
**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**


# NÁSTROJ PRO AUTOMATICKÉ ZAROVNÁNÍ TITULKŮ

**TOOL FOR AUTOMATIC SUBTITLE ALIGNMENT**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **DANIEL CHUDÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **TOMÁŠ MILET, Ing., Ph.D.**
**SUPERVISOR**



**BRNO 2023**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání bakalářské práce

144735

| | |
|---|---|
| Ústav: | Ústav počítačové grafiky a multimédií (UPGM) |
| Student: | **Chudý Daniel** |
| Program: | Informační technologie |
| Specializace: | Informační technologie |
| Název: | **Nástroj pro automatické zarovnání titulků** |
| Kategorie: | Počítačová grafika |
| Akademický rok: | 2022/23 |

Zadání:

1. Nastudujte formáty titulků, způsoby tvorby titulků, nástroje, problémy, které vznikají při různých video souborech.
2. Navrhněte aplikaci, která umožní automatické přečasování titulků na jiné vydání video souboru. Vytvořte, sesbírejte dataset různě přečasovaných titulků.
3. Implementujte navrženou aplikaci buď jako samostatný nástroj nebo jako součást, doplněk jiné, stávající aplikace.
4. Proměřte úspěšnost a kvalitu automatického přečasování.
5. Zhodnoťte a vytvořte demonstrační video.

Literatura:


- Alan V. Oppenheim, Ronald W. Schafer. Discrete-Time Signal Processing. Prentice Hall. p. 1. ISBN 0-13-216771-9. 1989.
- Diaz-Cintas, Jorge. Subtitling: theory, practice and research. The Routledge Handbook of Translation Studies (pp.285-299). RoutledgeEditors: Carmen Millán, Francesca Bartrina. January 2012.


Při obhajobě semestrální části projektu je požadováno:

- První dva body a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Vedoucí práce: | **Milet Tomáš, Ing., Ph.D.** |
| Vedoucí ústavu: | Černocký Jan, prof. Dr. Ing. |
| Datum zadání: | 1.11.2022 |
| Termín pro odevzdání: | 17.5.2023 |
| Datum schválení: | 31.10.2022 |

# Abstrakt

Hlavnou motiváciou tejto práce je zjednodušiť prečasovanie titulkov, kde vstupom sú originálny a upravený videosúbor a titulky korešpondujúce s originálnym videom. Príklad – editor videí musí vystrihnúť scénu z videa. Titulky korešpondujúce s vystrihnutou časťou videa musí ručne odstrániť a prečasovať časť titulkov, ktorá nasleduje za vystrihnutou sekciou. Nástroj túto prácu uľahčí práve tým, že ju automatizuje. Z ľubovolne upravenej verzie videosúboru (strihy), originálneho súboru a originálnych titulkov vznikne verzia titulkov, ktorá sedí na upravenú verziu videosúboru.

Cieľom je prispôsobiť originálne titulky na upravený videosúbor. Riešením je konverzia videosúborov na audio-súbory (.wav, wavfile), extrahovanie MFCC (Mel-frekvenčné cepstrálne koeficienty) a následne vzájomné porovnanie algoritmom Dynamic Time Warping (DTW). Z cesty zarovnania (výstup DTW) sa zistia rozdiely signálov (strihy vo videu) a na ich základe sa upravia titulky. Na otestovanie aplikácie bol vytvorený dataset public domain filmov a vlastných nahrávok. Vytvorená aplikácia poskytuje 69 – 90 % úspešnosť zarovnania titulkov na datasete s videami o dĺžke 1 – 60 minút.



# Abstract

The main motivation of this work is to simplify the retiming of subtitles, where the inputs are the original and edited video files and subtitles corresponding to the original video. Example – a video editor needs to cut a scene from a video. Subtitles that correspond with the clipped part of the video must be manually removed and the subtitle part that follows the clipped part must be manually re-timed. The tool makes this work easier by automating it. From an arbitrarily edited version of the video file (cuts), the original file and the original subtitles, a version of subtitles will be created that fits the edited version of the video file.

Simply put, the goal is to align the original subtitles with the edited video file. The solution is the conversion of video files to audio files (.wav, wavfile), extraction of MFCC (Mel-frequency cepstral coefficients) and subsequent mutual comparison with the Dynamic Time Warping (DTW) algorithm. From the alignment path (DTW output), signal differences (cuts in the video) are detected and subtitles are adjusted based on them. A dataset was created to test the application consisisting of public domain films and own recordings. The created application provides 69 – 90 % subtitle alignment success on a dataset that contains videos of length 1 – 60 minutes.



# Kľúčové slová

signál, titulky, algoritmus, strih, audio, video, waveform, SRT, DTW, MFCC, FFT, MSA, prevzorkovanie, podvzorkovanie, korelácia, prečasovanie, zarovnanie, synchronizácia, ffmpeg, tkinter

# Keywords

signal, subtitles, algorithm, cut, audio, video, waveform, SRT, DTW, MFCC, FFT, MSA, resampling, downsampling, cross-correlation, retiming, alignment, synchronisation, ffmpeg, tkinter


# Citácia

CHUDÝ, Daniel. *Nástroj pro automatické zarovnání titulků*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Tomáš Milet, Ing., Ph.D.

# Nástroj pro automatické zarovnání titulků

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Tomáša Mileta, Ph.D. Ďalšie informácie mi poskytol Ing. Jan Brukner. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Daniel Chudý
16. mája 2023


## Poďakovanie

Rád by som poďakoval pánovi Ing. Tomášovi Miletovi, Ph.D a Ing. Janu Bruknerovi za ich odbornú pomoc a podporu pri tvorbe tejto práce.

# Obsah

# Kapitola 1

# Úvod

V dnešnej digitálnej dobe zohráva audiovizuálny obsah kľúčovú úlohu v rôznych oblastiach, ako je zábava, vzdelávanie a šírenie informácií. Titulky slúžia ako kľúčový komponent pri sprístupňovaní videí širokému spektru publika. Manuálny proces zarovnania titulkov so zodpovedajúcim zvukom alebo videom však môže byť časovo náročný a únavný. Riešením tohto problému adresuje vývoj robustného a efektívneho nástroja (aplikácie) na automatické zarovnávanie titulkov. Koncept automatického titulkovania je pomerne nová praktika umožnená vzostupom umelej inteligencie (AI). na webe sa len ťažko hľadá video editor, ktorý túto (často platenú) funkciu neposkytuje.

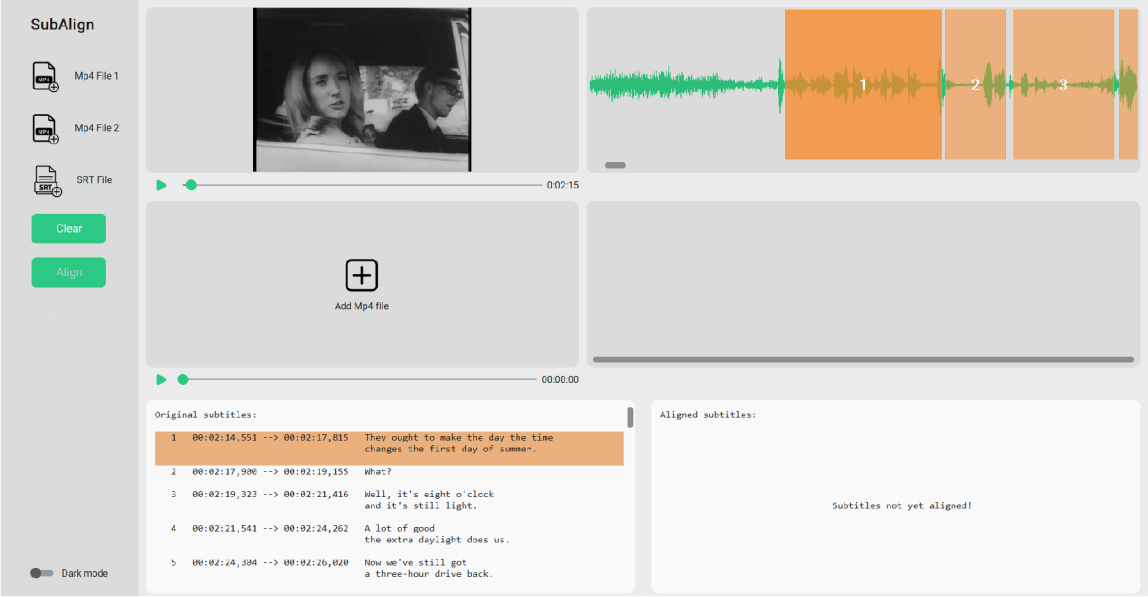
Cieľ práce je vytvoriť aplikáciu, ktorá užívateľovi umožní automaticky zarovnať titulky na ľubovolnú verziu videosúboru. Na tvorbu aplikácie **SubAlign** sa zvolil prístup bez použitia umelej inteligencie, ktorý sa sústredí na spracovanie a zarovnanie signálov známymi a dostupnými algoritmami (DTW, MFCC, ...).

Táto práca skúma niekoľko kľúčových konceptov a metód týkajúcich sa spracovania signálov, titulkov a algoritmov zarovnania. Na poskytnutie holistického porozumenia riešeného problému sa v kapitole 2 popisuje množstvo technológií a algoritmov (sekcia 2.1) vrátane spracovania signálu pomocou vzájomnej korelácie (ang. cross-correlation), Dynamic Time Warping (DTW), Mel-frekvenčných cepstrálnych koeficientov (MFCC), Fast Fourier Transform (FFT) a zarovnania biologických sekvencií (ang. Multiple Sequnce Alignment, MSA). Pojem *waveform* je priblížený v sekcii 2.2. Sekcia 2.3 sa zaoberá rôznymi formátmi titulkov (SRT, WebVTT, SAMI, MicroDVD), ich využitím a rozdielmi. Sekcie 2.4 a 2.5 sa venujú technikám spracovania signálu, ako je prevzorkovanie (ang. resampling) a podzorkovanie (ang. downsampling), ktoré sú využité na prispôsobenie audio alebo video signálov pre efektívnu analýzu a zarovnanie. V kapitolách 3 a 4 je priblížený návrh štruktúry, implementácia a fungovanie aplikácie, ako aj použité knižnice a nástroje tretích strán (FFmpeg, Tkinter, ...). Výstup testovania aplikácie a zhodnotenie výsledkov popisuje kapitola 5.

## 1.1 Predstavenie aplikácie

Výsledkom tejto bakalárskej práce je aplikácia **SubAlign** určená na synchronizáciu titulkov s videom.

Na uľahčenie užívateľskej interakcie je nástroj **SubAlign**[1] implementovaný pomocou populárneho multimediálneho nástroja FFmpeg a Python GUI knižnice založenej na jazyku Tcl, Tkinter. Táto voľba zaisťuje kompatibilitu s viacerými platformami (Windows, Linux, Mac-OS) a užívateľsky prívetivé grafické prostredie (obrázok 1.1), čo umožňuje tvorcom a prekladateľom jednoducho využívať nástroj pre potreby zarovnania titulkov. Samotný proces zarovnania využíva dva hlavné algoritmy MFCC a DTW a techniky na úpravu signálu ako downsampling na kompresiu audio-signálu. Predstavenie týchto algoritmov a techník je v kapitole 2 a ich presné využitie a zakomponovanie do aplikácie v kapitolách 3 a 4.




Obr. 1.1: Hlavné okno aplikácie. Ľavý panel slúži ako menu aplikácie, komponenty napravo od menu panelu slúžia na pridanie a prehrávanie videí, napravo je dvojica komponentov, do ktorých sa zobrazí waveform zvuku z videa a komponenty na spodku aplikácie sú na zobrazenie titulkov. Bližší popis aplikácie, jej používania a komponentov nájdete v sekcii 3.5 a v prílohe A.

## 1.2 Stav súčasných riešení

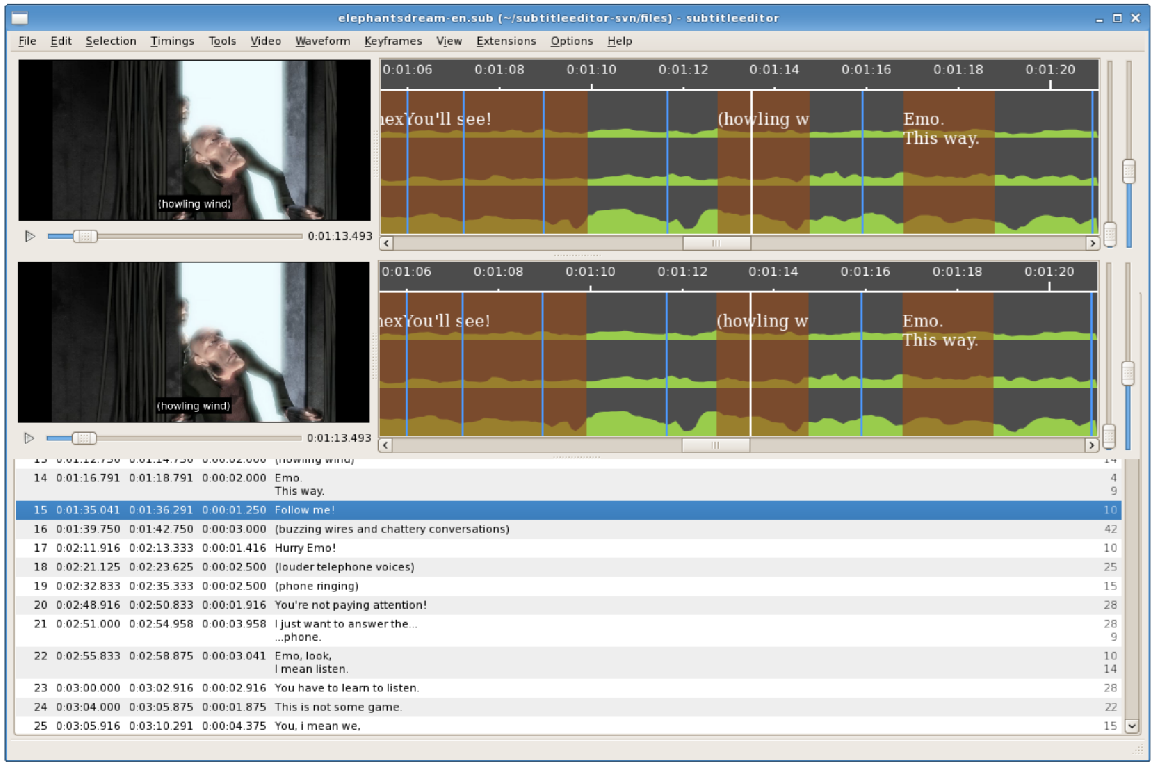
V tejto sekcii sa nachádza popis aplikácií či nástrojov, ktoré boli pre vývoj inšpiráciou alebo konkurenciou.

### 1.2.1 Subtitle Editor

Ako najväčšia inšpirácia na vzhľad a rozloženie výslednej aplikácie slúžil nástroj `Subtitle Editor`[2] pre operačný systém Linux. Umožňuje tvorbu titulkov, úpravu textu, času, dĺžky

[1] Návod na spustenie aplikácie sa nachádza v súbore `Readme.md` odovzdaného adresára.
[2] Subtitle Editor – http://kitone.github.io/subtitleeditor/.

a umiestnenia titulkov. Všetky úpravy na titulku sú reflektované v okne waveformu (obdĺžnik prehovoru) a vo videu (titulky sa automaticky zobrazia aj pri prehrávaní videa). Vzhľad aplikácie `Subtitle Editor` ilustruje obrázok 1.2.




Obr. 1.2: Upravená snímka aplikácie Subtitle editor slúžila taktiež ako mockup. Video a waveform na obrázku sú skopírované dvojmo – obrázok tak odpovedá potrebám cieľovej aplikácie. Dostupné z: https://www.debugpoint.com/3-great-subtitle-editors-in-linux-systems/.

### 1.2.2 Subaligner

V tejto podsekcii sa predpokladá, že čitateľ ovláda základné pojmy spracovania reči a hlbokého učenia (ang. deep learning).

Za najbližšiu konkurenciu by sa dal považovať nástroj `subalinger`[3]. Subaligner je všestranný nástroj na automatickú synchronizáciu titulkov s videom. Okrem zarovnania (synchronizácie) titulkov ponúka aj preklad titulkov či vytvorenie titulkov transkribovaním (**ASR**[4]) z videa. Na zarovnanie využíva MFCC (pozri sekciu 2.1.4) a neuronóvé siete (ang. Deep Neural Networks, DNN) s už predtrénovanými datasetmi, avšak poskytuje aj možnosť trénovať datasety vlastné. Poskytuje dve hlavné formy zarovnania, globálne zarovnanie, ktoré neurónovej sieti dodá vstup v podobe MFC koeficientov zvuku z videa a nesynchronizovaných titulkov a regionálne zarovnanie, ktoré vstup najprv rozdelí na časti.

Nástroj neposkytuje užívateľské rozhranie, práca s ním prebieha na príkazovom riadku. Ako už bolo vyššie spomenuté cieľová aplikácia tejto práce sa snaží titulky zarovnať bez

[3]subaligner – https://subaligner.readthedocs.io/en/latest/index.html
[4]ASR – automatic speech recognition, dostupné z: https://shorturl.at/jlqW7

použitia neurónových sietí (teda bez nutnosti použitia/trénovania datasetu). Na rozhodnutie použitia nástroja FFmpeg, MFCC a na výber mena cieľovej aplikácie nemala existencia a implementácia nástroja `Subalinger` vplyv.

### 1.2.3 Aeneas – forced alignment

Pojem **forced alignment**[5] predstavuje proces, ktorého cieľom je vytvoriť mapu synchronizácie textových prepisov (napr. titulkov) na zvuk (ľudskú reč) vo videu. Vstupom je teda nahrávka ľudskej reči a textový prepis tejto reči. Tento prístup využíva nástroj `aeneas` nasledovne – nahrávka sa prekonvertuje na mono WAV súbor a text sa pomocou **TTS** (text-to-speech) prevedie taktiež na mono WAV audio súbor. Zo súborov sa extrahujú MFC koeficienty, ktoré sa napokon zarovnajú pomocou DTW algoritmu. Celý postup je vysvetlený v dokumentácii nástroja[6].

[5] Zhrnutie nástrojov založených na forced alignment – dostupné z: https://github.com/pettarin/forced-alignment-tools

[6] Dokumentácia aeneas nástroja – dostupné z: https://github.com/readbeyond/aeneas/blob/master/wiki/HOWITWORKS.md

# Kapitola 2

# Teória

V tejto časti sú bližšie popísané už spomínané algoritmy ako DTW, Multiple Sequence Alignment (MSA) a techniky použité pri programovaní, waveform (.wav file), formáty titulkov, resampling a iné. Je priblížené ich fungovanie, vlastnosti a ich zmysel pre túto prácu.

## 2.1 Algoritmy

Nižšie rozvinuté princípy algoritmov nemusia byť nutne priamo použité v programe, ale sú inšpiráciou, uvádzajú do problému, alebo majú súvis s riešeným problémom.

### 2.1.1 Korelácia signálov

Korelácia signálov (vzájomná korelácia signálov, ang. cross-corelation) je funkcia, ktorá vyjadruje podobnosť dvoch signálov v čase. Laicky povedané, funkcia/graf korelácie vznikne, keď sa jeden signál prenesie cez druhý. Túto funkciu popisuje vzorec 2.1.

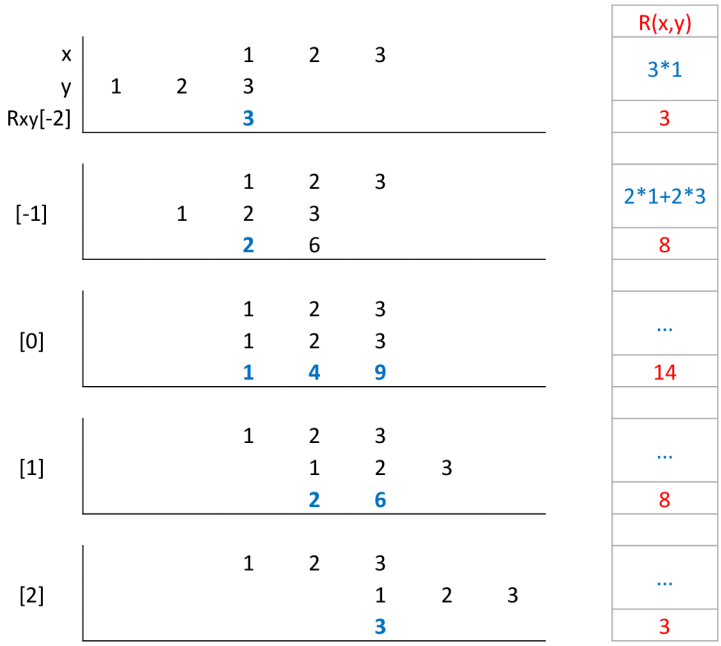
$$R_{xy}[n] = \sum_{m=-\infty}^{\infty} x[m]y[m-n], \qquad n = 0, \pm 1, \pm 2, ... \tag{2.1}$$

Rov. 2.1: Vzorec korelácie signálov, kde $x$ a $y$ sú korelované signály, výsledkom je pole korelačných koeficientov (popísané nižšie) $R_{xy}$, kde každý prvok je sumou súčinu zarovnaných prvkov signálov pre dané posunutie $n$. Vzorec (ako aj teoretické vlastnosti korelácie) prevzatý z práce Handbook of Digital Signal Processing [10, s. 667 – 668]. Vizuálne spracovanie vzorca vzájomnej korelácie signálov približujú obrázky 2.1 a 2.2.

.

#### Vlastnosti

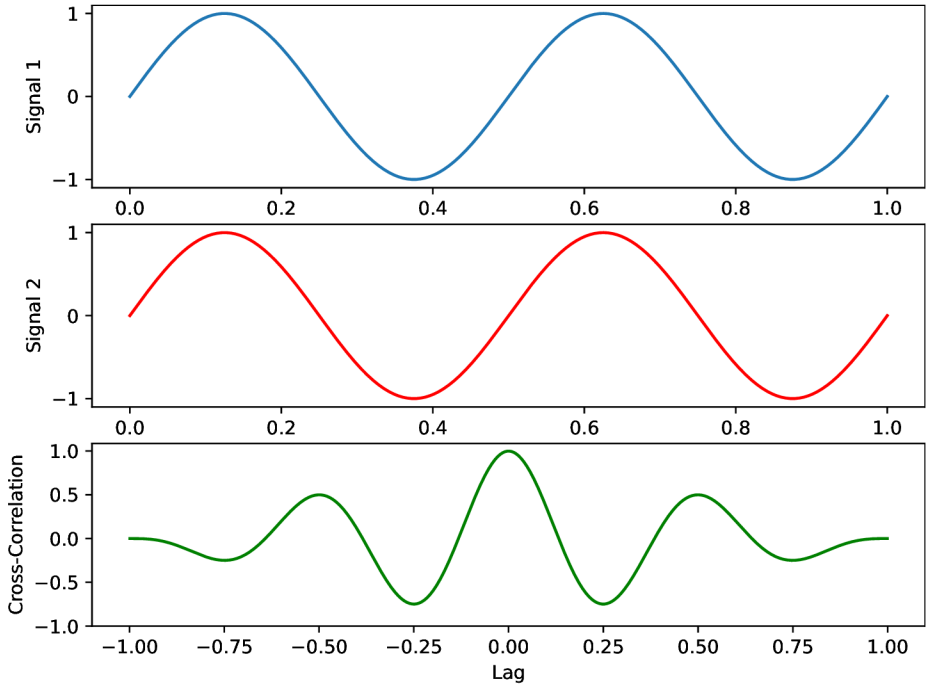
Korelácia slúži na zistenie podobnosti dvoch signálov, výstup je pole o veľkosti $|x| + |y| - 1$ (kde $x$ je referenčný signál a $y$ je signál posúvaný v čase), obsahujúce korelačné koeficienty, ktoré vyjadrujú mieru podobnosti signálov v jednotlivých časových bodoch. Index maxima tohto poľa vyjadruje časový bod, v ktorom sa dané signály najviac podobajú, t. j. vyjadruje kedy sú dané signály najlepšie zarovnané.




Obr. 2.1: Postup grafického riešenia vzájomnej korelácie signálov, zadané sú signály $x = \{1, 2, 3\}$ a $y = \{1, 2, 3\}$. Obrázok popisuje postup zhora nadol, začína sa so zarovnaním, kde je posledný prvok signálu $y$ zarovnaný s prvým prvkom signálu $x$, teda $R_{xy}[-2] = 1 \cdot 3 = 3$, výpočet pokračuje posunutím signálu $y$ o 1 prvok doprava, teda $R_{xy}[-1] = 1 \cdot 2 + 2 \cdot 3 = 8$ atď. Výsledok sa rovná $R_{xy} = \{3, 8, 14, 8, 3\}$. Koreláciu grafickou vizualizáciou postupu taktiež priblížil autor *Handbook of Digital Signal Processing* [10, s. 668].
Je dôležité poznamenať, že signály sú identické, čo vedie k výpočtu tzv. **autokorelácie**.

### Autokorelácia

Korelácia dvoch totožných signálov sa nazýva autokorelácia. Pri autokorelácii sú pole a graf korelačných koeficientov symetrické podľa stredu, t. j. index maxima je v strede výstupného poľa. Autokorelácia porovnáva referenčný signál so sebou samým v rôznych časových bodoch, to je užitočné napríklad na detekciu zašumeného periodického signálu (uvedené v práci *Analog Communication* [23, s. 86]). Autokorelácia má podobne ako korelácia odlišných signálov široké využitie v rôznych oblastiach.




Obr. 2.2: Vizualizácia vzájomnej korelácie signálov grafmi, na obrázku vidieť grafy vstupných signálov *Signal*1 a *Signal*2 a graf vzájomnej korelácie $Cross - Correlation$. Signály sú opäť totožné, takže podľa očakávania je graf korelácie symetrický podľa osy y (osa y je na obrázku posunutá na ľavý okraj grafu), v čase -0.25 a 0.25 sa dá pozorovať tzv. antikorelácia, to znamená, že signál *Signal*2 posunutý o 0.25 časových jednotiek doprava alebo doľava je najmenej podobný signálu *Signal*1. Maximálnu koreláciu dosahujú signály v čase 0, čo dokazuje totožnosť signálov.

### Aplikácia

Uplatnenie tejto techniky možno nájsť v mnohých oboroch, predovšetkým zistenie časového posunu sa používa pri rozpoznávaní reči (pozri. práca *Speech Recognition Using MATLAB and Cross-Correlation Technique* [12]), konkrétne pre rozpoznanie hovoriaceho.

Vďaka svojej vlastnosti nájsť výskyt signálu vo vnútri druhého signálu sa používa korelácia signálov napríklad v radaroch na detekciu cieľa v prostredí. V práci autorov D. Venu a N.V. Koteshwara Rao [24], ktorá sa zaoberá pasívnymi radarmi (radary, nevyužívajúce svoje vlastné vysielače, ale napr. FM vysielanie), autori navrhujú využitie algoritmu založeného na vzájomnej korelácii signálov na zistenie vzdialenosti cieľa od radaru. Všeobecne sa táto technika využíva na rozpoznávanie vzorov v sérii dát.

### Konvolúcia a vzájomná korelácia signálov

Konvolúcia je operácia veľmi podobná vzájomnej korelácii, kedy sa jeden signál „presunie“ cez druhý, ale pri konvolúcii sa signál, ktorý prislúcha druhému operandu $(x * y)$ otočí v čase (okolo osi y). Konvolúcia sa používa na transformáciu pôvodného signálu, napríklad aplikovanie low-pass filtru 2.5.1 (dolného priepustu) na signál.

### FFT a korelácia

Vzájomná korelácia môže najmä pri dlhších vstupných signáloch zabrať značné množstvo procesorového času. V každom kroku treba vykonať násobenie všetkých zarovnaných prvkov signálu pre všetky výsledné indexy korelácie. Tento všeobecný prístup ku korelácii má kvadratickú časovú zložitosť.

Riešením je použiť FFT. FFT alebo rýchla fourierova transformácia (ang. Fast fourier transform) je algoritmus, ktorý už je sám o sebe optimalizáciou klasickej diskrétnej fourierovej transformácie (DFT). Výpočet korelácie zrýchli FFT prevodom vstupných signálov na koeficienty fourierovej transformácie, t. j. komplexné čísla. Tieto polia sa potom jednoducho vynásobia tak, že sa prenásobí každý prvok s každým, pričom výsledok každého násobenia sa uloží do poľa. Výsledný signál vznikne prevedením poľa, ktoré je výsledkom násobenia fourierových koeficientov vstupných signálov, do časovej domény inverznou fourierovou transformáciou (IFT). Fakt, že operácia vzájomného prenásobenia prvkov fourierovej transformácie signálov (čiže ich reprezentácii vo frekvenčnej doméne) sa rovná ich konvolúcii (resp. vzájomnej korelácii) v doméne časovej sa nazýva konvolučný teorém, ktorý bližšie popisuje autor J. M. Blackledge v [2, s. 44 – 45]. Požitie FFT pri vzájomnej korelácii znižuje jej časovú zložitosť na logaritmickú $O(n \log n)$[1]. Rozdiel výkonnosti a efektívnosť FFT pri korelácii približuje obrázok 2.3.

[1]:
```
import numpy as np
import scipy.signal as sig
x = np.arange(0,1000,1)
y = np.arange(200,500,3)
%timeit -n 100 corr = sig.correlate(x, y, mode="full", method="direct")
%timeit -n 100 corr = sig.correlate(x, y, mode="full", method="fft")
```

```
94.6 µs ± 6.09 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
290 µs ± 34.5 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

[2]:
```
import numpy as np
import scipy.signal as sig
x = np.arange(0,10000,1)
y = np.arange(2000,5000,3)
%timeit -n 100 corr = sig.correlate(x, y, mode="full", method="direct")
%timeit -n 100 corr = sig.correlate(x, y, mode="full", method="fft")
```

```
8.54 ms ± 503 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
717 µs ± 23.5 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

Obr. 2.3: Porovnanie klasickej vzájomnej korelácie a korelácie s použitím FFT. V prvej bunke sa hľadá korelácia signálov o dĺžke $N_x = 1000$ a $N_y = 100$, kde vidieť že FFT koreláciu nijak nezefektívnil (skôr naopak), avšak v druhej bunke s dĺžkami $N_x = 10000$ a $N_y = 1000$ možno pozorovať značné zrýchlenie výpočtu (až 12-násobné zrýchlenie).

[1] Big O notation – https://en.wikipedia.org/wiki/Big_O_notation

### 2.1.2 DTW

Dynamic Time Warping (DTW) je algoritmus určený na zistenie podobnosti signálov rôznych dĺžok a rýchlostí (ľudský krok, reč). Určí podobnosť ľudského kroku aj v prípade rôznej dĺžky kroku alebo rozpozná podobnosť vyslovených slov aj keď rýchlosť vyslovenia sa nerovná. Vie namapovať jednotlivé vzorky dvoch podobných signálov na seba, pričom jeden z nich je viac rozprestrený v čase – týmto signály zarovná. V sekcii 4.4 autor knihy [15] M. Müller približuje ako je algoritmus DTW schopný vyhľadať určitú (sub)sekvenciu dát v dlhšej dátovej sekvencii. Uplatnenie nachádza pri rozpoznávaní reči, hĺbkovej analýze dát alebo na finančnom trhu.

#### Zjednodušený postup algoritmu[2]

1. Dva podobné, časovo nezávislé signály $x_{i..N}$ a $y_{j..M}$.
2. „Cenová“ matica (ang. cost matrix) $D_{N\times M}$.
3. Inicializácia cenovej matice, kde sa prvý riadok a prvý stĺpec nastaví na hodnoty $\infty$ a $D_{0,0}$ na hodnotu 0.
4. Traverzovanie matice a nastavovanie hodnôt (cien) buniek spôsobom vyjadreným vzorcom 2.2.
5. Zapamätanie si cesty, ktorou sa algoritmus dostal na jednotlivé bunky (zhoda, vkladanie, mazanie).
6. Po vyplnení matice sa spätne trasuje „vydláždená“ cesta od $D_{N,M}$ po $D_{0,0}$.

$$D_{i,j} = d\left(x_i, y_j\right) + \min \begin{cases} D_{i-1,j-1} & \text{...zhoda} \\ D_{i-1,j} & \text{...vkladanie} \\ D_{i,j-1} & \text{...mazanie} \end{cases} \tag{2.2}$$

Rov. 2.2, prevzaté z práce M. Müllera [15, s. 72] : Hodnota každej bunky $D_{i,j}$ sa vypočíta ako súčet vzdialenosti prvkov signálov na indexoch $i$ a $j$ a minimálnej hodnoty buniek predchádzajúcich ($D_{i-1,j-1}$, $D_{i-1,j}$, $D_{i,j-1}$), kde vzdialenosť prvkov signálov je daná absolútnou hodnotou ich rozdielu, $|x_i - y_j|$ (avšak môže byť použitá ľubovoľná funkcia vzdialenosti, napr. Euklidovská vzdialenosť, ako je uvedené v článku *Iterative Deepening Dynamic TimeWarping for Time Series* [6, kap. 2.2]), slová *zhoda*, *vkladanie* a *mazanie* a podrobnejšie fungovanie algoritmu sú bližšie vysvetlené v sekcii Vizualizácia a postup algoritmu.

Výsledkom je garantovaná cesta najmenšej ceny (celková cena zarovnania signálov je kumulatívny súčet hodnôt buniek, cez ktoré cesta vedie, t. j. hodnota bunky $D_{N,M}$) z cieľovej bunky matice $D_{N,M}$ do počiatku matice $D_{0,0}$. Táto cesta nám určuje na aké body signálu $y$ sa namapujú jednotlivé body signálu $x$. DTW teda určuje presné zarovnanie dvoch signálov a cenu tohoto zarovnania.

[2] Výklad vysvetľujúci postup DTW algoritmu – https://youtu.be/9GdbMc4CEhE

### Podmienky DTW cesty

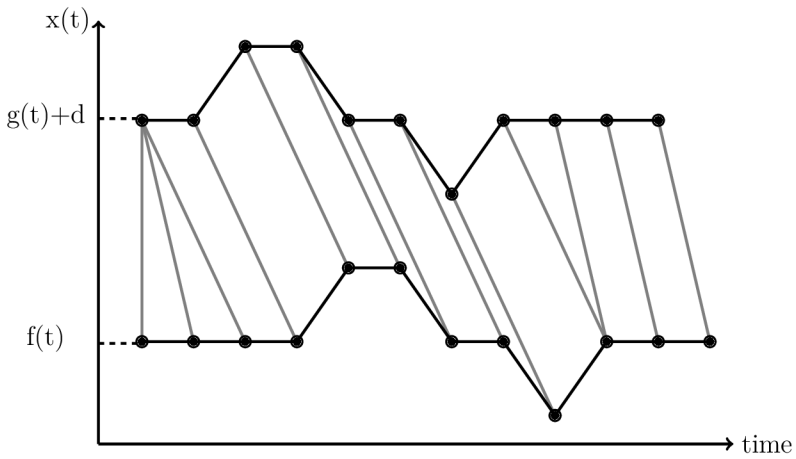
Optimálna cesta získaná DTW algoritmom (tzv. warping path) musí spĺňať tieto 3 podmienky:

- **hraničná podmienka** – cesta musí začínať v bunke $D_{0,0}$ a končiť v bunke $D_{N,M}$
- **podmienka monotónnosti** – pre indexy bunky v ceste nasledujúcej $i_1$, $j_1$ a indexy bunky v ceste predchádzajúcej $i_0$, $j_0$ platí: $(i_1 - i_0) \geq 0$ a $(j_1 - j_0) \geq 0$. Zjednodušene, cesta môže pri vizualizácii cenovej matice nasledovať iba na sever, východ alebo severovýchod, t. j. monotónne v čase
- **podmienka veľkosti kroku (kontinuity)** – pre indexy bunky v ceste nasledujúcej $i_1$, $j_1$ a indexy bunky v ceste predchádzajúcej $i_0$, $j_0$ platí: $(i_1 - i_0) \leq 1$ a $(j_1 - j_0) \leq 1$. Táto podmienka obmedzuje dĺžku kroku len na susedné bunky

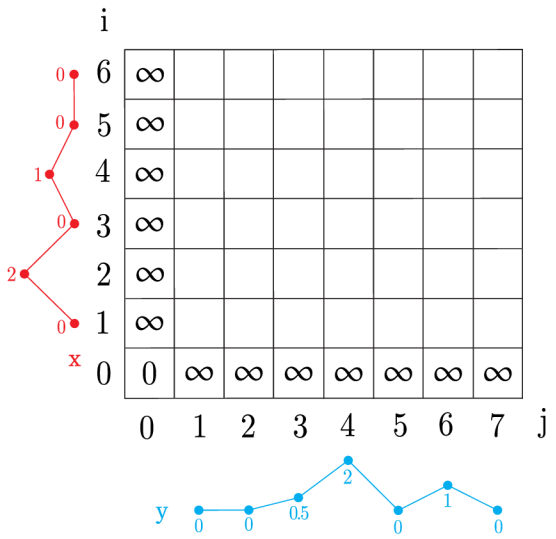
Viac o optimálnej ceste a jej podmienkach v *Iterative Deepening Dynamic TimeWarping for Time Series* [6, kap. 2.2] a v *Information Retrieval for Music and Motion* [15, kap. 4].

### Vizualizácia a postup algoritmu

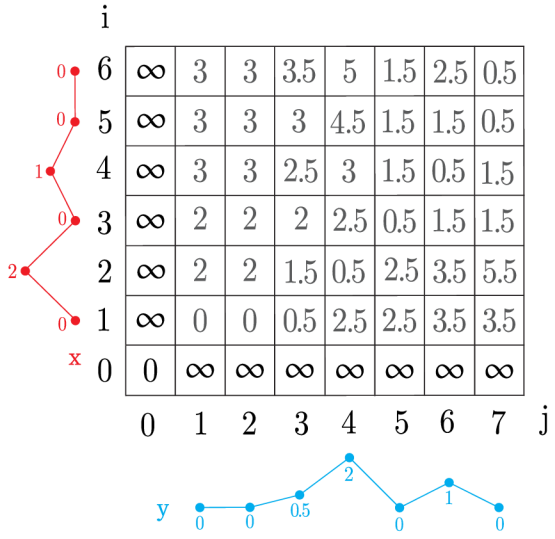
DTW algoritmus zabezpečí, že každý prvok signálu $x$ sa namapuje na nejaký prvok signálu $y$, túto skutočnosť zachytáva obrázok 2.4. Ilustráciu napĺňania matice cien a získanie výslednej cesty zobrazujú obrázky 2.5 a 2.6, indexovanie stĺpca a riadku bunky je $D_{riadok,stlpec}$.



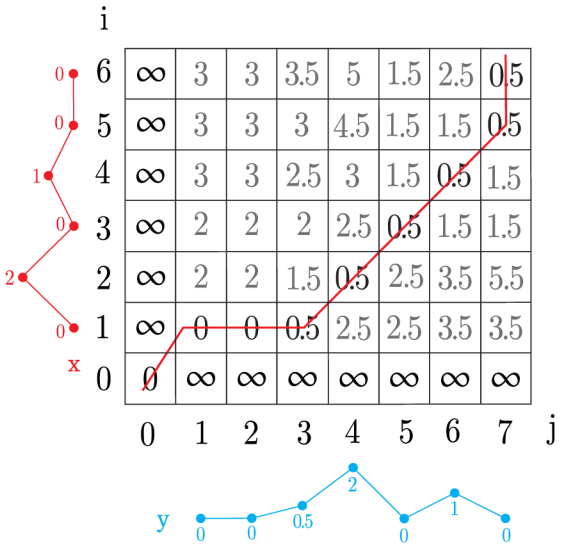

Obr. 2.4: Vizualizácia DTW, mapovanie prvkov jedného signálu $g(t)$ na prvky druhého sig. $f(t)$. Pre vhodné zobrazenie zarovnania je signál $g(t)$ posunutý o veľkosť $d$.




(a) Inicializovaná matica cien

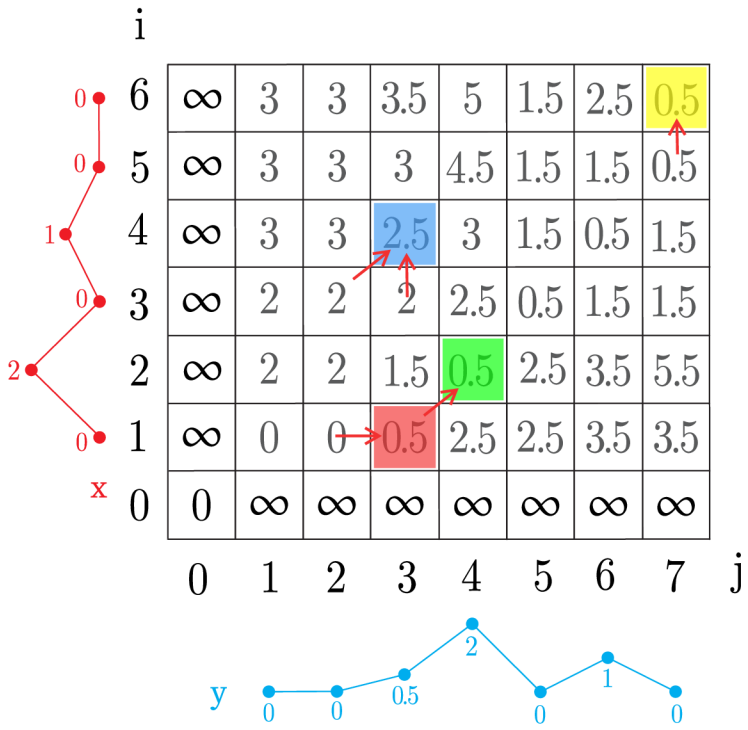



(b) Matica vyplnená cenami prechodov




(c) Vyznačená optimálna cesta

Obr. 2.5: Obrázok 2.5a inicializuje maticu cien – do stĺpca $D_{0,j}$ a riadku $D_{i,0}$ sa dosadí $\infty$ a bunka $D_{0,0}$ sa nastaví na 0. Obrázok 2.5b vizualizuje už vyplnenú maticu cenami jednotlivých prechodov. Vyznačenú optimálnu cestu zobrazuje obrázok 2.5c, tu taktiež vidno ako každá bunka prislúcha k zarovnaniu určitých bodov signálov $x$ a $y$, napr. hodnota bunky $D_{1,3}$ (0.5) symbolizuje zarovnanie (konkrétne cenu zarovnania) prvku $x[1] = 0.5$ s prvkom $y[3] = 0$ (pri indexovaní od 1), $D_{1,3} = |0.5 - 0| + min(0, \infty, \infty) = 0.5$.



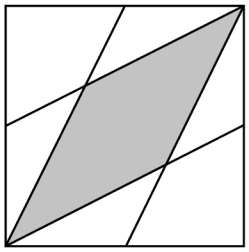

Obr. 2.6: Na pochopenie slov zhoda, mazanie a vkladanie/substitúcia zo vzorca 2.2, treba objasniť ich spojenie s cestou DTW algoritmu a to skutočnosť, že výber minimálnej hodnoty predchádzajúcich buniek súvisí okrem výpočtu ceny aj s ukladaním cesty a to tak, že zhoda↗, mazanie→ a vkladanie/substitúcia↑, symbolizujú smer kroku na aktuálnu bunku. Okrem smeru predstavujú aj operáciu, ktorá sa musí vykonať nad signálom $y$ aby vznikol signál $x$ – napr. minimum pri výpočte bunky $D_{1,3}$ pochádza z bunky $D_{1,2}$, teda nastáva mazanie, takže zo signálu $y$ treba vymazať prvok na indexe 3 aby sa priblížil k signálu $x$, bunka $D_{2,4}$ predstavuje zhodu (žiadna operácia) a bunka $D_{6,7}$ vkladanie/substitúciu (substitúcia predstavuje nahradenie prvku signálu $y$ prvkom signálu $x$). Po prevedení všetkých operácií na optimálnej ceste nad signálom $y$ vznikne signál $x$. Problém nastáva pri bunke $D_{4,3}$, kde vzorec vedie k výpočtu dvoch rovnakých minimálnych hodnôt ($D_{3,2} = D_{3,3} = 2$), rieši sa rôzne, napríklad lexikograficky, ako je uvedené v práci *Information retrieval for music and motion* [15, s. 73].

Ukladanie smeru, ktorým sa získala daná bunka je dôležité pri spätnom trasovaní optimálnej cesty (programovo to môže znamenať vytvorenie osobitnej matice na ukladanie indexov predchádzajúcej bunky, kde môže pomôcť práve rozdelenie na vkladanie, mazanie alebo zhodu).

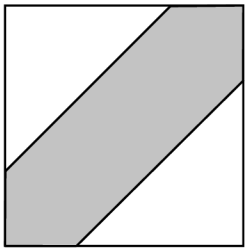
### Optimalizácia DTW algoritmu

Nutnosť kalkulovania všetkých hodnôt matice cien môže byť najmä pri dlhších signáloch náročné na priestor a procesorový čas – klasický DTW algoritmus má kvadratickú časovú a priestorovú zložitosť $O(n^2)$. Preto prichádzajú do úvahy určité optimalizácie (obmedzenia), napríklad:

- **Úprava podmienky veľkosti kroku** – umožní preskočiť bunku v ceste (zamedzí „zauzleniu“ cesty).
- **Váhy kroku** – možno určiť rôzne váhy pre rôzne smery cesty (demonštrované autormi R. Saabni a J. El-Sana v práci [21, s. 8]).
- **Kompresia vstupu** – redukcia veľkosti vstupných signálov (za určitej straty presnosti, napr. downsampling 2.5.1) prirodzene zefektívňuje algoritmus.
- **Globálne obmedzenia cesty** – obmedzí počet buniek, ktoré treba vyhodnotiť pri vypĺňovaní matice cien, obrázky 2.7a a 2.7b ilustrujú známe typy týchto obmedzení.



(a) Itakura parallelogram



(b) Sakoe-Chiba band

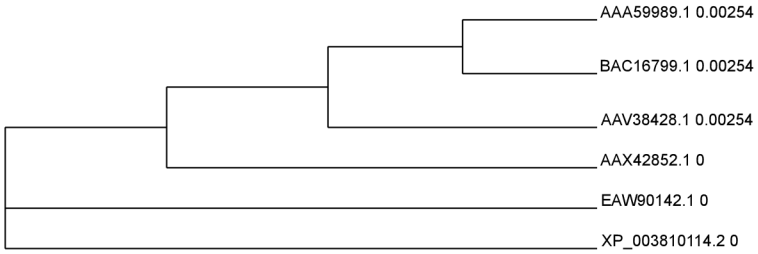
Viac o optimalizácii a obmedzeniach algoritmu DTW píšu M. Müller v [15, s. 74 – 78] a autori práce *Iterative Deepening Dynamic Time Warping for Time Series* [6, kap. 3]. K dispozícii je mnoho optimalizovaných variant DTW algoritmu, menovite: IDDTW, PDTW, ShapeDTW, LSDTW atď. Niektoré z uvedených variant porovnáva práca autorov A. Lahreche a B. Boucheham [13]. Pri implementácii nástroja bol použitý variant **FastDTW**, ktorý sa podľa autorov *Toward Accurate Dynamic Time Warping in Linear Time and Space* [22] približuje k lineárnej časovej a priestorovej zložitosti $O(n)$ pri chybovosti v jednotkách percent.

### FastDTW

Algoritmus dômyselne využíva obmedzenie cesty rekurzívnym „multi-levelovým“ spôsobom tak, že najskôr určí DTW cestu na veľmi obmedzenom rozlíšení signálov a jej okolia a postupne zvyšuje rozlíšenie a okolie cesty až dosiahne najvyššieho rozlíšenia. Nájdená cesta nie je vždy optimálna, ale vo väčšine prípadov je veľmi podobná optimálnej ceste klasického DTW algoritmu, avšak za použitia zlomku priestorových a časových prostriedkov. Bližšie informácie a porovnanie s klasickým prístupom popisujú autori Salvador, S. a Chan, P. v práci venovanej FastDTW algoritmu [22].

### 2.1.3 MSA

Multiple Sequence Alignment (MSA)[3] je zarovnanie biologických sekvencií (DNA, RNA) podobných dĺžok. Z výstupu je možné vyčítať homológiu a bližšie analyzovať dedičné a evolučné vzťahy medzi sekvenciami. Výstupom je napríklad aj fylogenetický strom, ktorý vizualizuje príbuznosť sekvencií, pozri obrázok 2.8.




Obr. 2.8: Vizualizácia fylogenetického stromu vyrobeného nástrojom Clustal Omega, napravo sú zobrazené názvy jednotlivých sekvencií.

Existuje mnoho rôznych metód MSA (Clustal Omega, MAFFT, MUSCLE, T-Coffee) špecializovaných na rôzne veľkosti sekvencií a implementovaných rozmanitým spektrom algoritmov (Dynamic programming, FFT, ...). Viac o metódach a aplikáciách MSA sa čitateľ dočíta v práci *Multiple sequence alignment modeling: methods and applications* [5].

[3]MSA –https://www.ebi.ac.uk/Tools/msa/

### 2.1.4 MFCC

Mel-frekvenčné cepstrálne koeficienty (MFCC) zachytávajú frekvenčné spektrum zvuku za pomoci viacerých algoritmov a úprav tak, aby čo najlepšie simulovali fungovanie ľudského ucha a následné spracovanie signálu mozgom. Tento spôsob umožňuje efektívne zachytiť dôležité aspekty zvuku na menšom počte hodnôt (koeficientoch), pričom (v ideálnom prípade) nesú rovnaké množstvo informácie ako pôvodný signál, čo napomáha v rôznych aplikáciách rozpoznania reči. Jeden krok extrahovania MFC koeficientov zahŕňa prevod frekvenčnej stupnice na tzv. Melovu stupnicu, ktorá lepšie vystihuje ako je človek schopný rozlíšiť rozdiel v rôznych frekvenciách (ľudské ucho má na nižších frekvenciách väčšie rozlíšenie ako na vyšších). Vzorec 2.3 prevodu hertzov na mely:

$$f_{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.3}$$

Rov. 2.3: Melova frekvenčná stupnica nadobúda logaritmickú podobu (pozri obrázok 2.9), vzorec prevzatý z práce autorov K. Sreenivasa Rao a Manjunath K.E. [20].

Pôvodne boli vytvorené na automatické rozpoznanie hovoriaceho, dnes sa okrem iného používajú aj získavanie informácií z hudby (ang. music information retrieval), ako uvádza autor práce M. Müller *Information retrieval for music and motion* [15, s. 65].

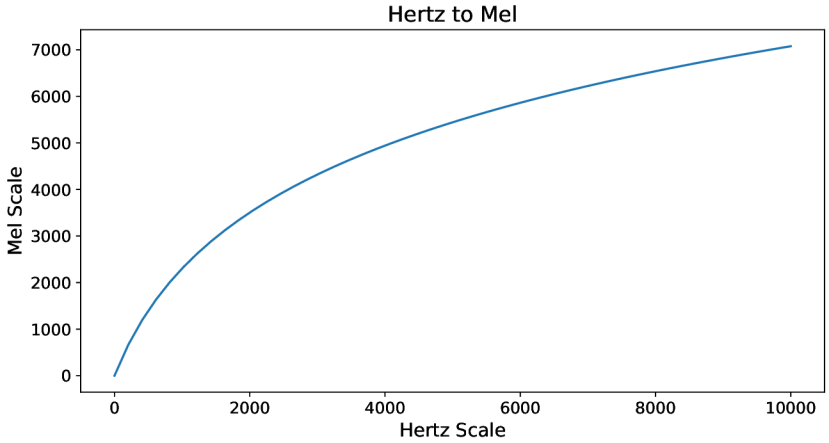
#### Stručný postup výpočtu

Kroky vedúce k extrakcii MFC koeficientov sú nasledovné:

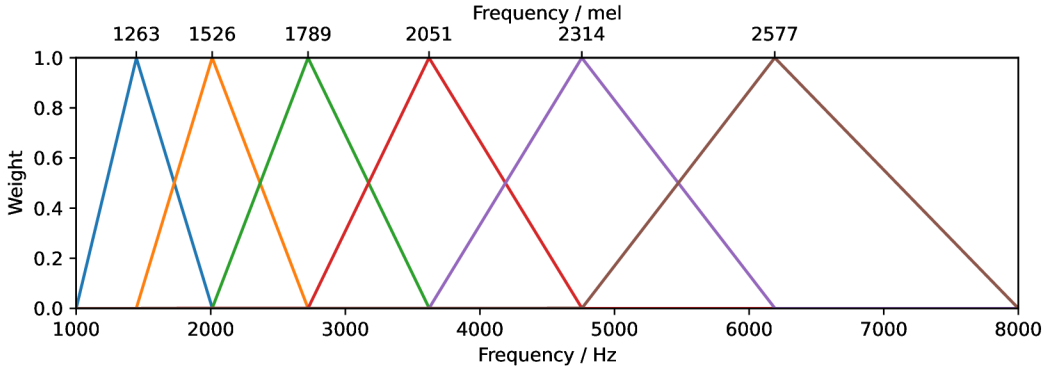
1. **Pre-emfáza** – aplikovanie horno-priepustného filtru na zdôraznenie vysokých frekvencií (podobne sa správa ľudské ucho)[4].
2. **Rámcovanie** – signál je rozdelený na prekrývajúce sa rámce (napr. dĺžka 20ms, prekryv 10ms) a na každý rámec je aplikovaná okienková funkcia (ang. window function), napr. Hammingove okno (ang. Hamming window) – týmto sa vyhladí signál pre ďalší krok.
3. **DFT** – na každé okno sa aplikuje DFT (FFT), tým vznikne frekvenčné spektrum a vypočíta sa energia umocnením.
4. **Aplikovanie melovej stupnice** – prevod DFT okien na melovu stupnicu sa zabezpečí nerovnomerným aplikovaním trojuholníkových filtrov (okien) z tzv. banky filtrov (ilustruje obrázok 2.10), ktorá je zostavená tak, aby simulovala melovu stupnicu (t. j. trojuholníky sú vo vyšších frekvenciách ďalej od seba ako v nižších, práve kvôli nelineárnemu vnímaniu rozdielu frekvencií ľudským uchom).
5. **DCT** – aplikovanie diskrétnej kosínusovej transformácie (DCT, ang. discrete cosine transform) na logaritmus koeficientov získaných nanesením trojuholníkových okien – výstup sú **Mel-frekvenčné cesptrálne koeficienty**.

Bližší popis postupu kalkulácie MFC koeficientov poskytujú autori práce *An Approach to Extract Feature using MFCC* [19] alebo autori K. Sreenivasa Rao a Manjunath K.E. v práci [20].

[4] Výklad vysvetľujúci MFCC postup – https://youtu.be/PPmNYwVbcts




Obr. 2.9: Graf prevodu Hz na mely, snaží sa dosiahnuť podobnosť vnímania ľudského ucha – človek počuje frekvencie logaritmicky, rovnako sú aj v reči podstatné informácie uložené logaritmicky.




Obr. 2.10: Banka filtrov – trojuholníkových okien, horná osa x zobrazuje frekvencie na melovej stupnici, dolná osa x na hertzovej, povšimnutia vhodný fakt je, že na melovej osi sú vrcholy okien rozmiestnené rovnomerne, avšak na hertzovej osi nerovnomerne.

## 2.2 Waveform

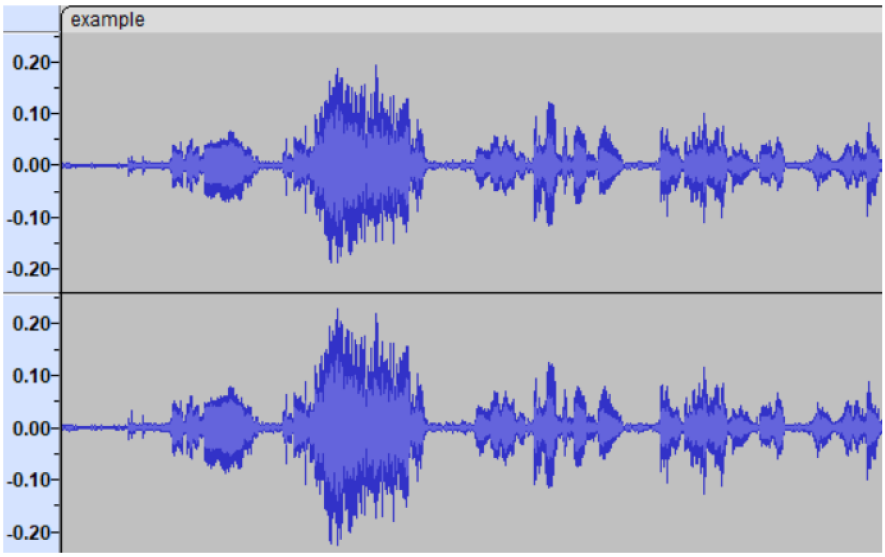
Waveform je graf, ktorý predstavuje tvar a priebeh signálu v čase. Zachytáva amplitúdu prvku (y-ová osa) signálu v čase (x-ová osa). Najčastejšie reprezentuje zmenu napätia (či inej fyzikálnej veličiny) na merači (osciloskope). Zvuk je v digitálnej podobe reprezentovaný postupnosťou vzoriek, ktorých hodnoty reprezentujú zmeny tlaku vzduchu v okolí mikrofónu, ktoré sú vyvolané zvukovými vlnami – mikrofón ich zachytí ako zmeny napätia a tie sú v istom časovom intervale odčítané a uložené ako jednotlivé vzorky (to robí A/D prevodník pomocou PCM[5]). Časový interval, v ktorom sa hodnoty odčítavajú sa nazýva **vzorkovacia frekvencia**.

### Vzorkovacia frekvencia audio-vizuálnych súborov

Na správnu digitálnu reprezentáciu spojitého signálu sa musí vzorkovať dvojnásobkom vstupnej frekvencie[6]. Keďže ľudské ucho dokáže rozlíšiť frekvencie medzi 20Hz – 20kHz, sa audio-súbory najčastejšie vzorkujú vzorkovacou frekvenciou 44.1kHz.

### WAV audio-súbor

WAV (.wav) je formát audio-súborov, ktorý zachytáva navzorkovaný audio-signál. Jeho hlavička, po ktorej nasledujú samotné dáta, obsahuje špecifikáciu dát ako veľkosť súboru, vzorkovaciu frekvenciu, počet kanálov, počet bitov na vzorku a iné. Oproti MP3 formátu (ktorý má na úkor kvality menšiu veľkosť) reprezentuje WAV formát nekomprimovaný audio-signál.




Obr. 2.11: Obrázok waveformu so stereo stopou v programe Audacity, stereo – dva kanály

Informácie v tejto kapitole boli čerpané z knihy *Electronics* [8, s. 47 – 48] a základy vedomostí z knihy autorov A. Bruce Carlson a Paul B. Crilly [4, s. 3] a knihy *Signals & Systems* [17, kap. 1.1].

[5]PCM – Pulzná kódová modulácia – https://en.wikipedia.org/wiki/Pulse-code_modulation
[6]Nyquist–Shannon vzorkovací teorém – https://cecm.indiana.edu/361/digitalaudio1.html

## 2.3 Formáty titulkov

Existuje viacero formátov titulkov. Niektoré poskytujú okrem základných funkcionalít (očíslovanie, čas konca a čas začiatku, text) aj štýlovanie (farby, fonty, pozadie), formátovanie a úpravu pozície na obrazovke.

### 2.3.1 SRT

Najznámejší a najpoužívanejší formát je SRT – SubRip file format. Neposkytuje žiadne štýlovanie a úpravu pozície, avšak rôzne prehrávače multimédií (napr. VLC) umožňujú jednoduché štýlovanie/formátovanie .srt súborov založené na HTML tagoch (tučné, kurzíva, podčiarknuť, farba). Je podporovaný väčšinou prehrávačov a viacerými sociálnymi platformami (Facebook, YouTube, ...). V tomto formáte majú titulky jednoduchú štruktúru, ilustrovanú výpisom 2.1.

```
1
00:00:00,160 --> 00:00:06,200
Uh, ja som bol v~tom zahranici,

2
00:00:06,600 --> 00:00:07,100
Ano.
```

Výpis 2.1: Ukážka SRT súboru, dva prehovory.

Jeden prehovor sa skladá z:

1. Čísla prehovoru
2. Času prehovoru (začiatok → koniec), vo formáte hh:mm:ss,ms
3. Text prehovoru

Medzi jednotlivými prehovormi je prázdny riadok. Viac k SRT súborom píšu autori Dick C.A. Bulterman, Jack Jansen a ostatní v [3, kap. 3.1.3]. Pre nástroj sa použil práve tento formát kvôli jeho rozšírenosti a jednoduchosti.

### 2.3.2 WebVTT

Web Video Text Tracks (WebVTT[7], .vtt) je relatívne nový formát založený na SRT formáte a je určený na špecifikáciu titulkov pre elementy <audio> a <video> elementom <track> v HTML5. Kompatibilný s HTML5 stránkami a vhodný na titulkovanie videí na webe.

### 2.3.3 SAMI

Formát Synchronized Accessible Media Interchange (SAMI[8], .smi) je jazyk, od firmy Microsoft určený na tvorbu titulkov na PC. Podporuje tvorbu titulkov určených pre nepočujúcich, ktoré okrem textu prehovoru zobrazujú do textu aj napr. zvuky okolia, emóciu, rozlíšenie hovoriaceho atď. Kompatibilné napríklad s Windows Media Player a VLC. Jazyk je podobný HTML a CSS. Využíva pestrú podmnožinu HTML tagov a poskytuje rôzne štýlovanie a formátovanie. Príklad .smi súboru – výpis 2.2.

[7] Viac o WebVTT na: https://www.w3.org/TR/webvtt1/
[8] Viac o SAMI na: https://shorturl.at/juAKT

```
<SAMI>
    <HEAD>
       <STYLE TYPE="text/css">
       <!--
          P {
           font-size: 24pt;
           text-align: center;
           color: white;
          }
          #Source {color: red; font-family: arial; font-size: 10pt;}
          .ENUSCC {Name: 'English Subtitles'; Lang: en-US; SAMIType: CC;}
          .CSCSCC {Name: 'Czech Subtitles'; Lang: cs-CS; SAMIType: CC;}
       -->
       </STYLE>
    </HEAD>
    <BODY>
        <SYNC Start=6300>
            <P Class = ENUSCC ID = Source>Narrator
            <P Class=ENUSCC>Uh, I~was abroad,
            <P Class = CSCSCC ID = Source>Rozpravec
            <P Class=CSCSCC>Uh, ja jsem byl v~zhranici,
    </BODY>
</SAMI>
```

Výpis 2.2: Ukážka SAMI súboru, kde vidieť podobnosť s HTML a CSS (prvok P), ďalej je možné vidieť rozlíšenie jazykov a odlišné formátovanie pri označení rozprávača a textu prehovoru.

### 2.3.4 MicroDVD

Formát MicroDVD[9] (.sub) je určený pre aplikáciu MicroDVD Player vyvíjanú firmou Tiamat Software medzi rokmi 2000 až 2001, na prehrávanie DVD. Jeho zaujímavosťou je, že čas zobrazenia titulkov je určený číslami snímok. Príklad .sub súboru – výpis 2.3.

```
{0}{250}Uh, bol som v~zahranici,
{251}{500}Ano.
```

Výpis 2.3: Ukážka MicroDVD súboru.

Existujú mnohé ďalšie formáty titulkov určené napríklad na živé televízne vysielanie, prehrávanie DVD/Blu-ray atď. Od vyššie uvedených formátov sa iné líšia typom zápisu (napr. XML), presnosťou času zobrazenia alebo typom zobrazenia (open, closed atď.).

**Rozdelenie titulkov podľa typu zobrazenia:**

- **open** – tiež hard subtitles, sú také titulky, ktoré sú napálené do videa tak, že sa stanú súčasťou každého snímku, nedajú sa vypnúť alebo zmeniť.
- **closed** – všetky vyššie spomenuté formáty patria do tejto kategórie, titulky sú v oddelenom súbore a potrebujú podporu prehrávača, ktorý ich zobrazí cez video, dajú sa vypnúť a zapnúť.

Viac o rôznych formátoch titulkov a všeobecne o praktikách titulkovania píšu autori Jorge D. Cintas a Aline Remael v knihe *Subtitling: Concepts and Practices* [7].

[9] Viac k MicroDVD na: https://en.wikipedia.org/wiki/MicroDVD

## 2.4 Konverzia a kompresia audiovizuálnych súborov

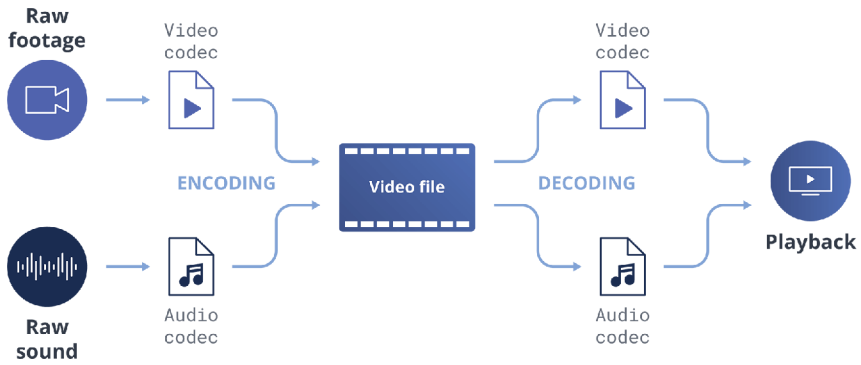
Konverzia a kompresia videa a zvuku je bežná súčasť aplikácií pracujúcich s multimediálnymi súbormi. Konverzia videa znamená zmenu formátu kódovania videa, (transkódovanie – za realizáciu tohto procesu sú zodpovedné kodeky, pozri sekciu 2.4.1) čo zahŕňa zmenu kvality, fps (snímky za sekundu), rozlíšenia atď. Súčasťou tohoto procesu môže byť aj *kompresia*. Kompresiu zvuku možno dosiahnuť napríklad „podvzorkovaním“ (downsampling, pozri sekciu 2.5.1). Jedným z najznámejších typov kompresie zvuku je kompresia na formát MP3[10]. Tento formát používa algoritmy na odstránenie nepodstatných dát zvukového súboru (lossy compression, pozri sekciu 2.4.1). Jeden z nástrojov na úpravu formátu audiovizuálnych súborov je FFmpeg (pozri sekciu 4.3).

### 2.4.1 Kodek a encoding

Názov kodek[11] (ang. codec) pochádza zo spojenia *coder-decoder*, čo naznačuje funkciu kodeku. Kodek slúži na zakódovanie videa alebo zvuku do menšej, digitálnej formy – takže slúži na kompresiu súboru a zároveň aj na dekódovanie súboru pre prehranie na médii. Kodek je teda nástroj na kompresiu a dekompresiu videa/audia, zatiaľ čo encoding (a decoding) sú procesy, ktoré kodeky využívajú, čo opisuje obrázok 2.12. Jeden z najznámejších a najpoužívanejších kodekov je MPEG (viac o MPEG a algoritme, ktorý používa píše autor Pan D. Y. v [18]), ďalej napríklad H.264, HEVC, AAC, MP3 atď.

#### Stratová a bezstratová kompresia

Stratová a bezstratová kompresia (ang. lossy and lossless compression) sú dva hlavné typy kompresie. Stratová kompresia niektoré časti súboru vymaže permanentne, napríklad pre audio-súbor sú to zvuky, ktoré človek nepočuje, zatiaľ čo bezstratová kompresia zmenší súbor bez straty informácie. Stratová kompresia je vhodná na audio/video súbory, bezstratová kompresia na súbory textové charakteru.




Obr. 2.12: Vizualizácia pracovania kodeku a jeho signifikancie pre multimediálne súbory. Dostupné z: www.epiphan.com/blog/h264-vs-h265/.

[10] MP3 formát – https://en.wikipedia.org/wiki/MP3

[11] What is Video Codec and the Difference with File Format and Extension – dostupné z: https://www.any-videoconverter.com/mac-tutorial/videocodec.html

## 2.5 Resampling

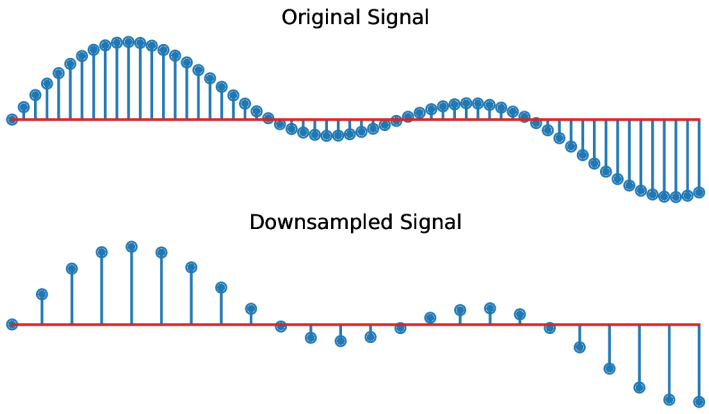
Prevzorkovanie (ang. resampling) je zmena vzorkovacej frekvencie signálu, čo znamená zmenu počtu vzoriek za sekundu, pričom sa zachová pôvodná dĺžka signálu. Táto technika sa používa pri spracovaní audio-signálu alebo obrazu. Rôzne prehrávacie platformy používajú rôzne vzorkovacie frekvencie zvuku napr. Na CD je zvuk vzorkovaný vzorkovacou frekvenciou 44.1 kHz, zatiaľ čo na DVD alebo Blu-ray dosahuje 48 kHz. Zníženie vzorkovacej frekvencie signálu možno dosiahnuť napr. decimáciou, zatiaľ čo zvýšenie vzorkovacej frekvencie sa nazýva interpolácia. Zmena vzorkovacej frekvencie môže spôsobiť artefakty (tzv. aliasing) vo výslednom signáli, preto je nutné používať vhodné „prevzorkovávacie“ techniky pre dané účely. Viac o technikách prevzorkovania píšu autori Crochiere, R.E. a Rabiner, L.R. v práci [9, kap. 2.3] a autori knihy *Discrete-time signal processing* [16, kap. 4.6].

### 2.5.1 Downsampling

Zníženie vzorkovacej frekvencie (ang. downsampling) prináša kompresiu signálu/súboru, čím sa šetria pamäťové zdroje a urýchli sa práca s týmto signálom.

#### Techniky

- **decimácia** – najjednoduchšia technika, zahŕňa ponechanie len každého n-tého prvku. Číslo n sa nazýva aj decimačný koeficient, napr. funkcia `dec(sig, 4)` by zanechala len každý 4-tý prvok, t. j. výsledný signál by bol 4-násobne menší. Demonštrované obrázkom 2.13.
- **priemer** – technika, ktorá spriemeruje susedné prvky a tým dosiahne zníženie vzorkovacej frekvencie. Spôsobuje menej artefaktov ako decimácia. Rôzne techniky porovnáva autor Becek K. v článku [1].
- **low-pass filter** – aplikuje sa tzv. dolný priepust (low-pass), t. j. filter, ktorý vyhladzuje vysoké frekvencie a teda prepúšťa nízke a potom sa na daný signál použije decimácia. Tento prístup zníži množstvo artefaktov – aliasingu, preto pôsobí ako anti-aliasingový filter (viac v [9, s. 35]), najmä pri vysokých frekvenciách.



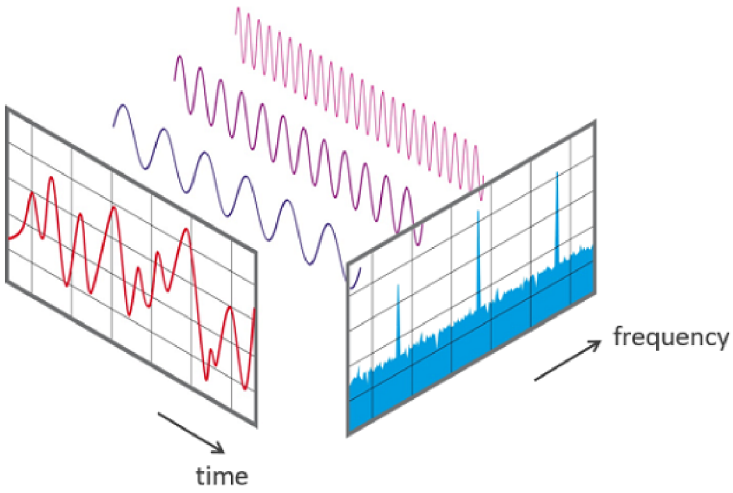

Obr. 2.13: Vizualizácia prevzorkovania, konkrétne downsamplingu (s využitím decimácie s koeficientom 2.5), horný graf vizualizuje vstupný signál a dolný graf vizualizuje prevzorkovaný signál.

## 2.6 Fourierová transformácia

Fourierová transformácia (FT, ang. Fourier transform) je dômyselná funkcia transformujúca signál v časovej doméne na reprezentáciu do frekvenčnej domény a naopak (Inverzná FT, ang. Inverse Fourier transform). Frekvenčná doména signálu reprezentuje všetky frekvencie (všetky sínusoidy), z ktorých sa daný signál skladá, čo demonštruje obrázok 2.14. Ak ide o diskrétny signál, tak frekvenčná doména reprezentuje každý prvok ako komplexné číslo – fourierov koeficient, pričom jeho veľkosť (modul) predstavuje amplitúdu danej sínusoidy a argument (fáza) daného komplexného čísla jej fázu. Inverzná operácia k fourierovej transformácii (IFT) zostaví pôvodný signál v časovej doméne sčítaním všetkých sínusoíd, ktoré sú reprezentované fourievými koeficientami (pri spojitom signály sú pojmy upravené, sčítanie→integrovanie a koeficienty→spektrálna funkcia).

### 2.6.1 DFT

DFT (Discrete fourier transform) je verzia fourierovej transformácie slúžiaca na transformáciu signálov mapovaných v diskrétnom čase. Problém tohoto algoritmu je jeho rýchlosť, ktorá hlavne pri väčšej dĺžke vstupného signálu nie je ideálna. Časová zložitosť algoritmu je kvadratická $O(n^2)$, čo sa vo svete algoritmov považuje za relatívne pomalý algoritmus.



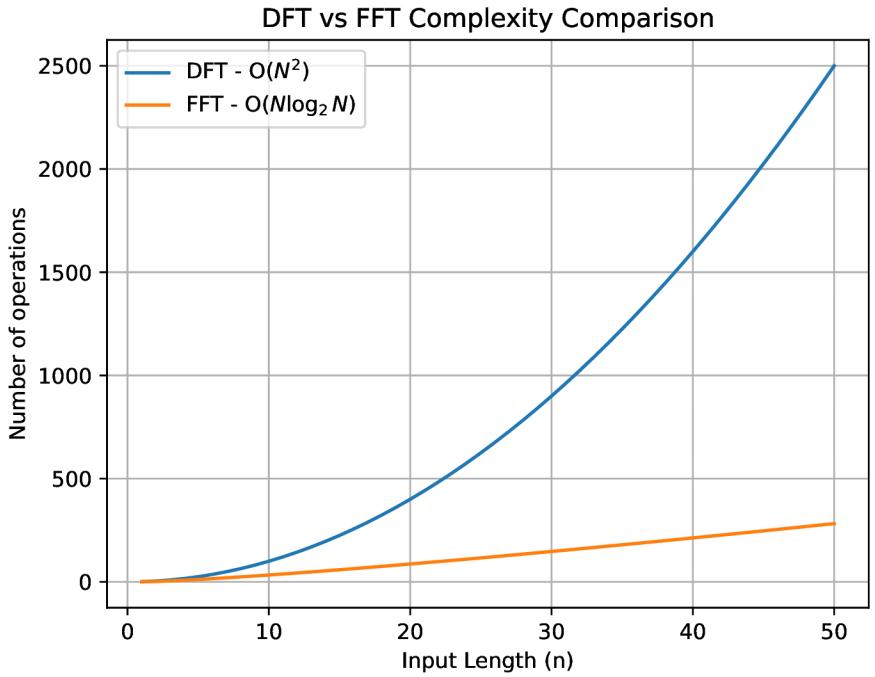

Obr. 2.14: Vizualizácia FT, obrázok sa snaží zobraziť ako k jednotlivým frekvenčným komponentom frekvenčnej domény prislúcha sínusoida. Obrázok dostupný z: www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft.

Vedomosti v tejto kapitole boli čerpané z diel *Discrete-time signal processing* [16, kap. 9] a *Introduction to Audio Analysis* [11, kap. 3.1].

### 2.6.2 FFT

Tento výkonnostný problém adresuje **FFT**[12] (Fast fourier transform). V podstate sa jedná o efektívnejšiu verziu DFT, vďaka ktorej sa rýchlosť algoritmu niekoľkonásobne zvýši aj pri kratších signáloch. Časová zložitosť FFT algoritmu je logaritmická $O(n \log n)$, obrázok 2.15. Dnes sa v praxi na výpočet fourierovej transformácie používa výhradne algoritmus FFT.

Hlavným rozdielom FFT je rekurzívny prístup riešenia problému – prvky vstupného signálu sa rozdelia na menšie časti, konkrétne podľa parity indexu, čiže vstupné pole prvkov sa rozdelí na dve polia o veľkosti N/2 (kde N je veľkosť vstupného signálu), tieto polia sa rozdelia na ďalšie polia (znova podľa rovnakých kritérií) atď., až na polia o veľkosti 1. Nakoniec sa rekurzívne spočíta DFT každého poľa až ku konečnému výsledku fourierovej transformácie vstupného signálu (využíva periodicitu a symetriu komplexnej exponenciály, viac v *Discrete-time signal processing* [16, kap. 9]).




Obr. 2.15: Obrázok vyjadruje porovnanie výpočtovej zložitosti klasického DFT algoritmu a FFT algoritmu na výpočet fourierovej transformácie.

[12] Videovýklad o algoritme FFT – https://youtu.be/h7ap07q16V0

# Kapitola 3

# Návrh riešenia

V tejto kapitole sú stanovené ciele a popísané všetky úlohy, ktoré je potrebné vykonať na ich dosiahnutie. Je tu opísané ako použiť algoritmy a prostriedky popísané v kapitole 2 na splnenie zadaného problému priblíženého v úvode práce.

## 3.1 Stanovenie cieľov

Na stanovenie cieľov si treba uvedomiť potreby užívateľa. Užívateľ chce predovšetkým pohodlne zadať vstup (video(á) a titulky) a dostať výstup (prečasované titulky) v rozumnom čase.

### Ciele

- **Pohodlnosť** – vyhnúť sa spúšťaniu z príkazovej riadky a vytvoriť užívateľské prostredie, ktoré je intuitívne a spĺňa funkčné požiadavky.
- **Čas** – implementácia algoritmov tak, aby pri čo najmenšej strate presnosti zvládli vykonať úlohu (prečasovanie) za rozumný časový úsek.
- **Kompaktnosť** – snaha o vytvorenie kompaktného riešenia.

Z cieľov je možné extrahovať úlohy: vytvorenie užívateľského prostredia (frontend) a vytvorenie algoritmu riešiaceho zadaný problém (backend).

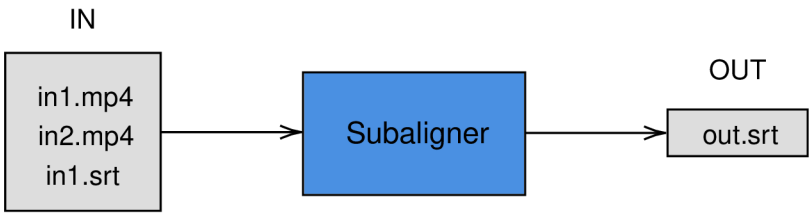
### Vstupy a výstupy

- **Vstupy** – video s titulkami, ktoré súhlasia s prehovormi vo videu a video, na ktoré treba titulky zarovnať (ďalej už len **in1.mp4**, **in1.srt** a **in2.mp4**).
- **Očakávaný výstup** – titulky zarovnané na prehovory druhého videa (ďalej už len **out.srt**).

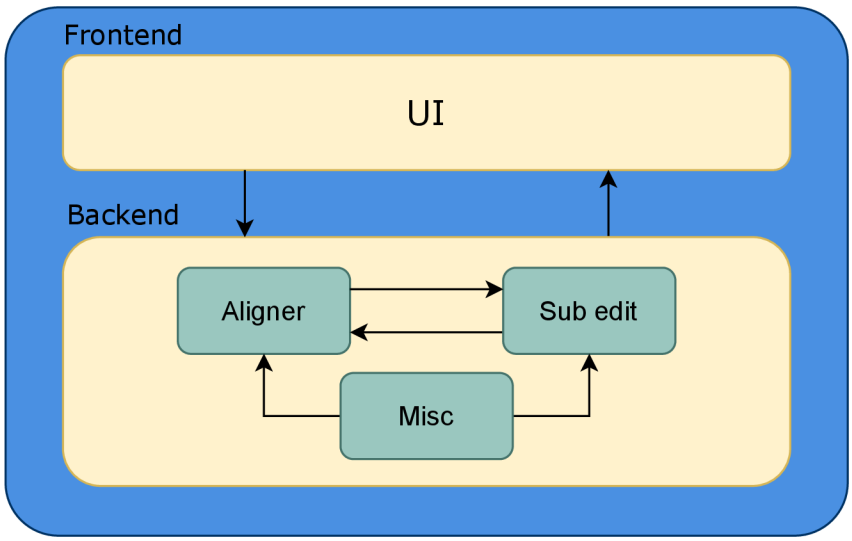
Očakávaný formát videosúborov je MP4 (.mp4) a formát titulkov SRT (.srt).

## 3.2 Štruktúra a fungovanie aplikácie

Návrh aplikácie sa delí na blok `Frontend`, `Backend` a `Misc`. Aplikácia očakáva tri vstupy (**in1.mp4**, **in2.pm4** a **in1.srt**) a výsledkom je jeden výstup (**out.srt**). Základnú štruktúru aplikácie demonštruje obrázok 3.1




(a) Najabstraktenjší pohľad – vstupy a výstupy




(b) Hlavná štruktúra aplikácie

Obr. 3.1: Obrázok 3.1a reprezentuje najabstraktenjší pohľad na štruktúru aplikácie. Zobrazuje vstup (IN) a výstup (OUT). Blokové schéma na obrázku 3.1b približujúce hlavnú štruktúru aplikácie. Súčasti `Aligner`, `SubEdit` a `Misc` sú vysvetlené v sekcii 3.4.

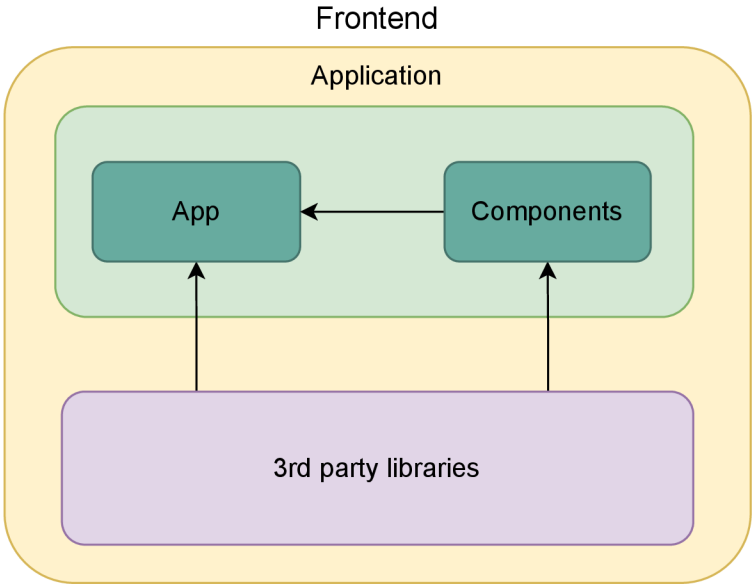
### Vysvetlenie dôležitých pojmov

Pojem **vymazané** titulky/video predstavuje časť titulkov/videa (**in1.srt**/**in1.mp4**), ktorá sa musela **vymazať** aby sedeli na video **in2.mp4**. Je nutné podotknúť, že fyzicky sa vymažú len titulky, video/signál sa len zarovná pomocou algoritmu. Pojem **zarovnanie** titulkov predstavuje prečasovanie titulkov či synchronizáciu titulkov s videom.

## 3.3 Frontend – návrh

V tejto sekcii je popísaný návrh riešenia užívateľského prostredia (UI). Návrh je vizualizovaný obrázkom 3.2. Pre vstupy treba vytvoriť grafické prostredie a nastaviť ich interakciu s užívateľom a medzi sebou. Čo musí spĺňať UI aplikácie:

- **Nahranie súborov** – užívateľ musí byť schopný nahrať potrebné súbory
- **Zobrazenie súborov** – vložené súbory je nutné zobraziť v rámci GUI
- **Interakcia s widgetmi** – užívateľ by mal byť schopný interagovať so vstupmi a výstupmi




Obr. 3.2: Blokové schéma abstraktného zobrazenia frontendu aplikácie. Znázorňuje prepojenie s knižnicami a štruktúru frontendu. Vysvetlenie jednotlivých modulov `App`, `Components` a `3rd party libraries` je v sekciách nižšie.

### Komponenty (Components)

Tento modul reprezentuje (uchováva) všetky komponenty (widgety) GUI aplikácie. A teda:

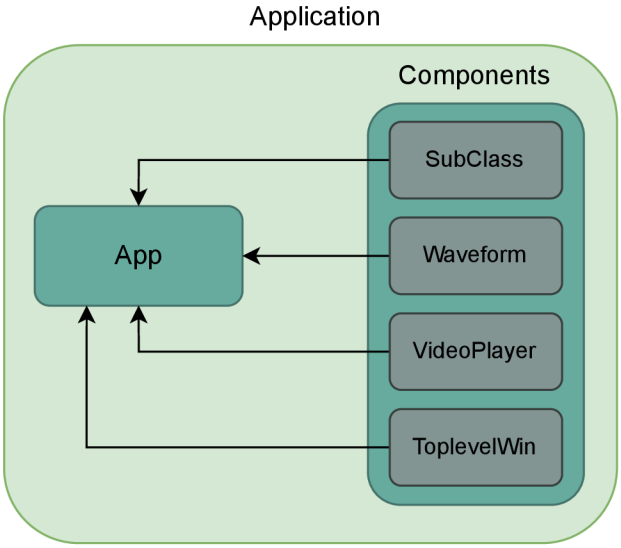
- widget na zobrazenie videa.
- widget na zobrazenie audio-signálu v podobe waveformu (pozri sekciu 2.2).
- widget na zobrazenie titulkov.
- komponenta pre vizualizáciu načítavania.

### Knižnice a nástroje tretích strán

Na zabezpečenie potrieb užívateľa je potrebné použiť vhodné prostriedky (knižnice) na tvorbu GUI. Zvolená knižnica/knižnice by mala/mali poskytovať:

- možnosti vykreslovania kriviek (pre waveform) a 2D tvarov (pre vyznačenie prehovorov),
- prehrávania a manipulácie (pause/play, posúvanie posuvníku atď.) s videom,
- možnosť zobraziť text a manipulovať (posúvať) s ním (titulky),
- možnosť vybrania a nahrania súborov (t. j. prístup k súborovému systému a vybratie súborov pomocou tlačidiel)
- interakciu medzi widgetmi pomocou obsluhy udalostí (on-click, on-hover atď.)

Modul `App` reprezentuje všetku logiku a komunikáciu (obsluha udalostí, ...) medzi jednotlivými komponentami (widgetmi) užívateľského prostredia.




Obr. 3.3: Blokové schéma približujúce modul components. Nachádzajú sa tu objekty reprezentujúce jednotlivé komponenty (widgety) aplikácie načtrtnuté v sekcii 3.3 – `SubClass`, `Waveform` a `VideoPlayer`. Návrh jednotlivých komponentov (widgetov) je popísaný v sekciách nižšie.

### SubClass

Komponenta `SubClass` reprezentuje titulky. Zabezpečuje ich zobrazenie v upravenej stĺpcovej forme (viac v popise obrázku 3.8c, touto reprezentáciou sa šetrí vertikálny priestor a umožní sa tak zobraziť viac titulkov v jednom viditeľnom okne), zvýraznenie pri nabehnutí nad titulok (on-hover) a zvýraznenie po kliknutí na titulok (on-click).

### Waveform

Komponenta `Wavefrom` predstavuje waveform zvuku z videa. Zaisťuje nahranie (nanesenie/vykreslenie) prehovorov z titulkov na waveform vo forme obdĺžnikov. Každý obdĺžnik (prehovor) má svoj index zdedený z indexu titulku, ktorý daný obdĺžnik (prehovor) reprezentuje. Dĺžka a umiestnenie prehovorov sa zistí z časov titulkov (viac v sekcii o module Misc 3.4). Zabezpečuje vykreslenie waveformu a obsluhu udalostí týkajúcich sa interakcie s waveformom.

### VideoPlayer

Prehrávanie a manipuláciu s videom zabezpečuje komponenta `VideoPlayer`. Okrem samotného prehrávania videa zariaďuje posúvanie času vo videu (posuvník), zastavenie a opätovné spustenie videa (pause/play) a aktualizáciu času videa.

### TopLevelWindow

`TopLevelWindow` je komponenta reprezentujúca okno načítania. Funguje ako vyskakovacie okno, ktoré sa po vykonaní daného načítania zavrie.

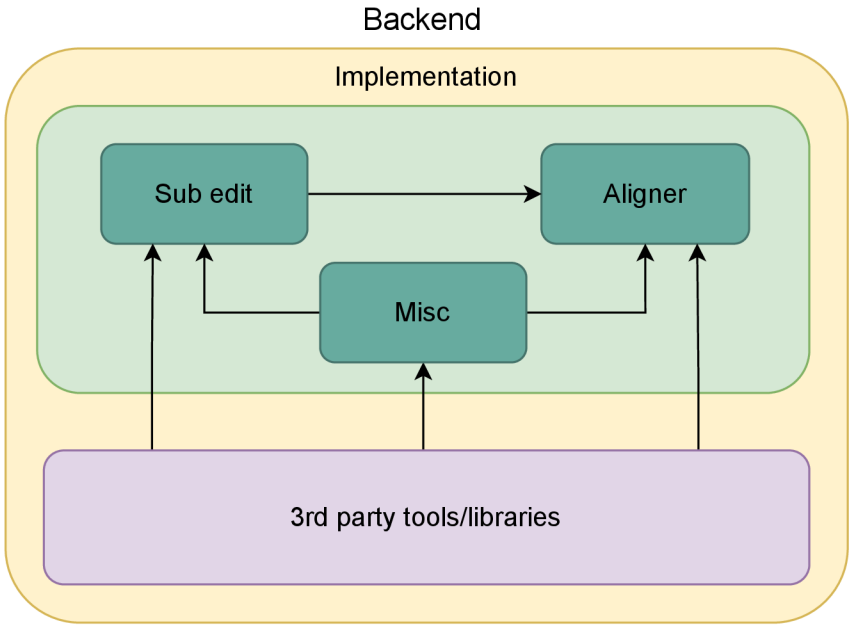
### App

Ako bolo vyššie spomenuté modul `App` zastupuje všetku logiku a komunikáciu medzi jednotlivými komponentami – modul App by sa dal považovať za hlavnú komponentu. Zabezpečuje obsluhu vyberania súborov a interakciu komponentov – viac k interakcii komponentov v sekcii 3.5.1 a 3.5.2.

Viac o fungovaní užívateľského prostredia v kapitole 3.5 a jeho implementácii sa čitateľ dočíta v kapitole 4.

## 3.4 Backend – návrh

V tejto sekcii je popísaný návrh hlavného algoritmu a jeho podporných súčastí. Hlavnú súčasť výpočtu tvorí konverzia audio-signálu vo forme waveformu (sekcia 2.2) na MFCC (Mel-frekvenčné cepstrálne koeficienty, pozri sekciu 2.1.4) a následné zarovnanie signálov algoritmom DTW, ktorý je priblížený v sekcii 2.1.2. Výsledok zarovnania sa následne analyzuje a nájdu sa dôležité body (strihy), ktoré sa použijú na úpravu pôvodných titulkov. Návrh časti backend je ilustrovaný na obrázku 3.4.




Obr. 3.4: Blokové schéma backendu aplikácie. Znázorňuje prepojenie s knižnicami a inými pomocnými prostriedkami a štruktúru backendu. Vysvetlenie jednotlivých modulov a prostriedkov je popísaný v sekciách nižšie.

### Aligner

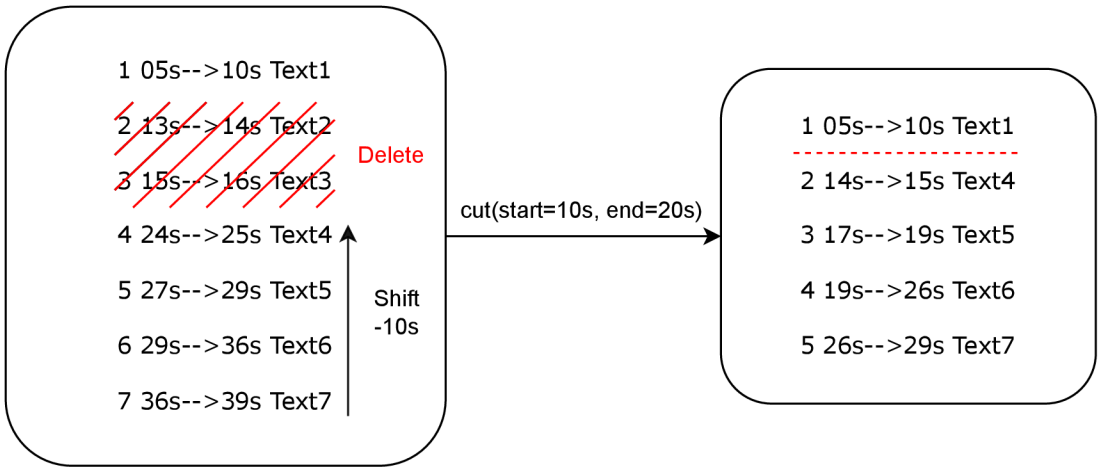
V tomto module prebieha realizácia hlavného algoritmu, a teda rozdelenie signálu na rámce, normalizácia týchto rámcov a extrakcia MFC koeficientov, ktorá aproximuje signál na menšom vzorku (viac v sekcii 2.1.4) a následné zarovnanie signálu algoritmom DTW – konkrétne jeho optimalizáciou FastDTW (viac v sekcii 2.1.2). Nakoniec sa podľa výslednej DTW cesty aproximujú časy a dĺžky strihov (chýbajúcich úsekov videa). Presný postup je popísaný sekciou 4.3.

### SubEdit

Modul riešiaci zarovnanie (prečasovanie) titulkov z výsledku (aproximácie strihov, strih je aproximovaný časom začiatku a konca strihu) modulu `Aligner`.

**Zarovnanie** prebieha nasledovne:

1. Rozdelenie titulkov na dve časti v čase začiatku strihu.
2. Vymazanie začiatku druhej časti o dĺžku strihu.
3. Posunutie titulkov druhej časti o dĺžku strihu.
4. Konkatenácia rozdelených častí.




Obr. 3.5: Prečasovanie titulkov podľa zisteného strihu. Strih vyjadrený funkciou `cut(start=10s, end=20s)`, kde parameter `start` je začiatok strihu a `end` je koniec strihu. Posunutie (dĺžka strihu) sa získava odčítaním koncového bodu od začiatočného bodu strihu ($start - end = 10s - 20s = -10s$).

### Misc

Modul obsahujúci pomocné programy na výpočet algoritmu či úpravu titulkov na zobrazenie potrebné v užívateľskom prostredí (viac v sekcii 3.5). Nachádza sa tu napríklad načítanie videí a ich konverzia na audio a následná kompresia. Dôležité je implementovať program, ktorý vráti časy prehovorov z titulkov v sekundách v zozname, ktorý sa použije v UI na zobrazenie prehovorov vo waveforme.

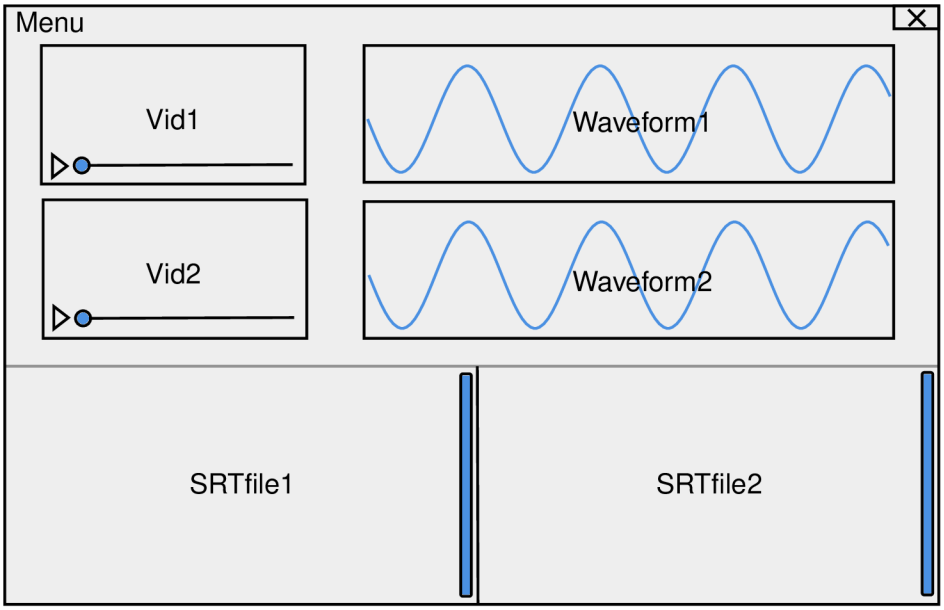
### Knižnice a nástroje tretích strán

Na vytvorenie optimálneho algoritmu výpočtu zarovnania je vhodné použiť prostriedky (nástroje/knižnice), ktoré uľahčia prácu alebo optimalizujú nasledujúce úkony:

- práca s titulkami – je vhodné vytvoriť vnútornú reprezentáciu titulkov a nepracovať len čisto s textovou formou.
- načítanie, rámcovanie a extrahovanie koeficientov zo signálu – je potrebné použiť čo najlepšiu implementáciu týchto úkonov.
- DTW algoritmus – rovnako dôležité je použiť implementáciu DTW algoritmu, ktorá zaručí najoptimálnejší výsledok (najlepší pomer rýchlosť/presnosť).
- konverzia a kompresia videa/zvuku – vybrať správnu knižnicu/nástroj na zefektívnenie práce s audiovizuálnym vstupom.

## 3.5 Užívateľské prostredie – UI

Prostredie aplikácie by malo užívateľovi poskytnúť grafický pohľad na vložené súbory a na výstup zarovnania. Preto je vhodné rozdeliť aspekty UI na **pred** zarovnaním a **po** zarovnaní. Mockup aplikácie zobrazuje obrázok 3.6.



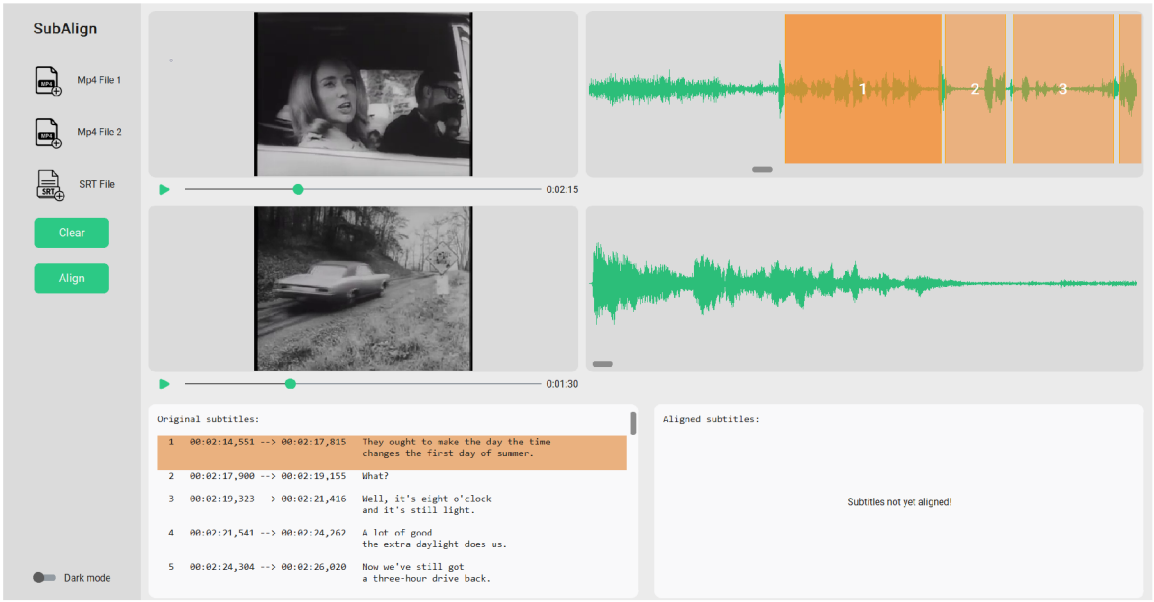

Obr. 3.6: Mockup aplikácie, rozdeľuje aplikáciu na 3x2 grid a to na dvojice komponentov: `Vid1` a `Waveform1`, `Vid2` a `Wavefevorm2` a `SRTfile1` a `SRTfile2` a menu.

### 3.5.1 Pred zarovnaním

Grafický pohľad je realizovaný trojicou komponentov – video, waveform (pozri sekciu 2.2) signálu a text titulkov. Užívateľ je schopný vybrať a vložiť MP4 súbor(y) a súbor s titulkami typu SRT (priblíženie SRT formátu v sekcii 2.3.1). V okne aplikácie je možné prehrať video, prezerať (skrolovať) text titulkov a waveform audio-signálu z videa.

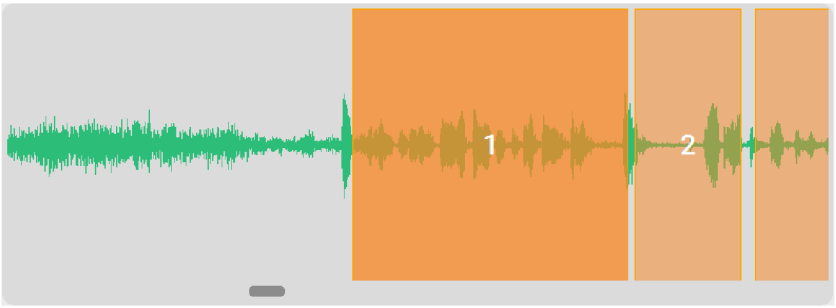
#### Interakcia komponentov

Po nahratí videa a príslušných titulkov sa do komponentu `Waveform1` nahrajú prehovory z titulkov vo forme obdĺžnikov (do komponentu `Waveform2` sa prehovory nahrajú až po zarovnaní titulkov, pozri sekciu 3.5.2). Užívateľovi je umožnené interagovať s prehovormi formou kliknutia na prehovor komponentu `SRTfile1/2` – tým sa zobrazí prehovor v komponente `Waveform1/2` a naopak. Po kliknutí na prehovor v komponentoch `SRTfile1/2` alebo `Waveform1/2` sa príslušné video (`Vid1/2`) presunie na čas prehovoru. Výsledný vzhľad aplikácie predstavuje obrázok 3.7.

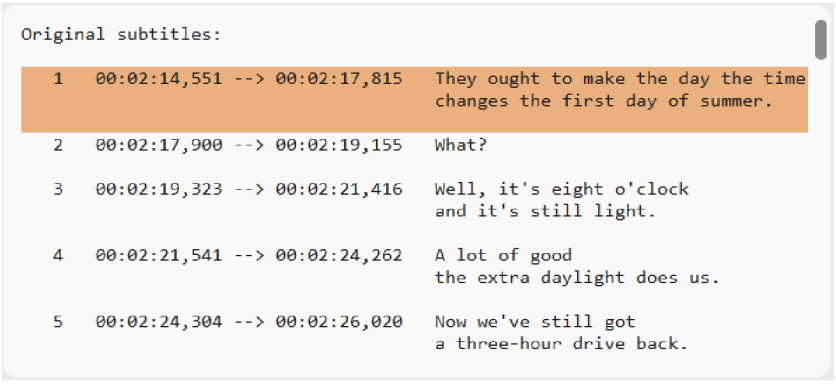



Obr. 3.7: Vzhľad aplikácie pred zarovnaním, nahrané dve video a titulky k prvému videu, vybraný prvý prehovor prvého videa, videá sú zastavené. Na druhom waveforme nie sú zobrazené prehovory, pretože nie sú k dispozícii zarovnané titulky. Popis jednotlivých komponentov približuje obrázok 3.8


(a) Menu panel




(b) Waveform audia s prehovormi – komponenta `Waveform1`

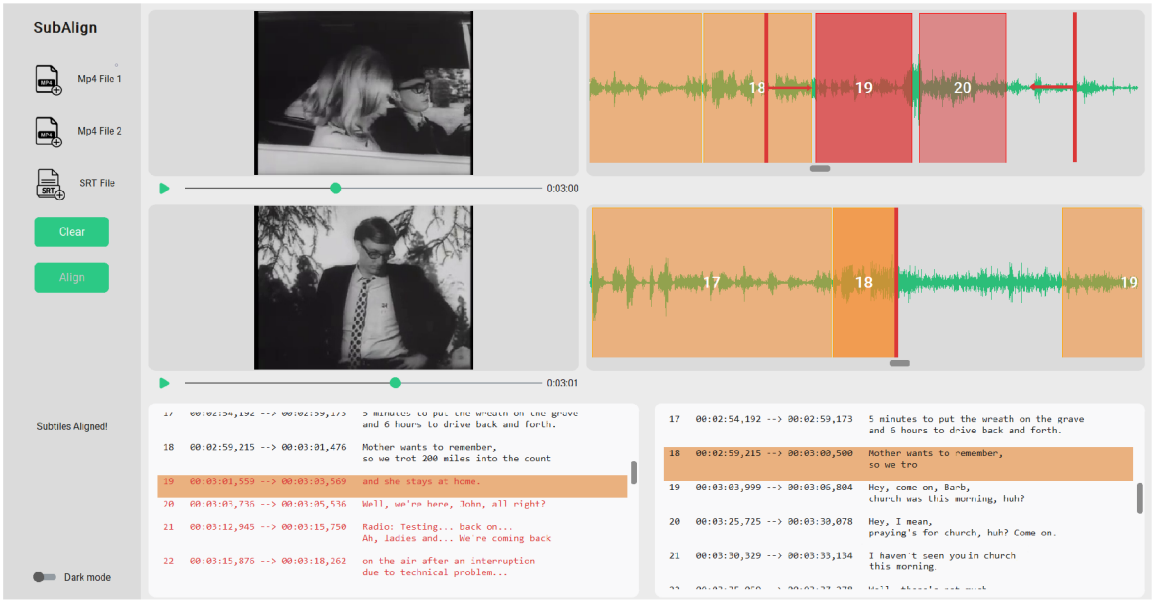



(c) Titulky k videu – komponenta `SRTfile1`

Obr. 3.8: Na obrázku 3.8a možno vidieť menu aplikácie, prvé tri tlačidlá na pridanie súborov, pod nimi tlačidlá `Clear` slúžiace na vyčistenie všetkých okien a `Align`, ktoré spustí proces zarovnávania (je aktívne až po pridaní potrebných súborov – Mp4 File 1 a SRT file) a prepínač farebnej témy (Dark, Light). Obrázok 3.8b približuje komponentu `Waveform1` – waveform audia z videa, prehovory vo forme obdĺžnikov (začiatok, koniec a dĺžka odpovedajú času prehovoru) a scrollbar. Obrázok 3.8c zobrazuje komponentu `SRTfile1` – nadobúda upravenú formu SRT formátu, kde sa každý prehovor rozprestrie do 3 stĺpcov (narozdiel od SRT formátu, kde je jeden prehovor rozdelený do riadkov) – **index**, **čas** a **text prehovoru**. Vyznačený prvý prehovor.

### 3.5.2 Po zarovnaní

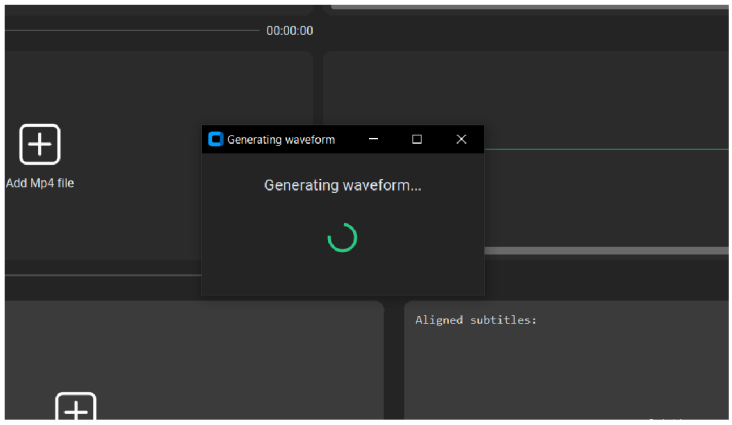
Po zarovnaní titulkov (kliknutie na tlačidlo `Align`) sa v komponente `SRTfile2` zobrazí text zarovnaných titulkov a do komponentu `Waveform2` sa nahrajú prehovory zarovnaných titulkov. V komponentoch `Waveform1` a `SRTfile1` sa zmenia vymazané (sekcia 3.2) prehovory na červenú farbu a kliknutím na vymazaný prehovor sa (okrem funkcionality spomenutej v sekcii 3.5.1) zobrazí chýbajúca sekcia signálu v komponente `Waveform2` (ktorá je taktiež červená), interakcia funguje aj opačným spôsobom (klik na chýbajúca sekciu → scroll na vymazané prehovory). V komponente `Waveform1` je taktiež indikovaný strih červenými obdĺžnikmi so šípkami označujúcimi smer strihu. Výstupný súbor titulkov pod rovnakým názvom ako druhý (zarovnávaný) videosúbor je uložený v rovnakom priečinku. Stav po zarovnaní demonštruje obrázok 3.9 a popis zobrazenia strihu vo waveformoch po zarovnaní ilustruje príloha A.




Obr. 3.9: Vzhľad aplikácie po zarovnaní. V komponente `SRTfile1` sú vymazané titulky zobrazené červenou farbou. Pridaná interakcia medzi červeným indikátorom vymazanej časti v komponente `Waveform2` a vymazanými prehovormi v komponente `Waveform1` je opísaná vyššie. Text indikujúci úspešné zarovnanie `Subtitles Aligned!` v menu paneli.

### 3.5.3 Rekapitulácia fungovania UI

Po spustení aplikácie je užívateľ privítaný prázdnymi oknami na výber 3 súborov: prvé video, druhé video a titulky. Užívateľ je shopný zmeniť farebný mód prepínačom označeným **Dark mode**. Po vybratí a načítaní (pozri obrázok 3.10) potrebných súborov (užívateľ môže použiť tlačidlá v menu aplikácie alebo v hlavnom okne aplikácie) sa užívateľovi umožní spustiť zarovnávanie tlačidlom **Align** (taktiež môže vyčistiť hlavné okno tlačidlom **Clear**) a umožní sa interakcia medzi prvým videom, audio-signálom a príslušnými titulkami (pre bližší popis interakcie pozri sekciu vyššie 3.5.1). Po zarovnaní má užívateľ k dispozícii interakciu medzi druhým videom, audio-signálom a zarovnanými titulkami (seckia 3.5.2).




Obr. 3.10: Načítavanie waveformu zvuku z videa. Aplikácia je v tmavom farebnom režime. Okno sa po načítani waveformu zavrie.

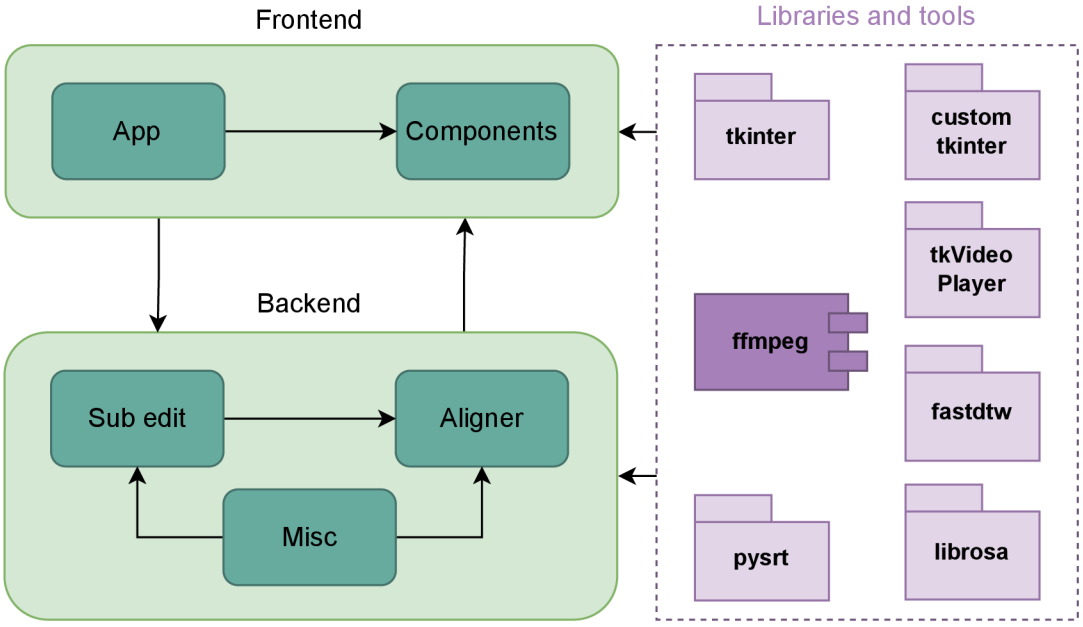
# Kapitola 4

# Implementácia

V tejto kapitole je bližšie popísaná konkrétna implementácia jednotlivých súčastí aplikácie a sú ozrejmené použité knižnice či nástroje tretích strán a ich využitie. Štruktúru implementácie vizualizuje obrázok 4.1.

## 4.1 Použité nástroje a prostriedky

Vývoj aplikácie prebiehal v programovacom jazyku Python (v. 3.10.8) za pomoci textového editora VS Code na operačnom systéme Windows. Na verzovanie bol použitý nástroj Git (GitHub). Na konverziu a kompresiu (pozri sekciu 2.4) videa (na audio) slúžil nástroj ffmpeg (v. 4.3.1, pozri sekciu 4.3).




Obr. 4.1: Blokové schéma približujúce knižnice použité v implementácii. Pozícia knižníc v schéme odzrkadľuje ich využitie v backende a frontende.

### Použité knižnice

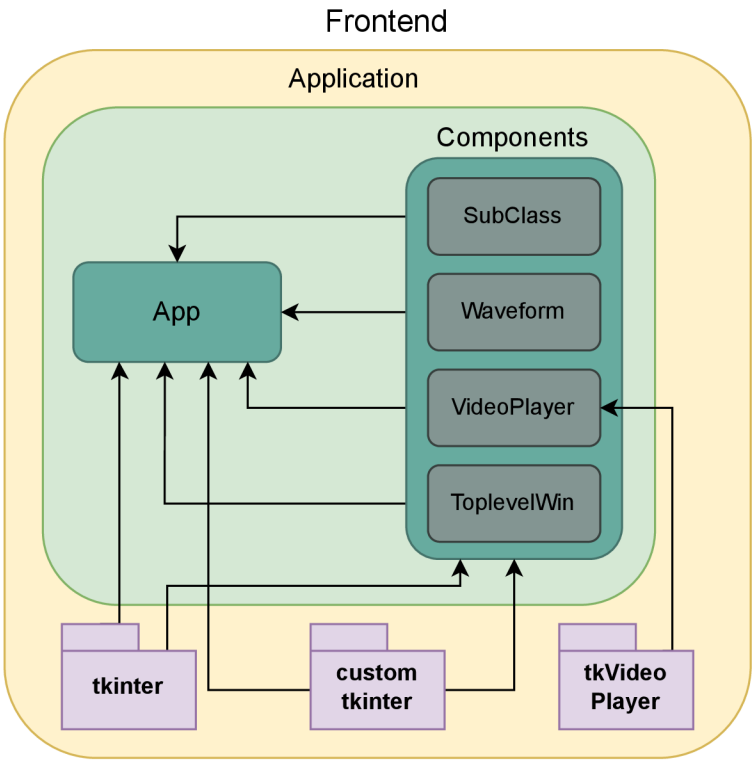
Pri implementácii aplikácie boli použité nasledujúce knižnice jazyka Python:

- **pysrt** (v. 1.1.2) – na prácu s titulkami.
- **librosa** (v. 0.9.2) – na načítanie a prácu s audio-signálom.
- **fastdtw** (v. 0.3.4) – na zarovnanie signálov v algoritme (pozri sekciu 2.1.2).
- **tkinter/customtkinter** (v. 8.6.12/v. 5.1.2) – na tvorbu užívateľského prostredia.
- **tkvideoplayer** (v. 2.3) – prehrávanie videa.
- **Pillow** (v. 9.3.0) – tvorba priehľadných obrázkov.
- **numpy** (v. 1.23.4) – na rýchlejšiu interakciu s dátami.

Knižnice a prostriedky zaujímavé pre implementáciu sú bližšie popísané v jednotlivých sekciách nižšie.

## 4.2 Frontend – implementácia

Ako jadro UI je použité rozhranie **Tkinter** (Tk interface) – prepája Python s Tcl/Tk nástrojmi. Komunikáciu a štruktúru frontendu vizualizuje obrázok 4.2 a presnejšie pomocou tried a vzťahov medzi nimi aj diagram tried na obrázku 4.3.




Obr. 4.2: Blokové schéma vyjadrujúce všetky závislosti súčastí frontendu. Jednotlivé komponenty (widgety) aplikácie: `SubClass`, `Waveform`, `VideoPlayer` a `ToplevelWindow` sú implementované ako triedy a popísané nižšie. Knižnice **tkinter** a **customtkinter** použité na implementáciu užívateľského rozhrania, ako aj knižnica sprostredkúvajúca prehrávanie videa **tkVideoPlayer** sú taktiež popísané nižšie.

### Tkinter

Tkinter[1] je Python rozhranie na tvorbu grafických užívateľských prostredí a je súčasťou štandardnej knižnice Pythonu. Tkinter umožňuje tvorbu jednoduchých grafických prostredí a manipuláciu komponentov v rámci tohto prostredia. Tkinter aplikácie majú zastaralý vzhľad, ale vývojár má veľkú voľnosť pri ich „kostumizovaní“. Rozhranie Tkinter vychádza z balíčku Tk jazyka Tcl (Tk je GUI balíček pre jazyk Tcl implementovaný v jazyku C) a prepája Tk s Python prostredím.

Pomocou Tkinter môžu vývojári vytvárať okná, tlačidlá, menu, textové polia a ďalšie prvky GUI a potom ich spájať s obsluhou udalostí (klik, scrolling a iné) aby vytvorili interaktívne aplikácie. Tkinter poskytuje širokú škálu štandardných widgetov a umožňuje tvoriť vlastné widgety a štýly.

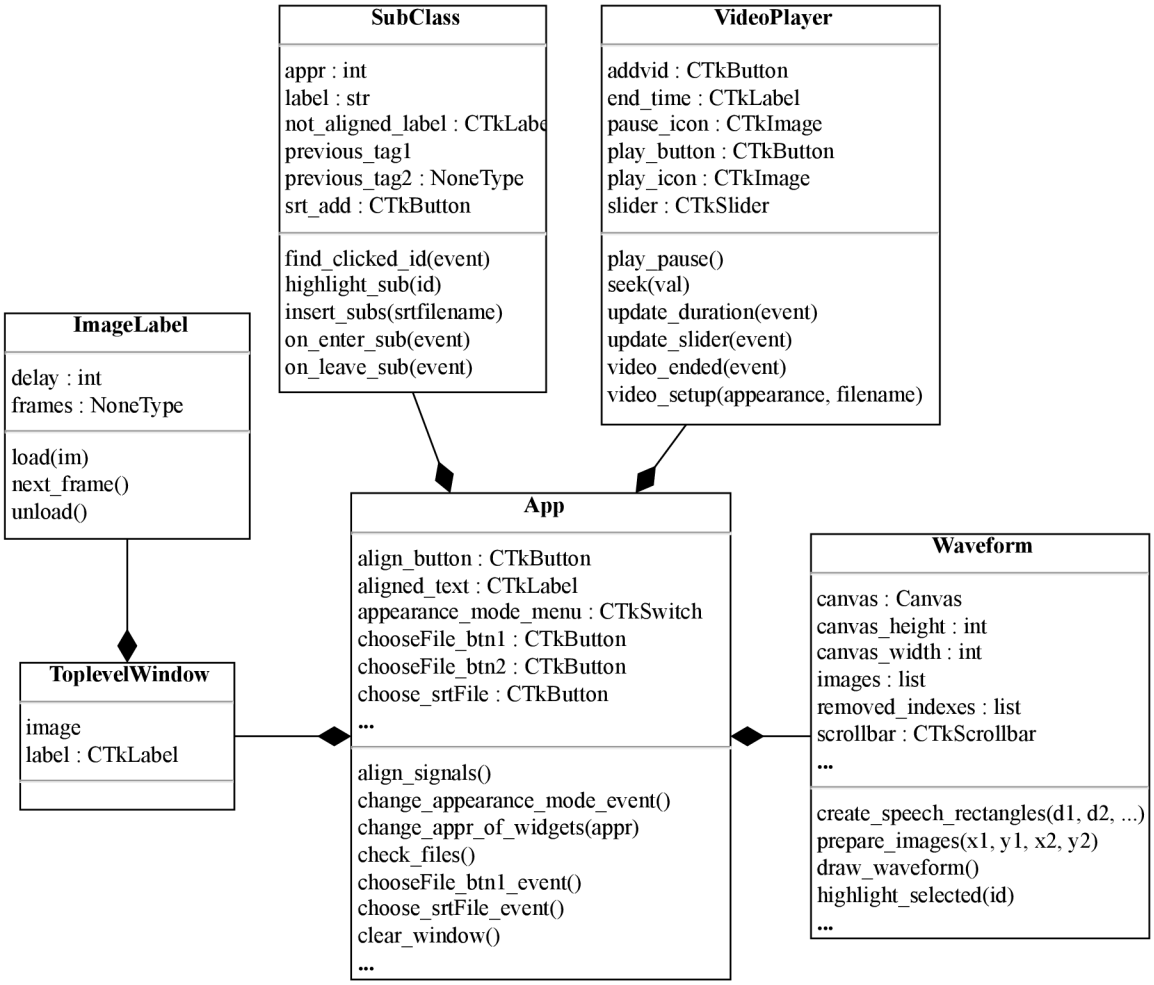
[1] Dokumentácia Tkinter – https://docs.python.org/3/library/tkinter.html

### CustomTkinter

Pre jeho archaický vzhľad by bolo použitie neupraveného rozhrania Tkinter pre modernú aplikáciu nedostačujúce. Preto sa zvolila možnosť kombinácie rozhrania Tkinter s grafickou knižnicou **CustomTkinter**[2] (založená na Tkinter), ktorá do práce s prostredím vnáša viac moderných aspektov – moderný vzhľad, zmena farebného motívu a zdokonalené prispôsobovanie a práca s widgetmi (vhodné je porovnanie knižnice CustomTkinter s Bootstrap[3] frameworkom pre vývoj webových stránok).

### TkVideoPlayer

Rozhranie Tkinter nepodporuje vloženie a prehrávanie videa, preto sa na tento problém použila knižnica **TkVideoPlayer**[4], ktorá je taktiež založená na rozhraní Tkinter. Umožňuje prehrávanie videa, zastavenie či pretáčanie videa a poskytuje informácie o súčasnom stave prehrávania.




Obr. 4.3: Diagram tried frontendu aplikácie. Zaujímavé časti implementácie sú popísané nižšie. Niektoré parametre a metódy nie sú uvedené, indikované troma bodkami (...).

[2]CustomTkinter – https://github.com/TomSchimansky/CustomTkinter
[3]Bootsrap – https://getbootstrap.com/
[4]TkVideoPlayer – https://github.com/PaulleDemon/tkVideoPlayer

### SubClass

`SubClass` je trieda reprezentujúca titulky v rámci užívateľského prostredia. Dedí od triedy `CTkTextbox` implementovanej knižnicou customtkinter. Trieda `CTkTextbox` poskytuje vertikálne aj horizontálne skrolovanie ak dôjde k pretečeniu textu v okne. Tmavé zvýraznenie titulkov pri prechode myšou je implementované naviazaním funkcie `on_enter_sub()` na udalosť `<Enter>` a funkcie `on_leave_sub()` na udalosť `<Leave>`, priblíženie implementácie vo výpise 4.1. `<Enter>` a `<Leave>` sú udalosti implementované knižnicou tkinter.

```
1     for sub in subs:
2         self.tag_bind(str(sub.index - 1), "<Enter>", self.on_enter_sub)
3         self.tag_bind(str(sub.index - 1), "<Leave>", self.on_leave_sub)
4
5     def on_enter_sub(self, event):
6         tags = findTags(event.x, event.y)
7         self.previous_tag1 = tags[1]
8         event.widget.tag_configure(tags[1], background="#dbdbdb")
9
10    def on_leave_sub(self, event):
11        event.widget.tag_configure(self.previous_tag1, background="")
```

Výpis 4.1: Naviazanie udalostí na zvýrazenie titulku pri prechode myšou nad ním. Na riadkoch 2 a 3 možno vidieť spomínané naviazanie na udalosť, avšak udalosť treba naviazať na každý titulok zvlášť pomocou tagu (každý titulok má tag o hodnote jeho indexu v SRT formáte), inak (ak by sa udalosť naviazala na jeden tag, ktorý by reprezenotval všetky titulky) sa zvýraznenie z titulku neodstráni pokiaľ ukazateľ myši neodíde z widgetu titulkov. Nepodstatné časti kódu sú zjednodušené alebo odstránené.

Ďalej trieda `SubClass` definuje metódu na načítanie titulkov `insert_subs()` (používa funkciu konvertujúcu titulok na stĺpcovú formu z modulu `Misc`), pomocné metódy `find_clicked_id()` a `highlight_sub()`, ktoré využíva metóda `sub_click_event()` (sekcia 4.2).

Všetka obsluha udalostí je založená na naviazaní danej udalosti (napr. klik) na určitú funkciu (riadok 1 vo výpise 4.2), v prípade kliku na prehovor (či už vo waveforme alebo titulkoch) je táto funkcia naviazaná len na objekty, ktoré obsahujú určitý **tag** (riadok 2 vo výpise 4.2) alebo je funkcia naviazaná priamo na widget knižnice – tlačidlo, srollbar atď. (riadok 3 vo výpise 4.2).

```
1     self.cancas.bind("<Enter>", self.foo)
2     self.canvas.tag_bind("tag", "<Button-1>", self.bar)
3     button = customtkinter.CTkButton(self, command=self.baz)
```

Výpis 4.2: Ukážka rôzneho naviazania obsluhy udalosti.

### Waveform

Waveform (audio-signál) v aplikácii reprezentuje trieda `Waveform`, ktorá dedí od triedy `CTkFrame` knižnice customtkinter. Okrem obsluhy udalostí týkajúcich sa generovania waveformu a interakcie s ním záštituje funkcie na prípravu a zobrazenie prehovorov vo waveforme – metóda `prepare_images()` a `create_speech_rectangles()`, bližšie popísané vo výpise 4.3.

```
    def prepare_images(self, x1, y1, x2, y2, **kwargs):
        alpha = int(kwargs.pop('alpha') * 255)
        orange_color = (250, 135, 35)
        fill = orange_color + (alpha,)
        image = Image.new('RGBA', (x2 - x1, y2 - y1), fill)
        self.images.append(ImageTk.PhotoImage(image))
```

Výpis 4.3: Ukážka tvorby priehľadných obrázkov na reprezentáciu prehovorov. Tkinter neumožnuje tvorbu priehľadných objektov vo widgete Canvas, riešením (idea prevzatá z: https://stackoverflow.com/a/54645103) je vytvoriť sadu priehľadných obrázkov, čo umožnuje knižnica **Pillow**, ktorá podporuje RGBA – farebný model, ktorý rozširuje klasický RGB model o štvrtý alfa kanál (ten udáva priehľadnosť). Premenná `image` je teda zoznam obrázkov reprezentujúcich jednotlivé prehovory, pričom index obrázku v poli prislúcha indexu daného prehovoru (off-by-one error). Metóda `create_speech_rectangles()` potom už len dané obrázky zobrazí na Canvas. Nepodstatné časti kódu sú zjednodušené alebo odstránené.

### Kreslenie waveformu

Waveform a objekty prehovorov sa vykresľujú na tkinter widget `Canvas`. Vykreslí sa vždy iba časť signálu na základe pozície scrollbaru. Vykresľuje sa podvzorkovaný signál na 800Hz pričom x-ové súradnice sa posúvajú po 0.1, teda 10 vzorkov sa vykreslí na jeden pixel, čím sa prakticky podvzorkuje na 80 vzorkov za sekundu (1 sekundu predstavuje 80 pixelov), pri pevne danej šírke Canvas widgetu (915 pixelov) sa do okna waveformu vykreslí približne 11.5 sekundy waveformu ($915/80 \doteq 11.5$). Vhodná frekvencia pre podvzorkovanie bola stanovená pomocou empirických pokusov. Približovanie waveformu nie je implementované.

### VideoPlayer

Za použitia knižnice `tkVideoPlayer` definuje trieda `VideoPlayer` všetky metódy potrebné na prehrávanie a manipuláciu s videom. Metódou `video_setup()` sa vytvorí inštancia triedy `TkinterVideo` knižnice `tkVideoPlayer` reprezentujúca médium, teda video. Okrem rutinných funkcionalít (pause/play, seek...) trieda implementuje obsluhu udalostí, ktoré aktualizujú čas prehrávania, posuvník prehrávania podľa času a obsluhu konca videa, viac vo výpise 4.4.

```
    self.video.bind("<<Duration>>", self.update_duration)
    self.video.bind("<<SecondChanged>>", self.update_slider)
    self.video.bind("<<Ended>>", self.video_ended)
```

Výpis 4.4: Udalosti definované knižnicou tkVideoPlayer, udalosť `«Duration»` nastane ihneď po nahratí a obsluhou sa zistí dĺžka videa, `«SecondChanged»` je udalosť prirodzene nastane každú sekundu (obsluhou sa aktualizuje poloha posuvníka a čas prehrávania) a jej obsluhou udalosti `«Ended»` sa resetuje posuvník.

### ToplevelWindow

Funkcia triedy `ToplevelWindow` je vytvorenie vyskakovacieho (ang. pop-up) okna pri načítavaní waveformu a zarovnania. Pri generovaní waveformu či zarovnávaní signálov sa pre tieto úlohy vytvorí vlastné vlákno, počas behu vlákna existuje aj inštancia triedy TopLevelWindow, ktorá sa po skončení behu vlákna zavrie (po zavretí okna sa daná úloha neukončí).

Zaujímavosťou implementácie pop-up okna je animácia načítavacieho kolečka (obrázok 3.10). Keďže tkinter nepodporuje animáciu napr. GIF obrázku, treba implementovať vlastné spracovanie animácie. Spracovanie animácie má na starosti trieda `ImageLabel` (dedí od tkinter triedy Label), ktorá nahrá všetky obrázky animovaného GIF súboru do premennej, ktorú potom metóda `next_frame` iteruje (a pri každej iterácii vymení obrázok inštancie, `self.config(image=next(self.frames))`, čo je funkcionalita triedy Label) až do zavretia pop-up okna. Kód animácie je prevzatý z: https://pythonprogramming.altervista.org/animate-gif-in-tkinter/.

### App

Hlavná trieda aplikácie, tvorí hlavný proces chodu aplikácie, volá funkcie backendu a obsluhuje tlačidlá pre vyberanie vstupov atď. Je tu vytvorená väčšina widgetov aplikácie, menu aplikácie, inicializácia hlavných komponentov (widgetov) aplikácie – `tkvideo1/2` (trieda `VideoPlayer`), `first_frame` a `second_frame` (trieda `Waveform`) a `orig_sub` a `aligned_sub` (trieda `SubClass`).

Ďalšie metódy na obsluhu udalostí (nie sú nutne súčasťou triedy `App`):

- `sub_click_event()` – obsluha udalosti kliknutia na titulok – zistí tag (index) kliknutého titulku a scrollnutím vo widgete waveformu zobrazí daný prehovor a zavolá metódu `highlight_selected()`.
- `selected_speech_event()` – obsluha udalosti kliknutia na prehovor vo waveforme – zistí tag (index) kliknutého prehovoru a scrollnutím vo widgete príslušných titulkov zobrazí daný titulok a zavolá metódu `highlight_selected()`.
- `highlight_selected()` – zvýrazní prehovor vo waveforme a príslušný titulok a pretočí video na príslušný čas.

### Komunikácia s backendom

Dôležitou premennou pre tvorbu prehovorov je premenná `segments`, ktorú vráti funkcia `get_speech_segments()` modulu `Misc` (viac v sekcii Misc nižšie). Premenná je typu poľa dvojíc, kde každá dvojica reprezentuje čas začiatku a konca prehovoru. Vynásobením vzorkovacou frekvenciou hodnoty v premenej `segments` reprezentujú indexy vzoriek začiatku a konca prehovoru v signáli získanom funkciou `get_wav()` modulu `Misc` – signál je reprezentovaný premennou typu pole (numpy.array) reálnych čísel (float) normalizovaných na hodnoty medzi -1 a 1. S pomocou tejto premennej sa potom v cykle volá funkcia `prepare_images()` na vytvorenie obrázkov pre prehovory.

Funkcia `align()` z modulu backend (sekcia 4.3) má tri výstupy:

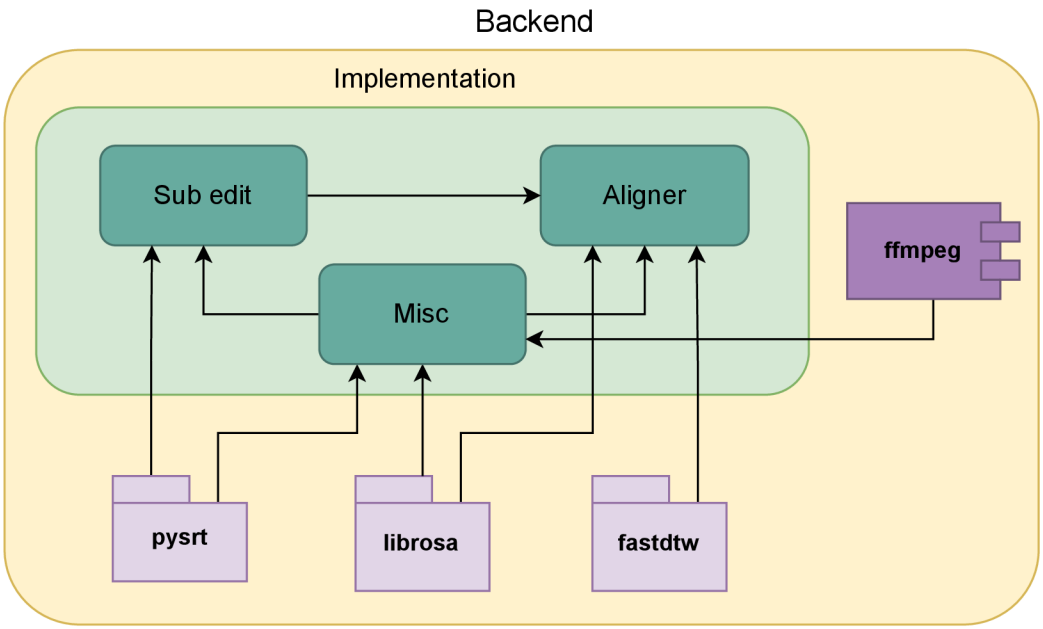
- `removed_indexes` – indexy reprezentujúce vymazané prehovory (index – poradové číslo prehovoru), sú dôležité pre vyznačenie vymazaných titulkov červenou farbou (v texte titulkov aj vo waveforme) – metóda `prepare_removed_images()` vymení obrázky v premennej `images` na indexoch obsiahnutých v `removed_indexes` za obrázky červenej farby (obrázok 3.9).
- `signal_mismatch` – indexy začiatku každého strihu (index – index v numpy poli signálu) – využité pri označení začiatku strihu červeným obdĺžnikom v druhom waveforme (obrázok 3.9).
- `grouped` – rovnaké indexy ako v `removed_indexes`, ale zoskupené ako pole polí, kde vnútorné polia reprezentujú jednotlivé strihy – potrebné na prekliknutie (popísané v sekcii 3.5.2) z vymazanej časti v jednom waveforme na vymazanú časť v druhom waveforme, zoskupenie je potrebné na správnu identifikáciu začiatku a konca strihu.

### Nedostatky implementácie frontendu

Knižnica `tkVideoPlayer` neposkytuje prehrávanie zvuku, pri pretočení videa môže niekedy nastať zrýchlenie prehrávania (fps bug). V interakcii kliknutia na prehovor vo waveforme alebo titulkách a následnom pretočení videa na daný segment môže nastať odchýlka asi -2 s. Pri nesprávnom nájdení počtu a dĺžky strihov algoritmom má interakcia medzi komponentami (rovnako aj interakcia s aplikáciou pri načítavaní či zarovnávaní signálov) nedefinované správanie.

## 4.3 Backend – implementácia

Sekcia sa venuje priblíženiu zaujímavostí a problémov implementácie backendu aplikácie. Náčrt štruktúry backendu je zobrazený blokovou schémou na obrázku 4.4.




Obr. 4.4: Blokové schéma zachytáva závislosti backendu aplikácie. Odkrýva použité knižnice **pysrt**, **librosa** a **fastdtw** a nástroj **ffmpeg**, ktorých presné využitie je popísané nižšie. Spolupracujú s modulmi `SubEdit`, `Alinger` a `Misc` na zarovnaní signálov a následne titulkov.

### Pysrt

Pysrt[5] je Python knižnica na prácu so súbormi titulkov formátu SubRip (prípona .srt, viac o tomto formáte v sekcii 2.3.1). Poskytuje jednoduché rozhranie na čítanie, písanie a manipuláciu so súbormi SRT v Pythone. Umožňuje načítať súbor SRT do programu, upraviť jeho obsah a potom ho uložiť späť na disk. Pysrt podporuje širokú škálu operácií so súbormi SRT vrátane zlučovania, rozdeľovania (slice) alebo posúvania (shift). Hlavným objektom titulkov načítaných pomocou pysrt je `SubRipItem`. Predstavuje jeden blok titulkov, ktorý pozostáva z indexu, časových razítiek (timestamp) označujúcich čas začiatku a konca titulku, dĺžky titulku a samotného textu titulku.

### Librosa

Librosa[6] je knižnica [14] jazyka Python na analýzu a spracovanie zvukových signálov. Poskytuje širokú škálu nástrojov a funkcií na prácu so zvukovými údajmi, ktoré napomáhajú pri rozpoznaní reči či získavaniu informácií z hudby (bližšie popísané v sekcii 2.1.4). Použitie funkcií knižnice librosa pri implementácii (`librosa.load()`, `librosa.util.frame()`, `librosa.util.normalize()` a `librosa.feature.mfcc()`) je popísané v sekcii 4.3.

[5] pysrt – https://github.com/byroot/pysrt
[6] librosa – https://librosa.org/doc/latest/index.html

### FastDTW – knižnica

Kalkuluje DTW algoritmus spôsobom popísaným v sekcii 2.1.2. Knižnica FastDTW[7] poskytuje možnosti optimalizácie priebehu algoritmu v podobe parametrov **radius** a **dist** funkcie `fastdtw()`, ktoré je možné prispôsobiť na úpravu výpočtu cesty.

### FFmpeg

FFmpeg[8] je nástroj na úpravu, konvertovanie/transformovanie, enkódovanie a dekodóvanie multimedií (viac o týchto operáciach v sekcii 2.4). Umožňuje konvertovanie videosúborov na iný typ videosúboru či na audio-súbor pričom dovoľuje nastaviť kvalitu (bitrate), počet snímkov za sekundu (fps), alebo vzorkovanie (vzorkovaciu frekvenciu). Podporuje obrovské množstvo video a audio kodekov (H.264, HEVC, AAC, MP3, ...) a transkódovanie medzi nimi. Okrem toho poskytuje filtre na úpravu audio-vizuálnych aspektov súboru, t.j. strih, orezanie, zmena rozlíšenia, korekciu farieb alebo zmenu hlasitosti a zakomponovanie titulkov. Nástroj FFmpeg má široké využitie v multimediálnom svete, využívaný je mnohými platformami ako YouTube alebo Twitch a je zakomponovaný do videoprehrávačov či nahrávačov ako VLC, OBS alebo ShareX.

**Použitie** – FFmpeg nemá žiadne užívateľské prostredie, na všetky úlohy využíva príkazový riadok (terminál). Napr. `ffmpeg -i in.mp4 out.wav`, týmto príkazom sa prevedie vstupný video súbor `in.mp4` na výstupný audio súbor `out.wav`, argumentom `-i` sa špecifikuje vstup. Súčasťou ffmpeg je aj program `ffplay`, slúžiaci na prehrávanie multimedií a `ffprobe` na analýzu metadát multimediálnych súborov.

### SubEdit

Modul `SubEdit` implementuje (v súbore `sub_edit.py`) zarovnanie titulkov za podpory knižnice pysrt vo funkcii `sub_edit()`. Vďaka pysrt je proces posunutia titulkov jednoduchý (výpis 4.5). Rozdelenie titulkov zabezpečuje metóda `slice()` a pomocou parametrov `starts_before` a `ends_after` sa titulky rozdelia na časť pred strihom a časť po strihu, použitím metódy `shift` sa posunie čas titulkov v druhej časti o dĺžku strihu a nakoniec sa časti konkatenujú podobne ako textové reťazce.

```
1    part1 = subs.slice(starts_before={'seconds': start})
2    part2 = subs.slice(ends_after={'seconds': end})
3    part2.shift(seconds=shift)
4    shifted_subs = part1 + part2
```

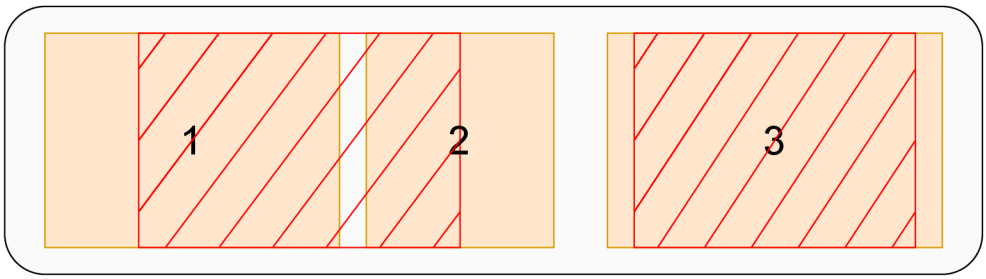
Výpis 4.5: Algoritmus posunutia titulkov (vysvetlené vyššie). Premenná `start` a `end` sú časové razítka strihu získané z modulu `find_cuts`.

[7] fastdtw – https://github.com/slaypni/fastdtw
[8] Dokumentácia ffmpeg je dostupná na https://ffmpeg.org/ffmpeg.html

**Problém** (edge case) nastáva pri prekryve strihu a prehovoru/titulku:

1. prehovor nezačína, ale končí vo vnútri strihu
2. prehovor začína, ale nekončí vo vnútri strihu
3. strih je vo vnútri prehovoru (veľmi zriedkavé)

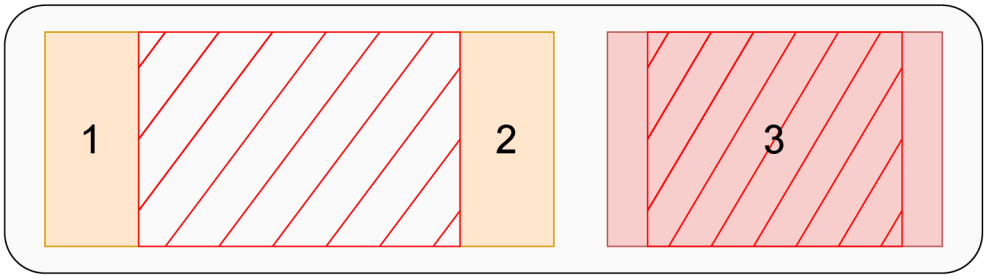
Problém je priblížený obrázkom 4.5.




Obr. 4.5: Vizualizácia rôznych možností prekryvu strihu s prehovorom. Oranžové obdĺžniky predstavujú prehovory a vyšrafovaný obdĺžnik predstavuje strih. Prehovor 1 predstavuje prehovor, ktorý končí (ale nezačína) vo vnútri strihu (problém 1), prehovor 2 predstavuje problém 2 a prehovor 3 – problém 3.

**Možnosti riešenia:**

- akonáhle strih zasahuje do prehovoru, titulok úplne vynechať (1, 2, 3)
- vynechať iba prehovory, ktoré začínajú vo vnútri strihu (2)
- vynechať iba prehovory, ktoré začínajú aj končia v strihu a titulky zasahujúce do strihu upraviť tak, aby súhlasili so strihom – predpokladá sa presné určenie strihu (obrázok 4.6).




Obr. 4.6: Riešenie prekryvu prehovorov a strihu implementované v module `SubEdit` – rešpektovať strih a upraviť prehovory. Rozumným riešením je vynechať všetky titulky, v ktorých je strih (veľmi zriedkavý problém), teda prehovor 3.

### Riešenie prekryvu

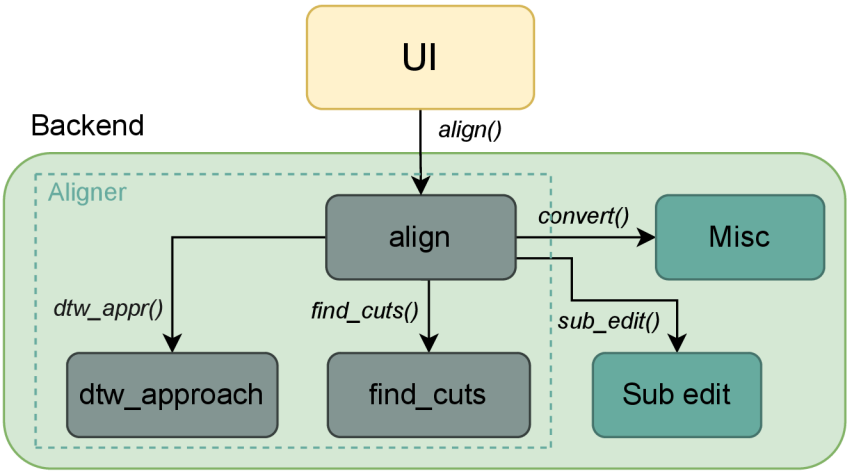
Riešenie (algoritmus 1) úpravy dĺžky (času) a textu titulkov je implementované pomerne naivným spôsobom – ak čas prehovoru zasahuje do strihu tak sa zistí dĺžka zasahujúcej sekcie prehovoru, z nej sa zistí zlomok prehovoru, ktorý do strihu nezasahuje (resp. zasahuje), koniec (resp. začiatok) titulku sa nastaví na začiatok (resp. koniec) strihu a text sa upraví podľa vypočítaného zlomku.

**if** *sub_invades_cut* **then**
  offset ← sub.end - cut.start
  delta ← 1 - offset / sub.duration
  sub.end ← cut.start
  sub.text ← sub.text[ : $delta \times len(sub.text)$]
**end**

**Algoritmus 1:** Pseudokód úpravy titulkov (vysvetlenie vyššie), konkrétne pre problém 1. Text sa upraví rezom textového reťazca (ang. string slicing).

### Aligner

Implementácia algoritmu na výpočet strihov je rozdelená do troch logických úsekov (súborov). Základ a riadiaca časť algoritmu (`align.py`), zistenie DTW cesty (`dtw_approach.py`) a aproximácia strihov (`find_cuts.py`).




Obr. 4.7: Blokové schéma približujúce modul `Alinger`. V súbore `align.py` je implementované riadiace centrum algoritmu, čo zahŕňa prípravu WAV audio-súborov funkciou `convert` (viac v sekcii Misc nižšie), volanie funkcií na zistenie DTW cesty a aproximáciu strihov, volanie funkcie `sub_edit()` na úpravu titulkov a napokon príprava výstupov potrebných pre frontend (sekcia 4.2).

### DTW cesta

Funkcia `dtw_appr()` v súbore `dtw_approach.py` implementuje algoritmus na výpočet DTW cesty nad vstupnými signálmi. Postup zahŕňa kroky popísané v sekcii 3.4, presnejší postup približuje algoritmus 2.

**input :** $file1$ (wav), $file2$ (wav) and $amp$ (0.25)
**output:** $DTWpath$

1 $x, sr1 \leftarrow$ `load(`*file1*`)`
2 $y, sr2 \leftarrow$ `load(`*file2*`)`
3 **if** $sr1 \neq sr2$ **then**
4 `error(`*"sr1 and sr2 are not same"*`)`
5 **end**
6 $frame\_len \leftarrow sr1 \times amp$
7 $frames1 \leftarrow$ `frame(`*x, frame_len*`)`
8 $frames2 \leftarrow$ `frame(`*y, frame_len*`)`
9 $norm\_frames1 \leftarrow$ `normalize(`*frames1*`)`
10 $norm\_frames2 \leftarrow$ `normalize(`*frames2*`)`
11 $mfccs1 \leftarrow$ `mfcc(`*norm_frames1, n_mfcc=10*`)`
12 $mfccs2 \leftarrow$ `mfcc(`*norm_frames2, n_mfcc=10*`)`
13 $dist, path \leftarrow$ `fastdtw(`*mfccs1, mfccs2*`)`
14 **return** *path*

**Algoritmus 2:** Hlavný algoritmus použitý na zistenie zarovnania vstupných signálov. Použité funkcie sú popísané nižšie. Premenná/parameter `amp` vyjadruje dĺžku jedného rámca (riadok 6), predvolená hodnota je *0.25* (dĺžka rámca 250ms).

**Funkcie použité v algoritme:**

- `load()` – načíta signál z WAV súboru, prevedie na mono, normalizuje a zistí vzorkovaciu frekvenciu.
- `frame()` – narámcuje signál na rámce o dĺžke 250ms (premenná `frame_len`).
- `normalize()` – normalizuje každý rámec zvlášť, pretože pri normalizácii celého signálu môže vzniknúť nezrovnalosť v celom signáli vďaka jednej vysokej či nízkej hodnote, teda normalizácia po rámcoch zachováva dynamický rozsah signálu[9].
- `mfcc()` – extrahuje MFCC koeficienty každého rámcu (`n_mfcc=10` – každý rámec je aproximovaný desiatimi vektormi o premennej dĺžke).
- `fastdtw()` – vykoná DTW algoritmus nad MFC koeficientami narámcovaných signálov za použitia FastDTW algoritmu s obmedzením cesty (`radius=10` – premenná určuje okolie cesty, ktorou algoritmus môže pokračovať, hodnota 10 poskytuje ideálnu kombináciu rýchlosti a presnosti).

[9]Dynamický rozsah – https://en.wikipedia.org/wiki/Dynamic_range

Na pochopenie niektorých súvislostí je potrebné uviesť internú podobu DTW cesty, teda premennej `path`, výpis 4.6.

```
[(0, 0), (1, 1), (2, 2), (3, 2), (4, 2), (5, 2), (6, 2), (7, 2), (8, 3),
(9, 4), (10, 5), (11, 5), (12, 5), (13, 5), (14, 6), (15, 7), (16, 8)]
```

Výpis 4.6: Interná reprezentácia DTW cesty (numpy.array). Vypísaná je DTW cesta medzi signálom o dĺžke 16 s a signálom o dĺžke 8 s. Je zrejmé, že každá dvojica predstavuje zarovnanie jednej sekundy signálov. Zvýraznené skupiny dvojíc reprezentujú body, kde nebolo možné nájsť zarovnanie signálov, teda strih, konkrétne reprezentuje 5 sekundový strih (od druhej do siedmej sekundy) a 3 sekundový strih (od desiatej do trinástej sekundy) v prvom signáli. Sekunda 0 – 1 je nultá sekunda.

### Hľadanie strihov

Súbor `find_cuts.py` implementuje funkciu `find_cuts()`, ktorá sa snaží aproximovať miesta, kde sa vstupné signály nerovnajú, teda aproximuje miesto (čas) a dĺžku strihov pomocou predom získanej DTW cesty. Algoritmus je priblížený vo výpise 4.7.

```
1     def find_cuts(path):
2         # 1. rozdiely v ceste medzi indexami druhého signálu
3         diffs = np.diff(path[:, 1])
4
5         # 2. časy/indexy kedy sa signály prestávajú rovnať - je tu strih
6         change_indices = np.where(diffs == 0)[0] + 1
7
8         # 3. oddelenie jednotlivých strihov od seba
9         real_change = np.where(np.diff(change_indices) > 1)[0] + 1
10        result = np.split(change_indices, real_change)
11        indexes_of_change = [(indexes[-1], indexes.size) for indexes in result]
```

Výpis 4.7: Algoritmus zistenia strihov. Vysvetlenie s použitím cesty z výpisu 4.6 je v postupe nižšie. Nepodstatné časti kódu sú zjednodušené alebo odstránené.

**Postup algoritmu:**

1. Určia sa strihy na základe rozdielu hodnôt prvkov druhého signálu v DTW ceste (je zrejmé, že počas strihu hodnota prvkov druhého signálu stagnuje, teda rozdiely počas strihu budú nulové).

   `[1 1 0 0 0 0 0 1 1 1 0 0 0 1 1 1]`

2. Určia sa indexy (časy) strihov (pričíta sa 1, aby odpovedalo reálnemu strihu).

   `[3 4 5 6 7 11 12 13]`

3. Jednotlivé strihy sa od seba oddelia (riadok 9 – 10) a uloží sa dvojica koniec strihu a dĺžka strihu pre každý strih, ukladá sa dĺžka a nie začiatok strihu, pre signifikantnú nepresnosť určenia začiatku a konca strihu (pričom zistenie dĺžky strihu zostáva presné) pri dlhších signáloch.

   `[[7 5] [13 3]]`

### Misc

Modul Misc označuje pomocné funkcie, ktoré asistujú backendu aj frontendu.

**Funkcie a ich použitie:**

- `convert()` – pomocou nástroja `ffmpeg` konvertuje a podvzorkuje vstupné video – príkazom `ffmpeg -i in.mp4 -ar 8000 in.wav` sa konvertuje MP4 súbor `in.mp4` na WAV súbor `in.wav` podvzorkovaný na 8000 Hz (vzorkovacia frekvencia je uložená v parametri `sr`).
- `get_wav()` – pripraví signál na nákres waveformu v UI.
- `process_sub()` – pripraví titulok na zobrazenie v UI (prevedie do stĺpovej formy).
- `get_speech_segments()` – zostaví `numpy` vektor dvojíc začiatku a konca každého prehovoru, ktorý sa použije na nákres prehovorov vo waveforme v UI.
- `time_convert()` – konverzia `SubRipTime` časového objektu (h, min, sec, ms) na sekundy – potrebné pri riešení prekryvu titulkov so strihom.

Funkcia `convert()` je na backende volaná s parametrom `sr` o hodnote 8000 (Hz), čo je na základe empirických zistení vhodná hodnota na zaručenie rýchlosti aj presnosti algoritmu.

### Nedostatky implementácie backendu

Implementovaný algoritmus pre hľadanie strihov je s narastajúcou dĺžkou vstupných videí a počtom strihov čoraz menej presný (viac v sekcii 5.2).

# Kapitola 5

# Testovanie a analýza výsledkov

Táto kapitola sa venuje testovaniu algoritmu aplikácie, analýze dosiahnutých výsledkov a ich porovnaniu s konkurenciou.

## 5.1 Databáza (zbierka)

Testovanie prebiehalo nad databázou (zbierkou), ktorá pozostáva z public domain (voľných diel) filmov[1] (`films`) a zo záznamov konzultácií (`meets`) s vedúcim tejto práce Ing. Tomášom Miletom, Ph.D. (s jeho povolením). Zozbierané videosúbory (1 – 95min) sú zostrihané na 41 videí rôznych dĺžok, ich názvy odpovedajú ich približnej dĺžke. Videá sú taktiež rozdelené do zložiek podľa ich zdroja (pôvodnej nahrávky) a počtu strihov, ktoré video obsahuje (1 až 4 strihy). Prípona každej nahrávky je `_cut.mp4|wav`.

Mapovanie názvu videa na počet strihov, ktoré video obsahuje:

- 1 strih – 5sec, 7sec, 8sec, 30sec, 90sec, 1min, 2min, 5min, 27min, 45min
- 2 strihy – 150sec, 4min, 6min, 8min,
- 3 strihy – 20sec, 13min, 20min
- 4 strihy – 41min

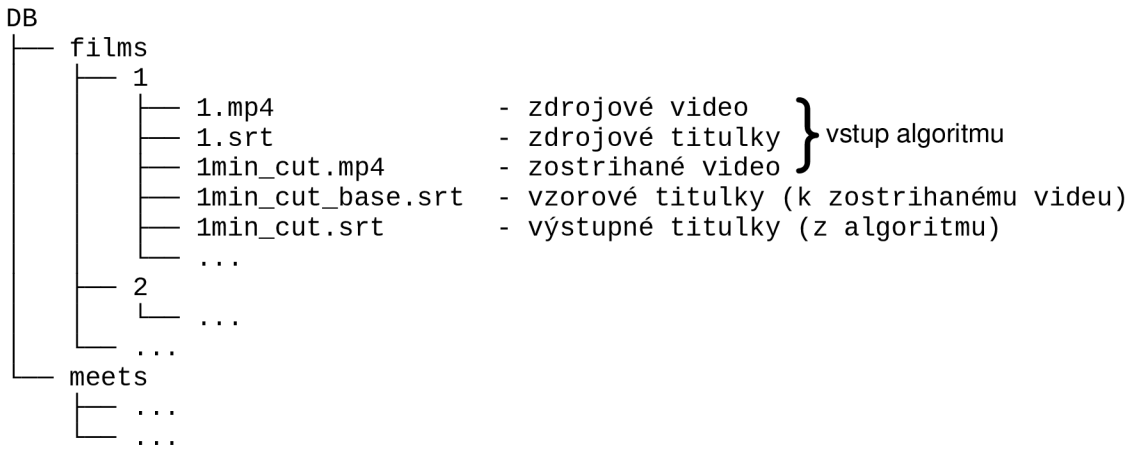
### Štruktúra databázy

Databáza je rozdelená na dva hlavné priečinky `films` a `meets`. V nich sú už podpriečinky obsahujúce samotné videá a titulky. Každý podpriečinok obsahuje zdrojové video s titulkami a jedno až štyri zostrihané videá (zo zdrojového videa) s ich vzorovými titulkami a výstupnými titulkami, ktoré vytvoril nástroj `SubAlign`.

### Formát testovania

Pre každý podpriečinok je zavolaný algoritmus aplikácie na každom zostrihanom videu, zdrojovom videu a zdrojovými titulkami. Výstupom sú titulky vytvorené algoritmom, ktoré sú následne porovnané so vzorovými titulkami – z tohto porovnania vzíde štatistika popísaná nižšie. Štruktúra databázy a popis podpriečinku s kontextom testovania je na obrázku 5.1.

[1] Filmy dostupné z: https://archive.org/details/prelinger




Obr. 5.1: Stromová štruktúra súborov databázy. Obrázok bližšie popisuje podpriečinok, ktorý znázorňuje jednu testovaciu jednotku. Názov zdrojového videa a titulkov sa zhoduje s názvom priečinku. Názov súboru so vzorovými titulkami má vždy príponu `_base.srt`.

## 5.2 Výsledky

Testovanie je zamerané na dve hlavné metriky zarovnania titulkov – **počet** správne zarovnaných titulkov a **presnosť** zarovnania (správny `shift`). Výstupné titulky pre daný súbor (ďalej len `out`) sú porovnané s titulkami vzorovými pre ten istý súbor (ďalej len `base`). Pre všetky výstupné titulky je teda zistený **počet** správne zarovnaných titulkov a **presnosť** správne zarovnaných titulkov.

### Počet (správne zarovnaných titulkov)

Počet správne zarovnaných titulkov je zistený porovnaním indexov titulkov `out` súboru a `base` súboru – pre toto porovnanie je dôležité zachovať indexy titulkov po strihu neupravené (algoritmus rozdelí v .srt súbore titulky v mieste strihu na dve časti, pričom vystrihnutú časť vynechá, čo spôsobí skok v indexoch, obrázok 3.5). Pre každý `out` súbor sa určí úspešnosť zarovnania dvomi údajmi. Jeden údaj je pomer správne zarovnaných indexov (titulkov) k zjednoteniu všetkých indexov referenčného a výstupného súboru – **skóre zarovnania**. Druhý údaj vyjadruje percento správne zarovnaných indexov (pomer prieniku indexov k počtu referenčných indexov) – **správnosť zarovnania**. Štatistiku počtu správne zarovnaných titulkov zobrazujú tabuľky 5.1 a 5.2.

**Príklad** štatistiky zarovnania podľa počtu správne zarovnaných titulkov:

```
base: [1, 2, 3, 8, 9, 10]

out:  [1, 2, 3, 4, 9, 10, 11]

Skóre zarovnania:     62.50 %

Správnosť zarovnania: 83.33 %
```

### Presnosť (správne zarovnaných titulkov)

Kvalitu zarovnania nemožno vyjadriť len počtom správne zarovnaných titulkov ale aj (časovou) **presnosťou** tohoto zarovnania. Konkrétne sa táto metrika sústredí na odchýlku času začiatku správne zarovnaných výstupných titulkov od referenčných titulkov. Príklad vo výpise 5.1. Štatistiku presnosti správne zarovnaných titulkov zobrazuje tabuľka 5.3.

```
BASE:                                   OUT:
1                                       1
00:00:00,160 --> 00:00:06,200           00:00:00,410 --> 00:00:06,450
Uh, ja som bol v~tom zahranici,         Uh, ja som bol v~tom zahranici,

2                                       2
00:00:06,600 --> 00:00:07,100           00:00:06,850 --> 00:00:07,350
Ano.                                    Ano.
```

Výpis 5.1: Príklad určenia presnoti zarovnania. V tomto prípade by odchýlky vyzerali nasledovne: (0.25, 0.25). Berú sa do úvahy iba správne zarovnané titulky (teda prienik `base` a `out` súboru). Titulky môžu meškať (kladná odchýlka) alebo predbiehať (záporná odchýlka), preto sa počíta s absolútnou hodnotou odchýlky.

| *Zbierka* | Skóre zarovnania | Správnosť zarovnania |
|---|---|---|
| *meets* | 69.34 % | 75.98 % |
| *films* | 77.42 % | 79.89 % |
| **Priemer** | **73.38 %** | **77.94 %** |

Tabuľka 5.1: Tabuľka štatistiky počtu správne zarovnaných titulkov databázy. V databáze sa vyskytlo päť zarovnaní s 0 % úspešnosťou, preto tabuľka 5.2 vynecháva tieto výskyty.

| *Zbierka* | Skóre zarovnania (bez 0 %) | Správnosť zarovnania (bez 0 %) |
|---|---|---|
| ***meets*** | 78.01 % | 85.48 % |
| ***films*** | 89.03 % | 91.88 % |
| **Priemer** | **83.52 %** | **88.68 %** |

Tabuľka 5.2: Tabuľka štatistiky počtu správne zarovnaných titulkov databázy s vynechanými výskytmi 0 % zarovnaní.

| | Počet titulkov s odchýlkami (s) | | | | | |
|---|---|---|---|---|---|---|
| Zbierka | 0 | $> 0$ a $\leq 0.5$ | $> 0.5$ a $\leq 1$ | $> 1$ a $\leq 2$ | $> 2$ | **Ø odchýlka** |
| *meets* | 133 | 63 | 38 | 210 | 70 | 0.86 s |
| *films* | 810 | 378/365 | 277/272 | 64/61 | 629/232 | 1.10/0.36 s |
| **Celkovo** | 943 | 441/428 | 305/300 | 274/271 | 699/302 | **0.98/0.61 s** |

Tabuľka 5.3: Tabuľka štatistiky presnosti správne zarovnaných titulkov databázy. Telo tabuľky je rozdelené do piatich kategórii podľa rôznej odchýlky titulkov. V databáze je jeden výskyt priemernej odchýlky začiatku titulkov v súbore väčší ako 10 s (13.71 s), niektoré údaje na riadku *films* a *Celkovo* reflektujú vynechanie tohto výskytu lomkou (x/x). Tabuľky kompletnej štatistiky testovania aplikácie nájdete v prílohe B.

#### Zhodnotenie výsledkov

Z výsledkov možno usúdiť, že nepresnosť algoritmu rastie s počtom strihov a zároveň s dĺžkou pôvodného videa a zostrihaného videa – túto skutočnosť podporuje fakt, že video s najväčším počtom strihov (4) má zároveň 41 minút (`films/41min_cut`). Nutne však dodať, že 2/5 výskytov s 0 % úspešnosťou zarovnania sú videá `meets/5sec_cut` a `meets/7sec_cut` vystrihnuté z videa o dĺžke 40 sekúnd – z toho vyplýva, že algoritmus má problém so zarovnaním moc krátkeho vzorku (to môže byť spôsobené tendenciou DTW algoritmu zarovnať každý prvok jedného signálu na prvok druhého signálu). Problém s nepresnosťou určenia strihu pri rastúcej dĺžke videa je pravdepodobne spôsobený kompresiou zvuku MFCC koeficientami a následnou nepresnosťou DTW algoritmu. Mohol by byť vyriešený druhou fázou algoritmu, ktorá by sa sústredila na nájdené strihy a na väčšom rozlíšení by bol na týchto segmentoch použitý rovnaký algoritmus na presnejšie zistenie strihov.

## 5.3 Porovnanie s konkurenciou

Táto sekcia sa venuje krátkemu porovnaniu presnosti zarovnania/synchronizácie titulkov výslednej aplikácie, Python nástroja `subaligner` a nástroja `Aeneas`. Bližší popis konkurenčných nástrojov je v sekcii 1.2. Na porovnanie sa vybrala zložka `films/3/sita-2/...` obsahujúca 4-minutovú zdrojovú nahrávku a titulky (`sita-2.mp4` a `sita-2.srt`) a 3 zostrihané videá s referenčnými titulkami (`1min_cut`, `90sec_cut` a `150sec_cut`).

Pri nástroji `subaligner` boli využité oba základné typy zarovnania – single-stage (globálne) a dual-stage (lokálne) a taktiež bolo použité transkribovanie (ASR, automatic speech recongnition) videí pomocou predtrénovaného modelu `whisper`[2]. Oba základné typy zarovnania sa podľa výsledkov môžu považovať za zaostávajúce pri zarovnávaní titulkov na zostrihané video, z dôvodu neschopnosti týchto zarovnaní skrátiť výsledný súbor titulkov, čoho výsledkom sú titulky, ktoré môžu byť v niektorej sekcii správne zarovnané, ale celkový výsledok je nedostatočný. Použitie ASR (napr. modelu `whisper`) je najpresnejšie a technologicky najvyspelejšie riešenie automatického zarovnania titulkov, avšak vyžaduje značné množstvo výpočtových zdrojov (overhead[3]), čo sa mitguje využitím paralelizmu a GPU.

Keďže nástroj `Aeneas` vyžaduje na zarovnanie presný prepis reči z videa (bez textu, ktorý sa vo videu rečou nevyskytne), boli zdrojové titulky upravené pre každý súbor tak, aby ho obsahovali. Tento úkon pridáva manuálnu intervenciu do procesu a obetuje mieru automatizácie, ale výsledky sú pre zaujímavosť uvedené.

Príkazy spustenia:

- **single-stage/dual-stage** – subaligner -m single/dual -v xx_cut.mp4 -s sita-2.srt -o xx_cut.srt
- **ASR (whisper)** – subaligner -m transcribe -v xx_cut.mp4 -ml eng -mr whisper -mf small -o xx_cut.srt
- **Aeneas** – python -m aeneas.tools.execute_task xx_cut.mp4 xx_cut.txt xx_cut.srt

Kompletný popis výsledkov je zobrazený tabuľkami 5.4 a 5.5.

[2]Informácie k modelu whisper – dostupné z: https://huggingface.co/openai/whisper-medium

[3]overhead – dodatočné výpočtové zaťaženie systému spôsobené rôznymi procesmi, ktoré nie sú priamo súčasťou hlavného problému

| Typ\súbor | 1min | 90sec | 150sec |
|---|---|---|---|
| single-stage | 32.26 % | 30.35 % | 62.90 % |
| dual-stage | 32.26 % | 30.65 % | 62.90 % |
| ASR (whisper) | 99 % | 98 % | 99 % |
| Aeneas | 100 % | 100 % | 100 % |
| **vlastný nástroj** | **95 %** | **85 %** | **100 %** |

Tabuľka 5.4: Tabuľka porovnávajúca počet správne zarovnaných titulkov rôznych prístupov. Vo všetkých bunkách sa uvádza **skóre zarovnania**. Nástroj `Aeneas` má na základe úpravy popísanej vyššie prirodzene skóre 100 %. Pri single a dual-stage zarovnaniach má dlhšie video väčšie skóre len vďaka väčšiemu počtu titulkov. Pri ASR (`whisper`) trankribovaní sa niektoré titulky rozdelili na viacero častí, alebo naopak spojili (oproti referenčným titulkom) takže tu percento vyjadruje textovú podobnosť titulkov s referenčným súborom.

| | Priemerná odchýlka (s) | | |
|---|---|---|---|
| Typ/súbor | 1min | 90sec | 150sec |
| single-stage | 0.00 | 119.84 | 29.15 |
| dual-stage | 0.95 | 118.89 | 29.18 |
| Aeneas | 0.66 | 0.50 | 0.42 |
| **vlastná nástroj** | **0.00** | **0.00** | **0.47** |

Tabuľka 5.5: Tabuľka porovnávajúca presnosť správne zarovnaných titulkov rôznych prístupov. Vo všetkých bunkách sa uvádza priemerná odchýlka začiatkov titulkov. Z tabuľky je zrejmé, že single a dual-stage zarovnania nástroja `subalign` nie sú vhodné na daný typ zarovnávania titulkov. Presnosť nástroja `Aeneas` je prirodzene vysoká. Transkribovanie pomocou modelu `whisper` v tabuľke nie je uvedené z dôvodu uvedeného v popise tabuľky 5.4, ale presnosť titulkov na výstupe ASR je ideálna.

# Kapitola 6

# Záver

Cieľom tejto práce bolo vytvoriť nástroj na automatické prečasovanie titulkov na inú verziu videosúboru. Výsledná aplikácia poskytuje intuitívne užívateľské prostredie, v ktorom je možné nahrať vstupné súbory a vytvoreným algoritmom zarovnať titulky.

Text práce je venovaný popisu rôznych formátov titulkov, problémom so spracovaním multimediálnych súborov a množstvu algoritmov či metód súvisiacich so spracovaním (audio) signálov a zistením ich podobnosti či časovým zarovnaním signálov ako DTW algoritmus, MFCC, vzájomná korelácia signálov (ang. cross-correlation), DFT (FFT) algoritmus, prevzorkovanie a podvzorkovanie digitálneho zvuku a jeho grafické zachytenie pomocou waveformu. Na účel testovania aplikácie bol zozbieraný dataset rôzne zostrihaných videosúborov s príslušnými titulkami, ktoré vychádzajú z public domain (voľne dostupných diel) filmov, reklám alebo vlastných videonahrávok. Z testovania vzišla výsledná štatistika kvality zarovnania a jej porovnanie s existujúcimi nástrojmi. K výslednej aplikácii bolo natočené aj krátke demonštračné video[1].

Výsledný algoritmus aplikácie má pri zarovnávaní titulkov úspešnosť zarovnania v rozmedzí 69 až 90 % s priemernou odchýlkou presnosti času zarovnaných titulkov približne 1 sekunda (testovanie prebiehalo na 41 rôznych videosúboroch). Hlavným nedostatkom algoritmu je nepresnosť času zarovnaných titulkov, ktorá narastá s dĺžkou videa a počtom rôznych strihov. Nepríjemnosťou v rámci užívateľského prostredia je skutočnosť, že knižnica použitá na videoprehrávač nepodporuje prehrávanie zvuku a pretáčanie pri niektorých videách spôsobí zrýchlenie prehrávania. Napriek nedostatkom ponúka aplikácia rýchle, technologicky a výpočtovo nenáročné riešenie automatickej synchronizácie titulkov s ľubovolnou verziou videosúboru.

Rozšírenia a plány do budúcna zahŕňajú zmenu GUI knižnice Tkinter z dôvodu narastajúcej komplexnosti UI (napr. na GTK3+) a zmenu knižnice na prehrávanie videa tkVideoPlayer (napr. na libvlc). Zaujímavou funkcionalitou aplikácie by bolo aj zobrazenie titulkov priamo vo videu. Ďalej by bolo vhodné optimalizovať aplikáciu zvýšením kompresie vstupu (aplikácia je najmä pri dlhšom vstupnom videosúbore náročná na spotrebu RAM – až 1 GB). Prínosná by taktiež bola revízia algoritmu vo forme podpory časovej nelineárnosti scén vo vstupných videách a spresnenia určenia strihov druhou fázou algoritmu (sústredenie druhej fázy len na segmenty signálu, ktoré obsahujú strih na väčšom rozlíšení). Na spoľahlivé zistenie presnosti aplikácie by bolo ideálne rozšíriť existujúcu databázu rôzne zostrihaných videosúborov a otestovať aplikáciu na tejto databáze.

[1] Demonštračné video – dostupné z: https://youtu.be/SSfNTlh1kRo.

# Literatúra


[1] Becek, K. Comparison of Decimation and Averaging Methods of DEM's Resampling. Unpublished. 2007. DOI: 10.13140/2.1.1147.8088. Dostupné z: http://rgdoi.net/10.13140/2.1.1147.8088.

[2] Blackledge, J. M. In: *Digital image processing*. Chichester, England: Horwood Publishing, November 2005, s. 44–45. Woodhead Publishing Series in Electronic and Optical Materials. ISBN 1-898563-49-7.

[3] C.A. Bulterman, D., Jansen, J., Cesar, P. a Cruz Lara, S. An Efficient, Streamable Text Format for Multimedia Captions and Subtitles. In: *ACM Symposium on Document Engineering - DocEng 2007*. Winnipeg, Canada: [b.n.], August 2007. Dostupné z: https://inria.hal.science/inria-00192467.

[4] Carlson, A. B. a Crilly, P. *Communication Systems*. 5. vyd. New York, NY: McGraw-Hill Professional, február 2009. ISBN 978-0073380407.

[5] Chatzou, M., Magis, C., Chang, J.-M., Kemena, C., Bussotti, G. et al. Multiple sequence alignment modeling: methods and applications. *Briefings in Bioinformatics*. November 2015, zv. 17, č. 6, s. 1009–1023. DOI: 10.1093/bib/bbv099. ISSN 1467-5463. Dostupné z: https://doi.org/10.1093/bib/bbv099.

[6] Chu, S., Keogh, E., Hart, D. a Pazzani, M. Iterative Deepening Dynamic Time Warping for Time Series. In: *Proceedings of the 2002 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, Apríl 2002, s. 5–6. DOI: 10.1137/1.9781611972726.12. Dostupné z: https://doi.org/10.1137/1.9781611972726.12.

[7] Cintas, J. a Remael, A. *Subtitling: Concepts and Practices*. Routledge, 2021. Translation practices explained. ISBN 9781138940543. Dostupné z: https://books.google.sk/books?id=KAjBzQEACAAJ.

[8] Crecraft, D. a Gorham, D. *Electronics*. 2. vyd. London, England: CRC Press, október 2018. 47 – 48 s. ISBN 9781482268416.

[9] Crochiere, R. a Rabiner, L. *Multirate Digital Signal Processing*. Prentice-Hall, 1983. 131 – 140 s. Prentice-Hall Signal Processing Series: Advanced monographs. ISBN 9780136051626. Dostupné z: https://books.google.sk/books?id=X_NSAAAAMAAJ.

[10] Elliott, D. F., Harris, F. J. et al. In: Elliott, D. F., ed. *Handbook of Digital Signal Processing*. Academic Press, 1987, s. 667 – 668. DOI: 10.1016/c2009-0-21739-9. ISBN 0-12-237075-9. Dostupné z: https://doi.org/10.1016/c2009-0-21739-9.

[11] Giannakopoulos, T. a Pikrakis, A. Chapter 3.1 - The Discrete Fourier Transform. In: Giannakopoulos, T. a Pikrakis, A., ed. *Introduction to Audio Analysis*. Oxford: Academic Press, 2014, s. 33–39. DOI: https://doi.org/10.1016/B978-0-08-099388-1.00003-0. ISBN 978-0-08-099388-1. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780080993881000030.

[12] Kabari, L. G. a Chigoziri, M. B. Speech Recognition Using MATLAB and Cross-Correlation Technique. *European Journal of Engineering and Technology Research*. European Open Science Publishing. august 2019, zv. 4, č. 8, s. 1 – 3. DOI: 10.24018/ejeng.2019.4.8.1437.

[13] Lahreche, A. a Boucheham, B. A Comparison Study of Dynamic Time Warping's Variants for Time Series Classification. *International Journal of Informatics and Applied Mathematics*. International Society of Academicians. 2021, zv. 4, č. 1, s. 56 – 71.

[14] McFee, B., Metsai, A. et al. *Librosa/librosa: 0.9.2*. 2022. DOI: 10.5281/zenodo.6759664. Dostupné z: https://doi.org/10.5281/zenodo.6759664.

[15] Müller, M. *Information retrieval for music and motion*. 1. vyd. Springer, 2007. 69–84 s. ISBN 978-3-540-74048-3.

[16] Oppenheim, A. V., Schafer, R. W. et al. *Discrete-time signal processing*. 2. vyd. Upper Saddle River, NJ: Pearson, december 1998. ISBN 0-13-754920-2.

[17] Oppenheim, A. V. a Willsky, A. S. Continuous-time and discrete-time signals. In: Horton, M., ed. *Signals and Systems*. 2. vyd. Upper Saddle River, NJ: Pearson, August 1996, kap. 1.1. ISBN 0-13-814757-4.

[18] Pan, D. Y. Digital Audio Compression. *Digital Tech. J.* USA: Digital Equipment Corp. mar 1993, zv. 5, č. 2, s. 28–40. ISSN 0898-901X.

[19] Parwinder Pal Singh, P. R. An Approach to Extract Feature using MFCC. *IOSR Journal of Engineering*. IOSR Journals. august 2014, zv. 4, č. 8, s. 21–25. DOI: 10.9790/3021-04812125. Dostupné z: https://doi.org/10.9790/3021-04812125.

[20] Rao, K. S. a E, M. K. In: *Speech Recognition Using Articulatory and Excitation Source Features*. Springer International Publishing, 2017, s. 85–88. DOI: 10.1007/978-3-319-49220-9. Dostupné z: https://doi.org/10.1007/978-3-319-49220-9.

[21] Saabni, R. a El Sana, J. Keyword searching for Arabic handwriting. In: *Proceedings of the International conference of frontiers in Handwriting recognition (ICFHR) 2008*. 2008, s. 1185–1195.

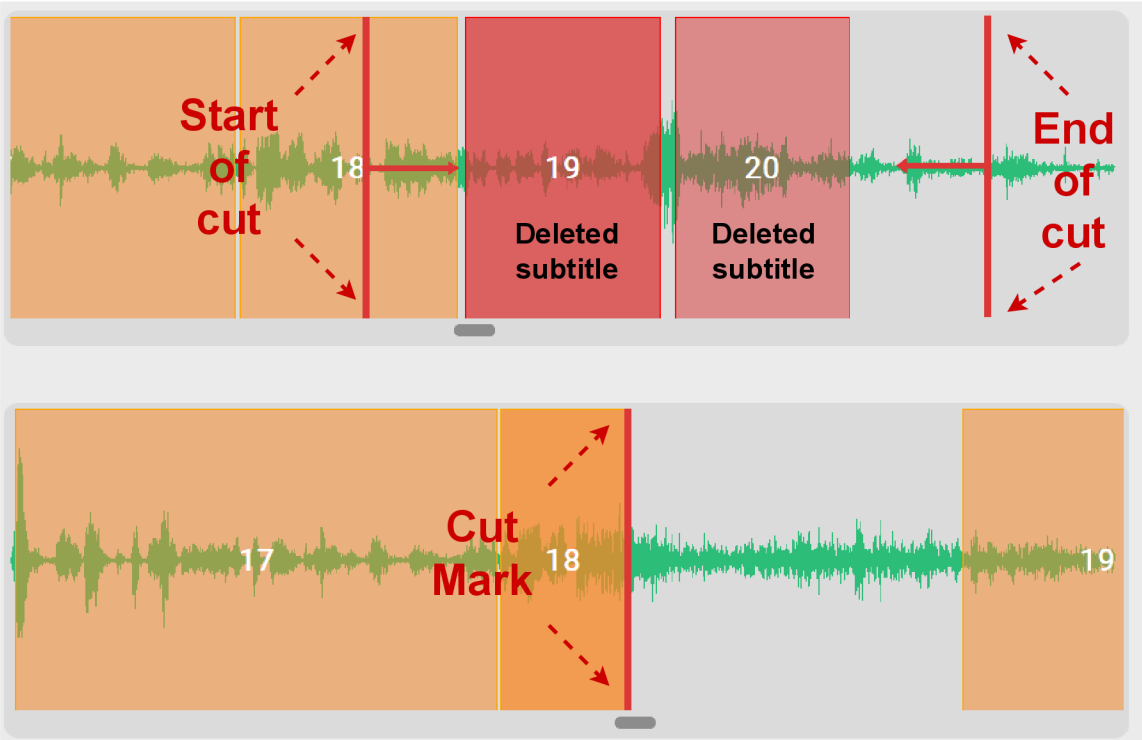
[22] Salvador, S. a Chan, P. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*. IOS Press. október 2007, zv. 11, č. 5, s. 561–580. DOI: 10.3233/ida-2007-11508. Dostupné z: https://doi.org/10.3233/ida-2007-11508.

[23] Sedha, R. S. In: *Analog Communication*. New Delhi, India: S Chand, Júl 2014, s. 86. ISBN 978-9383746743.

[24] Venu, D. a Rao, N. K. A cross-correlation approach to determine target range in passive radar using FM Broadcast Signals. In: *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. 2016, s. 524–529. DOI: 10.1109/WiSPNET.2016.7566190.

# Príloha A

# Vysvetlenie komponentov UI




Obr. A.1: Obrázok vysvetľuje objekty vo waveforme, ktoré symbolizujú strih. Červené obdĺžniky so šípkami predstavujú začiatok a koniec strihu v prvom waveforme, takisto červený obdĺžnik v druhom waveforme označuje strih. Vymazané prehovory (titulky) sú červené (19, 20).

# Príloha B

# Kompletná štatistika testovania

| ***Films*** | | |
|---|---|---|
| Názov súboru/Typ štatistiky | Skóre zarovnania | Správnosť zarovnania |
| 1\1min | 69.23 % | 69.23 % |
| 1\41min | 91.07 % | 95.22 % |
| 1\45min | 99.82 % | 99.82 % |
| 1\6min | 30.23 % | 59.09 % |
| 1_1\5min | 100.00 % | 100.00 % |
| 2\20min | 97.52 % | 98.74 % |
| 2\2min | 0.00 % | 0.00 % |
| 2\8min | 100.00 % | 100.00 % |
| 4\1min | 0.00 % | 0.00 % |
| 4\41min | 98.02 % | 98.41 % |
| 4\45min | 93.59 % | 97.33 % |
| 4\6min | 0.00 % | 0.00 % |
| 5\2min | 100.00 % | 100.00 % |
| 5\8min | 98.26 % | 98.83 % |
| 3\sita-1\150sec | 89.29 % | 89.29 % |
| 3\sita-1\1min | 90.00 % | 90.00 % |
| 3\sita-1\90sec | 75.76 % | 89.29 % |
| 3\sita-2\150sec | 100.00 % | 100.00 % |
| 3\sita-2\1min | 95.00 % | 95.00 % |
| 3\sita-2\90sec | 85.00 % | 89.47 % |
| 3\sita-3\150sec | 88.89 % | 88.89 % |
| 3\sita-3\1min | 90.00 % | 90.00 % |
| 3\sita-3\90sec | 88.89 % | 88.89 % |
| **Priemer** | **77.42 %** | **79.89 %** |
| **Priemer (bez 0 %)** | **89.03 %** | **91.88 %** |

Tabuľka B.1: Tabuľka kompletnej štatistiky počtu správne zarovnaných titulkov zbierky ***Films***.

| ***Meets*** | | |
|---|---|---|
| Názov súboru/Typ štatistiky | Skóre zarovnania | Správnosť zarovnania |
| 1\20sec | 50.00 % | 66.67 % |
| 1\5sec | 0.00 % | 0.00 % |
| 1\7sec | 0.00 % | 0.00 % |
| 1\8sec | 50.00 % | 100.00 % |
| 3-1\150sec | 100.00 % | 100.00 % |
| 3-1\1min | 100.00 % | 100.00 % |
| 3-1\90sec | 100.00 % | 100.00 % |
| 3-2\150sec | 87.10 % | 93.10 % |
| 3-2\1min | 53.85 % | 70.00 % |
| 3-2\90sec | 13.79 % | 21.05 % |
| 5\20min | 97.25 % | 97.25 % |
| 5\2min | 100.00 % | 100.00 % |
| 5\8min | 94.32 % | 94.32 % |
| 6\13min | 71.24 % | 78.99 % |
| 6\27min | 100.00 % | 100.00 % |
| 6\4min | 30.65 % | 46.34 % |
| 7\30sec | 100.00 % | 100.00 % |
| 8\30sec | 100.00 % | 100.00 % |
| **Priemer** | **69.34 %** | **75.98 %** |
| **Priemer (bez 0 %)** | **78.01 %** | **85.48 %** |

Tabuľka B.2: Tabuľka kompletnej štatistiky počtu správne zarovnaných titulkov zbierky ***Meets***.

| ***Films*** | | | | | | |
|---|---|---|---|---|---|---|
| | Počet titulkov s odchýlkami (s) | | | | | |
| Názov súboru | 0 | $> 0$ a $\leq 0.5$ | $> 0.5$ a $\leq 1$ | $> 1$ a $\leq 2$ | $> 2$ | Ø odchýlka (s) |
| 1\1min | 9 | 0 | 0 | 0 | 0 | 0.00 |
| 1\41min | 0 | 13 | 5 | 3 | 397 | 13.71 |
| 1\45min | 372 | 182 | 0 | 0 | 0 | 0.08 |
| 1\6min | 0 | 13 | 0 | 0 | 0 | 0.25 |
| 1_1\5min | 41 | 0 | 0 | 0 | 0 | 0.00 |
| 2\20min | 156 | 1 | 0 | 0 | 0 | 0.00 |
| 2\2min | 0 | 0 | 0 | 0 | 0 | - |
| 2\8min | 73 | 1 | 0 | 0 | 0 | 0.00 |
| 4\1min | 0 | 0 | 0 | 0 | 0 | - |
| 4\41min | 2 | 117 | 128 | 0 | 0 | 0.63 |
| 4\45min | 0 | 0 | 0 | 60 | 232 | 3.27 |
| 4\6min | 0 | 0 | 0 | 0 | 0 | - |
| 5\2min | 42 | 1 | 0 | 0 | 0 | 0.01 |
| 5\8min | 42 | 0 | 126 | 1 | 0 | 0.75 |
| 3\sita-1\150sec | 0 | 25 | 0 | 0 | 0 | 0.25 |
| 3\sita-1\1min | 0 | 0 | 0 | 0 | 0 | - |
| 3\sita-1\90sec | 9 | 16 | 0 | 0 | 0 | 0.16 |
| 3\sita-2\150sec | 20 | 1 | 18 | 0 | 0 | 0.47 |
| 3\sita-2\1min | 19 | 0 | 0 | 0 | 0 | 0.00 |
| 3\sita-2\90sec | 17 | 0 | 0 | 0 | 0 | 0.00 |
| 3\sita-3\150sec | 0 | 8 | 0 | 0 | 0 | 0.25 |
| 3\sita-3\1min | 0 | 0 | 0 | 0 | 0 | - |
| 3\sita-3\90sec | 8 | 0 | 0 | 0 | 0 | 0.00 |
| **Celkovo** | **810** | **378** | **277** | **64** | **629** | **1.10** |

Tabuľka B.3: Tabuľka kompletnej štatistiky presnosti správne zarovnaných titulkov zbierky ***Films***. Bunky označené spojovníkom (-) znamenajú, že daný súbor mal nulový počet správne zarovnaných titulkov alebo výsledkom zarovnania je prázdny súbor.

| ***Meets*** | | | | | | |
|---|---|---|---|---|---|---|
| | Počet titulkov s odchýlkami (s) | | | | | |
| Názov súboru | 0 | $> 0$ a $\leq 0.5$ | $> 0.5$ a $\leq 1$ | $> 1$ a $\leq 2$ | $> 2$ | Ø odchýlka (s) |
| 1\20sec | 0 | 0 | 0 | 1 | 1 | 2.42 |
| 1\5sec | 0 | 0 | 0 | 0 | 0 | - |
| 1\7sec | 0 | 0 | 0 | 0 | 0 | - |
| 1\8sec | 0 | 0 | 1 | 0 | 0 | 0.88 |
| 3-1\150sec | 10 | 1 | 0 | 14 | 0 | 0.58 |
| 3-1\1min | 10 | 0 | 0 | 0 | 0 | 0.00 |
| 3-1\90sec | 0 | 14 | 1 | 0 | 0 | 0.40 |
| 3-2\150sec | 10 | 0 | 0 | 0 | 17 | 2.09 |
| 3-2\1min | 7 | 0 | 0 | 0 | 0 | 0.00 |
| 3-2\90sec | 0 | 0 | 0 | 0 | 4 | 2.25 |
| 5\20min | 22 | 0 | 0 | 83 | 72 | 1.46 |
| 5\2min | 22 | 0 | 0 | 0 | 0 | 0.00 |
| 5\8min | 22 | 0 | 0 | 61 | 0 | 0.96 |
| 6\13min | 13 | 33 | 5 | 9 | 49 | 2.47 |
| 6\27min | 2 | 0 | 0 | 0 | 0 | 0.00 |
| 6\4min | 4 | 15 | 0 | 0 | 0 | 0.20 |
| 7\30sec | 5 | 0 | 0 | 0 | 0 | 0.00 |
| 8\30sec | 6 | 0 | 0 | 0 | 0 | 0.00 |
| **Celkovo** | **133** | **63** | **38** | **210** | **70** | **0.86** |

Tabuľka B.4: Tabuľka kompletnej štatistiky presnosti správne zarovnaných titulkov zbierky ***Meets***. Bunky označené spojovníkom (-) znamenajú, že daný súbor mal nulový počet správne zarovnaných titulkov alebo výsledkom zarovnania je prázdny súbor.