VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO ODSTRAŇOVÁNÍ ŠUMU V REÁLNÉM ČASE

MOBILE APPLICATIONS FOR REAL-TIME NOISE REMOVAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **FRANTIŠEK SILADI**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ONDŘEJ NOVOTNÝ,**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Siladi František**
Program: Informační technologie
Název: **Mobilní aplikace pro odstraňování šumu v reálném čase**
Mobile Applications for Real-Time Noise Removal
Kategorie: Zpracování signálů

Zadání:

1. Seznamte se s technikami na odstraňování šumu.
2. Seznamte se s dostupnými řešeními.
3. Vyberte vhodné metody, implementujte v libovolném jazyce a otestujte.
4. Přeimplementujte vybrané techniky do podoby mobilní aplikace.
5. Otestujte a zhodnoťte dosažené výsledky a použitelnost aplikace.

Literatura:

- Jan, Jiří: Číslicová filtrace, analýza a restaurace signálů, VUTIUM, 2002.
- dále dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Novotný Ondřej, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Cieľom tejto bakalárskej práce je vytvoriť mobilnú aplikáciu, ktorá bude v reálnom čase odšumovať reč. V prvej kapitole je popísaná teória súvisiaca so spracovaním signálov a popísané filtry na odšumenie. V nasledujúcej kapitole sú zhrnuté existujúce riešenia. Tretia kapitola obsahuje vlastný návrh a implementáciu. Na záver je zhodnotenie výslednej aplikácie a vyhodnotenie od užívateľov.

Abstract

The goal of this bachelor's thesis is to create a mobile application which will remove the noise from speech in real time. The first chapter describes the theory related to signal processing and described filters for denoising. The following chapter summarizes existing solutions. The third chapter contains my own design and implementation. At the end of this thesis is the evaluation of the application and evaluation from users.

Klíčová slova

Signály, spracovanie signálov, spracovanie reči, filter, šum, android

Keywords

Signals, signal processing, speech processing, filter, noise, android

Citace

SILADI, František. *Mobilní aplikace pro odstraňování šumu v reálném čase.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ondřej Novotný,

Mobilní aplikace pro odstraňování šumu v reálném čase

Prohlášení

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

.....
František Siladi
16. května 2019

Poděkování

Rád by som poďakoval vedúcemu bakalárskej práce Ing.Ondřejovi Novotnému za jeho rady, ochotu a čas, ktorý mi venoval na konzultáciach. Ďalej by som chcel poďakovať rodine za podporu a respondentom, ktorý mi poskytli spätnú väzbu.

Obsah

Kapitola 1

Úvod

Spracovanie signálov je jedným z významných a rýchlo sa rozvíjajúcich oborov s aplikáciami v rôznych oblastiach ľudskej činnosti. Aplikácie metód spracovania signálov sú veľmi široké a zahrňujú hi-fi audio, TV a rádio, mobilné telefóny alebo rozpoznávanie hlasu. Spracovanie signálov hraje významnú úlohu pri vývoji nových generácií mobilných, telekomunikačných a inteligentných systémov.

Ak sa človek nachádza v rušnom prostredí alebo niekde v spoločnosti, kde je v okolí hluk, tak pri rozhovore s iným človekom má ľudské telo schopnosť potlačiť nežiadúci zvuk a sústrediť sa iba na to, čo potrebuje. V takýchto situáciach vzniká množstvo video, nahrávok, ktoré už nemajú schopnosť potlačiť okolité ruchy a teda po opätovnom prehratí sú zvuky v pozadí nepríjemné a rušivé. Cieľom tejto práce je vytvoriť aplikáciu pre bežne mobilné zariadenia, ktorá by poskytla užívateľovi možnosť nahrávať zvuk v hlučnom prostredí, bez toho, aby užívateľ stratil informačnú hodnotu nahrávky.

Táto práca zhrnie základné informácie o analýze a spracovaní signálov. Bude sa zaoberať návrhom a implementáciou filtru, ktorý by dokázal zredukovať nežiadúci šum z reči a následné vytvorenie mobilnej aplikácie, ktorá by to dokázala v reálnom čase. V kapitole 2 sa budeme zaoberať teoretickým základom pre prácu so signálmi a zhrnieme si základné typy filtrov. Ďalej sa budeme venovať postupom, ktoré sú potrebné spraviť zo signálom pred spracovaním, následne si popíšeme metódy na redukciu šumu a techniky, ktoré nám objektívne dokážu zhodnotiť parametre signálu. Potom si v kapitole 3 preberieme existujúce riešenia, ktoré sa zaoberajú rovnakou alebo podobnou problematikou. V ďalšej kapitole (4) nájdeme návrh a implementáciu výslednej aplikácie. V posledných dvoch kapitolách, sa budeme venovať výslednej aplikácii a jej hodnoteniu, kapitoly 5 a 6.

Kapitola 2

Teória

Túto kapitolu tvoria informácie potrebné k spracovaniu signálov. Obsahuje základné pojmy a informácie o signáloch, ďalej obsahuje základné druhy filtrov a popis filtrácie. Ďalej sa tu nachádza podrobný popis predspracovania signálov a popis filtrov, ktoré slúžia na redukciu šumu z reči.

2.1 Základné pojmy

Spojitý signál nazývame spojitú funkciu $f(x)$ spojitej premennej x, ktorou je najčastejšie čas, môže ňou byť aj priestorová vzdialenosť alebo iná veličina. V našom pojatí je fyzikálna veličina, ktorá svojimi hodnotami signál vyjadruje, irelevantná.

Diskrétny signál je usporiadaná postupnosť hodnôt $f_i = f(i)$, ktorá je funkciou celočíselného indexu i. Veľmi často ide v prípade diskrétneho signálu o hodnoty nejakej funkcie $f(t)$ spojitej premennej v čase, ktorú vzorkujeme v časových okamžikoch t_i, i=0,1, ... ,n, ... , pričom najčastejšie je vzorkovanie rovnomerné, teda so vzorkovacou periódou T, takže $f(n) = f(t_n) = f(nT)$.

Číslicový signál je diskrétny signál, ktorého hodnoty sú vyjadrené pomocou čísel z nejakej konečnej číselnej množiny. Diskretizácia nie len do času, ale tiež čo do hodnôt, je podstatnou vlastnosťou číslicových signálov, ktoré v niektorých prípadoch významne ovplyvňujú výsledok spracovania. (Odchýlka medzi presnými hodnotami vzorkou a číselnými hodnotami vzorkou sa nazýva kvantizačný šum. Pri vysokej presnosti číselných hodnôt je zanedbateľný.)

Diskrétne spracovanie signálov sa dá charakterizovať ako prepočet vstupnej postupnosti hodnôt na postupnosť výstupnú. Sústavy, ktoré takýto prepočet realizujú, sa nazývajú diskrétne systémy. Sú charakterizované diferenciálnymi rovnicami, ktorých analýzou sa dajú obdržať informácie o vlastnostiach sústavy vzhľadom ku spracovaniu signálov.

Pokiaľ spracovávame diskrétne spojitý signál $x(t)$ zo zámerom získať opäť spojitý výstupný signál $y(t)$, je prvým článkom spracovania vzorkovač, ktorý v zadaných časových okamžikoch t_n získava vzorky vstupu x_n, tvoriace vstupnú postupnosť diskrétneho systému.

Na základe diskrétnej postupnosti $y(n)$, ktorú poskytuje diskrétny systém, je potom nutné vytvoriť spojitú funkciu $y(t)$ pomocou rekonštrukčného interpolaru. Táto sekcia bola spracovana podľa [12]

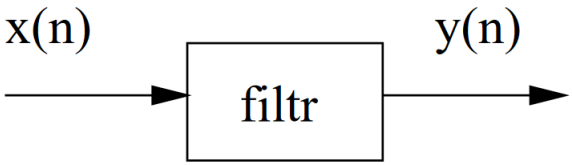
2.2 Filtrácia

Filtrácia je spravidla formulovaná ako spracovanie, slúžiace k výberu istých zložiek zo zmesi viacerých signálov a k potláčaniu zložiek iných. Všeobecnejšie sa dá chápať filtrácia ako prostriedok, umožňujúci meniť vlastnosti jednotlivých zložiek.

Zložky signálu sú najčastejšie chápané vo frekvenčnej oblasti – potom ide o harmonické komponenty, ktorých amplitúdy a časové vzťahy sa filtráciou zmenia, čo je vyjadrené dvoma frekvenčnými charakteristikami: amplitúdovou a fázovou. Vzhľadom k diskrétnemu charakteru signálu, sú tieto charakteristiky periodické a stačí udávať ich hodnotu len v rozsahu kmitočtu $< 0, \omega_s/2 >$, kde ω_s je uhlový vzorkovací kmitočet. V tejto sekcii boli použité informácie z [12] a doplnené o informácie z [24].

2.2.1 Lineárna filtrácia

Lineárny filter použijeme ak chceme nejak upraviť obsah kmitočtových zložiek v signály.



Obrázek 2.1: Lineárny filter. Prevzaté z [24]

Bežné filtry:

- Lineárne – zachovávajú lineárnu kombináciu: pokiaľ $x1(n) \rightarrow y1(n)$ a $x2(n) \rightarrow y2(n)$ potom $ax1(n) + bx2(n) \rightarrow ay1(n) + by2(n)$, kde a,b sú reálne čísla.
- Časovo invariantné – chovajú sa "stále rovnako", ak $x(n) \rightarrow y(n)$, potom tiež $x(n - n_0) \rightarrow y(n - n_0)$, kde n_0 je ľubovoľný posun. Niekedy však chceme, aby sa charakteristiky filtru v čase menili-adaptívne systémy, rečové rámce (zmena 10ms).
- Kauzálne – filter "nevidí do budúcnosti": $y(n)$ závisí iba na $y(m < n)$ a $x(m \leq n)$.

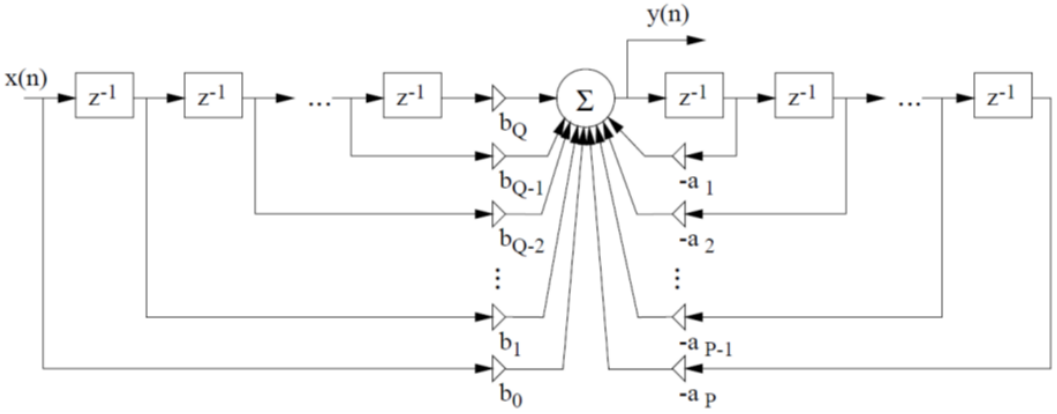
Schéma všeobecného filtru na obrázku 2.2. Blok z^{-1} označuje oneskorenie o 1 vzorku. Chovanie filtru sa dá zapísať ako referenčná rovnica:

$$y(n) = \sum_{k=0}^{Q} b_k x(n-k) - \sum_{k=1}^{P} a_k y(n-k), \tag{2.1}$$

kde $x(n-k)$ sú aktuálne a oneskorené verzie vstupu a $y(n-k)$ sú oneskorene verzie výstupu.

Základné typy filtrov:

- FIR (finite impulse response) – nerekurzívny: iba $b_0 \ldots b_n$ nenulov=e. Je vždy stabilný.



Obrázek 2.2: Obecný číslicový filter. Prevzate z [24]

- IIR (infinite impulse response) – čiste rekurzívny: iba $b_0, a_1..a_n$ nenulové.
- IIR (infinite impulse response) – všeobecne rekurzívny: a_i aj b_i nenulové.

Z diferenčnej rovnice sa však ťažko dá priamo poznať chovanie filtru vo frekvenčnej oblasti a tiež ťažko vyšetríme jeho stabilitu.Informácie boli čerpané z [24] a [12]

Filtry s konečnou impulznou charakteristikou

Filtry s konečnou impulznou charakteristikou (teda typu finite impulse response – FIR) sú plne definované N hodnotami tejto charakteristiky, ktoré tvoria súčasne vektor systémových konštant $h = [h_n], n =< 0, N-1 >$. Ich diferenčná rovnica je vyjadrená konečnou diskrétnou konvolúciou

$$y_n = \sum_{k=0}^{N-1} x_{n-k} h_k, \tag{2.2}$$

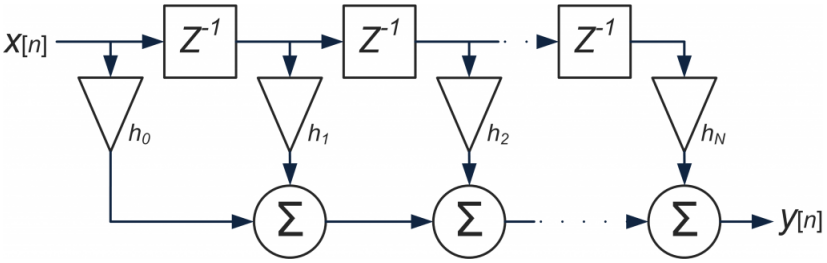
čo je súčasne vyjadrenie tzv. priameho realizačného algoritmu, takže hodnoty impulznej charakteristiky sú priamo systémovými realizačnými konštantami. FIR filtry sa dajú navrhnúť tak, aby ich fázová charakteristika bola presne lineárna. K tomu postačuje, aby impulzná charakteristika systému splňovala jednu z podmienok

$$h_n = h_{(N-1-n)} \text{ alebo } h_n = -h_{(N-1-n)}, \tag{2.3}$$

čo je tzv. symetrická charakteristika, resp. asymetrická charakteristika. Na obrázku 2.3 môžeme vidieť schému FIR filtru, $x[n]$ je vstupný signál a h_n označuje konštantu.

Filtry s nekonečnou impulznou charakteristikou

Filtry s nekonečnou impulznou odozvou (tj. typu infinity impulse response – IIR) sú vždy rekurzívne, pretože iba systém so spätnými väzbami môže túto vlastnosť zaistiť. Vo všeobecnosti sú ich možnosti bohatšie ako možnosti filtrov typu FIR. V porovnaní s FIR filtrami môžeme konštatovať, že pri porovnateľnej kvalite spracovania signálov sú menej náročné na rozsah výpočtu a tím na výkonnosť výpočtovej techniky, ktorá filter realizuje. Na druhú stranu, nevýhodou IIR filtru je, že nie je bezprostredný vzťah medzi hodnotami



Obrázek 2.3: FIR filter. Prevzaté z [14]

žiadanej frekvenčnej charakteristiky a systémovými konštantami realizačných štruktúr, čo vedie na náročný návrh. Ďalšími nevýhodami sú možnosti nestability pri nevhodnom návrhu, vyššia citlivosť na nepresnosti vznikajúce číslicovou realizáciou a skutočnosť, že tieto filtry majú principiálne vždy nelineárnu fázovú charakteristiku.

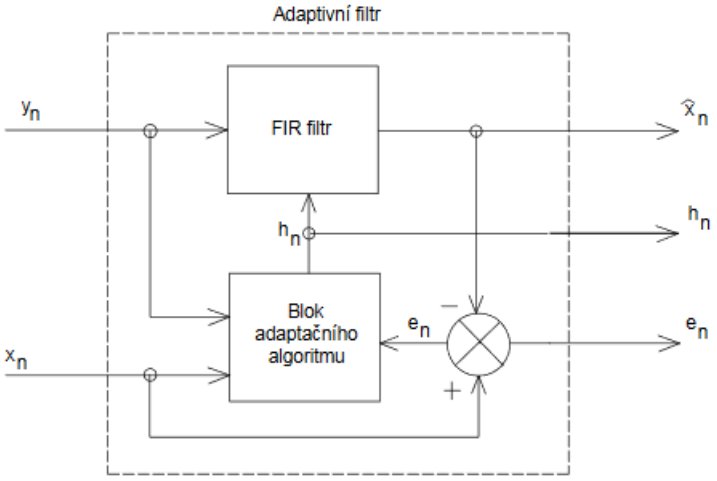
IIR filtry sú popísané všeobecnými rekurzívnymi diferenčnými rovnicami tvaru

$$y_n = \sum_{i=0}^{r} L_i x_{n-i} - \sum_{i=1}^{m} K_i y_{n-i}, \tag{2.4}$$

kde L_i resp. K_i sú systémové koeficienty v dopredných a spätných väzbách, r je počet oneskorení v nerekurzívnej časti systému a $m$je počet oneskorení v rekurzívnej časti, ktorý udáva súčasne rád systému.

2.2.2 Adaptívna filtrácia

Ak filter pracuje v neznámom prostredí, v ktorom priebežná identifikácia nie je možná, alebo ide o časovo premenné prostredie, ktorého vývoj do budúcnosti sa nedá predpovedať, je treba vytvoriť filter adaptívny. Ten je schopný sa v istom zmysle v danom prostredí učiť, t.j. získavať potrebné informácie (odhady potrebných veličín) v priebehu svojej práce. V princípe je potom možné očakávať, že filter bude s určitou rýchlosťou schopný reagovať



Obrázek 2.4: Adaptívny filter. Prevzaté z [12]

aj na zmeny tohto prostredia a teda spracovávať aj signály, generované nestacionárnymi procesmi, bez toho aby boli časovo premenné parametre dopredu známe.

Všeobecná bloková schéma adaptívneho filtru je na obr. 2.4. Na obrázku je možno vidieť, že filter má dva vstupy, pozorovací signál $\{y_n\}$, ktorý bude filtrom spracovávaný a trénovací signál $\{x_n\}$, ktorý viac či menej, tesne popisuje požadovaný výstupný signál filtru. Externé výstupy sú dopredu filtrovaný výstup – odhadovaný signál $\{\widehat{x}_n\}$, ktorý by mal byť aproximáciou originálu $\{x_n\}$, ďalej chybový signál $\{e_n\} = \{x_n - \widehat{x}_n\}$ a konečne tzv. identifikačný signál $\{h_n\}$ – postupnosť okamžitých hodnôt vektorov parametru filtru.

2.3 Predspracovanie reči (pre-processing)

Pred vlastným výpočtom filtra je vhodné urobiť niekoľko krokov. Informácie v tejto sekcii boli spracované podľa [24], [13] a [21].

2.3.1 Ustrednenie

Rovnako smerujúca zložka (dc-offset) nenesie žiadnu užitočnú informáciu, naopak môže byť pre ďalšie spracovanie rušivá (napr. výpočet energie). Bude teda dobre ju odstrániť odčítaním strednej hodnoty:

$$s[n] = s'[n] - \mu_s, \tag{2.5}$$

kde μ_s musíme odhadnúť. Stredná hodnota sa však dá počítať dvoma rôznymi spôsobmi:

- Offline – počíta sa jednoduchým priemerovaním po ukončení signálu:

$$\overline{s} = \frac{1}{N}\sum_{n=1}^{N} s[n] \tag{2.6}$$

- Online – ak nemáme k dispozícii celý signál (je príliš dlhý alebo neustále sa predlžuje), tak stredná hodnota sa dá vypočítať rekurzívne:

$$\overline{s}[n] = \gamma\overline{s}[n-1] + (1-\gamma)s[n], \tag{2.7}$$

  kde $y \to 1$. To je ekvivalentné filtrácii signálu filtrom s impulznou odozvou:

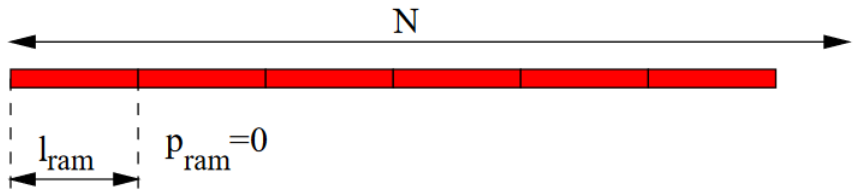
$$h = [(1-\gamma)(1-\gamma)\gamma(1-\gamma)\gamma^2...]. \tag{2.8}$$

2.3.2 Segmentácia na rámce

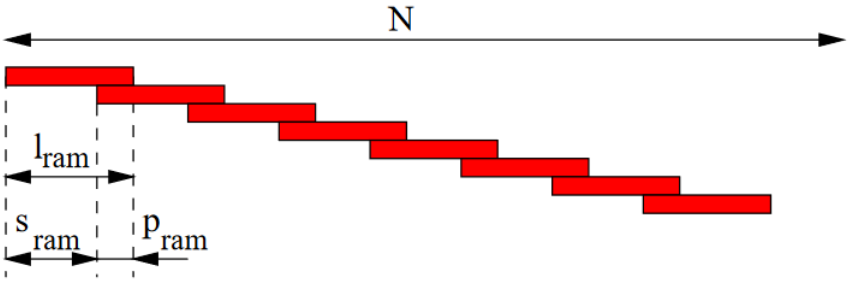
Rámce sú úseky signálu, na ktoré je nutné rečový signál pred ďalším spracovaním rozdeliť. Rečový signál považujeme za náhodný, pre metódy odhadu parametrov by mal byť stacionárny. Avšak nie je, a preto ho delíme na kratšie úseky (rámce, segmenty, mikrosegmenty, frames). Tam stacionárny bude alebo budeme dúfať, že bude. Parametre rámcov: dĺžka (length) l_{ram}, prekrytie (overlap) p_{ram}, posun rámca (frame shift) $s_{ram} = l_{ram} - p_{ram}$. Dĺžka rámcov by mala byť dostatočne malá, aby bolo možné pokladať signál daného úseku za stacionárny, ale na druhej strane dostatočne veľká, aby bolo možné dostatočne presne odhadnúť požadované parametre. Kompromisom je dĺžka rešpektujúca zotrvačnosť hlasového ústroja, typicky 20-25 ms (160-200 vzorkou pre vzorkovaciu frekvenciu 8000 Hz). Prekrytie rámcov by bolo vhodné:

- Malé alebo žiadne – zaisťuje rýchly časový posun v signále, malé nároky na pamäť/procesor, hodnoty parametrov sa však od jedného rámca k druhému môžu veľmi meniť.
- Veľké – zaisťuje pomalý časový posun, "vyhladené" priebehy parametrov, avšak má veľké nároky na pamäť/procesor. Výsledné parametre môžu byť naviac rámec od rámca príliš podobné, čo ide proti požiadavku nezávislosti pri rozpoznávaní pomocou skrytých Markovových modelov.

Kompromisom je teda dĺžka 10 ms, teda 100 rámcov za sekundu.



Obrázek 2.5: Segmentacia na ramce bez prekrytia. Prevzaté z [24]



Obrázek 2.6: Segmentacia na ramce s prekrytim. Prevzaté z [24]

U rámcov bez prekrytia obrázok 2.5, kde $p_{ram} = 0$ (bežné pre kódovanie), dostaneme ich počet prostým delením a zaokrúhlením dole. Predpokladajme, že posledný rámec, na ktorý už nemáme dostatok vzorkou zahadzujeme.

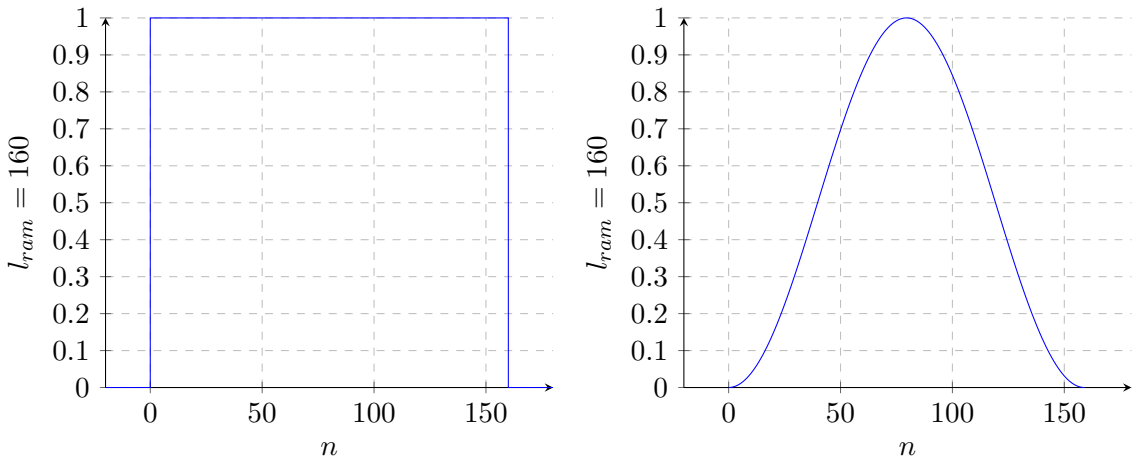
$$N_{ram} = \lfloor \frac{N}{l_{ram}} \rfloor \tag{2.9}$$

Pre rámce s prekrytím, obrázok 2.6, kde $p_{ram} \neq 0$ (bežné pre rozpoznávanie), je ich počet stanovaný na podla vzorca (za predpokladu, že signál je aspoň jeden rámec dlhý):

$$N_{ram} = 1 + \lfloor \frac{N - l_{ram}}{s_{ram}} \rfloor \tag{2.10}$$

2.3.3 Okienkové funkcie

Po rozdelení signálu na menšie rámce vznikajú na okrajoch rámcov strmé prechody, tento jav musíme odstrániť alebo zredukovať. Úlohou okna je vybrať príslušné vzorky signálu a prideliť im pri spracovaní určitú váhu $w[n]$. Váhová funkcia určuje typ okna. Často používané typy pri spracovaní rečového signálu sú:



Obrázek 2.7: Pravouhlé a Hammingovo okno

- Pravouhlé (rectangular) okno – so signálom nerobí žiadne zmeny.

$$w[n] = \begin{cases} 1 & \text{pre } 0 \leq n \leq l_{ram} - 1 \\ 0 & \text{inde} \end{cases} \tag{2.11}$$

- Hammingove okno – utlmí signál na okrajoch.

$$w[n] = \begin{cases} 0.54 - 0.46cos\left(\frac{2\pi n}{l_{ram}-1}\right) & \text{pre } 0 \leq n \leq l_{ram} - 1 \\ 0 & \text{inde} \end{cases} \tag{2.12}$$

2.4 Detekcia rečovej aktivity

Detektory reč/pauza sú obmedzujúcou časťou systémov na potláčanie prídavných zvukov v reči, pretože kvalita detektora určuje výkon celého systému na potláčanie šumu. Ak rozhodovanie o reči/pauze nie je správne, je cely proces potláčania šumu menej účinný alebo dokonca znehodnocuje užitočný rečový signál (zbytky hluku, rečové ozveny). Je to systém, do ktorého vstupuje čistý rečový signál či zmes reči a šumu. Výstupom je potom „1“, ak je v i-tom segmente prítomná reč a „0“ pre segment bez rečovej aktivity. Informácie o detektoroch rečovej aktivity boli spracovávané z [2] a [22]

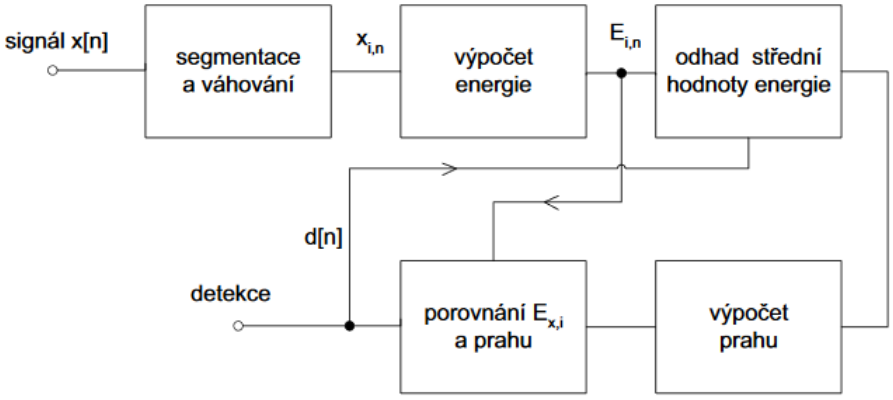
2.4.1 Ideálny detektor

Jedná sa o dokonalý detektor rečovej aktivity. V tomto detektore, ručne označíme úseky signálu s rečovou aktivitou. Pre veľké množstvo dát je tento spôsob veľmi nepraktický.

2.4.2 Energetický detektor

Jednoduchý energetický detektor s blokovou schémou na obr. 2.8 stanovuje prah pre detekciu reči na základe sledovania krátkodobej energie, prípadne výkonu, signálu. Tento typ detektoru v každom segmente počíta energiu signálu. Krátkodobá energia signálu sa dá definovať vzťahom

$$E_n = \sum_{k=-\infty}^{\infty} [s(k)w(n-k)]^2, \tag{2.13}$$



Obrázek 2.8: Schéma detektora rečovej aktivity.

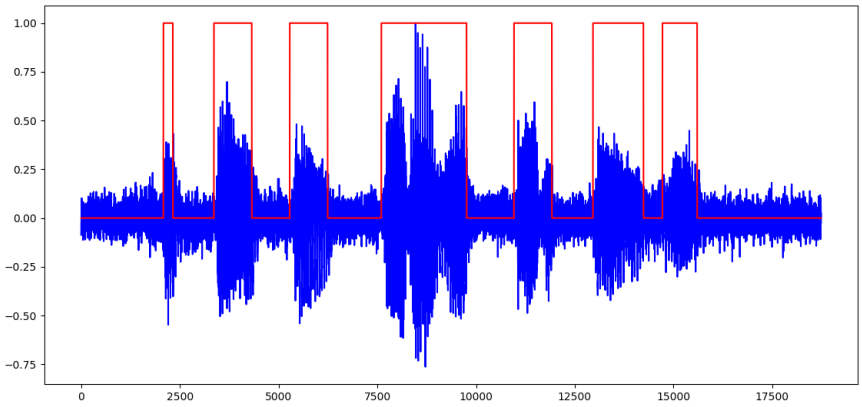
kde $s(k)$ je vzorka signálu v čase k a $w(k)$ je príslušný typ váhovacieho okna. Funkcia môže byť citlivá na veľké zmeny úrovne signálu, preto sa niekedy používa krátkodobá intenzita, ktorá tento nedostatok nemá

$$M_k = \sum_{k=-\infty}^{\infty} |s(k)|w(n-k) \tag{2.14}$$

Segment energie z rovnice 2.13 alebo 2.14 sa porovná s prahovou hodnotou energie E_p definovanou ako

$$E_p = 2E_d, \tag{2.15}$$

kde E_d je úroveň energie hluku pozadia aktualizovaná ako priemer posledných 20 rámcov, v ktorých nebola detekovaná reč. Aktualizácia sa robí iba v rečových pauzách, Jedna sa v podstate o vyhladenie priebehu energie jednotlivých rámcov, kde E je energia signálu v segmente. Detekčný proces je potom nasledujúci: ak platí, že E_n je väčšie ako $E - p$, je deketovaná reč, v opačnom prípade je energia šumu aktualizovaná.



Obrázek 2.9: Detektor rečovej aktivity

2.5 Metódy redukcie šumu v rečových signáloch

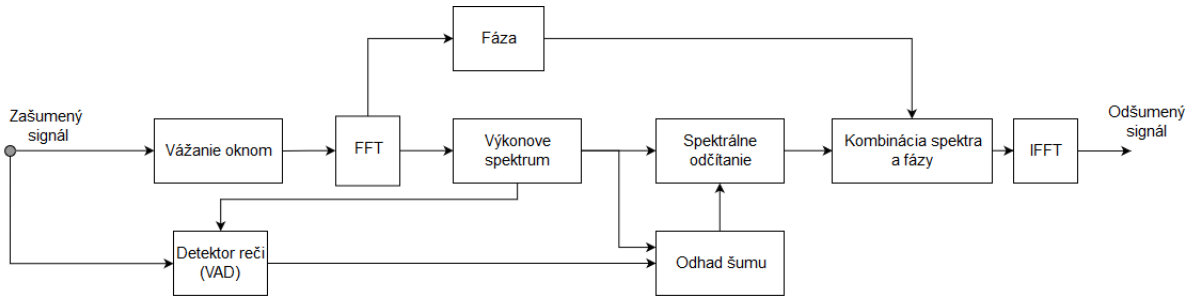
V tejto sekcii si priblížime metódy na redukciu šumu v signáloch. Metóda spektrálneho odčítania bola spracovaná z [13] a doplnená o informácie z [21]. Metóda local-mean filtra spracovaná podľa [7] a [9] a pre metódu Wienerovho filtra bol použitý zdroj [21].

2.5.1 Spektrálne odčítanie

Jednoduchou a široko študovanou metódou zdokonalenia reči je metóda spektrálneho odčítania ilustrovaná na obrázku 2.10. V tejto metóde sa odčíta odhad priemerného spektra šumu od spektra zašumeného signálu. Vychádza z princípu vzniku zašumeného rečového signálu. Pokiaľ sa uvažuje rečový signál $s(n)$ degradovaný aditívnym šumom pozadia $d(n)$, potom zmes reči a šumu sa môže vyjadriť ako

$$y(n) = s(n) + d(n). \tag{2.16}$$

Prvý krok pri spektrálnom odčítaní je rozdelenie signálu na krátke rámce 2.3.2, buď s prekrytím niektorých časti alebo bez prekrytia. Typická dĺžka rámcov je asi 20 ms. Pre signál so vzorkovacou frekvenciou 8 kHz to odpovedá 160 vzorkám. Ak použijeme 50% prekrytie rámcov, čo je približne 80 vzoriek, tak prvý rámec začína na vzorku 0, druhý na vzorku 80, tretí na vzorku 160 atď. Zvyčajne na každý rámec požijeme okienkovú funkciu 2.3.3, napríklad hammingove okno. Potom vezmeme diskrétnu Fourierovu tranformáciu[12] každého rámca a extrahujeme ich amplitúdové spektrum a fázu. Akonáhle máme vstupný signál rozdelený na rámce, tak mame $y_i(n)$, kde n označuje vzorky a i označuje jednotlivé rámce. Keď vypočítame komplexnú diskrétnu Fourierovu transformáciu (DFT), tak dostaneme $Y_i(k)$, kde i označuje číslo rámca. Potom $M_i(k)$ je amplitúdové spektrum rámca i. Aby sme dostali



Obrázek 2.10: schéma spektrálneho odčítania. Prevzaté z [21]

DFT každého rámca, aplikujeme:

$$Y_i(k) = \ DFT[y_i(k)w(k)] \tag{2.17}$$

kde $w(n)$ označuje okienkovú funkciu, 2.3.3. Amplitúdové spektrum pre rámec $y_i(n)$ je dané rovnicou 2.18 a fáza je daná rovnicou 2.19.

$$M_i(k) = |Y_i(k)| \tag{2.18}$$

$$\theta(k) = \tan^{-1}\left(\frac{ImY_i(k)}{ReY_i(k)}\right) \tag{2.19}$$

Ak použijeme vyššie uvedené kroky na každý rámec, mali by sme mať veľa spektier. Spoločným predpokladom je, že prvých niekoľko rámcov pozostáva len zo šumu, teda nie je tam detekovaná reč, takže by mali byť dobrým príkladom spektra šumu. Čím viac rámcov použijeme, tým lepší bude odhad šumu, ale nemôžeme do nich zahrnúť rámce s rečou. Keď už máme amplitúdu každého rámca a odhad šumu, pokračujeme samotným odčítaním, kde $\widehat{S}(w)$ reprezentuje odšumené spektrum, $M_i(w)$ reprezentuje zašumené spektrum a $\widehat{D}_i(w)$ pre odhad šumu:

$$\widehat{S}(w) = \begin{cases} M_i(w) - \widehat{D}(w) & \text{pre } M_i(w) > \widehat{D}(w) \\ 0 & \text{inde} \end{cases} . \tag{2.20}$$

Odhad šumu $\widehat{D}_i(w)$ sme získali z rámcov, v ktorých sme, pomocou detektoru reči, detekovali len šum. Existuje niekoľko modifikácii pre spektrálne odčítanie. Ak zapíšeme spektrálne odčítanie ako

$$\widehat{S}(w)^\gamma = M_i(w)^\gamma - \widehat{D}(w)^\gamma. \tag{2.21}$$

Pre $\gamma = 1$ máme amplitúdové spektrum, ale pre $\gamma = 2$ hovoríme o výkonovom spektrálnom odčítaní. Alternatívne môžeme použiť $\gamma = 0.8$ alebo $\gamma = 3$. Každá z týchto možnosti je trochu odlišnú charakteristiku ovplyvňujúcu redukciu šumu alebo znehodnotenie reči. Ďalšia modifikácia súvisí s odčítaním šumu:

$$\widehat{S}(w) = M_i(w) - \alpha\widehat{D}(w). \tag{2.22}$$

Pre klasickú metódu sa používa hodnota $\alpha = 1$, ale niekedy nie je dostatok hluku z pozadia odstránený, preto môžeme použiť napríklad $\alpha = 1.5$. Vo výsledku sa odstráni viacej šumu zo signálu. ale môže to viesť k poškodeniu hovorenej časti.

Ako môžeme vidieť vo vzťahu 2.20, fáza zašumenej reči sa nespracováva. Využíva sa vlastnosť ľudského ucha, kedy človek nevníma zmenu fázy. Po odčítaní signálu v spektrálnej oblasti je obnovený komplexný rečový signál získaný zo vzťahu

$$\widehat{s}(n) = IFFT\lfloor|\widehat{S}(w)|e^{jargY(w)}\rfloor. \tag{2.23}$$

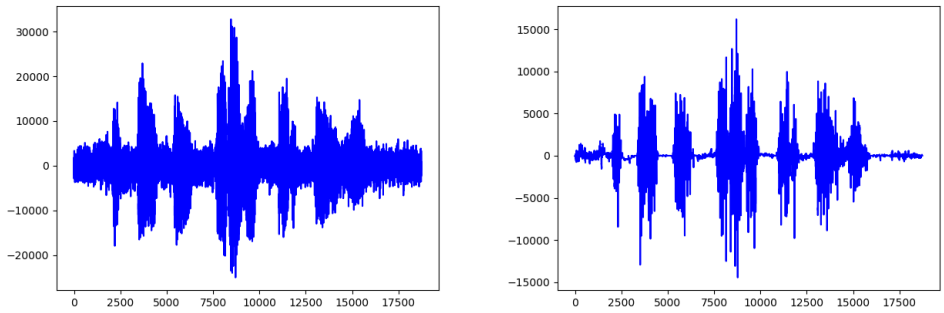
vo vzťahu 2.23 sa používa fáza degradovaného rečového signálu.

Veľký vplyv na kvalitu odčítania má detekor rečovej aktivity 2.4. Ak sú nesprávne určené rečové pauzy, tak celý proces odčítania môže byť znehodnotený.

Na obrázku 2.11 môžme vidieť ako technika funguje pri odšumovaní signálu zo stacionárnym šumom v pozadí.

Problém zo spektrálnym odčítaním spočíva v tom, že často skresľuje reč a to ma za následok vznik nepríjemných krátkych záblesk zvuku známe ako hudobný šum. Nedostatky metódy spektrálneho odčítania sa dajú zhrnúť nasledovne:[21]

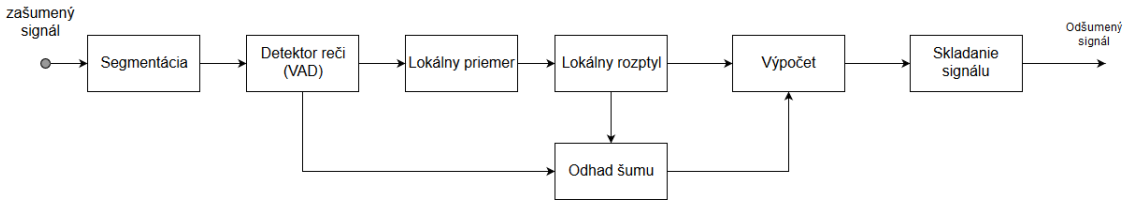
- Jediná štatistika používaná v spektrálnom odčítaní je priemer veľkosti spektra šumu. Priemer a rozptyl čistej reči a rozptyl šumu sa v procese odhadu nepoužívajú. Následkom toho nie sú rušivé šumy potlačené, čo ma za následok väčšie skreslenie než by to bolo v prípade, že by tieto informácie o rozptyle boli použité.
- Je treba použiť tvrdé rozhodovanie, aby sa predišlo tomu, že hodnoty spektra po odčítaní budú záporné alebo pod hodnotou šumového minima.
- Metóda spektrálneho odčítania nie je špecifická pre reč, sprektrálne trajektórie reči v čase nie sú modelované a používané v procese odšumovania.



Obrázek 2.11: Spektrálne odčitanie

2.5.2 Local-mean filter

„Mean filtering“ je jednoduchá, intuitívna a ľahko realizovateľná metóda vyhladzovania signálov, t.j. zníženie rozdielu medzi susednými vzorkami. Myšlienka priemerného filtrovania je jednoducho nahradiť každú hodnotu vzorku v signále priemernou hodnotou jeho susedov a jeho samotného. To ma za následok elimináciu hodnôt, ktoré nie sú reprezentatívne pre ich okolie. Priemerná filtrácia sa považuje za konvolučný filter. Podobne ako iné konvolúcie je založená na jadre, ktoré predstavuje tvar a veľkosť okolitých vzorkou, ktoré sa majú pri výpočte spriemerovať. Čím väčšie je jadro, tým dochádza k väčšiemu vyhladeniu vzorkou.



Obrázek 2.12: Schéma local-mean filter

Signál sa rozdelí na rámce 2.3.2, ideálne s prekrytim. Typická dĺžka rámcov sa pohybuje na úrovni 20 ms. Dôležitou súčasťou tejto metódy je detektor reči. V rámcoch signálu, kde bola nebola detekovaná reč sa použije väčšie jadro pre výraznejšie vyhladenie signálu a v rámcoch, kde bol detekovaná reč sa zasa použije menšie jadro. V rámcoch, v ktorých bola detekovaná reč, sa vypočíta lokálny priemer m_x a lokálny rozptyl σ_x^2 ako vzájomná korelácia (cross-correlation)[15] vstupného signálu a váhovacej funkcie 2.3.3.

$$m_x = \frac{1}{M}\sum_{i=0}^{M} x_i(n)w_i(k) \tag{2.24}$$

$$\sigma_x^2 = \left(\frac{1}{M}\sum_{i=0}^{M} x_i^2(n)w_i(k)\right)/m_x^2 \tag{2.25}$$

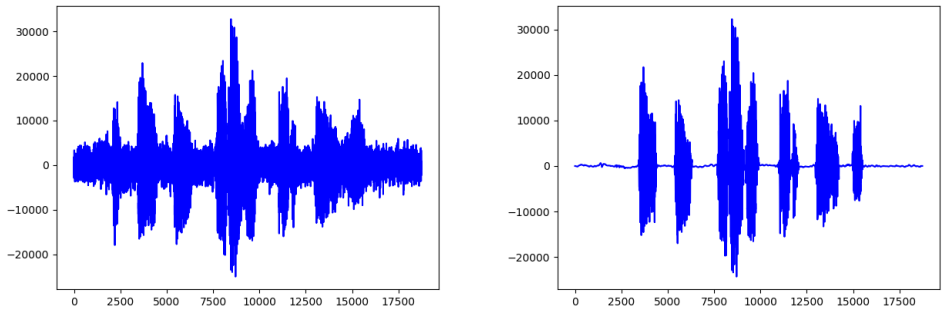
Ďalej odhad šumu σ^2 sa vypočíta ako

$$\sigma^2 = \frac{1}{M}\sum_{i=0}^{M-1} \sigma_x^2, \tag{2.26}$$

kde M označuje dĺžku rámca z ktorého počítame odhad šumu. Ako vstupný signál si označíme x, tak potom výstup je

$$y = \begin{cases} \frac{\sigma^2}{\sigma_x^2} m_x + (1 - \frac{\sigma^2}{\sigma_x^2})x & \sigma^2 \leq \sigma_x^2 \\ m_x & \sigma^2 > \sigma_x^2 \end{cases} \tag{2.27}$$

Na obrázku 2.13 môžme vidieť ako technika funguje pri odšumovaní signálu zo stacionárnym šumom v pozadí.



Obrázek 2.13: Local-mean filter

2.5.3 Wiener filter vo frekvenčnej oblasti

Wienerov filter vo frekvenčnej oblasti bol definovaný v roku 1949. Je založený na teórii najmenších štvorcových chýb (least square error), ktorá tvorí základ adaptívnych lineárnych filtrov závislých od údajov. Tieto filtre zohrávajú ústrednú úlohu v širokom spektre aplikácii, ako sú lineárne predikcie, redukcia echa, reštaurácia signálu alebo identifikácia systému. Koeficienty filtra s najmenšou štvorcovou chybou sú počítané tak, aby minimalizovali priemernú štvorcovú vzdialenosť medzi výstupom filtra a požadovaným alebo cieľovým signálom. Vypočítanie koeficientov pre Wienerov filter vyžaduje odhady pre autokoreláciu vstupu a vzájomnú koreláciu vstupu a požadovaného signálu.

Vo svojej základnej forme, filter najmenších štvorcových chýb predpokladá, že signály sú stacionárne procesy. Avšak, ak sú koeficienty filtra pravidelne prepočítavané a aktualizované pre každý blok potom sa filter prispôsobí na priemernú charakteristiku signálu a stáva sa z neho „block-adaptive“ filter. Takýto filter môže byť použitý pre signály, ktoré môžu byť považované za takmer stacionárne nad relatívne malým blokom vzoriek. Považujme signál $x(m)$ za čistý signál, ktorý je znečistený šumom $n(m)$, potom zmes týchto signálov $y(m)$ je definovaná ako

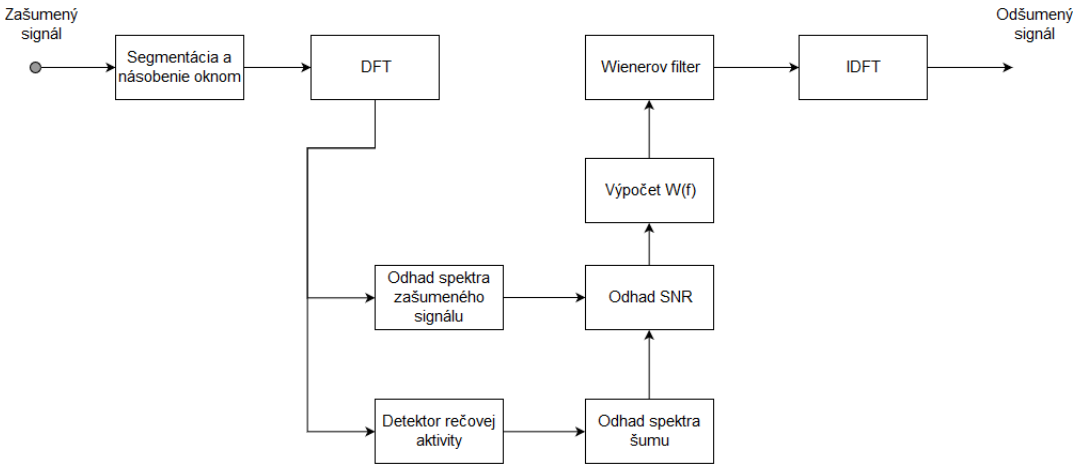
$$y(m) = x(m) + n(m). \tag{2.28}$$

Vo frekvenčnej oblasti je zašumený signál $Y(f)$ daný ako:

$$Y(f) = X(f) + N(f), \tag{2.29}$$

kde $X(f)$ a $N(f)$ sú čistý signál a šum vo frekvenčnej oblasti. Výstup Wienerovho filtra $\widehat{X}(f)$ je produkt vstupného zašumeného signálu $Y(f)$ a frekvenčnej odozvy filtra $W(f)$:

$$\widehat{X}(f) = W(f)Y(f). \tag{2.30}$$



Obrázek 2.14: Schéma Wienerovho filter[21]

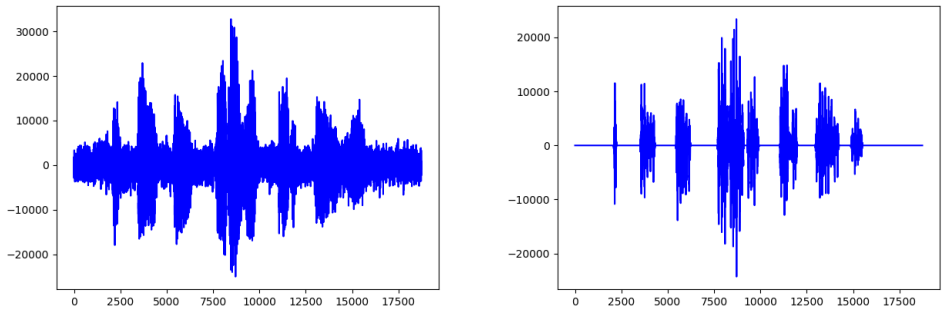
Frekvenčná odozva Wienerovho filtru sa získa ako:

$$W(f) = \frac{P_{XX}(f)}{P_{XX}(f) + P_{NN}(f)}, \tag{2.31}$$

kde $P_{XX}(f)$ a $P_{NN}(f)$ sú výkonové spektra signálu a šumu. Delením čitateľa a menovateľa rovnice 2.31 výkonovým spektrom šumu $P_{NN}(f)$ a nahradením $SNR(f) = P_{XX}(f)/P_{NN}(f)$ dostávame

$$W(f) = \frac{SNR(f)}{SNR(f) + 1}, \tag{2.32}$$

kde SNR je pomer signálu a šumu. Premenná $SNR(f)$ v rovnici 2.32 je vyjadrená ako pomer výkonových spektier a nie v bežnejších jednotkách logaritmického pomeru alebo dB. Preto $SNR(f) = 0$ vyjadruje nulový obsah signálu alebo $SNR(f) = 1$ vyjadruje rovnosť výkonového spektra signálu a šumu $P_{XX}(f) = P_{NN}(f)$. Na obrázku 2.15 môžme vidieť ako technika funguje pri odšumovaní signálu zo stacionárnym šumom v pozadí.



Obrázek 2.15: Wienerov filter

Dôležitou súčasťou implementácia je segmentácia 2.3.2, násobenie okienkovou funkciou 2.3.3 a detektor rečovej aktivity 2.4. Hlavným praktickým problémom pri implementácii Wienerovho filtru je, že požadovaný signál je pozorovaný v šume, a že autokorelácia, alebo

výkonové spektrum, nie sú ľahko dostupné. Obrázok 2.14 znázorňuje konfiguráciu blokového diagramu systému na implementáciu Wienerovho filtru pre redukciu šumu. Implementácia tohto filtra vyžaduje odhady spektrálneho pomeru signálu k šumu. Odhad šumu sa získa z rámcov, v ktorých nie je detekovaná reč a odhad čistých výkonových spektier signálu sa môže získať odčítaním odhadu šumových spektier od spektier šumového signálu.

2.6 Signal-to-noise

Na určenie sily signálu je potrebné vypočítať, to čo sa nazýva pomer signálu k šumu (SNR). Patrí k najjedoduchším technikám objektívneho hodnotenia. Čím vyšší je pomer, tým ľahšie sa zistí skutočny signál alebo sa získajú užitočne informácie zo signálu. Je teda definovaný ako pomer výkonu signálu P_{signal} k výkonu šumu pozadia P_{noise}. V elektronike sa meria signál a šum v decibeloch (dB).

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{2.33}$$

Za nevýhodu sa dá považovať to, že na signál pozerá ako na celok a v=ysledná hodnota je určena ako priemer. Z toho vyplýva, že aj v niektorých častiach veľmi znehodnotený signál môže v konečnom dôsledku udávať prijateľnu hodnotu pomeru signálu a šumu. Informácie v tejto sekcii boli spracované podľa [24] a [8]. Na meranie odhadu SNR bola použita aplikácia *SNR_estimation.py*, popísana v [6].

2.7 PESQ

Algoritmus Perceptual Eavluation of Speech Quality (PESQ) je objektívna metóda merania rečovej kvality. V podstate PESQ predpovedá subjektívne skóre Mean OpinionScores (MOS) porovnaním zašumených nahrávok reči s pôvodnými verziami týchto rečových nahrávok. Algoritmus meria účinky jednosmerného skreslenia reči a šumu na kvalitu reči, účinky straty hlasitosti, oneskorenie alebo echo sa neodrážajú na PESQ skóre. PESQ skóre sa dáva na stupnici od 1 do 5, kde 5 označuje najvyššiu možnú kvalitu. Na meranie PESQ bol použitý program v jazyku C dostupny zo stránky `https://github.com/TuFengzhi/PESQ`. Informácie v tejto sekcii boli čerpané z [18].

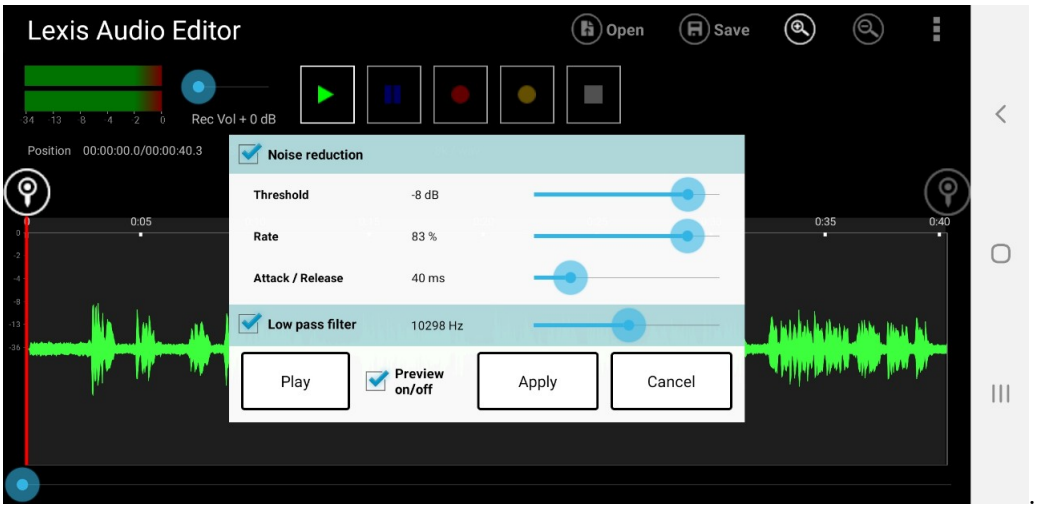
Kapitola 3

Existujúce aplikácie

V tejto kapitole budú predstavené existujúce mobilné aplikácie, ktoré sú dostupné v obchode Google Play a slúžia na nahrávanie a redukciu šumu. Primárne sa zamerám na filtru na odšumenie, ďalej na hodnotenie dizajnu a intuitívnosti ovládania.

3.1 Lexis Audio Editor

Jednoduchá, intuitívna aplikácia, ktorá dokáže nahrávať zvuk alebo načítať zvuk z mobilu. Aplikácia dokáže vizualizovať priebeh signálu, má jednoduché ovládanie pomocou tlačidiel play, stop, pause alebo record. Taktiež obsahuje celú radu efektov, ktoré sa dajú aplikovať na signál. Avšak všetky efekty sa aplikujú na signál až po jeho nahraní, teda aplikácia pracuje v offline režime a nedokáže vizualizovať alebo spracovávať signál v reálnom čase. Pred odšumením poskytuje možnosť nastavenia hranice, pod ktorú sa bude šum potláčať a na šum s vyššou frekvenciou používa „low pass“ filter. Viz obrázok 3.1. Dostupné z https://play.google.com/store/apps/details?id=com.pamsys.lexisaudioeditor&hl=sk.



Obrázek 3.1: Lexis Audio Editor

3.2 WaveEditor for Android Audio Recorder & Editor

Ďalšia aplikácia pôsobí prepracovanejším grafický prostredím. Taktiež je schopná spracovávať nahrávky zo zariadenia a dokáže nahrať aj nové pomocou mikrofónu. Uložené nahrávky zvládne vizualizovať, oproti predchádzajúcej aplikácii dokáže približovať a oddiaľovať zobrazene dáta. Opäť obsahuje veľké množstvo efektov aj keď niektoré sú dostupné iba v platenej verzii. Na redukciu šumu používa software „noise gate“, ktorý sa používa na ovládanie hlasitosti zvukového signálu. Dostupný na https://play.google.com/store/apps/details?id=io.sbaud.wavstudio&hl=en_US.

Obrázek 3.2: WaveEditor for Android

3.3 Mp3, WAV Noise ReducerÑoise Free Audio Converter

Posledná aplikácia, tak ako ostatné pred ňou, dokáže zvuk nahrávať alebo načítať zo zariadenie, a následne ho odšumiť. Nedokáže pracovať v reálnom čase. V aplikácii sa nachádza aj okno pre vizualizáciu vzoriek, ale to pôsobi skôr len na navodenie dojmu práce

Obrázek 3.3: Mp3, WAV Noise Reducer

zo zvukom a neplní to reálnu funkciu zobrazenia hodnôt. Obrázok 3.3. Dostupné z https://play.google.com/store/apps/details?id=com.inverseai.noice_reducer

3.4 Zhrnutie

Na Google Play sa mi nepodarilo nájsť zadarmo dostupnú aplikáciu na mobilné zariadenie, ktorá by v reálnom čase odšumovala vstup z mikrofónu a vysielala do slúchadiel. Aplikácie, ktoré sa podarilo nájsť pracovali v offline režime. Na druhej strane najlepšia sa javila aplikácia WaveEditor, ktorej sa najlepšie podarilo zredukovať šum z nahrávky a zároveň nie veľmi poškodiť hovorené slovo.

Kapitola 4

Návrh a implementácia

Návrh aplikácie by sme vedeli rozdeliť do niekoľkých častí. V prvej časti som sa zaoberal špecifikáciou, čo od aplikácie očakávame, aká ma byť jej funkcionalita a taktiež zhodnotením už existujúcich aplikácii s rovnakou alebo podobnou funkcionalitou. Ďalšou časťou návrhu bolo vytvorenie prototypu aplikácie, ktorá spracovávala offline nahrávky a pomocou filtru odstraňovala nežiadúce zvuky z nahrávok. V ďalšej, tretej časti, vzniká funkčný filter na odšumovanie. Vo štvrtej časti vzniká aplikácia pre mobilné zariadenia, ktorá spracováva zvuk v reálnom čase, za použitia filtru z tretej časti, a grafické prostredie aplikácie. Informácie o architektonickom vzore boli čerpané z [16] a informácie o Model-View-Controllery boli čerpané z [23] a [11].

4.1 Návrh

4.1.1 Implemenačné prostredie

Na začiatku bol použitý programovací jazyk Python a vývojové prostredie Spyder. Tento jazyk bol zvolený hlavne pre jeho jednoduchosť a intuitívnosť pri práci so zvukom. Boli použité hlavne knižnice NumPy a SciPy, pre prácu zo zvukovými nahrávkami a matplotlib pre vizualizáciu dat. Keďže výsledná aplikácia mala byť pre mobilné zariadenia, tak ďalším prostredím bolo Android studio a jazyk Java. Keďže aplikácia mala pracovať v reálnom čase, tak trebalo využiť triedy, ktoré ukladajú informácie z mikrofónu telefónu a vysielajú do slúchadiel, MediaRecord a AudioTrack.

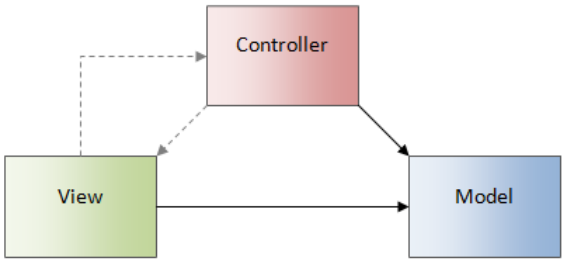
4.1.2 Architektonický vzor

Je to vzor popisujúci problém pri návrhu a implementácii softwaru. Pomáha definovať, akým spôsobom majú jednotlivé časti, podsystémy a systémy medzi sebou pracovať. Vzor ponúka riešenie, ktoré sa dá použiť v rôznych kontextoch. Dobrá architektúra redukuje riziká znižuje riziko spojené s vytváraním technického riešenia aplikácie. Je flexibilná pre prípad aplikovania zmien v budúcnosti a zároveň schopná sa prispôsobiť požiadavkám, ktoré neboli známe na začiatku alebo boli plánované na neskoršiu fázu vývoja.

4.1.3 Model-View-Controller

Model-view-controller(MVC), ktorý rozdeľuje dátový model aplikácie, užívateľské rozhranie a riadiacu logiku na tri nezávislé časti tak, že zmena niektorej z nich ma minimálny vplyv na ostatné. Vytvorenie aplikácie s využitím architektúry MVC vyžaduje vytvorenie troch

nezávislých komponentov Model, View a Controller, obrázok 4.1. Vzor nedefinuje len role objektov, ale definuje aj spôsob, akým medzi sebou komunikujú.



Obrázek 4.1: Model-View-Controller[5]

Model

Objekty Modelu zapuzdrujú údaje špecifické pre aplikáciu a definujú logiku a výpočty, ktoré tieto údaje spracovávajú. Objekt modelu môže napríklad reprezentovať znak v hre alebo kontakt v adresári. Objekty modelu môžu mať jeden alebo mnoho vzťahov s ostatnými objektami. Veľa údajov, ktoré sú súčasťou trvalého stavu aplikácie (či už je tento trvalý stav uložený v súboroch alebo v databázach), by sa po načítaní do aplikácie malo nachádzať v objektoch modelu. Model nevie, odkiaľ dáta v parametroch prichádzajú a ani ako budú výstupné dáta použité. Pretože modelové objekty prezentujú znalosti s konkrétnou doménou, tak môžu byť znovu použité v podobnej situácii. Užívateľské vstupy z View, ktoré vytvárajú alebo upravujú údaje, komunikujú cez objekt Controlleru a vedú k vytvoreniu alebo aktualizácii objektu Modelu. Ak sa zmenia objekty Modelu (napríklad prídu nové dáta z internetu), oznámia to Controlleru, ktorý aktualizuje príslušný objekt View.

View

View objekt je objekt v aplikácii, ktorý môže používateľ vidieť. View objekt dokáže meniť, to čo užívateľ vidí na ploche a zároveň dokáže reagovať na podnety od užívateľov. Hlavným poslaním tohto objektu je zobrazenie údajov z objektov Modelu aplikácie a umožnenie úpravy týchto údajov. Napriek tomu sú typicky objekty View oddelené od objektov Modelu v aplikácii MVC. View sa dozvedá o zmenách v objektoch Modelu pomocou Controlleru a a tiež posiela zmeny vyvolané užívateľom cez Controller do Modelu, napríklad text zadaný v textovom poli.

Controller

Objekt Controlleru pôsobí ako prostredník medzi jedným alebo viacerými objektami View a medzi jedným alebo viacerými objektami Modelu. Controller je teda objekt, pomocou ktorého sa View dozvedá o zmenách v objektoch Modelu a naopak. Tieto objekty môžu tiež vykonávať nastavenia a koordináciu úloh pre aplikáciu a riadiť životný cyklus iných objektov. Objekt Controlleru interpretuje akcie užívateľa vykonané v objektoch zobrazenia a odosiela nové alebo zmenené údaje modelovej vrstve. Keď sa zmenia objekty Modelu, tak Controller odošle do objektu View nové údaje a ten ich zobrazí.

Princíp

Aj keď môže existovať množstvo spôsobov realizovania MVC, všeobecne platí:

1. Užívateľ zrealizuje nejakú akciu v užívateľskom rozhraní (napríklad stlačenie tlačidla).
2. Controller obdrží oznámenie o tejto akcií z objektu View.
3. Controller pristúpi k Modelu a v prípade potreby ho aktualizuje na základe operácie užívateľa.
4. Aplikačná logika v Modely spracuje zmenené dáta (napríklad aplikácia filtra na odšumenie)
5. Komponent View použije zaktualizovaný model pre zobrazenie aktualizovaných dat užívateľovi (napríklad zmení farbu deaktivovaného tlačidla).
6. Užívateľské rozhranie čaká na ďalšiu akciu o užívateľa, ktorá cyklus opätovne zaháji.

Výhody vzoru

- Jednoduché sprístupnenie pre rôznych klientov. Pre nového klienta je potrebná iba zmena View, v niektorých prípadoch aj špeciálna zmena Controlleru. Model, ako kľúčová časť systému ostáva nezmenená.
- Minimalizácia duplicitného kódu. Napríklad bez oddelenia a zapuzdrenia Modelu by sa pre každé nove View musela programovať aplikačná logika na novo.
- Rozdelenie vývojarských rolí. Možnosť paralelne a nezávisle pracovať na všetkých troch častiach a známa musí byť len rozhranie na prepojenie jednotlivých objektov. Zmeny sa taktiež realizujú len v danom komponente a nerozhádžu ostatné časti.
- Znovupoužiteľnosť kódu.
- Vysoká komplexnosť návrhu a ľahká rozšíriteľnosť.

4.2 Implementácia

4.2.1 Python

V prvotnej fáze bol vývoj aplikácie realizovaný v jazyku Python verzie 2. Z tohto dôvodu bolo ako vývojové prostredie použité prostredie Anaconda. Toto prostredie bolo zvolené pre jednoduché spracovanie a vizualizáciu zvukových súborov. Aplikácie pracovali v režime offline. Na vstupe mali dva argumenty, prvý argument obsahoval cestu k súboru typu .wav, ktorý obsahoval zašumenú zvukovú nahrávku zo vzorkovacou frekvenciou 8000 kHz, druhý parameter obsahoval cestu a meno súboru typu .wav, do ktorého sa mala odšumená nahrávka uložiť.

Spektrálne odčítanie

Aplikácia obsahuje tri funkcie, *main*(), ktorá obsahuje spracovanie vstupných parametrov, otvorenie vstupného súboru na čítanie, volanie hlavnej funkcie filtru a zápis odfiltrovanej nahrávky do súboru. Ďalej aplikácia obsahuje funkciu $VAD\_detector$, ktorej vstupom sú dve

parametre, vzorky vstupnej nahrávky a dĺžka jedného rámca. Táto funkcia je implementovaná pomocou algoritmu energetického detektoru, sekcia 2.4.2, a výstup funkcie tvorí pole o veľkosti počtu rámcov, kde každý rámec má priradenú hodnotu „1“ alebo „0“ podľa toho, či bola detekovaná reč alebo nie. Poslednou, treťou funkciou je funkcia *spectral_subtraction*, ktorej vstupom je zašumený súbor. Funkcia tvorí jadro aplikácie a využíva metódu spektrálneho odčítania popísanú v sekcii 2.5.1. Využíva funkciu $VAD\_detector$ a výstupom je pole vzorkou, ktoré tvoria odšumenú nahrávku.

Local-mean filter

Aplikácia druhého filtra obsahuje taktiež tri funkcie, $main()$, $VAD\_detector$, ktoré sú rovnaké ako v prípade spektrálneho odčítania, sekcia 4.2.1, a funkciu $local\_mean\_filter()$, ktorej vstupom sú vzorky zašumenej nahrávky. Základ funkcie tvorí algoritmus local-mean filter, popísaný v sekcii 2.5.2.

Wienerov filter

Program tretieho filtru obsahuje funkciu $main()$, ktorá obsahuje načítanie vstupných argumentov, volanie funkcie filtru, prístup k datam z nahrávky a zapisovanie odfiltrovaných dat do súboru. Ďalej obsahuje funkciu $VAD\_detector$, ktorá spracuváva vstupné hodnoty a na výstup dáva pole obsahujúce hodnoty pre rámce z rečov a bez nej. Tretia funkcia, je funkcia $wiener\_filter$, obsahuje spracovanie vstupých dat pomocou algoritmu popísanom v sekcii 2.5.3 a výstup tvorí odšumená postupnosť data.

Knižnice

Najpoužívanejšie knižnice, ktoré boli použité v programoch sú:

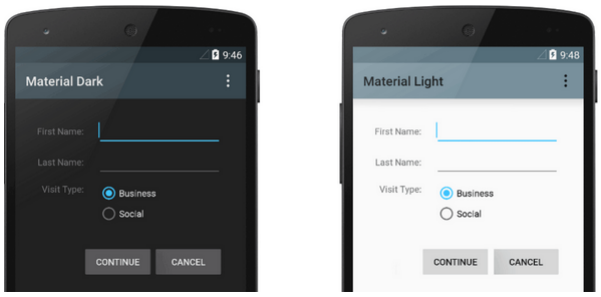
- SciPy [20] – je to knižnica, ktorá obsahuje veľké množstvo balíčkov pre vedecké výpočty. V aplikácii sa používa hlavne balíček *io* pre načítanie a zapisovanie wav súborov, potom balíček $fftpack$, pre diskrétnu Fourierovú transformáciu a spätnú diskrétnu Fourierovú transformáciu a balíček *signal*, ktorý obsahuje funkcie pre spracovanie signálov.
- Numpy [17] – je to základný balík pre vedecké výpočty v Pythone. Obsahuje okrem iného objekt n-rozmerného poľa, užitočnú lineárnu algebru, Fourierovu transformáciu alebo možnosti náhodných čísel. V aplikácii bola hlavne použitá na tvorbu a prácu s jedno alebo dvoj-rozmerným poľom. Funkcie ako *abs* alebo *angle* na získanie magnitúdy a fázy signálu, *correlate* pre použitie vzájomnej korelácie alebo *mean* a *sum* na priemer a sčítanie prvkov v poli.
- matplotlib [10] – je to Python 2D grafická knižnica, ktorá produkuje grafy, histogramy, výkonové spektra, stĺpcové grafy, chybové grafy a iné v rôznych formátoch a interaktívnych prostrediach. V aplikácii bola použitá na vizualizáciu dosiahnutých pokrokov a výpočtov pri spracovaní zvukových nahrávok.
- sys modul [1] – modul poskytuje funkcie a premenné, ktoré sa používajú na manipuláciu s rôznymi časťami behového prostredia Python. Presnejšie na načítanie a spracovanie vstupných argumentov programu.
- soundfile [4] – knižnica pre čítanie a zápis zvukových súborov. Špeciálne bola použitá kvôli možnosti automaticky načítať normovaný súbor v rozmedzí <-1,1>.

4.2.2 Java

Výsledná aplikácia mala byť vo forme mobilnej aplikácie, preto bolo ako vývojové prostredie vybrané Android studio a programovací jazyk Java. Android studio nie je vhodným prostredím pre spracovanie a vizualizáciu zvukových súborov, preto bol vývoj vhodného filtru tvorený najprv v Pythone. Ako metóda pre mobilnú aplikáciu bola zvolená metóda local-mean filter, ktorá je popísana v sekcii 2.5.2.

Trieda *MyView* a resources

Resources su dodatočné súbory a statický obsah, ktorý kód používa, napríklad bitmapy, definície rozloženia, používateľské rozhranie, pokyny na animáciu a ďalšie. Vždy by sa tu mali nachádzať externé zdroje ako napríklad obrázky, aby mohli ostať nezávislé. Tiež by tu mali byť alternatívne definície zoskupené do špeciálne pomenovaného priečinka. Počas behu, Android používa špecifický zdroj na základe aktuálneho nastavenia. Napríklad iné užívateľské prostredie na základe veľkosti obrazovky. *Style* je kolekcia atribútov, ktoré určujú vzhľad jedného zobrazenia. *Style* môže špecifikovať atribúty ako farba pozadia, veľkosť písma, farba písma, pozícia a rozloženie prvkov na displeji a iné. Téma je typ štýlu, ktorý sa aplikuje na celú aplikáciu alebo hierarchiu zobrazenia, nie len na individuálne zobrazenie. Keď sa aplikuje štýl ako téma, tak na každé zobrazenie v aplikácii sú aplikované atribúty definované v *style*, obrázok 4.2. *Strings* obsahuje textové reťazce použiteľné v aplikácii s voliteľným štýlom textu a formátovaním. *Colors* obsahuje farby, voliteľné odtiene farieb, ktoré môžu byť priradené prvkom aplikácie. Priečinok mipmap obsahuje informácie o vzhľade ikony aplikácie a jej pozadí. Priečinok drawable obsahuje všekty obrázky, ktoré sú použité v aplikácii. A nakoniec *Activity_xml* definuje štruktúru užívateľského rozhrania v aplikácii, napríklad v aktivite. Všetky prvky v rozložení sú vytvorené pomocou hierarchie objektov *View* a *ViewGroup*. Zobrazenie zvyčajne kreslí niečo, s čím môže užívateľ pracovať.



Obrázek 4.2: Príklad zmeny témy v Android studio. Prevzaté z []

Trieda *MyView* obsahuje funkcie, ktoré pracujú z grafickým užívateľským rozhraním. Nastavujú text, reagujú na stlačenie tlačidiel, nastavujú tlačidla na neaktívne a informujú controller o zmenách. Taktiež zobrazujú vyskakovacie okno, ktorým aplikácia žiada o povolenia pre aplikáciu.

Trieda *MainActivity*

Táto trieda pôsobí, v chierarchii Model-View-Controller ako Controller, ktorý riadi celý chod aplikácie. Ak užívateľ urobí nejakú akciu v grafickom rozhraní, tak tak táto trieda je

o tom informovaná a reaguje na to, tak že odošle triede $MyView$ informáciu a zmene textu alebo zmene farby tlačidla a zároveň, ak je to potrebné tak vola funkcie Modelu.

Triedy $ArrayOperation, MyFilter, VAD\_detector, WavRecorder$

Táto skupina tried patrí do kategórie Modelu. Trieda $ArrayOperation$ obsahuje funkcie pre prácu z reťazcami, ako odčítanie dvoch reťazcov, priemer z reťazca alebo funkcie na prevod reťazca obsahujúci čísla typu integer na čísla typu double. Obsahuje tiež funkcie na hammingove okno a vzájomnú koreláciu. Triedy $MyFilter$ a $VAD\_detector$ obsahujú algoritmy na redukciu šumu, resp. na detekciu reči vo zvuku. Na vstupe majú postupnosť vzoriek a ich výstupom je odšumená postupnosť vzoriek, resp. označenie sektora, v ktorom bola detekovaná reč. Najhlavnejšia trieda, $wavRecorder$, obsahuje načítavanie zvuku z mikrofónu mobilu, obsahuje funkcie na ukladanie audia do súborov a taktiež vysielanie zvuku do reproduktora/slúchadiel mobilu. Nastavuje sa v nej vzorkovacia frekvencia zvuku, volajú sa tu triedy na rečový detektor a zvukový filter.

Dôležité balíčky

V tejto sekcii si spomenieme najpodstatnejšie balíčky, ktore výrazne pomohli s tvorbou aplikácie. Pre viacej informácii navštívte https://developer.android.com/docs:

- android.view – poskytuje triedy, ktoré tvoria základné používateľské rozhranie a spravujú základné rozloženie obrazovky a interakciu s užívateľom.
- java.io – poskytuje vstup a výstup sytému prostrednícstvom dátových tokov, serializácie a súborového systému.
- java.util – obsahuje kolekcie, rozhrania, model udalosti, dátum a čas a rôzne triedy nástrojov (generátor náhodných čísel, bitove pole, ...)
- android.widget – je to balíček, ktorý obsahuje (väčšinou vizuálne) prvky užívateľského rozhrania, ktoré sa majú použiť na obrazovke zariadenia. Taktiež poskytuje možnosť navrhnúť vlastný design.
- android.media – poskytuje triedy, ktoré spravujú mediálne rozhranie audio a video. Rozhranie Media API sa používa na prehrávanie a, v niektorých prípadoch, na nahrávanie mediálnych súborov. To zahŕňa audio (prehrávanie MP3 alebo iných hudobných súborov, zvonenie, zvukové efekty) a video (prehrávanie videa vysielaného cez internet alebo z lokálneho úložiska). Ďalšie triedy poskytujú možnosť rozpoznať tváre ľudí, ovládať smerovanie zvuku, ovládať zvonenia a vibrácie telefónu a iné.

Kapitola 5

Výsledná aplikácia

Aplikácia bola vyvíjaná pre operačný systém Android. Ako referenčné zariadenie bolo použité zariadenie Samsung Galaxy S9+ s verziou operačného systému 9.

5.1 Android

Android je rozsiahla „open source“ platforma, ktorá vznikla hlavne pre mobilné zariadenia. Zahŕňa v sebe operačný systém, založený na jadre Linuxu, používateľské rozhranie a aplikácie. Vyvíja ho konzorcium Open Handset Alliance, ktorého cieľom je progresívny rozvoj mobilných technológii, ktoré budú mať výrazne nižšie náklady na vývoj a distribúciu, a zároveň spotrebiteľom prinesú inovatívne, používateľsky prívetivé, prostredie. Jadro Androidu bolo navrhnuté pre prácu na rôznom hardvéri a použiteľné bez ohľadu na chipset, veľkosť a rozlíšenie obrazovky. Android bol predstavený v roku 2007, pričom prvé komerčné zariadenie so systémom Android bolo spustené v septembri 2008. Operačný systém prešiel viacerými hlavnými verziami, pričom súčasná verzia je 9 „Pie“, vydaný v auguste 2018. Zdrojový kód Androidu je známy ako Android Open Source Project a je primárne licencovaný pod licenciou Apache. Android je najpredávanejší operačný systém na svete pre mobilné zariadenia od roku 2011. Od mája ma viac ako dva miliardy aktívnych používateľov mesačne.Informácie dostupné z [19] a [3]

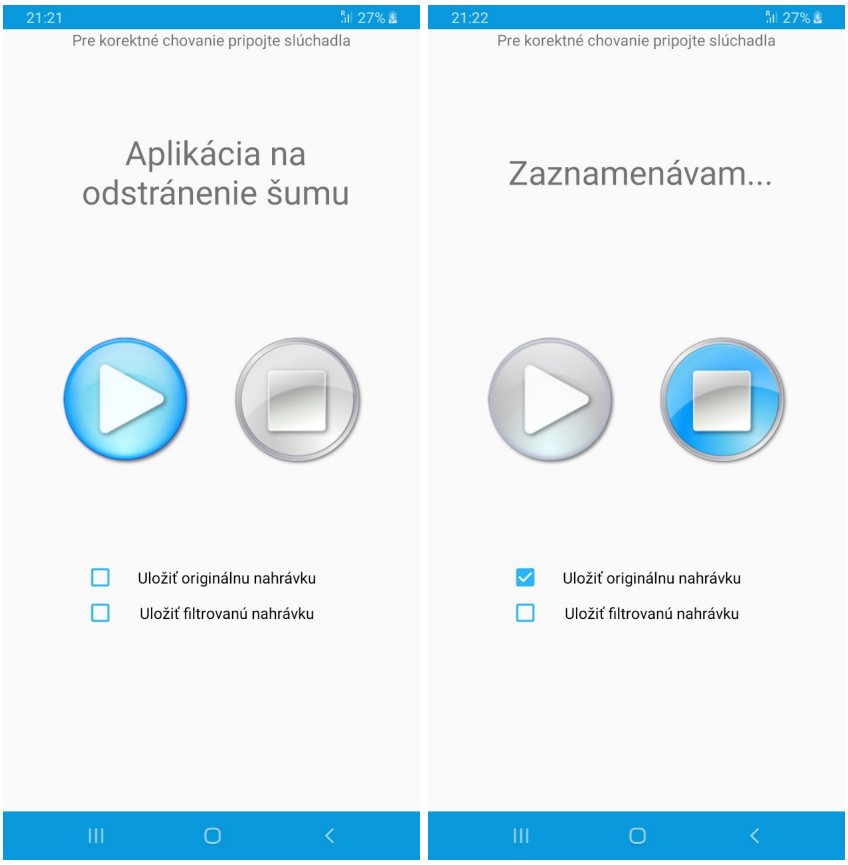
5.2 Referenčné zariadenie

Aplikácia bola primárne vytvárané a testované na zariadení Samsung Galaxy S9+. Toto zariadenie poskytuje operačný systém 9, chipset Exynost 9810, osemjadrový procesor (4x2,7 GHz + 4x1,79 GHz) a 6 GB pamäť RAM. Podrobnejšie informácie na https://www.samsung.com/us/smartphones/galaxy-s9/specs/.

5.3 Aplikácia

Výsledná aplikácia má jednoduché užívateľské rozhranie, viz obrázok 5.1. Keďže dôraz bol kladený na funkcionalitu odšumovania, neobsahuje žiadne prvky, ktoré by uľahčovali analýzu spracovania signálov. Užívateľ si ma možnosť pred spustením nahrávania možnosť vybrať možnosť, či chce uložiť nahrávanú a filtrovanú nahrávku do mobilu.Tie sú dostupne vo formáte .wav v internej pamäti telefónu. Samotné nahrávanie sa spúšťa tlačidlom play a zároveň sa v hornej časti zobrazí správa, ktorá informuje užívateľa, že sa začalo nahrávanie.

Pri nahrávaní a zapnutej hlasitosti ma užívateľ okamžitý výstup v sluchátkach alebo v inom výstupnom zvukovom zariadení. Neodporúča sa používať reproduktor na tom istom mobile, na ktorom sa používa mikrofón na nahrávanie, pretože vzniká nepríjemná ozvena. Užívateľ ukončuje nahrávanie a prehrávanie tlačidlom stop.



Obrázek 5.1: Výsledná aplikácia

Kapitola 6

Hodnotenie aplikácie

Pri vyhodnocovaní aplikácie bol použitý spôsob rozhovoru, kde som pozoroval správanie užívateľa pri reálnom používaní aplikácie a ten mi dával okamžitú spätnú väzbu. Spôsobom osobného testovania bolo aj zamedzené aby niekto hodnotil nezmyselne kladne alebo záporne a tým znehodnotil výsledky. Užívateľ mal možnosť s mobilom nahrať a vypočuť si predpripravené nahrávky s rôznou hodnotou pomeru signálu a šumu. Následne užívateľ odpovedal na pár jednoduchých otázok o používaní aplikácie a celkovom dojme.

6.1 Dotazník

Užívateľom boli postupne prehrávané nahrávky. Najprv to boli nahrávky, kde v pozadí sa vyskytoval rôzny ruch, od zvukov tlačiarne, cez pípanie pokladne až po zvuk kolotočov. V každej nahrávke sa postupne zvyšovalo SNR, ktoré začínalo na úrovni 5 dB a pokračovala až na úroveň 20 dB. V druhej fáze to boli nahrávky zo stacionárnym zvukom, čiže zvukom, ktorý sa v čase veľmi nemení, zvuk vysávača alebo dažďa. Každému užívateľovi boli pustené dve nahrávky z danej kategórie a ten pridelil každej kategórii výslednú známku. Otázky boli rozdelené do dvoch časti, a to takých, že na otázky 1 až 3 užívateľ odpovedá len raz, a na otázky 4 a 5 odpovedá po vypočutí každej nahrávky.

1. Ako hodnotíte jednoduchosť ovládania? (1 – 10)
2. Ako hodnotíte vzhľad aplikácie? (1 – 10)
3. Celkové hodnotenie aplikácie? (1 – 10)
4. Ako hodnotíte redukciu šumu v pozadí? (1 – 10)
5. Ako hodnotíte kvalitu reči po odšumení nahrávky? (1 – 10)
6. Pripomienky.

6.2 Vyhodnotenie

Dotazník vyplnilo 7 ľudí. S každým užívateľom som bol v priamom kontakte a každý užívateľ vyplnil všetky odpovede. Väčšina užívateľ hodnotila dizajn takejto aplikácie ako druhotný a dôležité pre nich bolo ako kvalitne funguje. Ako môžeme vidieť v tabuľke Vo všeobecnosti bolo odšumenie hodnotené lepšie ako kvalita hlasu. Užívatelia sa najviac sťažovali na rušivé zvuky v pozadí ako je praskanie a nezrozumiteľnosť reči.

č.otázky	1	2	3	4	5
priemer. známka	8	6	5	7	6

Tabulka 6.1: Zhrnutie výsledkov

Ako môžeme vidieť v tabuľke 6.1, tak užívatelia boli spokojnejší s kvalitou odšumenia ako s kvalitou hovoreného slova po odšumení. Tiež nahrávky zo stacionárnym šumom v pozadí boli hodnotené kladnejšie. Podrobnejšie výsledky sa nachádzajú v prílohe B.

Pri testovaní bolo zistené, že ak je pri nahrávaní detekovana reč, čiže niekto práve rozpráva do mikrofónu, po dlhší časový úsek a zároveň sa mení hladina zvuku v pozadí, napríklad začne silnejšie pršať alebo prichádza nejaké auto, tak aplikácia nie je schopná aktualizovať hladinu šumu. Tá sa aktualizuje len v častiach kde nie je detekovaná reč a tým pádom sa znižuje schopnosť aplikácie odšumieť danú situáciu.

Kapitola 7

Záver

Hlavným cieľom práce bolo vytvoriť mobilnú aplikáciu, ktorá by bola schopná v reálnom čase zredukovať nežiadúci šum z hovorenej reči. V práci sa mi podarilo implmentovať tri typy filtrov v jazyku Python a následne som si jeden z nich vybral a naprogramoval ho v jazyku Java pre mobilné zariadenie. Na začiatku som si zopakoval znalosti s ISS a rozšíril ich o znalosti z odborných kníh a článkov. Z nich bola v úvode zhrnutá teória súvisiaca so spracovaním signálov, boli popísané typy filtrov a podrobnejšie popísane filtre na redukciu šumu. Pri vývoji som si precvičil programovanie v jazyku Python a hlavne prácu zo zvukovými signálmi. Taktiež som si zopakoval jazyk Java, zoznámil sa s prostredím Android Studio a programovaním mobilných aplikácii.

Do budúcna by trebalo vylepšiť kvalitu hovorenej reči, keďže tá pri testovaní najviac vadila ľuďom. Ďalej by bolo vhodné pridať do aplikácie možnosť výberu filtru a dovoliť užívateľovi meniť vzorkovaciu frekvenciu. Taktiež by bolo vhodné doplniť do mobilnej aplikácie vizualizáciu signálu, prípadne nejaké možnosti ďalšieho spracovania.

Literatura

[1] Sys modul.
URL https://docs.python.org/2/library/sys.html

[2] ADAMEC, M.: *MODERNÍ ROZPOZNÁVAČE ŘEČOVÉ AKTIVITY*. FIT VUT v Brně, 2008.
URL
https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=5788

[3] handset alliance, O.: Industry Leaders Announce Open Platform for Mobile Devices. 2007.
URL http://www.openhandsetalliance.com/press_110507.html

[4] Bechtold, B.: SoundFile.
URL https://pypi.org/project/SoundFile/

[5] Bernard, B.: *Úvod do architektury MVC*. 2009.
URL https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[6] Chanwoo, K.; M.Stern, R.: *Robust Signal-to-Noise Ratio Estimation Based on Waveform Amplitude Distribution Analysis* . September 2008.
URL http://www.cs.cmu.edu/afs/cs/user/robust/www/Papers/KimSternIS08.pdf

[7] Chernenko, S.: *Mean filter, or average filter*.
URL http://www.librow.com/articles/article-5

[8] Chiacchio, M.: *Signal-to-Noise (S/N) Ratio: Definition & Formula* .
URL https://study.com/academy/lesson/signal-to-noise-s-n-ratio-definition-formula.html

[9] Gonzales-Barajas, J.; Montenegro, D.: *Average Filtering: Theory, Design and Implementation*. 04 2016.
URL https://www.researchgate.net/publication/299485551_Average_Filtering_Theory_Design_and_Implementation

[10] Hunter, J. D.: Matplotlib.
URL https://matplotlib.org/

[11] Inc, A.: *Cocoa Core Competencies*. Apple Inc., 2018.
URL https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[12] Jan, J.: *Číslicova filtrace, analýza a restaurace signálu*. VUTIUM, 2002, ISBN 80-214-2911-9.

[13] Jiří Poruba, L. M.: Odfiltrování rušivých signálů ze zašumělé řeči. November 2002.
URL http://www.elektrorevue.cz/clanky/02047/index.html#Kap3

[14] Kibbe, H.: *Finite Impulse Response Filters Using Apple's Accelerate Framework – Part I* . Januar 2014.
URL http://hamiltonkibbe.com/finite-impulse-response-filters-using-apples-accelerate-framework-part-i/

[15] Kohn, A.: *Autocorrelation and crosscorrelation methods*. 2006.
URL https://www.researchgate.net/publication/229766295_Autocorrelation_and_Cross-Correlation_Methods

[16] Microsoft: *Microsoft application architecture guide*. Microsoft, 2009, ISBN 9780735627109.
URL https://www.intertech.com/Downloads/eBook/ApplicationArchitectureGuide.pdf

[17] Oliphant, T.: NumPy.
URL https://www.numpy.org/index.html

[18] Pennock, S.: *Accuracy of the Perceptual Evaluationof Speech Quality (PESQ) algorithm* .
URL http://wireless.feld.cvut.cz/mesaqin2002/full09.pdf

[19] Statista: Number of available applications in the Google Play Store from December 2009 to December 2018. 2019.
URL https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[20] Travis Oliphant, E. J., Pearu Peterson: SciPy.
URL https://www.scipy.org/

[21] Vaseghi, S. V.: *Advenced Digital Signal Processing and Noise Reduction*. WILEY, 2008, ISBN 978-0-470-75406-1.
URL http://sharif.ir/~bahram/sp4cl/MainReferences/advancedDigitalSignalProcessingAndNoiseReduction.pdf

[22] Vondrášek, M.: *Odhad SNR řečového signálu snímaného hlučném prostředí*. Fakultra elektrotechnická ČVUT v Prahe, 2004.
URL http://noel.feld.cvut.cz/speechlab/publications/032_diplomka04.pdf

[23] Vrážel, D.: *ROZŠIŘITELNÝ INFORMAČNÍ SYSTÉM SDRUŽENÍ SDC S VÍCEVRSTVOU ARCHITEKTUROU*. FIT VUT v Brně, 2007.
URL https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=115912

[24] Černocký, J.: Zpracovacovaní řečových signálů. FIT VUT v Brně, Prosinec 2006.
URL https://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf

Příloha A

Obsah CD

- BP.pdf – elektronická verzia tohto dokumentu
- doc – zdrojový kód tejto práce
- srcPython – zdrojové súbory v Pythone
- srcJava – zdrojové súbory v Jave
- denoiser.apk – android aplikácia

Příloha B

Výsledky dotazníka

V tejto časti budú uvedené podrobnejšie výsledky dotazníka. V tabuľke B.1 sú uvedené výsledky hodnotenia redukcie šumu, v tabuľke B.2 sú výsledky hodnotenia kvality hlasu po redukcii a v tabuľke B.3 sú odpovede na otázky 1-3.

6-9dB	9-12dB	12-15dB	15-18dB	18-21dB	5-12dB	12-20dB
6	5	7	5	7	8	8
4	5	7	7	7	6	7
5	6	6	6	6	7	7
3	3	4	5	4	6	6
4	6	4	6	8	6	8
4	4	5	6	7	6	7
6	6	6	6	9	7	8

Tabulka B.1: Výsledky hodnotenie šumu

6-9dB	9-12dB	12-15dB	15-18dB	18-21dB	5-12dB	12-20dB
7	9	9	9	7	4	6
5	6	6	6	6	6	8
4	6	6	6	6	6	8
6	4	5	5	7	5	9
5	3	6	4	7	6	8
5	2	2	4	6	6	8
7	6	6	6	8	5	6

Tabulka B.2: Výsledky hodnotenie hlasu

Najčastejšie pripomienky:

- zvonenie telefónu v nahrávke pôsobí nepríjemne praskavo
- reč pôsobí sekavo, nerozumieť konce slov
- zvuk tlačiarne neodstránený, pôsobí to nepríjemne
- aplikáciu treba vylepšiť

1.Jednoduchosť	2.Vzhľad	3.Aplikácia
9	6	6
8	6	5
9	6	5
6	5	4
8	7	5
8	7	4
8	7	6

Tabulka B.3: Výsledky dotazníka

- odšumenie je v rečových pauzách dobré ale keď osoba rozpráva tak jej hlas je taký chrapľavý