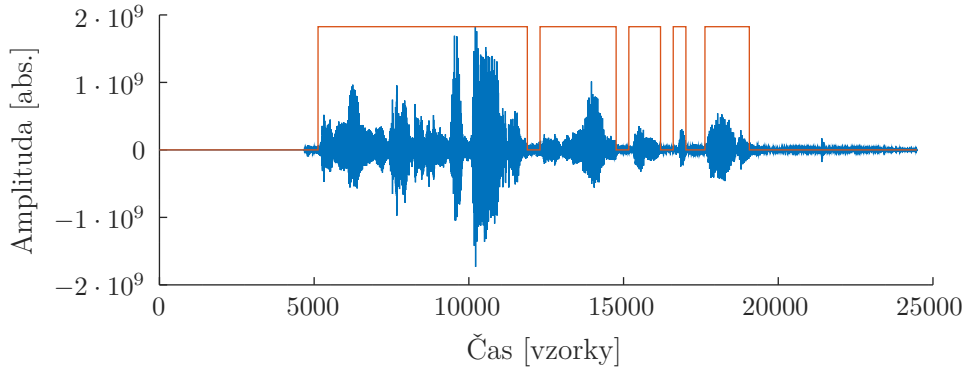
$$E_m = \sum_{n=0}^{N-1} \left(z_{m,n} w_n\right)^2 = \sum_{n=0}^{N-1} s_{m,n}^2 \ , \qquad \text{pro } \ m \in \langle 0, M-1 \rangle \subset \mathbb{N} \tag{5.1}$$

kde $z_{m,n}$ je n-tý vzorek m-tého segment akustického signálu, w_n je n-tý vzorek váhovacího okna o N vzorcích a M je počet segmentů akustického signálu. Způsob získání těchto proměnných je popsán v sekcích 4.3.1 a 4.3.2. E_m pak označuje energii m-tého segmentu akustického signálu.

Po stanovení energií všech M segmentů je určena jejich střední hodnota $\overline{E_m}$. Odstraněny jsou následně ty segmenty, jejichž energie vyhovuje nerovnosti

$$E_m < \eta \overline{E_m},$$

kde η je konstantní. Pro výpočty v praktické části diplomové práce byla na základě experimentů vybrána hodnota $\eta = 0{,}1$. Obrázek 5.4 ukazuje výsledek detekce navrženým algoritmem. Vytvořená implementace v jazyce C++ je podrobně popsána v sekci 7.3.3.



Obrázek 5.4: Detekované segmenty s hlasovou aktivitou při $\eta = 0{,}1$.

5.4 Detekce znělých segmentů

Kapitola 3 (přesněji 3.1.3) pojednávající o biometrických vlastnostech lidského hlasu zavádí pojmy znělých a neznělých zvuků. Na základě frekvenční analýzy znělých segmentů akustického signálu, která je vyobrazena v grafu 3.2, se jejich extrakce jeví jako rozumná myšlenka, protože právě tyto segmenty s formantovou strukturu nesou informaci o vlastnostech hlasového traktu mluvčího. V následujících oddílech textu jsou popsány analyzované metody jejich rozpoznávání.

5.4.1 Detekce založená na ZCR a energii signálu

Jedná se o metodu detekce znělých segmentů, která je založena na technice počítání průchodů signálu nulou, anglicky *Zero Crossing Rate (ZCR)*. Výsledek počítání indikuje frekvenční charakteristiku řečového signálu. U znělých segmentů, jejichž formantová struktura se objevuje v pásmu nízkých frekvencí přibližně od 0 do 500 Hz, bude tento indikátor nabývat nízkých hodnot. Na druhou stranu u neznělých zvuků se vlivem turbulentního proudění vzduchu energie soustředí na vyšších frekvencích, což vede k vysoké hodnotě ZCR. Počet průchodů nulou je dán vztahem

$$Z_m = \sum_{n=1}^{N-1} \frac{1}{2N} |\mathrm{sgn}(s_n) - \mathrm{sgn}(s_{n-1})| \ , \qquad \text{pro } \ m \in \langle 0, M-1 \rangle \subset \mathbb{N}$$

kde s_n představuje n-tý vzorek m-tého z M segmentů akustického signálu o N vzorcích a funkce $\mathrm{sgn}(x)$ je definována jako

$$\mathrm{sgn}(x) = \begin{cases} 1: & \forall x \geq 0 \\ -1: & \forall x < 0. \end{cases}$$

Druhou extrahovanou charakteristikou je krátkodobá energie signálu, která je dána vztahem (5.1). Znělé segmenty řečového signálu mají vlivem své periodicity na rozdíl od neznělých energii vysokou [16].

Autoři citového článku ovšem neuvedli žádné prahové hodnoty definovaných indikátorů, takže tato metoda nakonec využita nebyla.

5.4.2 Detekce integrováním znělé oblasti spektra

Tato technika je stejně jako metoda 5.4.1 založena na frekvenční charakteristice znělých a neznělých zvuků. Zavádí pojem *znělé oblasti*, která představuje část frekvenčního spektra v rozmezí 0–500 Hz. Integrováním této oblasti získáváme tzv. *úroveň znělosti*. Segment řečového signálu s odhadem spektra S, je označen za znělý, právě když splňuje nerovnost

$$\frac{\sum_{k=0}^{N-1} S_k}{N} < \frac{\sum_{l=0}^{M-1} S_l}{M},$$

kde N, resp. M je počet vzorků spektra, resp. znělé oblasti. M je dáno výrazem $M = \frac{2f_{vm}N}{f_s}$, kde f_{vm} je nejvyšší frekvence znělé oblasti (500 Hz) a f_s je frekvence vzorkování signálu [21].

Pro svou jednoznačnost byla tato technika implementována v knihovně `libpe`. Detailní popis je dispozici v sekci 7.3.4.

5.4.3 Porovnání podobnosti s modelem

Jedná se o metodu detekce znělých segmentů založenou na technice měření podobnosti signálů DTW, která je popsána v sekci 5.1. Metoda zavádí model odhadu spektra znělého segmentu řečového signálu. Tento model je dán součtem Gaussových funkcí, které jsou určeny vztahem

$$f(x) = ae^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

kde a určuje výšku křivky s vrcholem v bodě $x = \mu$ a σ udává šířku křivky ve výšce, kde nastává inflexe ($ae^{\frac{1}{8}}$) [17]. Segment řečového signálu je pak označen jako znělý, právě když výsledná cena borcení, tj. podobnost odhadu spektra segmentu a navrženého modelu, je menší než stanovený práh.

Použitý model i výsledky této metody jsou popsány v experimentu 9.2.2, jehož výsledek ukazuje obrázek 9.2.

Model spektra znělého segmentu je implementován v knihovně `libpe` jako třída, skrze jejíž veřejné metody je do modelu možné vkládat formanty daných frekvencí. Třída je popsána v sekci 7.3.4.

6 Použité technologie

Jedním z vedlejších cílů autora této diplomové práce bylo podrobné nastudování programovacího jazyka C++ [25]. V tomto objektovém jazyce, který je popsán v oddíle 6.1, bylo vytvořeno veškeré programové vybavení praktické části práce. Jazyk byl kromě osobních preferencí volen na základě vysoké výpočetní náročnosti prováděných experimentů. Využity byly i nabyté znalosti paralelizace výpočtů z předmětu *Paralelní programování* [8]. Další a možná i vhodnější cestou, kterou se autor kvůli svým osobním preferencím nevydal, by bylo využití knihoven jazyka *Python* [9].

Nejen pro tvorbu grafických rozhraní byl na základě výběru jazyka zvolen C++ framework Qt, který je přiblížen v sekci 6.2.

6.1 Programovací jazyk C++

C++ je nízkoúrovňový objektově orientovaný programovací jazyk, který byl vytvořen Bjarnem Stroustrupem v 80. letech minulého století v Bellových laboratořích jako objektová nadstavba jazyka C. Je standardizován Mezinárodní organizací pro normalizaci (ISO) a poslední verze jeho standardu je ISO/IEC 14882:2017, která je rovněž známa po označením C++17. Díky objektovému návrhu jazyk elegantně řeší uvolňování alokovaných zdrojů, které jsou nyní těsně svázány s platností objektů obalových tříd (v literatuře je toto označováno jako idiom RAII) [14]. V diplomové práci je hojně využíváno nástrojů objektového programování (dědičnosti, dynamického polymorfismu atd.), dále šablonových tříd a s nimi spojeného statického polymorfismu a lambda výrazů [25].

6.2 Framework Qt

Qt je multiplatformní framework pro C++ s otevřeným zdrojovým kódem. Je spravován norskou společností The Qt Company a jeho poslední verzí v době realizace práce je 5.14. Framework nabízí obdobnou funkcionalitu a datové struktury jako standardní knihovna jazyka C++ (STL) [10]. Kromě toho bylo během praktické části diplomové práce využito i modulů tvorby grafických rozhraní aplikací (Qt Widgets pro stolní počítače a Qt Quick pro mobilní zařízení). Pro paralelizaci výpočtů v experimentech byl použit modul Qt Concurrent [11].

7 Knihovna `libpe`

Během tvorby praktické části diplomové práce bylo zapotřebí testovat techniky, jejichž implementace v jazyce C++ není dle provedeného průzkumu volně dostupná nebo nesplňovala nutné požadavky. Příkladem může být aplikace algoritmu K-means v n-rozměrném prostoru společně s definicí vlastní metriky. Z tohoto a také čistě vzdělávacího hlediska bylo rozhodnuto o tvorbě samostatné statické knihovny, která nakonec obsahuje většinu užitých technik včetně jednotkových testů, které verifikují správnost implementovaných algoritmů.

Námět na tvorbu této knihovny vzešel již během autorovy bakalářské práce. První verze zdrojového kódu, která představuje zlomek nynější podoby, byla ale kompletně přepracována do moderního C++11. Místo globálního konfiguračního souboru jsou třídy šablonové a ve všech případech dynamických alokací je využíváno idiomu RAII, což zásadním způsobem zvyšuje bezpečnost a přehlednost kódu.

7.1 Užité technologie

Kvůli značným výpočetním nárokům a rozsahu navržených experimentů byl pro vývoj knihovny použit jazyk C++, který je zběžně popsán v sekci 6.1.

Původní myšlenkou byla implementace s použitím standardní šablonové knihovny jazyka (STL) [25]. Vytvořená knihovna je ovšem použita v aplikacích napsaných s použitím frameworku Qt. Dokumentace sice obsahuje výroky o kompatibilitě těchto dvou technologií, ale dle předchozích zkušeností autora práce propojitelnost mnohdy není až tak samozřejmá. Pro vývoj knihovny byl kvůli záruce hladké integrace nakonec zvolen framework Qt.

7.2 Struktura knihovny

Soubory knihovny jsou strukturovány dle zvyklostí C++ projektů do následující adresářové struktury.

`build`

Obsahuje přeložené binární soubory pro operační systémy (architektury procesorů)

- *Android* `armeabi-v7a`
- *Android* `x86`
- *Linux* `x64`
- *Windows* `x64`

`deps`

Kromě zdrojových souborů frameworku Qt je pro překlad knihovny nutná ještě další autorem vytvořená C++ knihovna nazvaná `libfp`. Z této knihovny je konkrétně využita implementace cyklického bufferu, na níž je postavena segmentace obecného signálu.

`doc`

Zdrojové soubory knihovny, zejména pak hlavičkové, obsahují velice podrobné programátorské komentáře všech tříd a metod. Pomocí nástroje *Doxygen* [3] je možné z těchto komentářů automaticky generovat dokumentaci ve formátu PDF. Adresář kromě souboru dokumentace obsahuje i konfigurační soubor, který slouží nástroji Doxygen jako návod pro její vytvoření.

`include`

Adresář, který je požadován hlavně programátory využívajícími funkcionalitu knihovny. Obsahuje totiž veřejné hlavičkové soubory, které je pro využití patřičných tříd nutné zahrnout pomocí direktivy `#include` v hlavičkových souborech aplikací. Vnořený adresář s názvem knihovny zajišťuje snadné rozpoznání závislostí z `libpe` a ostatních použitých knihoven ve zdrojových souborech aplikací.

`src`

Zde jsou k dispozici všechny zdrojové soubory knihovny. Nechybí ani návod k sestavení ve formátu `.pro` nástroje *qmake*. Jednotlivé třídy jsou rozděleny do adresářů podle svého účelu a jsou blíže popsány v sekci 7.3. Podrobný popis tříd obsahuje programátorská dokumentace v adresáři `doc`.

`test`

Poslední adresář obsahuje vytvořené testy, které verifikují výsledky implementovaných algoritmů. Pro jednotkové testování byla použita knihovna Qt Test [13]. Testovány tak byly například algoritmy K-means, DTW, metody pro určování vzdálenosti nebo datové struktury matice či PCM signálu. Pro testování implementace Fourierovy transformace byla vytvořena aplikace s grafickým rozhraním napojená na mikrofon zařízení.

7.3 Implementovaná funkcionalita

Knihovna `libpe` obsahuje implementace všech dosud probraných technik zpracování akustického signálu, tj. pořízení, předzpracování, detekce hlasové aktivity a znělých segmentů, krátkodobou spektrální analýzu, extrakci příznaků (MFCC koeficientů), DTW, K-means apod. Většina tříd pracujících se signálem je šablonových. Parametr šablony `T`, který se bude v dalším textu objevovat zapsaný v syntaxi jazyka C++ v lomených závorkách, označuje datový typ vzorku zpracovávaného signálu. Všechny implementace jsou zběžně popsány následujícími sekcemi textu. Podrobnou programátorskou dokumentaci je možné nalézt v adresáři `doc`.

7.3.1 Pořízení a předzpracování signálu

Pořízení signálu je pomocí knihovny možné buď přímo z mikrofonu zařízení, nebo z bezhlavičkového binárního souboru PCM vzorků s příponou `.raw`. Kontejnerem signálu je třída `PcmSignal<T>`, která kromě vektoru vzorků obsahuje informaci o vzorkovací frekvenci. Načtení vzorků je možné provést ze souborů formátu `.mp3`, resp. `.raw` pomocí objektů tříd `Mp3Loader<T>`, resp. `RawLoader<T>`. U třídy `RawLoader<T>` je třeba, aby typ `T` rovněž odpovídal typu vzorků v souboru.

Pro předzpracování existuje v knihovně třída `Preprocess`, která obsahuje statické metody `removeDC<T>` pro odstranění DC ofsetu a `preemphasis<T>` pro aplikaci preemfáze. Díky šablonám je možné tyto techniky aplikovat na signály s například celočíselnými či reálnými vzorky, a to beze změny kódu. Třídy implementují techniky popsané v sekcích 4.1 a 4.2.

7.3.2 Rozdělení signálu na segmenty a váhování

Rozdělení signálu na segmenty je založeno na cyklickém bufferu, který je implementován ve třídě `CyclicBuffer<T>` autorovy další knihovny `libfp`. Rozdělovaný signál je možné získat buď z existující instance třídy `PcmSignal<T>`, nebo přímo z mikrofonu zařízení. V prvním případě hovoříme o třídě `SignalSegmenter<T>`, která má kromě jiných i veřejné metody `writeSignal` a `nextSegment`. Pokud cyklický buffer obsahuje nějaké vzorky, metoda `nextSegment` vrátí segment signálu dané délky. Pokud počet vzorků v bufferu je menší než nastavená velikost bufferu, metoda jej automaticky doplní nulami.

Třída `IOSignalSegmenter<T>` je implementována jako vstupně výstupní zařízení, do kterého zapisují moduly frameworku Qt pracující se zvukovými zařízeními. Nashromážděné segmenty jsou předávány pomocí systému *Sig-*

nal & Slot [12]. Tyto třídy implementují techniky popsané v sekci 4.3.1.

Váhování segmentů je zaručeno třídou `WindowFunction<T>`, která obsahuje ryze virtuální metodu pro inicializaci použitého okna. V knihovně existuje pouze jediný potomek této třídy s názvem `HammingWindow<T>`, který vytváří okno dle vztahu (4.1).

7.3.3 Detekce řečové aktivity

Problém detekce řečové aktivity, který je popsán v sekci 5.3, řeší třída `NaiveVoiceDetector<T>`. Ta obsahuje jedinou veřejnou metodu, jejímž parametrem je pole segmentů akustického signálu. Pomocí techniky popsané v sekci 5.3 jsou vybrány segmenty s řečovou aktivitou a výsledkem metody je pole tzv. *iterátorů* [25], které je označují ve vstupním poli. Příkladem výsledku je obrázek 5.4. Prototyp této metody ukazuje zdrojový kód 7.1, na kterém je poprvé vidět „krása“ jazyka *C++*.

```
QVector<typename QVector<QVector<T>>::iterator>
   getVoiceSegments(QVector<QVector<T>>& segments);
```

Zdrojový kód 7.1: Prototyp metody `getVoiceSegments`.

7.3.4 Detekce znělých segmentů

Jediná implementovaná technika, která je založena na integrování znělé oblasti, je vysvětlena v sekci 5.4.2. Detekce je možná voláním statické metody `Voicing::isVoiced<T>()`, která očekává odhad spektra segmentu signálu, vzorkovací frekvenci f_s a horní hranici znělé oblasti f_{vm}.

Metoda založená na měření podobnosti odhadu spektra s modelem, která je popsána sekcí 5.4.3, byla pouze experimentálně testována v aplikaci *HashVisualisation*. Experiment je popsán v sekci textu 9.2.2. V knihovně `libpe` je ovšem implementován model odhadu spektra `VoiceModel<T>`, jehož veřejná metoda `addFormant` dovoluje modelovat formantovou strukturu odhadu spektra znělé hlásky pomocí Gaussových funkcí.

7.3.5 Krátkodobá spektrální analýza

Pomocí třídy `FFT`, jejíž základem je knihovna *KissFFT* [5], je možný výpočet odhadů jak komplexních spekter, tak i výkonových spektrálních hustot, které jsou v textu práce zkráceně označovány jako odhad spektra a jejich

výpočet je popsán sekci 4.3.3. Metody třídy na svém vstupu očekávají váhované segmenty se vzorky typu `float`. Díky tomu tato třída není šablonová, tj. parametr `<T>` není uveden. Kvůli dynamické alokaci struktur potřebných pro výpočet FFT je velikost segmentů udávána již při konstrukci objektu třídy. Výsledné odhady spekter jsou dimenze $\frac{N}{2} + 1$, kde N je nastavený počet vzorků segmentů. Pokud N není mocninou čísla dvě, metody třídy dané segmenty automaticky doplní nulami na nejbližší vyšší mocninu. V této práci byly všechny grafy odhadů spekter vypočteny pomocí instance této třídy.

7.3.6 Výpočet MFCC koeficientů

Z diagramu 4.4, který ukazuje postup výpočtu MFCC koeficientů, je patrné, že většinu jeho kroků již knihovna implementuje a jejich popis je možné najít v předchozích oddílech textu. Po výpočtu odhadů spekter je dalším krokem jejich filtrace pomocí trojúhelníkových filtrů. Tvorbu a správu těchto filtrů zaručuje třída `MelFilterBank`. Filtry vytvořené touto třídou jsou ukázány na obrázku 4.8.

Získání MFCC koeficientů, tj. výpočet melovských koeficientů, jejich logaritmování a následný převod na kepstrální pomocí DCT zaručuje třída `MFCC`, jejíž komponentou je objekt třídy `MelFilterBank`.

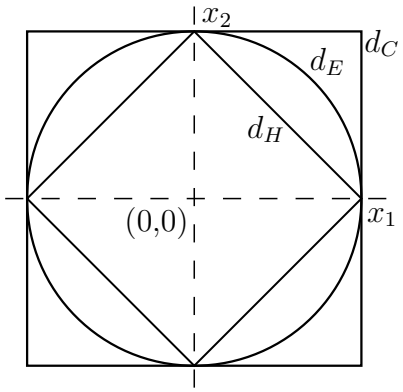
7.3.7 Měření vzdálenosti

Pro implementaci algoritmů DTW nebo K-means bylo nutné zavést možnost měření vzdálenosti dvou bodů v n-rozměrném prostoru. Z tohoto důvodu byla vytvořena třída `Metrics`, která obsahuje statické metody pro výpočet *Hammingovy*, *Eukleidovy* a *Čebyševovy* metriky. Význam metrik demonstruje obrázek 7.1. Pro zvýšení udržitelnosti kódu jsou metody šablonové. Prototyp metody pro výpočet Hammingovy metriky ukazuje zdrojový kód 7.2, na kterém je vidět využití konstantních referencí zabraňujících kopírování dat.

```
template <class T>
static T hamming(const QVector<T>& v1, const QVector<T>& v2);
```

Zdrojový kód 7.2: Prototyp metody výpočtu Hammingovy metriky.

Poslední možností měření „vzdálenosti“ je metoda DTW. Jak je ale popisována v sekci 5.1, technika DTW nesplňuje všechny tři axiomy definující metriku a její použití například u K-means může způsobit divergenci algoritmu.



Obrázek 7.1: Místa bodů se stejnou vzdáleností od počátku dvojrozměrné souřadné soustavy při použití Hammingovy (d_H), Eukleidovy (d_E) a Čebyševovy (d_C) metriky [23].

7.3.8 Algoritmus K-means

O tvorbě vlastní implementace algoritmu K-means, jenž je popsán v sekci 5.2, bylo rozhodnuto na základě průzkumu existujících řešení, která ve většině případů poskytují možnost shlukové analýzy pouze ve dvourozměrném prostoru. Z hlediska navržených algoritmů pro výpočet otisku osoby je toto řešení nepoužitelné, a tak byla vytvořena třída `KMeans<T>`. Šablonový parametr `T` označuje datový typ prvků vektorů určujících pozici analyzovaných bodů. Pro zpřehlednění kódu třída definuje typ `Point` (bod) jako `QVector<T>`. Při konstrukci objektu třídy je možné definovat vlastní metriku pro určení vzdálenosti bodů. Výchozí možností je Eukleidova metrika implementovaná ve třídě `Metrics`. Prototyp konstruktoru ukazuje zdrojový kód 7.3. Díky použití typu `std::function` je možné definovat vlastní metriku pomocí lambda funkce.

```
explicit KMeans(std::function<T(const Point&, const Point&)>
                    metric = &Metrics::euclDistN<T>);
```

Zdrojový kód 7.3: Prototyp konstruktoru třídy `KMeans<T>`.

Body náležící jednomu shluku jsou označeny polem konstantních iterátorů. U shluků je dále uvedena poloha centroidu a informace o součtu vzdáleností bodů od něj. Všechny tyto údaje obaluje struktura `Cluster`, jejíž definici ukazuje zdrojový kód 7.4.

Výsledek shlukovací analýzy je dán instancí struktury `Result`, která obsahuje pole nalezených shluků, hodnotu míry zkreslení J a potřebný počet iterací algoritmu. Definici této struktury obsahuje zdrojový kód 7.5.

Výběr počátečních pozic centroidů je možné provádět náhodně z analyzovaných bodů, ručním zadáním nebo pomocí úpravy K-means++. Nechybí ani veřejné metody pro určení shluku neznámého bodu.

```
struct Cluster {
   QVector<typename QVector<Point>::const_iterator> points;
   T totalDistance;
   Point centroid;
};
```

Zdrojový kód 7.4: Definice typu `KMeans<T>::Cluster`.

```
struct Result {
   QVector<Cluster> clusters;
   T totalDistance;
   int iterations;
};
```

Zdrojový kód 7.5: Definice typu `KMeans<T>::Result`.

7.3.9 Algoritmus DTW

Další třídou knihovny `libpe` je `DTW<T>`, která implementuje stejnojmenný algoritmus popsaný v sekci 5.1. Účelem metod této třídy je výpočet cenové matice, ze které lze následně získat cenu borcení. Podobně jako u třídy `KMeans<T>` je metrika pro získání vzdálenosti mezi dvěma vzorky signálu určena již při konstrukci objektu třídy. Výpočet ceny borcení signálů `s1` a `s2` je popsán zdrojovým kódem 7.6.

```
QVector<double> s1 = { ... };
QVector<double> s2 = { ... };

DTW<double> dtw([](double s1i, double s2i) -> double {
   return std::abs(s1i - s2i);
});

double warp_cost = dtw.calcWarpCost(s1, s2);
```

Zdrojový kód 7.6: Výpočet ceny borcení pomocí třídy `DTW<T>`.

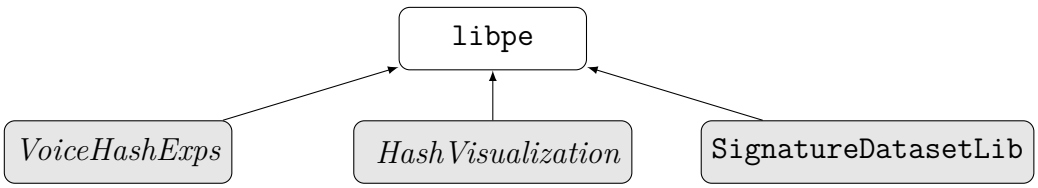
7.3.10 Ostatní matematické operace a tisk dat

Kromě komponent zmíněných v předchozích sekcích textu knihovna obsahuje třídy pro základní statistické operace, normalizaci, kvantizaci a tisk vektorů do souboru. Základní statistické operace, jako je maximum, minimum, median, modus apod., jsou implementovány jako šablonové metody třídy `Statistics`. Normalizaci a kvantizaci signálů implementují třídy `Normalization<T>` a `Quantization<T>` (společně s potomky `UniformQuantization<T>` a `NonUniformQuantization<T>`).

Poslední, ale neméně důležitou třídou knihovny je třída `VectorPrinter<T>`, pomocí které je možné signály v zájmu jejich vizualizace tisknout, například do skriptů jazyka *Octave* [7]. Touto cestou vznikly téměř všechny grafy v textu práce.

7.4 Význam knihovny v systému

Z diagramu 7.2 je patrné, že na knihovně `libpe` je postaveno všechno vytvořené programové vybavení. Aplikace *VoiceHashExps* a *HashVisualization* využívají téměř všechny dostupné třídy knihovny. Knihovna `SignatureDatasetLib` používá třídy obalující PCM vzorky akustického signálu promluv podepisovaných osob.



Obrázek 7.2: Diagram využití knihovny `libpe`.

8 Testovací dataset

Pro testování implementovaných algoritmů byla shromážděna množina multimodálních podpisů nazvaná *SignatureDataset* obsahující

- celkově 153 videozáznamů (66 se ženami a 87 s muži)
- od 34 dobrovolníků (15 žen a 19 mužů).

Popis získaných videozáznamů a celkové struktury nashomážděného datasetu je k dispozici v sekcích 8.1 a 8.2.

Spolu s datasetem byla vytvořena i miniaturní knihovna v C++, která obsahuje třídy dovolující správu a načítání vzorků datasetu. Knihovna je blíže popsána v sekci 8.3.

Pro snadnou a unifikovanou tvorbu videozáznamů byla vytvořena mobilní aplikace, která byla rozšířena mezi 5 přispěvatelů, kteří jsou uvedeni v tabulce 8.1. Podrobný popis aplikace obsahuje sekce 8.4.

8.1 Popis videozáznamů

Obsahem pořízených multimodálních podpisů je obličej dobrovolníka zarovnaný podle jeho nosu na střed snímku. Dobrovolník dále do kamery mobilního zařízení vysloví frázi

„*Jmenuji se <jméno> <příjmení> a přijímám tento balík.*“,

kde *<jméno>* a *<příjmení>* jsou povinné fiktivní údaje, které si dobrovolník volí sám.

Kvůli snaze o maximální robustnost navržených algoritmů byli přispěvatelé instruováni, aby videozáznamy téže osoby pořizovali v různých prostředích, s různě upravenými vlasy, bez pokrývky hlavy nebo s ní a podobně. V množině pořízených videozáznamů se tak vyskytují kuriózní případy, kdy dotyčná osoba nosí paruku, či dokonce přilbu.

videozáznamy v datasetu jsou pojmenovány podle schématu

`<id-dobrovolníka>_<id-videa>_<pohlaví>_<věk>.mp4`,

kde údaje ve špičatých závorkách představují povinné údaje. V oficiální podobě datasetu jsou videozáznamy rozděleny podle přispěvatelů, čili atribut

`id-dobrovolníka` tedy představuje primární klíč dobrovolníka v rámci jednoho přispěvatele. Pomocí skriptu `MergeVideos.sh` a nástroje `VideoIntegrator` jsou pak všechna videa sloučena do jednoho adresáře a identifikátory jsou automaticky přečíslovány, aby měly již globální význam. Aplikace `VideoIntegrator` je popsána v sekci 8.5. Atribut `id-videa` je jednoznačným identifikátorem multimodálního podpisu daného dobrovolníka.

8.2 Adresářová struktura datasetu

Dataset kromě samotných multimodálních podpisů obsahuje i množinu komponent, které s těmito videozáznamy usnadňují další práci. Celý dataset je tak rozčleněn na následující adresáře.

`collector`

V tomto adresáři je k dispozici spustitelná verze mobilní aplikace *VideoCollector* pro operační systém *Android*. Nechybí ani všechny zdrojové soubory včetně kompletní programátorské dokumentace, která byla vygenerována nástrojem *Doxygen* [3]. Tato aplikace je blíže popsána v sekci 8.4.

`gdpr`

Obsahuje tzv. *Poučení o zpracování osobních údajů*, které bylo nutné nechat podepsat od všech zúčastněných dobrovolníků. Adresář obsahuje kromě dokumentu ve formátu PDF i zdrojový kód v jazyce TEX (LATEX).

`raw`

Tento adresář obsahuje další podadresáře, které obsahují již konkrétní multimodální podpisy pořízené jednotlivými přispěvateli. Názvy podadresářů odpovídají schématu (8.1), kde údaje ve špičatých závorkách označují obligatorní údaje a popisují konkrétní přispěvatele.

```
<pořadové-číslo>␣<jméno>␣<příjmení>␣(<typ-telefonu>)
```
(8.1)

`integrator`

Adresář obsahuje konzolovou aplikaci určenou ke sloučení složek multimodálních podpisů od jednotlivých přispěvatelů do finálního datasetu. Adresář obsahuje spustitelný soubor pro *OS Linux* a všechny zdrojové soubory nutné pro překlad na jiné platformy. Posledním významným souborem je programátorská dokumentace. Tato aplikace je blíže popsána v sekci 8.5.

Tabulka 8.1: Tabulka přispěvatelů do datasetu multimodálních podpisů.

Přispěvatel(ka)	**Mobilní telefon**	**Pořízená videa**
František Pártl	Lenovo K5 Plus	54
Monika Hanušová	Lenovo Vibe P1M	45
Lenka Benešová	Samsung Galaxy J5	35
Michal Šteňo	Asus ZE620KL	14
Simona Kolářová	Huawei P9 lite	5

`lib`

Poslední z adresářů obsahuje miniaturní knihovnu pro C++ složenou z množiny tříd, které dalším aplikacím umožňují načítání videí datasetu. Tato knihovna je podrobně popsána v sekci 8.3.

8.3 Knihovna `SignatureDatasetLib`

Tato knihovna vznikla za účelem unifikované a snadné manipulace s multimodálními podpisy v rámci aplikací pracujících s datasetem. Jejím obsahem je množina C++ tříd, dovolující aplikaci načtení nejenom metadat podpisů (identifikátory, věk apod.), ale například i vlastních vzorků akustických signálů z jejich zvukových stop. Konkrétně pro práci se zvukem je využita knihovna `libpe`, která je podrobně rozebrána v kapitole 7. `SignatureDatasetLib` je založena na frameworku Qt, který je popsán v sekci 6.2. Zdrojové kódy jsou udržovány v repozitáři, jehož submodulem je právě knihovna `libpe`.

8.3.1 Vytvořené třídy

Třída `SignatureSample`

Jedná se o třídu představující obecný vzorek datasetu *SignatureDataset*. Mezi atributy třídy patří

- identifikátor dobrovolníka,
- identifikátor podpisu dobrovolníka,
- pohlaví
- a věk.

Z tohoto seznamu je patrné, že data instance této třídy neobsahují video ani audiozáznam. Třída také definuje výčtový typ, který vymezuje možná pohlaví dobrovolníků. Inicializace třídy je dána relativní nebo absolutní cestou k samotnému vzorku datasetu. Název tohoto souboru je rozložen pomocí regulárního výrazu. Pokud název výrazu neodpovídá, záznam je označen jako nevalidní, tj. návratovou hodnotou metody `isValid()` je *nepravda*. Na tuto situaci všechny vytvořené aplikace reagují jeho ignorováním.

`SignatureFolder`

Tato třída obsahuje jedinou veřejnou metodu, která ze zadaného adresáře načte všechny dostupné vzorky datasetu. Výsledkem metody je vektor sdílených ukazatelů (`QSharedPointer`) na objekty třídy `SignatureSample`, u kterých je zaručena jejich validita.

`AudSignSamp`

Jedná se o šablonovou třídu, která je potomkem třídy `SignatureSample`. Představuje zvukovou stopu videozáznamu datasetu. Pro konstrukci objektu této třídy je potřeba cesta k binárnímu souboru, který vzorky signálu obsahuje. Soubor lze z videozáznamu vytvořit pomocí skriptu `ExtractAudio.sh` a nástroje *FFmpeg* [4]. Parametr šablony `<T>` pak značí konkrétní typ vzorku akustického signálu. Dataset *SignatureDataset* používá 32-bitové celočíselné znaménkové vzorky při vzorkovací frekvencí $f_v = 8000$ Hz. Konstrukci objektu třídy podle této konfigurace ukazuje zdrojový kód 8.1 (kód zanedbává identifikátor objektu). Pro správu vzorků a vzorkovací frekvenci akustického signálu je využita šablonová třída `PcmSignal<T>` knihovny `libpe`, která je popsána v sekci 7.3.1.

```
AudSignSamp<int32_t>("signaturedataset/1_1_M_100.raw", 8000);
```

Zdrojový kód 8.1: Příklad konstrukce objektu třídy `AudSignSamp<T>`.

8.4 Aplikace *VideoCollector*

V zájmu zaručení jednotného scénáře získávání multimodálních podpisů jednotlivými přispěvateli byla vytvořena mobilní aplikace s označením *VideoCollector*. Postup, jakým se jednotliví přispěvatelé při pořizování videozáznamů řídí, je popsán v sekci 8.4.1. Pro vývoj aplikace byl kvůli možnosti využití zdrojových souborů knihovny `SignatureDatasetLib` a autorovým

předchozím zkušenostem zvolen programovací jazyk C++ společně s frameworkem Qt. Tyto technologie jsou popsány v kapitole 6.

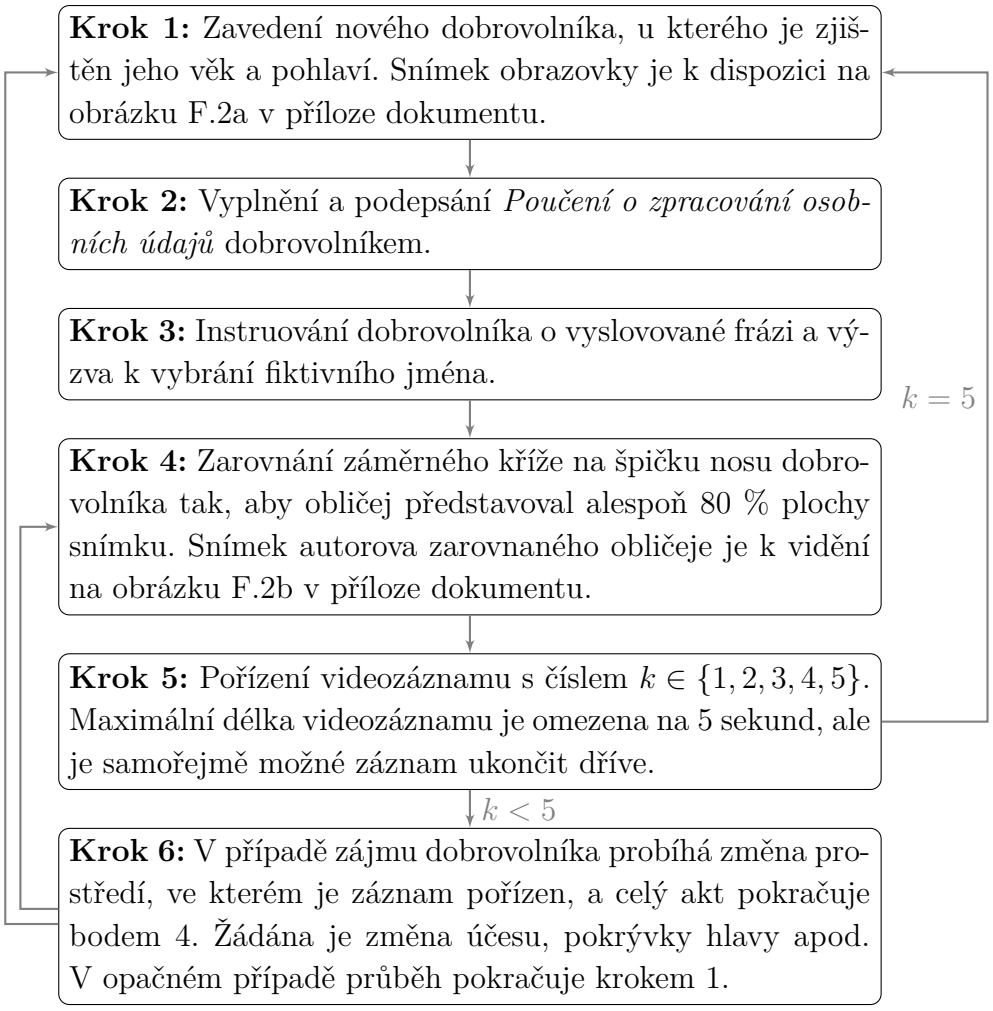
8.4.1 Scénář pořízení podpisu

Po instalaci mobilní aplikace na zařízení přispěvatele se ve výchozím umístění videozáznamů vytvoří adresář `VideoCollector_videos`. Do tohoto adresáře jsou později ukládány všechny pořízené multimodální podpisy dobrovolníků. Samotný akt pořízení podpisu probíhá podle diagramu 8.1. Po tom, co přispěvatel už o další spolupráci nestojí nebo již nemá k dispozici další dobrovolníky, kontaktuje autora práce, který pomocí USB připojení stáhne videozáznamy ze zařízení.

Při aktu stahování videozáznamů bylo registrováno zajímavé chování všech zařízení. Adresář `VideoCollector_videos` totiž není viditelný ani ve správci souborů systému *Android* a ani při připojení mobilního zařízení k osobnímu počítači. Soubory jsou však viditelné z aplikace *VideoCollector*. Nehledě na typ zařízení je přístup k videozáznamům umožněn až po jeho restartu. Operační systém *Android* totiž pro přístup k datům uživatele nabízí aplikacím službu zvanou *MediaStore* [6]. Služba aktivně načítá informace o souborech úložiště pouze při svém startu, tedy při spouštění zařízení. Aplikace *VideoCollector* této služby nevyužívá a k souborům přistupuje přímo. Aby byl soubor viditelný i bez restartu, musela by aplikace při zápisu souboru volat specializovanou metodu v jazyce *Java*, což je ve frameworku *Qt* obtížně implementovatelné.

8.5 Aplikace *VideoIntegrator*

Poslední programovou součástí datasetu je aplikace *VideoIntegrator*, jejíž účelem je sloučení videozáznamů jednotlivých přispěvatelů do jednoho globálního adresáře. Stejně jako ostatní komponenty datasetu je založená na zdrojových kódech knihovny `SignatureDatasetLib`.



Obrázek 8.1: Průběh pořizování podpisů pomocí aplikace *VideoCollector*.

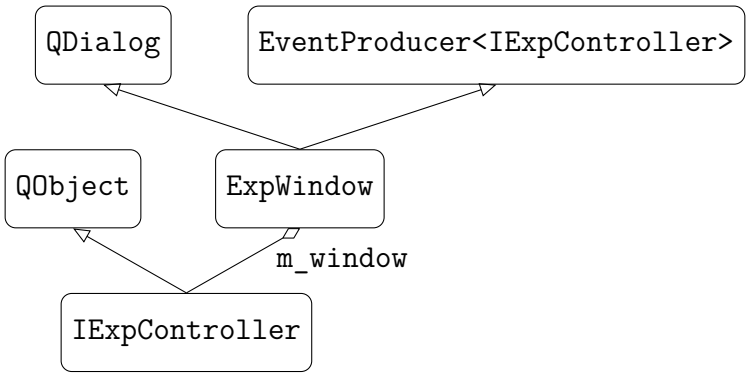
9 Aplikace *HashVisualisation*

Klíčovým krokem při návrhu algoritmů extrakce příznaků ze zvukových stop nashromážděných multimodálních podpisů byly vizualizační experimenty. Díky nim bylo možné přímo na vzorcích datasetu *SignatureDataset* zkoumat charakteristiky, které jsou pro osoby potenciálně příznačné. První experimenty byly připraveny jako krátké skripty v jazyce Python [9]. Protože ale kvůli načítání vzorků bylo nutné měnit zdrojový kód, k datasetu nebylo vytvořeno API pro jazyk Python a značná část zamýšlených technik byla implementována v jazyce C++, bylo rozhodnuto pro tvorbu aplikace s grafickým rozhraním integrující všechny navržené vizualizační experimenty nad datasetem – *HashVisualisation*. Zdrojový kód aplikace je napsán v jazyce C++ a využívá framework Qt. Obě tyto technologie jsou popsány v kapitole 6. V aplikaci je využito téměř všech algoritmů implementovaných v knihovně `libpe`, která je popsána kapitolou 7. Samotný zdrojový kód je pak přiblížen v sekci 9.1. Scénáře a možné konfigurace vytvořených vizualizačních experimentů uvádí sekce 9.2. Uživatelská příručka včetně popisu grafického rozhraní je k dispozici v sekci E.1 přílohy dokumentu.

9.1 Struktura zdrojového kódu

Zdrojový kód aplikace je silně objektově orientován a je postaven na MVC architektuře. Rozhraní aplikace, tj. dvě okna, která zabezpečují výběr experimentu a jeho následné provedení, komunikují s kontrolními objekty pomocí návrhového vzorku *Observer (Pozorovatel)* [24]. Jeho implementací je virtuální třída `EventProducer<T>` autorovy knihovny `libfp`, která pro šíření zpráv používá mechanismus *Signal & Slot* frameworku Qt [12]. Parametr šablony `T` představuje typ objektu *pozorovatele*. Potomek této třídy, v tomto případě třída `ExpWindow` obalující grafické rozhraní, implementuje virtuální metody, určené k vytvoření mapování signálů okna na obslužné metody *pozorovatele*, tj. virtuální třídy `IExpController`. Zpětná komunikace probíhá přes referenci `m_window` obsaženou ve třídě *pozorovatele*. Popsanou situaci ukazuje diagram tříd 9.1.

Problém existence většího počtu experimentů je řešen pomocí dynamického polymorfismu. Každý implementovaný experiment má totiž stejné grafické rozhraní, ale specializované ovladače, které jsou všechny potomky třídy `IExpController`. Tento rodič definuje společné metody, jako je načtení da-



Obrázek 9.1: Schéma dědičnosti a kompozice okna experimentu.

tasetu, spuštění experimentů v odděleném vlákně nad vybranými vzorky datasetu apod. Dále deklaruje ryze virtuální metody, jejichž účelem je tvorba grafických komponent nutných pro stanovení konfigurace konkrétního experimentu (`initConfigInput`) a samotnou metodu jeho provedení (`performExperiment`).

Tuto architekturu je možné popsat na příkladu provádění experimentů. Uživatel pomocí grafického rozhraní navolí konfiguraci experimentu a vybere zpracovávané vzorky datasetu. Při stisknutí tlačítka „Spustit experiment“ je podle návrhového vzoru Observer spuštěna obslužná metoda ovladače `IExpController`. V těle této metody je prostřednictvím reference na okno nejdříve vyprázdněn výstupní graf. Dále je otevřeno dialogové okno s indikátorem průběhu výpočtu. Metoda `map` modulu *QtConcurrent* frameworku Qt [11] paralelně pro každý zvolený vzorek volá virtuální metodu `performExperiment`. Na jejím začátku proběhne extrakce konfigurace experimentu z grafického rozhraní. Jakmile je výpočet experimentu dokončen, emituje se signál s výsledky, na který reaguje grafické rozhraní jejich vykreslením do grafu. Díky využití mechanismu *Signal & Slot*, který garantuje sekvenční spouštění obslužných metod, nemůže dojít k souběhu při vykreslování.

9.2 Vizualizační experimenty

Po tom, co uživatel v úvodním okně zadá adresář se soubory akustických podpisů datasetu *SignatureDataset*, má na výběr z šesti experimentů, které jsou popsány následujícími oddíly textu.

9.2.1 Naivní detekce řečové aktivity

V tomto experimentu může uživatel zkoušet různé konfigurace naivní detekce řečové aktivity, která je popsána v sekci 5.3. Konkrétně lze nastavovat:

- zdali má být normalizována střední hodnota akustického signálu,
- má-li být na signál aplikována preemfáze, u které je i možnost volby koeficientu α,
- jaký je volen práh η, jehož význam vysvětluje sekce 5.3,
- velikost segmentů akustického signálu
- a jejich překryv.

Výsledkem experimentu je pak vizualizace průběhu akustického signálu s vyznačenými úseky promluv. Výstup experimentu je vidět na obrázku 5.4. Kvůli vysokému počtu vykreslovaných vzorků během testování často docházelo k „zamrznutí“ aplikace. Kvůli tomu byla do vizualizace zavedena decimace výsledného signálu, která je aplikována tak, aby počet vzorků vykreslovaného signálu nikdy nepřesáhl hodnotu 5000. Ze stejného důvodu není u tohoto experimentu doporučena analýza většího počtu vzorků datasetu. Ovladač, který obsahuje implementaci experimentu, se nazývá `NVADController`.

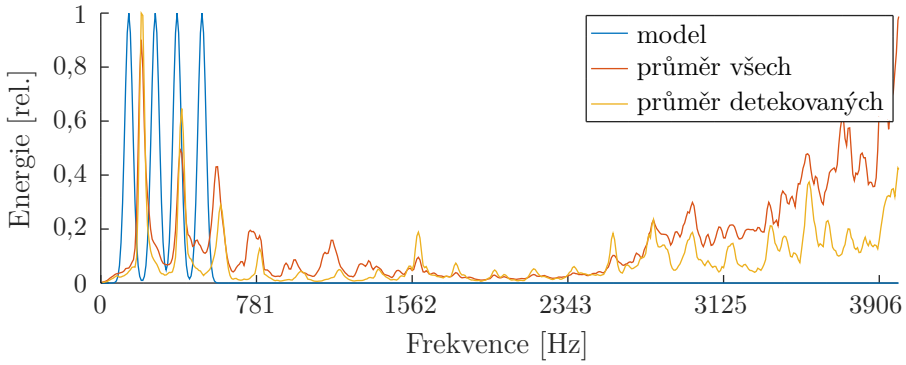
9.2.2 Průměrné spektrum s detekcí znělých segmentů

V tomto experimentu je testována navržená technika detekce znělých segmentů popsaná v sekci 5.4.3. Akustický signál je zde zprvu zpracován stejným způsobem jako v experimentu 9.2.1. Po identifikaci segmentů s hlasovou aktivitou jsou vypočteny odhady jejich spekter, u kterých je stanovena „podobnost“ (cena borcení) s modelem odhadu spektra znělé hlásky. Daná procentuální část nejpodobnějších odhadů spekter je ponechána a zprůměrována, čímž vzniká výsledek experimentu. Kromě nastavení shodných s experimentem 9.2.1 je k dispozici možnost:

- zakázání či povolení detekce segmentů s řečovou aktivitou,
- volby procentuální části zachovávaných s modelem nejpodobnějších segmentů,
- zobrazení modelu do výsledné vizualizace,
- vykreslení průměrů všech segmentů, aby bylo možné vidět efekt navržené detekce,

- normalizace rozsahu hodnot mezi $\langle 0, 1 \rangle$
- a případná kvantizace výsledku s daným krokem.

Výsledek experimentu ukazuje obrázek 9.2, na kterém je vidět referenční model a průměry všech odhadů spekter i části s modelem nejpodobnějších. Formantová struktura modelu je tvořena čtyřmi Gaussovými funkcemi, jejichž parametry jsou uvedeny tabulce 9.1. Obrázek ukazuje mírné zvýšení energií ve znělé oblasti a zároveň mírné snížení ve vyšších frekvencích. Na základě provedených experimentů se ovšem navržená technika ukazuje jako nevhodná pro další použití. Jediným vylepšením, které ovšem nebylo implementováno, by mohlo být automatické nastavení absolutní velikosti amplitudy Gaussových funkcí.



Obrázek 9.2: Výsledek experimentu průměrování spekter s detekcí znělých segmentů při zachování 20 % nejpodobnějších odhadů spekter.

Tabulka 9.1: Parametry Gaussových funkcí tvořících model.

Formant	**Frekvence (μ)**	**Šířka (σ)**	**Výška (a)**
F_0	140 Hz	20 Hz	1
F_1	270 Hz	20 Hz	1
F_2	375 Hz	20 Hz	1
F_3	500 Hz	20 Hz	1

9.2.3 Průměrný odhad spektra

Tento experiment umožňuje uživateli vizualizaci průměrných odhadů spektrálních hustot zvolených nahrávek datasetu *SignatureDataset*. Prvotní zpracování signálu je stejné jako v experimentu 9.2.1, s čímž je spojena i konfigurace algoritmu.

Mezi další volitelné parametry patří:

- povolení či zakázání detekce segmentů s řečovou aktivitou,
- odstranění neznělých segmentů technikou integrování znělé oblasti, která je popsána v oddíle 5.4.2 (je také možné měnit nejvyšší frekvenci znělé oblasti f_{vm})
- povolení či zakázání normalizace výsledných spekter
- a jejich kvantizace se zvoleným krokem.

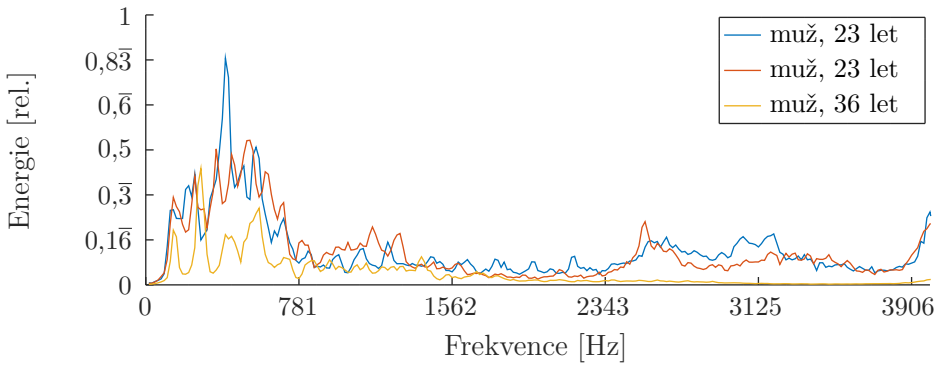
Výsledek experimentu demonstruje obrázek 9.3, který ukazuje odhady spekter promluv dvou dobrovolníků s a bez detekce znělých segmentů technikou integrování znělé oblasti. Ukazuje se, že detekce znělých segmentů nemá nijak významný efekt na výsledný průměrný odhad spektra, a u některých navržených algoritmů dokonce zhoršuje jejich rozlišovací schopnost. Pro ukázku vysokého vnitrotřídního rozptylu je pro 23-letého dobrovolníka vykreslen průměr odhadů spekter ze dvou akustických podpisů. Třetí odhad spektra pochází od 36-leté osoby, u které je oproti ostatním jasně zřetelná formantová struktura. Z experimentu rovněž pochází i grafy 3.2 a 4.2. Okno s probíhajícím experimentem můžete vidět na snímku obrazovky F.3.

9.2.4 Průměrné MFCC koeficienty

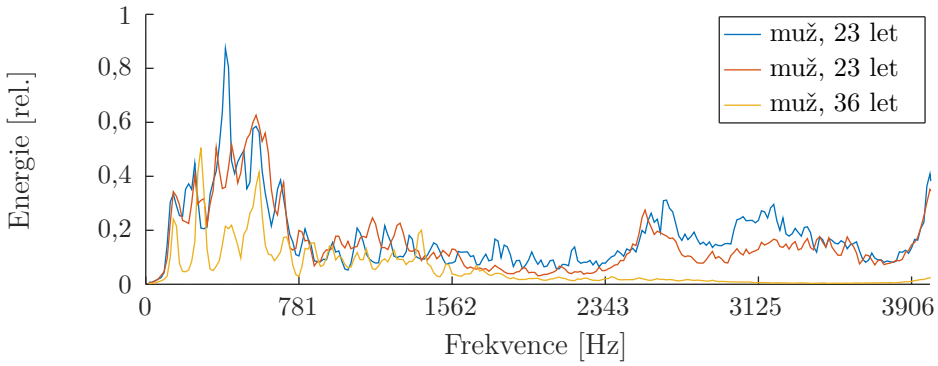
Na rozdíl od experimentu průměrování odhadů spekter, který je popsán v sekci 9.2.3, je zde pro každý segment akustického signálu určen vektor MFCC koeficientů. Výsledkem experimentu je průměr vektorů přes všechny segmenty signálu. Jelikož je výpočet odhadu spektra součástí výpočtu MFCC koeficientů, sdílí experiment většinu konfigurace s experimentem průměrování odhadů spekter. Navíc může uživatel nastavit pouze:

- počet filtrů melovské banky, tj. počet určovaných melovských koeficientů L,
- počet výsledných MFCC koeficientů M,
- a zdali má být vypuštěn první z koeficientů, který je úměrný logaritmu energie signálu [26]. Jeho hodnota je oproti ostatním koeficientům relativně vysoká, což znemožňuje pozorování rozdílů mezi ostatními koeficienty u více mluvčích.

Výsledek experimentu můžete vidět v grafu 9.4. Vykreslené křivky pocházejí z audio nahrávek stejných dobrovolníků jako na obrázku 9.3. V grafu je vidět již minimální, resp. značný rozdíl mezi koeficienty téže, resp. jiné osoby.

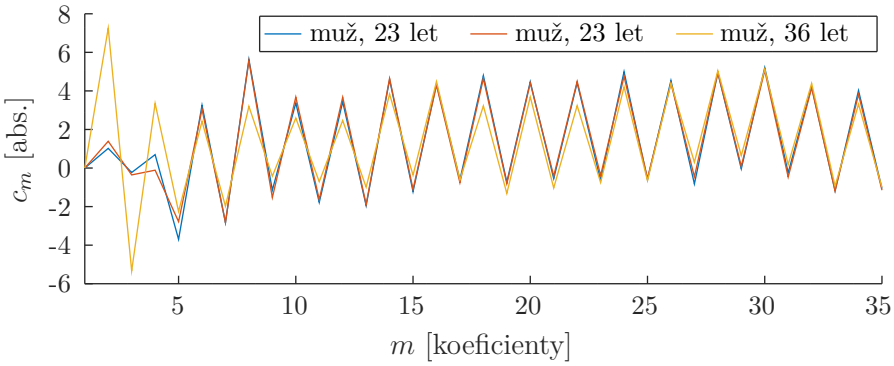


(a) S detekcí znělých segmentů.



(b) Bez detekce znělých segmentů.

Obrázek 9.3: Výsledek průměrování odhadů spekter promluv s a bez detekce znělých segmentů technikou integrování znělé oblasti pro dva dobrovolníky.



Obrázek 9.4: Výsledek experimentu průměrování MFCC koeficientů pro stejné dva dobrovolníky jako v grafu 9.3.

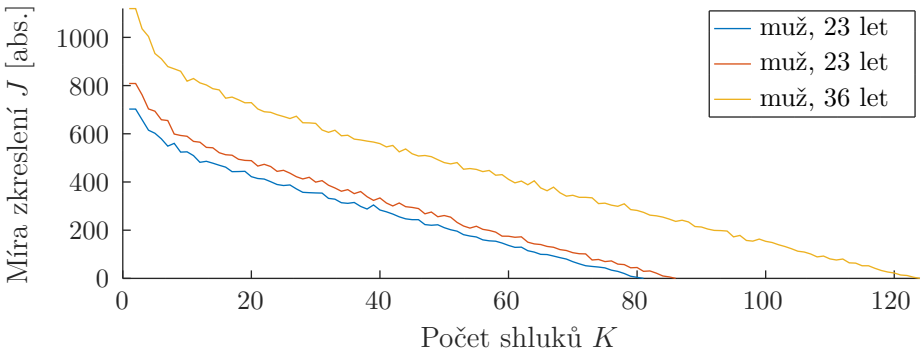
9.2.5 Modus průměrných MFCC koeficientů

Tento experiment byl navržen v návaznosti na úspěšný experiment průměrování MFCC, který je popsán v sekci 9.2.4. Charakteristikami popisujícími osobu jsou zde opět MFCC koeficienty. Místo průměrování přes všechny segmenty jsou vektory rozděleny do daného počtu skupin, pro které průměrování probíhá odděleně. U výsledných průměrů je následně stanoven jejich modus, tj. nejčastěji se vyskytující vektor, jehož počet výskytů je ostře větší než jedna. Určený modus je výsledkem experimentu.

Graf, který by v této sekci demonstroval výsledek experimentu, bohužel neexistuje, jelikož určení reprezentativního vektoru často není možné kvůli kritériu počtu výskytů ostře větší než jedna. Porovnávání koeficientů typu `float` je prováděno voláním funkce `qFuzzyCompare` frameworku Qt. I přes fiasko, kterým tento experiment bezpochyby je, byl v aplikaci ponechán.

9.2.6 Metoda lokte shlukové analýzy nad MFCC

Vektory příznaků extrahované z jednotlivých segmentů řečového signálu, tj. například odhady spekter či MFCC koeficienty lze chápat jako body v n-rozměrném prostoru, kde n je počet složek těchto vektorů. Poslední implementovaný vizualizační experiment je ověřením hypotézy o existenci shluků těchto bodů. Pro kontrolu správnosti této hypotézy bylo využito metody lokte algoritmu K-means++, která je popsána v sekci 5.2.1. Experiment nabízí stejnou volbu konfigurace jako u experimentu 9.2.4. Výsledek je kromě obrázku 5.3 vidět i na grafu 9.5. Z neexistence lokte zde plyne, že body v příznakovém prostoru žádné shluky bohužel netvoří. To autora práce od dalšího experimentování s algoritmem K-means ovšem neodradilo.



Obrázek 9.5: Výsledek metody lokte nad vektory MFCC koeficientů pro stejné dva dobrovolníky jako v grafech 9.3 a 9.4.

10 Navržené algoritmy

Na základě provedených vizualizačních experimentů, které byly integrovány do aplikace *HashVisualisation* popsané v kapitole 9, byla navržena množina algoritmů extrakce dílčího otisku podepisované osoby ze zvukové stopy jejího multimodálního podpisu. Popis jednotlivých algoritmů je uveden v následujících částech kapitoly. Použitelnost vypočítaných dílčích otisků byla předmětem experimentů implementovaných v aplikaci *VoiceHashExps*, která je popsána v kapitole 11. Konkrétní experimenty ukazuje sekce 11.4, souhrnné zhodnocení algoritmů pak sekce 11.5.

Všechny navržené algoritmy vycházejí z odhadů spekter či MFCC koeficientů počítaných z jednotlivých segmentů řečového signálu. První série algoritmů se snaží z nahrávky extrahovat jednoznačné příznaky. Tento způsob se ukázal jako neuskutečnitelný, a tak se dílčím otiskem stala série příznaků společně s konstantním skalárním prahem. Dva otisky jsou pak označeny za ekvivalentní, je-li jejich podobnost vyjádřená stanovenou metrikou nižší než hodnota prahu.

10.1 Průměrování vektorů MFCC

První z navržených algoritmů vychází z úspěšného vizualizačního experimentu průměrování vektorů MFCC koeficientů, který je popsán v sekci 9.2.4. Experiment, jehož výstup je vidět v grafu 9.4, ukazuje minimální, resp. znatelný rozdíl mezi průměrným vektorem koeficientů dvou stejných, resp. různých řečníků. Algoritmus tedy nedělá nic jiného, než že provede uniformní kvantizaci hodnot složek průměrného vektoru MFCC koeficientů. Výsledek kvantizace je pak extrahovaným dílčím otiskem. Hodnota kvantizačního kroku Q a použitelnost algoritmu je určena experimentem popsaným v sekci 11.4.1.

10.2 DTW nad průměry odhadů spekter

První z algoritmů, jehož výsledkem není vektor exaktních příznaků, je založen na technice DTW, která je popsána v sekci 5.1. Výsledným otiskem je zde průměr odhadů spekter, který je zobrazen například v grafu 9.3. Díky tomu, že výpočet odhadu spektra zanedbává fázové posuny jednotlivých spektrálních složek a průměr odstraňuje informaci o počtu segmentů či jejich chro-

nologickém uspořádání, není z tohoto otisku možné rekonstruovat původní zprávu obsahující identitu mluvčího. Průměry odhadů spekter jsou považovány za ekvivalentní, tj. představují totožnou osobu, pokud jejich cena borcení určená algoritmem DTW je menší než stanovený práh. Hodnota prahu a použitelnosti metody na vzorcích datasetu *SignatureDataset* je dána experimentem popsaným v sekci 11.4.2.

10.3 DTW nad průměrným vektorem MFCC

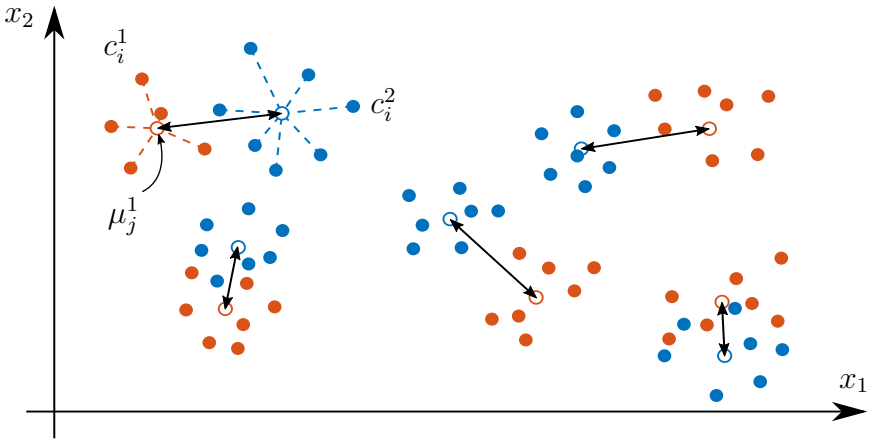
Jedná se o další algoritmus, který byl navržen na základě úspěšného vizualizačního experimentu průměrování vektorů MFCC koeficientů popsaným v sekci 9.2.4. Výsledným dílčím otiskem je průměrný vektor MFCC koeficientů, ze kterého díky filtraci melovskou bankou, zanedbáním jejich části a důvodům popsaným v sekci 10.2 nelze zpětně rekonstruovat původní řečový signál. Dva takto vytvořené otisky jsou označeny za ekvivalentní, pokud jejich podobnost, tj. cena borcení, je menší než konstantní práh, jehož odvozením se zabývá experiment 11.4.2.

10.4 DTW nad odhady spekter

Podobně jako algoritmus popsaný v sekci 10.2 je tento založen na měření podobnosti mezi odhady spekter segmentu řečového signálu dvou řečníků. Rozdílem je absence průměrování těchto odhadů. Výsledným dílčím otiskem je tak vektor vzniklý složením odhadů spekter popořadě za sebe. Dle autorovy bakalářské práce [27], která dokazuje možnost rekonstrukce řečového signálu z odhadů spekter jednotlivých signálů s kvalitou dostatečnou pro rozeznání vyřčených slov, takto vytvořený otisk nesplňuje nutné vlastnosti dílčího otisku, které jsou popsány v kapitole 1. Navíc je délka takto vytvořeného dílčího otisku závislá na délce řečového signálu. I přes tento fakt byla použitelnost algoritmu testována v experimentu, který je podrobně popsán v části textu 11.4.2.

10.5 DTW nad vektory MFCC

Tento algoritmus pracuje na obdobném principu jako v případě algoritmu vytváření dílčího otisku pomocí DTW nad průměrným vektorem MFCC koeficientů. Výsledným otiskem ovšem není průměr vektorů, ale všechny vektory popořadě seřazené za sebe. Stejně jako v případě algoritmu popsaném v sekci 10.4 je na základě poznatků z autorovy bakalářské práce možné



Obrázek 10.1: Ukázka tvorby otisků a principu měření jejich podobnosti pomocí K-means nad 2D příznaky řečníků rozlišených barvami ● a ●. Prázdné značky označují centroidy shluků, které byly umístěny pouze ilustračně.

z vektorů MFCC koeficientů rekonstruovat původní řečový signál přiměřené kvality a délka otisku je závislá na délce řečového signálu. Bylo rozhodnuto, že toto riziko bude eliminováno až v případě, že se algoritmus ukáže jako použitelný, a tak byl navržen experiment, který je popsán v sekci 11.4.2.

10.6 K-means++ nad odhady spekter

Vizualizační experiment popsaný v sekci 9.2.6 sice ukázal na neexistenci shluků v bodech příznaků extrahovaných z řečového signálu řečníků, ale i přesto byl navržen algoritmus tvorby dílčího otisku pomocí algoritmu K-means. Výsledným otiskem je množina K centroidů. Díky konstantní hodnotě K je délka otisku vždy stejná. Kvůli průměrování také není možné z příznaků extrahovat původní řečový signál. Jasnou nevýhodou je nedeterminismus algoritmu K-means, který ovšem nemusí být podmínkou neúspěchu. Podobnost dvou otisků je stanovena pomocí střední kvadratické nebo Čebyševovy vzdálenosti, kdy jsou vzájemně porovnávány vždy nejbližší centroidy. Zjednodušenou situaci ukazuje obrázek 10.1.

10.7 K-means++ nad MFCC

Princip tvorby otisků i určování jejich ekvivalence je shodný s algoritmem K-means++ nad odhady spekter, který je popsán v sekci 10.6. Jediným rozdílem je, že příznaky extrahované ze segmentů řečového signálu nejsou odhady spekter, ale vektory MFCC koeficientů.

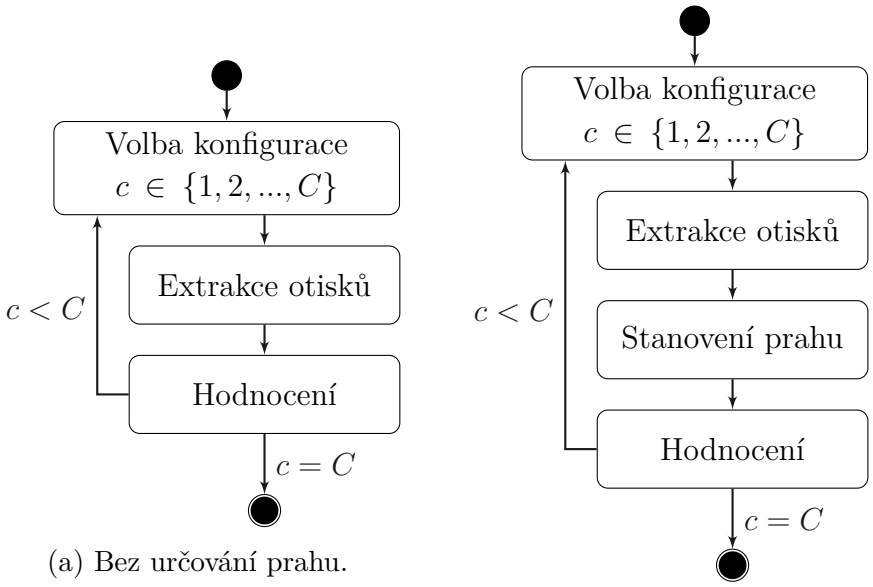
11 Aplikace *VoiceHashExps*

Na základě krátkodobé spektrální analýzy byla navržena možná řešení extrakce dílčího otisku osoby ze zvukové stopy jejího multimodálního podpisu. Tyto algoritmy, které jsou popsány v kapitole 10, bylo třeba podrobit důkladnému testování, k čemuž vznikla aplikace *VoiceHashExps*. Obecný popis všech experimentů je uveden v sekci 11.1.

Ve všech případech je otisk dán jedním nebo množinou extrahovaných příznakových vektorů. Podle povahy algoritmu je pak ekvivalence dvou otisků určována přímo podle rovnosti složek vektorů nebo s využitím specializované techniky a skalární prahové hodnoty. Navržené algoritmy podle použitých technik a extrahovaných příznaků tvoří hierarchii, kterou odráží i struktura zdrojového kódu aplikace popsaná v sekci 11.3. Samotný popis jednotlivých experimentů je pak ukázán v oddílu 11.4. Finální zhodnocení algoritmů je nakonec popsáno v sekci 11.5.

11.1 Průběh obecného experimentu

Experimenty jsou založeny na principu iterativního procházení zadaných konfigurací hyperparametrů jednotlivých algoritmů. Dataset *SignatureDataset* je náhodně v poměru 4:1 rozdělen na tzv. *trénovací* a *testovací vzorky*. Během každé iterace je využito testovaného algoritmu k extrakci otisků ze zvukových stop všech načtených vzorků datasetu. Průběh se mění podle toho, zda algoritmus vyžaduje určení prahové hodnoty pro porovnání dvou otisků. Prahová hodnota je určena jako maximum ze vzdáleností mezi otisky týchž osob přes všechny dobrovolníky, jejichž akustické podpisy náleží trénovací množině. Tím je zaručena vlastnost dílčího otisku o ekvivalenci dvou otisků téže osoby alespoň v trénovacích datech. Konfigurace algoritmu je nakonec na základě vypočítaných dílčích otisků zhodnocena, o čemž pojednává sekce 11.1.1. Výsledky hodnocení jsou uváděny v příslušném `.csv` souboru či v podrobných textových souborech, které obsahují i konkrétní vypočítané otisky. U experimentů, které jsou z důvodu nedeterminismu navrženého algoritmu spouštěny vícekrát, je tato možnost zakázána, protože velikost výstupního adresáře často dosahovala desítek gigabytů. Soubory s podrobným hodnocením, resp. určováním prahu jsou automaticky ukládány do podadresáře `evaluations`, resp. `measurements`. Blokové diagramy popisující průběh experimentů ukazuje obrázek 11.1.



(a) Bez určování prahu.

(b) S určováním prahu.

Obrázek 11.1: Schéma průběhu testování jednotlivých konfigurací algoritmů aplikací *VoiceHashExps* se stanovením prahové hodnoty pro určování ekvivalence otisků a bez ní.

11.1.1 Hodnotící kritéria

S každou procházenou konfigurací hyperparametrů testovaného algoritmu vzniká množina dílčích otisků, u kterých jsou ověřovány jejich nutné vlastnosti popsané v kapitole 1. Konfigurace jsou v rámci hodnocení známkovány podobně jako na vysoké škole. Testování probíhá v dále popsaných etapách.

1. Vyhodnocení, zda u všech trénovacích otisků platí tvrzení o jejich ekvivalenci pro tutéž osobu. Pokud neplatí, další testování již nemá význam, vzorky jsou uvolněny z paměti a konfigurace je hodnocena známkou 4. V opačném případě je hodnocením konfigurace známka 3 a testování probíhá etapou 2.

2. Určení procenta nerozlišitelných otisků různých dobrovolníků. U každého otisku je testována jeho nonekvivalence se všemi otisky ostatních dobrovolníků. Výsledkem je hodnota $e \in \langle 0, 1 \rangle$, která je dána poměrem ekvivalentních otisků vůči všem testovaným. Lepší je ta konfigurace, jejíž hodnota je bližší nule. V případě, že $e < 0{,}2$, konfigurace získává známku 2.

3. Stanovení procenta správně verifikovaných testovacích vzorků. Testována je ekvivalence otisků testovacích vzorků s ostatními otisky téhož

dobrovolníka. Výsledná hodnota $s \in \langle 0, 1 \rangle$ je dána poměrem počtů správně verifikovaných a všech provedených porovnání. Žádoucí je, aby hodnota s byla blízká jedné. Pokud je $e = 0$ a zároveň $s = 1$, konfigurace dostává známku 1.

Ve výstupních `.csv` souborech jsou pro každou konfiguraci uvedeny hodnoty e („Nerozlišitelných“) a s („Test“). Podle typu algoritmu soubor obsahuje i hodnotu prahu $p \in \langle 0, +\infty)$. U nedeterministických algoritmů, tj. algoritmy založené na technice K-means, je test každé konfigurace prováděn dvacetkrát. Soubor tak kromě průměrů $\overline{e}$ a $\overline{s}$ obsahuje i jejich rozptyly.

11.2 Označení hyperparametrů algoritmů

Navržené algoritmy vyžadují nastavení množiny hyperparametrů. Platná množina těchto parametrů je v textu označováno jako konfigurace algoritmu. Tabulka 11.1 představuje výčet všech hyperparametrů navržených algoritmů. Uvedená alternativní označení využívá aplikace *VoiceHashExps*, zatímco výrazy v prvním sloupci jsou užívány v dalších částech textu diplomové práce.

Tabulka 11.1: Výčet možných hyperparametrů všech navržených algoritmů.

Hyperparametr	**Alternativně**	**Význam**
$N \in \mathbb{N}$	`ss`	počet vzorků segmentů
$O \in \langle 0, 1)$	`so`	procentuální překryv segmentů
$L \in \mathbb{N}$	`fc`	počet melovských koeficientů
$M \in \{1, ..., L\}$	`mc`	počet MFCC koeficientů
$Q \in (0, +\infty)$	`q`	krok při kvantizaci příznaků
$\eta \in (0, 1\rangle$	`t`	práh při detekci řeči
$V \in \{0, 1\}$	`only_voiced`	příznak detekce znělých segmentů
$K \in \mathbb{N}$	`k`	počet hledaných shluků

11.3 Struktura zdrojového kódu

Aplikace *VoicehashExps* je napsána v jazyce C++ s využitím knihovny Qt. Kód je silně objektově orientován a využívá dynamického polymorfismu. Pro každý z navržených algoritmů byl vytvořen experiment obalený příslušnou třídou. Díky společným vlastnostem experimentů tvoří schéma dědičnosti

odpovídajících tříd stromovou strukturu vyobrazenou v diagramu F.5 přílohy dokumentu. Kořenem stromu je třída `IExperiment`, která obsahuje veškeré metody společné pro všechny experimenty. Jejími potomky jsou třídy rozdělující experimenty podle použité techniky, protože každá z technik byla testována odlišně. Například algoritmy založené na technice K-means byly kvůli svému nedeterminismu spouštěny vícekrát. Listy stromu jsou nakonec konkrétními implementacemi extrakce dílčího otisku. Výrazy `ConfT` a `HashT` jsou šablonové parametry. `ConfT` označuje nespecifikovanou třídu konfigurace testovaného algoritmu a `HashT` představuje datový typ prvku příznakového vektoru, tj. dílčího otisku. V případě experimentů založených na technice K-means značí `HashT` datový typ souřadnice bodu v příznakovém prostoru. Z diagramu je zřejmé, že v poslední, barevně odlišené úrovni stromové struktury jsou již tyto šablonové parametry nahrazeny konkrétními třídami konfigurace a datovými typy. Popisy tříd určených pro získání prahové hodnoty nutné pro stanovení ekvivalence dvou otisků nebo samotné hodnocení konfigurací jsou k dispozici v sekci 11.3.1.

11.3.1 Důležité třídy

`IExperiment<ConfT>`

Třída je společným předkem všech implementovaných experimentů. Konstruktor třídy očekává následující parametry.

- Umístění adresáře se zvukovými záznamy datasetu *SignatureDataset* ve formátu `.raw`, kde vzorky musejí být 4-bytové s malým endianem a frekvencí vzorkování $f_v = 8000$ Hz.
- Umístění adresáře, kam budou vypsány výsledky provedeného experimentu. Pokud zadaný adresář již existuje, je před spuštěním dalšího experimentu vyprázdněn.
- Instance třídy `ConfigVector<ConfT>` obsahující vektor všech testovaných konfigurací.
- Procentuální část testovacích vzorků. Výchozí hodnota je 20 %.

Úlohou metod třídy je načtení datasetu ze zadaného umístění a jeho náhodné rozdělení na trénovací a testovací vzorky. Přístup k obou částem načteného datasetu či konfiguracím je potomkům třídy umožněn skrze množinu chráněných (`protected`) metod. Metoda rovněž zabezpečuje výpis načtených vzorků datasetu či aktuálně testované konfigurace do konzolového výstupu.

`AudSignHash<HashT>`

Tato třída slouží jako obal vypočítaných dílčích otisků podle návrhového vzoru *Cradle (Přepravka)* [24]. Je potomkem třídy `SignatureSample` knihovny `SignatureDatasetLib`. Díky tomu obsahuje i všechny informace o dobrovolníkovi, jemuž otisk náleží. Prvky příznakového vektoru, který je otiskem dobrovolníka, jsou datového typu `HashT`.

`DistanceEvaluator<ConfT, HashT>`

Instance této třídy mají za úkol stanovit práh, při kterém platí ekvivalence otisků téže osoby, což je hlavní vlastnost navrhovaného dílčího otisku. Konstruktor očekává objekt třídy `std::function`, tj. ukazatel na funkci nebo statickou metodu, který určí vzdálenost dvou zadaných otisků. Druhým parametrem je pak umístění výstupního adresáře, do jehož podadresáře `measurements` metody třídy ukládají hlášení o stanovení prahů. Příklad inicializace objektu této třídy je zobrazen ve zdrojovém kódu 11.3. Vektor zpracovávaných otisků je plněn průběžně prostřednictvím veřejných metod. Při zapsání posledního otisku implementace experimentů volají metodu `measConfig`, která provede určení prahu jako maximální vzdálenosti mezi otisky jednotlivých dobrovolníků. Výsledkem metody je instance struktury `DistanceEvaluator<ConfT, HashT>::Measurent`, která obsahuje stanovený práh, konfiguraci algoritmu a identifikátor dobrovolníka s největší vzdáleností. Tato měření jsou ukládána za účelem tvorby finálního hlášení na konci experimentu.

`SignatureEvaluator<ConfT, HashT>`

Metody této třídy hodnotí algoritmy v daných konfiguracích podle postupu definovém v sekci 11.1.1. Konstruktor očekává objekt třídy `std::function`, tj. ukazatel na funkci či statickou metodu, pomocí které je možné určit ekvivalenci dvou zadaných otisků. Druhým parametrem je umístění výstupního adresáře, kam budou případně ukládána podrobná hlášení včetně samotných otisků.

`AlgConfig`

Jedná se o ryze virtuálního předka (rozhraní) všech tříd konfigurací. Třída předepisuje metody `toString` a `isValid`. Účelem první z vytvořených metod je převod hodnot hyperparametrů do řetězce znaků, který je čitelný pro člověka. Konkrétní implementace rozhraní k označování hyperparametrů využívají alternativních názvů uvedených v tabulce 11.1. Implementace

druhé metody ověřují, že nastavené hodnoty hyperparametrů jsou validní, tj. velikost překryvu je ostře menší než jedna, počet MFCC koeficientů M je ostře menší než počet filtrů melovské banky L apod. Platí, že třídy v textu označené šablonovým parametrem `ConfT` jsou potomky této třídy.

`ConfigVector<ConfT>`

Během experimentů byly testovány v některých případech i desítky tisíc možných konfigurací algoritmu. Tato třída využívá techniky tzv. *variadických šablon* [25] k tomu, aby jejich tvorbu učinila přehlednou a snadnou. Všechny třídy konfigurací jsou vytvořeny podle návrhového vzoru přepravka, což znamená, že pro jejich konstrukci je třeba zadání všech H hyperparametrů. Podle třídy konfigurace, která je dána šablonovým typem `ConfT`, tedy konstruktor třídy *ConfigVector* vyžaduje H vektorů možných hodnot jednotlivých hyperparametrů. Kartézským součinem pak vzniká vektor všech testovaných konfigurací, jejichž validita je zaručena voláním virtuální metody `AlgConfig::isValid`. Možnou inicializaci vektoru konfigurací ukazuje zdrojový kód 11.1.

```
ConfigVector<DTWESPDConfig> configurations(
   QVector<double>() << 0.1,                  // t
   QVector<int>() << 256 << 512 << 1024,      // ss
   QVector<float>() << 0.0f << 0.25f << 0.5f, // so
   QVector<bool>() << false << true           // only_voiced
);

DTWESPDMeanExp mfccExp(..., configurations, ..., ...);
mfccExp.run();
```

Zdrojový kód 11.1: Možná inicializace objektu třídy `ConfigVector`.

11.4 Provedené experimenty a jejich výsledky

Následující sekce textu obsahují konkrétní údaje o experimentech, které byly provedeny nad navrženými algoritmy s různými hyperparametry. U každého experimentu je uvedena tabulka nejlepších výsledků. Souhrnné hodnocení navržených algoritmů je uvedeno v sekci 11.5. Obecný postup experimentu aplikace *VoiceHashExps* popisuje sekce 11.1. Výsledky experimentů jsou uloženy v adresáři `data/output`. Souborů s podrobným hlášením je pro ilustraci v adresáři jen několik kvůli jejich značnému datovému objemu.

11.4.1 Průměrování vektorů MFCC

Prvním a také posledním algoritmem, který pro určení ekvivalence nepoužívá prahovou hodnotu, je algoritmus průměrování MFCC koeficientů, který je popsán v sekci 10.1. Experiment představuje třída `MFCCExp` s třídou konfigurace `MFCCExpConfig`. Testováno bylo celkově 19 818 konfigurací, které jsou tvořeny hyperparametry:

- $\eta = 0{,}1$,
- $L \in \{20, 23, ..., 173\}$,
- $N \in \{256, 512\}$,
- $M \in \{5, 10, ..., 110\}$,
- $O \in \{0; 0{,}25; 0{,}5\}$,
- a $Q \in \{0{,}05; 0{,}1; 0{,}2\}$.

Výsledky experimentu, včetně několika pro ukázku ponechaných podrobných výpisů, jsou uloženy v adresáři `brute_mfcc`. Hodnocení algoritmu bohužel skončilo známkou 4, protože pro žádnou z testovaných konfigurací nebylo splněno tvrzení o ekvivalenci otisků téže osoby. Zdrojový kód 11.2 ilustruje inicializaci objektu třídy `SignatureEvaluator` s použitou funkcí pro určení ekvivalence otisků.

```
SignatureEvaluator<MFCCExpConfig, float> evaluator(
   [](const AudSignHashPtr<float>& s1,
         const AudSignHashPtr<float>& s2) -> bool {
      // pro jednoduchost neni overena rovnost dimenze otisku

      for (int i = 0; i < s1->hash().size(); ++i) {
         if (!qFuzzyCompare(s1->hash()[i], s2->hash()[i]))
            return false;
      }

      return true;
   },
   "cesta/k/vystupnimu/adresari"
);
```

Zdrojový kód 11.2: Ilustrace inicializace objektu třídy `SignatureEvaluator`.

11.4.2 Algoritmy založené na technice DTW

Do této kategorie spadají experimenty zaobalené ve třídách `DTWESPDExp` a `DTWESPDMeanExp`, které využívají konfigurace `DTWESPDConfig`, a třídách `DTWMFCCExp`, `DTWMFCCMeanExp` využívajících konfigurace třídy `DTWMFCCConfig`. Skupiny tříd jsou v předchozí větě záměrně rozděleny podle extrahovaného dílčího otisku. Část názvu „`Mean`“ pak označuje experimenty, které nevyužívají spojování příznakových vektorů, ale jejich průměru. V experimentech bylo využito následujících konfigurací:

- $\eta = 0{,}1$,
- $N \in \{256, 512, 1024\}$,
- $O \in \{0; 0{,}25; 0{,}5\}$,
- $L \in \{20, 23, ..., 173\}$,
- $M \in \{5, 8, ..., 110\}$
- a $V \in 0, 1$.

Během všech čtyř experimentů bylo testováno celkem 23 508 různých konfigurací. Výsledky nejlepších konfigurací jsou uvedeny v tabulce 11.2. Pro určení prahu p bylo využito třídy `DistanceMeter`, jejíž inicializace je ilustrována ve zdrojovém kódu 11.3.

Tabulka 11.2: Nejlepší konfigurace algoritmů založených na DTW (seřazeno dle hodnoty e).

Agregační metoda	***e***	***p***	***s***	**Známka**
spojováním	0,94	$1{,}51 \cdot 10^{12}$	0,96	3
odhadů	0,94	$2{,}31 \cdot 10^{12}$	0,92	3
spekter	0,95	$3{,}16 \cdot 10^{12}$	0,96	3
spojováním	0,97	1310,19	0,96	3
vektorů	0,97	1305,65	1	3
MFCC	0,97	1337,89	1	3
průměrováním	0,84	$2{,}44 \cdot 10^{10}$	1	3
odhadů	0,85	$6{,}35 \cdot 10^{10}$	0,96	3
spekter	0,85	$6{,}35 \cdot 10^{10}$	0,96	3
průměrováním	0,18	38,62	0,96	2
vektorů	0,18	38,33	0,96	2
MFCC	0,18	38,05	0,96	2

```
DistanceMeter<ConfT, float> meter(
   [](const AudSignHashPtr<float>& sign1,
         const AudSignHashPtr<float>& sign2) -> float {
      DTW<float> dtw([](float a, float b) -> double {
         return std::abs(static_cast<double>(a - b));
      });

      return static_cast<float>(
         dtw.calcWarpCost(sign1->hash(), sign2->hash())
      );
   },
   "cesta/k/vystupnimu/adresari"
);
```

Zdrojový kód 11.3: Inicializace objektu třídy `DistanceEvaluator` v experimentech s technikou DTW.

11.4.3 Algoritmy založené na technice K-means++

Poslední rodina algoritmů extrakce dílčích otisků je založena na technice K-means++. Algoritmy jsou popsány v sekcích 10.6 a 10.7. Příznakovými vektory, které představují analyzované body, jsou buď odhady spekter jednotlivých segmentů nebo vektory MFCC koeficientů. Podle toho jsou rozlišovány experimenty obalené třídami `KMeansMFCCExp` a `KMeansESPDExp`, které používají třídy konfigurace `KMeansMFCCConfig` a `KMeansESPDConfig`.

Pro měření vzdálenosti mezi dvěma otisky při určování jejich ekvivalence byla použita Čebyševova metrika, jejíž výsledky nebyly uspokojivé, a tak byla z finální podoby práce odstraněna. Její implementace v kódu ovšem stále existuje. Druhým způsobem je střední kvadratická chyba, tj. průměr druhých mocnin Eukliedových vzdáleností vždy dvou nejbližších centroidů – centroidy druhého otisku jsou postupně přiřazovány k centroidům prvního podle algoritmu K-means. Situaci popisuje obrázek 10.1. Poslední, čistě experimentální postup, používá pro určení „vzdálenosti“ mezi centroidy při určování ekvivalence otisků techniku DTW, a to i přes fakt, že sekce 7.3.7 mluví o tom, že výsledná cena borcení nesplňuje všechny axiomy definující metriku. Z výsledků DTW je opět vypočtena střední kvadratická chyba.

Během těchto experimentů bylo testováno 70 140 různých konfigurací. Hyperparametry byly postupně nastavovány na následující hodnoty:

- $\eta = 0{,}1$,
- $N \in \{256, 512, 1024\}$,
- $O \in \{0; 0{,}25; 0{,}5\}$,
- $L \in \{20, 25, ..., 170\}$,
- $M \in \{5, 10, ..., 110\}$,
- $K \in \{5, 8, ..., 50\}$
- a $V = 0$.

Nejlepší výsledky jednotlivých metod ukazuje tabulka 11.3.

Tabulka 11.3: Nejlepší konfigurace algoritmů založených na K-means (seřazeno dle hodnoty $\overline{e}$).

Příznak	**Ekviv.**	$\overline{e}$	σ_e	$\overline{p}$	σ_p	$\overline{s}$	σ_s	**Zn.**
vektory MFCC	střední	0,18	0	52,03	2,72	0,96	0	2
	kvadr.	0,18	0	55,00	3,40	0,96	0	2
	chyba	0,18	0	74,51	6,38	0,96	0	2
odhad spektra	střední	0,99	0	$1{,}85 \cdot 10^{21}$	$1{,}44 \cdot 10^{41}$	1	0	3
	kvadr.	0,99	0	$1{,}70 \cdot 10^{21}$	$2{,}42 \cdot 10^{41}$	1	0	3
	chyba	0,99	0	$1{,}58 \cdot 10^{21}$	$9{,}23 \cdot 10^{40}$	1	0	3
vektory MFCC	DTW	0,15	0	922,72	503,07	0,93	0	2
		0,15	0	1227,10	280,90	0,95	0	2
		0,16	0	978,35	435,68	0,95	0	2
odhad spektra	DTW	0,98	0	$6{,}85 \cdot 10^{22}$	$5{,}27 \cdot 10^{40}$	1	0	3
		0,98	0	$8{,}06 \cdot 10^{22}$	$3{,}06 \cdot 10^{42}$	1	0	3
		0,98	0	$1{,}78 \cdot 10^{22}$	$2{,}42 \cdot 10^{42}$	1	0	3

11.4.4 Metoda lokte techniky K-means++

Poslední dvojice implementovaných experimentů pro každou testovanou konfiguraci určí optimální počet shluků pomocí metody lokte, která je popsána v sekci 5.2.1. Výsledné křivky jsou zapsány do skriptů, které lze spustit pomocí nástroje *Octave*. Tyto experimenty byly kvůli pohodlí uživatele přesunuty do aplikace *HashVisualisation*, ale i zde byly ponechány. Vizualizační experiment je popsán v sekci 9.2.6.

11.5 Zhodnocení algoritmů

Provedené experimenty, které jsou popsány v sekci 11.4, ukazují neschopnost navržených algoritmů extrahovat takové otisky akustických podpisů, které by byly pro dobrovolníky unikátní. Z toho vyplývá nutnost existence prahové hodnoty používané při určení ekvivalence dvou otisků.

Algoritmy užívající prahovou hodnotu jsou založeny na technikách DTW, K-means++ nebo kombinaci obou. V experimentech byly jako otisky využity jak odhady spekter segmentů řečového signálu, tak vektory MFCC koeficientů. Zatímco algoritmy využívající odhady spekter dosahují nejlepších výsledků při hodnotě poměru nerozlišitelných otisků $e = 0{,}94$ (algoritmus DTW nad průměrným odhadem spektra), MFCC koeficienty při použití stejné techniky tento poměr posouvají až na hodnotu $e = 0{,}18$. Tento značný rozdíl je viditelný i u algoritmů založených na technice K-means++. Na základě experimentů je tedy možné říci, že užití odhadů spekter jako otisků není vhodné. Potvrzuje se tak i výsledek vizualizačních experimentů, které je popsány v sekcích 9.2.3 a 9.2.4.

Zatímco DTW je deterministický algoritmus, tj. pro stejnou dvojici akustických podpisů vrátí vždy stejný výsledek, inicializace techniky K-means++ je založena na rovnoměrném rozložení pravděpodobnosti výběru počátečních poloh centroidů. Experimenty s touto technikou byly proto provedeny s dvaceti opakováními a z jejich výsledků pak byl vypočten průměr a rozptyl. Z tabulek 11.3 a 11.2 je patrné, že při použití průměru MFCC koeficientů jako otisku osoby dosahuje technika K-means++ s nejlepší hodnotou $\overline{e} = 0{,}18$ stejných výsledků jako algoritmy založené na technice DTW. Hodnota rozptylu σ_e je blízká nule a rozptyl σ_p je rovněž minimální. Z podrobných výsledků experimentů, které jsou k dispozici na přiloženém datovém nosiči, je patrné, že nejlepších výsledků dosahují algoritmy při velikosti segmentů $N = 256$ a počtu melovských, resp. MFCC koeficientů blízko hodnotám $L = 120$, resp. $M = 110$. Experimenty také prokázaly, že detekce znělých segmentů pomocí techniky popsané v sekci 5.4.2 ve všech případech pouze zvyšuje hodnotu ukazatele e. Poslední a také dle hodnoty $\overline{e} = 0{,}15$ nejlepší algoritmus je založen na kombinaci obou technik. V tabulce 11.3 je ovšem důležité si všimnout vysokých hodnot σ_p, které potvrzují nevhodnost užití techniky DTW při určování ekvivalence dvou otisků.

Na základě provedených experimentů byl pro své výsledky, determinismus a potvrzenou neexistenci shluků v bodech příznakového prostoru vybrán algoritmus založený na technice DTW, využívající jako otisk osoby průměrný vektor MFCC koeficientů.

12 Závěr

První kapitola této diplomové práce stručně popisuje základní metody užívané při rozpoznávání uživatele podle 2D obrazu jeho obličeje. Verifikace osoby pomocí volumetrického modelu či termografického snímku je popsána pouze velmi stručně, protože projekt vyžaduje nasazení technologie na běžných mobilních telefonech, které potřebnou technologií obvykle nedisponují. Je vhodné připomenout, že analýza vizuální komponenty videozáznamu není součástí této diplomové práce a v rámci výzkumného projektu ji řešil Bc. Patrik Patera.

Další část textu práce se zabývá podrobnou analýzou biometrických charakteristik řečového signálu, které vycházejí z popsaného procesu jeho tvorby člověkem. Je zde rovněž popsána znělost hlásek a s ní spojená struktura formantových frekvencí.

Řečový signál byl analyzován jak v časové, tak frekvenční oblasti. Pomocí navržené techniky naivní detekce řečové aktivity byly detekovány segmenty akustického signálu s řečovou aktivitou, což se v nadcházejících experimentech ukázalo jako klíčové. Pro parametrizaci odhadů výkonových spektrálních hustot byla použita technika mel-frekvenčních kepstrálních koeficientů (MFCC). Na základě konzultací s vedoucím diplomové práce byly provedeny experimenty s technikami DTW, K-means++ a detekcí znělých segmentů řečového signálu, které jsou v textu rovněž podrobně rozebrány.

Všechny dosud uvedené techniky byly implementovány v knihovně pro C++ s názvem `libpe`. Knihovna obsahuje třídy pro předzpracování signálu (preemfáze a odstranění DC ofsetu), tvorbu segmentů signálu zadané délky a překryvu, jejich váhování pomocí Hammingova okna, spektrální analýzu užívající knihovny *KissFFT* a výpočet MFCC koeficientů. Knihovna dále obsahuje implementaci navržené naivní detekce řečové aktivity či znělých segmentů. Nechybí ani implementace technik DTW a K-means++. Ke všem implementovaným algoritmům vznikly i jednotkové testy, které ověřují správnost prováděných výpočtů. Třídy knihovny jsou šablonové, proto například změna datového typu pořízených vzorků řečového signálu vyžaduje pouze minimální změnu zdrojového kódu aplikace.

Za účelem testování navržených algoritmů byla pomocí vytvořené mobilní aplikace *VideoCollector* rozšířené mezi pět přispěvatelů (pěti různými mobilními telefony) nashromážděna množina multimodálních podpisů čítající

celkově 153 videozáznamů od 34 dobrovolníků (15 žen a 19 mužů), která byla označena jako *SignatureDataset*. Všichni dobrovolníci podepsali tzv. *Poučení o zpracování osobních údajů*. Videozáznamy obsahují detail obličeje dobrovolníka, který vyřkne předem definovanou větu. Pro snadnou a unifikovanou práci se vzorky datasetu byla vytvořena i knihovna pro C++ nazvaná jako `SignatureDatasetLib`.

Dalším krokem při návrhu algoritmů pro extrakci dílčích otisků ze zvukových stop byla vizualizace z nich extrahovaných charakteristik. Vznikla tak aplikace s grafickým rozhraním, která byla nazvána *HashVisualisation*. V této aplikaci si uživatel může vyzkoušet navržené vizualizační experimenty přímo na vzorcích nashromážděného datasetu. Jejich výsledkem bylo ověření navržené techniky naivní detekce řečové aktivity a demonstrace vysokého, resp. nízkého rozptylu odhadů spekter téže, resp. různých osob. Na druhou stranu experimenty ukázaly zcela opačný rozptyl při použití vektorů MFCC koeficientů. Předmětem dalšího vytvořeného experimentu je metoda lokte techniky K-means++. Experiment na základě neexistence lokte u všech vzorků datasetu prokázal rovnoměrné rozložení bodů v příznakovém prostoru.

Podle provedených vizualizačních experimentů byla navržena množina algoritmů pro extrakci dílčích otisků ze zvukových stop multimodálních podpisů osob. Při verifikaci osoby podle extrahovaných otisků se nejdříve využívalo přímého porovnávání koeficientů. Další možností bylo ověření ekvivalence otisků pomocí stanovené konstantní prahové hodnoty. „Vzdálenost“ dvou otisků byla stanovena technikami DTW, K-means++ či kombinací obou.

Pro testování navržených algoritmů byla vytvořena konzolová aplikace napsaná v jazyce C++. Úlohou aplikace bylo procházení desítek tisíc možných konfigurací navržených algoritmů. Pro každou z nich aplikace pomocí testovaného algoritmu extrahovala dílčí otisky ze zvukových stop všech vzorků nashromážděného datasetu. Kvůli značné výpočetní náročnosti byl celý výpočet paralelizován pomocí knihovny *Qt Concurrent*. Experimenty ukázaly, že využití agregovaných odhadů spekter kvůli jejich vysokému vnitrotřídnímu rozptylu nesplňuje požadovaný účel. Na druhou stranu algoritmy pracující s vektory MFCC koeficientů snížily podíl nerozlišitelných otisků až na hodnotu $e = 0{,}15$. Kvůli tomu, že vizualizační experimenty prokázaly rovnoměrné rozložení bodů v příznakovém prostoru, byl nakonec vybrán deterministický algoritmus založený na technice DTW, využívající jako otisk osoby průměrný vektor MFCC koeficientů. Tento algoritmus nedokáže rozlišit 18 % otisků vzorků datasetu a při testování správně ověřil 96 % testovacích otisků.

Literatura

[1] *Courseware KIV Strojové učení* [online]. [cit. 2020-03-27]. Dostupné z: https://courseware.zcu.cz/portal/studium/courseware/kiv/su.

[2] *DC offset* [online]. [cit. 2020-03-18]. Dostupné z: https://manual.audacityteam.org/man/dc_offset.html.

[3] *Doxygen* [online]. [cit. 2020-03-20]. Dostupné z: http://www.doxygen.nl/.

[4] *FFmpeg* [online]. [cit. 2020-03-24]. Dostupné z: https://www.ffmpeg.org/.

[5] *Kiss FFT* [online]. [cit. 2020-02-18]. Dostupné z: https://sourceforge.net/projects/kissfft/.

[6] *Android MediaStore* [online]. [cit. 2020-04-12]. Dostupné z: https://developer.android.com/reference/android/provider/MediaStore.

[7] *GNU Octave: Scientific Programming Language* [online]. [cit. 2020-03-30]. Dostupné z: https://www.gnu.org/software/octave/.

[8] *Courseware KIV Paralelní programování* [online]. [cit. 2020-03-30]. Dostupné z: https://courseware.zcu.cz/portal/studium/courseware/kiv/ppr.

[9] *Python documentation* [online]. [cit. 2020-03-30]. Dostupné z: https://docs.python.org/.

[10] *Qt Documentation* [online]. [cit. 2020-03-30]. Dostupné z: https://doc.qt.io/.

[11] *Qt Concurrent* [online]. [cit. 2020-03-30]. Dostupné z: https://doc.qt.io/qt-5/qtconcurrent-index.html.

[12] *Signals & Slots* [online]. [cit. 2020-03-30]. Dostupné z: https://doc.qt.io/qt-5/signalsandslots.html.

[13] *Qt Test* [online]. [cit. 2020-03-30]. Dostupné z: https://doc.qt.io/qt-5/qttest-index.html.

[14] *RAII* [online]. [cit. 2020-03-20]. Dostupné z: https://en.cppreference.com/w/cpp/language/raii.

[15] Arthur, D. – Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, s. 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

[16] Bachu, R. G. et al. Separation of Voiced and Unvoiced Speech Signals using Energy and Zero Crossing Rate. University of Bridgeport, Electrical Engineering Department, 2008.

[17] Bartsch, H.-J. *Matematické vzorce*. Academia, 2006. ISBN 80-200-1448-9.

[18] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag Berlin, 2007. ISBN 0387310738.

[19] Cooley, J. – Tukey, J. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*. 1965, 19, 90, s. 297–301.

[20] Drahanský, M. – Doležal, M. – Orság, F. *Biometrie*. M. Drahanský, 2011. ISBN 978-80-254-8979-6.

[21] Ekštein, K. *Hybrid methods of acoustic-phonetic analysis of spontaneous speech [disertační práce]*. University of West Bohemia, Faculty of Applied Sciences, 2004.

[22] Hughes, T. – Mierle, K. *Recurrent neural networks for voice activity detection*. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 7378-7382.

[23] Komenda, M. Podobnosti a vzdálenosti ve vícerozměrném prostoru. Masarykova univerzita, Institut biostatistiky a analýz Lékařské fakulty, 2015.

[24] Pecinovský, R. *Návrhové vzory*. Computer Press, 2015. ISBN 978-80-251-1582-4.

[25] Prata, S. *Mistrovství v C++ [4. vyd.]*. Computer Press, 2013. ISBN 978-80-251-3828-1.

[26] Psutka, J. et al. *Mluvíme s počítačem česky*. Academia, 2006. Dostupné z: `http://www.kky.zcu.cz/en/publications/PsutkaJ_2006_Mluvimes`. ISBN 80-200-1309-1.

[27] Pártl, F. *Software pro mobilního klienta a pro operátorskou konzoli systému PocketEAR [bakalářská práce]*. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, 2018.

[28] ROKYTA, R. – MAREŠOVÁ, D. – TURKOVÁ, Z. *Somatologie: učebnice*. Wolters Kluwer ČR, 2016. ISBN 978-80-7552-306-8.

[29] Senin, P. Dynamic Time Warping Algorithm Review. 01 2009.

[30] Smith, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997. ISBN 0966017633.

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

A Zkratky

AAM Active Appearance Model (statistický model tvaru a textury objektu)

ASM Active Shape Model (statistický model tvaru objektu)

DC Direct Current (označení pro stejnosměrný elektrický proud)

DCT Discrete Cosine Transform (diskrétní kosinová transformace)

DFT Discrete Fourier Transform (diskrétní Fourierova transformace)

DTW Dynamic Time Warping (algoritmus měření podobnosti mezi dvěma signály)

FFT Fast Fourier Transform (rychlá Fourierova transformace)

IEC International Electrotechnical Commission (organizace vydávající normy pro elektrotechnické a příbuzné obory)

ISO International Organization for Standardization (světová federace normalizačních organizací)

IDFT Inverse Discrete Fourier Transform (inverzní diskrétní Fourierova transformace)

LDA Linear Discriminant Analysis (metoda snížení dimenze prostoru)

MFCC Mel-Frequency Cepstral Coefficients (metoda parametrizace odhadu výkonové spektrální hustoty krátkodobého segmentu akustického signálu)

PDF Portable Document Format (souborový formát pro ukládání dokumentů)

PCA Principal Component Analysis (metoda snížení dimenze prostoru)

PCM Pulse-Code Modulation (modulační metoda analogově/digitálního převodu)

RNN Recurrent Neural Network (třída umělých neuronových sítí)

RAII Resource Acquisition Is Initialization (idiom popisující chování programovacího jazyka)

SVD Singular Value Decomposition (metoda rozkladu matice)

SW Software (programové vybavení informačních nebo komunikačních technologií)

STL Standard Template Library (standardní knihovna jazyka C++)

ZCR Zero-Crossing Rate (počet průchodů signálu nulou)

OS Operační systém (základní programové vybavení počítače)

USB Universal Serial Bus (univerzální sériová sběrnice pro připojení periférií k počítači)

API Application Programming Interface (rozhraní pro programování aplikací)

MVC Model–View–Controller (třívrstvá softwarová architektura)

B Poděkování dobrovolníkům

Touto cestou by autor práce velice rád poděkoval všem zúčastněným dobrovolníkům, kteří chvilkou svého času značně pomohli při testování navržených algoritmů.

Díky patří těmto osobám:

- Karolína Balejová
- Bc. Josef Baloun
- Lenka Benešová
- Václav Čepelák
- Bc. Kamila Doležalová
- Jiří Frank
- Monika Hanušová
- Jan Houdek
- Markéta Janková
- Lucie Jarolínová
- Bc. Jan Kašák
- Kristýna Kašová
- Simona Kolářová
- Bc. Kateřina Kratochvílová
- Tomáš Linhart
- Mgr. Theodor Pártl
- Bc. Jaroslav Pártl
- Mgr. Hana Pártlová
- Hana Pártlová
- Františka Sobotová
- Amálie Stará
- Lucie Steinerová
- Veronika Šrámková
- Bc. Jan Šťastný
- Radek Šťastný
- Michal Šteňo
- Michal Šteňo ml.
- Bc. Václav Váchal
- Ing. Petr Vaněček, Ph.D.
- Bc. Jakub Vašta
- Bc. Lukáš Voráček
- Ing. Jindřich Wolf

C Obsah přiloženého DVD

Datový nosič umístěný v předních deskách diplomové práce obsahuje následující adresáře:

`Aplikace HashVisualisation`
Adresář s aplikací *HashVisualisation* pro výkon vizualizačních experimentů s datasetem *SignatureDataset*. Aplikaci popisuje kapitola 9.

`Aplikace VideoCollector`
Adresář s mobilní aplikací *VideoCollector* pro sběr multimodálních podpisů dobrovolníků přispěvateli. Aplikace je popsána v sekci 8.4.

`Aplikace VoiceHashExps`
Adresář s konzolovou aplikací *VoiceHashExps* pro výkon experimentů s navrženými algoritmy extrakce otisků. Aplikaci popisuje kapitola 11.

`Audiostopy`
Adresář s akustickými podpisy ve formátu `.raw`, který je čitelný pro aplikace *HashVisualisation* a *VoiceHashExps*.

`Dataset SignatureDataset`
Adresář s nashromážděným datasetem multimodálních podpisů, který je popsán v kapitole 8.

`Diplomová práce`
Obsahuje zdrojový text diplomové práce v jazyce TEX (LATEX) včetně zdrojových souborů a obrázků ve formátech `.png`, `.svg` a `.m` (*Octave*).

`Knihovna libpe`
Adresář s knihovnou `libpe`, která je podrobně popsána v kapitole 7.

`Knihovna SignatureDatasetLib`
Adresář s knihovnou `SignatureDatasetLib`, která unifikuje práci se vzorky datasetu *SignatureDataset*. Knihovna je popsána v sekci 8.3.

`Poster`
Adresář s posterem diplomové práce včetně zdrojového souboru formátu `.pub` (*Microsoft Publisher*).

D Překlad vytvořeného SW

Obsahem této kapitoly je návod pro překlad vytvořeného programového vybavení. Ve všech projektech je vytvořen adresář `build`, který obsahuje spustitelné verze softwaru pro všechny běžně používané operační systémy. Výjimkou je jedině aplikace *VideoCollector*, která je přeložena pouze pro *OS Android*.

V případě potřeby překladu aplikací pro jiné operační systémy či architektury je postup následující:

1. Základem všech projektů je framework Qt. Pokud jej ještě nemáte nainstalovaný, stáhněte jej z `https://www.qt.io/download`.

2. Po úspěšné instalaci frameworku vytvořte prázdný adresář pro binární soubory překládané aplikace.

3. Pomocí příkazové řádky spusťte nástroj *qmake*, který ze zadaného projektového souboru vytvoří příslušný *makefile*. Projektovým souborem je myšlen soubor s příponou `.pro`, který se ve vytvořených projektech nachází v adresáři `src`. V následujícím příkazu značí `PROJECT_DIR`, resp. `PROJECT_FILE` adresář obsahující projekt překládané aplikace, resp. název projektového souboru.

```
qmake PROJECT_DIR/src/PROJECT_FILE.pro -spec linux-g++
```

4. Dalším krokem je samotný překlad a sestavení programu pomocí příkazu *make*.

```
make -j4
```

5. Viditelnost sdílených knihoven frameworku Qt je nutná. Pokud adresář s těmito knihovnami není součástí systémové proměnné `PATH`, knihovny je v závislosti na operačním systému možné shromáždit pomocí nástrojů *windeployqt* nebo *linuxdeployqt* frameworku Qt.

6. Přeloženou aplikaci lze nyní spustit podle uživatelské příručky, která je obsažena v kapitole E přílohy dokumentu.

E Uživatelská příručka

E.1 Aplikace *HashVisualisation*

Spuštění aplikace probíhá bez argumentů. Na úvodní obrazovce je uživatel vyzván, aby zadal cestu k adresáři se soubory zvukových nahrávek promluv podepisovaných osob. Adresář je možné získat pomocí skriptu `ExportAudio.sh` datasetu *SignatureDataset*. Spuštění experimentů je povoleno po zadání existujícího adresáře. Dostupné experimenty jsou popsány v sekci 9.2.

Po kliknutí na požadovaný experiment se otevře obrazovka, v jejíž levé části je seznam načtených akustických podpisů, který je v případě nenalezení platných nahrávek v adresáři prázdný. Pravá horní část obrazovky obsahuje možné konfigurace experimentu. Pouhým pohybem posuvníků tak uživatel může měnit počet vzorků segmentu, překryv, koeficient preemfáze apod. V případě nejasností je možné na malý moment přidržet kurzor nad nejasným popiskem, což způsobí zobrazení nápovědy. Tlačítko pro spuštění experimentu je v levém dolním rohu. Zaškrtnutím políčka „Automaticky spustit“ je experiment spuštěn při jakékoliv změně konfigurace nebo testovaných nahrávek. Po uzavření dialogového okna s indikátorem postupu výpočtu se výstup experimentu zobrazí ve střední části okna, kterou lze skrytím konfigurace zvětšit. Dolní část obrazovky obsahuje krátký popis experimentu. Popsané okno experimentu je zobrazeno na obrázcích F.3 a F.4.

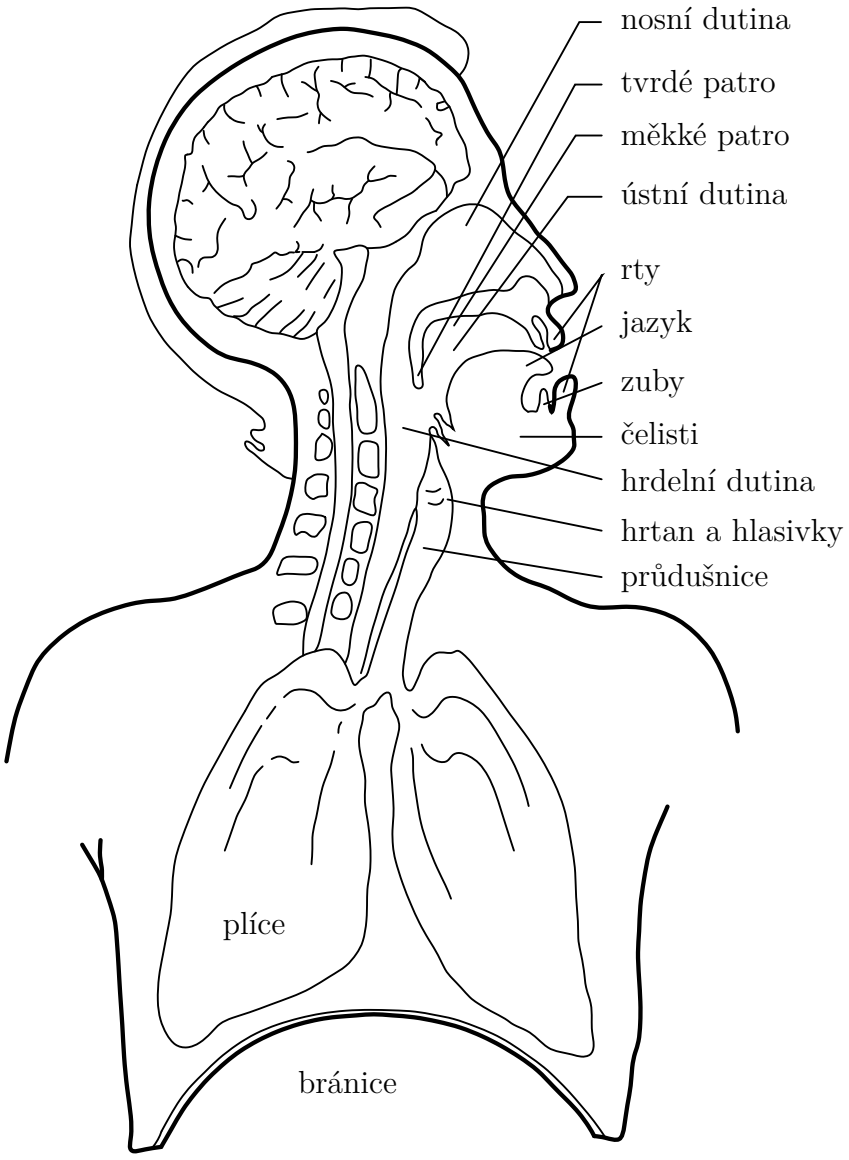
E.2 Aplikace *VoiceHashExps*

Pořadí a význam argumentů nutných ke spuštění této konzolové aplikace popisuje následující schéma

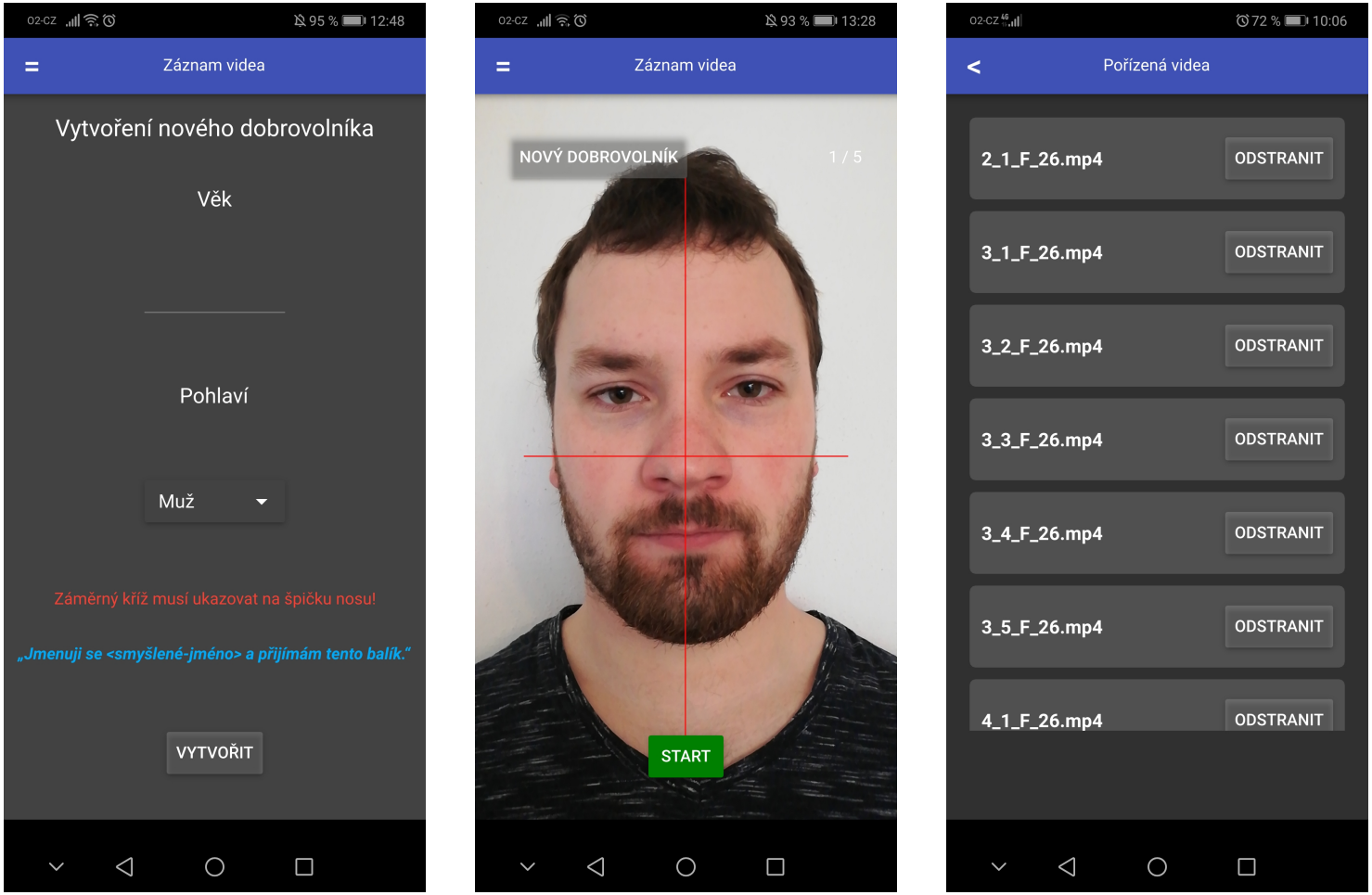
`voicehashexps <dataset> <výstupní-adresář> [experiment]`,

kde výrazy v závorkách `<>`, resp. `[]` značí povinné, resp. nepovinné argumenty. Parametr `dataset` představuje umístění adresáře s `.raw` soubory nahrávek akustických podpisů, které je možné získat spuštěním skriptu `ExtractAudio.sh`. Argument `výstupní-adresář` označuje umístění adresáře výsledků experimentu, který bude při spuštění kompletně vyprázdněn. Posledním a zároveň nepovinným parametrem je číselný identifikátor spuštěného experimentu, který je možné vybrat až po spuštění aplikace.

F Obrazová příloha

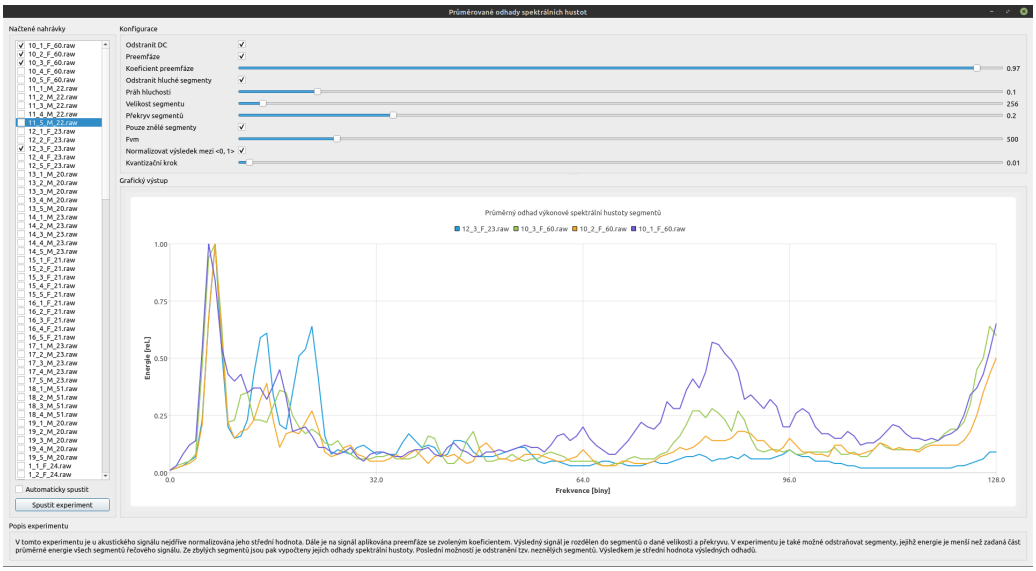


Obrázek F.1: Frontální řez hlasovým traktem člověka [26].

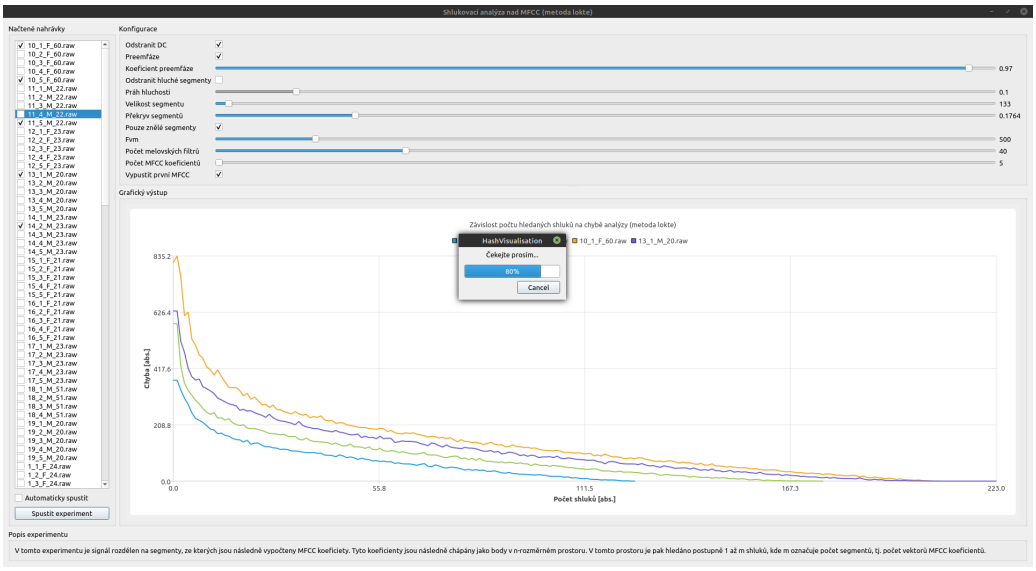


(a) Zavedení dobrovolníka. (b) Záznam podpisu. (c) Seznam pořízených podpisů.

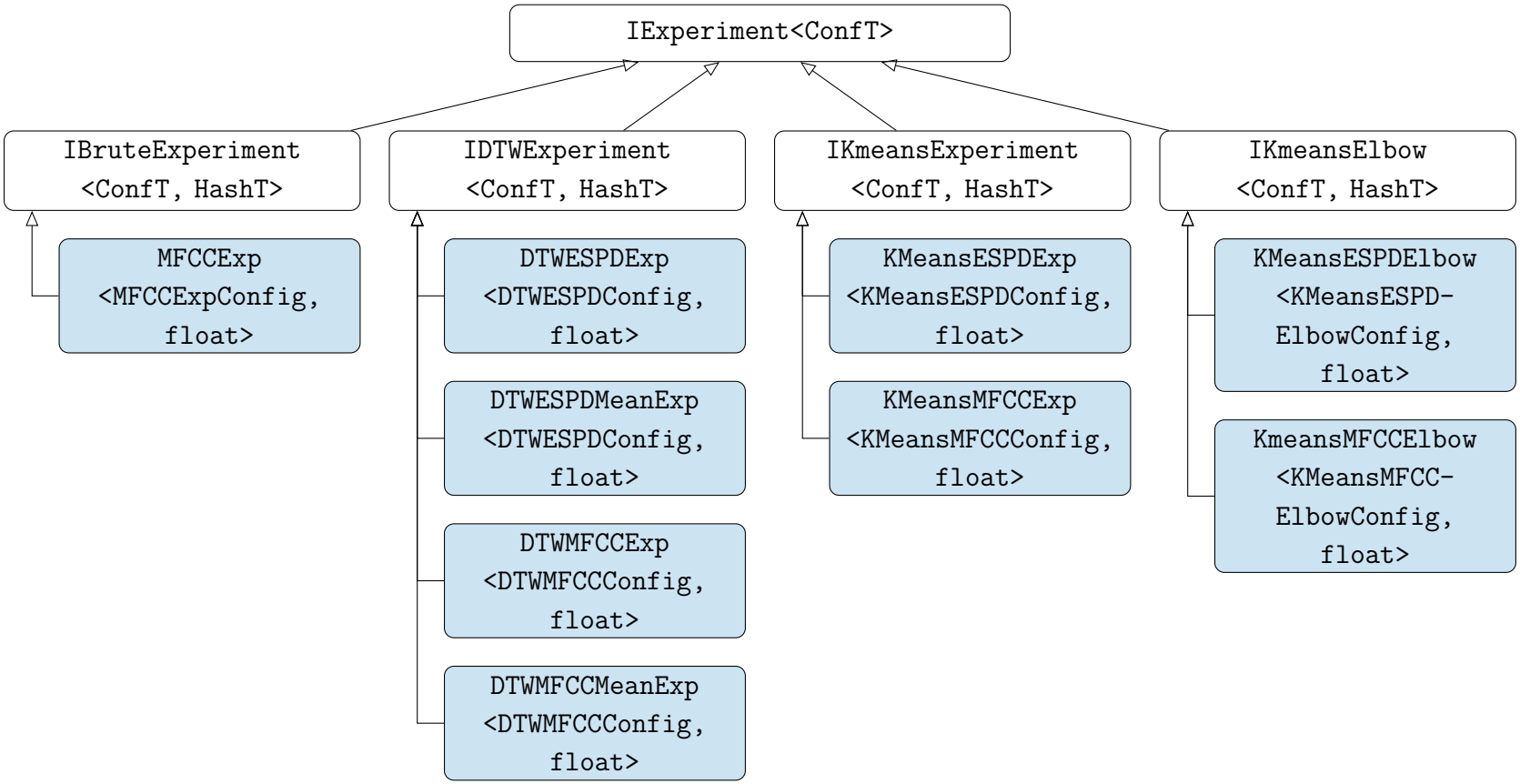
Obrázek F.2: Snímky obrazovky z aplikace *VideoCollector*.



Obrázek F.3: Obrazovka vizualizace průměrných odhadů výkonových spektrálních hustot vybraných dobrovolníků.



Obrázek F.4: Obrazovka vizualizace výsledků metody lokte shlukovací analýzy nad vektory MFCC koeficientů u vybraných dobrovolníků.



Obrázek F.5: Schéma dědičnosti implementovaných experimentů aplikace *VoiceHashExps*.