

**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

**ODMAZÁVÁNÍ TITULKŮ Z VIDEA**  
HARDSUB REMOVER

**BAKALÁŘSKÁ PRÁCE**  
BACHELOR'S THESIS

**AUTOR PRÁCE**  
AUTHOR

**VEDOUCÍ PRÁCE**  
SUPERVISOR

**EMMA KROMPAŠČÍKOVÁ**

**Ing. TOMÁŠ MILET, Ph.D.**

**BRNO 2023**

## Zadání bakalářské práce



144145

Ústav: Ústav počítačové grafiky a multimédií (UPGM)  
Studentka: **Krompaščíková Emma**  
Program: Informační technologie  
Specializace: Informační technologie  
Název: **Odmazávání titulků z videa**  
Kategorie: Počítačová grafika  
Akademický rok: 2022/23

### Zadání:

1. Nastudujte knihovny pro práci s videem a techniky odmazávání částí videa. Prozkoumejte stávající řešení pro odmazávání částí videa.
2. Navrhněte aplikaci sloužící pro detekci a odmazávání titulků.
3. Implementujte navrženou aplikaci.
4. Proměňte a sepište závěry.
5. Vytvořte demonstrační video a aplikaci zveřejněte.

### Literatura:

- Chun Yang, el. al.: Chinese text-line detection from web videos with fully convolutional networks, Big Data Analytics (2018)

Při obhajobě semestrální části projektu je požadováno:  
Body 1 a 2 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: 10.5.2023

Datum schválení: 31.10.2022

## Abstrakt

Cieľom tejto práce je navrhnúť techniky odmazávania pevne vstavaných titulkov z videa a následne ich implementovať vo forme desktopovej aplikácie. Práca popisuje a implementuje štyri techniky odmazávania titulkov zo snímky – dve sú založené na inpaintingu a ďalšie dve využívajú obrazové filtre na rozmazanie vybranej oblasti. Je popísaná a implementovaná optimalizovaná metóda na detekciu textu vo videu s využitím bisekcie, ktorá umožňuje výrazne skrátiť čas spracovania oproti detekcii textu na každom snímku. Je využívaná knižnica Keras-OCR na detekciu textu a knižnica OpenCV na jeho zamazanie. Desktopová aplikácia má používateľské rozhranie vytvorené pomocou knižnice Electron, spracovanie obrazu je vykonávané pomocou skriptu v jazyku Python.

## Abstract

The goal of this thesis is to propose techniques for removing hardcoded subtitles from video and then implement them in the form of a desktop application. The thesis describes and implements four techniques for removing subtitles from an image – two are based on inpainting and other two use image filters to blur the selected area. An optimized method for detecting text in video is described and implemented using bisection, which enables the reduction of the processing time compared to the detection of text on each frame. The library Keras-OCR is used for text detection and the OpenCV library for its removing. Desktop app has a user interface built using the Electron library, image processing is executed using a Python script.

## Klúčové slová

vstavané titulky, odmazávanie titulkov, rozpoznávanie textu, rekonštrukcia obrazu

## Keywords

hardsubs, subtitles removing, text detection, image inpainting

## Citácia

KROMPAŠČÍKOVÁ, Emma. *Odmazávání titulků z videa*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

# Odmazávání titulků z videa

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Tomáša Mileta, Ph.D. Uviedla som všetky literárne pramene, publikácie a dalšie zdroje, z ktorých som čerpala.

.....  
Emma Krompaščíková  
9. mája 2023

## Poděkování

Chcela by som sa poděkovat mýmu vedúcemu práce pánu Ing. Tomášovi Miletovi, Ph.D. za venovaný čas a odbornú pomoc.

# Obsah

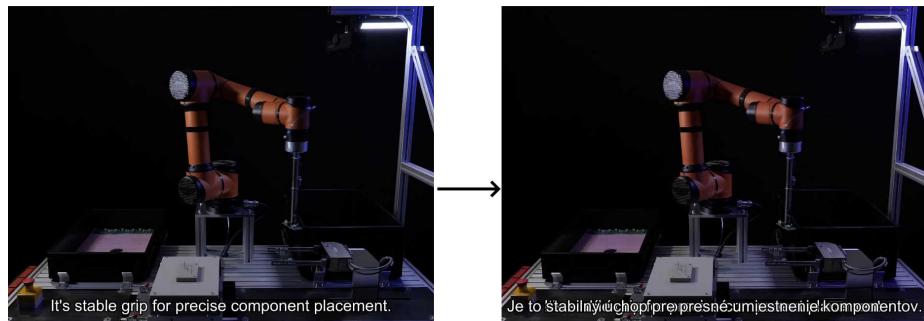
<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Existujúce riešenia</b>	<b>3</b>
<b>3</b>	<b>Teoretický základ</b>	<b>7</b>
3.1	Titulky . . . . .	7
3.2	Detekcia textu . . . . .	9
3.3	Zamazávanie . . . . .	9
3.4	Prahovanie . . . . .	12
3.5	Erózia a dilatácia . . . . .	13
3.6	Bisekcia . . . . .	14
<b>4</b>	<b>Funkcionálny popis riešenia</b>	<b>16</b>
4.1	Backend . . . . .	17
4.2	Frontend . . . . .	20
<b>5</b>	<b>Realizácia funkcionálneho návrhu</b>	<b>25</b>
5.1	Electron aplikácia . . . . .	25
5.2	Použité knižnice jazyka Python . . . . .	28
5.3	Hľadanie hraníc titulkov vo videu . . . . .	30
<b>6</b>	<b>Experimenty a zhodnotenie</b>	<b>32</b>
6.1	Doba trvania odmazávania . . . . .	32
6.2	Kvalita odmazávania . . . . .	36
<b>7</b>	<b>Záver</b>	<b>39</b>
	<b>Literatúra</b>	<b>40</b>
<b>A</b>	<b>Obsah priloženého pamäťového média</b>	<b>42</b>

# Kapitola 1

## Úvod

V súčasnej dobe je sledovanie filmov, prípadne seriálov, zálubou mnohých ľudí. Krátia si tým čas pri cestovaní, rozširujú si obzory, či jednoducho unikajú pred realitou. Pokial je dabing filmu v inom jazyku, než ktorému sledovateľ rozumie, prichádzajú na rad titulky. Titulky sú písané slová, ktoré bud prepisujú dialóg, alebo doplňujú informácie o aktuálnej scéne do čitateľnej a zrozumiteľnej formy vo filmoch, videohrách alebo v inej forme vizuálneho média. Sú využívané najmä na preklad reči z cudzieho jazyka. Okrem toho často pomáhajú ľuďom, ktorí si chcú obohatiť svoju slovnú zásobu alebo tým, ktorí sa chcú zlepšiť v cudzom jazyku. Titulky taktiež podporujú dostupnosť obsahu ľuďom, ktorí sú hluchí, alebo trpia inou sluchovou poruchou.

Bakalárska práca sa zaoberá odmazávaním pevne vložených titulkov, ktoré používateľ vo videu mať nechce, či už z estetických alebo praktických dôvodov. Je to problémom práve vtedy, keď sú titulky v cudzom jazyku pevne vstavané, ale používateľ chce pridať titulky v jazyku, ktorému rozumie. Dochádza k prekrytiu a text je ľažko čitateľný (obr. 1.1). Okrem toho sa môžu v titulkoch vyskytnúť napríklad gramatické chyby, ktoré sa nedajú opraviť.



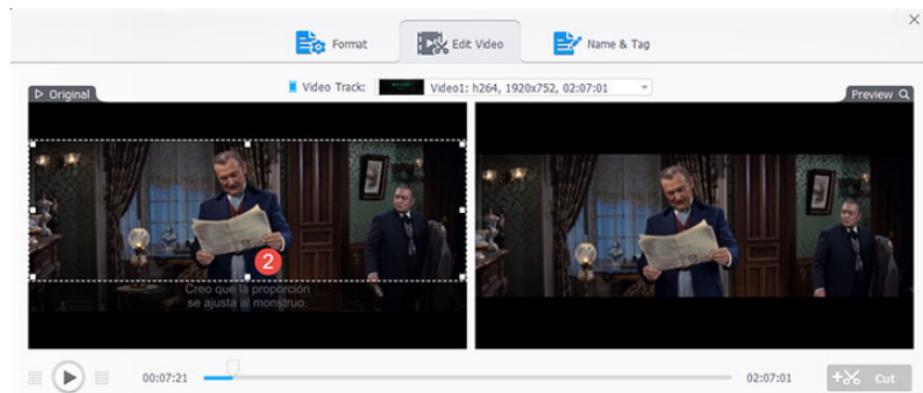
Obr. 1.1: Obrázok zobrazuje prekrytie titulkov. Na ľavom obrázku sú pôvodné titulky, na pravom sú cez pôvodné titulky pridané titulky v inom jazyku. Je to nežiadúce a náročne čitateľné.

Proces odstraňovania titulkov je však náročný na implementáciu, pretože vyžaduje kombináciu niekolkých technológií, konkrétnie na dekódovanie videa; detekciu, v ktorej časti obrazu sa vyskytujú nejaké textové informácie; či schopnosť zamazávať časť snímky. Každá z týchto podúloh nachádza riešenia v technológiách, ako napríklad knižnica OpenCV, ktorá je schopná dekódovať video a ktorá dokáže s pomocou masky zamazat vybranú oblasť snímky. Text na snímkach môže byť rozpoznávaný napríklad nástrojom Keras-OCR.

## Kapitola 2

# Existujúce riešenia

V tejto kapitole sú popísané dostupné konkurenčné nástroje. Funkčnosť všetkých aplikácií je testovaná na rovnakom videu, konkrétnie na snímku naľavo z obr. 1.1. V poslednej časti kapitoly je porovnanie môjho riešenia s ostatnými. Prekvapivo, mnoho nájdených riešení odporúčalo jednoducho video veľkostne orezať tak, aby titulky neboli viditeľné (obr. 2.1). Tým sa však zmenší rozmer videa, či sa orežú potenciálne dôležité časti obrazu, a preto je toto riešenie nepostačujúce.

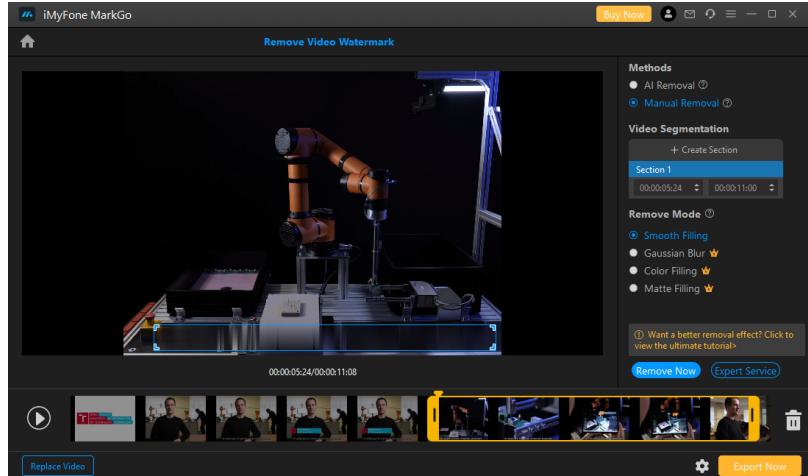


Obr. 2.1: Snímka obrazovky aplikácie z webového článku o odstránení titulkov [13] poskytuje ako riešenie zmenšenie rozmeru videa orezaním. Riešenie nie je dostačujúce, pretože orezaním sa môžu stratíť významné časti videa.

### iMyFone MarkGo

Ako uvádza oficiálna stránka iMyFone MarkGo [8], ide o nástroj podporovaný umelou inteligenciou, ktorý slúži na upravovanie fotiek a videí – okrem samotného odmazávania titulkov nástroj služuje aj odstránenie vodoznakov (ang. *watermarks*), dokonca si poradí aj s editovaním PDF súborov. Aplikácia je dostupná pre zariadenia s operačným systémom Windows a Android.

Po stiahnutí je aplikácia z používateľského hľadiska prívetivá (obr. 2.2), no tlačidlo v pravom hornom rohu so slovami „Buy me“ naznačuje, že jej plné funkcie je možné využiť iba po zakúpení predplatného. Po označení miesta, kde sa titulky nachádzajú, je možné vybrať až zo štyroch metód odmazania – Smooth Filling, Gaussian Blur, Color Filling a Matte Filling.



Obr. 2.2: Obrázok zobrazuje aplikáciu iMyFone MarkGo, ktorá umožňuje odmazávať titulky. Obsah vybranej sekcie označenej modrým obdĺžnikom je zamazaný a titulky nie sú viditeľné.

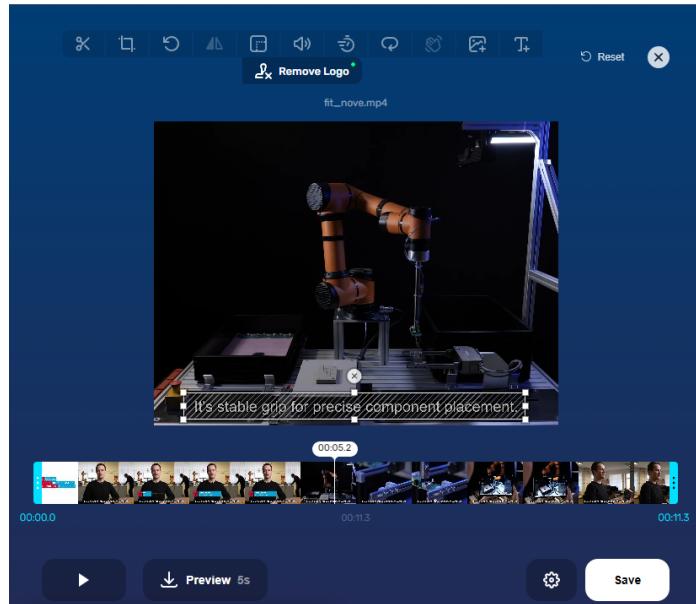
Aplikácia ukáže, ako bude výsledné video vyzerať po odmazaní vybraného textu. Voľným okom je ľahko rozpoznateľné, kde sa titulky pred zamazaním nachádzali. Avšak, aplikácia zamazáva snímky, v ktorých sa práve žiadne titulky nenachádzajú, pretože možnosť rozpoznania s využitím umelej inteligencie je primárne určená na odmazanie vodoznakov. Na exportovanie videa nie je možné použiť bezplatnú verziu, a preto je aplikácia pre používateľa, ktorý si odmieta kúpiť predplatné vo výške 7,95 \$ za mesiac, na odmazanie titulkov nevhodná.

### Webová stránka [online-video-cutter.com](http://online-video-cutter.com)

Táto online webová aplikácia zobrazená na obrázku 2.3 poskytuje mnohé funkcie užitočné pre prácu s videom – jednou z nich je aj zamazanie nami vybraného miesta, kde sa nachádzajú titulky, v mnohých formátoch videa. Na rozdiel od nástroja z predošej sekcie, dovoľuje video bezplatne exportovať, no len do dĺžky tridsať minút. Pre dlhšie videá je potrebné predplatné vo výške 6 € mesačne, rovnako ako aj pre videá s veľkosťou viac ako 500 MB. Aplikácia takisto automaticky zamazáva aj časti videa, kde sa titulky nenachádzajú a zbytočne tým vzniká rozmažanie tam, kde potrebné nie je.

### HitPaw

HitPaw je projekt, ktorý pokrýva veľké množstvo nástrojov na úpravu fotiek a videí. Online webová aplikácia umožňuje v prehliadači ľahko vybrať zónu, kde sa nachádzajú nechcené titulky, no pri exportovaní nastáva problém – aplikácia žiada prihlásenie a zanecháva v exportovanom videu svoj vodoznak. Desktopová aplikácia (obr. 2.4) je podporovaná pre Windows a macOS, no rovnako ako pri online verzii, pre uloženie videa je potrebné prihlásenie. Okrem toho je bezplatný export obmedzený iba na dve videá za deň a aplikácia bez zaplatenia nedovoľuje použiť funkciu automatickej detekcie nežiadúceho objektu umelou inteligenciou.



Obr. 2.3: Na snímke obrazovky je ukázaná online webová aplikácia online-video-cutter.com. Umožňuje výber oblasti na zamazanie, no titulky vníma ako logo a preto ich odstraňuje aj v prípade, že práve na obrazovke nie sú.

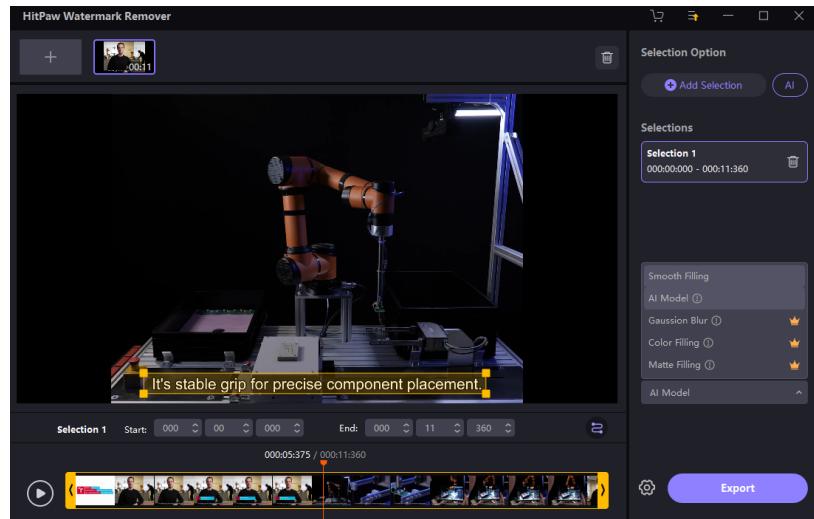
### Porovnanie s mojím riešením

Markantný rozdiel medzi mojou aplikáciou s názvom Subtitle Remover (obr. 2.5) a vyššie spomenutými existujúcimi riešeniami je v tom, že moja aplikácia má otvorený zdrojový kód<sup>1</sup>. Všetky funkcie aplikácie sú dostupné bez potreby platenia, čo je ďalší aspekt, v ktorom sa Subtitle Remover líši od ostatných.

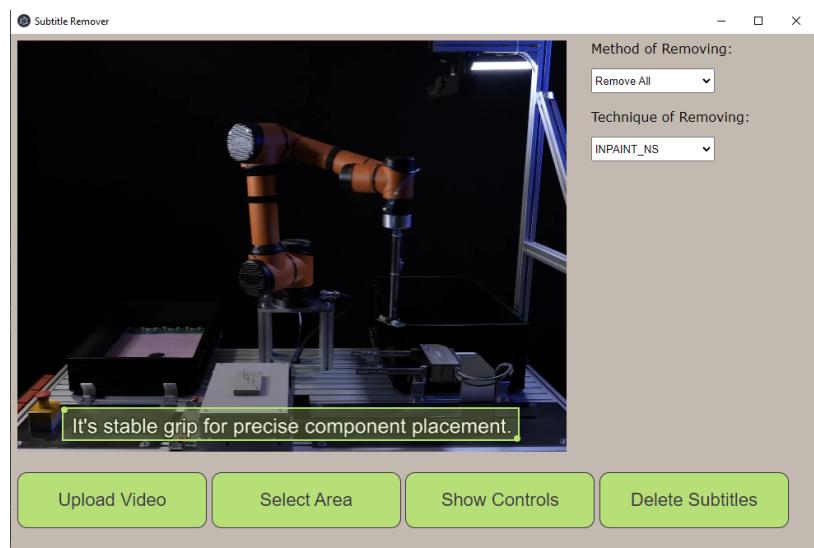
Aplikácia poskytuje dve metódy odmazávania – prvá metóda s názvom Remove All zamazáva celú vybranú oblasť, bez ohľadu na to, či tam práve sú alebo nie sú titulky. Subtitles Only je druhá metóda, ktorá pomocou detektie textu rozpozná, či je v snímke potrebné text odstrániť, alebo nie. Pokiaľ sa v určitých úsekoch titulky neobjavujú, obraz ostane pôvodný.

Subtitle Remover umožňuje používateľovi zvoliť medzi štyrmi technikami zamazávania – každá technika má svoje výhody a nevýhody, čo sa kvality obrazu a dĺžky spracovania týka. Sám používateľ si zvolí, ktorá mu najviac vyhovuje. Po spustení odmazávania sa zobrazí modálne okno s odhadovaným časom spracovania.

<sup>1</sup><https://github.com/emminka/subtitle-remover>



Obr. 2.4: Aplikácia HitPaw je prehľadná a ľahko ovládateľná. Poskytuje viacero spôsobov zamazania, bohužiaľ nie v bezplatnej verzii.



Obr. 2.5: Aplikácia Subtitle Remover je navrhnutá tak, aby bola pre používateľa jednoduchá aj bez predchádzajúcich skúseností s aplikáciami na úpravu videa.

# Kapitola 3

## Teoretický základ

Cieľom kapitoly je zhrnúť súčasný stav poznania v oblasti titulkov a zamazávania a poskytnúť čitateľovi ucelený prehľad o teoretickom základe potrebnom pre vývoj aplikácie pre odmazávanie titulkov.

Prvá podkapitola je zameraná na samotné titulky – ich druhy, rozloženie vo videu a trvanie. Následne sú spomenuté techniky zamazávania, ktoré sú klíčové pre vytvorenie užitočnej aplikácie. Tieto techniky zahŕňajú inpainting, Gaussov filter a medianový filter. Ďalšia podkapitola je venovaná prahovaniu a posledná sa zaoberá dilatáciou, ktorá je dôležitou technikou v oblasti spracovania obrazu.

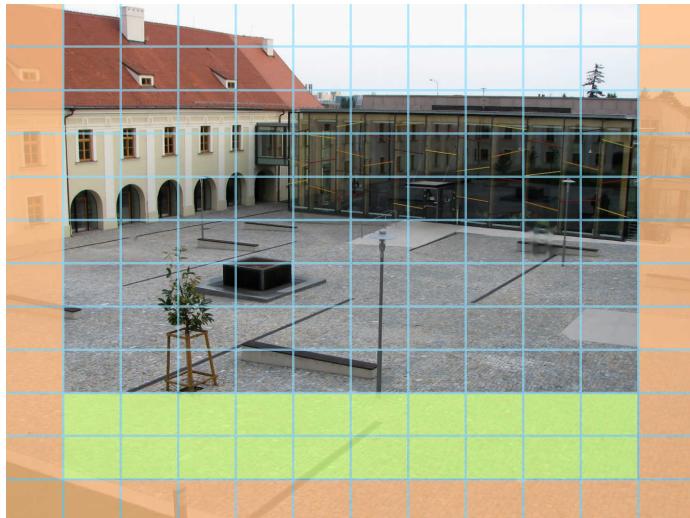
### 3.1 Titulky

Vo všeobecnosti rozoznávame dva typy titulkov – pevne vstavané titulky (*Hardsub*, *Hard-coded subtitles*, *Open Caption*) a mäkké, či skryté titulky (*Soft subtitles*, *Closed caption*), v tejto práci sa používa pojem mäkké. Zatiaľ čo vypnutie mäkkých titulkov nie je problémom, vstavané titulky, ako ich názov napovedá, nie je možné jednoducho odstrániť. Majú pevnú pozíciu a sú neodlúčiteľnou súčasťou videa. Ako píše Jorge Díaz-Cintas ako prispievateľ v Handbook of Translation Studies [6], z lingvistického hľadiska sa rozlišujú intralingválne titulky, známe aj ako titulky pre sluchovo postihnutých – SDH (*Subtitles for the Deaf and Hard of Hearing*), kde jazyk titulkov a videa sú zhodné, a interlingválnymi titulkami, kde je preložený text z pôvodného jazyka do iného jazyka.

Táto bakalárská práca využíva poznatky zo štandardu pre titulky v Európe [16], ktorý poskytuje súbor viacerých pravidiel. Priblížené sú len pravidlá týkajúce sa práce – nerelevantné časti pravidiel, ako pravidlá interpunkčných znamienok či úprava konečného textu, ktorými sa štandard zaoberá, nie sú bližšie rozoberané.

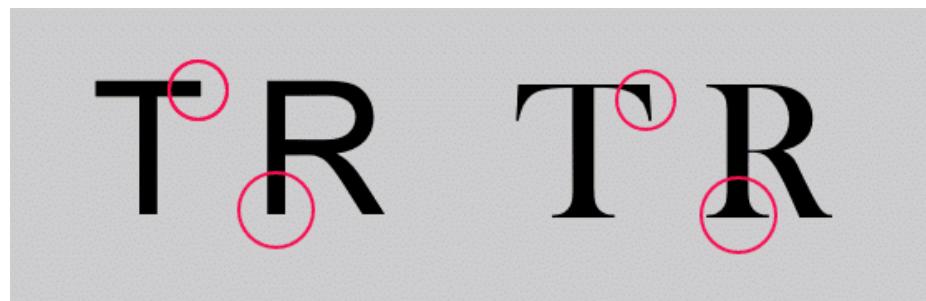
Rozloženie titulkov by malo byť zamerané na maximálnu možnú čitateľnosť a zrozumiteľnosť vloženého textu. Čo sa týka priestorových parametrov a rozloženia, titulky by mali byť umiestnené v dolnej časti obrazovky tak, aby prekrývali oblasť, ktorá pre diváka nie je veľmi dôležitá (obr. 3.1). Najnižší riadok titulkov by mal byť umiestnený aspoň v  $\frac{1}{12}$  z celkovej výšky obrazovky nad jej dolným okrajom, rovnako by mala byť medzera aj z pravej a ľavej strany, aby sledovateľ nemusel po obrazovke veľa cestovať očami. Titulky by nikdy nemali zaberať tri a viac riadkov, aby titulky naraz zakrývali maximálne  $\frac{2}{12}$  obrazovky.

Text by mal byť dalej vycentrovaný, pretože väčšina akcie sa odohráva v strede obrazovky a pre diváka je pohodlnejšie, ak je vzdialenosť medzi akciou a začiatkom titulkov kratšia. Každý riadok titulkov by mal obsahovať približne tridsať päť znakov, aby sa doň



Obr. 3.1: V časti vyznačenej oranžovou farbou by sa titulky vyskytovať nemali z dôvodu pohodlnosti pre diváka – na ľavom a pravom okraji by malo byť k dispozícii miesto, aby sa divák nemusel pohybovať očami príliš daleko pozdĺž okrajov obrazovky, aby si prečítał riadok titulkov. Pre tento účel by malo byť k dispozícii aspoň  $\frac{1}{12}$  šírky obrazovky vľavo od prvého znaku a aspoň  $\frac{1}{12}$  šírky obrazovky vpravo od posledného znaku pre každý riadok titulkov. Najnižšia  $\frac{1}{12}$  obrazovky by mala byť takisto bez titulkov, pretože pre diváka nie je pohodlné vzdalovať sa pri akcii až na kraj obrazovky. Zelená oblasť je vhodná pre umiestnenie titulkov, pričom v prípade jednoriadkového titulku by mal byť titulok umiestnený na spodnom z dvoch riadkov, aby sa minimalizovalo rušenie obsahu snímky.

mohla voľne zmestíť uspokojivá časť hovoreného či preloženého textu. Zvýšenie počtu znakov na riadok na viac ako štyridsať znakov znižuje čitateľnosť titulkov, pretože veľkosť písma je nevyhnutne zmenšená. Pre jednoduchšiu čitateľnosť sú preferované bezpätkové fonty, pretože sa tým zvyšuje ich čitateľnosť (obr. 3.2).



Obr. 3.2: Písmena T a R na pravej strane obsahujú serify, inak povedané pätky. Pre ľahšiu čitateľnosť na obrazovke je tento font oproti písmenám T a R naločno pre titulky nevhodný.

Rýchlosť čítania textu priemerného človeka vo veku 14 až 65 rokov činí priemerne 165 slov za minútu, čo je 2,75 slov za sekundu. Z tohto dôvodu má pravidlá aj trvanie zobrazenia jednotlivých titulkov. Doba trvania plného dvojriadkového titulku obsahujúceho 14 až 16 slov by mala byť maximálne približne 5,5 sekundy. Avšak odhad by mal byť rozšírený na približne 6 sekúnd, pretože je potrebné pridať aj cca 0,25 až 0,5 sekundy, ktoré ľudský mozog potrebuje na spracovanie sledovaných titulkov.

Rovnako dôležité ako udržanie plného dvojriadkového titulku na obrazovke aspoň 6 sekúnd pre zabezpečenie dostatočného času na čítanie je udržiavať tento titulok na obrazovke nie viac ako 6 sekúnd, pretože to by mohlo spôsobiť automatické opäťovné čítanie titulkov, najmä pre rýchlych čitateľov. Pre jednoriadkové titulky obsahujúce 7 až 8 slov by mala byť potrebná maximálna doba trvania približne 3,5 sekundy. Trvanie zobrazenia titulku s jedným slovom by malo byť aspoň 1,5 sekundy – kratšia doba by titulok len narýchlo zobrazila na obrazovke a dráždila by oči divákov.

## 3.2 Detekcia textu

Detekcia textu na obrázkoch [17] je úloha, ktorá sa zaobráva identifikáciou a lokalizáciou textových prvkov v obrazu. Táto technika je často používaná v oblasti optického rozpoznávania znakov (OCR), kde je cieľom previesť text na obrázku do strojovo čitateľnej formy.

Existuje viacero prístupov k detekcii textu na obrázkoch – jeden z najbežnejších je založený na konvolučných neurónových sieťach (CNNs), ktoré sú špecializované na prácu s vizuálnymi dátami. Tieto siete sa dokážu automaticky naučiť detektovať textové oblasti v obrázkoch na základe vzorov v trénovacej dátovej sade.

Okrem detekcie textu sa v OCR aplikáciách používajú aj rôzne techniky na rozpoznávanie samotného textu. Medzi najbežnejšie patria metódy založené na rekurzívnych neurónových sieťach (RNNs), ktoré dokážu prevádzkať textové obrazy na skutočný text.

## 3.3 Zamazávanie

Odstraňovanie nežiaducích elementov z obrazového materiálu je častou požiadavkou v oblasti spracovania obrazu. Zamazávanie alebo skrytie určitých častí obrazu môže byť vhodné z mnohých dôvodov, vrátane ochrany súkromia, odstránenia nechcených prvkov z fotografie alebo videa, alebo jednoducho pre úpravu estetického vzhľadu obrazu.

Existuje mnoho metód a techník, ktoré sa používajú na odstraňovanie nežiaducích častí obrazu, a to v mnohých sférach, od medicíny až po filmovú produkciu. Niekoľko tieto techniky používajú na zamazávanie celých objektov alebo oblastí obrazu, zatiaľ čo v iných prípadoch sa používajú na jemné úpravy, ako je odstránenie šumu z obrazu alebo vylepšenie kvality videa.

Zamazávanie titulkov vo videu je technika, ktorá sa používa na odstránenie alebo skrytie nežiaducích titulkov. Existuje niekoľko metód, ktoré sa používajú na zamazávanie titulkov vo videu, vrátane inpaintingu, Gaussovho rozmazania a mediánového filtru.

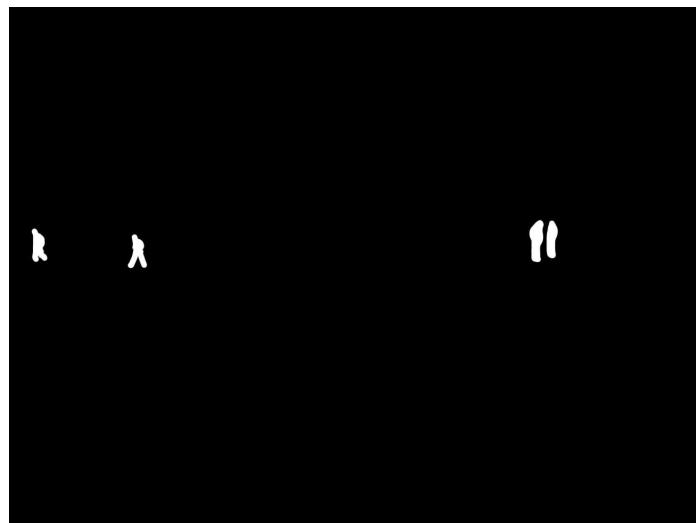
### 3.3.1 Inpainting

Inpainting je metóda, ktorá dokáže časť obrázku zmazať a nahradí ju nejakou inou (obr. 3.3). Do slovenčiny sa dá preložiť ako *Rekonštrukcia obrazu*, či *Retušovanie*, no v práci bude používaný pôvodný anglický termín. Dokáže odstrániť rušivý objekt vo videu, či v obrázku pomocou masky (obr. 3.4), ktorá daný objekt označí a následne premaľuje pozadím zvyšného obrázku.

Na diagrame 3.5 je vidieť, že využitie je veľmi široké – najčastejšie ide o *Image restoration*, inak povedané *Obnova obrazu*, *Photo-editing* alebo aj *Editovanie obrazu* a *Image coding and transmission*, či *Obrazové kódovanie a prenos*.



Obr. 3.3: Fotografia naľavo je pred odstránením nežiadúcich častí (osôb), napravo je vidieť opravenú fotografiu metódou inpainting s pomocou masky.

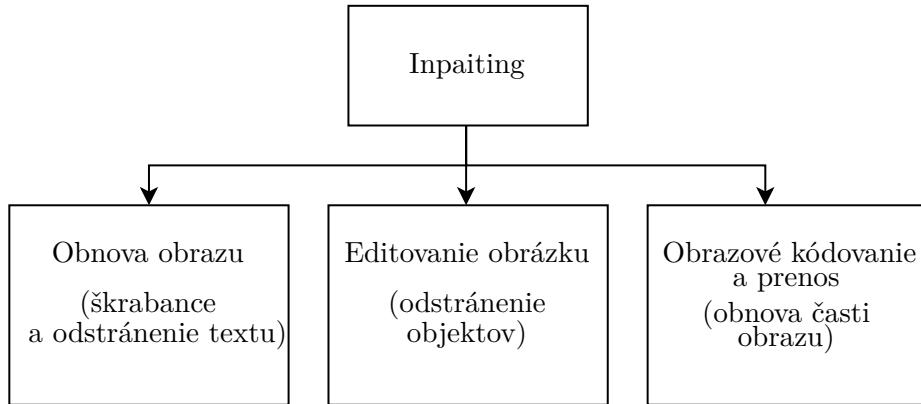


Obr. 3.4: Maska čierou farbou označuje oblasť, ktorá má zostať pôvodná a bielou oblasti, ktoré majú byť inpaintované. Na konkrétnej maske sú bielou označené osoby, ktoré sú na fotke rušivé a ktoré majú byť z obrázku odstránené.

### 3.3.2 Gaussovo rozostrenie

Gaussovo rozostrenie je matematický obrazový filter, ktorý vyhľadzuje obraz alebo signál na základe Gaussovej funkcie, čo umožňuje zachovanie hladkých prechodov a prirodzeného vzhľadu obrazu. Ako je spomenuté v [3], Gaussovo rozostrenie, Gaussovo vyhľadzovanie, či Gaussov filter je vykonaný konvolúciou každého bodu vo vstupnom poli s Gaussovým jadrom a následným sčítaním na vytvorenie výstupného pola. Prakticky ide o filter s dolnou prieplšťou, pričom primárnym účelom je znížiť v obraze šum a množstvo detailov – efekt je vhodný pre skrytie oblastí na obrázku bez ich zničenia. Vizualizácia rozdielu medzi obrazom s Gaussovým rozostrením a originálnym obrazom je viditeľná na obr. 3.6. Je to dosiahnuté konvolúciou obrazu s Gaussovým jadrom, ktoré je v dvojdimenziomálnom priestore definované nasledujúco:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.1)$$



Obr. 3.5: Diagram [7] zobrazuje aplikácie inpaintingu a cieľ každej kategórie. *Image restoration* je termín označujúci proces obnovy poškodeného alebo znehodnoteného obrazu na pôvodný stav. Tento proces sa často používa na odstránenie šumu, rozmazania alebo iných defektov z obrazu, čím sa zlepšuje jeho kvalita a estetický vzhľad. V kontexte inpaintingu obnova obrazu zahŕňa obnovenie chýbajúcej oblasti obrazu na základe okolitého obsahu. V prípade inpaintingu môže byť photo-editing použitý na úpravu obnoveného obsahu chýbajúcej oblasti tak, aby bol čo najrealistickejší. V kontexte obrazového kódovania a prenosu pri inpaitingu je dôležité zohľadniť efektivitu kompresie, aby sa minimalizoval objem prenášaných dát a zároveň zachovala kvalita obnoveného obrazu, pretože zamazávanie obrazu môže mať vplyv na štruktúru, textúru a farbu obrazu. Správny výber metód kompresie a prenosu môže mať vplyv na výsledný vizuálny dojem obnoveného obrazu. Medzi bežné metódy kompresie a prenosu obrazových dát v kontexte zamazávania patrí napríklad bezstratové kompresné formáty ako PNG alebo bezstratové varianty JPEG, ktoré môžu byť vhodné pre situácie, kde je dôležité zachovať vysokú kvalitu obnoveného obrazu.

pričom:

- $(x, y)$  sú súradnice pixelu,
- $\sigma$  je smerodajná odchýlka [14].

Hodnota smerodajnej odchýlky ovplyvňuje rozptyl Gaussovo rozostrenia a teda aj mieru rozmazania okolo konkrétnego pixelu. Čím je vyššia hodnota  $\sigma$ , tým je väčšie rozostrenie, pretože filter pracuje na väčšom okolí.

### 3.3.3 Mediánový filter

Mediánový filter [3] je typ digitálneho obrazového filtra, ktorý sa používa na odstránenie šumu z digitálnych obrázkov alebo signálov. Jeho hlavným cieľom je redukovať šum, ktorý môže byť prítomný vo vstupnom obraze alebo signáli. Mediánový filter funguje na základe konceptu mediánu, čo je hodnota, ktorá je v strede hodnôt usporiadanejho súboru hodnôt.

Používa jadro – je to malá oblasť okolo každého pixelu v obraze, cez ktorú prechádza filter. Mediánový filter prechádza okolitými hodnotami pixelov a nahrádza hodnotu filtra mediánom hodnôt v okolí. Týmto spôsobom môže byť efektívne odstránený šum, pričom sa zachováva ostrosť hrán a detailov v obraze alebo signáli (obr. 3.7.).



Obr. 3.6: Na ľavej strane je pôvodný obrázok, na pravej je po použití Gaussovoho filtra. Čím väčšie jadro je použité, tým bude obrázok viac rozmazaný.



Obr. 3.7: Na pravej strane je obrázok po použití mediánového filtra. Podobne ako pri Gaussovom rozmazaní, čím väčšia veľkosť jadra je použitá, tým bude obrázok viac rozmazaný.

### 3.4 Prahovanie

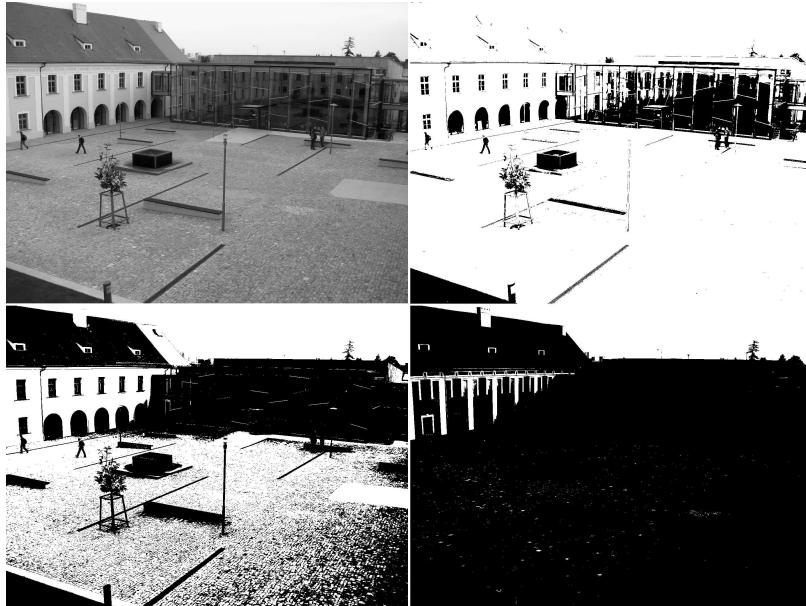
Prahovanie (ang. *thresholding*) [2, 12, 19] je proces transformácie digitálneho obrazu, ktorý redukuje obrázok v odtieňoch sivej na čiernobiely na základe prahu (obr. 3.8). Tento prah je predom určená hodnota a každý pixel v obrázku sa porovnáva s touto hodnotou. Ako je zobrazené v definícii 3.2, ak je hodnota pixelu vyššia ako prahová hodnota, pixel sa transformuje na bielu farbu, inak sa transformuje na farbu čiernu. Tento proces sa často používa na zvýraznenie detailov v obrázkoch a na odstránenie nežiaducích detailov alebo šumu.

$$g(x, y) = \begin{cases} 1, & \text{ak } f(x, y) > T \\ 0, & \text{ak } f(x, y) \leq T \end{cases} \quad (3.2)$$

Kde:

- $g(x, y)$  je výstupný obraz,
- $f(x, y)$  je vstupný obraz,
- $T$  je hodnota prahu,

- 0 a 1 sú nové hodnoty pre vstupnú hodnotu  $f(x, y)$  pod a nad prahom, pričom 1 reprezentuje bielu farbu a 0 čiernu.



Obr. 3.8: Na ľavom hornom obrázku je pôvodná snímka v odtieňoch sivej, ktorá má rozmanitú škálu hodnôt odtieňov sivej od čiernej až po bielu. Obrázok napravo hore ukazuje ten istý obrázok, ktorý bol podrobenej procesu prahovania s hodnotou prahu 64. Každý pixel v obrazu, ktorý má hodnotu nižšiu alebo rovnú 64 bol transformovaný na čiernu farbu, zatiaľ čo každý pixel s hodnotou vyššou ako 64 bol transformovaný na bielu farbu. Obrázok naľavo dole ukazuje snímku po prahovaní s hodnotou prahu 128 a obrázok napravo dole zobrazuje výsledok prahovania s prahom 192 – viditeľné sú len najsvetlejšie oblasti pôvodnej snímky, ktoré sú zobrazené bielou farbou.

Existujú rôzne techniky prahovania, ktoré sa líšia výberom prahu a spôsobom jeho aplikácie na vstupný obraz. Niektoré techniky sú adaptívne a prah sa mení v závislosti od vlastností obrazu, zatiaľ čo iné techniky sú globálne a používajú jednu prahovú hodnotu pre celý obraz. Okrem toho sú niektoré techniky založené na štatistických vlastnostiach obrazu, ako je napríklad priemerná hodnota alebo štandardná odchýlka pixelov, zatiaľ čo iné techniky sa spoliehajú na iné metriky, ako je napríklad entropia alebo energetické charakteristiky. Výber správnej techniky prahovania závisí od konkrétneho problému a ciela digitálneho spracovania obrazu.

### 3.5 Erózia a dilatácia

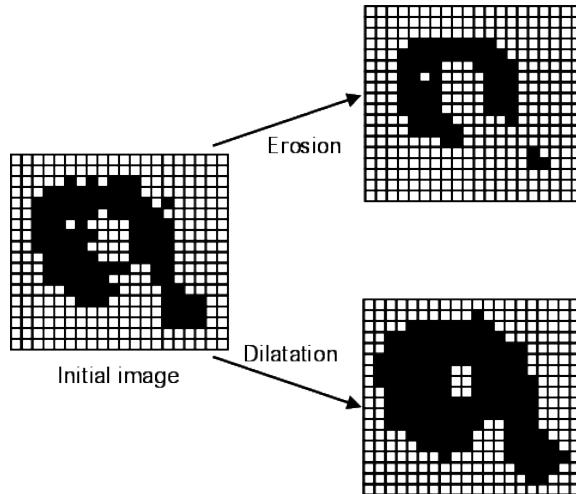
Ako John C. Russ píše vo svojej knihe [19], morfológické operácie sú triedou techník spracovania binárnych obrazov používaných na ich úpravu. Najčastejšie používané techniky sú erózia a dilatácia. Tieto operácie zahrňajú pridávanie alebo odstraňovanie pixelov z binárneho obrazu podľa určitých pravidiel, ktoré závisia od vzoru susedných pixelov.

Erózia sa používa na odstránenie pixelov z prvkov na obrázku. Táto operácia môže pomôcť eliminovať cudzie pixely, ktoré by nemali byť prítomné v obrazu, ako je bodový šum alebo chyby čiar. Klasickým pravidlom erózie je odstránenie akéhokoľvek pixelu dotykajú-

ceho sa iného pixelu, ktorý je súčasťou pozadia. Tým sa odstráni vrstva pixelov z okraja všetkých prvkov a oblastí, čo spôsobí určité zmenšenie rozmerov.

Naopak dilatácia sa používa na pridávanie pixelov do obrazu (obr. 3.10). Klasickým pravidlom dilatácie je pridať akýkoľvek pixel pozadia, ktorý sa dotýka iného pixelu, ktorý je už súčasťou oblasti popredia. Tým sa pridá vrstva pixelov po obvode všetkých prvkov a oblastí, čo spôsobí určité zväčšenie rozmerov. Dilatácia tiež vypĺňa malé diery v rámci prvkov a používa sa na spojenie rozpojených častí objektu.

Kedže erózia a dilatácia (obr. 3.9) spôsobujú zmenšenie alebo zväčšenie veľkosti regiónov, sú niekedy známe ako zmenšovanie a rast (ang. *shrinking and growing*).



Obr. 3.9: Obrázok zobrazuje rozdiel medzi eróziou a dilatáciou nad rovnakým obrázkom. Prevzaté z [11].



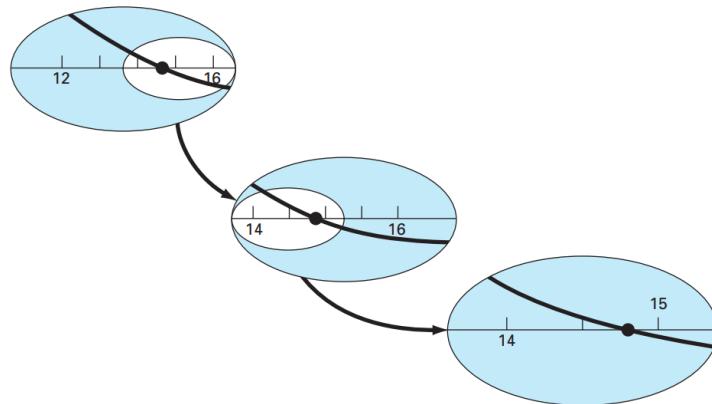
Obr. 3.10: Na ľavej strane je zobrazený obrázok po prahovaní s prahom 100, na ľavej strane je daný obrázok po dilatácii.

## 3.6 Bisekcia

Metóda bisekcie [4] alebo metóda polenia intervalu je typ inkrementálneho vyhľadávacieho algoritmu, ktorý zahrňa delenie intervalu na polovicu na hľadanie koreňa, čo je miesto kde

sa funkcia rovná nule. Táto metóda musí byť použitá nad monotónnou funkciou a spočíva v tom, že interval sa rozdelí na dve polovice a následne sa vyhodnotí, na ktorej z polovic sa nachádza koreň. Tento postup sa zopakuje na vybranej polovici intervalu, v ktorej sa koreň nachádza. Postup sa opakuje až kým nie je dosiahnutá dostatočná presnosť alebo dokým nie je koreň nájdený (obr. 3.11).

Táto metóda je jednoduchá a spoľahlivá, pretože v každom kroku zaručuje, že sa interval, v ktorom sa nachádza koreň, zmenší na polovicu. Pre opakované delenie intervalu na polovice môže byť táto metóda pomalá pri hľadaní koreňa na veľmi veľkom intervale.

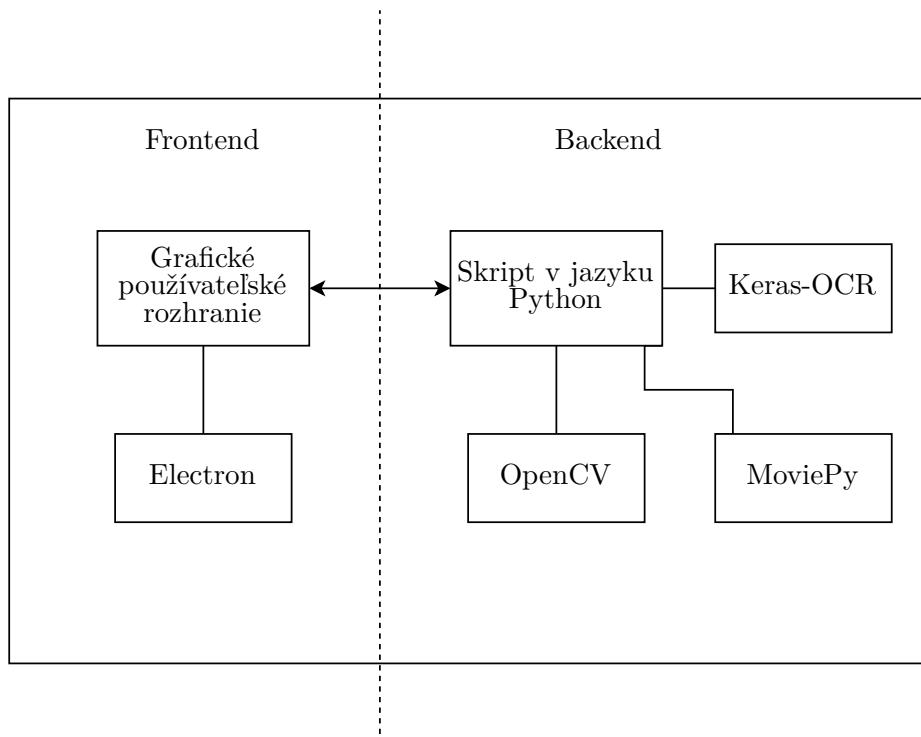


Obr. 3.11: Ukážka metódy bisekcie. Prevzaté z [4].

## Kapitola 4

# Funkcionálny popis riešenia

Ako je zobrazené na obrázku 4.1, aplikácia sa skladá z dvoch hlavných častí – prvou je GUI aplikácia vytvorená s knižnicou Electron 5.1 v jazyku JavaScript; druhou časťou je skript v jazyku Python, ktorý vykonáva manipuláciu s videom (detekcia textu (kapitola 3.2), zamazávanie (kapitola 3.3)).



Obr. 4.1: Aplikácia Subtitle Remover sa skladá z dvoch hlavných navzájom prepojených častí. Informácia sa do backendu posielá pomocou argumentov spúšťacieho príkazu. Pythónový skript je možné použiť aj samostatne pomocou príkazového riadku, bez použitia frontendu.

GUI aplikácia slúži používateľovi na jednoduchú konfiguráciu zamazávania, kde používateľ nahrá video a sám si vyberá časť videa, kde sa nachádzajú titulky na zamazanie (obr. 4.2). Následne GUI aplikácia volá skript v jazyku Python, konfigurácia je zasielaná pomocou argumentov príkazového riadku.



Obr. 4.2: Diagram ukazuje postup práce s aplikáciou. V prvom kroku sa načíta vstupné video – tento krok je nevyhnutný pre ďalšiu prácu s videom. V druhom kroku je potrebné vybrať oblasť titulkov (kapitola 4.2.1) a v poslednom kroku sa spúšta skript na odmazanie titulkov.

## 4.1 Backend

GUI aplikácia poskytuje prívetivé rozhranie na prácu so skriptom, ktorý spracúva obraz. Aplikácia poskytuje viaceré techník a metód odmazávania. V sekcií parametrov (obr. 4.12) má používateľ na výber z dvoch metód a štyroch techník odmazávania.

### 4.1.1 Metódy odmazávania titulkov

Prvá metóda je metóda `Remove all` – výber tejto možnosti zaručí, že celá vybraná oblasť titulkov bude vo výslednom videu zamazaná bez ohľadu na to, či existujú vo videu snímky, na ktorých sa titulky nenachádzajú. Táto metóda slepo zamazáva celú vybranú oblasť a je vhodná, pokiaľ má používateľ v pláne odstraňovať titulky, ktoré nie sú v latinskej abecede (ide napríklad o titulky v čínštine, hebrejčine, atď.). Oblast je zamazávaná pomocou masky, ktorej tvorba je popísaná v algoritme 1. Táto maska sa vykreslí raz na začiatku programu a v tejto metóde je až do konca behu programu nemenná (obr. 4.3).

---

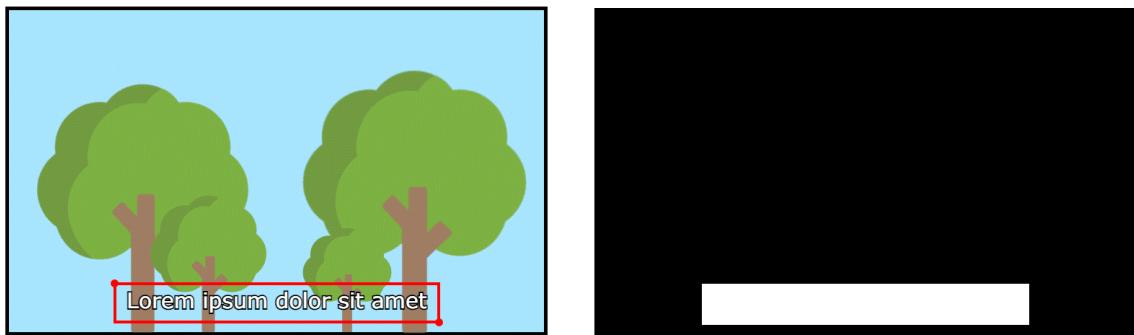
**Algoritmus 1:** Algoritmus na vytvorenie masky pri metóde `Remove all`

---

**Data:** Výška videa **VV**, šírka videa **SV**, súradnice oblasti **Lx**, **Ly**, **Rx**, **Ry**

**Result:** Maska s vyznačenou oblasťou titulkov **M**

- 1 `cierna_snimka = vykreslenie_obdlznika(VV, SV);`
  - 2 `M = vykreslenie_obeasti(cierna_snimka, Lx, Ly, Rx, Ry);`
- 



Obr. 4.3: Na pravom obrázku je zobrazená maska pre video, kde sú titulky odmazávané metódou `Remove all`. Maska je rovnaká pre všetky snímky vo videu nehľadiac na to, či sa práve na snímke titulky nachádzajú.

**Subtitles only** je názov druhej metódy odmazávania. Ako jej pomenovanie napovedá, metóda odstraňuje titulky len v prípade, že sa práve na snímkach videa nachádzajú. Ich nájdenie má na starosti knižnica Keras-OCR (kapitola 5.2.2).

Zistovať však, či sa titulky nachádzajú na každom jednom snímku je veľmi zdĺhavé a časovo neefektívne. Ako je spomenuté v kapitole 3.1, titulky by mali byť na obrazovke minimálne 1,5 sekundy. Typický počet snímok za sekundu vo filmoch [18] je 24, preto je po zaokrúhlení ako východzia hodnota nastavená kontrola titulkov na každej tridsiatej snímke. Túto hodnotu môže používateľ zmeniť podľa svojich preferencií, napríklad keď vie, že sa titulky vo videu menia príliš rýchlo. Text sa teda hľadá na každej tridsiatej snímke a keď je detegovaný, nájde sa pomocou metódy bisekcie presná prvá snímka začiatku určitých titulkov.

V tejto metóde je odmazávanie presnejšie – každý konkrétny výskyt titulkov má vytvorenú presnú masku a pokial sa vo videu v určitých miestach titulky neobjavujú, tak zostávajú dané snímky nezmenené. Tvorba masky je zobrazená v algoritme 2 a konkrétnie orezané masky titulkov pri rôznych prípadoch sú na obrázkoch 4.4, 4.5 a 4.6.

---

**Algoritmus 2:** Algoritmus na vytvorenie masky pri metóde **Subtitles only**

---

**Data:** Prvá snímka s titulkami **FT**, výška videa **VV**, šírka videa **SV**, súradnice oblasti **Lx, Ly, Rx, Ry**

**Result:** Presná maska **PM**

- 1 cierna\_snimka = vykreslenie\_obdlznika(VV, SV);
  - 2 frame\_predchadzajuci = FT - 1;
  - 3 urobit\_sedotonove(frame\_predchadzajuci, FT);
  - 4 orezat(frame\_predchadzajuci, FT, Lx, Ly, Rx, Ry);
  - 5 rozdiel = absolutny\_rozdiel(frame\_predchadzajuci, FT);
  - 6 zprahovanie = urobit\_prahovanie(rozdiel);
  - 7 jadro = urob\_jadro();
  - 8 zdilatovane = dilatacia(zprahovanie, jadro);
  - 9 PM = spojit(cierna\_snimka, zdilatovane);
- 

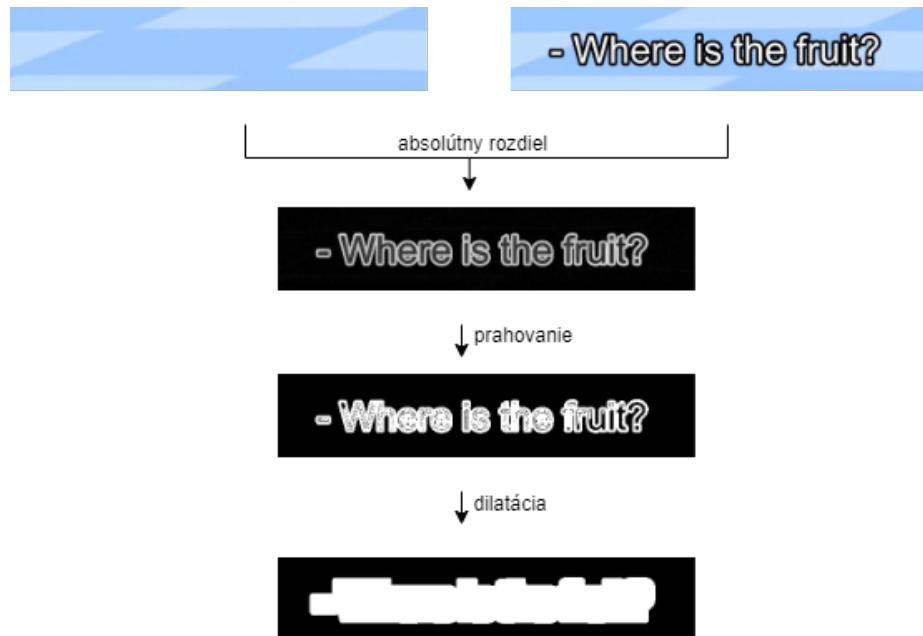
#### 4.1.2 Techniky odmazávania titulkov

Po zvolení jednej z metód odstraňovania titulkov (kapitola 4.1.1) má používateľ na výber zo štyroch techník zamazávania. Na orezanom ukážkovom obrázku 4.7 sú uplatnené všetky techniky zamazávania s presnou maskou vytvorenou metódou **Subtitles only**.

Prvá technika je zamazávanie pomocou algoritmu Navier-Stokes [5], ktorého cieľom je vytvoriť revidovaný obraz, v ktorom je odmazaná oblasť hladko zlúčená s obrazom tak, aby to nebolo viditeľné na prvý pohľad bežným divákom.

Algoritmus je založený na mechanike tekutín a využíva parciálne diferenciálne rovnice. Zahŕňa priame riešenie rovníc Navier-Stokes pre nestlačiteľnú tekutinu. Po tom, čo používateľ vyberie oblasť na zamazanie, algoritmus najskôr prechádza pozdĺž hrán z oblastí, ktoré sú známe, do neznámych oblastí a automaticky transportuje informácie do oblasti inpaintingu. Algoritmus využíva izofoty, čo sú čiary alebo krivky v diagrame ktoré spájajú body s rovnakou intenzitou svetla alebo jasu.

Marcelo Bertalmio a spol. [1] vybudovali analógiu medzi funkciou intenzity obrazu a tokom tekutiny v 2D nestlačiteľnej tekutine a použili techniky z výpočtovej dynamiky tekutín

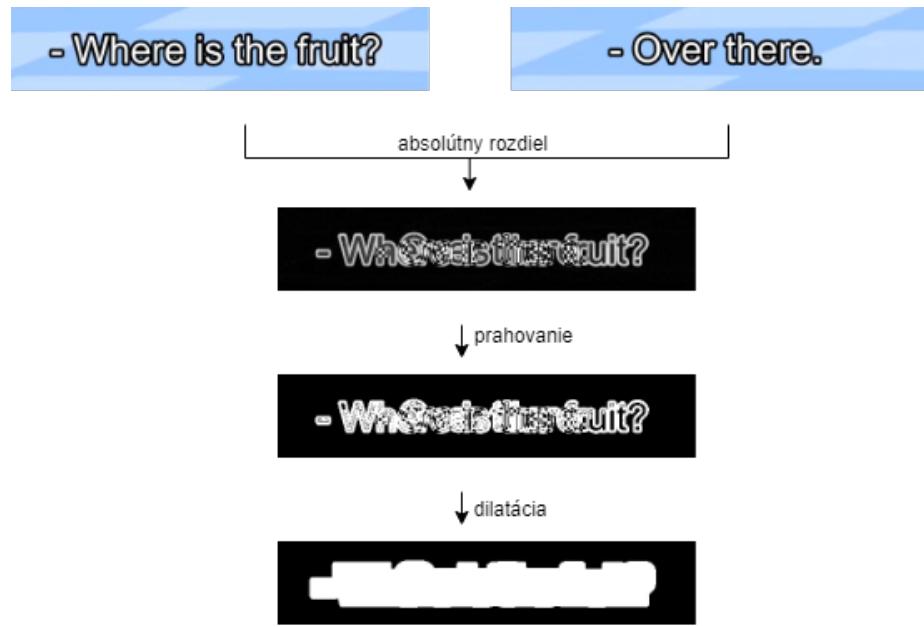


Obr. 4.4: Na obrázku je tvorba masky pre snímku s titulkami (napravo hore) za predpokladu, že predchádzajúca snímka titulky neobsahovala (naľavo hore). Vstupné snímky do absolútneho rozdielu sú orezané na veľkosť oblasti titulkov zvolenej používateľom a ich farebná škála je zmenená na šedotónovú. Po vykonaní rozdielu je text titulkov pekne viditeľný, no maska musí byť čiernobiela a preto podstúpi prahovanie. Dilatácia následne vyplní medzery v slovách a zväčší okolie titulkov. Upravená oblasť sa vloží na presné miesto z akého bola orezaná a maska pre konkrétné titulky je vytvorená.

na produkciu približného riešenia problému inpaintingu. Použitie tejto techniky je vizuálne ukázané na obrázku 4.8.

Technika inpaintingu obrázka založená na metóde rýchleho pochodu (ang. *Fast Marching Method*) je druhá technika, ktorá je používaná na zamazanie titulkov. Autorom tohto algoritmu je Alexander Telea [20] a preto je skrátene táto technika v aplikácii Subtitle Remover pomenovaná ako **Inpainting Telea**. Oblast, ktorá má byť zamazaná je nazývaná neznámou oblasťou a obklopuje ju oblasť známa. Algoritmus najprv inpaintuje všetky pixely ležiace na hranici oblastí a postupne pokračuje smerom k stredu neznámej oblasti. Na inpaintovanie každého pixelu v neznámej oblasti sa používa váhová funkcia, ktorá zohľadňuje susednú oblasť pixelu a zabezpečuje, že inpaintovaný pixel je ovplyvnený viac pixelmi, ktoré sú k nemu bližšie, a menej pixelmi, ktoré sú ďalej [5]. Vizuálna ukážka uplatneného algoritmu je zobrazená na obrázku 4.9.

Ďalšou technikou je využitie Gaussovoho filtra (kapitola 3.3.2). Masku sa však nedá uplatniť rovnakým spôsobom ako pri dvoch vyššie spomenutých inpaintingoch, pretože v nich sa maska predáva priamo vo funkcií knižnice OpenCV (kapitola 5.2.1). Na uplatnenie masky a tvorbu výslednej snímky pri výbere techniky **Gaussian Blur** je preto použitý algoritmus 3 – najprv je aplikovaný Gaussov filter na vstupnú snímku a masku. Tým sa redukuje šum v oblastiach, kde sa nachádzajú titulky. Následne je použitá funkcia dilatácie na masku – cieľom je zväčšíť biele pixely, ktoré predstavujú titulky, a tým zabezpečiť, že pri zmiešaní so snímkom budú titulky úplne zakryté; ďalej je maska prekonvertovaná do trojkanálového formátu. Zo snímky je pomocou masky extrahovaná oblasť výsledného snímku kde boli ti-



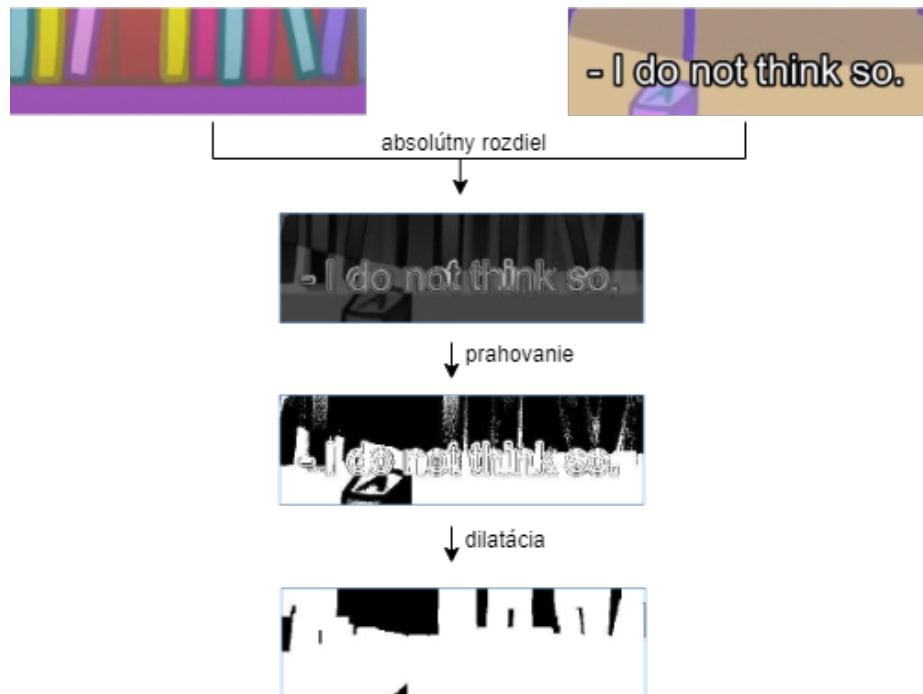
Obr. 4.5: Obrázok zobrazuje tvorbu masky pre titulky so znením „- Over there.“. Keď predchádzajúca snímka (na ľavej strane hore) obsahuje iné titulky ako snímka po ňom (na pravej strane hore), postup tvorby masky je rovnaký ako na obrázku 4.4. Po absolútnom rozdielie už však text nie je jasne čitateľný, no prahovanie a dilatácia vytvoria oblasť na zamaskovanie. Je viditeľné, že oblasť nie je úplne presným obrysom titulkov na zamazanie, napriek tomu ich však celé pokrýva a titulky budú zamazané s malým okolitým rozmažaním.

tulky, zo vstupného pôvodného snímku je extrahovaný invertovanou maskou zvyšok snímky – po ich sčítaní vznikne výsledná snímka, ktorej príklad je na obrázku 4.10.

Poslednou technikou zamazávania je použitie mediánového filtra (kapitola 3.3.3). Maska je na snímku uplatňovaná podobným spôsobom ako pri predchádzajúcej technike – postup je zobrazený v algoritme 4 a výsledok na obrázku 4.11.

## 4.2 Frontend

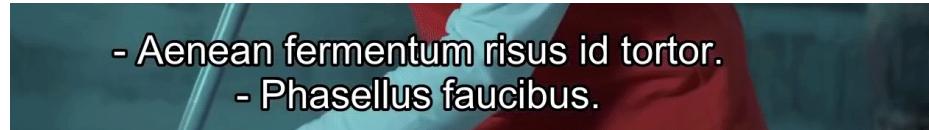
Užívateľské rozhranie je vizualizované v návrhu na obrázku 4.12, ktorý bol vyrobený pred vytvorením samotnej aplikácie, aby bolo známe, kde sa rozvrhnú jednotlivé komponenty používateľského rozhrania. Ako je napísané v popise na obrázku 4.12, aplikácia je ovládaná pomocou tlačidiel zobrazených na obrázku 4.13. Tlačidlo **Upload Video** umožní používateľovi vybrať zo zariadenia video vo formáte \*.mp4. Tlačidlo **Select Area** je bližšie priblížené v kapitole 4.2.1 a tlačidlo **Hide Controls** skryje ovládacie prvky nad videom pre lepšiu viditeľnosť titulkov. Za predpokladu že by bol dodržaný štandard pre titulkovanie spomenutý v kapitole 3.1, by dané tlačidlo potrebné nebolo. Titulky sa však môžu nachádzať na úplnom spodku obrazovky a cez ovládacie prvky videa by titulky boli zle viditeľné. Následne je možné ovládacie prvky znova zobraziť a používateľ môže video pretáčať a skontrolovať, že oblasť titulkov je pokrytá celá. Posledné tlačidlo **Delete Subtitles** spúšta zamazávanie titulkov, dodatočné zmeny už nie sú možné. Otvorí sa modálne okno s približným časom odstránenia a ukazovateľom postupu odmazávania (obr. 4.14).



Obr. 4.6: Obrázok zobrazuje posledný z prípadov, ktoré môžu pri tvorbe masky nastat – zmena scény a predchádzajúca snímka sa od snímky aktuálnej teda úplne líši. Aj v tomto prípade je postup štandardný podľa algoritmu 2, no je viditeľné, že výsledná maska neobkresluje presne titulky ako to bolo v prípade na obrázku 4.4. Stále je však maska použiteľná a v najhoršom možnom prípade, pri úplnej farebnej zmene scény pri začiatku titulkov, by výsledná maska mala čisto bielu oblasť titulkov ako to bolo v prípade metódy Remove All.

#### 4.2.1 Oblast titulkov

Jedným z nevyhnutných krokov pre odmazanie titulkov je výber oblasti, kde sa titulky nachádzajú. Oblast sa vyberá po stlačení tlačidla **Select Area** a následnými dvoma kliknutiami na video. Prvé kliknutie, ako plávajúca notifikácia v aplikácii napovedá, označuje ľavý horný roh oblasti, druhé kliknutie označuje pravý dolný roh titulkov.



Obr. 4.7: Snímka z videa na otestovanie všetkých techník odmazávania, ktoré aplikácia Subtitles Remover ponúka.



Obr. 4.8: Odmazávanie pomocou algoritmu Navier-Stokes.

---

**Algoritmus 3:** Algoritmus na uplatnenie masky pri technike Gaussovho rozmazania

---

**Data:** Snímka na zamazanie titulkov **snímka**, maska pasujúca na snímku **maska**

**Result:** Snímka bez titulkov **snímka\_bez**

```
1 gauss_snímka = gauss_filter(snímka);
2 gauss_filter(maska);
3 jadro = vytvorenie_jadra();
4 dilatovana_maska = dilatacia(maska, jadro);
5 farebna_maska = sfarbenie_masky(dilatovana_maska);
6 znormálizovanie_masky(farebna_maska);
7 a = nasobenie(1 - farebna_maska, snímka);
8 b = nasobenie(farebna_maska, gauss_snímka);
9 snímka_bez = scitanie(a, b);
```

---



Obr. 4.9: Snímka po odmazaní technikou založenou na metóde rýchleho pochodu. Výsledok je podobný ako pri algoritme Navier-Stokes a líší sa len v detailoch.



Obr. 4.10: Zamazanie oblasti Gaussovým rozostrením.



Obr. 4.11: Snímka po zamazaní mediánovým filtrom.

---

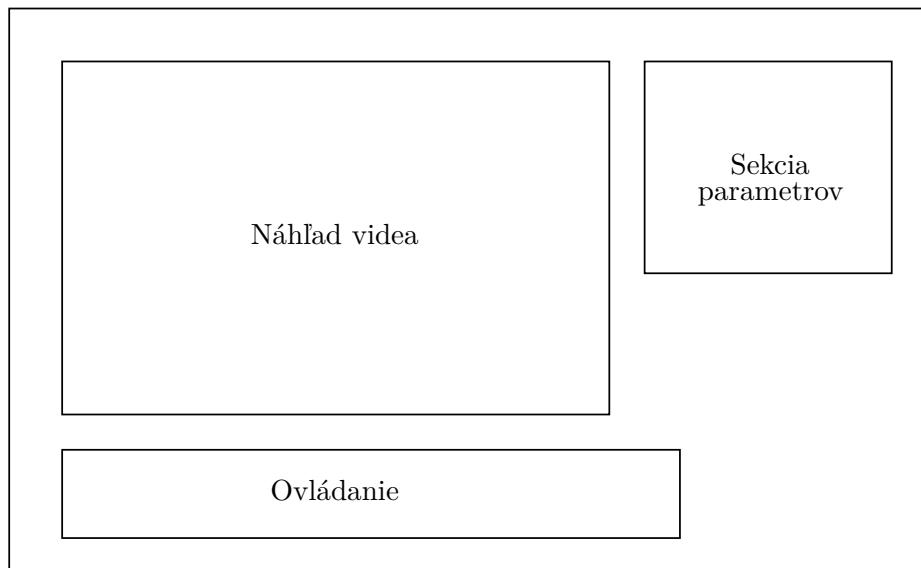
**Algoritmus 4:** Algoritmus na uplatnenie masky pri použití mediánového filtra

---

**Data:** Snímka na zamazanie titulkov **snímka**, maska pasujúca na snímku **maska**

**Result:** Snímka bez titulkov **snímka\_bez**

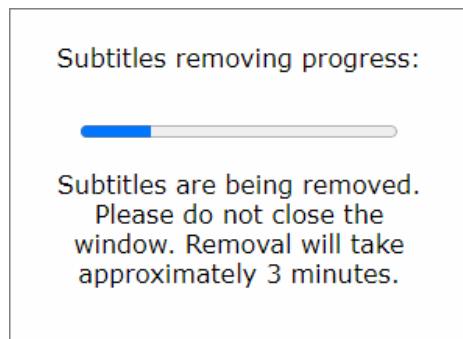
- 1 `median_snímka = median_filter(snímka);`
  - 2 `farebná_maska = sfarbenie_masy(maska);`
  - 3 `znormalizovanie_masy(farebná_maska);`
  - 4 `a = nasobenie(1 - farebná_maska, snímka);`
  - 5 `b = nasobenie(farebná_maska, median_snímka);`
  - 6 `snímka_bez = scitanie(a, b);`
- 



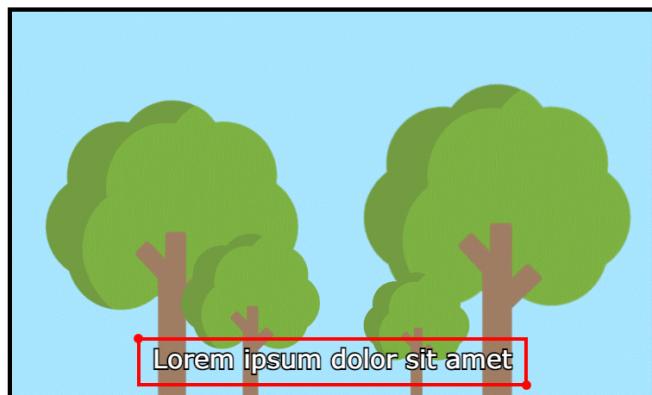
Obr. 4.12: Návrh (*mock-up*) aplikácie na odmazávanie titulkov. Po spustení aplikácie sa zobrazí okno, ktoré obsahuje oblasť na prehrávanie videa, štyri tlačidlá na ovládanie a sekciu parametrov na spresnenie odmazávania. Jednotlivé tlačidlá slúžia na vybranie a nahranie videa s titulkami, ďalej na vybranie oblasti videa, kde sa titulky objavujú. Oblasť sa vyberá dvoma klikmi na video, a to v oblasti ľavého horného rohu a pravého dolného rohu titulkov. Následne sa dá táto vymedzená časť zväčšovať/zmenšovať a presúvať podľa preferencií používateľa. Je potrebné, aby táto oblasť pokrývala všetky titulky, ktoré sa vo videu majú odstrániť. Dá sa to jednoducho skontrolovať pretáčaním videa pomocou ovládacích prvkov videa. Posledným tlačidlom sa spustí na pozadí skript a titulky budú zamazané.



Obr. 4.13: Tlačidlá na ovládanie aplikácie. Tmavozelenou sú zobrazené neaktívne tlačidlá, ktoré sa stanú aktívnymi až po vykonaní akcie svetlozeleného tlačidla. Tým sa zabráni napríklad výber oblasti nad nenahraným videom. Rovnako nie je možné spustiť odmazávanie titulkov bez vybranej oblasti tlačidlom **Select Area**.



Obr. 4.14: Modálne okno, ktoré vidí používateľ po spustení odmazávania. Prvým krokom je výpočet odhadovanej dĺžky odmazávania, nasleduje zapĺňanie progress baru a na konci odmazávania okno upozorní, že titulky boli odmazané a aplikáciu je možné zavrieť.



Obr. 4.15: Na obrázku je výrazným rámom ohraničená oblasť titulkov vo videu. Čahaním ľavého horného rohu sa mení pozícia oblasti, ktorá sa celá presúva a čahaním pravého dolného rohu je menená veľkosť oblasti.

## Kapitola 5

# Realizácia funkcionálneho návrhu

Programovacích jazykov, s pomocou ktorých sa dajú spracúvať videosúbory je mnoho, vela z nich má knižnice, ktoré podporujú načítanie a dekódovanie videosúborov a výpočet na grafickej karte. Jedným z najpoužívanejších z nich je Python, ktorý je využitý aj v tejto práci.

Táto bakalárská práca využíva na vytvorenie grafického používateľského rozhrania JavaScript, pretože umožňuje použitie knižnice Electron, ktorá je vhodná na vytváranie desktopových aplikácií s využitím webových technológií. Na pozadí aplikácie beží skript jazyku Python, s ktorým je aplikácia prepojená (obr. 4.1).

### 5.1 Electron aplikácia

Electron JS [15] je framework, ktorý je určený na vytváranie webových aplikácií s použitím HTML, CSS, and JavaScriptu. Bol vyvinutý spoločnosťou GitHub a prvýkrát sa objavil v roku 2013. Jeho pôvodný názov bol Atom Shell a open source sa stal v roku 2014. V apríli o rok neskôr bol premenovaný do aktuálnej podoby. Hlavnou výhodou Electronu je to, že umožňuje využívať existujúce znalosti webového vývoja a aplikovať ich pri tvorbe desktopových aplikácií. Aplikácie vytvorené pomocou Electronu bežia na operačných systémoch Windows, macOS a Linux.

#### 5.1.1 index.html

Súbor `index.html` je základným súborom pre aplikáciu, ktorá je vyvíjaná v Electrone. Obsahuje HTML kód, ktorý definuje štruktúru a obsah hlavného okna. Sú v ňom definované základné prvky ako hlavička HTML dokumentu, ktorá obsahuje CSS štýly, skripty, meta značky a iné informácie, ktoré sú potrebné pre správne zobrazenie a fungovanie okna. V tele dokumentu sú definované jednotlivé prvky aplikácie, ako tlačidlá, rozbaľovacie ponuky (ang. *dropdowns*), či video prehrávač. Okrem toho je súčasťou súboru aj kód jazyku JavaScript, ktorý slúži na zachytávanie udalostí vyvolanými používateľom.

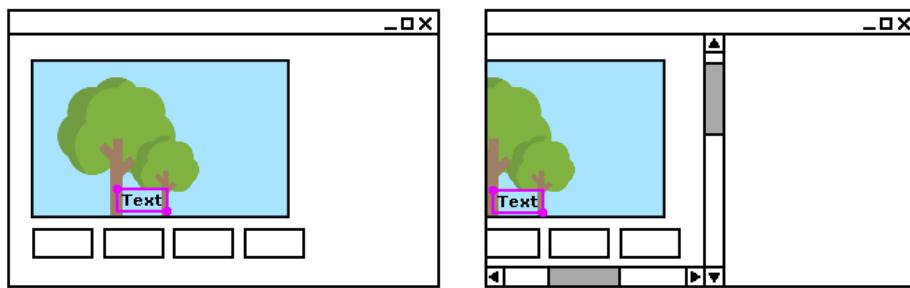
#### Vytváranie bounding boxu

Pri práci s videom v JavaScripte je možné pridať event listener (slovensky voľne preložené ako *poslucháč udalostí*, pojem sa v slovenskom jazyku často nepoužíva), ktorý reaguje na udalosti, najmä kliknutie myšou. Tento event listener má ako parameter objekt `MouseEvent`, v ktorom sa nachádzajú rôzne informácie o udalosti, ktorá nastala. Jednou

z týchto informácií sú súradnice, na ktoré bolo kliknuté – konkrétnie sa jedná o vlastnosti `clientX` a `clientY`, ktoré udávajú pozíciu kurzoru myši vzhľadom k oknu aplikácie. Pre získanie presných súradníč je potrebné zahrnúť aj veľkosť posunu (ang. *offset*) v rámci okna aplikácie. Táto pozícia sa získava pomocou vlastnosti `pageXOffset` alebo `pageYOffset` na globálnom objekte `window`. Z týchto hodnôt sú vypočítané súradnice dvoch bodov. Príklad získania jednej súradnice ľavého x-ového bodu je zobrazená v rovnici 5.1.

$$Lx = event.clientX + window.pageXOffset \quad (5.1)$$

Ako je zobrazené na obrázku 5.1, správne vytvorenie bounding boxu (slovensky preložiteľné ako hraničný obdĺžnik, slovensky pojem sa nevyužíva) je zaistené aj v prípade, že používateľ otvorí aplikačnú konzolu a video nie je viditeľné na obrazovke celé.



Obr. 5.1: Na obrázku na ľavej strane je zobrazená celá snímka videa s vytvoreným bounding boxom, ktorý slúži na vyznačenie titulkov. Tento bounding box je vytvorený pomocou event listenera, ktorý reaguje na kliknutie myšou a získava súradnice kurzoru myši vzhľadom k videu v aplikácii, braný je ohľad aj na posun ako je ukázané v rovnici 5.1. Pri obrázku na ľavo je toto posunutie 0. Obrázok napravo ukazuje, že správne vytvorenie bounding boxu je zaistené aj v prípade, že používateľ otvorí aplikačnú konzolu a video nie je viditeľné na obrazovke celé – je to zaistené pričítaním hodnôt `window.pageXOffset`, resp. `window.pageYOffset`.

### Posúvanie a menenie veľkosti bounding boxu

Po vyznačení oblasti titulkov (kapitola 4.2.1) je vytvorený obdĺžnik, ktorý však z rôznych dôvodov nemusí zaberať celú oblasť titulkov, a preto je možné zmeniť jeho veľkosť a presunúť ho tak, aby boli titulky pokryté celé. Obdĺžnik obsahuje dva tahacie body – bod v ľavom hornom rohu slúži na zmenu pozície obdĺžnika a druhý bod v pravom dolnom rohu slúži na zväčšenie, či zmenšenie obdĺžnika.

Pomocné funkcie naslúchajú udalostiam tahania na danom HTML prvku úchytného bodu. Počas zmeny pozície prvkmu sa prepisuje hodnota pozície obdĺžnika podľa toho, či je presúvaný ľavý horný, alebo pravý dolný roh.

#### 5.1.2 renderer.js

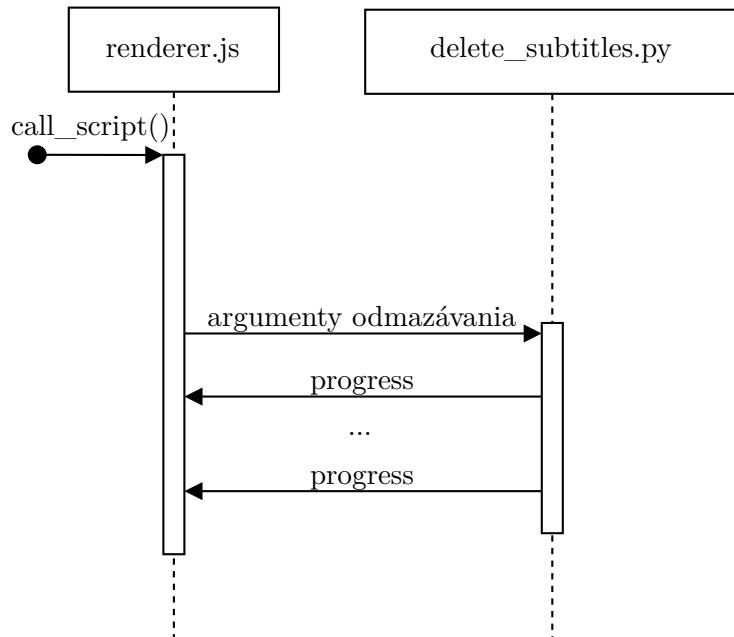
Súbor `renderer.js` je nevyhnutný pre spustenie skriptu napísaného v jazyku Python. Skript sa spúšta s pomocou Node.js modulu `child_process`, ktorý umožňuje spúštať externé programy a komunikovať s nimi prostredníctvom prúdu vstupu či výstupu.

V tomto súbore je definovaná funkcia `call_script()`, ktorá spúšta externý skript s názvom `delete_subtitles.py` (obr. 5.2). Funkcia sa spúšta s určitými argumentami, ktoré sú

predané pythonovému skriptu. Medzi argumenty nevyhnutne predávané na príkazový riadok patria:

- ľavá horná x-ová súradnica označujúca pozíciu boxu nad titulkami,
- ľavá horná y-ová súradnica označujúca pozíciu boxu nad titulkami,
- pravá dolná x-ová súradnica označujúca pozíciu boxu nad titulkami,
- pravá dolná y-ová súradnica označujúca pozíciu boxu nad titulkami,
- cesta k súboru videa,
- výška videa,
- šírka videa,
- metóda odstránenia,
- technika odstránenia,
- hodnota kroku udávajúca, kolko snímok od seba budú detegované titulky.

Súbor ďalej obsahuje kód, ktorý sleduje výstup z pythonového skriptu. Konkrétnie ide o stav progress baru, ktorý je zobrazovaný pre používateľa v modálnom okne a o časovú informáciu o približnej dĺžke trvania odmazávania titulkov. Po dokončení skriptu je táto udalosť oznamená v GUI aplikácii. Video s príponou `_noSUB_yesSOUND.mp4` je následne uložené v rovnakom priečinku ako pôvodné video.



Obr. 5.2: Na sekvenčnom diagrame je zobrazená komunikácia medzi GUI aplikáciou a pythonovým skriptom. Pri spustení skriptu sú odoslané argumenty príkazového riadku nevyhnuté pre odmazávanie titulkov. Skript späť pravidelne odosiela informáciu o postupe odmazávania.

## 5.2 Použité knižnice jazyka Python

Jazyk Python [10] bol vytvorený v skorých 90-tych rokoch minulého storočia pánom Guido van Rossum. Je vytvorených mnoho knižníc a nástrojov pre Python, ktoré robia z jazyka vhodný nástroj pre širokú škálu úloh. V tejto sekcií sú popísané najdôležitejšie knižnice týkajúce sa bakalárskej práce.

### 5.2.1 OpenCV

Ako hovoria G. Bradski a A. Kaehler vo svojej knihe [3], OpenCV (Open Source Computer Vision Library) je open source knižnica, zameraná na počítačové videnie. Je napísaná v jazyku C a C++ a je určená pre rôzne operačné systémy, vrátane systémov Linux, Windows a Mac OS X. OpenCV podporuje rôzne programovacie jazyky vrátane Pythonu, Javy a MATLABu, čo umožňuje ľahké použitie knižnice v rôznych aplikáciach a projektoch. OpenCV poskytuje široké spektrum funkcií pre rozpoznávanie a sledovanie objektov, spracovanie videozáznamov a obrazu, kalibráciu kamery či napríklad aj detekciu tváre.

OpenCV je veľká knižnica, no pre túto prácu sú relevantné len niektoré jej podčasti, ako rozdelenie videa na snímky či práca s obrazom. Nasleduje zoznam dôležitých funkcií a ich popis.

#### **cv2.VideoCapture()**

Funkcia slúži na načítanie videa zo súboru. Uloží video do premennej, ktorá je ďalej spracovávaná.

#### **cv2.VideoWriter()**

Táto funkcia slúži na zápis videa do súboru. Prijíma štyri argumenty, ktorými sú názov výstupného súboru, kam sa má video zapísť, kompresný kodek použitý pri zápisе videa, počet snímok za sekundu a rozmer videa.

#### **cv2.rectangle()**

Ako bolo spomenuté v kapitole 4.1.1, pri metóde odmazávania Remove All je vytvorená maska pre všetky snímky rovnaká. Pomocou funkcie `cv2.rectangle()` je nad čiernym pozadím masky vykreslená oblast podľa parametrov prijatých z príkazovej riadky. Ide o súradnice ohraničenia boxu, ktoré si používateľ zvolil.

#### **cv2.inpaint()**

Táto funkcia je uplatňovaná na každú snímku, z ktorej majú byť odmazané titulky, pokiaľ je vybraná technika Inpainting NS alebo Inpainting Telea. Prijíma masku obsahujúcu oblast na zamazanie a podľa vybranej metódy prijíma konštantu `cv2.INPAINT_NS` alebo `cv2.INPAINT_TELEA`.

#### **cv2.GaussianBlur()**

Pri zvolení techniky odmazania Gaussian Blur má funkcia 3 parametre. Prvým je vstupný obrázok, na ktorý sa aplikuje Gaussov filter, druhý je veľkosť jadra Gaussovoho filtra, ktorá

určuje, ako veľký priestor sa má použiť na vyhľadenie. V tejto práci sa používa jadro o veľkosti  $151 \times 151$  pixelov, ktoré vizuálne vyzerá najlepšie (obr. 5.3). Posledným parametrom je špecifikácia smerodajnej odchýlky v smere osi X a Y, ktorá je nastavená na nulu a funkcia ju automaticky určí na základe veľkosti jadra.



Obr. 5.3: Rôzne veľkosti jadra sú porovnané pri odmazávaní rovnakej časti videa z obrázku 4.7. Veľkosť použitého jadra je uvedená pri každej snímke, pričom ako najlepší výsledok je zvolené jadro o veľkosti  $151 \times 151$  pixelov.

### `cv2.medianBlur()`

Poslednou dôležitou funkciou je funkcia na aplikovanie medianového filtra na daný snímok. Funkcia zistí medián všetkých pixelov v oblasti jadra a nahradí stredný prvok jadra touto mediánovou hodnotou. Veľkosť jadra bola zvolená ako hodnota 71. Porovnanie s inými veľkosťami je na obrázku 5.4.



Obr. 5.4: Obrázok zobrazuje porovnanie snímok po uplatnení mediánoveho filtra na oblasť titulkov zo snímky 4.7. Veľkosť jadra 71 vyšla ako najvhodnejšia pre zamazanie titulkov.

### 5.2.2 Keras-OCR

Keras-OCR je knižnica strojového učenia pre rozpoznávanie znakov (OCR – Optical Character Recognition), ktorá využíva Pythonovú knižnicu Keras. V bakalárskej práci je knižnica Keras-OCR nevyhnutnou súčasťou pri odmazávaní titulkov metódou **Subtitles Only**. Text sa v snímkach rozpoznáva vždy len vo vybranej oblasti zvolenej používateľom a uloží sa do zoznamu. Teoreticky by bolo možné rozpoznávanie textu aj na celej veľkosti snímky, no bolo by to veľmi časovo náročné.

### 5.2.3 SequenceMatcher

SequenceMatcher knižnica v Pythone poskytuje funkcie na porovnávanie dvoch sekvencií znakov a zisťovanie, ako sú si podobné. Metóda `ratio()` je jednou z funkcií, ktoré knižnica ponúka. Táto metóda slúži na výpočet koeficientu podobnosti medzi dvoma reťazcami, pričom jeho hodnota je vrátená v tvare desatinného čísla v rozmedzí od 0 do 1. Výpočet je daný ako:

$$\text{koeficient\_podobnosti} = 2.0 * M/T \quad (5.2)$$

kde:

- $M$  je počet zhôd medzi sekvenciami,
- $T$  je celkový počet prvkov v oboch sekvenciách.

Ak sú sekvencie identické, teda majú rovnaké prvky na rovnakých pozíciách, potom počet zhôd  $M$  bude rovný počtu prvkov v sekvenciach a celkový koeficient podobnosti bude 1,0. Ak naopak sekvencie nemajú nič spoločné, počet zhôd  $M$  bude 0 a koeficient podobnosti bude 0,0. Všeobecne platí, že čím je vyššia hodnota koeficientu podobnosti, tým väčšia je podobnosť medzi sekvenciami [9].

Funkcia Keras-OCR (kapitola 5.2.2) nie je vždy presná, a nerozpozná slová úplne dokonalo. Častým problémom je, že funkcia rozpozná slová vo vete „Majte sa, dovidenia.“ raz ako zoznam nasledujúcich slov: [’majte’, ’sa’, ’dovidenia’], inokedy správne ako [’majte’, ’sa’, ’dovidenia’]. Bodka na konci vety bola v prvom prípade rozpoznaná ako písmeno o, a preto je pri porovnávaní, či sa zmenil titulok, potrebné brať ohľad aj na tento problém. Podobnosť vyššie vypísaných zoznamov je 0,66 aj napriek tomu, že pôvodný text bol rovnaký. Vlastný koeficient podobnosti je nastavený ako číslo 0,39, pretože je braný ohľad aj na pozíciu slov v zozname a pri skúšaní rozpoznávania rôznych textov vyšlo toto číslo ako najlepšie. Ked je koeficient z výsledku funkcie `ratio()` väčší ako 0,39, je brané že text sa nezmenil a titulok ostáva rovnaký. Pokiaľ je koeficient menší, text je s najväčšou pravdepodobnosťou zmenený.

### 5.2.4 MoviePy

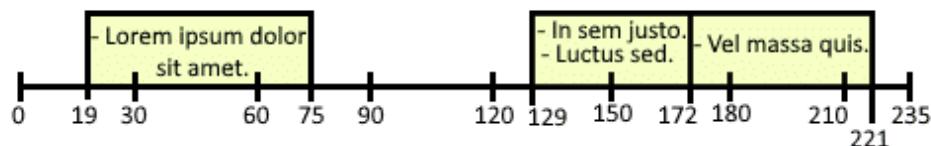
Uložené video bez titulkov má príponu `_noSUB_noSOUND.mp4`, čo značí, že je bez zvuku. Je to z toho dôvodu, že knižnica OpenCV sa primárne zameriava na spracovanie obrazu a nie na prácu so zvukom. Knižnica MoviePy slúži na prácu s videami a zvukovými súbormi v jazyku Python. Pre prácu s videom je potrebné načítať vstupné video pomocou funkcie `VideoFileClip()`, ktorá načíta video súbor bez titulkov a bez zvuku a vytvorí z neho objekt, ktorý je možné použiť na ďalšie operácie. Funkciou `AudioFileClip()` je načítaná zvuková stopa z pôvodného videa s titulkami. Pomocou funkcie `set_audio()` sa pridá zvukový súbor k videu bez titulkov, odstráni sa súbor videa bez zvuku a nový video súbor so zvukom je uložený s príponou `_noSUB_yesSOUND.mp4`.

## 5.3 Hľadanie hraníc titulkov vo videu

Aby bolo zamazávanie titulkov zaručené iba na snímkach, kde sa titulky nachádzajú, musí byť zvolená metóda odmazávania `Subtitles Only`. Nájdenie presných snímok má na starnosti v kóde funkcia `find_exact_frame()`, ktorá využíva metódu bisekcie na zúženie rozsahu hľadania a postupne hľadá snímkový interval, v ktorom sa nachádzajú titulky. Funkcia má štyri vstupné argumenty:

- `start_frame` – číslo začiatočnej snímky,
- `end_frame` – číslo koncovej snímky,
- `start_text` – text na začiatočnej snímke,
- `end_text` – text na koncovej snímke.

Snímkový interval je zapisovaný do premennej s názvom `all_info_subtitles`. Je to vnorený zoznam (ang. *nested list*), ktorý obsahuje zoznamy v tvare [ [a,b,c], [a,b,c], ...]. Tieto zoznamy obsahujú informácie o (ne)výskytu titulkov, písmeno **a** prezentuje začiatočnú snímku, písmeno **b** snímku koncovú a písmeno **c** obsahuje buď nulu alebo jednotku, pričom nula nesie informáciu o tom, že nič sa na danom intervale snímok odmazávať nemá, a jednotka hovorí, že zo snímok v zozname sa odmazávajú titulky. Príklad výpočtu vnoreného zoznamu je na obrázku 5.5.



Obr. 5.5: Na obrázku je zobrazený príklad titulkov vo videu spolu s číslom ich začiatočnej a koncovej snímky. Na prázdných miestach sa titulky nenachádzajú. Vnorený zoznam by pre tento prípad vyzeral nasledovne: [[0,18,0], [19,75,1], [76,128,0], [129,172,1], [173,221,1], [222,235,0]], pričom pokiaľ zoznam na poslednom mieste obsahuje nulu, tak snímky ostatú na danom intervale bez zmeny, a číslo jedna znamená, že sa na snímkovom intervale vyskytujú titulky, ktoré je potrebné odmazat.

## Kapitola 6

# Experimenty a zhodnotenie

Kapitola je zameraná na praktické otestovanie aplikácie. Experimenty boli vykonané na štyroch rôznych videách na operačnom systéme Windows 10 s verziou Pythonu 3.9.2. Všetky testované videá boli stiahnuté zo stránky YouTube s licenciou Creative Commons.

Video číslo jedna a dva má dĺžku 58 sekúnd a je to trailer k dobrodružnému filmu – druhé video má však oveľa hustejšie titulky, než to prvé. Tretie video o dĺžke 5 minút a 4 sekundy je krátka animovaná rozprávka. Posledné štvrté video má dĺžku 10 minút a 2 sekundy a zobrazuje scenérie z prírody. Titulky nad všetkými videami sú k obsahu nerelevantné a sú v pseudolatinčine.

Každé video predstavuje odlišný typ obsahu, čím je zabezpečené pokrytie rôznych situácií, ktoré môžu nastať pri používaní aplikácie. Trailer k filmu predstavuje rýchle a dynamické zmeny scén, čo testuje schopnosti aplikácie prispôsobiť sa rýchlym zmenám. Krátka animovaná rozprávka obsahuje jasné kontúry objektov a animácií, video s prírodnými scenériami testuje aplikáciu v situáciách, kedy sú scény na obrazovke viac pokojné.

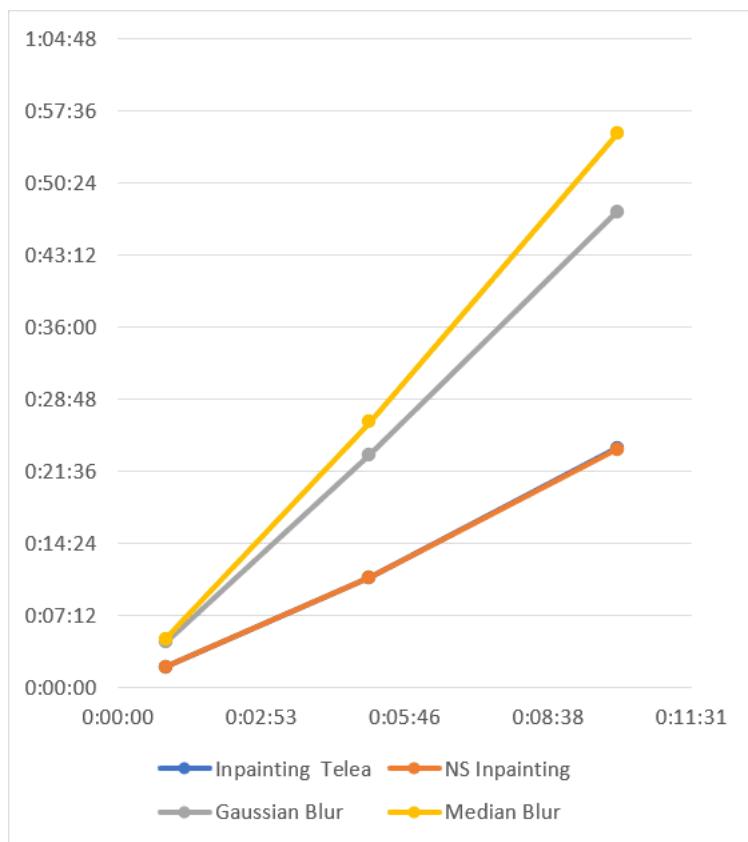
### 6.1 Doba trvania odmazávania

Testovanie prebiehalo na stolnom počítači s procesorom AMD Ryzen 5 2600 a nebolo vykonávané v kontrolovanom prostredí. Výsledky sú preto iba orientačné, a doba behu programu bude na rôznych zariadeniach iná. Všetky štyri videá sú testované na každej metóde a každej technike odmazávania.

Veľmi veľký vplyv na čas odmazávania má množstvo titulkov vo videu. Pri metóde **Remove All** v tabuľke 6.1 je vidieť, že videá číslo 1 a 2 majú časový rozdiel spracovania minimálny, no pri metóde **Subtitles Only** v tabuľke 6.2 je už rozdiel značne viditeľný. Grafické porovnanie výsledných časov týchto dvoch videí je zobrazené na obr. 6.3.

	Remove All			
	Inpainting NS	Inpainting Telea	Gaussian Blur	Median Blur
Video č. 1	00:02:01	00:02:01	00:04:28	00:04:48
Video č. 2	00:02:15	00:02:12	00:04:24	00:04:48
Video č. 3	00:10:55	00:10:57	00:23:12	00:26:30
Video č. 4	00:23:53	00:23:44	00:47:31	00:55:20

Tabuľka 6.1: Tabuľka zobrazuje výsledné časy odmazávania titulkov jednotlivými technikami pri metóde Remove All. Technika Inpainting NS má časové spracovanie takmer rovnaké ako technika Inpainting Telea. Techniky Gaussian Blur a Median Blur trvajú približne dvakrát tak dlho než oba inpaintingy. Výsledky sú zobrazené graficky na obr. 6.1.



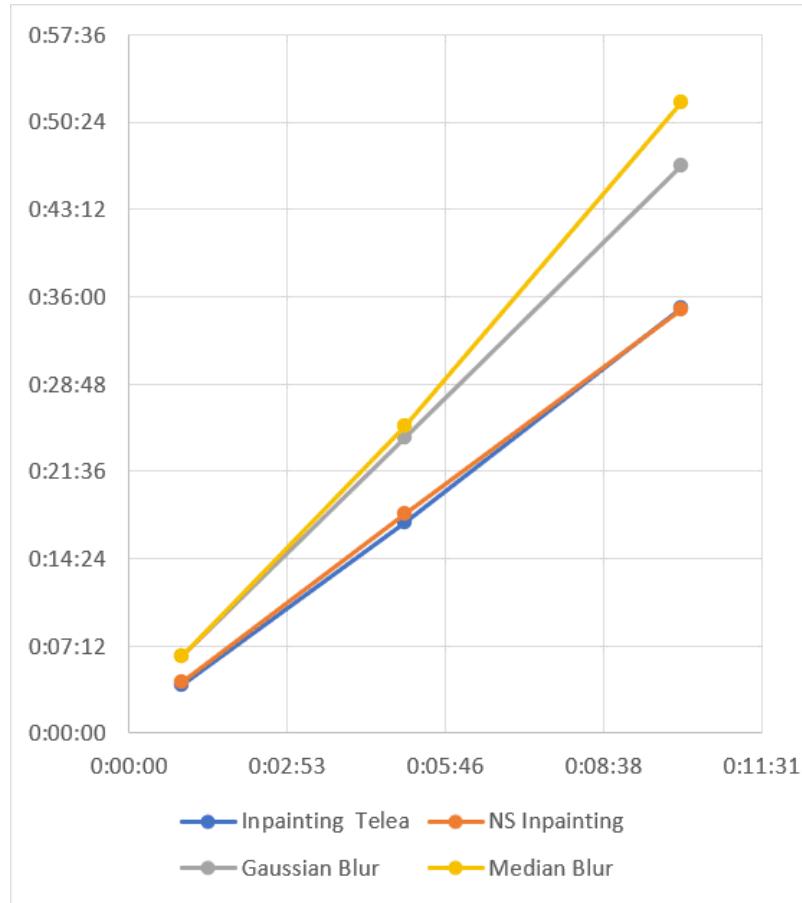
Obr. 6.1: Graf zobrazuje vzťah dĺžky videa a doby spracovania pri metóde Remove All. Video č. 2 na grafe zobrazené nie je, pretože rozdiel oproti videu č. 1 je zanedbateľný.

Ako je viditeľné na obr. 6.1, doba trvania rastie približne lineárne, z čoho by bolo možné extrapolovať aj približné trvanie odmazávania z dlhšieho videa. Pri 58 sekundovom videu a metóde Remove All je priemerný čas odmazania 3 min a 22 s, čo značí, že priemerné odmazávanie jednej sekundy videa trvá 3,48 sekúnd. Pri videu s dĺžkou 5 min a 4 s je priemerný čas odmazania 17 min a 53 s. Sekunda videa sa priemerne odstraňovala 3,52 s. Titulky z posledného videa o dĺžke 10 min a 2 s sa odstraňovali priemerne 3,75 s. Z toho vyplýva, že priemerná doba odstraňovania titulkov pri metóde Remove All 3,58 sekúnd na jednu sekundu videa. Tento čas po zaokrúhlení je používaný na vypočítanie približnej dĺžky trvania odmazávania v aplikácii pri danej metóde.

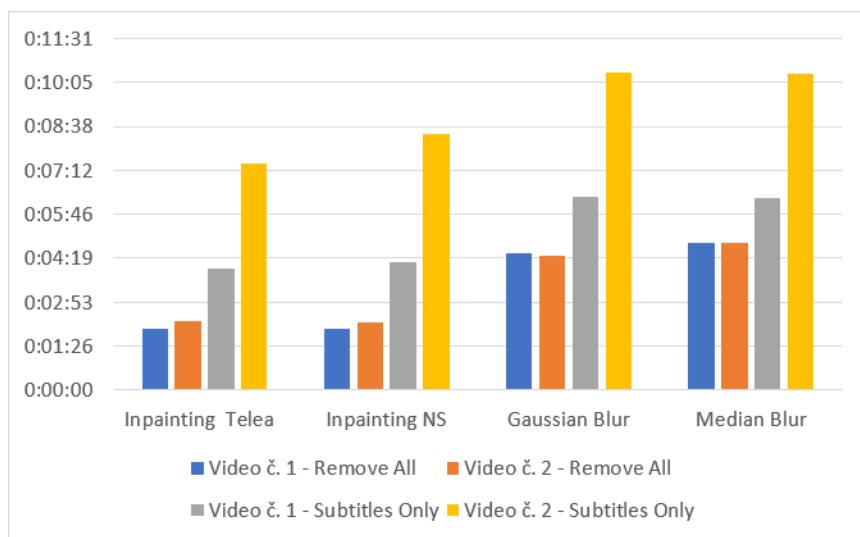
	Subtitles Only			
	Inpainting NS	Inpainting Telea	Gaussian Blur	Median Blur
Video č. 1	00:03:59	00:04:11	00:06:19	00:06:17
Video č. 2	00:07:26	00:08:24	00:10:24	00:10:22
Video č. 3	00:17:22	00:18:08	00:24:25	00:25:19
Video č. 4	00:35:04	00:34:54	00:46:45	00:52:03

Tabuľka 6.2: Pri vybranej metóde bolo testovanie uskutočnené s východzou hodnotou kontroly titulkov vo videu. Zvýšenie tejto hodnoty by značilo kratšie odmazávanie a naopak zníženie hodnoty by malo za dôsledok dlhší čas odmazávania, pretože by sa text musel rozpoznať na snímkach častejšie. Rozdiel medzi časom odmazávania medzi videami č. 1 a 2 je veľmi veľký aj napriek tomu, že obe videá majú dĺžku 58 s. Je to z dôvodu častejšieho výskytu titulkov pri videu č. 2 kde je potrebné rozpoznať a odmazávať viac textu.

Priemerná doba odmazávania pri metóde **Subtitles Only** (obr. 6.2) pri 58 sekundových videách je 7 min a 10 s, čo je odmazanie titulkov zo sekundy videa za 7,41 sekúnd. Pri videu s dĺžkou 5 minút a 4 sekundy je priemerná doba odmazania 21 minút a 18 sekúnd, čo je priemerne 4,2 sekundy na odstránenie titulkov z jednej sekundy videa. Z posledného videa o dĺžke 10 min a 2 s je odstraňovanie titulkov hotové za priemerne 41 minút a 56 sekúnd, čo je 4,17 sekundy potrebných na odmazanie jednej sekundy. Z výpočtov je dané, že priemer na odstránenie titulkov z jednej sekundy videa je 5,26 sekundy. Tento priemer zaokruhlený nahor je použitý pre výpočet priemerného času odmazávania vybranou metódou.



Obr. 6.2: Graf zobrazuje vzťah dĺžky videa a doby spracovania pri metóde **Subtitles Only**. Video č. 2 na grafe zobrazené nie je a výsledky sú lineárne ako aj pri metóde **Remove All**.



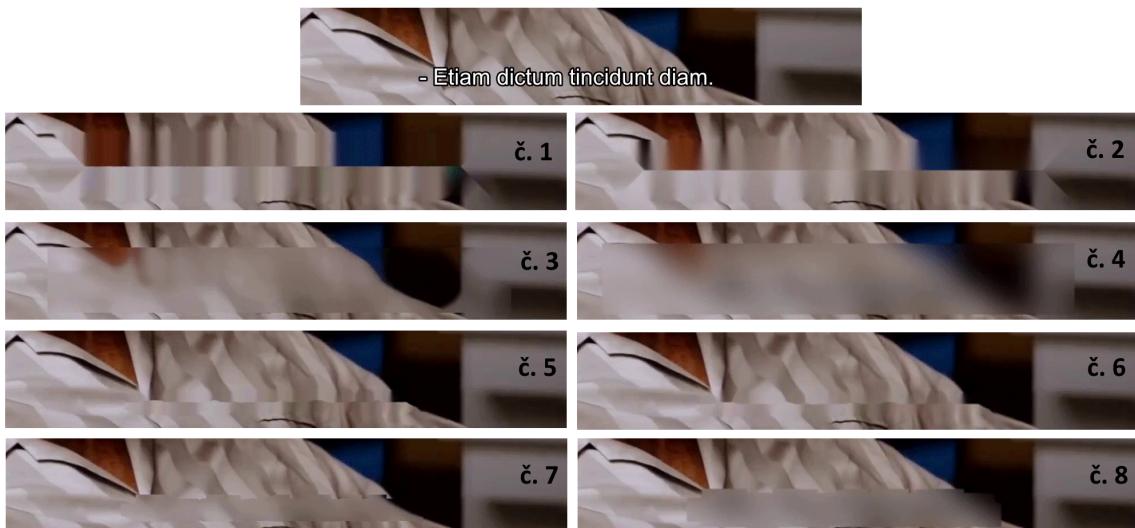
Obr. 6.3: Na grafe je zobrazené porovnanie dĺžky odstraňovania titulkov na 58 sekundových videách. Video č. 1 obsahovalo menej titulkov než video č. 2 a táto skutočnosť sa odrazila na výsledných časoch.

## 6.2 Kvalita odmazávania

V tejto časti sú porovnané snímky z videí č. 1, 3 a 4. Porovnania jednotlivých odmazávaní sú na obrázkoch 6.4, 6.5, 6.6 – všetky obsahujú 9 snímok, pričom 8 z nich je výsledok po odmazávaní rôznom technikou a metódou. Schéma zobrazenia jednotlivých výsledkov je v tabuľke 6.3.

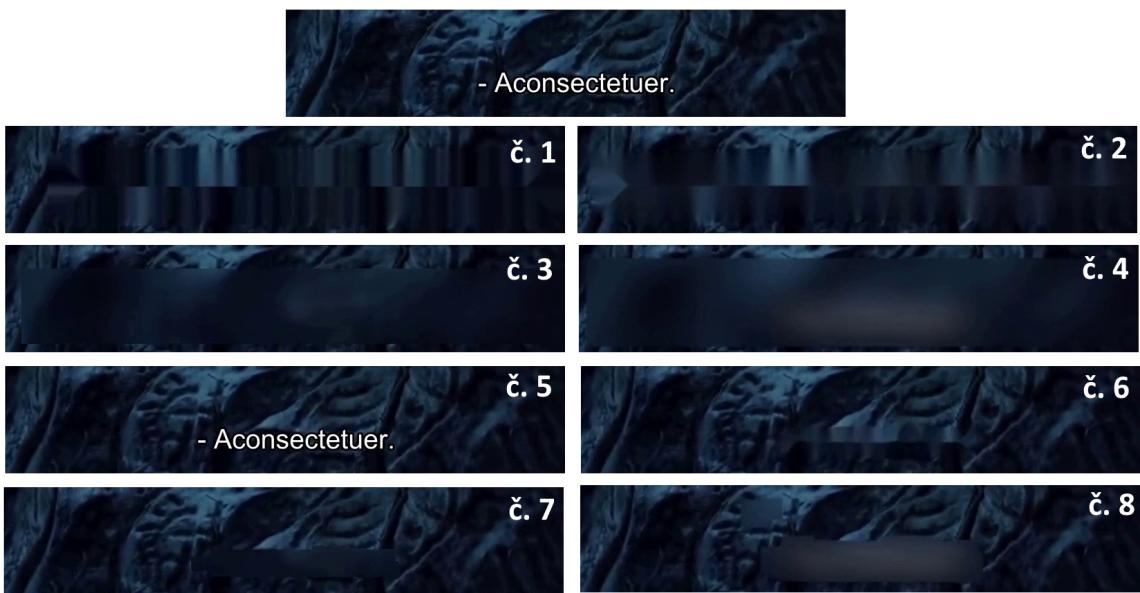
originál	
č. 1 – Remove All – Inpainting NS	č. 2 – Remove All – Inpainting Telea
č. 3 – Remove All – Median Blur	č. 4 – Remove All – Gaussian Blur
č. 5 Subtitles Only – Inpainting NS	č. 6 – Subtitles Only – Inpainting Telea
č. 7 – Subtitles Only – Median Blur	č. 8 – Subtitles Only – Gaussian Blur

Tabuľka 6.3: Tabuľka zobrazuje schému týkajúcu sa obrázkov 6.4, 6.5, 6.6.

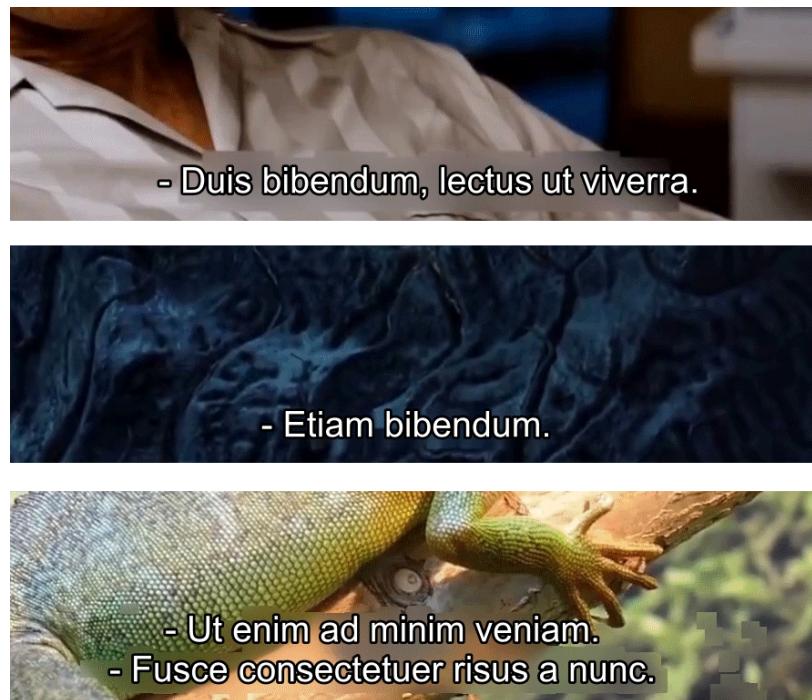


Obr. 6.4: Na obrázkoch je zobrazené zamazanie titulkov všetkými technikami a metódami z testovaného videa č. 1. Výsledky sú v tomto prípade uspokojivé a predpokladané, ako najlepšie zamazanie sa javí číslo 6, čo je technika odstránenia **Inpainting Telea** pri metóde **Subtitles Only**.

Vo všeobecnosti má najlepšiu kvalitu zamazania metóda **Subtitles Only**, pretože oblasť zamazávania je menšia ako pri technike **Remove All**. Z jednotlivých techník vzhľadovo najlepšie vyzerá pri väčších plochách technika **Median Blur**, pri menších oblastiach sú vhodnejšie oba inpaintingy. Pokial sú titulky odstraňované z dôvodu pridania titulok vlastných, pôvodné zamazané oblasti sú mälo viditeľné a nerušivé (obr. 6.7).



Obr. 6.5: Obrázky ukazujú odmazanie na videu č. 3. Na ukážke č. 5 titulky zamazané bohužiaľ neboli kvôli chybe v rozpoznávaní textu. Veta „Aconsectetuer.“ bola raz rozpoznaná správne ako [’Aconsecteuer’] a inokedy nesprávne ako [’Aconsectetuero’]. Podobnosť týchto dvoch refázov je 0, a titulky preto neboli odmazané. Pri viacslovných vetách tento problém nenastáva, pri jednoslovných sa však bohužiaľ občasne vyskytuje (viac v kapitole 5.2.3). Okrem tohto defektu sú zvyšné výsledky vyhovujúce, pričom ako najvhodnejšie techniky vyzierajú Inpainting Telea a Median Blur pri metóde Subtitles Only.



Obr. 6.7: Obrázok zobrazuje snímky s pridanými novými titulkami nad odmazanú oblast. Sú ľahko čitateľné, a zamazaná oblasť za nimi nepôsobí rušivo.



Obr. 6.6: Obrázok ukazuje výsledky odmazávania titulkov na vybranej snímke videa č. 4. Pri metóde odmazávania **Subtitles Only** je viditeľné, že maska zaberá väčšie okolie, než tesne okolo titulkov. Je to z toho dôvodu, že predchádzajúce titulky napojené na aktuálne boli dvojriadikové a maska bola vytvorená tak ako je ukázané na obrázku 4.5. Najlepšie odmazanie sa javí pri metóde **Subtitles Only** a technike odmazávania titulkov **Median Blur**.

# Kapitola 7

## Záver

Cieľom práce bolo navrhnuť techniky odmazávania titulkov z videa a prakticky tieto technicky implementovať vo forme aplikácie.

Popísané a implementované boli štyri techniky odmazávania titulkov zo snímky – dve sú založené na inpaintingu a ďalšie dve využívajú obrazové filtre na rozmazanie vybranej oblasti.

Práca popísala a implementovala optimalizovanú metódu na detekciu textu vo videu s využitím bisekcie, ktorá umožňuje výrazne skrátiť čas spracovania oproti detekcii textu na každom snímku. Táto metóda takisto umožňuje zmenšiť odmazávanú oblasť pomocou vyrábania masky, ktorá tesne obaľuje text. Používateľ si môže zvoliť, či chce alebo nechce zvoliť optimalizovanú metódu na základe charakteru videa a jeho preferencií.

Výsledná aplikácia je vytvorená pomocou webových technológií a je desktopová s využitím frameworku Electron. Samotná manipulácia s videom a snímkami je implementovaná v jazyku Python s využitím knižníc OpenCV, Keras-OCR a MoviePy.

Aplikácia podporuje formát videa MP4 rôznych rozmerov a rôznych počtom snímok za sekundu. Primárne sú podporované titulky písané v latinskej abecede, ale s malými obmedzeniami (väčšia odmazávaná plocha a väčšie množstvo rozmazených snímok) funguje na ľubovoľný jazyk. Poradí si aj s rôznymi farbami textu titulkov.

Používateľ si pri používaní aplikácie nahrá video zo svojho zariadenia, vyberie si oblasť snímky, v ktorej sa titulky vo videu vyskytujú a následne si vyberie metódu a techniku odmazávania podľa vlastných preferencií.

Experimentálne boli vyskúšané rôzne videá a bola sledovaná doba behu aplikácie pri ich odmazávaní. Závislosť dĺžky videa, použitej metódy, techniky a doby behu bola popísaná v kapitole o experimentoch. Takisto bola analyzovaná kvalita odmazania titulkov rôznych metod a techník na videách rôzneho žánru.

Všetky nájdené predošlé existujúce riešenia mali jednu z dvoch nevýhod – buď boli platené, či už jednorazovo alebo paušálne; alebo neboli primárne určené na odmazávanie titulkov, ale iba na odmazávanie vodoznakov alebo log. Oproti existujúcim riešeniam je výsledný program tejto práce zverejnený na internete s otvoreným zdrojovým kódom, a teda je zadarmo a dá sa voľne modifikovať zdrojový kód.

# Literatúra

- [1] BERTALMIO, M., BERTOZZI, A. a SAPIRO, G. Navier-Stokes, fluid dynamics, and image and video inpainting. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* Február 2001, zv. 1, s. I–355. DOI: 10.1109/CVPR.2001.990497. Dostupné z: [https://www.researchgate.net/publication/3940597\\_Navier-Stokes\\_fluid\\_dynamics\\_and\\_image\\_and\\_video\\_inpainting](https://www.researchgate.net/publication/3940597_Navier-Stokes_fluid_dynamics_and_image_and_video_inpainting).
- [2] BRADLEY, D. a ROTH, G. Adaptive Thresholding using the Integral Image. *J. Graphics Tools*. Január 2007, zv. 12, s. 13–21. DOI: 10.1080/2151237X.2007.10129236. Dostupné z: [https://www.researchgate.net/publication/220494200\\_Adaptive\\_Thresholding\\_using\\_the\\_Integral\\_Image](https://www.researchgate.net/publication/220494200_Adaptive_Thresholding_using_the_Integral_Image).
- [3] BRADSKI, G. a KAEHLER, A. *Learning OpenCV: Computer Vision with OpenCV Library*. 1. vyd. United States of America: O'Reill Media, Inc., september 2008. ISBN 978-0-596-51613-0.
- [4] CHAPRA, S. C. a CANALE, R. P. *Numerical Methods for Engineers*. 6th. New York, NY: McGraw-Hill Education, 2010. 124-132 s. ISBN 9780073401065.
- [5] CHATTERJEE, P., JANA, S. a GHOSH, S. Comparative Study of OpenCV Inpainting Algorithms. 2021. Dostupné z: [https://globaljournals.org/GJCST\\_Volume21/2-Comparative-Study-of-OpenCV.pdf](https://globaljournals.org/GJCST_Volume21/2-Comparative-Study-of-OpenCV.pdf).
- [6] DÍAZ CINTAS, J. Handbook of Translation Studies. Volume 1. [online]. John Benjamins. 2010, [cit. 2023-4-17]. Dostupné z: [https://www.academia.edu/22522144/2010\\_Subtitling](https://www.academia.edu/22522144/2010_Subtitling).
- [7] ELHARROUSS, O., ALMAADEED, N., AL-MÁADEED, S. a AKBARI, Y. Image inpainting: A review. *CoRR*. 2019, abs/1909.06399. Dostupné z: <http://arxiv.org/abs/1909.06399>.
- [8] EVANS, D. *How to Remove Subtitles from Video Permanently* [online]. iMyFone, 14. marca 2022 [cit. 2023-01-17]. Dostupné z: <https://filme.imyfone.com/watermark/remove-subtitle-from-video/>.
- [9] FOUNDATION, P. S. *Difflib — Helpers for computing deltas: Documentation*. [cit. 2023-07-05]. Dostupné z: <https://docs.python.org/3/library/difflib.html>.
- [10] FOUNDATION, P. S. *History and License: History of the software* [online]. [cit. 2023-05-09]. Dostupné z: <https://docs.python.org/3/license.html?highlight=history>.
- [11] GENDRIN, C. Chemical imaging and chemometrics for the analysis of pharmaceutical solid dosage forms. November 2008. Dostupné z: <https://theses.hal.science/tel-00341106/document>.

- [12] GONZALEZ, R. C. a WOODS, R. E. *Digital image processing*. 3. vyd. Upper Saddle River, N.J.: Prentice Hall, 2008. 760-764 s. ISBN 9780131687288 013168728X 9780135052679 013505267X.
- [13] HWUNG, C. *Easy Steps to Remove Subtitles From MP4, MKV, AVI, etc* [online]. Digiarty Software, marec 2023 [cit. 2023-05-02]. Dostupné z: <https://www.videoproc.com/video-editor/remove-subtitles-from-mp4.htm>.
- [14] JANÁKOVÁ, I. *Filtrace šumu a poruch* [online]. 2023. Dostupné z: [http://vision.uamt.feec.vutbr.cz/ZVS/lectures/07\\_Filtrace\\_sumu\\_a\\_poruch.pdf](http://vision.uamt.feec.vutbr.cz/ZVS/lectures/07_Filtrace_sumu_a_poruch.pdf).
- [15] KAMUNYA, T. *Introduction to Electron JS: Complete Guide and Learning Resources* [online]. Geekflare, 21. decembra 2022 [cit. 2023-04-04]. Dostupné z: <https://geekflare.com/electron-js-learning-resources/>.
- [16] KARAMITROGLOU, F. A Proposed Set of Subtitling Standards in Europe. UMIST, Manchester, UK. [cit. 2023-04-17]. Dostupné z: <https://translationjournal.net/journal/04stndrd.htm>.
- [17] LONG, S., HE, X. a YAO, C. Scene Text Detection and Recognition: The Deep Learning Era. *CoRR*. 2018, abs/1811.04256. Dostupné z: <http://arxiv.org/abs/1811.04256>.
- [18] PAZHOOHI, F. a KINGSTONE, A. The Effect of Movie Frame Rate on Viewer Preference: An Eye Tracking Study. *Augmented Human Research*. December 2021, zv. 6. DOI: 10.1007/s41133-020-00040-0. Dostupné z: [https://www.researchgate.net/publication/348679933\\_The\\_Effect\\_of\\_Movie\\_Frame\\_Rate\\_on\\_Vi...](https://www.researchgate.net/publication/348679933_The_Effect_of_Movie_Frame_Rate_on_Vi...)
- [19] RUSS, J. C. *The image processing handbook*. 4. vyd. CRC Press, 2002. ISBN 0-8493-1142-X.
- [20] TELEA, A. An Image Inpainting Technique Based on the Fast Marching Method. *Journal of Graphics Tools*. Január 2004, zv. 9. DOI: 10.1080/10867651.2004.10487596. Dostupné z: [https://www.researchgate.net/publication/238183352\\_An\\_Image\\_Inpainting\\_Technique\\_Based\\_on\\_the\\_Fast\\_Marching\\_Method](https://www.researchgate.net/publication/238183352_An_Image_Inpainting_Technique_Based_on_the_Fast_Marching_Method).

## Príloha A

# Obsah priloženého pamäťového média

V priečinku `LaTeX/` sa nachádzajú zdrojové súbory k tomuto PDF dokumentu. V súbore `README.md` sa nachádza návod na použitie aplikácie. V koreňovom adresári sa taktiež nachádza video popisujúce bakalársku prácu spolu s ukážkami na príkladoch.

Priečinok `app/` obsahuje spustiteľnú aplikáciu s názvom `Subtitles_remover_8_5.exe`. Zdrojový kód k aplikácií spolu so zoznamom závislostí pre Python uložených v `requirements.txt` a závislostí pre Node.js uložených v `package.json` sa nachádza v priečinku `app/resources/app/`. Frontend aplikácie je obsiahnutý v súboroch `renderer.js`, `main.js` a `index.html`. Súbor `delete_subtitles.py` obsahuje aplikačný backend.

V priečinku `videos/` sa nachádzajú dve sady videí – prvá sada je s titulkami, aby si aplikáciu mohol používateľ vyskúšať aj sám. Druhá sada obsahuje 2 rôzne videá. Na každom z videí boli titulky odmazané oboma metódami odmazávania, pričom pri metóde odmazávania s názvom `Subtitles Only` boli použité všetky 4 techniky odmazávania titulkov a pri metóde `Remove All` bola použitá na ukážku technika `Median Blur`.

```
CD-ROM
├── app/
│   └── resources/
│       └── app/
│           ├── delete_subtitles.py
│           ├── index.html
│           ├── main.js
│           ├── package.json
│           ├── renderer.js
│           └── requirements.txt
└── Subtitles_remover_8_5.exe

└── LaTeX/
    └── README.md

└── videos/
    ├── removed_subtitles/
    │   ├── video1_gauss.mp4
    │   ├── video1_median.mp4
    │   ├── video1_ns.mp4
    │   ├── video1_remove_all_median.mp4
    │   ├── video1_telea.mp4
    │   ├── video2_gaus.mp4
    │   ├── video2_median.mp4
    │   ├── video2_ns.mp4
    │   ├── video2_remove_all_median.mp4
    │   ├── video2_telea.mp4
    └── with_subtitles/
        ├── video1.mp4
        ├── video2.mp4
        ├── video3.mp4
        └── video4.mp4
└── video.mp4
```