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

ČASOVÝ SNÍMEK Z OBRAZU STACIONÁRNÍ KAMERY

TIME LAPSE FROM STATIONARY CAMERA IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ANDREA FICKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Studentka: **Ficková Andrea**
Program: Informační technologie
Název: **Časový snímek z obrazu stacionární kamery**
Time Lapse from Stationary Camera Image
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte problematiku pořizování časového snímku (time lapse) z videa/kamery.
2. Vyberte několik dostupných dohledových kamer streamujících na internet vhodných pro pořizování časového snímku. Na těchto datech proveďte kritiku omezení primitivního časového snímku. Pořiďte vlastní vývojová a testovací data.
3. Navrhněte algoritmy zpracování videa vedoucí k překonání omezení primitivního časového snímku.
4. Implementujte systém, který bude online přijímat videostream a zaznamenávat pokročilý časový snímek. Vytvořte uživatelské rozhraní pro ovládání/nastavení tohoto systému.
5. Vytvořte několik demonstračních časových snímků demonstrující různé pokročilé aspekty vytvořeného systému.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Johannes Peter Kopf, Michael F. Cohen, Richard Szeliski: Hyper-lapse video through time-lapse and stabilization, US Patent US10002640B2, 2014
- Jason N. Laska, Greg R. Nelson, Greg Duffy, Hiro Mitsuji, Lawrence W. Neal, Cameron Hill: Method and system for generating a smart time-lapse video clip, US Patent US9170707B1, 2014
- Seonghyeon Nam, Chongyang Ma, Menglei Chai, William Brendel, Ning Xu, Seon Joo Kim: End-To-End Time-Lapse Video Synthesis From a Single Outdoor Image, CVPR 2019
- Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, Jiebo Luo: Learning to Generate Time-Lapse Videos Using Multi-Stage Dynamic Generative Adversarial Networks, CVPR 2018
- Moein Shakeri, Hong Zhang: Moving Object Detection in Time-Lapse or Motion Trigger Image Sequences Using Low-Rank and Invariant Sparse Decomposition, ICCV 2017

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cieľom tejto práce je navrhnúť systém, ktorý minimalizuje či dokonca eliminuje obmedzenia časozberného videa zaznamenaného stacionárnou kamerou. Zamerala som sa na tri obmedzenia, ktoré som mala možnosť odpozorovať: nepokojné objekty, prudké svetelné zmeny a nežiadúce objekty. Vznikajúce obmedzenia kazia dojem videa a zapríčiňujú nesústredenosť na predmet pozorovania. V práci som navrhla a implementovala systém, ktorý dokáže spracovať už nasnímané video uložené na disku a nahrávať časozberné video zo streamu. Cieľom navrhnutého systému je aplikovať jeden z algoritmov na jemu príslušné obmedzenie. Hlavným výsledkom je vyprodukovanie časozberného videa, ktorého sledovanie bude pre ich pozorovateľa príjemnejšie.

Abstract

The goal of this thesis is to design a system which minimizes or even eliminates the restrictions of time-lapse video captured by a stationary camera. I focused on the three of these restrictions, that I had the opportunity to observe: restless objects, abrupt illumination changes, and undesirable objects. The appearance of restrictions spoils the impression of video and causes a distraction for the object of observation. Within this thesis I designed and implemented a system that can process a captured video stored in the computer and capture time-lapse video from a video stream. The goal of the designed system is to apply one of the algorithms to the corresponding restriction. The main result is the production of a time-lapse video, which will be more pleasant for their observers to watch.

Kľúčové slová

časozberné video, časozber, časová snímka, stacionárna kamera, obmedzenia, nepokojné objekty, svetelné zmeny, detekcia pohybu, eliminácia obmedzení

Keywords

time-lapse, time-lapse video, stationary camera, restrictions, restless objects, luminance changes, motion detection, elimination of restrictions

Citácia

FICKOVÁ, Andrea. *Časový snímek z obrazu stacionární kamery.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Časový snímek z obrazu stacionární kamery

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána prof. Ing. Adama Herouta, Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

. . . . . .
Andrea Ficková
28. mája 2020

Poďakovanie

Rada by som poďakovala svojmu vedúcemu tejto práce, pánovi prof. Ing. Adamovi Heroutovi, Ph.D., za jeho odborné vedenie, rady, prístup, podporu a trpezlivosť počas tvorby tejto práce. Poďakovanie taktiež patrí mojej rodine a kolegom z VUT FIT, ktorí ma podporovali nie len počas zhotovovania tejto práce, ale aj počas celého môjho bakalárskeho štúdia.

Obsah

Kapitola 1

Úvod

Časozberné video umožňuje sledovať v krátkom čase vývoj dlhotrvajúcich udalostí alebo deja. Môže slúžiť na pozorovanie populácie zvierat, ich spôsob života, sledovanie rastu flóry alebo vývoj buniek. Avšak jeho využitie sa dá nájsť aj mimo vedné odbory. Pre mnohých ľudí je tvorba takýchto videí koníčkom a dá sa pojať veľmi kreatívnym spôsobom.

V tejto práci prebieha analýza toho, čo sa bude diať vo videu, ak bude zaznamenaná snímka napr. za sekundu času a následné prehrávanie zaznamenaných snímiek bude v niekoľkonásobne vyššej frekvencii. Práca sa zameriava na zaznamenávanie časozberného videa stacionárnou kamerou, čo znamená, že kamera je objektívom namierená na jedno miesto alebo objekt, ktorý je predmetom pozorovania.

Objektami pozorovania vo videách, ktoré som zhotovila, boli väčšinou pohyby ľudí, vozidiel, prechod mračien, meniace sa svetlo, chvenie stromov vo vetre, horiaca sviečka, či kvitnúce kvety a pod. Tieto deje boli dlhotrvajúce a ich doba zaznamenávania trvala aj niekoľko hodín. Ak by boli tieto deje zobrazené v bežnom videu ich priebeh by bol plynulý, avšak v časozbernom videu tomu tak nie je. Vyskytujú sa v ňom prudké zmeny svetla alebo pohybu, ktoré jeho kvalitu sledovania znižujú. Takéto zmeny sa označujú ako obmedzenia časozberného videa. Na zhotovených videách som odpozorovala tri obmedzenia: nepokojnosť objektov, prudké svetelné zmeny a nežiadúce objekty. Cieľom práce je tieto javy minimalizovať či dokonca eliminovať, vďaka čomu sa divák môže sústrediť na dej, kvôli ktorému bolo časozberné video vytvorené. Prínosom tejto práce sú navrhnuté algoritmy, ktoré tieto javy potláčajú a pre ich implementáciu som využívala prevažne knižnicu OpenCV, ktorá slúži pre prácu s videom a snímkami.

Práca je rozdelená do niekoľkých kapitol. V druhej kapitole opisujem časozberné video, jeho využitie, približujem vyskytujúce sa obmedzenia a spôsoby zaznamenávania takého videa. Tretia kapitola je o použitých nástrojoch pre túto prácu, kde opisujem aj knižnicu OpenCV, korektúru snímiek, ktorá bola potrebná pre túto prácu a zhotovovanie vlastného datasetu. Vo štvrtej kapitole píšem o návrhu API, čo zahŕňa zhotovenie časozberného videa, snímanie videa zo streamovacích kamier a spôsob eliminácie obmedzení. Tento návrh je bližšie opísaný v piatej kapitole z hľadiska implementácie. V nej uvádzam použité funkcie implementované mnou alebo tie, ktoré sú súčasťou knižníc. V šiestej kapitole som tieto algoritmy zhodnotila a v poslednej siedmej kapitole je celkovo zhrnutá práca spolu s návrhmi na možné vylepšenia a rozšírenia vytvoreného systému.

Kapitola 2

Časozberné video

Časozber, *časozberné video* alebo anglicky *time-lapse* [7] (z latinského slova *lapsus*, čo znamená „prechod časom“), umožňuje zobraziť dej omnoho rýchlejšie než je jeho reálny priebeh. Rýchlosť snímaného deja závisí od preferencie užívateľa, teda koľkonásobne chce priebeh deja zrýchliť.

Bežné video, ktorým môže byť napr. film v televízii, v kine, podomácky zhotovené video a pod., je snímané a spätne prehrávané frekvenciou minimálne 24 snímiek za sekundu (fps, frames per second). Sledovanie obrazu, na ktorom prebieha prehrávanie snímiek v takejto rýchlosti, vníma mozog ako plynulo prebiehajúci dej. Fps videa záleží na nastavenej frekvencii zariadenia, prípadne v nastavení pri renderingu videa alebo filmu. Časozberné video je vo svojej podstate rovnaké ako klasické video, líši sa len spôsobom zaznamenávania. Počas nahrávania takého videa sa zaznamenáva snímka s frekvenciou raz za sekundu, minútu, hodinu, či dokonca deň. Snímka zaznamenaná za takúto jednotku času sa nazýva *časová snímka*. Tieto snímky sú spätne prehrané v bežnej frekvencii minimálne 24 fps, tým pádom sa dej videa prejaví vo videu ako zrýchlený.

Ak by chcel niekto pozorovať kvet od zasadenia cez jeho klíčenie až po rozkvitnutie, musel by predtým kvetom sedieť príliš dlho než by odpozoroval výrazné zmeny. Iná situácia vzniká, ak by bola kvetina zasadená a jej pestovateľ by sa na ňu prišiel pozrieť pravidelne raz za nejaký čas, napr. raz za 5 hodín, alebo raz za 12 hodín a pod. V tedy by mohol spozorovať výrazné zmeny a teda raz by spozoroval vyklíčenú stonku, neskôr puky až po niekoľkých dňoch rozkvitnutý kvet. Takýto princíp platí u časozberného videa, kedy by si pestovateľ odfotil tento kvet raz za nejakú dobu a vyprodukoval video zo zriadených fotografií.

Podobný prípad sa týka rastu detí. Rodičia zvyknú byť so svojim dieťaťom každý deň a len ťažko si všimnú zmenu v tom, ako vyrástli. Zatiaľčo na návštevu príde raz za mesiac ich teta a všimne si, ako veľmi za posledný mesiac sa ich výška zmenila. Z toho vyplýva, že ak má človek niečo alebo niekoho pred očami každý deň, tak si malých zmien nemusí povšimnúť. Ak však rodič bude zhotovovať fotku dieťaťa každý deň počas celého mesiaca a zo všetkých fotiek vytvorí video, spozoruje, akými zmenami za ten mesiac dieťa prešlo. Rýchlosť rastu dieťaťa, samozrejme, záleží na veku, čím sú staršie tým menšie zmeny nastávajú za určitú dobu. Podobných prípadov je oveľa viac, no tie sú zhrnuté v sekcii 2.1.

2.1 Využitie časozberného videa

Časozberné video je možné použiť pre efektívnu kompresiu času. Dôvody, prečo vznikajú časozberné videá sú rôzne. Závisí od toho, čo je žiadané alebo potrebné pozorovať a následne

zdokumentovať. Tieto videá môžu byť zhotovené amatérsky alebo profesionálne kvôli hobby, alebo vede. Časozberom je možné sledovať vývoj alebo pokrok:

- technického charakteru: stavanie budovy, mosta, lode, a pod.,
- umeleckého charakteru: kresba alebo maľba diela,
- prírodovedeckého charakteru: západ alebo východ slnka, pohyb mračien, kvitnutie stromov alebo kvetov,
- a iných udalostí.

Vyprodukované videá môžu slúžiť nie len pre technickú stránku veci (sledovanie pokroku stavby budovy) ale aj pre relax (väčšinou videá prírody), hlavne ak majú zvukový podklad. Stavba budovy alebo vývoj rastliny môže trvať niekoľko mesiacov až rokov a vďaka zachyteniu snímky raz za určitú dobu, je možné sledovať tieto deje od začiatku po samotný koniec behom pár minút. Takto je možné sledovať udalosti, ktoré v reálnom živote nie je možné zachytiť. Dá sa potom určiť, po akej dobe nastáva nejaká výrazná zmena na sledovanom objekte, napr. po koľkých dňoch začala klíčiť rastlina.

Časozberné video našlo uplatnenie aj v športe [10], hlavne pre žurnalistov. Tí môžu zaznamenať nejakú dôležitú udalosť v zápase a čas, kedy daná udalosť nastala, napr. v akom čase padol gól. Čas môžu zaznamenať podľa časomiery nachádzajúcej sa na obraze záznamu zápasu. Takto zápas môžu zdokumentovať v článku bez sledovania celého zápasu trvajúceho príliš dlho.

Aj rodinný videoalbum sa dá obohatiť časozberným videom napr. rastu detí (obr. 2.1), rastu bruška tehotnej mamičky alebo cesty na výlet zachytenej z okna auta.
Časozbery našli využitie aj vo vedeckých výskumoch [16], napr.:

- monitorovanie populácie netopierov [6], z čoho sa dá predpokladať, že sa používa pre monitorovanie populácie akéhokoľvek druhu zvierat,
- pozorovanie buniek [5], mikroskópia,
- monitorovanie energetickej kaskády (energy cascade) oceánu [4]
- sledovanie rastu rastlín.

Počas monitorovania populácie netopierov bolo odpozorované počas akého obdobia v roku sa netopiere objavili na pozorovanom mieste, v akom počte a naopak, kedy netopiere začali kolóniu opúšťať. Taktiež bol zaznamenaný dosiahnutý maximálny počet netopierov, ktorý kolónia dosiahla. Okrem toho bol odpozorovaný výskyt prvých mláďat a to vrátane ich počtu, spôsobu narodenia, či po akom čase prvýkrát vzlietli. Pozorované bolo aj ich správanie sa počas rôznych období roka.

Pri časozbernom snímaní buniek bolo pozorované, čo sa s bunkou deje v určitom čase, prípadne po koľkých dňoch nastáva u bunky nejaký zmenený stav. Môže sa jednať o mutáciu buniek alebo ich delenie.

V tejto sekcii sa nedá spomenúť všetky možnosti využitia, pretože si myslím, že časozberné video je možné vyprodukovať sledovaním čohokoľvek a možnosti sú len tak obmedzené ako kreativita samotného človeka.

Obr. 2.1: Ukážka časových snímiek zachytávajúce rast dieťaťa od malého dievčatka až po tínedžerku [15, 8].

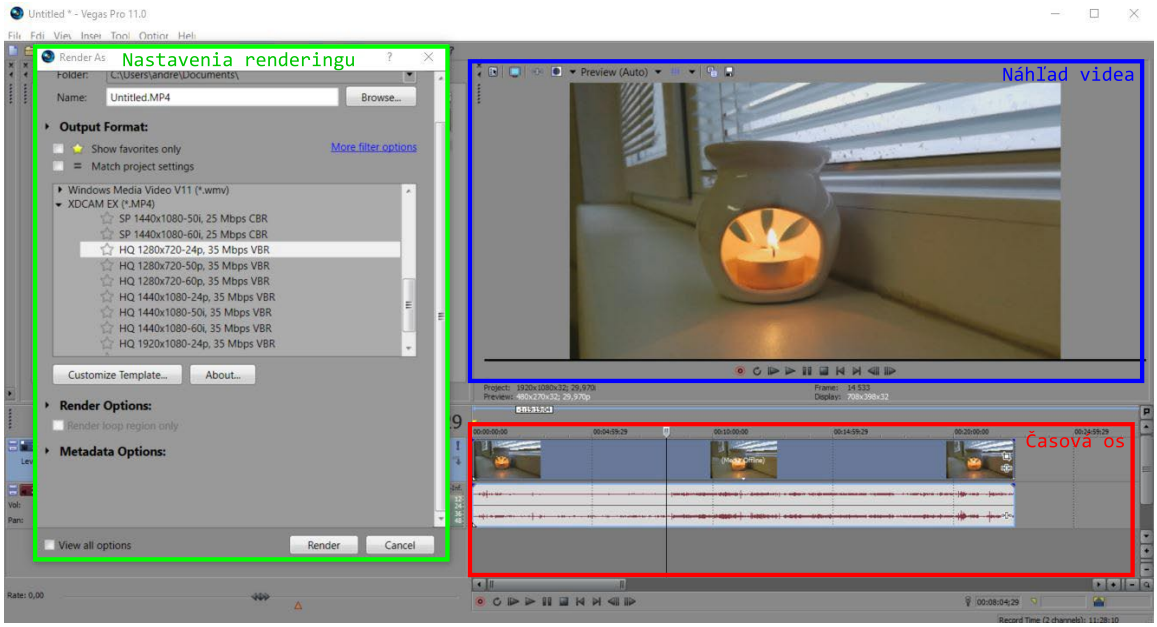
2.2 Spôsoby zaznamenávania časozberného videa

Spôsoby snímania časozberného videa sú rôzne, záleží od techniky, ktorou bude video zaznamenané. Kamerové záznamy sa v dnešnej dobe natáčajú so smartfónom, digitálnym fotoaparátom, zrkadlovkou, kompaktným fotoaparátom, GoPro a inými viac alebo menej profesionálnymi kamerami.

Jedným zo spôsobov, ktorým je možné vytvoriť časozberné video je natočiť ľubovoľným zariadením video záznam akéhokoľvek deja a následne ho zrýchliť. Zaznamenaný dej na videu je možné zrýchliť pomocou video-editačného softvéru na počítači. Softvér pre editáciu videa už môže byť nainštalovaný spolu s operačným systémom, prípadne sa dá zadarmo stiahnuť z internetu alebo zakúpiť. Softvér pre editáciu videa znázorňuje obrázok 2.2, v ktorom je zobrazené prostredie programu Sony Vegas Pro 11[1]. V takomto softvéri si môže užívateľ skrátiť video tak, ako požaduje a následne uložiť renderovaním. Rendering [2] je proces tvorby nového mediálneho súboru, ktorý je založený na zjednotení niekoľkých odkazov ku klipom (obrázky, video, text) uložených na časovej osi editačného softvéru. Rendering neprebieha len v softvéroch pre úpravu videa, ale aj v 2D alebo 3D modelovacích programoch.

Pre zhotovenie časozberného videa je možné využiť aj mód pre vytvorenie časozberného videa, ktorým niektoré zariadenia disponujú. Na každom zariadení táto funkcionalita môže niesť rôzny názov. Po oboznámení sa užívateľa s jeho zariadením sa mód časozberu pravdepodobne nájde. Na obrázku 2.3 je screenshot aplikácie z displaya smartfónu, kde sa tento mód nachádza. Pre vytvorenie videa stačí položiť kameru a namieriť ju na požadovaný objekt alebo miesto, ktoré týmto módom bude snímané. Dôležitá je frekvencia zaznamenania časovej snímky, ktorú je možné nastaviť na požadovanú hodnotu. Avšak nie všetky zariadenia týmto nastavením disponujú, ich frekvencia je napevno nastavená a nemožno ju zmeniť. Snímanie sa väčšinou ukončí manuálne. Po ukončení je časozberné video hotové a uložené v zariadení, či už v internej alebo externej pamäti. Ďalšia úprava vo video-editačnom softvéri

[1] https://www.vegascreativesoftware.com/us/vegas-pro/?_oB=vegas-pro



Obr. 2.2: Editačný program Sony Vegas Pro 11. Na obrázku je znázornené nastavenie renderingu, časová os, na ktorej je možné skladať k sebe videá alebo obrázky, pridávať efekty, text a pod., ktoré budú zrenderované a náhľad videa ako bude vyzerať vo finále.

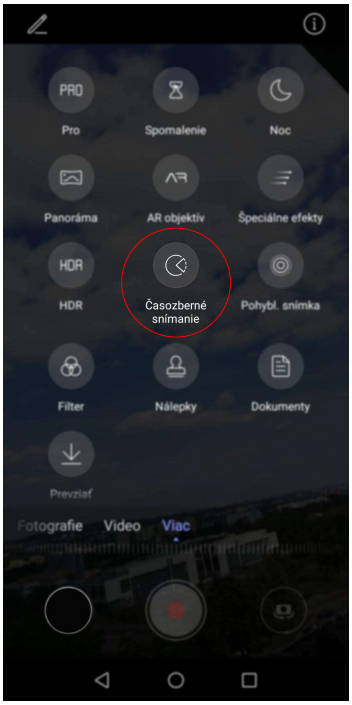
nie je potrebná, ale ak je vyprodukované video dlhšie než je požadované, môže sa skrátiť viac vo vybranom programe. Mód pre vytvorenie časozberného videa nemusia vlastniť niektoré smartfóny. Pre nich existujú aplikácie poskytujúce funkciu pre časozberné snímanie, ktorej sa nastaví požadovaná frekvencia snímania prípadne môžu byť umožnené ďalšie nastavenia, ako je doba snímania, či rozlíšenie videa a pod. Nastavenia takejto aplikácie znázorňuje obrázok 2.4, konkrétne sa jedná o aplikáciu *Time Lapse camera*[2] stiahnutú z *Google Play*. Tieto aplikácie vyprodukujú hotové časozberné video a uložia do pamäti smartfónu.

Ďalším spôsobom je tzv. „photo lapse“, kedy sa zaznamenávajú snímky podľa nastaveného časovača na zariadení. Tomuto časovaču sa určí, ako často má zaznamenať snímku zo snímaného obrazu (napr. raz za sekundu času). Takto bude zariadenie zriaďovať snímky raz za určitú dobu. Z týchto jednotlivých snímiek nie je automaticky zhotovené video a užívateľ si ho musí vyprodukovať sám opäť pomocou softvéru pre editáciu videa. Tento proces vyžaduje manuálne ukladanie snímiek za sebou v danom programe. Treba klásť dôraz na počet snímiek nasledujúcich po sebe v jednej sekunde na časovej osi. Malo by to byť minimálne 24 snímiek. Následne je potrebné video vyrenderovať a časozberné video je hotové. Funkcionalitu photo lapse pre smartfóny poskytuje taktiež aplikácia *Time Lapse camera*, ktorá už bola spomenutá v predchádzajúcom odstavci. Následný zápis snímiek do videa umožňuje aj systém navrhnutý v tejto práci.

2.3 Obmedzenia časozberného videa

Po vyprodukovaní jednoduchého časozberného videa akýmkoľvek spôsobom uvedeného v sekcii 2.2 sa v ňom môžu vyskytnúť javy, ktoré sú buď zbytočné alebo kazia sledujúcemu dojem z celkového videa. Pre príklad uvediem ulicu, ktorá je snímaná stacionárnou kamerou.

[2] https://play.google.com/store/apps/details?id=com.mountaindehead.timelapsproject&hl=en



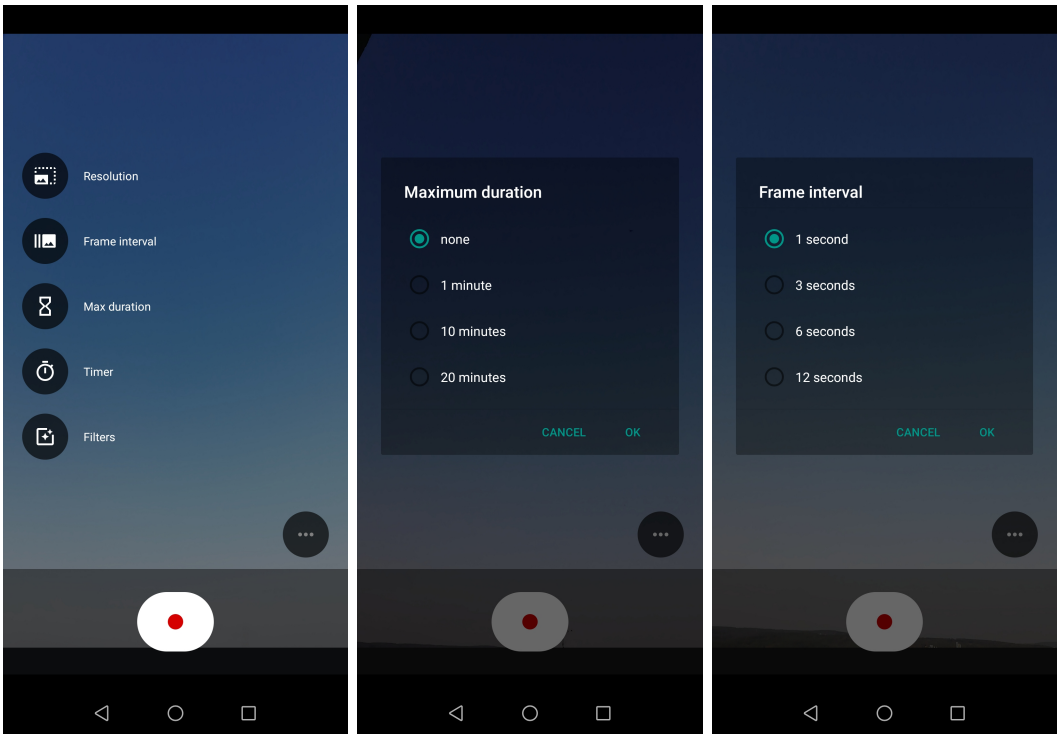
Obr. 2.3: Screenshot nastavení videa v smartfóne s OS Android, kde sa nachádza aj mód pre časozber pod názvom „Časozberné snímanie“.

Na obraze tejto kamery by bolo možné pozorovať rôzne deje, jedným z nich môže byť prechádzajúce auto. V bežnom videu o frekvencii 30 fps by bolo možné pozorovať plynulý pohyb auta v časovom rozmedzí napr. 3 sekúnd na 90 snímkach. V časozbernom videu z tejto ulice, ktoré by bolo snímané o frekvencii 1 fps, by auto bolo zachytené možno na 3 časových snímkach, tým pádom tento dej na videu o frekvencii 30 fps netrvá ani sekundu času. Na videu to vyzerá ako rýchle prebliknutie a nevytvára príjemný efekt pre pozorovateľa, preto je tento jav obmedzením časozberného videa. Príklad takého videa sa nachádza v médiu v súbore `videa/namestie1.mp4`.

Ďalším príkladom môže byť zaznamenávanie časových snímiek v prírode. Na obraze takého videa je možné pozorovať strom alebo kvety, no vplyvom vetra sú v pohybe. V časozbernom videu takýto strom alebo kvet môže vyzerať nepokojne, vykonáva nepokojné pohyby. Video zachytávajúce tento dej je priložené v médiu v súbore `videa/orgovan.mp4`. Počas sledovania mnou zhotovených časozberných videí, som si povšimla tri obmedzenia, na ktoré som sa zamerala:

- nepokojné alebo nekľudné objekty vo videu,
- prudké svetelné zmeny, ktoré spôsobujú blikanie vo videu,
- zbytočné, nežiadúce objekty, napríklad auto uvedené v príklade.

Jednotlivé obmedzenia približujem v kapitole 4, kde poukazujem detailnejšie na to, ako sa prejavujú a taktiež popisujem, ako som sa ich snažila eliminovať a predkladám dosiahnutý výsledok.



Obr. 2.4: Ukážka nastavení mobilnej aplikácie pre tvorbu časozberného videa pre OS Android.

Kapitola 3

Použité znalosti a nástroje

Pre implementáciu navrhnutých algoritmov, o ktorých píšem v kapitole 4, som zvolila programovací jazyk Python. Pre prácu s videom som potrebovala použiť vhodné nástroje a po naštudovaní problematiky spracovania videa v jazyku Python, som zvolila knižnicu OpenCV [19, 1]. Táto knižnica ponúka funkcie nie len pre čítanie, spracovanie a zápis videa, ale aj snímiek.

V tejto kapitole taktiež opisujem *gamma korektúru*, ktorú som zakomponovala do návrhu algoritmu pre elimináciu prudkých svetelných zmien vznikajúcich vo videu.

3.1 Knižnica OpenCV

Knižnica OpenCV[1] (Open Source Computer Vision Library) obsahuje množstvo rôznych algoritmov pre počítačové videnie. Táto knižnica je modulárna štruktúra, zahŕňa niekoľko zdieľaných a statických knižníc. V implementácii som z OpenCV použila moduly pre spracovávanie obrazu, čítanie a zápis obrázka, vstup/výstup videa a základné funkcie definujúce jednoduché dátové štruktúry, ako je pole.

3.1.1 Použité moduly a ich funkcie

Modul pre spracovávanie obrazu

Tento modul zahŕňa funkcie pre spracovanie obrazu, napr. pre filtrovanie obrazu, geometrické transformácie, kresliace funkcie atď. Z tohto modulu som využila funkciu pre zmenu farebného modelu snímky, napríklad z BGR do RGB a naopak, z RGB do HSV a naopak, či z BRG do GRAY atď. Modul disponuje množstvom konverzných kódov uvedených ako parameter funkcie pre konverziu modelu daného obrázka či snímky.

Modul pre čítanie a zápis obrázka

Modul obsahuje funkcie pre čítanie a zápis obrázkov z a do súboru. Funkcia pre čítanie obrázka načíta a vráti obrázok zo súboru, ktorého cesta je uvedená ako jej parameter. Podporuje viacero formátov: `*.jpg`, `*.bmp`, `*.png`, `*.tiff` atď. Typ čítaného obrázka je determinovaný podľa jeho obsahu. Farebný obrázok je čítaný vo farebnom modeli BRG. Jedná sa o RGB model, ktorého kanály sú uložené v opačnom poradí. Avšak funkcii sa môže zadať aj

[1] https://docs.opencv.org/4.3.0/d1/dfb/intro.html

parameter, ktorý určí v akom farebnom modeli obrázok po načítaní vráti. Funkcia pre zápis obrázka má cestu, kam sa obrázok zapíše, uvedenú ako parameter spolu so samotným obrázkom.

Modul pre vstup a výstup videa

Modul ponúka funkcie a triedy pre čítanie a zápis videa, kontrolu či inicializácia vstupného videa prebehla v poriadku a má aj funkcie, ktoré čítané video uzatvárajú. Z tohto modulu som využila triedy pre zachytenie a zápis videa, ktoré sú alfou a omegou pre túto prácu. Trieda pre zachytenie videa umožňuje čítať video zo súboru, zachytiť video z kamery alebo streamu. Modul poskytuje funkciu, ktorú možno zavolať nad vytvoreným objektom spomenutej triedy a získať o videu informáciu na základe vstupného parametru funkcie. Môže to byť fps videa, celkový počet snímiek vo videu, šírka, výška, kodek atď. Z tohto modulu som využila aj funkciu, ktorá číta a vracia snímku po snímke z videa, avšak treba ju použiť v cykle. S vrátenými snímkami sa dá pracovať ako s obrázkami, ktoré vráti funkcia z predchádzajúceho modulu.

Trieda pre zápis videa umožňuje nastaviť vlastnosti (fps, kodek, apod.) pre výstupné video. Pre zápis snímiek do videa modul ponúka funkciu, ktorá sa volá nad objektom triedy pre zápis.

Modul pre jednoduché dátové štruktúry

Tento modul zahŕňa základné dátové štruktúry, napr. multi-dimenzionálne pole a taktiež funkcie pre prácu s nimi, napr. vyhľadávacie tabuľky, ktoré som využila pre mapovanie hodnôt daných gamma korektúrou opísanú v nasledujúcej sekcii.

3.2 Korektúra snímky

Vo všeobecnosti, korektúra alebo vylepšenie snímky pokrýva viacero aspektov: sýtosť, ostrosť, odstránenie šumu, vyváženie tónov, úprava tónov alebo vylepšenie kontrastu. Všetky tieto aspekty je možné ručne vylepšiť v editačných softvéroch pre fotografie. Rovnako, ako softvéry pre editáciu videa, tak aj softvéry pre editáciu fotografie môžu byť nainštalované spolu s operačným systémom, prípadne sa môžu stiahnuť z internetu alebo zakúpiť. Základnú editáciu fotografií poskytujú aj smartfóny.

Pre vylepšenie celkového videa som potrebovala vylepšiť jednotlivé snímky. Konkrétne som potrebovala upraviť ich svetlosť, resp. jas. Keďže som nemohla tisícky snímiek upravovať ručne v softvéri, resp. som potrebovala automatické vykonávanie tejto úpravy, musela som nájsť spôsob, ako túto úpravu uskutočním v programe. Po prieskume som pre modifikáciu jasu našla úpravu pomocou gamma korektúry, ktorá zmení kontrast snímky, čím sa zmení aj jej jas.

Gamma korektúra

Gamma [14] je jedna z charakteristík každého digitálneho obrázka. Definuje vzťah medzi číselnou hodnotou pixelu a svetlosťou daného pixelu. Bez gamma by prejav tieňov digitálneho obrázka či snímky nebol možný. Zabezpečuje zobrazenie svetla a tieňov tak, aby sa snímky podobali čo najviac obrazu, ktoré vidí ľudské oko. Digitálne snímky sú *gamma kódované*.

Gamma je definovaná vzorcom:

$$I_{out} = I_{in}^{\gamma} \tag{3.1}$$

Pričom I_{in} je snímka na vstupe, ktorej jas bude upravený, I_{out} je výstupná snímka s upraveným jasom a γ je koeficient, ktorý určuje, ako sa zmení jas snímky:

$$\gamma \begin{cases} < 1 & \text{jas snímky sa zníži} \\ = 1 & \text{jas snímky sa nemení} \\ > 1 & \text{jas snímky sa zvýši} \end{cases} \qquad \gamma \in \mathbb{R}$$

Hodnota tohto koeficientu je zvolená podľa toho, ako veľmi je vyžadované snímku zosvetliť alebo stmaviť. Gamma korektúru je možné uskutočniť jednoducho v ľubovoľnom programe pre úpravu obrázkov a snímiek. Na obrázku 3.1 je znázornená korektúra gamma v programe Adobe Photoshop CS6[2], kde je vidieť zmenu kontrastu a nastavenie gamma hodnôt. Pôvodné hodnoty jasov pixelov je možné meniť pomocou vyhľadávacej tabuľky (LUT, Look-up table), ktorá mapuje hodnoty v rozmedzí $\langle 0; 255 \rangle$ na ich nové gammou upravené hodnoty. Hodnoty V_i do LUT sa počítajú pomocou nasledujúcej rovnice:

$$V_i = \left(\frac{i}{255}\right)^{\frac{1}{\gamma}} \times 255; \quad i \in \langle 0; 255 \rangle \tag{3.2}$$

Z vytvorenej tabuľky sa pôvodné hodnoty pixelov namapujú na tie nové a ich hodnota sa zmení, vo výsledku je obrázok so zmeneným kontrastom.

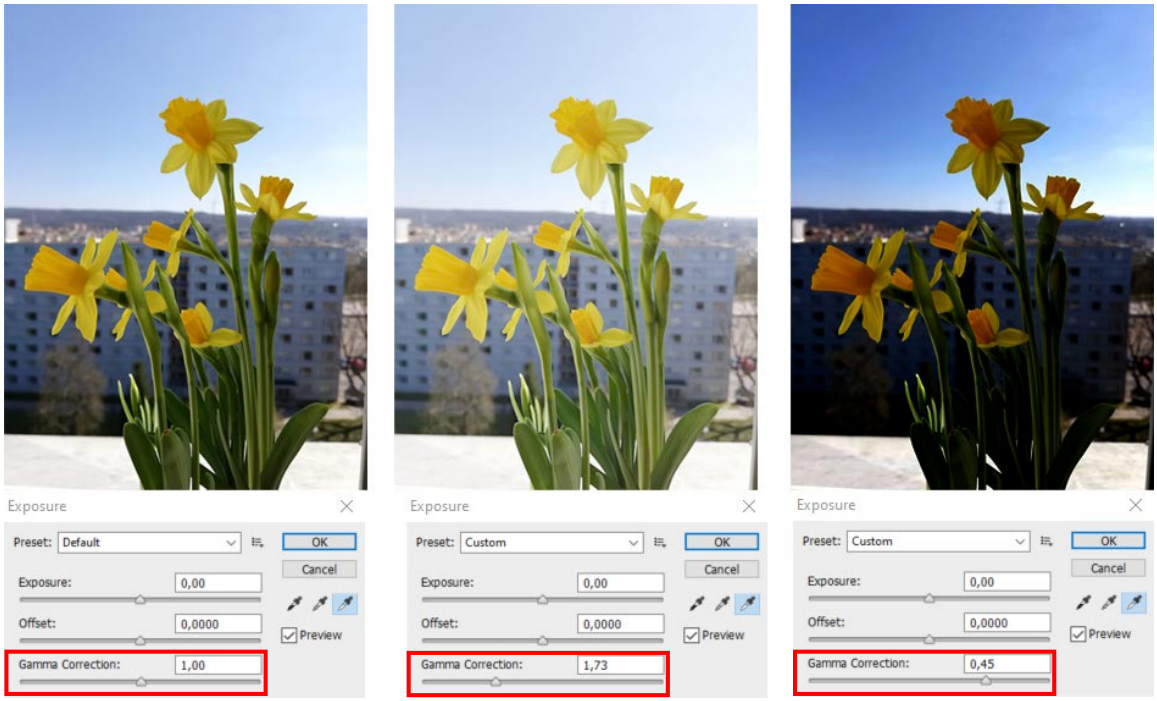
3.3 Zhotovenie datasetu

Na začiatku mojej práce bolo potrebné natočiť niekoľko videí, na ktorých som po vyprodukovaní časozberného videa spozorovala obmedzenia.
Pre ich natáčanie som použila:

- smartfón,
- online dohľadové kamery[3,4]
- IP kameru zapožičanú z VUT FIT.

Pri natáčaní boli zariadenia v statickej polohe. Všetky videá, ktoré som zhotovila, boli dokopy snímané približne 60 hodín. Medzi nich patria východy a západy slnka na rôznych miestach, námestia, a lyžiarske zjazdovky. Nasnímané boli aj iné udalosti, kde je viditeľný progres: kvitnúce narcisy, horiaca sviečka alebo vlastná kresba.

[2] https://www.adobe.com/sk/products/photoshop.html?promoid=PC1PQQ5T&mv=other
[3] https://webkamery-krkonose.cz/
[4] https://snow.cz/webkamery/



Obr. 3.1: Korektúra gamma znázornená na snímke v programe Adobe Photoshop CS6. Snímka naľavo je originál, snímka v strede je po korektúre zosvetlená a snímka napravo je stmavená.

3.3.1 Snímanie so smartfónom

So smartfónom som natáčala klasické videá s frekvenciou 30 fps, ktorých dĺžka odpovedala dobe snímania. Z týchto videí som zhotovila časozberné videá pomocou navrhnutej API. Pre natáčanie časozberného videa mobilom, som použila už spomínanú aplikáciu *Time Lapse camera*[5] z *Google Play*, kde bolo možné nastaviť frekvenciu zaznamenania časovej snímky. Potom som si takéto video zrýchlila ešte viac pomocou navrhnutej API, ak bolo treba, prípadne som pracovala na eliminácii obmedzení, ak boli spozorované.

3.3.2 Snímanie streamu

Videá streamu boli zhotovované IP kamerou alebo online dohľadovou kamerou. Pre snímanie online streamu som vyhľadala na internete webové stránky, ktoré tieto streamy poskytujú. Prevažne sa jednalo o lyžiarské zjazdovky alebo o časti miest.

Videá zhotovené s IP kamerou boli natáčané doma, pretože kamera nedisponovala vlastným zdrojom energie. Napriek tomu sa mi podarilo natočiť zaujímavé videá. Pri zhotovovaní časových snímiek z online streamu som sa riadila postupom, ktorý opisujem v sekcii 4.2.2.

[5] https://play.google.com/store/apps/details?id=com.mountaindehead.timelapsproject

Obr. 3.2: Fotka zo zákulisia snímania časozberného videa IP kamerou.

Kapitola 4

Návrh riešenia

V tejto kapitole sa venujem návrhu API pre zhotovenie časozberného videa a elimináciu jeho obmedzení. Vysvetľujem spôsob jej spustenia a fungovania. Opisujem zhotovenie vlastného datasetu videí, ako sa konkrétne obmedzenia prejavujú a vysvetľujem, ako fungujú jednotlivé algoritmy pre ich minimalizovanie.

4.1 Návrh API

Na vstupe API je video nachádzajúce sa na disku alebo stream z IP prípadne dohľadovej online kamery, alebo súbor snímiek, z ktorých má byť zhotovené video. Video zapísané na disku sa číta prostredníctvom cesty k nemu, ktorá je jedným zo vstupných údajov. Obraz zo streamovacích kamier sa predá programu na čítanie odkazom k samotnému streamu.

Výstupom je video, ktoré je časozberné. Toto video môže byť iba zrýchleným videom alebo takým zrýchleným videom, na ktoré bol aplikovaný algoritmus pre elimináciu alebo minimalizáciu niektorého z obmedzení uvedených v sekcii 2.3. V API je možné určiť, akej dĺžky má byť časozberné video. Pri nahrávaní videa z IP alebo dohľadovej online kamery sa dá určiť aj doba snímania. Tieto údaje sa predávajú programu ako vstupné argumenty.
Všetky parametre, ktoré je možné programu predať, sú následovné:

- cesta k videu alebo odkaz k streamu IP, alebo dohľadovej kamery,
- cesta na disk, kam sa zapíše video,
- dĺžka časozberného videa na výstupe vo formáte `hh:mm:ss`
- doba snímania videa (ak je na vstupe odkaz IP alebo dohľadovej kamery) vo formáte `hh:mm:ss`,
- „prepínač“, ktorým sa volí jedna z operácií: vyprodukovanie časozberného videa z existujúceho videa alebo snímiek, alebo eliminácia niektorého z obmedzení. Pre každé obmedzenie existuje iný prepínač.

Nie všetky parametre musia byť zadané naraz, dokonca nie všetky kombinácie môžu byť spolu na vstupe. Cesta k videu musí byť na vstupe vždy, rovnako ako cesta pre výstupné video, ináč dôjde k chybe programu. Ak nie je zadaná doba snímania z IP alebo dohľadovej kamery, prípadne dĺžka časozberného videa na výstupe, nastane chyba programu. Povinným argumentom je tzv. „prepínač“, ktorým možno zvoliť jednu zo štyroch operácií: jednoduché časozberné video (z videa alebo snímiek), eliminácia nepokojných objektov, prudkých svetelných zmien alebo nežiadúcich objektov.

V jednom behu programu je možné eliminovať iba jedno obmedzenie, preto sa môže uviesť iba jeden z „prepínačov“. Ak by API používal akýkoľvek užívateľ, potreboval by najprv vytvoriť časozberné video a až potom by mohol spozorovať obmedzenie, ktoré by rád odstránil. Takto som postupovala pri spracovaní dát, preto som stanovila nasledujúce opatrenia. Obmedzenia nie je možné eliminovať, ak sa číta obraz streamu. Taktiež nie je možné v jednom behu programu produkovať časozberné video a zároveň aplikovať algoritmus pre obmedzenie. Výnimkou je algoritmus pre elimináciu nepokojných objektov, ktorý bližšie opisujem v sekcii 4.5. Videá sa zapisujú na zvolené miesto na disku vo formáte `.mp4` s frekvenciou takou, akú malo pôvodné video, ak ide o video čítané z disku. V ostatných prípadoch (nahrávanie streamu, zhotovenie videa zo snímiek) je zvolená hodnota 30 fps. Videá sa ukladajú v rovnakom rozlíšení, aké malo vstupné video.

4.2 Čítanie videa z odkazu na vstupe

Veľmi dôležitým vstupným údajom je cesta k videu alebo odkaz k streamu, z ktorého bude vyprodukované časozberné video. V tejto kapitole približujem v akej forme majú byť vstupné údaje a ako získať odkazy pre streamy, ktorých obraz bude snímaný.

4.2.1 Čítanie videa z disku

Čítanie videa z disku je veľmi jednoduché, potrebná je len správna cesta k videu vrátane samotného názvu videa spolu s jeho formátom, napr.:

```
\moja\cesta\k\videu\nazov.mp4
```

Video môže byť zhotovené akoukoľvek kamerou či smartfónom, prípadne použitím tejto API, ktorá zaznamenáva stream z IP kamery alebo online dohľadovej kamery. Dĺžka takého videa môže odpovedať dobe jeho snímania. V tom prípade je video určené pre jednoduché časozberné video. Samozrejme, skrátiť je možné akékoľvek video, aj také, ktoré už časozberné je a požaduje sa urýchliť ho ešte viac. Iba na video čítané z disku je možné aplikovať algoritmy pre elimináciu obmedzení.

4.2.2 Čítanie videa zo streamu

Čítanie videa zo streamu sa od čítania klasického videa líši tým, že nie je konečné. Navyše, cesta k nemu sa získava iným spôsobom. Vstupom môže byť odkaz k obrazu IP kamery alebo online dohľadovej kamery. Vďaka tomu odkazu je možné ich obrazy zaznamenávať.

Odkaz k IP kamere

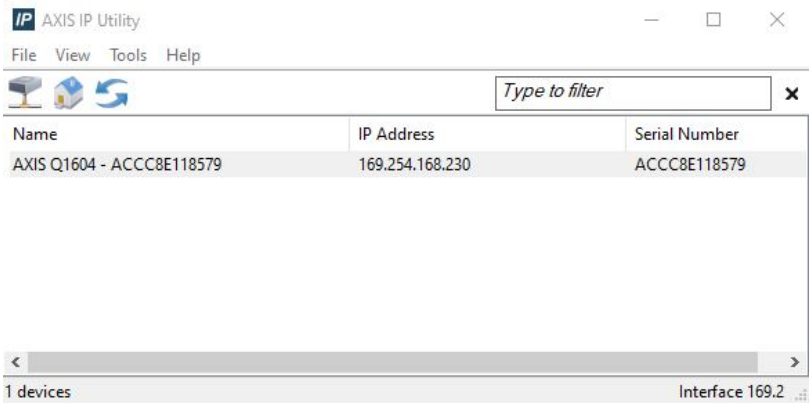
Pre zhotovenie časozberného videa IP kamerou je potrebné predať API správny odkaz, z ktorého program môže čítať obraz kamery. Pre tento odkaz je potrebné vedieť IP adresu, ktorá je kamere priradená po jej zapojení do PC. Adresu je možné zistiť prostredníctvom aplikácie, ktorú poskytuje výrobca danej kamery. Kamera môže byť zabezpečená užívateľským meno a heslom.
Formát vstupného odkazu k obrazu zabezpečenej kamery vyzerá takto:

```
rtsp://username:password@⟨IPadresa⟩/
```

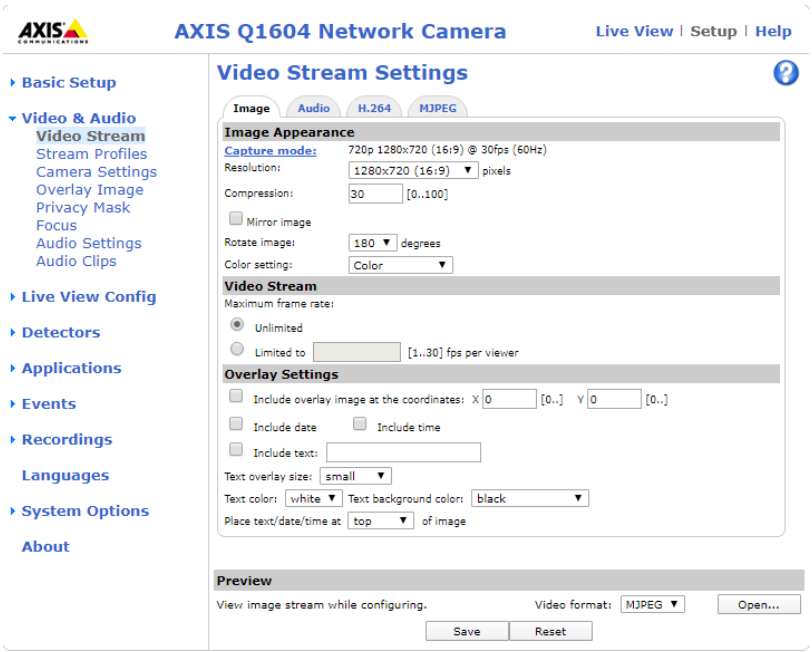
Ak k čítaniu nie je potrebné užívateľské meno a heslo, adresa vyzerá takto:

`rtsp://⟨IPadresa⟩/`

Na zistenie odkazu IP kamery, som použila aplikáciu *AXIS IP Utility*[1]. Rozhranie aplikácie poskytuje tabuľku znázornenú na obrázku 4.1 so všetkými pripojenými IP zariadeniami od spoločnosti *AXIS*, vrátane kamery, ktorú som mala k dispozícii. Po kliknutí na položku v tabuľke aplikácia presunie užívateľa do GUI zobrazeného vo webovom prehliadači, ako znázorňuje obrázok 4.2. V ňom je možné vykonať rôzne nastavenia videa, napr. úprava jasu, kontrastu, fps, rozlíšenie atď. Rozlíšenie alebo fps je taktiež možné nastaviť priamo v odkaze na vstupe. Správnosť odkazu sa dá overiť vo VLC[2] prehrávači stlačením `Ctrl+N`, kedy sa zobrazí okienko, do ktorého sa vloží odkaz. Stream sa spustí v prehrávači, ak je odkaz správny.

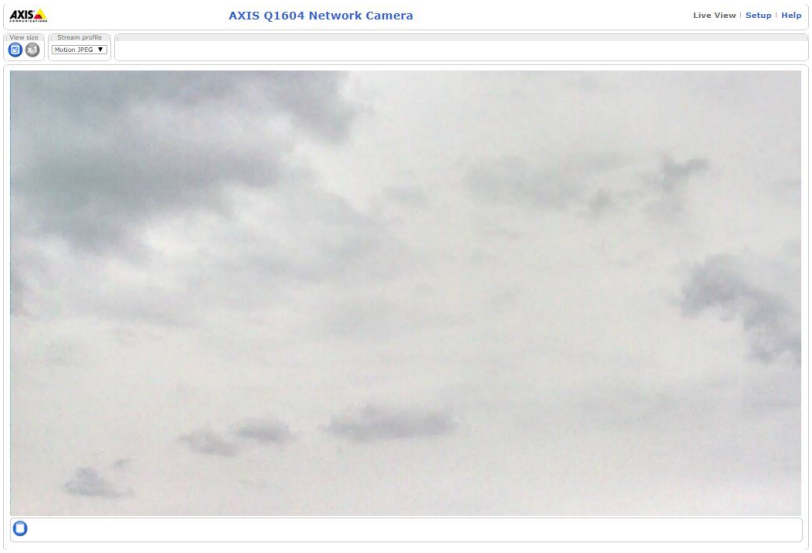


Obr. 4.1: Aplikácia AXIS IP Utility pre získanie IP adresy kamery.



Obr. 4.2: Nastavenia kamery v GUI.

[1] https://www.axis.com/support/downloads/axis-ip-utility
[2] https://www.videolan.org/index.sk.html



Obr. 4.3: Obraz snímaný IP kamerou zobrazený v GUI.

Odkaz k online dohľadovým kamerám

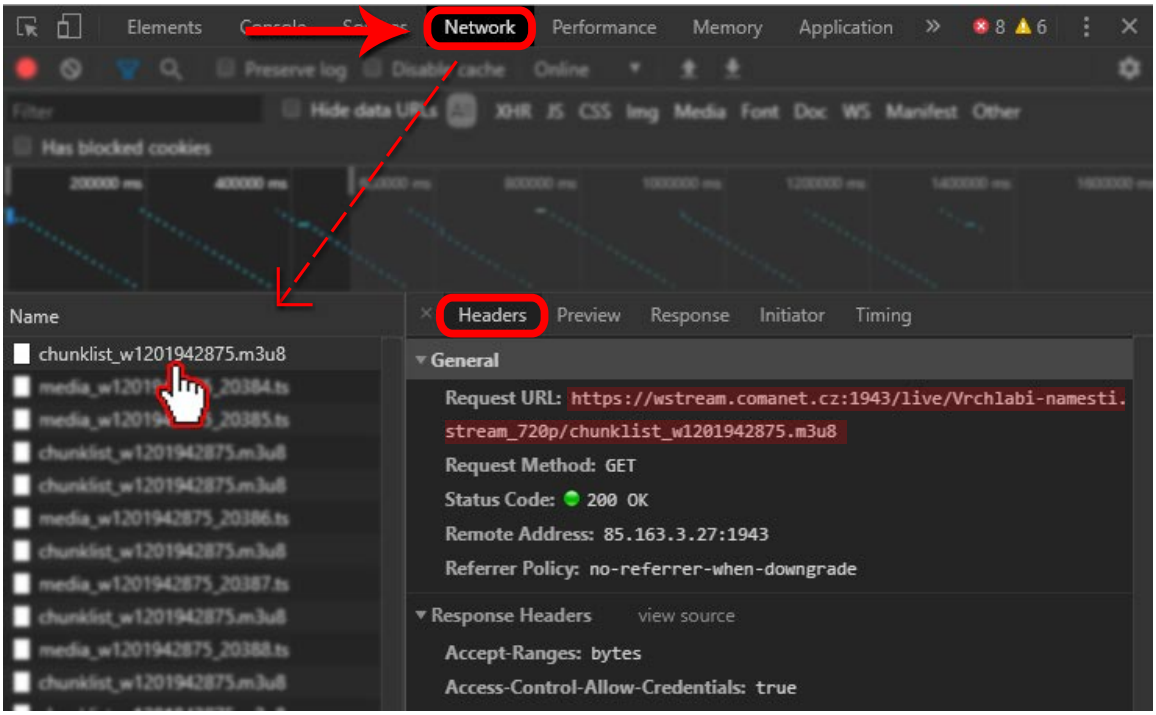
Zaznamenanie časozberného videa z obrazu online dohľadovej kamery je veľmi podobné ako u IP kamery, líši sa len odkazom na vstupe API. Jedná sa o formát videa `.m3u8` [12]. Cesta k tomuto odkazu sa dá získať parsovaním HTML zdrojového kódu webovej stránky, na ktorej beží stream alebo ručne v možnostiach pre vývojára. K týmto možnostiam sa dá dostať po zvolení *preskúmať prvok* z kontextového menu vo webovom prehliadači, v ktorom je otvorená požadovaná webová stránka s online streamom (napr. `https://webkamery-krkonose.cz/`).

V možnostiach pre vývojára sa nachádza ikona pre označenie elementu HTML, ktorou je možné označiť prehrávač obrazu kamery. Postup, ako získať odkaz, znázorňuje obrázok 4.4. Zo sekcie *Elements* je potrebné sa presunúť na sekciu *Network*. Zobrazí sa zoznam, kde sa môžu prvky líšiť od tých, ktoré ilustruje obrázok, každopádne je v ňom stále aspoň jeden s príponou `.m3u8`. Ten treba zakliknúť ľavým tlačidlom myši a zobrazí sa ďalšia tabuľka, kde je potrebné zakliknúť sekciu s HTTP hlavičkami *Headers*. V hlavičke je uložený hľadaný odkaz, ktorý je treba skopírovať a vložiť na vstup programu. Odkaz možno získať aj zobrazením kontextového menu po kliknutí pravého tlačidla myši na spomínaný prvok. V kontextovom menu je potreba zvoliť `Copy > Copy link address`. Obe postupy som realizovala na prehliadačoch *Google Chrome* a *Mozilla Firefox*, ktoré používam v anglickom jazyku.

Ako som už spomenula, k odkazu sa dá dostať aj vyparsovaním z HTML zdrojového kódu pomocou regulárneho výrazu. To znamená, že na vstupe by mohol byť aj odkaz na webovú stránku. Avšak táto metóda sa mi neosvedčila pri všetkých stránkach, z ktorých som videá zaznamenávala, resp. nebolo možné z každého HTML kódu tento odkaz vyparsovať. Istejší spôsob je získať odkaz `.m3u8` ručne, ako už bolo vysvetlené.

4.3 Frekvencia zápisu časovej snímky

Ako už bolo spomenuté, API produkuje časozberné videá z existujúceho videa na disku alebo z obrazu online kamier. Stanovenie správnej frekvencie zaznamenania časovej snímky



Obr. 4.4: Časť prostredia pre vývojára, v ktorej sa nachádza potrebný odkaz pre vstup API na zachytenie obrazu zo streamu.

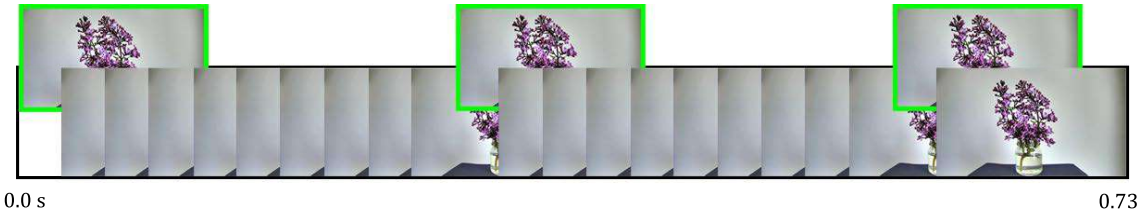
je dôležitá časť pri zhotovovaní časozberného videa. V opačnom prípade by nešlo o zhotovovanie časovej snímky pre časozberné video, ale len o klasické video. Túto frekvenciu vypočíta program pre obe prípady. Jej výpočet pre video čítané z disku a video čítané z IP alebo dohľadovej kamery sa trošku líši, avšak ide o jednoduché výpočty.

Výpočet frekvencie pre video čítané z disku

Výpočet frekvencie je možno vykonať viacero spôsobmi. Najoptimálnejší spôsob, ktorý som zhodnotila pre video čítané z disku je vydelenie celkového počtu snímiek v celom videu počtom snímiek v časozbernom videu. Množstvo snímiek v pôvodnom videu zistí program. Počet snímiek časozberného videa sa vypočíta vynásobením fps, ktorým bude video zapísané s dĺžkou videa v sekundách.

Výpočet frekvencie pre video čítané z online kamery

Video online streamu je vo svojej podstate nekonečné, preto sa nedá zistiť konečný počet snímiek čítaného videa. Avšak na vstupe je zadaná doba snímania a dĺžka výstupného videa. Tieto hodnoty sa prevedú na sekundy a vydelí sa doba snímania dĺžkou časozberného videa. Tento medzivýsledok sa vydelí číslom 60, čím je vypočítaná frekvencia časozberného videa produkovaného z IP kamery alebo online dohľadovej kamery. Ideálne by bolo vynásobiť fps streamovacej kamery a čas nahrávania v sekundách, čím by sa vypočítal počet snímiek videa podobne ako v prvom prípade. Avšak program vie zistiť fps iba z existujúceho videa na disku, nie zo streamovacej kamery.



Obr. 4.5: Znázornenie výberu snímiek z videa.

4.4 Jednoduché časozberné video

Princíp algoritmu pre jednoduché časozberné video [3] spočíva v tom, že z konečného počtu snímiek, ktoré video má, je vybraných len pár z nich. Pri čítaní videa z disku prejde algoritmus každou snímkou videa. Niektoré snímky „ignoruje“ a niektoré uloží pre zápis do výstupného videa. Snímky sú ukladané pravidelne podľa vypočítanej frekvencie, ktorej výpočet približujem v sekcii 4.3. Na obrázku 4.5 je znázornený výber snímiek z krátkeho úseku videa. Toto video už bolo časozberným videom, ktoré malo takmer 10 minút a snímané bolo asi 10 hodín. Video bolo stále príliš dlhé, tak som ho skrátila na 1 minútu, takže algoritmus bude zapisovať každú desiatu snímku (hodnota fps u nich bola rovnaká).

Pri zaznamenávaní časozberného videa zo streamu sa snímky obrazu ukladajú raz za nejaký čas, ktorý sa vypočíta pomocou už vypočítanej frekvencie f:

$$t = \frac{1}{f} \tag{4.1}$$

Pred zápisom prvej snímky sa zistí pomocou programu čas jej zápisu a tento čas sa odpočítava v cykle od aktuálneho času. Ak sa tento rozdiel rovná alebo je väčší než vypočítané t, tak sa zapíše ďalšia snímka. Snímanie končí vtedy, keď rozdiel aktuálneho času a času začiatku cyklu je rovný alebo väčší ako stanovená doba snímania.

4.5 Nepokojné objekty v časozbernom videu

Ako prvému obmedzeniu som sa venovala nepokojným objektom v časozbernom videu. V tejto podkapitole opisujem, ako sa táto nepokojnosť môže prejavovať, návrh môjho algoritmu a dosiahnutý výsledok.

4.5.1 Prejav nepokojných objektov

Ako je uvedené v kapitole 3, zaobstaraných bolo niekoľko testovacích videí. Jedno z nich bolo video horiacej sviečky, ktoré malo dĺžku trvania približne 1 hodinu a 45 minút s počtom snímkov za sekundu 30. Po zrýchlení videa, resp. vyprodukovaním jednoduchého časozberného videa, sa plamienok sviečky nepokojne kmitá. Tento plamienok môže byť nazvaný nepokojným alebo nekľudným objektom. Video tejto sviečky sa nachádza v priloženom médiu v súbore `videa/sviecka1.mp4`.

Na obrázku 4.6 sa nachádzajú dve po sebe idúce snímky z časozberného videa, ktoré ešte nebolo upravené žiadnym algoritmom. Pri porovnaní týchto dvoch snímiek z obrázka 4.7 sa dá povšimnúť, že plamienok sviečky na oboch snímkach je naklonený pod iným uhlom. To spôsobuje, že objekt (v tomto prípade plamienok) v časozbernom videu pôsobí nepokojne.

Obrázok 4.7 detailnejšie zobrazuje snímky z obrázka 4.6. Rozdiel v sklone plamienka je vidieť zreteľnejšie, taktiež som ho znázornila prerušovanou modrou úsečkou, ktorá spája vrchol plamienka a stred pomocnej tyrkysovej úsečky.

Obr. 4.6: Dve za sebou idúce snímky v časovom snímku.

Obr. 4.7: Detail snímiek obrázka 4.6. Prerušovanou priamkou je znázornený sklon plamienka.

4.5.2 Minimalizovanie nepokojnosti objektu

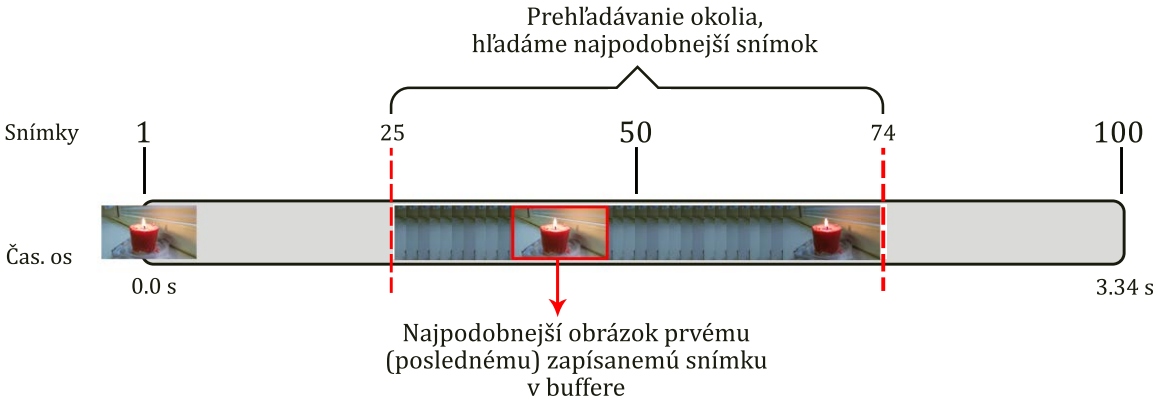
Pre minimalizovanie, prípadne odstránenie nepokojnosti objektu som navrhla algoritmus, ktorý je znázornený na obrázku 4.8. Tento algoritmus ako jediný zhotovuje časozberné video a zároveň eliminuje obmedzenie. Pre jeho aplikovanie sú vhodné vstupné videá väčšej dĺžky než bude vyprodukované časozberné video. Najideálnejšie je video, ktorého dĺžka odpovedá dobe jeho snímania (tzn. nie je zrýchlené).

Na spomínanom obrázku 4.8 je vidieť znázornenú časovú os a zápis snímiek. Snímky, ktoré budú vo finálnom videu sú priebežne zapisované na disk. Je možné ich zapisovať aj do pamäti, čo sa však neosvedčilo pri časovo dlhších videách, kde bolo potrebné uložiť viac snímiek. Vtedy program predčasne skončil kvôli nedostatku pamäti, takže zápis na disk som zvolila ako istý spôsob ukladania. Do spomínaného súboru sa zapíše prvá snímka, ako je znázornené aj na obrázku 4.8. Frekvenciu zápisu snímiek do časozberného videa som v tomto prípade zvolila 50, čo znamená, že analyzovaná bude každá 50. snímka v poradí z pôvodného videa a jej okolie. V tomto okolí sa hľadá najpodobnejšia snímka k poslednej zapísanej snímke a to tak, že všetky snímky z okolia sa porovnajú s poslednou zapísanou snímkou, a najpodobnejšia snímka z okolia sa uloží na spomínaný disk. Stále sa porovnávajú práve

dve snímky. Veľa snímiek z videa je zahodených, preto je tento algoritmus navrhnutý tak, že produkuje časozberné video a zároveň eliminuje obmedzenie.

Okolie som určila so zámerom porovnať čo najviac snímiek a zároveň sa vyhnúť možnému vzniku prieniku dvoch okolí. To by mohlo spôsobiť zápis rovnakej snímky dvakrát, preto optimálnou hodnotou je polovica hodnoty frekvencie zápisu snímky.

Pretože sa z okolia zapisuje iba jedna snímka, je potrebné, aby video spracovávané algoritmom bolo dlhšie ako video, ktoré bude vyprodukované. Ak by bola dĺžka videa na výstupe rovnaká ako dĺžka vstupného videa, tak sa žiadna zmena na videu neprejaví, neexistovalo by okolie, z ktorého vyberať najpodobnejšiu snímku. Čím väčší časový rozdiel bude medzi vstupným a výstupným videom, tým bude väčšie okolie, a tým je možné dosiahnuť lepších výsledkov.



Obr. 4.8: Znázornenie algoritmu pre vyhľadávanie najpodobnejšej snímky.

4.5.3 Výsledky algoritmu pre elimináciu nepokojnosti objektu

Pohyb plamienka po prevedení algoritmu, je omnoho pokojnejší. Upravené video je priložené v médiu `videa/sviecka1_hyper.mp4`.

Ďalším pokusným videom boli kvety orgovánu vo vetre pričom vidieť prechod mrakov a obloha sa stmavuje kvôli zapadajúcemu slnku. Z približne hodiny a pol zaznamenávania časových snímiek bolo vyprodukované časozberné video dlhé 4 a pol minúty. Toto video som mnou implementovaným algoritmom skrátila ešte viac na dĺžku 30 sekúnd. V skrátenom videu som odpozorovala, že kvety boli kvôli vetru rozkmitané a nepokojné. Na pôvodné video som aplikovala algoritmus a na vstupe programu som nastavila dĺžku videa opäť 30 sekúnd. Vo výsledku sú kvety orgovánu „pokojnejšie“. Obe videá pred a po prevedení algoritmu prikladám v médiu v súbore `videa` pod názvom `orgovan.mp4` a `orgovan_hyper.mp4`.

4.6 Prudké svetelné zmeny v časozbernom videu

Ďalším z obmedzení sú prudké svetelné zmeny vo videu. Laicky sa dá povedať, že vo videu vzniká blikanie. Odstránenie tohto javu spočíva v úprave jasu, resp. svetlosti snímiek. Návrh algoritmu bol inšpirovaný fragmentom kódu z GitHub[3].

[3] https://github.com/gondsm/timelapse_deflickerer

4.6.1 Prejav prudkých svetelných zmien

Ďalšie z testovacích videí bolo východ slnka, jeho následný pohyb, resp. bol zachytený deň od východu slnka a kúsok dňa. Video je priložené v médiu v súbore `videa/vychodslnka.mp4`. Kamera snímala približne 6 hodín vo frekvencii 1 fps. V tomto časozbernom videu som odpozorovala prudké zmeny jasu, laicky povedané, vo videu nastáva blikanie. To spôsobuje hlavne slnko a mraky, ktoré neustále menia svoju polohu, obraz sa raz zosvetľuje, inokedy tmavne. Počas pozorovania celého 6-hodinového záznamu, bez zhotovovania časozberu, svetelné zmeny by sa prejavovali plynulo, zatiaľ čo pri zhotovovaní časozberu o frekvencii 1 fps sú v jednotlivých snímkach svetelné zmeny prudšie. Týmto vzniká blikanie, ktoré je pre sledovateľa videa nepríjemné.

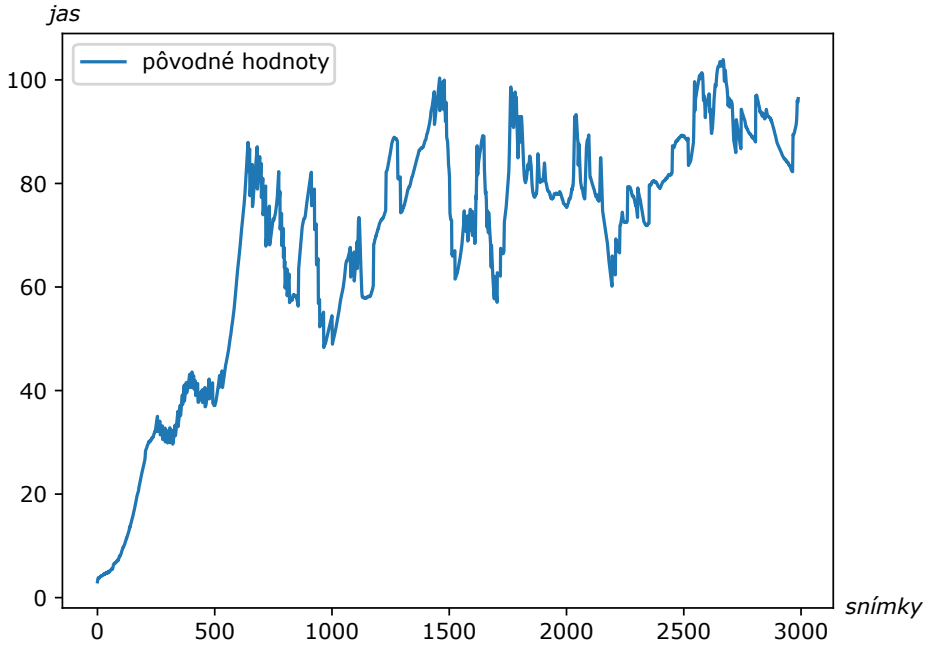
4.6.2 Eliminácia prudkých svetelných zmien

Na začiatku sa všetky snímky videa zapíšu na disk kvôli jednoduchej práci s nimi v ďalších krokoch. Princíp algoritmu spočíva vo vypočítaní hodnôt jasu pre každú snímku, nájdení polynómu aproximujúceho tieto hodnoty a následná úprava jasov snímiek pomocou gamma korektúry so zámerom priblížiť ich hodnoty k vypočítanému polynómu.

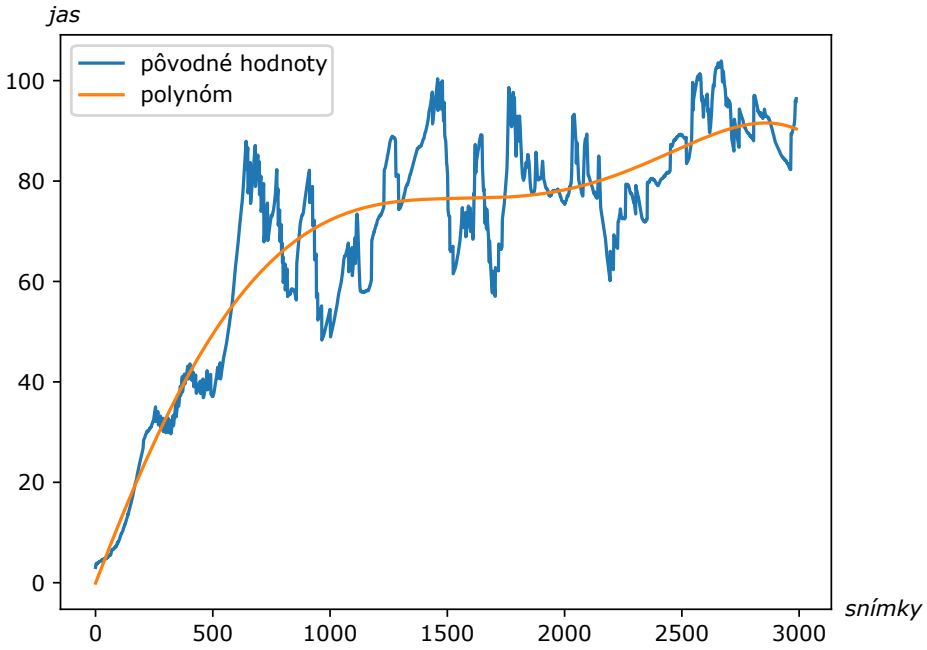
Hodnota jasu snímky sa vypočíta spriemerovaním hodnôt jasov jednotlivých pixelov danej snímky. Hodnoty svetlostí jednotlivých snímiek zo spomínaného videa sú graficky zobrazené na obrázku 4.9. Os y znázorňuje hodnoty jasu a os x znázorňuje snímky. Z grafu je vidieť výrazné zmeny hodnôt jasov snímiek. Tieto hodnoty je potrebné podrobiť úpravou, po ktorej vzhľad krivky nadobudne plynulejší priebeh. Preto sa vypočíta polynóm aproximujúci túto krivku, ktorého priebeh je znázornený na obrázku 4.10. Na základe tohto polynómu sa hodnoty svetlosti upravia tak, aby sa nové hodnoty jasov k nemu približovali. Hodnoty svetlostí snímiek sa upravia pomocou zostavenej vyhľadávacej tabuľky (LUT, Look-up table), ktorá mapuje hodnoty v rozmedzí $\langle 0; 255 \rangle$ na ich nové gammou upravené hodnoty. To, ako bude tabuľka vyzerať závisí od vypočítanej gammy, ktorá je pre každú snímku iná. Hodnota gammy závisí od vypočítaného rozdielu pôvodnej hodnoty jasu a požadovanej hodnoty jasu pre daný obrázok danú polynómom.
Nech d je rozdiel pôvodnej hodnoty jasu a požadovanej hodnoty jasu, potom:

$$\gamma = 1 + \frac{4.0d}{255} \tag{4.2}$$

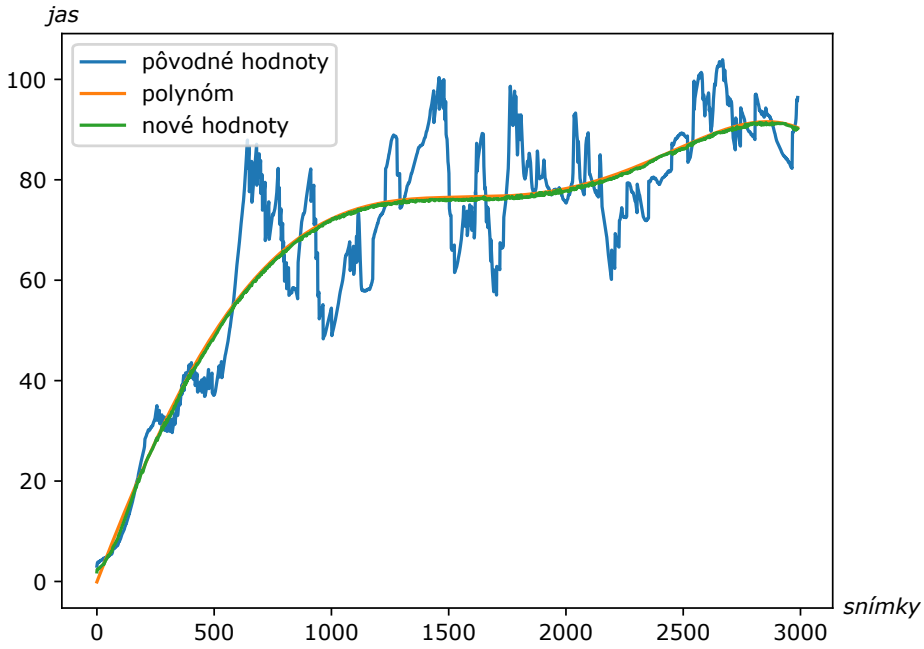
Posledný graf na obrázku 4.11 zobrazuje grafický priebeh nových hodnôt jasov jednotlivých snímiek. Krivka nových hodnôt sa veľmi tesne približuje k polynómu.



Obr. 4.9: Hodnoty jasov snímiek vykreslené na grafe.



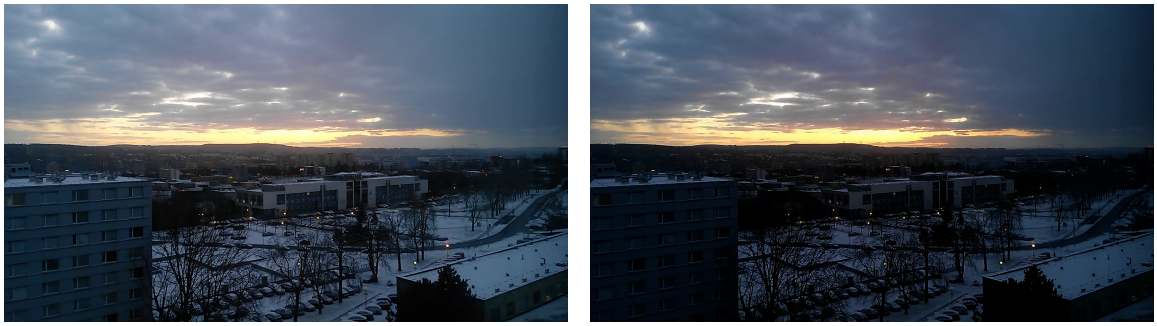
Obr. 4.10: Polynóm aproximujúci hodnoty jasov snímiek.



Obr. 4.11: Krivka nových hodnôt približujúca sa k polynómu.

4.6.3 Výsledky eliminácie prudkých svetelných zmien

Na obrázku 4.12 sú dve snímky z videa, jedná sa o tú istú snímku, avšak na pravom obrázku je upravený jas pomocou gamma korektúry. V priloženom médiu sa nachádza video upravené týmto eliminačným algoritmom v súbore `videa/vychodslnka_flick.mp4`. V zložke `videa` sú ďalšie videá pred a po prevedení tohto algoritmu. V názve sú označené slovom *flick*.



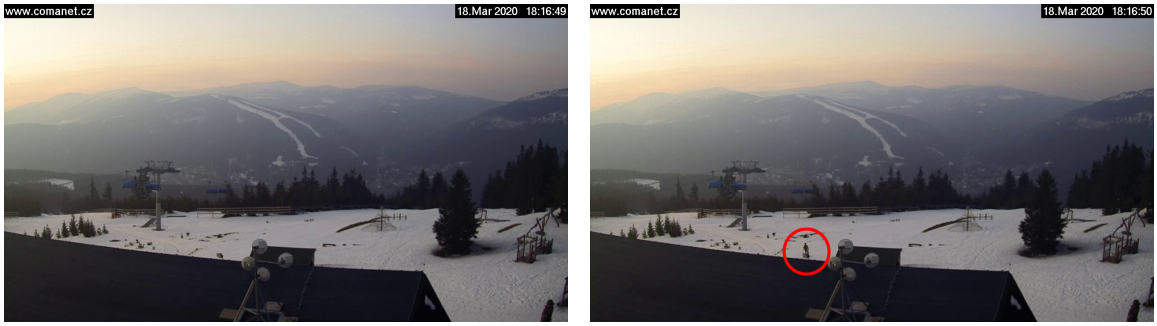
Obr. 4.12: Rovnaká snímka pred úpravou jasu naľavo a po úprave napravo.

4.7 Nežiadúce objekty v časozbernom videu

V tejto sekcii opisujem výskyt pohybu v časozbernom videu, avšak tento názov je veľmi špecifický, preto som toto obmedzenie pomenovala ako *nežiadúce objekty v časozbernom videu*. Toto obmedzenie som sa pokúšala eliminovať dvoma spôsobmi, jedným z nich je detekcia pohybu, avšak to sa neosvedčilo pri každom testovacom videu. Druhá metóda je založená na základe odstránenia *Salt and pepper* šumu. Obe metódy popisujem v nasledujúcich podsekciách, spolu s prejavom tohto obmedzenia a dosiahnutými výsledkami.

4.7.1 Prejav nežiadúcich objektov

Z online dohľadovej kamery lyžiarskej zjazdovky[4] som z približne hodiny záznamu zhotovila časozberné video trvajúce minútu. Na tomto videu v jednom časovom úseku prešli tri osoby. Obrázok 4.13 zobrazuje dve po sebe idúce snímky časozbernom videu zo spomínanej zjazdovky. Zatiaľčo na snímke naľavo je vidieť iba prírodu, na snímke vpravo prechádza prvá z troch spomínaných osôb. Na videu tento jav trvá ani nie sekundu a pôsobí to ako preblknutie, čo pôsobí pre sledujúceho rušivo.



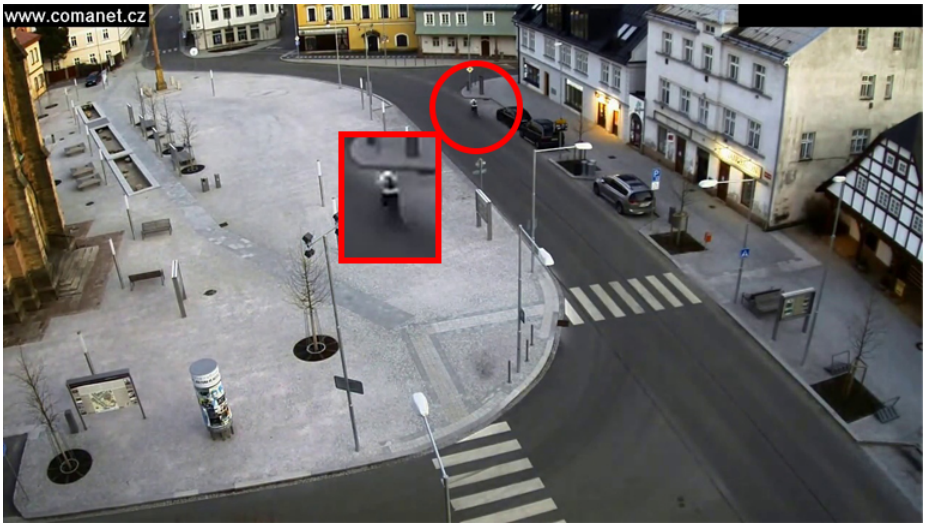
Obr. 4.13: Dve po sebe nasledujúce snímky, pričom na snímke napravo je pohyb, ktorý je potrebné eliminovať.

4.7.2 Odstránenie nežiadúcich objektov

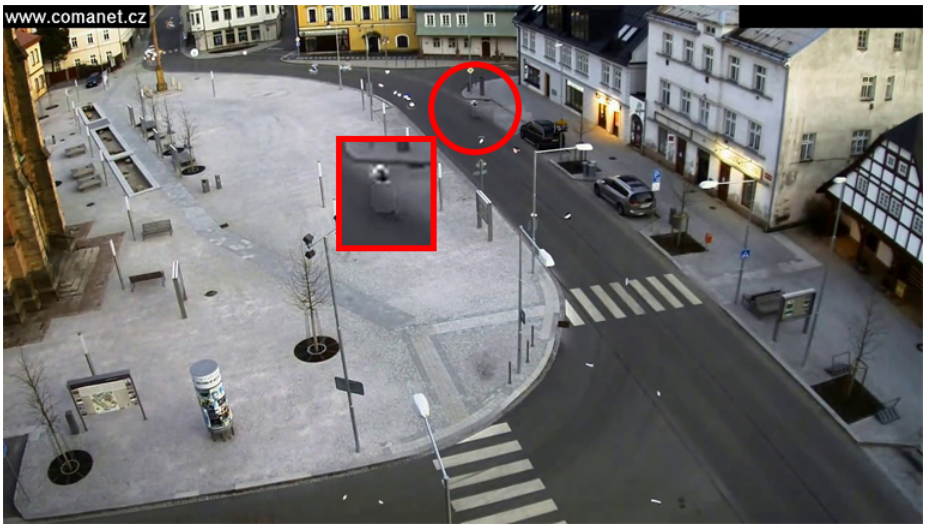
Odstrániť nežiadúce objekty, resp. pohyb z videa som sa pokúšala pomocou detekcie pohybu. Zaznamenala som súradnice pixelov, na ktorých bol objekt (v tomto prípade človek) zachytený a nahradila som ich pixelmi tých istých súradníc z predchádzajúcej snímky, kde objekt nebol. Na videu zo spomínanej zjazdovky bola táto metóda úspešná a pohyb bol odstránený, avšak po experimentoch na iných videách som zistila, že nevystačoval.

Ďalšie časozberné video bolo zhotovené z dohľadovej kamery námestia. Na tomto videu premávalo viacero áut, cyklistov a chodcov. Na obrázku 4.14 je vidieť jedného z cyklistov. Výsledok po aplikovaní spomenutého algoritmu znázorňuje obrázok 4.15. Dá sa spozorovať, že bola odstránená iba časť cyklistu a v konečnom dôsledku tam ostal viditeľný fľak. Taktiež na celej snímke si je možné povšimnúť akési častice na rôznych miestach. Podobný prípad je vidieť aj na obrázku 4.16.

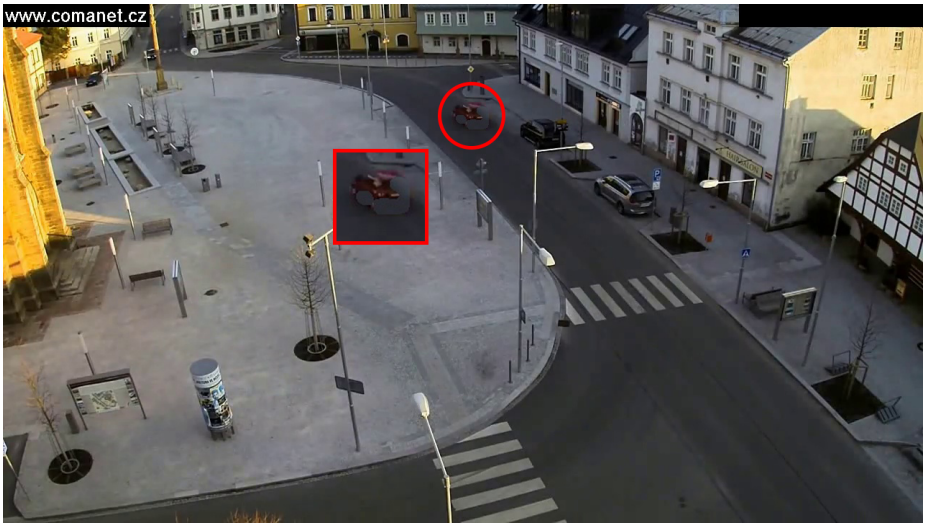
[4] https://webkamery-krkonose.cz



Obr. 4.14: Zachytený cyklista na dohľadovej kamere.



Obr. 4.15: Neúspešný pokus o odstránenie cyklistu z videa.



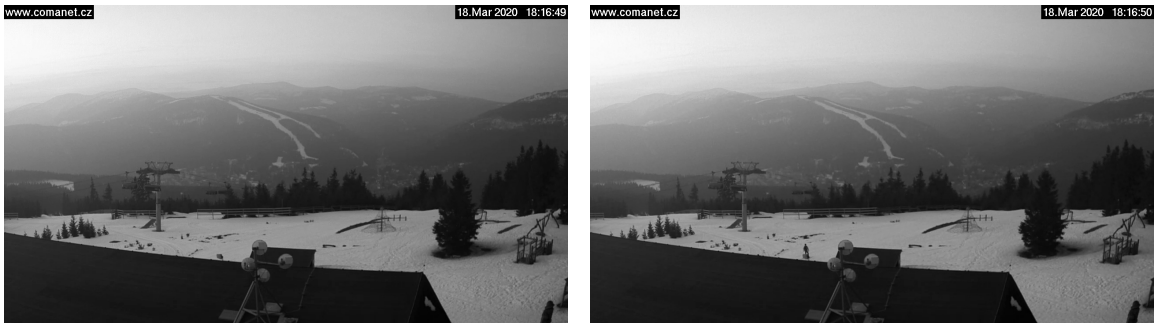
Obr. 4.16: Neúspešný pokus o odstránenie auta z videa.

Eliminácia objektov na základe detekcie pohybu

Myšlienka odstránenia objektu z videa, resp. zo snímku pomocou detekcie pohybu spočíva vo výmene pixelov snímku v čase *t*, na ktorom bol zaznamenaný objekt, za pixely predošlého snímku v čase *t-1* na tých súradniciach, kde bol zaznamenaný objekt na snímku v čase *t*.

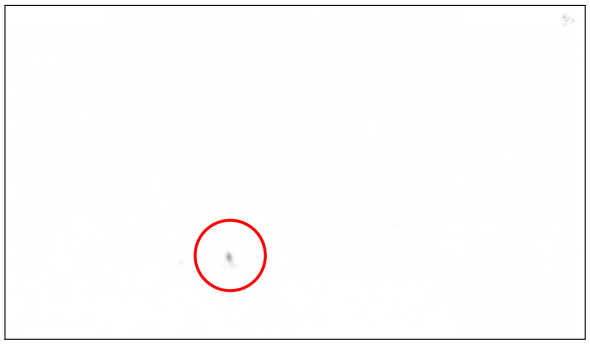
Pri detekcii pohybu [13] nás farby snímiek nemusia zaujímať, preto snímky z obrázka 4.13 sa prevedú do šedého modelu, ako je vidieť na obrázku 4.17. Šum obrázka by mohol byť detekovaný algoritmom ako objekt, preto sa aplikuje na snímky Gaussian blur pre ich vyhladenie (obrázok 4.18). Následne sa vypočíta rozdiel týchto snímiek odčítaním pomocou absolútnej hodnoty. Vo výsledku sa získa snímka, ako znázorňuje obrázok 4.19. Je na ňom je malý „flak“, ktorý je našim nežiadúcim objektom. Keďže je obrázok čierno biely, dá sa zistiť, ktoré pixely je potrebné zmeniť.

Nedá sa povedať, že je táto metóda úplne nefunkčná, ale hodí sa skôr pre tie „jednoduchšie“ prípady, ako bol lyžiar na obrázkoch. Ide teda o jeden zreteľný objekt na obraze, zatiaľ čo na ulici je pohybu a nie príliš zreteľných objektov priveľa.



Obr. 4.17: Snímky z videa prevedené do šedého modelu.

Obr. 4.18: Snímky z videa prevedené do Gaussian blur.

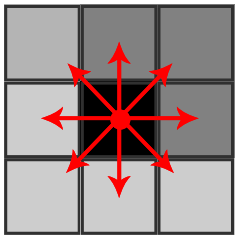


Obr. 4.19: Fľak znázorňujúci objekt. Snímka bola zinvertovaná kvôli tlači – na pôvodnej snímke je pozadie čierne a objekt biely.

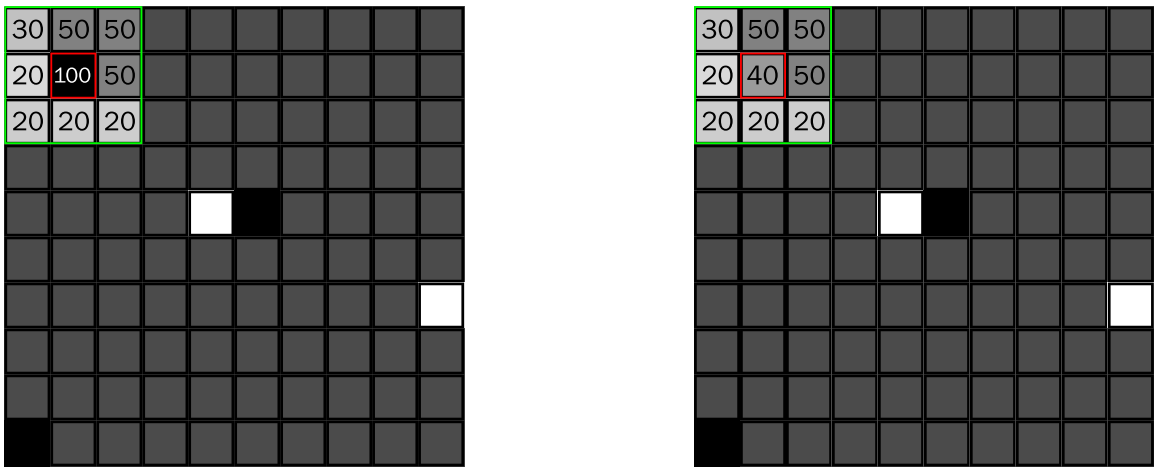
Eliminácia objektov na základe odstránenia Salt and pepper šumu

Keďže predchádzajúca metóda nebola úspešná, zhotovila som algoritmus, ktorý je založený na princípe metódy pre odstránenie *Salt and pepper* šumu zo snímky. Tento šum sa prejavuje na snímkach tak, že sa na rôznych miestach nachádzajú biele a čierne pixely. Tie pripomínajú soľ a čierne korenie roztrúsené na snímke, ako ilustruje obrázok 4.20. Salt and pepper šum je možné odstrániť tak, že pixel, ktorý tento šum tvorí sa nahradí novým pixelom. Jeho hodnota sa vypočíta zohľadnením okolitých pixelov šumivého pixelu o rozmere 3x3, z ktorých sa vypočíta priemer. Okolie pixelu tvorí 8 pixelov, priemer sa vypočíta z deviatich pixelov, ako znázorňuje obrázok 4.22 [11, 17].

Obr. 4.20: Znázornenie Salt and pepper šumu. Naľavo je pôvodný obrázok, napravo so šumom [18].

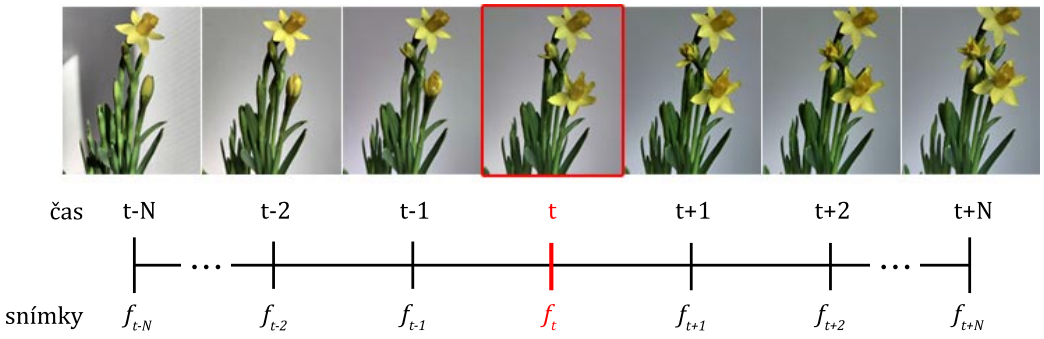


Obr. 4.21: Pixel a jeho okolie, z ktorých sa počíta priemer.



Obr. 4.22: Ukážka odstránenia Salt and pepper šumu. Pixel v červenom ráme je spracovávaný pixel. Pixely, z ktorých sa počíta priemer, sú v zelenom ráme. Čísla v jednotlivých pixeloch sú hodnoty intenzity pixelov uvedené v percentách.

V mnou zhotovenom algoritme spomínané „okolie“ pixelu v čase t o súradniciach $[x, y]$ predstavujú pixely zo snímiek v čase $t - 1$ až $t - N$ na rovnakej pozícii. Tieto pixely som nazvala *minulosťou* pixelu v čase t. Pre úplnosť navrhnutého algoritmu minulosť snímky v čase t nestačí a je potrebné využiť aj snímky nasledujúce po nej, ktoré som nazvala *budúcnosťou*. Sú to snímky pohybujúce sa v čase $t + 1$ až $t + N$. Takže *budúcnosťou* pixelu



Obr. 4.23: Snímky zobrazené na časovej osi. Snímka v červenom rámčeku je spracovávaná. Jej minulosťou sú snímky f_{t-1} až f_{t-N} a budúcnosťou sú f_{t+1} až f_{t+N}.

v čase t na pozícii $[x, y]$ sú pixely $[x_{t+1}, y_{t+1}]$ až $[x_{t+N}, y_{t+N}]$. Tento časový priebeh znázorňuje obrázok 4.23. Počet snímiek z minulosti som určila 9, rovnako ako je tomu v algoritme Salt and pepper, od ktorého je návrh inšpirovaný. Ak sa pixel $[x_t, y_t]$ prejaví ako „šumivý“, nahradí sa priemerom pixelov $[x_{t-1}, y_{t-1}]$ až $[x_{t-N}, y_{t-N}]$. Čo v tomto prípade znamená „šumivý“ pixel, resp. „šumivé“ pixely? Sú to pixely objektu, ktorý sa na snímke v čase t objaví a na snímkach nasledujúcich po nej sa už nenachádza. RGB hodnoty pixelu snímiek v čase $t-1$ až $t-N$ na pozícii $[x, y]$ sa pohybujú v nejakom rozmedzí, preto im je určený interval $\langle a; b\rangle$. Ak RGB zložky pixelu na rovnakej pozícii v čase t nepatria do intervalu, skontrolujú sa pixely z budúcnosti, ktorých je tiež 9. Pixel je nahradený spomínaným priemerom pixelov z minulosti, ak sa RGB zložky pixelov z budúcnosti pohybujú v hodnotách ako RGB zložky pixelov z minulosti.

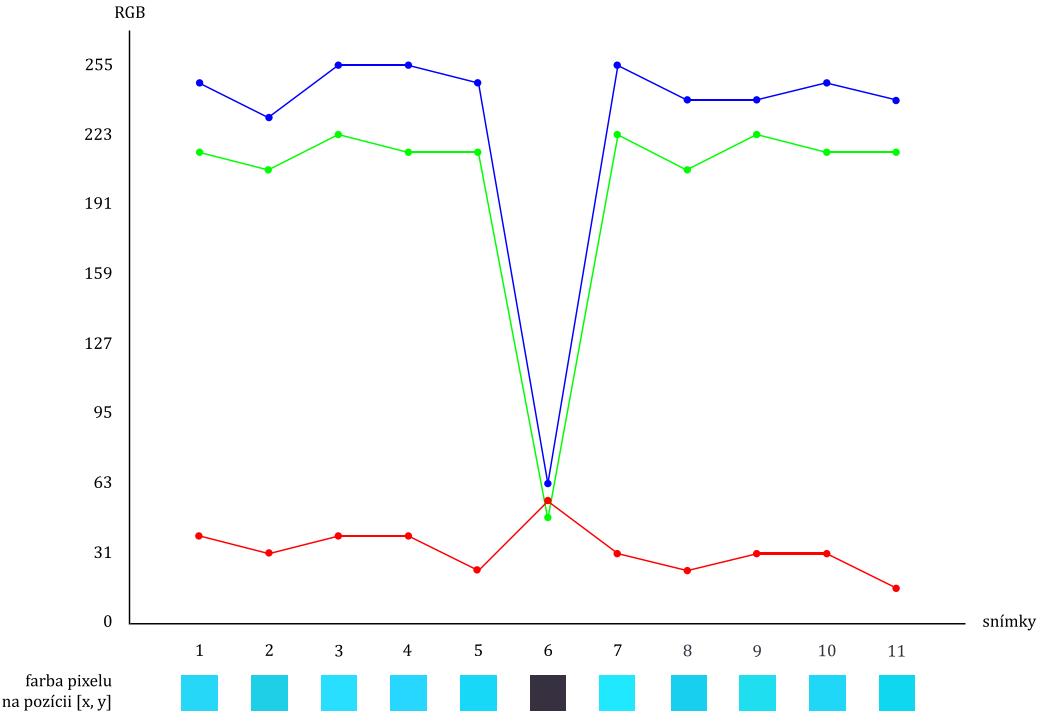
Obrázky 4.24 a 4.25 zachycujú grafický priebeh hodnoty pixelu na 11 snímkach na pozícii $[x, y]$, pre ilustráciu nie sú podstatné konkrétne súradnice. Na grafe z obrázka 4.24 sa hodnoty RGB snímiek 1 až 5 pohybujú v určitých hodnotách a každá zložka RGB má svoje minimum a maximum. Z nich sa stanovia intervaly pre každú zložku RGB uvedené v tabuľke:

	min	max
R_{1-5}	23	39
G_{1-5}	207	233
B_{1-5}	231	255

Na snímke 6 sa tieto hodnoty výrazne vychyľujú od príslušných intervalov, takže je tento pixel vyhodnotený ako šumivý, resp. sa v ňom vyskytla zmena. Na základe tohto zistenia algoritmus skontroluje tento pixel na snímkach, ktoré nasledujú. Ako zobrazuje graf, hodnoty jednotlivých RGB zložiek na snímkach 7 až 11 sa opäť pohybujú v hodnotách ako predtým na snímkach 1 až 5, preto je táto zmena vyhodnotená ako dočasná. Pixel sa tak nahradí priemerom pixelov 1 až 5. Graf na obrázku 4.25 zobrazuje iný scenár. Intervaly RGB zložiek snímiek 1 až 5 sú rovnaké ako v prvom prípade a zmena opäť nastáva na pixele snímky 6. Avšak RGB hodnoty pixelov snímiek 7 až 11 sa už nepohybujú v intervaloch ako predtým na snímkach 1 až 5. Tentokrát vzniknú nové intervaly pre snímky 7 až 11:

	min	max
R_{7-11}	47	63
G_{7-11}	55	63
B_{7-11}	63	71

Do týchto intervalov dokonca spadajú RGB hodnoty snímky 6 a zmena je vyhodnotená ako trvalá, preto RGB hodnoty pixelu na snímke, kde zmena začala, nie je potrebné meniť.

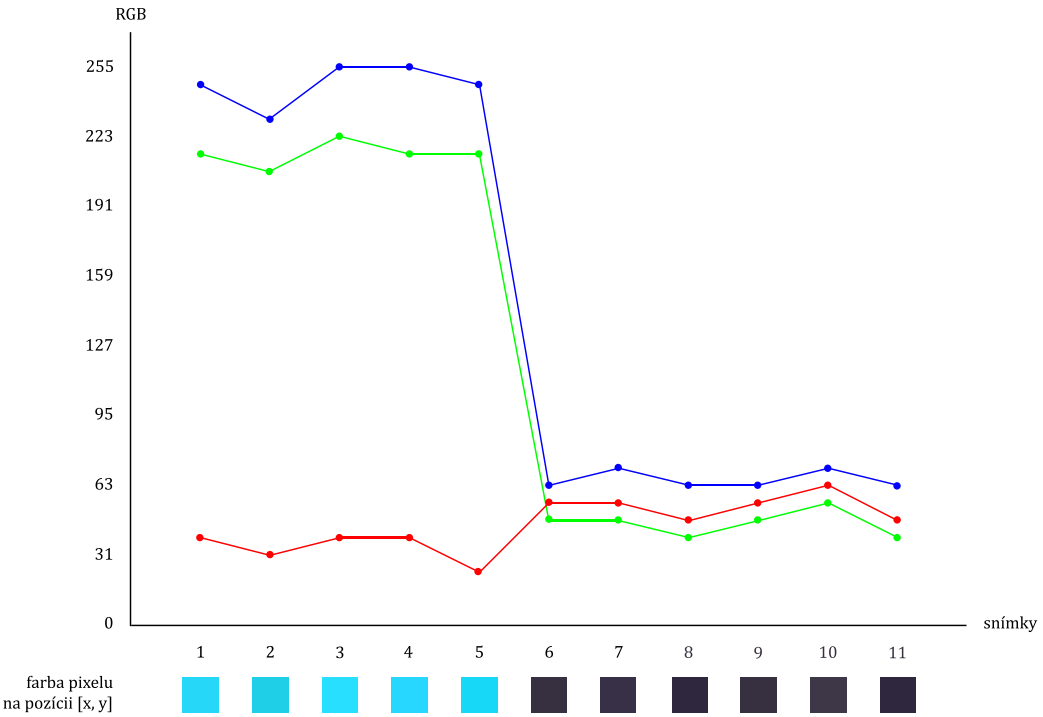


Obr. 4.24: Grafom vykreslené RGB hodnoty pixelu na pozícii [x, y] na 11 snímkach. Zobrazená je dočasná zmena na snímke 6.

4.7.3 Výsledky eliminácie pohybu z videa

Ako som už naznačila, detekcia pohybu nebola vo výsledku funkčná pre všetky prípady. Lepšie výsledky preukázal algoritmus založený na eliminácii Salt and pepper šumu zo snímky.

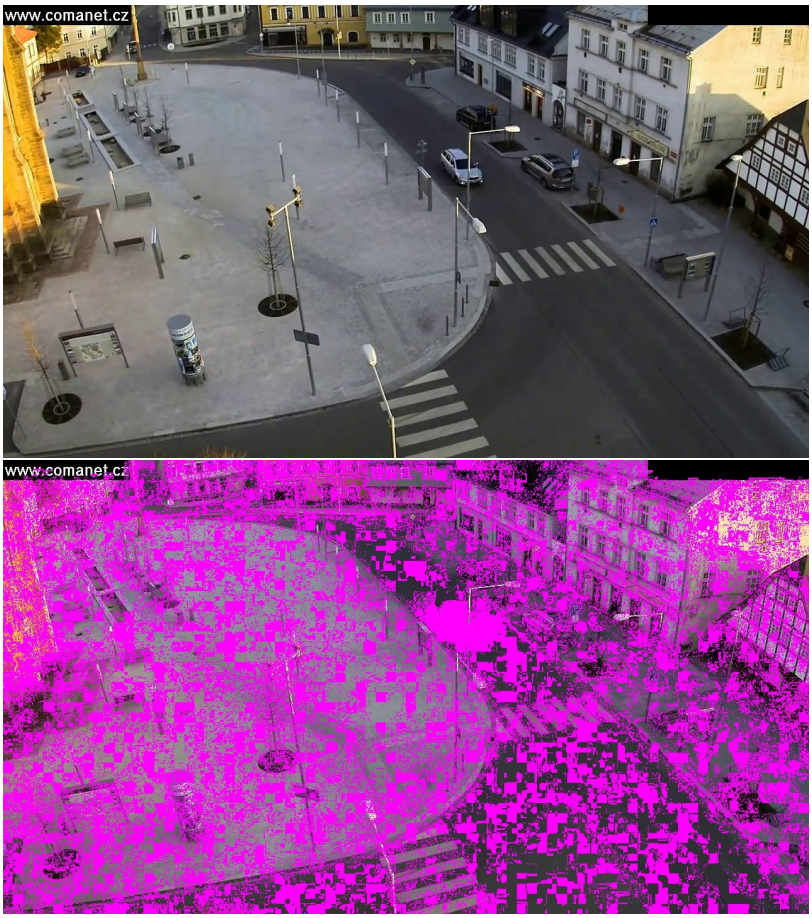
Pre overenie, ako tento algoritmus funguje a znázornenie, na akých miestach nastala zmena pixelov, nahradila som pixely, ktoré boli zdetekované ako „šumivé pixely“, pixelmi fuchsiovej farby. Taktiež som zvolila iný počet snímiek z minulosti, z ktorých je určený interval pre každý pixel. Na obrázkoch 4.26 a 4.27 sú znázornené dva experimenty, kde bolo potrebné odstrániť auto. V prvom experimente (obrázok 4.26) boli použité 3 snímky z minulosti pre stanovenie intervalu. V druhom experimente (obrázok 4.27) bolo pre tento interval použitých 7 snímiek. Na oboch experimentoch je vidieť, že na miestach, kde sa nachádzajú autá dochádza k zhluku fuchsiových pixelov. Avšak tieto pixely sú roztrúsené po celom snímku, pretože na videu dochádza k šumu a k jemným farebným zmenám. Pri prvom experimente, kde interval tvorilo menšie množstvo snímiek, algoritmus detekoval omnoho viac týchto farebných zmien. Porovnaním týchto dvoch pokusov je zrejmé, že použitím väč-



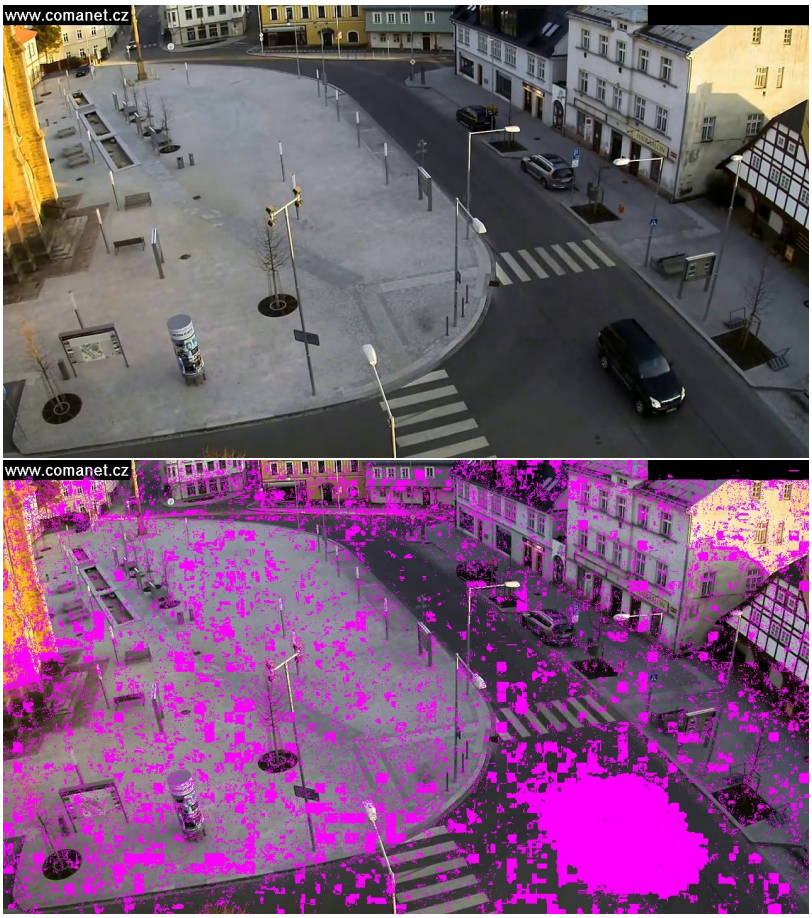
Obr. 4.25: Grafom vykreslené RGB hodnoty pixelu na pozícii [x, y] na 11 snímkach. Zobrazená je trvalá zmena pixelu začínajúca na snímke 6.

šieho množstva snímiek pre stanovenie intervalov hodnôt pixelov, je výsledok precíznejší a s menším obsahom farebného šumu.

Po aplikovaní algoritmu na celé video, je pohyb odstránený. Výsledok je v priloženom médiu v súbore `videa/namestie1_move.mp4`.



Obr. 4.26: Zmena šumivých pixelov snímky, kedy pre stanovenie intervalu boli použité 3 snímky. Fuchsiová farba znázorňuje, ktoré pixely boli zmenené.



Obr. 4.27: Zmena šumivých pixelov snímky, kedy pre stanovenie intervalu bolo použitých 7 snímiek. Fuchsiová farba znázorňuje, ktoré pixely boli zmenené.

Kapitola 5

Implementácia

V tejto kapitole opisujem implementáciu algoritmov, ktoré som navrhla pre elimináciu obmedzení v časozbernom videu, prácu s knižnicou OpenCV, jej metódami a objektami. Taktiež opisujem mnou implementované funkcie a ich použitie.

5.1 Spracovanie vstupného videa

Po načítaní argumentov prebieha kontrola, či daná cesta k videu existuje, prípadne, či je odkaz k videu správny. Rovnako sa kontroluje, či je formát zadaného času korektný. Po tejto kontrole sa vytvoria dva objekty, jeden pre čítanie videa a druhý pre zápis. Pre zachytenie videa sa vytvorí objekt triedy `VideoCapture` z knižnice OpenCV. Ako parameter sa predá cesta k videu akéhokoľvek formátu alebo odkaz k online streamu z IP kamery alebo online dohľadovej kamery. Pre zápis sa vytvorí objekt triedy `VideoWriter`, taktiež z knižnice OpenCV. Tá má na vstupe parametre, s ktorými sa definuje zápis výstupného videa. Sú to cesta, kde sa video zapíše, formát, v akom bude video zapísané, fps a rozlíšenie videa v pixeloch:

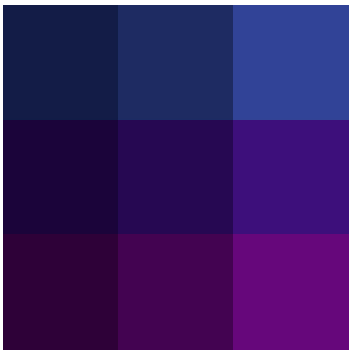
```
cv.VideoWriter(cesta, formát, fps, (výška, šírka))
```

Formát je predaný ako objekt triedy `VideoWriter_fourcc` knižnice OpenCV, jej parametrom je kodek. Videá sú ukladané do formátu `mp4` a jeho kodek v Pythone je `mp4v`. Rozlíšenie videa sa získa pomocou metódy `get()` volanou nad objektom pre čítanie videa. Jeho parameterom je metóda `CAP_PROP_FRAME_HEIGHT` pre zistenie výšky videa alebo `CAP_PROP_FRAME_WIDTH` pre zistenie šírky videa tiež z knižnice OpenCV. Po týchto nastaveniach program pokračuje volaním príslušných funkcií na základe vstupného parametru, ktorý slúž ako prepínač.

5.2 Čítanie snímiek z videa

Čítanie snímiek prebieha v cykle `while`, v ktorom sa nad objektom triedy `VideoCapture()` volá metóda `read()`. Táto funkcia číta, dekóduje a vráti nasledujúcu snímku videa. Taktiež vráti návratovú hodnotu `true` alebo `false` v prípade úspechu, alebo neúspechu čítania snímky. Funkcia vracia snímku štandardne v BGR farebnom modeli (ak ide o farebný obrázok) ako matrix, čo znamená, že vráti pole. V tomto poli sú ďalšie polia predstavujúce riadky obrázka, teda údaje na osi y. Pri čítaní farebnej snímky sa v každom z týchto polí nachádzajú ďalšie troj-prvkové polia a každé z nich reprezentuje jeden pixel. V týchto poliach

sú hodnoty BGR kanálov prípadne po zmenení farebného modelu funkciou `cvtColor()` sa môže jednať o RGB, HSV model a pod. Pixely obrázkov v šedej škále nie sú reprezentované poľom hodnôt ale iba jednou hodnotou v rozmedzí 0 až 255. Obrázok 5.1 pre ilustráciu znázorňuje niekoľkonásobne priblížený načítaný obrázok o rozmere 3×3 px vo farebnom modeli BGR.



Obr. 5.1: Znázornenie pixelov obrázka o rozmere 3×3 px.

Reprezentácia obrázka 5.1 v poli po prečítaní vyzerá takto:

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}70 & 28 & 18\end{bmatrix}\\ \begin{bmatrix}99 & 42 & 29\end{bmatrix}\\ \begin{bmatrix}150 & 67 & 49\end{bmatrix}\end{bmatrix}\\ \vdots\\ \begin{bmatrix}\begin{bmatrix}57 & 2 & 46\end{bmatrix}\\ \begin{bmatrix}81 & 3 & 81\end{bmatrix}\\ \begin{bmatrix}123 & 8 & 103\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Pristúpiť k jednotlivým pixelom nie len tohto obrázka, resp. ich hodnotám BGR je možné spôsobom, ako zobrazuje nasledujúca časť algoritmu napísaného v Pythone. V komentároch sú hodnoty premenných po prvom behu oboch cyklov `for`. Čítanie pixelov začína v ľavom hornom rohu smerom doprava.

```
for y in frame:
    for x in y:             # y: [[70 28 18][99 42 29][150 67 49]]
        frame_bgr = x       # x: [70 28 18]
```

Následne je možné prechádzať cyklom pole `x` reprezentujúce pixel a pracovať s jednotlivými hodnotami BGR.

5.3 Zápis výstupného videa

Každý algoritmus zapisuje svoje výstupné snímky do súboru, z ktorého ich treba zapísať do výstupného videa. Pre tento účel slúži funkcia `timelapse()`, ktorá sa zavolá po vykonaní každého algoritmu. Jej parametrami sú cesta k súboru, kde sú snímky zapísané a objekt triedy `VideoWriter` pre zápis výstupného videa. Snímky sa do súboru ukladajú pod číselným názvom podľa poradia, v ktorom boli zapísané, preto sa snímky v tejto funkcii najprv

zoradia pomocou funkcie `naturalsort()`[1]. Zoradené snímky sú v cykle `for` zapisované do videa pomocou metódy `write()`, volanou nad objektom triedy `VideoWriter`. Metóde `write()` je predaný ako parameter snímka, ktorá sa zapíše. Po vyprodukovaní videa sa súbor so snímkami odstráni.

5.4 Algoritmus pre elimináciu nepokojných objektov

Po načítaní argumentu („prepínača“) program vstúpi do časti programu, kde je implementovaná časť pre elimináciu nepokojných objektov. Na začiatku sa vypočíta frekvencia zápisu snímky a okolie. Postup pre výpočet frekvencie zápisu snímky opisujem v sekcii 4.1. Dôležité je opomenúť, že pre zistenie počtu snímiek vstupného videa, ktorý je potrebný pre výpočet, som použila metódu `get()` zavolanú nad objektom triedy `VideoCapture`. Jej vstupným parametrom je funkcia knižnice OpenCV `CAP_PROP_FRAME_COUNT`. Nech *freq* je frekvencia zápisu, potom sa okolie `env` vypočíta takto:

$$env = \frac{freq}{2}$$

Keď sú tieto čiastkové výpočty hotové, zavolá sa funkcia pre elimináciu nepokojných objektov `eliminate_hyperactivity()`. Na vstupe má vytvorené objekty tried `VideoCapture` a `VideoWriter`, hodnota frekvencie zápisu a hodnota okolia. V tejto funkcii sa čítajú snímky v cykle `while` z objektu triedy `VideoCapture`, reprezentujúca vstupné video. Prvá načítaná snímka sa zapíše do súboru na disku. V cykle sa ukladajú snímky do poľa. V tomto poli je uložené okolie, z ktorého sa hľadá najpodobnejšia snímka poslednej zapísanej snímke.

Keď je pole zaplnené požadovaným počtom snímiek, zavolá sa funkcia `simm()`, ktorá vráti najpodobnejšiu snímku. Jej parametrami sú posledná zapísaná snímka a spomínané okolie ako pole. V tejto funkcii v cykle `for` je porovnávaná naposledy zapísaná snímka s každou snímkou zo spomínaného poľa. Pre samotné porovnanie dvoch snímiek som využila funkciu `compare_simm()` z knižnice `sci-kit` [9], ktorá vracia desatinné číslo v rozmedzí 0 až 1, 0 pre odlišné snímky, 1 pre totožné. Pred týmto porovnaním je potrebné snímky previesť do šedej škály, pretože funkcia nepracuje s farebnými snímkami. Pre prevedenie farebného modelu snímky som používa ďalšiu metódu z OpenCV `cvtColor()`, ktorá sa volá nad objektom reprezentujúci snímku. Táto funkcia má ako parameter kód pre konverziu, v tomto prípade `COLOR_BGR2GRAY`. Šedá snímka je uložená v pomocnej premennej, kvôli uchovaniu a vráteniu farebnej snímky. V tomto cykle prebieha zároveň hľadanie maxima z hodnôt, ktoré vracia `compare_simm()`. Stále sa zapamätá najvyššia vrátená hodnota a k nej príslušná snímka. Po dokončení cyklu funkcia `simm()` vráti RGB snímku, pre ktorú `compare_simm()` funkcia vrátila najvyššiu hodnotu podobnosti. Vrátená snímka sa uloží do súboru, tým pádom je „poslednou zapísanou“ a pole sa anuluje pre nové okolie.

Po ukončení cyklu `while` sa na konci funkcie `eliminate_hyperactivity()` volá funkcia `timelapse()`, ktorá snímky zo spomínaného súboru zapíše do videa.

5.5 Algoritmus pre elimináciu svetelných zmien

Implementácia tohto algoritmu má tri základné časti: výpočet aktuálnej svetlosti snímiek, nájdenie polynómu, ktorý ich aproximuje a upravenie pôvodných hodnôt svetlostí so zá-

[1] https://stackoverflow.com/questions/4836710/is-there-a-built-in-function-for-string-natural-sort

merom priblížiť ich k hodnotám polynómu. Pre výpočet hodnôt jasov všetkých snímiek bola implementovaná funkcia `calculate_frames_luminances()`. Hodnota svetlosti jednej snímky sa zistí tak, že sa vypočíta svetlosť pre každý pixel snímky. Svetlosť jedného pixelu sa počíta spriemerovaním jednotlivých zložiek RGB:

$$I = \frac{(R + G + B)}{3} \tag{5.1}$$

Tieto hodnoty sú uložené v poli. Z nich sa počíta polynóm, ktorého výpočet prebieha vo funkcii `fit_luminance_curve()`. Táto funkcia vráti polynóm vypočítaný pomocou metódy `polyfit()` a `polyval()` z knižnice `NumPy`. Metóda `polyfit()` vráti vektor koeficientov minimalizujúce chybu aproximácie, ktorý je na vstupe metódy `polyval()` vracajúci hľadaný polynóm. Zmena jasov kvôli ich priblíženiu k polynómu prebieha vo funkcii `calculate_new_luminances()`. V tejto funkcii je využitá metóda `Pool()` a `map()` z knižnice `multiprocessing`. Metóda `Pool()` zistí počet CPU, ktoré je možné využiť pre funkciu, ktorú volá `map()`. V tomto prípade volá funkciu `set_luminance()`, jej parametrami sú samotný obrázok, jeho hodnota jasu, hodnota jasu, ktorú by mala snímka mať a cesta k súboru, kde sa nové snímky uložia. Pre zmenu jasu obrázka na požadovanú hodnotu gamma korektúrou je potrebné vypočítať samotnú hodnotu gamma:

$$\gamma = 1 + \frac{4 * (l_a - l_b)}{255}, \tag{5.2}$$

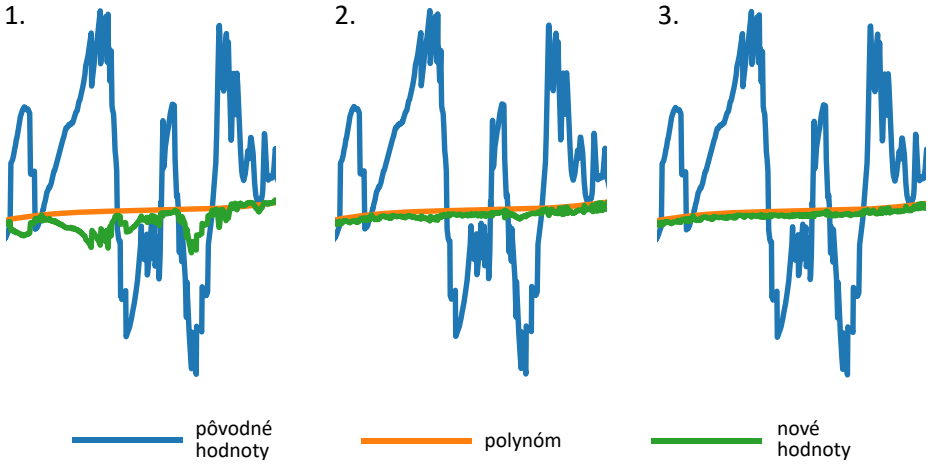
pričom l_a je pôvodná hodnota jasu obrázka a l_b je hodnota jasu, ktorá by mala byť dosiahnutá. Delenie číslom 255 sa vykonáva, kvôli preškálovaniu hodnoty jasu z intervalu prirodzených čísel $\langle 0; 255 \rangle$ na škálu racionálnych čísel $\langle 0; 1 \rangle$. Pripočítanie čísla 1 vo vzorci je kvôli východzej hodnote $\gamma = 1$, kedy nenastáva zmena jasu a teda pri jeho zmene sa hodnota γ pohybuje okolo hodnoty 1. Ak by nastala situácia, kedy $l_a = l_b$, tak sa jas obrázka nezmení.

Pre upravenie jasu sa používa funkcia `adjust_gamma()`. Na jej vstupe je samotný obrázok, ktorého jas bude upravený a vypočítaná γ. Na základe γ hodnoty je zhotovená LUT tabuľka, reprezentovaná poľom, do ktorého sa ukladajú hodnoty následovne:

```
table = np.array([((i / 255.0) ** (1/gamma)) * 255
        for i in np.arange(0, 256)]).astype("uint8")
```

V tejto tabuľke sa namapujú nové hodnoty jasu k pôvodným, pomocou funkcie `LUT()` z knižnice OpenCV, ktorá má na vstupe vytvorenú tabuľku a samotný obrázok. Funkcia vracia upravenú snímku s novými hodnotami svetlostí pixelov podľa LUT. Tento postup od počítania hodnoty γ po mapovanie v LUT tabuľke sa opakuje 3-krát v cykle `for`. V každom ďalšom behu cyklu sa vypočíta nová hodnota jasu z upravenej snímky použitím funkcie `calculate_luminance()`, čím sa získa nová hodnota l_a pre ďalší výpočet γ. Na obrázku 5.2 je vidieť, ako sa nové hodnoty približujú k polynómu po každom prevedení cyklu `for`.

Keď sú všetky snímky upravené zavolá sa funkcia `timelapse()` pre vyprodukovanie videa z týchto snímiek.



Obr. 5.2: Detailný pohľad na úseky grafov znázorňujúce priebehy kriviek po prvom, druhom a treťom priebehu cyklu. Je vidieť, že po každom prevedení cyklu sa krivka znázorňujúca nové hodnoty viac približuje ku krivke polynómu.

5.6 Algoritmus pre elimináciu nežiadúcich objektov

Pre elimináciu nežiadúcich objektov, resp. pohybu, sa zavolá funkcia `eliminate_motion()`. Jej parametrom je objekt triedy `VideoCapture` a objekt triedy `VideoWriter`. V cykle `while` opäť prebieha čítanie snímiek vstupného videa. Aj v tomto algoritme sa pracuje s okolím snímky, preto sa na začiatku uloží 19 snímiek do poľa. Po naplnení poľa potrebným počtom snímiek sa zavolá funkcia `check_env()`, ktorej parametrom je to samotné pole. Spracováva sa 10. snímka v poradí. Pole sa rozdelí na minulosť tejto snímky (prvých 9 snímiek z poľa) a budúcnosť (posledných 9 snímiek). Následne sa hľadajú „šumivé“ pixely. Najprv sa určí interval pre každú RGB zložku na každej pozícii pixelu od $[x_0; y_0]$ až $[x_{N-1}; y_{M-1}]$, pričom rozmer snímiek je $N \times M$. Inak povedané, hľadá sa maximum a minimum pre každú RGB zložku. Tieto hodnoty sa ukladajú do dvoch slovníkov, v jednom sa nachádzajú maximálne hodnoty zložiek RGB a druhom zas tie minimálne hodnoty. Kľúčom v oboch slovníkoch je číselné poradie pixelu od 0 do $(N \times M) - 1$.

Ak sa pixel nachádza na pozícii $[x, y]$ a rozmer snímky je $W \times H$, potom poradie pixelu O sa počíta ako:

$$O = y \times W + x$$

K tomuto kľúču slovníka je priradené pole s RGB maximálnymi alebo minimálnymi hodnotami:

$$\begin{aligned} \{\ 0 &: [R_{min_0}, G_{min_0}, B_{min_0}], \\ 1 &: [R_{min_1}, G_{min_1}, B_{min_1}], \\ &\vdots \\ N &: [R_{min_N}, G_{min_N}, B_{min_{M \times N-1}}]\ \} \end{aligned}$$

$$\begin{aligned}\{\ 0 : [R_{max_0},\, G_{max_0},\ B_{max_0}],\\ 1 : [R_{max_1},\, G_{max_1},\ B_{max_1}],\\ \vdots\\ N : [R_{max_N},\, G_{max_N},\ B_{max_{M\times N-1}}]\ \}\end{aligned}$$

Pre nájdenie minimálnych a maximálnych hodnôt RGB zložiek som implementovala funkcie `find_px_min()` a `find_px_max()`. Obe funkcie majú na vstupe pole snímiek. Z týchto snímiek nájdu pre každú súradnicu pixelu minimálnu alebo maximálnu hodnotu každej zložky RGB a vrátia ich ako už spomenutý slovník.

Týmto sú nájdené intervaly RGB zložiek prislúchajúce k jednotlivým súradniciam. Ak RGB pixelu spracovávaného snímku patria do intervalu, cyklus pokračuje načítavaním ďalších pixelov. Ak hodnoty RGB daného pixelu nepatria do intervalov, tak je pixel šumivý a jeho súradnice sa uložia poľa, v ktorom budú ďalšie súradnice týchto pixelov. Následne je potrebné zistiť, či RGB hodnoty týchto pixelov budú mimo alebo vnútri nových RGB intervalov pixelov, ktoré tentokrát budú zo snímiek z budúcnosti. Postup nájdenia týchto intervalov je taký istý ako pre snímky z minulosti a využité sú rovnaké funkcie. Rozdiel je len v tom, že sa už nekontroluje každý pixel snímiek, ale len tie pixely, ktorých súradnice sú zapísané v poli. To znamená, že ani funkcie `find_px_min()` a `find_px_max()` nebudú vracať slovníky s intervalmi pre RGB zložky všetkých pixelov ale len pixelov na daných súradniciach. Preto tieto funkcie budú mať na vstupe okrem poľa snímiek aj pole so súradnicami. Ak sa RGB hodnoty šumivého pixelu nachádzajú v nových intervaloch, zmena je vyhodnotená ako trvalá a tento pixel sa nebude meniť. V opačnom prípade sa RGB šumivého pixelu nahradí priemerom RGB hodnôt pixelov z minulosti.

Po overení všetkých šumivých pixelov funkcia `check_env()` vráti snímku, ktorá sa zapíše do súboru. Po spracovaní všetkých snímiek sa vo funkcii `eliminate_motion()` zavolá posledná funkcia `timelapse`, ktorá už bola spomenutá.

Kapitola 6

Zhodnotenie

Naimplementované algoritmy dokážu vyriešiť problém, na ktorý sú určené a video dokážu vylepšiť. Nie vo všetkých zhotovených časozberných videách sa prejavovali obmedzenia a tak som mala menej videí vhodných na testovanie, než ich bolo skutočne zaznamenaných. Videá upravené algoritmami a ich pôvodné verzie sa nachádzajú v súbore `videa`, ďalej budú uvedené už len názvy videí.

Predpokladala som, že algoritmus pre elimináciu nepokojných objektov by mohol byť univerzálny a schopný vylepšiť akékoľvek video. Algoritmus som otestovala na časozberných záznamoch z námestí, kde sa nachádzalo veľa pohybu. Pohyb nebol úplne odstránený, ale minimalizovalo to počet prechádzajúcich áut, čím sa video vylepšilo. Výsledok sa nachádza na médiu pod názvom `namesti1_hyper.mp4`. Tento algoritmus som aplikovala aj na časozbernom videu kresby, čo video trochu vylepšilo (`kresba_hyper.mp4`). Taktiež som tento algoritmus aplikovala na odstránenie svetelných zmien ale výsledky sa takmer neprejavili. Vylepšiť som sa pokúšala viacero videí, no vylepšenia sa väčšinou nepreukázali a to kvôli tomu, že časozberné videá boli prikrátke (netrvali ani 5 minút). V okolí bolo príliš málo snímiek pre výrazne vylepšenie videa.

Aj napriek viacerým časozberným videám prírody zachytávajúce slnko a oblohu sa neprejavili prudké svetelné zmeny na každom z nich. Každopádne algoritmus sa prejavil s dobrými výsledkami a to hlavne vo videu `vychodslnka_flick.mp4`, ktoré bolo spomenuté v sekcii 4.6. V pôvodnom videu `vychodslnka.mp4` bol nadmerný prejav blikania.

Algoritmus pre elimináciu nežiadúcich objektov sa prejavil dobre, síce s menšími nedostatkami vo výsledku. Video upravené týmto algoritmom nesie názov `namestie1_move.mp4`. Nevýhodou algoritmu je príliš dlhé spracovanie videa napriek tomu, že video je dlhé len niekoľko sekúnd. Táto doba je nevyhovujúca, hlavne v prípade, ak je pohybu veľmi málo. Myslím si, že tento nedostatok by mohol byť upravený detekciou pohybu, kedy sa spracujú len tie snímky, na ktorých bude zdetekovaný pohyb. Ako som spomenula v sekcii 4.7, detekcia pohybu nebola úspešná kvôli nepresnosti, preto predpokladám, že by mohla nastať situácia, kedy sa neodstráni pohyb aj napriek jeho výskytu.

Všetky časozberné videá upravené eliminačnými algoritmami sa nachádzajú v zozname v prílohe B.

Kapitola 7

Záver

Cieľom tejto práce bolo vytvoriť systém, ktorý minimalizuje alebo dokonca eliminuje obmedzenia časozberného videa. V rámci riešenia som naštudovala problematiku zaobstarania časozberných videí a oboznámila som sa s technológiami pre spracovávanie videa na implementačnej úrovni v jazyku Python a knižnice OpenCV. Pre testovanie systému som zhotovila vlastný dataset, ktorý obsahoval časozberné videá. Doba snímania všetkých týchto videí bola približne 60 hodín. Pre ich zaznamenávanie som okrem vlastnej techniky a online streamovacích kamier použila aj IP kameru zapožičanú z mojej fakulty.

Nepokojné objekty, prudké svetelné zmeny a nežiadúce objekty boli obmedzenia, ktoré som na svojich zhotovených videách mala možnosť pozorovať. Nie všetky zhotovené videá ich obsahovali. Pre každé z obmedzení bol navrhnutý a následne implementovaný eliminačný algoritmus. Po experimentoch sa ukázalo, že použitie algoritmu pre elimináciu nepokojných objektov vie vylepšiť aj videá obsahujúce iné obmedzenia, než pre ktoré bol pôvodne určený. Na základe tohto zistenia sa dá považovať tento algoritmus za multifunkčný.

Výsledkom tejto práce je program, ktorý nasníma časozberné video z obrazu streamovacej kamery, z už existujúceho videa alebo časových snímiek zo súboru. Z týchto videí následne minimalizuje alebo eliminuje obmedzenia, ktorým bola táto práca venovaná.

V budúcnosti by tento program mohol byť rozšírený o GUI. Buď by šlo o softvér alebo webovú stránku. Vývoj GUI by zahŕňal vytvorenie názvu programu a jeho logo. Do programu by sa mohlo pridať viac funkcií, napríklad nastavenie fps, výber formátu (mp4, wmv, avi a pod.) alebo kvality videa (720p, 1080p a pod.) v akej sa uloží video, prípadne pridanie hudby do videa. Aplikácia potrebuje hlavne optimalizáciu, predovšetkým algoritmus pre elimináciu nežiadúcich objektov. Pri spozorovaní ďalších menej bežných obmedzení by mohli byť implementované ďalšie algoritmy.

Počas riešenia tejto bakalárskej práce som sa naučila viac o spracovaní nie len videa ale aj samotných snímiek. Rozšírila som si svoje doterajšie znalosti o videách. Napriek tomu, že som v minulosti pripravila v komerčných programoch pár časozberných videí, nevedela som o princípe ich zhotovovania a na celú problematiku mám nový pohľad. Taktiež som sa naučila ako pracovať so snímkami a s ich pixelmi, porozumela som ako sú snímky a obrázky kódované v počítači, čo mi rozšírilo prehľad o počítačovej grafike, ktorá je mojím koníčkom.

Literatúra

[1] *The OpenCV Reference Manual*. 2.4.13.7. OpenCV, April 2014.

[2] Bowen, C. J. *Grammar of the Edit*. 4. vyd. Routledge, júl 2017. ISBN 978-1138632202.

[3] CodingEntrepreneurs. *How to create a Timelapse with OpenCV and Python // OpenCV PYTHON Tutorial #7* [online]. 2018 [cit. 2020-05-01]. Dostupné z: https://youtu.be/WTan-vRdPto.

[4] Dickinson, N. A. Time-Lapse Acoustic Imaging of the Oceanic Energy Cascade. Júl 2019.

[5] Futai, N., Gu, W., Song, J. a Takayama, S. Handheld Recirculation System and Customized Media for Microfluidic Cell Culture. *Lab on a chip*. Február 2006, roč. 6, s. 149–54.

[6] Guixé, D. Time-lapse photography as an effective method for bat population monitoring. *Barbastella*. Október 2016, Barbastella 9 (1).

[7] Higgins, M. *Time-Lapse Photography Art and Techniques*. 1. vyd. Crowood, október 2016. ISBN 978-1785002090.

[8] Hofmeester, F. *Portrait of Lotte, 0 to 18 years* [online]. 2017 [cit. 2020-22-04]. Dostupné z: https://youtu.be/nPxdhnT4Ec8.

[9] Julian Avila, T. H. *Scikit-learn Cookbook*. 2. vyd. Packt Publishing, november 2017. ISBN 978-1787286382.

[10] Lough, K. *Using time lapse photography as a storytelling device*. 2013. Dizertačná práca. Murray State University.

[11] Priya, K. a Pugazhenthi, D. *Salt and Pepper Noise Removal Algorithm by Novel Morpho Filter* [online]. 2014 [cit. 2020-19-04]. Dostupné z: https://scialert.net/abstract/?doi=jas.2014.950.954.

[12] Pythonista, I. *Downloading Video Streams using Python* [online]. 2019 [cit. 2020-23-04]. Dostupné z: https://youtu.be/bytnxnZFLeg.

[13] Rosebrock, A. *Basic motion detection and tracking with Python and OpenCV* [online]. Máj 2015. Dostupné z: https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/.

[14] Singnoo, J. a Finlayson, G. Understanding the Gamma Adjustment of Images. In:. Január 2010.

[15] Taggart, E. *Loving Dad Makes Time-Lapse of Daughter Growing Up From 0 to 18 Years Old* [online]. November 2017. Dostupné z: https://mymodernmet.com/time-lapse-video-children-growing-up-frans-hofmeester/.

[16] Terry, M., Brostow, G., Ou, G., Tyman, J. a Gromala, D. Making Space for Time in Time-Lapse Photography. Júl 2004.

[17] Udacity. *Smoothing Process Over an Image Using Average* [online]. 2015 [cit. 2020-21-04]. Dostupné z: https://youtu.be/ZoaEDbivmOE.

[18] Vašíček, Z. [online]. [cit. 2020-16-04]. Dostupné z: http://www.fit.vutbr.cz/~vasicek/imagedb/.

[19] Villán, A. F. *Mastering OpenCV 4 with Python*. 1. vyd. Packt Publishing, marec 2019. ISBN 978-1789344912.

Príloha A

Obsah priloženého média

timelapse_maker.py	zdrojový súbor eliminačných algoritmov
dataset	videá z datasetu, ktoré nie sú upravené
thesis	zdrojové súbory pre e-verziu tejto práce
videa	súbor s pôvodnými videami a upravenými verziami
promo_video.mp4	prezenčné video k tejto práci
promo_plagat.pdf	prezenčný plagát k tejto práci
bp.pdf	text práce
README.md	návod k práci so skriptom

Príloha B

Zoznam videí na médiu

Videá sa nachádzajú v súbore `videa`. Názvy videí obsahujú na svojom konci slovo označujúce, algoritmus, ktorým boli spracované: `hyper` – algoritmus pre elimináciu nepokojných objektov, `move` – algoritmus pre elimináciu nežiadúcich objektov a `flick` – algoritmus pre elimináciu prudkých svetelných zmien.

kresba.mp4	Video kresby, kde ruka pôsobí ako nepokojný objekt.
kresba_hyper.mp4	Video kresby po aplikovaní algoritmu pre elimináciu nepokojných objektov. Video je trochu vylepšené.
les.mp4	Video stavby doby prevzaté zo stránky youtube (https://www.youtube.com/watch?v=3o9ilYC16ek), kde sa vyskytuje blikanie.
les_flick.mp4	Blikanie minimalizované z pôvodného videa.
namestie1.mp4	Video námestia z dohľadovej kamery, kde sa prejavuje veľa pohybu ľudí a áut.
namestie1_hyper.mp4	Video námestia, na ktoré bol aplikovaný algoritmus pre elimináciu nepokojných objektov, čo trochu vylepšilo video a pohybu je na ňom menej.
namestie1_move.mp4	Video toho istého námestia po aplikovaní algoritmu pre elimináciu nežiadúcich objektov.
namestie2.mp4	Video iného námestia taktiež z dohľadovej kamery, kde sa prejavuje veľa pohybu ľudí a áut.
namestie2_move.mp4	Video námestia, na ktoré bol aplikovaný algoritmus pre elimináciu nepokojných objektov, čo trochu vylepšilo video a pohybu je na ňom menej.
orgovan.mp4	Časozberné video kvetov orgovánov vo vetre, ktoré sa prejavujú nepokojne. Vo výsledku sú kvety omnoho pokojnejšie.
orgovan_hyper.mp4	Časozberné video kvetov orgovánov vo vetre. Na videu bol aplikovaný algoritmus pre elimináciu nepokoných objetov.

sviecka1.mp4	Časozberné video horiacej sviečky, kde sa prejavuje plamienok veľmi nepokojne, pôvodné video malo dĺžku 1h 43m.
sviecka1_hyper.mp4	Časozberné video horiacej sviečky, na ktorom bol aplikovaný algoritmus pre elimináciu nepokoných objetov. Po jeho aplikovaní je plamienok sviečky pokojnejší.
sviecka2.mp4	Ďalšie video horiacej sviečky pre testovanie.
sviecka2_hyper.mp4	Video horiacej sviečky, na ktorom bol aplikovaný algoritmus pre elimináciu nepokoných objetov. Po jeho aplikovaní je plamienok sviečky pokojnejší.
vychodslnka.mp4	Video zachytávajúce východ slnka, na ktorom prebiehajú prudké svetelné zmeny.
vychodslnka_flick.mp4	Video východu slnka po prevedení algoritmu na elimináciu prudkých svetelných zmien.
vychodslnka2.mp4	Opäť video zachytávajúce východ slnka, na ktorom svetelné zmeny nie sú tak prudké.
vychodslnka2_flick.mp4	Vylepšené video východu slnka po prevedení algoritmu na elimináciu prudkých svetelných zmien. Zmena je menej bádateľná, pretože pôvodné video veľa zmien neprejavovalo. Najväčší rozdiel je vidieť medzi 16. až 25. sekundou videa, pri porovnaní pôvodného a upraveného videa.

Príloha C

Manuál

Vyprodukovanie jednoduchého časozberného videa z videa na disku:

```
-i cesta/k/videu/nazov.mp4 -o cesta/k/ulozeniu/nazov2.mp4 -b -t hh:mm:ss
```

Vyprodukovanie jednoduchého časozberného videa z obrazu streamovacej kamery:

```
-i <odkaz k streamu> -o cesta/k/ulozeniu/nazov.mp4 -r hh:mm:ss -t hh:mm:ss
```

Vyprodukovanie jednoduchého časozberného videa zo snímiek zhotovených metódou photo-lapse:

```
-i cesta/k/súboru -o cesta/k/ulozeniu/nazov.mp4
```

Aplikovanie algoritmu pre elimináciu nepokojných objektov:

```
-i cesta/k/videu/nazov.mp4 -o cesta/k/ulozeniu/nazov2.mp4 -n -t hh:mm:ss
```

Aplikovanie algoritmu pre elimináciu prudkých svetelných zmien:

```
-i cesta/k/videu/nazov.mp4 -o cesta/k/ulozeniu/nazov2.mp4 -f
```

Aplikovanie algoritmu pre elimináciu nežiadúcich objektov:

```
-i cesta/k/videu/nazov.mp4 -o cesta/k/ulozeniu/nazov2.mp4 -m
```

Argument		Funkcia/hodnota
-i	--input-video	cesta alebo odkaz k videu
-o	--output-video	cesta kam sa výstupné video uloží
-t	--timelapse-time	dĺžka výstupného časozberného videa vo formáte hh:mm:ss
-r	--record-time	doba snímania obrazu streamovacej kamery vo formáte hh:mm:ss
-b	--basic-timelapse	prepínač – zhotovovanie časozberného videa z videa na disku
-p	--photo-lapse	prepínač – zhotovovanie časozberného videa zo snímiek na disku
-n	--nervosity	prepínač – eliminácia nepokojných objektov
-f	--flickering	prepínač – eliminácia prudkých svetelných zmien
-m	--motion	prepínač – eliminácia nežiadúcich objektov
-h	--help	vypíše pomocnú správu

Príloha D

Plagát