VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

EVALUACE METOD ZPRACOVÁNÍ OBRAZU PRO ÚLOHU SLEDOVÁNÍ LETÍCÍHO OBJEKTU POMOCÍ DRONU

EVALUATION OF IMAGE PROCESSING METHODS FOR FLYING OBJECT TRACKING USING DRONE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

STANISLAV DANIŠ

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. JIŘÍ KREJSA, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Stanislav Daniš**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Evaluace metod zpracování obrazu pro úlohu sledování letícího objektu pomocí dronu

Stručná charakteristika problematiky úkolu:

Hlavním problémem v úloze sledování rychle se pohybujícího objektu pomocí dronu vybaveného palubní kamerou je identifikace tohoto objektu s minimálním zpožděním. Podstatou práce je evaluace metod zpracování obrazu získaného kamerou. Jedná se o metody již implementované a běžně dostupné například v knihovně OpenCV.

Cíle bakalářské práce:

Seznamte se s metodami zpracování obrazu v úloze sledování pohybujících se objektů.
Navrhněte a realizujte systém obrazů pohybujících se objektů ve vnitřních prostorách.
Vytvořte množinu obrazů s definovanou rychlostí pohybu.
Analyzujte dostupné metody zpracování obrazu s ohledem na odolnost vůči motion blur efektu.

Seznam doporučené literatury:

NURUZZAMAN F.: Open Source Computer Vision for Beginners, Amazon Digital Services LLC

SATYA M.: Object Tracking using OpenCV, dostupné online: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Bakalárska práca bola zameraná na evaluáciu sledovacích algoritmov, obsiahnutých v knižnici OpenCV voči motion blur efektu. Bol vytvorený dataset videí s definovanými rozsahmi vybraných parametrov, ktoré ovplyvňovali vznik motion blur efektu (rýchlosť, vzdialenosť od objektu, pozícia svetla a ďalšie). Každé z videí bolo spracované pomocou sledovacieho algoritmu. Z týchto výsledkov boli v závere formulované zistenia o efektivite každého z algoritmov pre definovanú situáciu.

Summary
The bachelor thesis was focused on the evaluation of tracking algorithms contained in the OpenCV library against the motion blur effect. A dataset of videos was created with defined ranges of selected parameters that influenced the motion blur effect (speed, distance from the object, position of light and others). Each of the videos was processed using a tracking algorithm. From these results, findings on the effectiveness of each of the algorithms for a defined situation were formulated in the conclusion.

Klíčová slova
Motion blur efekt, OpenCV, Spracovanie obrazu, Sledovacie algoritmy, Detekcia farby, Evaluácia sledovacích algoritmov

Keywords
Motion blur efect, OpenCV, Image processing, Tracking algorithm, Color detection, Evaluation of tracking algorithms

DANIŠ, S.*Evaluace metod zpracování obrazu pro úlohu sledování letícího objektu pomocí dronu*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2020. 46 s. Vedoucí doc. Ing. Jiří Krejsa, Ph.D.

Prehlasujem že, som túto prácu vypracoval sám za použitia odbornej literatúry, ktorú som uviedol na konci práce v časti Literatúra.

Stanislav Daniš

Ďakujem rodine, priateľom a známym ktorí ma podporovali a pomáhali mi v priebehu celého štúdia. Ďakujem doc. Ing. Jiří Krejsovi Ph. D. za odborné rady, nápady a pripomienky pri vypracovaní bakalárskej práce.

Stanislav Daniš

Obsah

1. Úvod

Aj keď si to možno niekto neuvedomuje, túžba ľudstva po vytvorení bezpilotného vzdušného prostriedku siaha hlboko do histórie. V 60. rokoch 20. storočia nastal prelom v tranzistorovej technológií a sa začali vo veľkom rozširovať do komerčnej sféry na diaľku ovládané RC modely lietadiel. Zatiaľ čo prvé drony prišli na scénu až na začiatku 21. storočia a to hlavne na vojenské účely. Až v roku 2010 bol na trhu dostupný prvý dron pre bežné využitie. Od tohto okamihu priemysel založený na vývoji, výrobe a predaji dronov neustále rastie a do budúcnosti sa predpokladá zachovanie tohoto trendu. Prvé použitie pre bežného človeka bolo samozrejme iba jednoduché lietanie a užívanie si pohľad cez kameru. Postupne sa však pre drony našlo uplatnenie v ďalších a ďalších odvetviach (napr.: zabezpečenie okolia, výstražné majáky, mapovanie okolia, doručovanie balíkov a podobne). Ich popularita sa dá vysvetliť nástupom cenovej dostupnosti elektroniky a vďaka širokým možnostiam použitia tejto technológie v našom živote.

V oblasti počítačového videnia (computer vision) sú zahrnuté také témy ako spracovanie obrazu, sledovanie objektu, strojové učenie a ďalšie. Dá sa povedať, že tak ako je pre ľudí informácia ktorú získavame z očí (obraz okolia) jedna z najdôležitejších, tak toto isté platí aj pre počítačové videnie. Podobne ako vývoj dronov aj počítačové videnie začalo v 60. rokoch 20. storočia a odvtedy sa táto oblasť vyvinula do veľkosti ako ju dnes poznáme. Zlepšeniu výpočetného výkonu a celková snaha o nájdenie hraníc nových technológií. podporili popularizáciu počítačového videnia

Využitie dronov a počítačového videnia je stále častejšie, tak isto ako aj ich spojenie. Častou úlohou je využitie algoritmov na sledovanie objektu počas toho ako dron letí. V knižnici zameranej na počítačové videnie, ako je napríklad OpenCV, sú takého algoritmy implementované. Aj keď každý z nich funguje teoreticky celú dobu a na každom videu, v skutočnosti závisí na množstve premenných parametrov, ako sú kvalita videa, rýchlosť pohybu, svetlo, pozadie a ďalšie. Spojením týchto parametrov dostávame aj motion blur efekt, ktorý spôsobuje, že algoritmus môže v určitom momente stratiť sledovaný objekt. Pre výber vhodného sledovacieho algoritmu pre danú aplikáciu je nutné vykonať testovanie pri zmene parametrov, avšak za rovnakých podmienok získania všetkých videí.

2. Rešerš

Pre dosiahnutie správnych výsledkov bola spracovaná rešeršia, ktorá je zhrnutá v tejto kapitole. Jednotlivé teoretické základy bolo treba rozobrať a pochopiť ich problematiku. Kapitola sa v prvej časti zaoberala definíciou a vznikom motion blur efektu na fotografií ako aj na videu. Oblasť počítačového videnia obsahuje množstvo spôsobov, ako môže byť vložený obraz upravený a sledovaný. V ďalšom cieli kapitoly boli preto vysvetlené metódy spracovania obrazu a základy sledovania objektu použité pre riešenie zadania.

2.1. Motion blur efekt

Motion blur je efekt, ktorý vznikne ak sa predmet pohybuje pred objektívom a technické nastavenie fotoaparátu/kamery dovoľujú zachytiť tento pohyb. Jedná sa o rozmazanie a natiahnutie pôvodného objektu.

2.1.1. Motion blur efekt na fotografií

Každá zachytená fotografia predstavuje moment v čase, kedy bola fotografia vyhotovená. Uzávierka fotoaparátu rozhoduje o tom, koľko svetla dopadne pri expozícií do objektívu. Dĺžka otvorenia uzávierky sa nazýva expozičný čas. Vo väčšine prípadov je tento čas dostatočný na to, aby bola fotografia zachytená ostro a bez rozmazania. Ak sa však predmet, alebo celá scéna pred objektívom pohybuje, pôvodné nastavený expozičný čas nemusí byť dostatočný a na výslednej fotografií vznikne práve motion blur efekt, ktorý sa javí ako rozmazanie pohybujúceho sa objektu. Vo fotografovaní sa tento efekt používa na vyjadrenie pohybu na fotografií, napríklad práve nastavením požadovaného expozičného času.[1, 2, 3]

(a) Bez motion blur efektu (b) S motion blur efektom

Obr. 2.1: Motion blur efekt na fotografií

2.1.2. Motion blur efekt vo videu

Motion blur efekt sa vyskytuje aj vo videu. Ľudským okom je tento efekt ale nepostrehnuteľný, pretože ľudské oči vytvárajú obraz rovnako ako video, teda už s motion blur

efektom. Ak by však video bolo pustené po jednotlivých snímkoch samostatne, pri nedostatočne rýchlom expozičnom čase by bolo možné pozorovať rozmazanie tak, ako to opisuje podkapitola 2.1.1.

Aby bolo dosiahnuté video bez motion blur efektu potrebovali by sme kameru, ktorá má veľmi rýchli expozičný čas. Takáto technika sa používa napríklad pri natáčaní športových podujatí, aby bolo možné si prehrať spomalený záznam v dobrej kvalite. [1, 2, 3]

2.2. Python

Programovací jazyk Python vytvoril Guido van Rossum na začiatku roku 1990. Je to interpretovaný, univerzálny, interaktívny programovací jazyk, ktorý je voľne prístupný a má veľmi veľkú komunitu používateľov na celom svete. Podporuje objektovo orientované aj štrukturované programovanie. Python cieli na vytvorenie úhľadného, jednoduchého, čistého a čitateľného kódu, ktorému každý porozumie. Použitie Pythonu je možné v množstve aplikácií ako sú návrhy webových stránok, tvorbu grafických rozhraní, vývoj softvéru. Je využívaný aj vo vede na simulácie, analýzu veľkých dát, spracovanie obrazu či strojové učenie. Python obsahuje po inštalácií množstvo modulov v zabudovanej knižnici, ktoré môže užívateľ hneď použiť a zároveň existuje veľké množstvo ďalších voľne prístupných modulov vytvorených komunitou.

V tejto práci bola použitá verzia Python 3.7. Použité dodatočné knižnice v tejto práci sú predstavené v podkapitole 2.2.1 až 2.2.5. [4, 5, 6]

2.2.1. Tkinter

Tkinter je GUI balík (rozhranie), ktorý sa nachádza priamo v štandartnej knižnici Pythonu, slúži na vytváranie grafického rozhrania pre užívateľov. Výhodou je jednoduché a intuitívne navrhovanie celého rozhrania. Nevýhodu, ktorá môže niekoho odradiť, je zastaranejší celkový vzhľad ikon.

V tejto práci bola použitá verzia Tkinter 8.6. Tento balík bol použitý v podkapitole 5.2. [7, 8]

2.2.2. NumPy

NumPy je knižnica zameraná na numerické operácie. Je používaná spolu s knižnicou OpenCV (podkap. 2.2.5) a Matplotlib (podkap. 2.2.4) v jazyku Python[1]. Hlavnou výhodou je tvorba N- dimenzionálnych polí a používania zápisu podobného ako v MATLAB-e[2].

V tejto práci bola použitá verzia NumPy 1.18.1. Tento balík bol použitý v podkapitole 5. [9, 10]

2.2.3. Pandas

Knižnica Pandas pre Python umožňuje užívateľovi jednoduché zaobchádzanie a analyzovanie dát, ktoré sa práve používajú. Výhodou knižnice je, že podporuje prácu s rôznymi dátami, ako sú napríklad CSV, Excel, SQL databázy a ďalšie. Pandas dokáže tieto súbory

[1] Všetky údaje uložené ako pole v OpenCV sú spracované ako Numpy pole
[2] MATLAB - programovací jazyk a prostredie prevažne určený na výpočetné úlohy

previesť na objekty a tak sa práca v Pythone, ktorý podporuje objektovo orientované programovanie, zásadne zjednoduchší. Niektoré z najpoužívanejších možností knižnice:

- nástroje na čítanie a zapisovanie rôznych dát,
- zlučovanie a spájanie dát,
- filtrácia dát.

V tejto práci bola použitá verzia Pandas 1.0.1. Táto knižnica bola použitá v podkapitole 5. [10, 11]

2.2.4. Matplotlib

Matplolib je knižnica, ktorá slúži na vizualizáciu a vytváranie grafov v Pythone. Od jednoduchých bodových a čiarových cez hystogramy, koláčové grafy, časové priebehy po 3D a interaktívne grafy, vytvorené spolu s grafickým rozhraním pre užívateľa. Knižnica ponúka niekoľko možnosti úpravy grafov do požadovanej podoby. Výhodou použitia Matplotlib je to, že tak ako OpenCV (podkap. 2.2.5), používa Numpy knižnicu, čo predstavuje ľahkú integráciu týchto troch knižníc do jedného celku.

V tejto práci bola použitá verzia Matplotlib 3.1.3. Táto knižnica bola použitá v podkapitole 5.5. [12, 13]

2.2.5. OpenCV

OpenCV (open computer vision) je knižnica s otvoreným prístupom zameraná na oblasť počítačového videnia a strojového učenia. Knižnica je vydaná pod licenciou BSD (Berkeley Software Distribution), ktorá dovoľuje súkromné aj komerčné použitie za účelom vytvorenia spoplatneného softvéru. Knižnica obsahuje viac ako 2500 optimalizovaných algoritmov, medzi ktorými sú klasické algoritmy na spracovanie obrazu ako aj algoritmy strojového učenia. Niektoré z modulov ktoré OpenCV obsahuje:

- imgcodecs (vkladanie, ukladanie obrazu),
- imgproc (základné operácie spracovania obrazu- filtrácia, morfologické operácie, ...),
- videio (vkladanie, ukladanie videa),
- video (analýza pohybu medzi snímkami, sledovanie objektov vo videu),
- highgui (vytvorenie grafického rozhrania- okno, reagovanie na myš, ...),
- objdetect (detekcia objektov) a
- tracking (implementované sledovacie algoritmy- v extra moduloch).

V tejto práci bola použitá verzia OpencCV contrib 3.4.5.20. OpenCV contrib je knižnica, ktorá obsahu aj extra moduly. Toto rozšírenie a jeho verzia bola zvolená na základe najlepšej overenej funkčnosti pre potreby zadania. Je vydaná pod licenciou MIT, ktorá má podobné podmienky ako BSD. Táto knižnica bola použitá v podkapitole 2.3, 2.4 a v kapitole 5. [14, 15, 16]

2.3. Spracovania obrazu

Základné spracovanie obrazu sa dá rozdeliť do nasledujúcich 3 krokov:

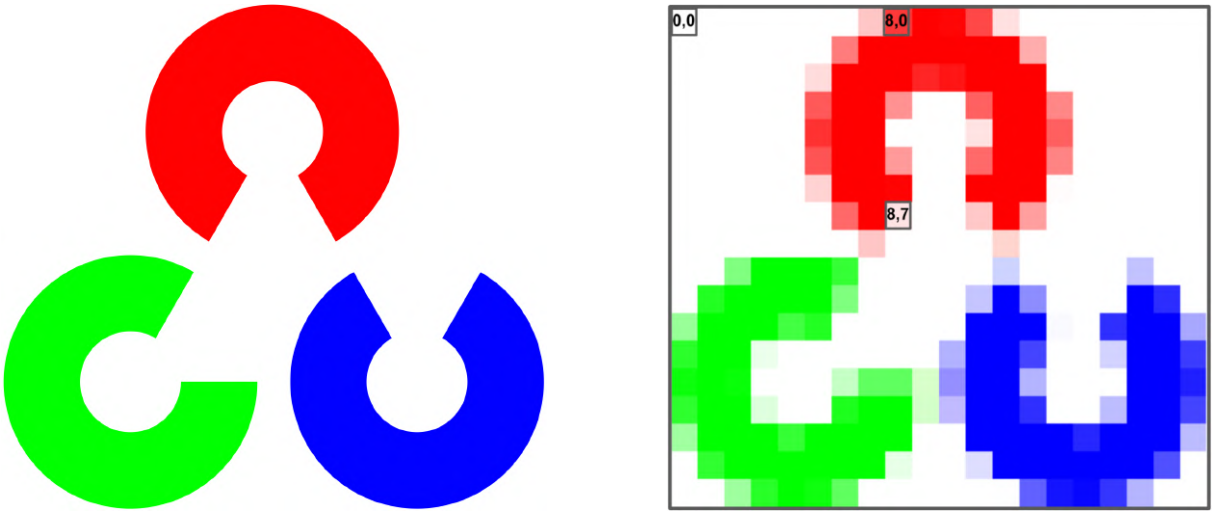
1. Nahranie obrazu/fotky ktorá bude spracovaná.
2. Aplikovanie algoritmu, požadovanej úpravy.
3. Zobrazenie výsledku. [17]

Digitálny obraz

Zachytení 2D obraz 3D sveta je v digitálnom prostredí reprezentovaný ako číselná sada hodnôt, ktoré sa nazývajú obrazové prvky (pixely- px). Digitálny obraz je tak popísaní v priestorových súradniciach pomocou x, y (x- osa, y- osa) tak, ako to je vidieť na obr. 2.2b (možnosť iného zvolenia počiatku [0, 0]).

Každý jeden pixel predstavuje najmenšiu časť obrazu, viac pixelov na rovnakej ploche teda znamená vyššie rozlíšenie. Ak má obraz napríklad rozlíšenie 1200x1114 px (obr. 2.2a), šírka je vytvorená z 1200 px a výška z 1114 px a celkovo je obraz vytvorený z 1 336 800 pixelov. Názorná ukážka obrazu vytvoreného pomocou pola pixelov je na obr 2.2b. Pre lepšiu predstavu je tiež zobrazená fotografia s rozlíšením 20x18 px. Hodnota pixelu záleží na použitom farebnom modely (počet a rozsah kanálov) na obraze. Napríklad hodnota pixelu čierno bieleho obrazu je 1 alebo 0, šedého obrazu od 0 do 255, RGB (červená, zelená, modrá) obrazu od 0 do 255 pre každý kanál. Pre viac kanálové farebné modely vzniká výsledná farba ich kombináciou.

Výsledný digitálny obraz je tvorený pixelmi, ktoré sú umiestnené v 2D poli so súradnicovou hodnotou x a y. A samotný pixel obsahuje hodnotu/y, ktoré tvoria jeho farbu. Takže obraz je 2- rozmerná matica s jednou, alebo viacerými hodnotami v každé pozícií. [16, 17, 18, 19]



(a) Ukážka rozlíšenia 1200x1114 px [20]

(b) Ukážka rozlíšenia 20x18 px [17]

Obr. 2.2: Rozlíšenie loga OpenCV

2.3.1. Úvod do spracovania obrazu v OpenCV

Knižnica OpenCV (podkap. 2.2.5) obsahuje rôzne možnosti spracovania obrazu. V podkapitole 2.3.1 až 2.3.4 sú vysvetlené samostatne tie, ktoré boli použité ako celok v podkapitole 5.4.1.

```
1  # načítanie OpenCV
2  import cv2 s cv
3
4  # nahranie obrazu
5  obr = cv.imread("adresa uloženého obrazu")
6  # rozmery nahraného obrazu
7  (v, s) = obr.shape[:2]
8  print(f"Info o nahranej fotke:\nVýška: {v}\nŠírka: {s}\nTyp uloženia: {type(obr)")
9
10 # vykreslenie kruhu
11 cv.circle(obr, stred, polomer, farba, hrubka)
12 # vykreslenie čiary (b_1 = počiatok, b_2 = koniec)
13 cv.line(obr, b_1, b_2, farba, hrubka)
14 # vykreslenie obdĺžnika (h_l = horný-ľavý rok, d_p = dolný pravý roh)
15 cv.rectangle(obr, h_l, d_p, farba, hrubka)
16 # vypísanie textu (b_1 = počiatok, v_p = veľkos písma)
17 cv.putText(obr, "Spracovanie obrazu v OpenCV", b_1, styl, v_p, farba, hrubka)
18
19 # zobrazenie obrazu
20 cv.imshow("názov", obr)
21 cv.waitKey(0)
22
23 # uloženie obrazu
24 cv.imwrite("adresa uloženia", obr)
```

Obr. 2.3: Skript- nahranie obrazu

Inštalácia OpenCV

Knižnica OpenCV je dostupná pre viaceré programovacie jazyky a operačné systémy. Všetky možnosti inštalácie podľa vybraného jazyka a operačného systému sú uvedené v oficiálnych návodoch na stránke [21].

Nahranie a uloženie fotky/obrazu

Základnou úlohou v spracovaní obrazu je nahranie obrazu, uloženie obrazu a zistenie základných informácií. Tento cieľ je spracovaný na riadku 5 a 24 obr. 2.3.

Na obr. 2.4 sú informácie o nahranom obraze. Typ uloženia je numpy pole tak ako sme v podkap. 2.2.2 povedali, vďaka ktorému je ďalšie spracovanie obrazu ľahšie. [17, 22]

```
Info o nahranej fotke:
Výška: 3456
Šírka: 4608
Typ uloženia: <class 'numpy.ndarray'>
```

Obr. 2.4: Výsledok nahranie a uloženie fotky

Vypísanie textu a vykreslenie tvarov

Dôležitou funkciou v spracovaní obrazu je vykreslenie tvarov (napr.: označenie hľadaného objektu) a vypísanie textu (napr.: priradenie názvu tomu istému objektu pri detekcií viacerých objektov). OpenCV ponúka pri vykreslení geometrických tvarov niekoľko možností (napr.: čiara, kruh, obdĺžnik, elipsa). Na riadkoch 10 - 17 obr. 2.3 je spracovanie takejto úlohy a výsledok je zobrazený na obr. 2.5a. [17, 22]

Súradnicový systém

OpenCV dovoľuje prístup k jednotlivým pixelom na obrázku a to pomocou numpy pola a súradnicového systému ktorým sa riadi. Obr. 2.2b zobrazuje tento súradnicový systém. Vďaka tomuto prístupu sa mení farba jedného pixelu či celej oblasti. Či nahradiť časť obrazu iným obrazom (obr. 2.5b - ľavý horný roh). Spracovanie je na riadku 2 obr. 2.6.

ROI (region of interest - oblasť záujmu) je tvorené vybranou časťou pôvodného obrazu. Táto oblasť je zvolená pred samotným spustením skriptu alebo sa môže meniť podľa voľby používateľa. Spracovanie je na riadkoch 4 - 8 obr. 2.6 a na obr. 2.5b sú ukázané výsledky. [17, 22]

(a) Vypísanie textu a vykreslenie tvarov

(b) ROI a zmena farby pixelov

Obr. 2.5: Výsledky

```
1  # priama zmena farby pixelov
2  z_obr[0:150,0:150] = farba
3  # výber roi
4  roi = cv.selectROI('vyber roi', z_obr)
5  # nakreslenie vybraného roi
6  cv.rectangle(z_obr, (int(roi[0]), int(roi[1])),
7               (int(roi[0]+ roi[2]),int(roi[1] + roi[3])), farba, hrubka)
8  cv.destroyWindow('vyber roi')
9
10 # prevod na formát šedej, RGB a HSV
11 seda = cv.cvtColor(z_obr, cv.COLOR_BGR2GRAY)
12 rgb = cv.cvtColor(z_obr, cv.COLOR_BGR2RGB)
13 hsv = cv.cvtColor(z_obr, cv.COLOR_BGR2HSV)
```

Obr. 2.6: Skript- zmena farby pixelov a prevod farebných formátov

Farebný model v OpenCV

RGB (červená, zelená, modrá) je bežne používaný farebný model v digitálnom obraze. Z historických dôvodov je však v OpenCV použitý farebný model BGR (modrá, zelená, červená). Tento farebný model je v niektorých aplikáciach treba zmeniť prevodom na iný (napr.: RGB, šedý alebo HSV). Na riadkoch 10 - 13 obr. 2.6 je táto úloha spracovaná a na obr. 2.7 sú ukázané výsledky. [17, 22]

(a) Šedý model (b) RGB model (c) HSV model

Obr. 2.7: Prevod do rôznych farebných modelov

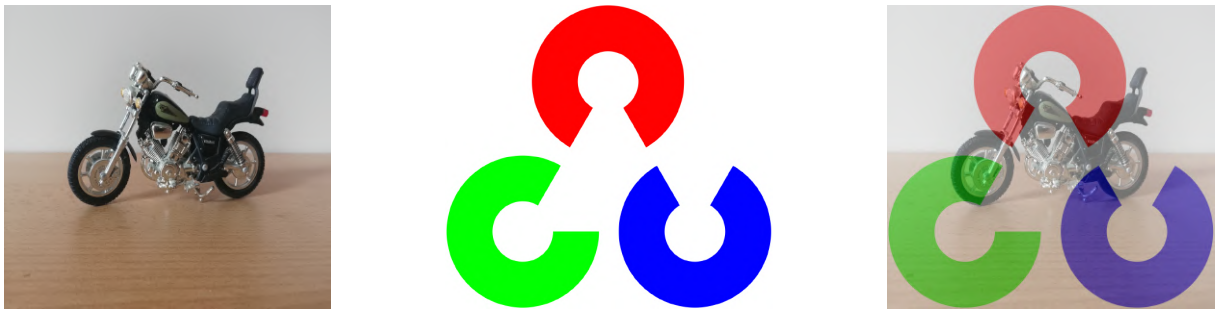
2.3.2. Aritmetické operácie

Zmiešanie dvoch obrazov

Spojenie dvoch obrazov dovoľuje jednoduchá operácia $obr3 = obr1 + obr2$, za podmienky, ak sú obidva obrazy rovnakého typu. Ak má požadovaný výsledok vytvoriť dojem zmiešania alebo priehľadnosti jedného obrazu cez druhý, každému obrázku sa musí pridať váha. Výsledok je vytvorený podľa rovnice:

$$g(i,j) = (1-\alpha)f_0(i,j) + \alpha f_1(i,j) \tag{2.1}$$

Zmena váhy prekrytia je určená hodnotou α od 0 do 1. Spracovanie takejto operácie je na riadku 2 obr. 2.9 a na obr. 2.8 je ukázaný výsledkoch. [17, 22]



(a) Vstup 1 (b) Vstup 2 [20] (c) Výsledok zmiešania

Obr. 2.8: Výsledok- zmiešanie dvoch fotografií

```
1 # aplikovanie prekrytie obr_1 a obr_2
2 prekrytie = cv.addWeighted(obr_1, alpha, obr_2, 1 - alpha, 0, obr_2)
3
4 # aplikovanie erózie a dilatacie na vstup (obraz) (None = 3x3 matica)
5 erozia = cv.erode(vstup, None, iterations=1)
6 dilatacia = cv.dilate(vstup, None, iterations=1)
7
8 # bitové operácie
9 obr_and = cv.bitwise_and(obr_1, obr_2)
```

Obr. 2.9: Skript- aritmetické operácie

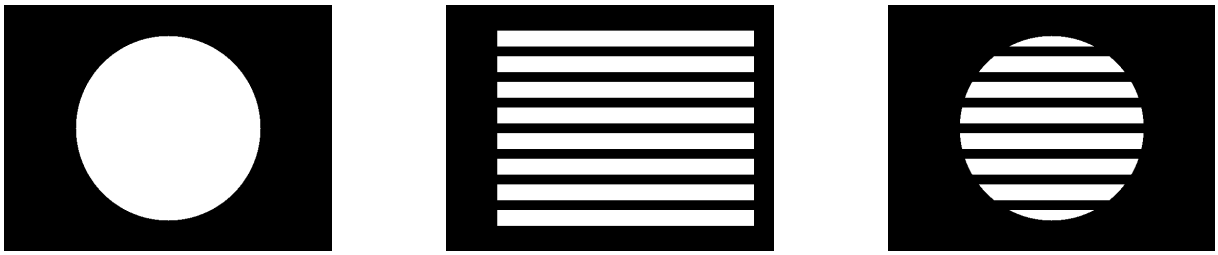
Bitové operácie

Knižnica OpenCV obsahuje bitové operácie AND, OR, XOR a NOT. V tab. 2.1 je vypísaná kombinácia dvoch bitov X a Y a ich výsledky pomocou bitových operácií (bitová operácia NOT vracia obrátenú hodnotu bitu).

X	Y	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Tabuľka 2.1: Tabuľka bitových operácií

Jednoduchosť a rýchlosť bitových operácií sú dôvody prečo sa používajú v množstve aplikácií spracovania obrazu. Kedy na čierno bielom obraze, čierna farba predstavuje 0 a biela 1. Týmto spôsobom sa dá vytvoriť binárna maska (napr.: 2.10a, 2.10b), ktorá slúži napríklad na nájdenie zhody medzi dvoma obrazmi či na filtráciu. Na riadku 9 obr. 2.9 je táto úloha spracovaná a na obr. 2.10 je ukázaný výsledkoch. [17, 22]



(a) Vstupný obraz 1 (b) Vstupný obraz 2 (c) Výsledok

Obr. 2.10: Binárna operácia AND

2.3.3. Morfologické operácie

Morfológia je široká oblasť spracovania obrazu založená na základe tvaru objektu na obraze. Na vykonanie takejto operácie je potreba vstupný obraz (väčšinou bitový, ale môže byť aj šedý) a štrukturálny prvok (kernel) ako vstup, ich kombináciou a použitím operácií z teórie množín vzniká nový obraz.

Najbežnejšie používaný štrukturálny prvok je matica 3x3 (tab. 2.2), takže prvok je oproti matici obrazu x-násobne menší a narozdiel od obrazu nemá počiatok v rohu ale v

1	1	1
1	1	1
1	1	1

Tabuľka 2.2: Štrukturálny prvok 3x3

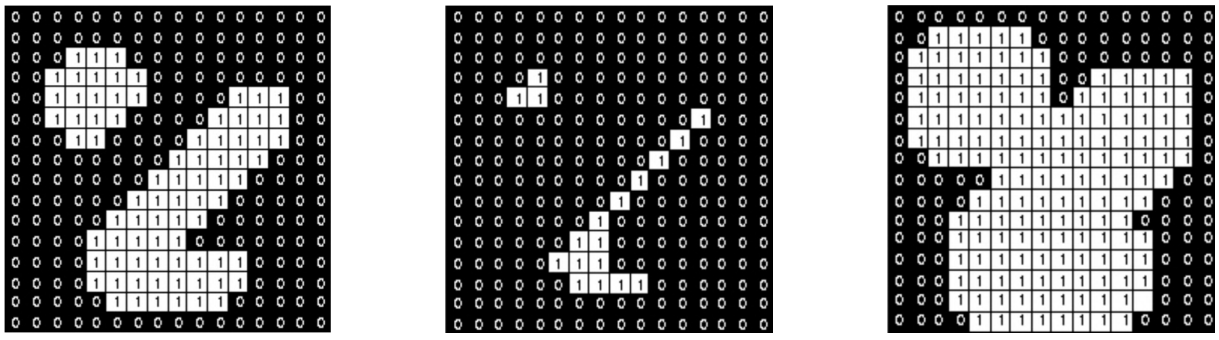
strede matice. Operácia je vykonaná postupným posúvaním kernelu po celom obraze tak, aby sa každý pixel obrazu a jeho okolie odpovedajúce veľkosti kernelu porovnalo s hodnotami samotného kernelu a výsledná hodnota z tejto operácie je iba jedna nová hodnota (pixel) nového obrazu. Veľkosť a tvar štrukturálneho prvku sa volí podľa potreby daného obrazu a požadovaného výsledku. Možnosťou je aj použitie toho istého prvku viackrát za sebou pre dosiahnutie väčšej morfologickej zmeny. OpenCV ponúka množstvo morfologických možnosti úpravy obrazu, ako je erózia, dilatácia, otvorenie (opening), zatvorenie (closing) či morfologický gradient. [17, 22]

Erózia

Cieľom erózie je "potlačiť" hranice objektu (biely pixel- 1), takže objekt sa stane menším a okolie (čierny pixel- 0) sa zväčší. Pri použití štrukturálneho prvku 3x3, aký je na obr. 2.2, a postupným posúvaním sa docieli finálneho stavu, ktorý je na obr. 2.11b. Na riadku 5 obr. 2.9 je úloha spracovaná pomocou OpenCV. [17, 23]

Dilatácia

Výsledkom dilatácie je "posunutie" hraníc objektu tak, aby sa objekt na výslednom obraze zväčšil oproti pôvodnému. Takže rozšírenie oblasti bielych pixelov a zmenšenie oblasti s čiernymi pixelmi. Pri použití štrukturálneho prvku 3x3, aký je na obr. 2.2 a postupným posúvaním sa docieli finálneho stavu, ktorý je na obr.2.11c a na riadku 6 obr. 2.9 je úloha spracovaná pomocou OpenCV. [17, 24]



(a) Pôvodný obraz [23] (b) Po erózií [23] (c) Po dilatácií [24]

Obr. 2.11: Výsledok morfologických operácií

2.3.4. Kontúry

Dôležitou informáciu, ktorá je získaná z obrazu, sú kontúry alebo tiež obrys, ktoré sú definované ako jedna krivka, ktorá spája všetky body okolo objektu (končí tam kde začína).

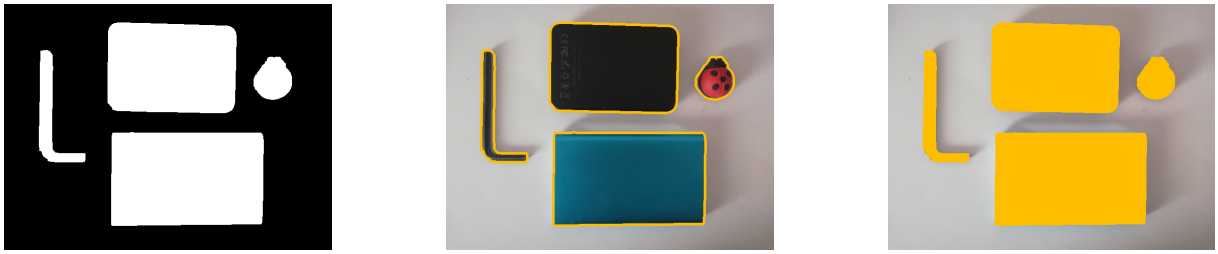
Takto vytvárajú a definujú tvar objektu, ktorý je ďalej použitý v analýze tvaru a detekcií či v rozpoznávaní objektu .

Na nájdenie kontúr na obraze pomocou OpenCV je využitý implementovaný Suzukiho algoritmu (Suzuki's algorithm). Takto nájdený obrys sa uloží do pola ako súradnice bodov. Implementácia v OpenCV dovoľuje použitie hierarchie[3] pre nájdenie a uloženie kontúr, ktoré sa nachádzajú vo vnútri iných kontúr a použitie ďalších funkcií na spracovanie týchto súradníc (napr.: image moments, oblasť kontúr, konvexný obal (convex hull), vykreslenie kontúr). Na obr. 2.12 je táto úloha spracovaná a na obr. 2.13 sú ukázané výsledky. [17, 22]

```
# obraz traba nahrať a vytvoriť jeho šedú kópiu
# pri RGB obaze treba aplikovať treshold napr. na bielu farbu
# podľa potreby sa aplikuje erózia a dilatácia
# nájdenie kontúr obrazu (m_v = metóda vyhľadávania (hierarchie), m_a = metóda aproximácie)
res, contours, hierarchy = cv.findContours(obraz, m_v, m_a)

# vykreslenie kontúr (p = číslo poradia uloženej kontúry (-1 = všetky))
cv.drawContours(obraz, kontury, p, farba, hrubka)
```

Obr. 2.12: Skript- kontúry



(a) Po tresholde (b) Kontúry RGB obrazu (c) Plné kontúry

Obr. 2.13: Výsledok kontúry

2.4. Sledovanie objektu

V podkapitole 2.4.1 sú vysvetlené základy pri práci s videom v OpenCV, podkapitola 2.4.2 predstavuje sledovacie algoritmy, ktoré obsahuje knižnica OpenCV a sú použité v kapitole 5.

2.4.1. Úvod do sledovania objektu v OpenCV

Práca s videom

Základné použitie video pri práci v knižnici OpenCV sa dá rozdeliť na 3 kroky:

1. Vloženie videa,
2. spustenie prehrávania videa,
3. uloženie videa.

[3]hierarchia- spôsob usporiadania kedy môžu mať prvky (okrem prvého) pod sebou podriadené ďalšie prvky

Na obr. 2.14 sú spracované tieto úlohy.

```
# pripojenie videa/kamery
kamera = cv.VideoCapture(0)
# vytvorenie nového videa (FourCC = kód pre kompresiu videa, s_z_s = snímky za sekundu)
nove_video = cv.VideoWriter("cesta uloženia", FourCC, s_z_s, rozlsenie)

# nekonečný cyklus čítania videa
while kamera.isOpened():
#   čítanie jedného snímku (res = true/false pre správne čitanie snímku)
    res, snimka = kamera.read()
    if ret:
#       zapísanie snímky do nového videa
        nove_video.write(snimka)

#       aplikácia operácií na každom snímku
#       ukázanie videa
        cv.imshow("video", snimka)
        k =cv.waitKey(1)

        if k == ord('q'):
            break
    else:
        break

# ukončenie prístupu ku kamere, novému videa a zatvorenie oknien
nove_video.release()
kamera.release()
```

Obr. 2.14: Skript- práca s videom

Porovnanie detekcie a sledovania objektu

OpenCV ponúka široké možnosti detekcie objektu, ako sú Haar kaskáda, zhoda znakov (Feature Matching), HOG + Linear SVM alebo strojové učenie (Deep learning). Aj keď s detekciou je možné dosiahnuť vysokú presnosť sledovania je to na úkor vyššieho výpočetného výkonu (potreba nájsť objekt na každom snímku). A obmedzeniam ktoré plynú z off- line tréningu[4] takéhoto klasifikátoru (zmena natočenia, svetla, tvaru atd. objektu nemusela byť zahrnutá v datasete pri tréningu). Detektor tak nebude výhodné spustiť na niektorých zariadeniach kvôli obmedzenému vnútornému výkonu (napr.: drony, raspberry pi) a pre potrebu zachovania dostatočne rýchlej detekcie a zobrazenia výsledku v reálnom čase.

Pre vyššie napísané problémy sa pri úlohe sledovania objektu používajú namiesto detektorov sledovacie algoritmy (SA). OpenCV obsahuje množstvo spôsobov sledovania objektu pri ktorých je vyžadované iba prvotné označenie (detekcia) a následne je táto informácia predaná algoritmu, ktorý samostatne obnovuje sledovacie okno na základe svojho vnútorného fungovania. Detekcia objektu tak bola spustená iba na jednom snímku (pre korekciu fungovania sa detektor môže zapnúť každých x- snímkov). Pri použití SA sa z každého predchádzajúceho snímku, kde bol nájdený sled. objekt odčítajú užitočné informácie (rýchlosť, tvar, smer pohybu, umiestnenie na obraze atd.) a pomocou nich je možne jednoduchšie predpovedať ďalší pohyb objektu alebo rozoznať ten istý predmet medzi dvoma snímkami. SA sú tak vďaka zachovaným informáciám rýchlejšie ako detektory, ktoré začínajú odznova na každom snímku. A poskytujú lepšie výstupné dáta (sledovanie toho istého objektu). Bežným problémom detekcie objektu na obraze je tiež

[4]off- line tréning- tréning prebiehal na už vytvorenom datasete pred použitím v ďalšej aplikácií

chybná/žiadna detekcia spôsobená napríklad prekrytím sledovaného objektu alebo zmenou osvetlenia povrchu. SA sa s takouto chybou vie lepšie vysporiadať, keďže narozdiel od detektorov využíva on- line tréning priamo na aplikovanom videu. [17, 19, 22, 25]

Použitie implementovaných sledovacích algoritmov z knižnice OpenCV je jedna z možnosti sledovania objektu.

2.4.2. Implementované sledovacie algoritmy

Nasledujúca podkapitola obsahuje opis implementovaných sledovacích algoritmov v knižnici OpenCV. Popisy funkčnosti všetkých SA sú prevzaté z dokumentov podľa ktorých boli implementované do knižnice.

Boosting tracker

Boosting SA je založený na on- line verzií AdaBoost algoritmu. Hlavnou myšlienkou algoritmu je spájanie slabých klasifikátorov do jedného. Tréning klasifikátoru je vykonaný v reálnom čase s pozitívne a negatívne označenými snímkami objektu. Po prvom označení objektu na obraze, SA určí túto snímku ako pozitívnu. Snímky s rovnakou veľkosťou sú z pozadia obrazu zobrané ako negatívne vzorky. Na nasledujúcom snímku je spustený klasifikátor a každej pozícií je priradená hodnota podľa zhody s pozitívnou vzorkou. Do pozície s najväčšou hodnotou zhody je presunuté okno SA. Na ďalšej snímke sa opakuje rovnaký proces. Z každého nového snímku je získaný pre klasifikátor pozitívne označený objekt. Sledovanie sa stáva presnejším a odolnejším voči zmene tvaru objektu. [26]

MIL tracker

MIL SA (Multiple Instance Learning- učenie na viacerých vzorkách) bol podbone spracovaný ako Boosting SA (on- line AdaBoost). Pre MIL nie sú definované pozitívne a negatívne vzorky, ale sety (bags), ktoré obsahujú niekoľko vzoriek obrazu. Setom je priradená pozitívna hodnota ak aspoň jedna vzorka setu je pozitívna. Do pozitívneho setu sa zaradí vzorka aktuálnej pozície a vzorky blízkeho okolia označenia. Takto dostáva klasifikátor obecnejšiu informáciu o tom, čo je pozitívna a čo negatívna vzorka na obraze. [27]

KCF SA

KCF (Kernelized Correlation Filters) vychádza z Boosting a MIL SA. V každom snímku sú nájdené pozitívne označené vzorky sledovaného objektu a okolia. Všetky tieto označenia v sledovaní predstavujú nadbytočné sledovacie okná, keďže sa veľké množstvo z nich vzájomne prekrýva. KCF SA aplikuje Fourierovu transformáciu na tieto označenia a tá dovolí ich rýchle spojenie bez nutnosti ich opakovania. Vďaka tejto operácií SA sleduje objekt rýchlejšie a dosahuje väčšiu presnosť. [28]

MedianFlow SA

MedianFlow SA je založený na predpoklade, že pri sledovaní toho istého objektu, dopredu v čase (snímok 1,2,..) alebo naspäť v čase (snímok 9,8,..), trajektória sledovania je pre

obidva prípady rovnaká. SA je založený na hľadaní chýb a nezrovnalostí medzi týmito trajektóriami. Ich minimalizovaním sa snaží docieliť najlepšie sledovanie objektu. [29]

CSRT SA

CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability- diskriminačno korelačný filter s kanálom a priestorovou spoľahlivosťou) vytvára mapu priestorovej spoľahlivosti (spatial reliability map), ktorá prispôsobuje podporu filtru pre nájdenie sledovaného objektu. Toto riešenie zväčšuje oblasť trénovania zahrnutím viac vzoriek pozadia, čím sa vylepšiju sledovanie vybranej oblasti na obraze. Zároveň predchádza limitácií sledovania obdĺžnikových oblastí. [30]

MOSSE SA

MOSSE (Minimum Output Sum of Squared Error- minimálny výstup súčtu chyby na druhú) je založený na adaptívnom korelačnom filtri. Sledovanie objektu je vykonané pomocou konvolúcie. SA bol vytvorený ako jednoducho implementovateľný algoritmus, ktorý dokáže sledovať objekt pri zmene natočenia, tvaru, svetelných podmienok a aj pri čiastočnom prekrytí. Na rozdiel od bežného poskytuje možnosť spracovať video približne pri 669 snímkoch za sekundu, keď bežný SA spracuje video pri 25 - 30 snímkoch za sekundu.[31]

TLD SA

TLD (tracking, learning and detection- sledovanie, učenie a detekciu) je prezentovaný ako framework než SA, pretože dlhodobé sledovanie objektu rozkladá na tri podúlohy, SA sleduje objekt z jedného snímku do druhého, detektor hľadá všetky možnosti sledovania, ktoré už boli zaznamenané a následne upravuje sledovanie objektu. Treťou vykonávanou podúlohou je učenie sa na chybách detektoru, aby bolo možné sa im vyhnúť v ďalších snímkoch. SA sa snaží vypnúť on- line tréning ak sledovaný objekt je prekrytý alebo úplne zmizol z obrazu a tak sa zakázať učenie zo zle označených vzoriek objektu. [32]

GOTURN SA

GOTURN (Generic Object Tracking Using Regression Networks- Generické sledovanie objektov pomocou regresných sietí) je jediný SA založený na neurónovej sieti (off- line tréning vykonaný na videách). Tracker sa naučil sledovať vzťah medzi pohybom objektu a samotným vzhľadom objektu. Vďaka tomu dokáže sledovať aj nový objekt, ktorý sa nenachádzal v tréningovom sete. Oproti bežným SA poskytuje možnosť spracovania videa pri približne 100 snímkoch za sekundu. [33]

2.5. Práce zaoberajúce sa rovnakou problematikou

Nasledujúca podkapitola prezentuje postup a výsledky dvoch prác, ktoré sa zaoberali evaluáciou niektorých SA implemetovaných v knižnici OpenCV.

2.5.1. Práca autorov Peter Janku a spol.

Názov práce bol Comparison of tracking algorithms in OpenCV. [34] V tejto práci autori prezentujú svoje výsledky pre porovnanie niekoľkých SA. Troch detektorov, ktoré pracuju na princípe zhody znakov (feature detector), troch SA MIL, Boosting, MedianFlow a frameworku- TLD. Autori v práci použili veľký dataset videí s rôzne definovanými problémami pre úlohu sledovania objektu od Yi Wu a ďalších. Pre splnenie zadania tejto práci boli prezentované iba výsledky z videí s motion blur efektom (podkap. 2.1). Pre evaluáciu úspešnosti bola použitá rovnica:

$$C_{suc}(f) = | \frac{r_t \cap r_g}{r_t \cup r_g} |, \tag{2.2}$$

kde $C_{suc}(f)$ je funkcia úspešnosti pre snímok f, r_t je sledovacie okno vrátené zo SA a r_g je správne okno sledovania. Táto rovnica poskytla hodnotu veľkosti prekrytia medzi dvoma oknami. Pre meranie presnosti použili rovnicu:

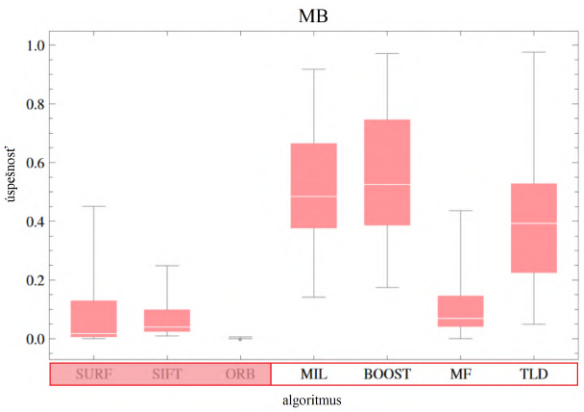
$$C_{prec}(f) = | \frac{r_t}{r_g} |, \tag{2.3}$$

kedy pre najlepšiu hodnotu presnosti bol výsledok rovnice rovný 1. Pre zistenie časovej náročnosti pre každý algoritmus autori odmerali čas pre spracovanie konkrétneho snímku daným algoritmom. Autori pre každé video spočítali snímky, ktorých úspešnosť prekrytia bola väčšia alebo rovná ako 0,5 a zároveň presnosť bola menšia ako 2 a čas vyhodnotenia jedného snímku bol nižší ako 1.

Autori tiež zistili, že počas testovania vznikol u niektorých SA problém, kedy strácali sledovaný objekt hneď na začiatku alebo po určitej dobre. Pre riešenie problému bolo vykonané opätovné spustenie daného SA, po nameraní 30 neúspešných snímkoch.

	SURF	SIFT	ORB	MIL	BOOST	MF	TLD
IV	0.057	0.065	0.000	0.644	0.596	0.138	0.372
SV	0.076	0.089	0.001	0.549	0.559	0.218	0.409
OCC	0.051	0.068	0.000	0.572	0.586	0.185	0.400
DEF	0.023	0.044	0.000	0.616	0.596	0.146	0.388
MB	0.099	0.075	0.001	0.521	0.560	0.114	0.425
FM	0.083	0.086	0.001	0.506	0.521	0.165	0.394
IPR	0.064	0.079	0.001	0.500	0.567	0.206	0.366
OPR	0.053	0.060	0.001	0.562	0.585	0.231	0.372
OV	0.158	0.094	0.002	0.468	0.486	0.124	0.293
BC	0.066	0.076	0.001	0.692	0.661	0.153	0.374
LR	0.100	0.081	0.002	0.488	0.545	0.312	0.373

(a) Tabuľka priemernej úspešnosti [34]



(b) Graf úspešnosti algoritmov [34]

Obr. 2.15: Výsledky pre videá s motion blur efektom

Aj keď predpoklad autorov bol, že TLD bude najlepší z testovaných algoritmov, hlavne kvôli jeho komplexnému riešeniu sledovania, ukázalo sa, že to tak nie je. Prvým dôvodom bolo, že MIL a Boosting boli počas sledovania obnovované kvôli strate sled. objektu. Druhý dôvod bol, že TLD až príliš zasahoval do vlastnej opravy sledovania. Takže sledovanie objektu neprebiehalo tak plynulo, ako by bolo možné. Z výsledkov však autori usúdili, že sledovanie prebiehalo, avšak TLD nedokázal držať sledovacie okno na rovnakých súradniciach referenčného okna. Posledný dôvod uvádzajú nedostatočnú implementáciu v knižnici

OpenCV. Pre porovnanie, graf celkovej úspešnosti algoritmov bol podobný tomu na obr. 2.15b.

2.5.2. Práca autorov Ville Lehtola a spol.

Názov práce bol Evaluation of Visual Tracking Algorithms for Embedded Devices. [35] Autori evaluácie si dali dva hlavné ciele: nájsť ako presné sú SA a zistiť ako efektívne sú implmentované v knižnci OpenCV. V praci boli použité SA: Boosting, MIL, MedianFlow, TLD a KCF. Pre evaluáciu bol použitý dataset 100 videí od Wu and Lim. Evaluácia prebiehala na zariadení Raspberry Pi 3 B v1,2. Táto práca neobsahuje konkrétny problém evaluácie SA na datasete videí s definovaným motion blur efektom.

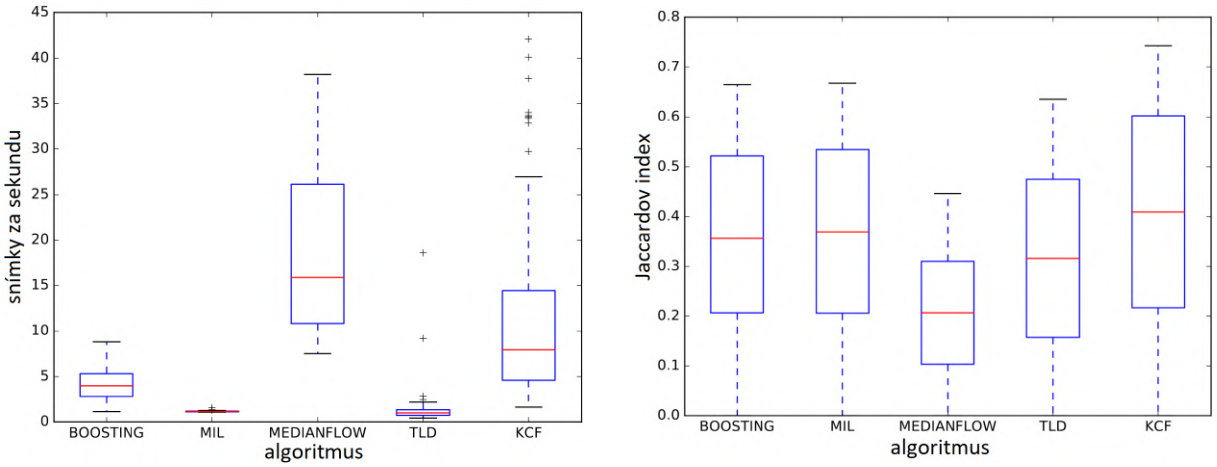
Presnosť evaluácie bola autormi meraná pomocou Jaccardovho indexu:

$$J(A, B) = \frac{A \cap B}{A \cup B} \tag{2.4}$$

Jaccardov index reprezentoval presnosť merania pri porovnaní okna testovaného algoritmu s referenčným oknom. Globálnu podobnosť medzi oknami SA a skutočnými oknami v každom snímku určili autori ako:

$$J_R(A) = \frac{\sum_{i=1}^{n} J(A_i, R_i)}{n}, \tag{2.5}$$

kde n sú snímky videa, R je označené skutočné okno v každom snímku a A sú sledovacie okná algoritmu v každom snímku.



(a) Priemerné FPS pre algoritmus [35]

(b) Rozloženie Jaccardovho indexu [35]

Obr. 2.16: Výsledky

Z obr. 2.16a bolo autormi zistené, že v spracovaní snímkov za sekundu je MedianFlow najrýchlejší, KCF druhý. Podľa výsledkov, jedine tieto dva SA by sa dali použiť v aplikácií, kde treba zobraziť výsledok v reálnom čase. Ďalší nasledoval Boosting, MIL a TLD vyšli ako najpomalšie. Na obr. 2.16b je ukázané rozloženie Jaccardovho indexu (presnosť algoritmov). MedianFlow vyšiel ako najmenej presný a KCF vyšiel s najlepším priemerom. Autori ďalej skúšali SA na konkrétnom videu (futbal) a uviedli nasledujúce výsledky:

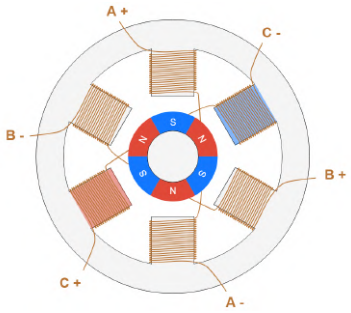
- Boosting SA skoro stratil sledovaný objekt, ale dokázal sa opraviť,
- MIL SA sa postupne zhoršovala presnosť až nakoniec stratil sledovaný objekt,
- u SA MedianFlow bola dosiahnutá veľmi dobrá presnosť aj keď na obr. 2.16b skončil ako posledný (predpoklad, že toto chovanie je spôsobené faktom, že ak tento SA stratí sledovaný objekt, nie je schopný obnoviť pozíciu sledovania),
- TLD na tomto videu nebol schopný sledovať objekt,
- SA KCF ukázal najlepšie sledovanie počas celého priebehu videa.

2.6. Hardvér

V podkap. 2.6 sú vysvetlené hlavné komponenty, ktoré sú použité na vyhotovenie testovacieho zariadenia (podkap. 4). Základný popis funkčnosti krokového motora, jeho ovládania a ďalších použitých komponentov.

2.6.1. Krokový motor

Krokový motor je impulzne napájaný motor s nespojitým priebehom. Po príchode impulzu na vstup, sa motor pootočí o určitý krok a zastaví sa. Ak sú na vstup postupne púšťané vstupné impulzy, rotor sa synchronne otáča s týmito impulzmi. Uhol natočenie hriadele a rýchlosť otočenia sú úmerné frekvencií vstupných impulzov. Kedy pri malej frekvencií je chod motoru trhavý, ale od určitej frekvencie je chod motoru prakticky plynulý. Postupným privádzaním impulzov na cievky fázy A, B, C, A, B sa krokový motor otáča v smere hodinových ručičiek (obr. 2.3). Pre dosiahnutie jemného kroku sa u skutočných motorov používa väčší počet cievok. [36, 37, 38]



Obr. 2.17: Krokový motor- princíp činnosti [39]

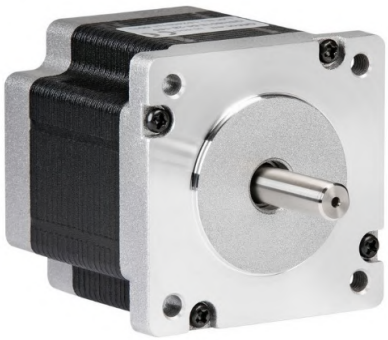
Ovládač A4988 riadi pohyb a chod krokového motora. Krokové motory riadi budením fázy vinutia v istej časovej postupnosti. Ovládač musí zaistiť výkonové budenie fáz motoru a vytvoriť časovú postupnosť budenia.

Pre podmienku presnosti motorov pri opakovanom pohybe boli vybrané 2 krokové motory (podkap. 2.3) Nema 17 (obr. 2.19) do TZ. V tab. 2.3 boli uvedené špecifikácie týchto motorov.

Model	17HS4401
Krokový uhol	1,8°
Menovitý prúd	1,7 A
Menovitý napätie	12 V
Krútiaci moment	0,4 Nm
Počet drôtov	4
Počet fázi	2

Tabuľka 2.3: Parametre a špecifikácie krokového motoru

Obr. 2.18: Microsoft LifeCam HD-3000 [40]



Obr. 2.19: Krokový motor Nema 17 [41]

2.6.2. Ovládač A4988

A4988 je ovládač bipolárných krokových motorov, ktorý slúži na nastavenie kroku motora so zabudovaný prekladačom ("translator") pre ľahké ovládanie. Vďaka zabudovanému prekladaču je možné krokový motor ovládať iba pomocou 2 pinov z kontroleru (podkap. 2.6.3). Jeden pre určenie smeru rotácie a druhý pre riadenie krokov motora. Ovládač umožňuje výber z piatich módou pre nastavenie kroku motora- celý, 1/2, 1/4, 1/8 a 1/16. Logické napätie je od 3 V do 5,5 V, maximálny prúd pre fázu je 2 A s chladičom a 1 A bez chladiča. Požadované prevádzkové napätie je od 8 V do 35 V. *Výrobca: Pololu.*

2.6.3. Arduino

Arduino je elektronická platforma, ktorá spojuje ľahko použiteľný hardvér a softvér. Arduino dosky môžu byť použité na čítanie vstupov, ako je stlačenie tlačidla, a vytvoriť na základe toho výstup (napr.: spustiť motor, prepojiť obvod, zapnúť svetlo). Doska je riadená pomocou vytvorených inštrukcií v programovacom jazyku Arduino s použitím vývojového prostredia (IDE). Existuje niekoľko druhov dosiek, ktoré sa líšia na základe veľkosti, čipu, možností použitia (napr.: Arduino UNO, Arduino Nano a ďalšie). Niekoľko výhod použitia:

- jednoduché ovládanie a obrovská komunita používateľov,
- jednoduché vytvorenie prototypu zariadenia bez nutnosti veľkých znalostí elektroniky a programovania,
- obsahuje veľko množstvo komunitou vytvorených knižníc pre rozšírenie. [42]

2.6.4. Luxmeter

Luxmeter je zariadenie slúžiace na meranie intenzity svetla. Luxmeter, ktorý bol použitý počas testovania je založený na senzore Adafruit TSL2561, komunikácia s PC bola zaistená pomocou Arduina Nano. Senzor je schopný detekovať intenzitu v rozsahu od 0,1 do 40 000+ Lux. *Výrobca: Adafruit.*

2.6.5. Kamera

Pri výbere kamery bolo požadované, aby mala stredne dobré rozlíšenie a aby nebolo nutné riešiť zaostrovanie (tzv. fixed focus). Na celý priebeh testovania bolo vybraná kamera Microsoft LifeCam HD-3000 (obr. 2.18), ktorá bola použitá pre vytvorenie datasetu videí na TZ. Parametre boli zobrazené v tab. 2.4. *Výrobca: Microsoft*

Rozlíšenie videa	1280 × 720 px
Reálne rozlíšenie	720p pri 30 snímkach/s
Pomer strán	16 : 9
Rozhranie	USB 2.0
Rozmery	109 × 44,5 mm

Tabuľka 2.4: Parametre a špecifikácie kamery Microsoft LifeCam HD-3000

Na kamere bolo vypnuté automatické nastavovanie expozície, ktoré spôsobovalo chyby vo videu.

3. Formulácia problému a ciele práce

V kap. 2 boli spracované všetky potrebné teoretické základy, ktoré sú treba zvládnuť na to, aby mohlo byť zadania tejto bakalárskej práce vypracované. V podkap. 2.5 boli následne predstavené dve práce s podobným zameraním ako je táto. Vyhodnotenia autorov poskytli predpoklady na výsledky práce.

V nasledujúcom zozname sú uvedené ciele práce, ktoré bolo treba dosiahnuť:

1. Zoznámenie sa s metódami spracovania obrazu pre knižnicu OpenCV.
2. Navrhnutie a realizovanie systému obrazov pohybujúceho sa objektu vo vnútorných priestoroch.
3. Vytvorenie množiny obrazov s definovanou rýchlosťou pohybu.
4. Analyzovanie dostupných metód spracovania obrazu s ohľadom na odolnosť voči motion blur efektu

Okrem cieľu 1 bol každý z cieľov práce rozdelili na niekoľko samostatných častí. Do nasledujúceho textu boli napísané ich stručné zadania a približný postup riešenia.

Cieľ 1- tento cieľ práce je splnený priamo v podkap. 2.3, kde boli vysvetlené všetky možnosti počítačového videnia, ktoré boli použité z knižnice OpenCV na spracovanie zadania. A podkap. 2.4, ktorej cieľom bolo ukázať rozdiel medzi detekciou a sledovaním objektu. A následné krátke predstavenie implementovaných SA.

Cieľ 2 bol rozdelený na návrh testovacieho zariadenia (kap. 4) a jeho fyzickú realizáciu. Zariadenie vzniklo na základe potreby vykonať experimet vo vnútorných priestoroch s možnosťou kontrolovanej zmeny hodnôt parametrov. V kap. 4 sú napísané všeobecné predstavy o tom, čo od zariadenia bolo očakávané ako aj konštrukčný návrh a návrh ovládania. Zariadenie bolo následovne zostavené na základe tohto návrhu. Fyzická realizácia nie je v práci dokumentovaná.

V cieli 3 je prvá časť tvorená výberom vhodných parametrov, ktoré ovplyvňovali vytvorenie motion blur efektu na videu. Podmienkou bola možnosť realizácie zmeny parametrov vo vnútorných priestoroch. Nasledujúcou časťou, bol výber vhodného základného rozsahu pre každý z parametrov. Cely tento proces bol popísaný v podkap. 5.1. Po výbere vhodných parametrov a ich rozsahu bolo v podkap. 5.2 vytvorené polo automatizované nahrávanie videí datasetu.

Cieľ 4 popisoval celkovú aplikáciu SA, získané dáta a predstavenie ich grafických či iných výsledkov. V podkap. 5.4 boli vysvetlené referenčné a sledovacie okná, ktoré slúžia na určenie úspešnosti SA. V závere kap. 5 boli popísané výsledky pre každý SA a zobrazené grafy pre úspešnosť sledovania. Hlavné výsledky boli venované odolnosti SA voči motion blur efektu a popisu ich kritických parametrov, ktoré spôsobili zhoršenie alebo stratu sledovaného objektu.

4. Testovacie zariadenie

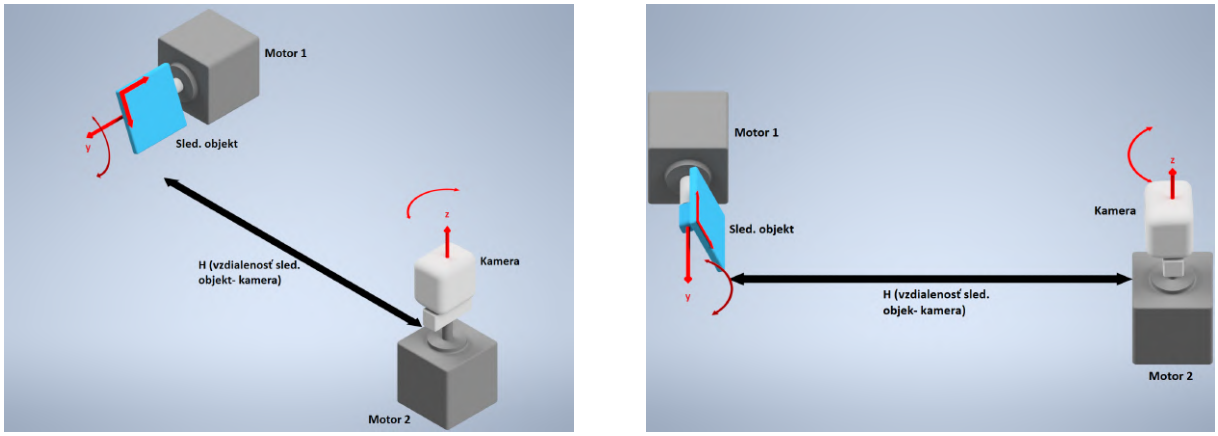
Táto kapitola sa venuje návrhu konštrukcie a ovládania testovacieho zariadenia (TZ) pre dosiahnutie všetkých požiadavkov od vytvoreného datesetu. Zariadenie bolo následne vytvorené a použité v evaluácií SA.

4.1. Spracovanie návrhu testovacieho zariadenia

Pred samotným návrhom TZ bol vytvorený zoznam požiadavok na zariadenie:

- ovládanie pomocou PC,
- jednoduchosť návrhu konštrukcie a ovládania,
- vytvorenie efektívneho testovania pri zmene sledovaných parametrov,
- reprezentovaný pohyb dronu pri sledovaní objektu,
- zmena vzdialenosti pri zachovaní rovnakého rozloženia medzi kamerou a sledovaným objektom.

Na obr. 4.1 je vytvorený model pomocou programu Inventor, na ktorom bola ukázaná základná myšlienka, ako rozložiť 3D pohyb kamera/dron - objekt. Kamera prichytená na drona zaznamená tento pohyb počas letu. Motor 1 natáčal sledovaný objekt okolo osi y z polohy, keď je objekt zvisle orientovaný do polohy kolmo na kameru. Motor 2 otáčal kameru okolo osi z v rozsahu, keď je na obraze zachytený pravý okraj sledovaného objektu pri pravom okraji obrazu a končí v mieste, keď je na obraze ľavý okraj sledovaného objektu pri ľavom okraji obrazu zachyteného z kamery. Takto bola v tomto modely simulovaná pomocou motoru 1 situácia pohybu sledovaného objektu z pozície pod dronom do pozície, kedy sa dron a sledovaný objekt ocitnú na rovnakej úrovni. Motorom 2 simuluje pohyb postupného vodorovného posunu sledovaného objektu. Zmena hodnoty H simuluje zmenu vzdialenosti medzi sledovaným objektom a kamerou (dronom).

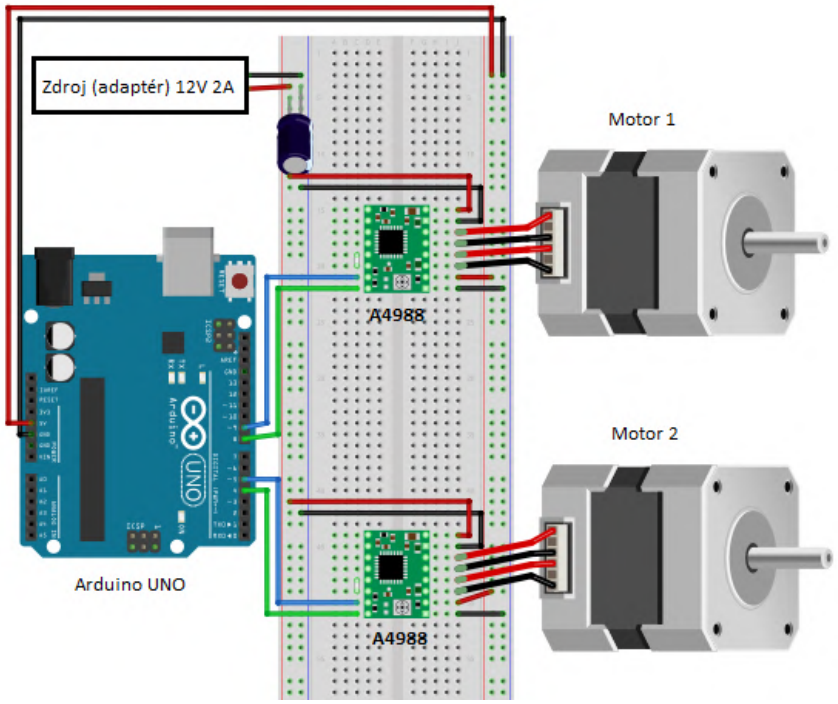


(a) Od zadu (b) Zo strany

Obr. 4.1: Základná myšlienka TZ

4.1.1. Návrh ovládania

Pri výbere ovládania, v tomto prípade krokových motorov, boli požadované tri funkcie: jednoduchá možnosť zmeny rýchlosti kamery, zmena kroku natočenia sledovaného objektu a ovládanie priamo pomocou PC. Pre ovládanie preto bola vybraná platforma Arduino UNO (2.6.3) spolu s ovládačmi (driver) krokových motorov A4988 2.6.2. Konečný návrh zapojenie je ukázaný na obr. 4.2.

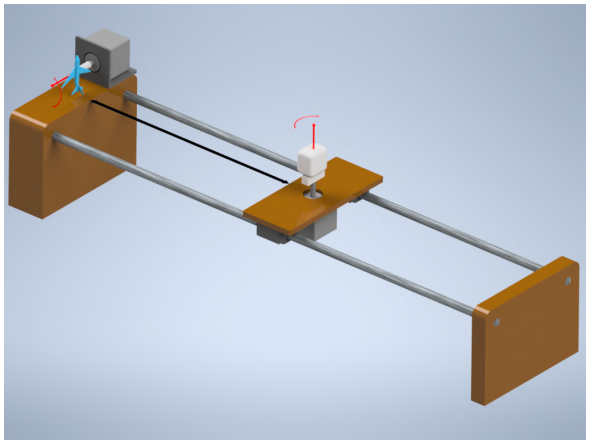


Obr. 4.2: Zapojenie ovládania

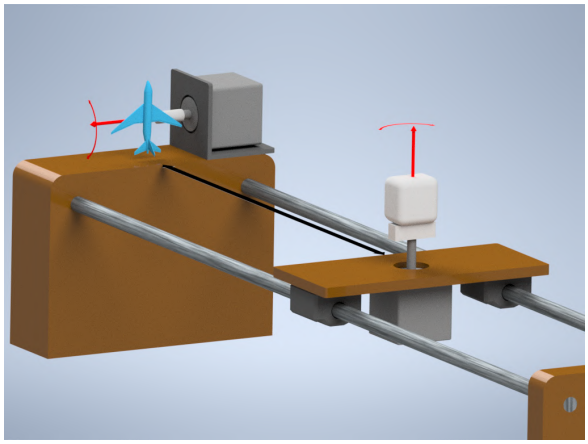
Krokové motory sú ovládané jednoduchým kódom, ktorý bol napísaný pre arduino. Vytvorený opakujúci sa "for" cyklus postupne posielal pulzy do pinu, ktorý ovláda krok motora. Medzi zapnutím a vypnutím tohto pinu boli ponechané určité oneskorenia, ktoré sú rozhodujúce pi stanovení rýchlosti otáčok motora. Napríklad ak bol motor nastavený na plný krok (1,8°) a bolo požadované, aby sa motor otočil o 360°, tak bolo do pinu na ovládanie krokov poslaných 200 impulzov.

4.1.2. Celkové vyhotovenie

Na obr. 4.3 je model konečnej realizovanej podoby TZ. Maximálna nastaviteľná vzdialenosť kamery od sledovaného objektu $H = 620\ mm$. K sade TZ bol navyše pridaný luxmeter na meranie intenzity svetla, ktorý zároveň dokáže posielať hodnoty do PC a jedna LED dioda umiestnená v ľavom dolnom rohu sledovaného objektu. Význam a použitie diody bol vysvetlený v podkap. 5.4.1.

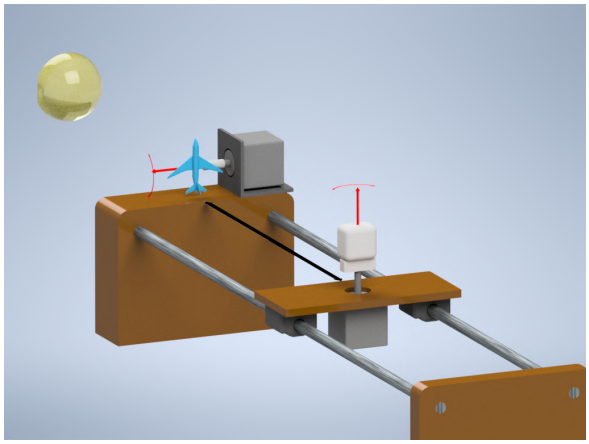


(a) Pohľad z boku

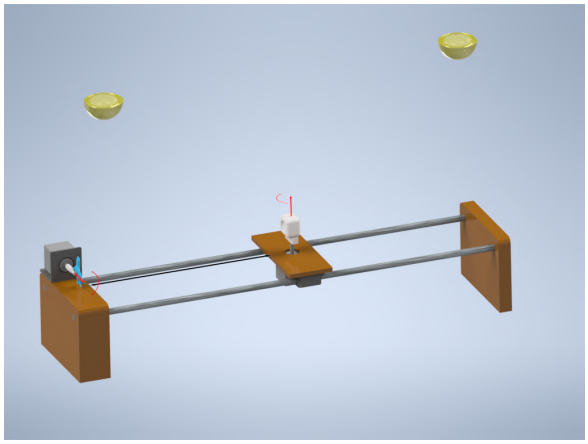


(b) Pohľad od zadu

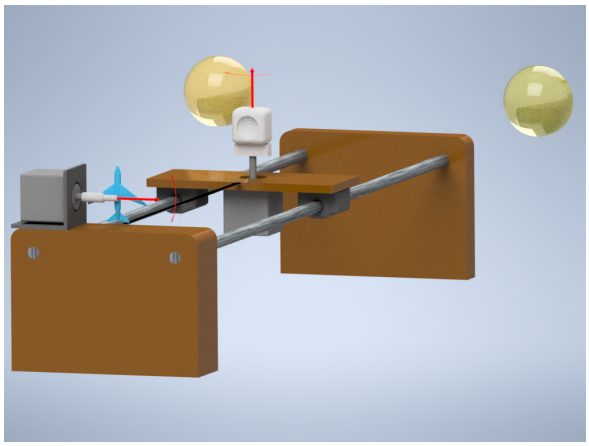
Obr. 4.3: Finálne vyhotovenie TZ



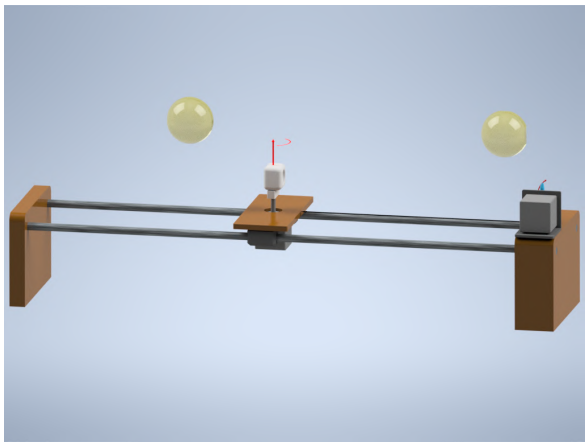
(a) Z predu



(b) Z vrchu



(c) Zo zadu



(d) Z boku

Obr. 4.4: Pohľad na TZ- poloha svetla

5. Experiment

Táto kapitola je venovaná celému priebehu experimentu. V prvej časti je uvedený rozsah parametrov použitých pre vytvorenie datasetu. Získanie dát po aplikácií SA a celkové vyhodnotenie experimentu a ukázanie grafických výsledkov (grafy) sú uvedené v druhej časti.

5.1. Podmienky a parametre testovania

Pri vytvorení datasetu boli dodržané rovnaké podmienky pre každé video s rovnako nastavenými parametrami. Následne tak bolo možné výsledky medzi sebou porovnať s minimálnymi chybami. Podmienky pre testovanie, ktoré boli nastavené a dodržané na každom z vytvorených videí datasetu:

- pozadie za sledovaným objektom bolo zvolené ako jednofarebné (predpoklad oblohy či objektov vzdialených tak ďaleko, že pozadie splýva do jednej farby),
- sledovaný objekt začínal v určenej nulovej polohe (pohľad z vrchu na lietadlo) a končil v polohe o 90° pootočený oproti začiatočnej polohe,
- kamera začínala v polohe, kedy je pravá hrana sledovaného objektu na pravom okraji obrazu z kamery a končila v polohe, kedy je ľavá hrana sledovaného objektu na ľavom okraji obrazu z kamery,
- automatické nastavenie expozície kamery bolo vypnuté,
- krok pohybu kamery a sledovaného objektu bol zvolený vždy rovnako,
- luxmeter je orientovaný vždy smerom na zdroj svetla a umiestnený pri sledovanom objekte,
- nahrávanie bolo vykonané bez kompresie videa pri 30 snímkoch za sekundu s rozlíšením 640 x 480 px.

Pri výbere parametrov, ktoré sa menili na jednotlivých videách boli vybrané tie, ktoré výrazne ovplyvňovali úroveň motion blur efektu (podkap. 2.1) a spôsobovali stratenie sledovaného objektu. Vybrané parametre: zmena natočenia sledovaného objektu (krok), rýchlosť pohybu kamery, vzdialenosť medzi sledovaným objektom a kamerou H, poloha osvetlenia, intenzita osvetlenia. Obr. 4.4 obsahuje viacero vymodelovaných TZ so všetkými polohami svetla.

Základné testovanie

Základné testovanie slúžilo na nájdenie rozsahov zvolených parametrov v medziach, kedy výsledky pre každý SA a hodnoty nastavenia rýchlosti, vzdialenosti, umiestnenia svetla a intenzity dávali ešte stále pozitívne výsledky. Tab. 5.1 ukazuje vybrané základné rozsahy každého parametru. Základné testovanie zahrňovalo aj prvotné vyskúšanie SA, ako súčasť určovania základného rozsahu parametrov. Pri tomto testovaní sa zistilo, že SA GOTURN (podkap. 2.4.2) nie je schopný sledovať objekt, tak ako je určený v tomto zadaní (na sledovanie tváre či ruky funguje dostatočne dobre- bol vykonaný jednoduchý test).

Rýchlosť	Vzdialenosť H	Intenzita	Poloha svetla	Natočenie
[ot/min]	[mm]	[lux]	pohľad	[°]
3	200 (36 %)	1300-1500	z predu	0
5	360 (10 %)	600-700	z vrchu	3,6
9	620 (4 %)	300-350	zo zadu	...
12	-	80-100	z boku	...
18	-	50	-	90

Tabuľka 5.1: Rozsahy parametrov

Pre prvú a druhú vzdialenosť boli ešte vytvorené videá s rýchlosťou 20, 23 a 26 ot/min. Každé video bolo nahrané pri zmene niektorého z vybraných parametrov a ku každému videu bol vytvorený excel súbor (obsahuje: luxmeter hodnota, natočenie sledovaného objektu a číslo snímky (frame)), ktorý bol pomenovaný podľa nastavený parametrov na videu (napr.: Vz1_Sa_Psb).

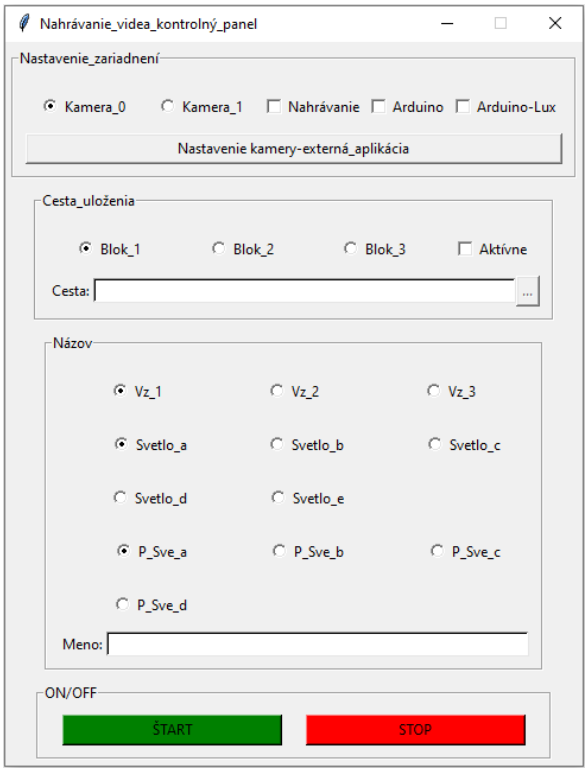
5.2. Polo automatizácia vytvorenia datasetu

Najjednoduchší zmeniteľný parameter bola rýchlosť rotácie kamery. Keďže zmena bola vykonaná automaticky pomocou PC. Zatiaľ čo ostatné parametre bolo nutné meniť manuálne. V takom prípade hrozila väčšia chyba rozptylu pre videá s rovnako nastavenými parametrami (napr.: schopnosť dva krát trafiť rovnaké miesto pozície svetla). Pre zníženie takéhoto rozptylu na minimum bolo testovanie na jedno konkrétne rozloženie (poloha svetla, intenzita svetla a vzdialenosť H) vykonané pre všetky rýchlosť na jeden krát (jedno video). Takéto nahrávanie zaistilo lepšiu efektívnosť, presnosť a umožnilo neskôr použiť iba jedno označenie na každé video z rovnakej vzdialenosti H (celkovo 3 vzdialenosti = 3 označenia). Nasleduje jednoduchý popis celej testovacej procedúry každého videa.

1. Nastavenie parametrov (poloha svetla, intenzita svetla a vzdialenosť H).
2. Bola nastavená prvá rýchlosť z tab. 5.1 rotácie kamery.
3. Sledovaný objekt a kamera má polohu 0.
4. Bolo spustené nahrávanie videa.
5. Kamera sa otočila do svojej koncovej polohy a vrátila sa naspäť.
6. Sledovaný objekt spravil krok +3,6 a cyklus bodov 4. a 5. sa opakoval.
7. Ak sledovaný objekt dosiahol svoju koncovú polohu. TZ počká na vrátenie objektu do polohy 0.
8. Rýchlosť otáčania kamery bolo zmenená podľa tab. 5.1
9. Ak bola použitá posledná rýchlosť z tab. 5.1, nahrávanie bolo spustené od bodu 1. so zmenou nastavených parametrov (poloha svetla, intenzita svetla a vzdialenosť H).

Pre jednu vzdialenosť H a základný rozsah parametrov z tab. 5.1 bolo vytvorených 20 videí. Celkovo tak 60 videí pre tri vzdialenosti H. Excel súbor v prílohe ?? obsahuje plán testovania, kedy jeden riadok predstavuje jedno video.

Ná záznam videa bol použitý GUI (obr. 5.1) vytvorený pomocou Tkinter (podkap. 2.2.1). Hlavnou výhodou bola rýchla zmena názvu videa bez nutnosti reštartovať celý skript. Ďalšími výhodami boli funkcie, ako výber používanej kamery, nastavenie možností



Obr. 5.1: GUI pre nahrávanie videa

nahrávania, voľba miesta uloženia, voľba názvu súboru (možnosť ponechať generovaný) či nastavenie parametrov kamery pomocou externej aplikácie

5.3. Experiment vs realita

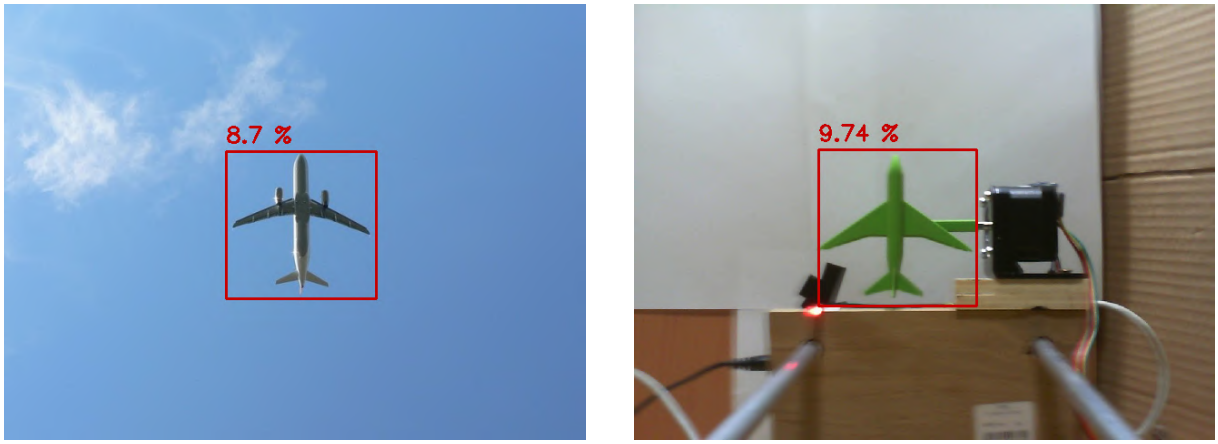
Táto podkapitola sa zoberala možnosťami aplikácie dosiahnutých výsledok na reálnu situáciu (objekt sledovaný dronom). Kvôli existujúcemu rozdielu medzi skutočnými parametrami (rýchlosť objektu, intenzita osvetlenie a ďalšie) a parametrami, ktoré boli zvolené (tab. 5.1) bolo treba definovať tento rozdiel.

V tab. 5.1 bol vybraný rozsah parameteru rýchlosť rotácie kamery. Rýchlosť, ktorá však bola v reálnej aplikácií problémová je rýchlosť sledovaného objektu. Z experimentu mohla byť táto rýchlosť získaná pomocou rovnice:

$$v = \omega * R; \tag{5.1}$$

kde v je rýchlosť objektu v metroch za sekundu, ω rýchlosť kamery v otáčkach za sekundu a R polomer reprezentujúci vzdialenosť kamera- objekt. Kvôli obmedzeniam TZ

bolo zaoblenie vzniknuté rotačným pohybom kamery okolo svojej osi iba minimálne. Keďže vzdialenosť H a skutočná vzdialenosť medzi kamerou a dronom nebola určená pomerom, za polomer R v rovnici bola dosadená vzdialenosť H, ktorá odpovedala percentuálnemu prekrytiu objektu na obraze, uvedeného v tab. 5.1 v stĺpci Vzdialenosť H v zátvorkách. Táto vypočítaná hodnota v percentách udáva koľko z obrazu kamery tvorí označený sledovaný objekt. Pre reálnu aplikácie by tak bola vybraná z tab. 5.1 vzdialenosť H, ktorá odpovedá hodnote prekrytia pre danú vzdialenosť. Následne z toho vypočítaná rýchlosť v sledovaného objektu. Obr. 5.2 ukazuje túto aplikáciu.



(a) Reálna situácia [43]

(b) Situácia z experimentu

Obr. 5.2: Experiment vs realita

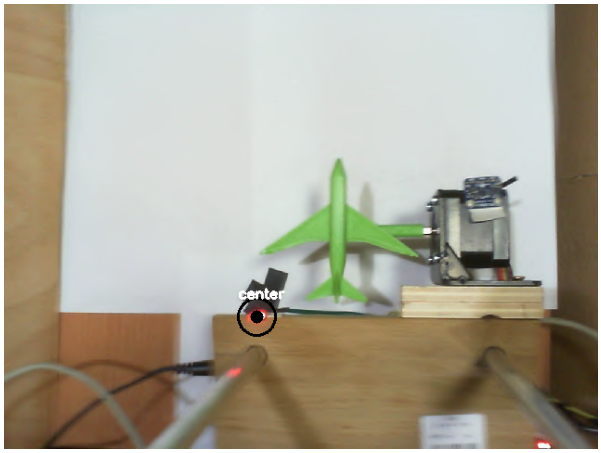
Ostatné rozsahy parametrov by boli v reálne situácií použité rovnaké aké sú uvedené v tab. 5.1.

5.4. Úspešnosť sledovania

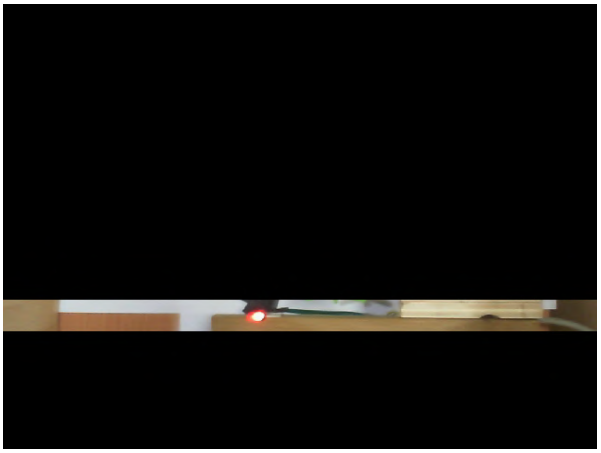
Dataset videí bol vytvorený konkrétne za účelom evaluácie SA so zameraním na motion blur efekt. Pre nájdenie celkového počtu snímkov s úspešne označeným objektom, bolo treba priradiť každému snímku hodnotu (celé okno sledovania), ktorá odpovedá skutočnej pozícií sledovaného objektu na snímke. V podkap. 5.4.1 je teoreticky vysvetlený postup získania referenčného bodu, ktorý bol použitý na vytvorenie referenčného okna. Druhým potrebným oknom pre zistenie úspešnosti sledovania, bolo okno samotného SA. Ich prekrytím bola určená úspešnosť. Podkap. 5.4.2 popisuje toto prekrytie a podmienky, ktoré určujú úspešné označenie sledovaného objektu.

5.4.1. Referenčný bod

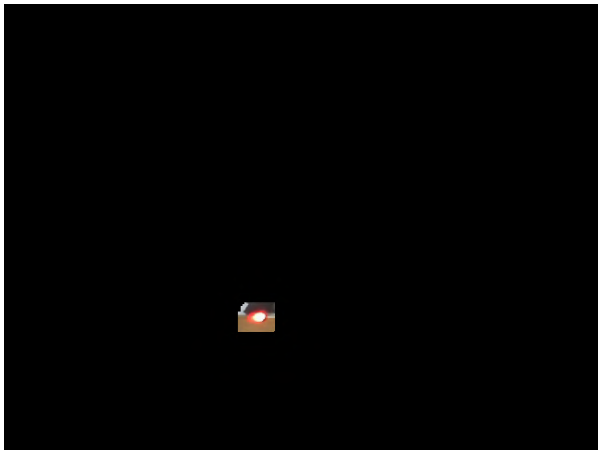
Pre možnosť vyhodnotiť úspešnosť SA, vytvorený dataset potreboval referenčnú hodnotu objektu na snímke každého videa, podľa ktorej sa bude okno SA porovnávať. Na tento účel bol použitý detektor na rozpoznávanie farby (DF).Takýto detektor na nájdenie a sledovanie objektu na základe farby je jeden z najzákladnejších v obore počítačového videnia. Pomocou OpenCV veľmi dobre a jednoducho implementovateľný.



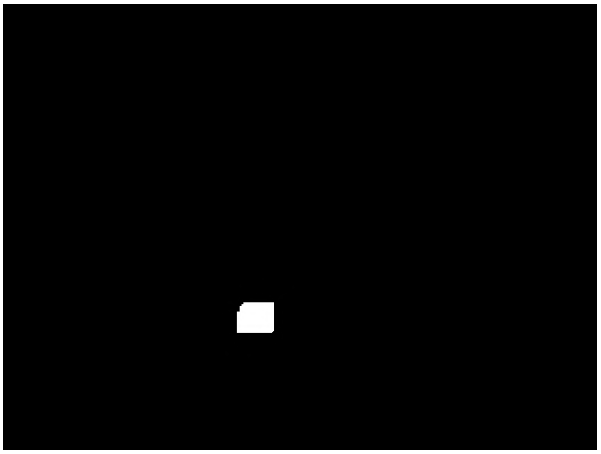
(a) Výsledok spracovania na pôvodnej snímke



(b) Pás hľadania referenčného bodu



(c) Aplikácia filtru



(d) Aplikácia filtru- binárny obraz

Obr. 5.3: DF

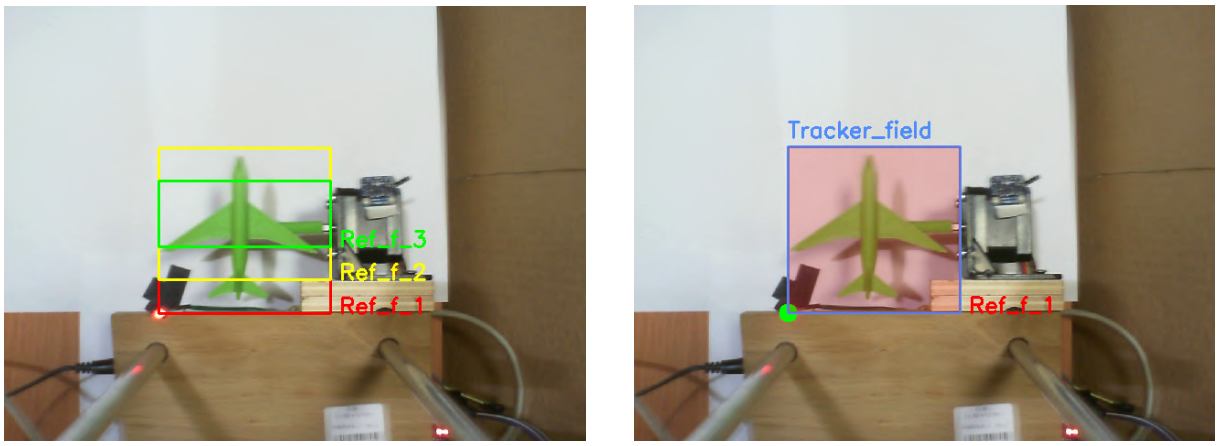
Ako prvé, na začiatku každého snímku bol zmenený použitý farebný model, z BGR na HSV (podkap. 2.7). Následne bol aplikovaný filter, ktorý stlmil na obraze každú farbu okrem zvolenej. Ako výstup z tejto operácie bol vytvorený binárny obraz (maska), ktorý sa mohol aplikovať na pôvodnú snímku (obr. 5.3c). Pre zlepšenie výsledku bola použitá erózia (podkap. 2.3.3) a dilatácia (podkap. 2.3.3) na binárnom obraze. Ako posledný krok boli nájdené kontúry (podkap. 2.3.4) z filtrovaného (binárneho) výsledku. Pri správnom nastavení každej časti bol dosiahnutý požadovaný výsledok: nájdenie konkrétne farby (objektu) na obraze. Správne určenie hodnôt farby (rozsah farby, ktorá bola hľadaná) bolo najdôležitejším nastavením. OpenCV ponúka možnosť vytvorenia trackbaru, ktorý slúžil na nájdenie správnych hodnôt pre filtráciu farby priamo počas prehrávania videa. Rovnakým spôsobom bola nájdená aj správna hodnota opakovania pre eróziu a dilatáciu.

V tejto práci boli použité pre zlepšenie výsledku dva filtre za sebou, ktoré poskytovali nutnú filtráciu v niektorých podmienkach. DF bol nastavený na vyhľadanie červenej farby (referenčný bod). Pre zlepšenie celkovej úspešnosti hľadania, bol pás v ktorom sa referenčný bod pohybuje vybraný pomocou ROI (podkap. 2.3.1) a ako jediný zachovaný z pôvodnej snímky (obr. 5.3b). Všetko ostatné bolo vylúčené z hľadania, priradením každému pixelu mimo vybranú oblasť čiernu farbu. Na obr. 5.3 boli zobrazené niektoré z vykonaných úprav.

Pre jednoduchšie spracovanie videa bola vyskúšaná možnosť, zaznamenať súradnice pri jednom použití DF a použiť ich vo videu s rovnakou hodnotou H. Žiadne video však nebolo nahrané dvakrát tak isto a táto možnosť sa ukázala ako neefektívna (nájdený stred odbiehal od skutočného strede LED).

5.4.2. Prekrytie okien

Na začiatku spustenia spracovania videa SA, boli označené dve ROI z prvého snímku. Prvé predstavovalo označenie LED. Z tohto označenia vznikol pás na obr. 5.3b. Druhé označenie predstavovalo označený sledovaný objekt. Okno takto označeného sledovaného objektu bolo zaznamenané a jeho ľavý dolný bod bol nahradený bodom, ktorý je získaný z DF pri zachovaní rozmerov (šírka, výška) pôvodného označenia. A SA bol inicializovaný s pôvodným označením objektu. Takto bolo vytvorené previazanie medzi referenčným oknom sledovaného objektu a oknom, ktoré riadi SA.



(a) Všetky referenčné okná

(b) Prekrytie referenčného okna a okna SA

Obr. 5.4: Prekrytie okien

Ďalšie referenčné okná

Pri predpoklade, že niektoré SA menili rozmery svojich oknien (šírka, výska) podľa toho ako sa mení uhol sledovaného objektu (napr.: natáčanie, zmena obrysu) boli z pôvodného referenčného okna vytvorené ďalšie dve o niečo menšie. Kvôli možnosti rozlíšiť medzi úspešným sledovaním a nepatrným prekrytím referenčného a sledovacieho okna. Na obr. 5.4a sú ukázané všetky použité referenčné okná.

Výpočet prekrytia a podmienka úspešného označenia

Hodnota prekrytia bola daná vzorcom:

$$X = \frac{Y_{Spol}}{Y_{Cel}} * 100 \tag{5.2}$$

Pomocou bitových operácií (podkap. 2.3.2) bola zistená spoločná oblasť medzi referenčným oknom a oknom SA. Hodnota Y_{Spol} predstavovala obsah tejto spoločne oblasti, Y_{Cel} bola hodnota celkového obsahu jedného z referenčných okien alebo okna SA a X bola

hodnota v percentách, ktorá predstavuje koľko z hodnoty Y_{Cel} bolo vyplnené. Pre každé z referenčných okien tak boli vypočítané 2 hodnoty. Dokopy na jeden snímkom tak pripadalo 6 hodnôt.

Z výsledkov predchádzajúcich meraní a z veľkosti sledovaného objektu (lietadlo) v okne SA bolo zvolené, že ak hodnota X pre referenčné a aj pre okno SA malo hodnotu $\geq$ 60 % bolo sledovanie úspešné. Táto podmienka bola najviac určovaná na hodnotách prvého (najväčšieho) referenčného okna a okna SA, keďže úlohou ďalších dvoch referenčných okien bolo hlavne sledovať správanie SA počas zmeny uhlu natočenia objektu a tak pomôcť k vyhodnoteniu.

5.5. Získané dáta

Kapitola 5.5 je venovaná analýze a následnej vizualizácií nameraných dát, ktoré boli spracované každým sledovacím algoritmom a uložené vo forme excel súborov. Každý súbor obsahuje pôvodné údaje, ktoré boli získane pri vytváraní videa (podkap. 5.1) a všetkých 6 hodnôt, ktoré odpovedajú percentuálnemu prekrytiu referenčných a sledovacích okien pre každé natočenie.

Chyba datasetu

Po prvotnom preskúmaní nameraných dát bola v datasete nájdená chyba, spôsobená hardvérom (podkap. 2.6) a to buď kamerou (2.6.5) alebo krokovým motorom (2.3). Na grafoch bolo v náhodných miestach značné poskočenie z jednej ustálenej hodnoty na hodnotu, ktorá bola o niekoľko desiatok posunutá. Ak jeden zo SA bol výrazne ovplyvnení týmto skokom boli takto porušené dáta vylúčené z následnej analýzy pre všetky SA tak, že z chybného videa boli zobrané iba namerané hodnoty pred chybou. Celková chyba, ktorá by ovplyvnila testovanie bola približne 10 % a z toho pre 43 % poškodených videí bolo nutné odobrať viac ako polovicu dát. Najvýraznejšie ovplyvnila chyba tretiu vzdialenosť a preto aj keď výsledky pre túto vzdialenosť boli vyhodnotené, v porovnané s prvou a druhou vzdialenosťou neboli rovnako presnosné.

5.5.1. Grafy výsledkov

Celkový počet vytvorených excel súborov z merania bolo 700. Pre prvú a druhú vzdialenosť obsahoval každý súbor 8 rýchlostí, pre tretiu vzdialenosť obsahoval 5 rýchlostí. Zrozumiteľne zobrazenie takéhoto veľkého počtu dát v jednom grafe nebolo možné, preto bola analýza rozdelená na vyhodnotenie niekoľkých grafov, ktoré sa zaoberali porovnaním chovania pri zmene určitého parametru/ov z tab. 5.1. Väčšina výsledkov bola založená práve na grafoch, ktoré neboli priamo v práci ukázané, ale bolo možné ich vygenerovať pomocou priloženého skriptu *Spracovanie.py*

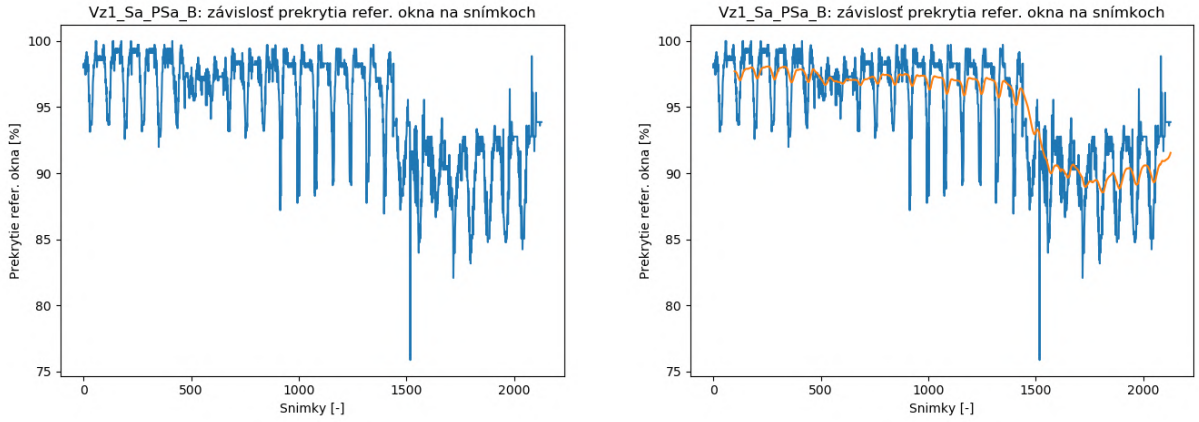
Závislosť prekrytia referenčného okna na snímkoch

Na základných grafoch závislosti prekrytia referenčného/sledovacieho okna na snímkoch bolo ukázané základné správanie SA. Pre lepšie pochopenie tvaru grafu boli pôvodné dáta

filtrované pomocou pohybujúceho sa priemeru (SMA- simple moving average), ktorý je daný rovnicou:

$$SMA = \frac{A_1 + A_2 + .. + A_n}{n}, \tag{5.3}$$

kde A_n bola hodnota za určité časové obdobie a n predstavovalo celkový počet časových hodnôt.

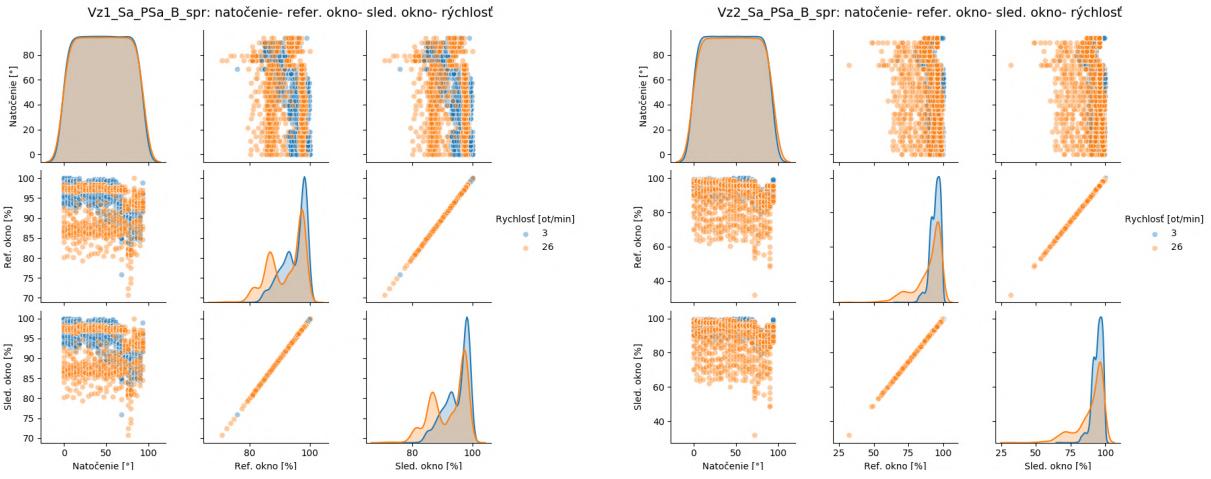


(a) Nefiltrovaný priebeh (b) Filtrovaný priebeh

Obr. 5.5: Závislosť prekrytia na snímkoch

Závislosť natočenia, prekrytia referenčného a sledovacieho okna

Pair plot graf (obr. 5.6) bol použitý na ukázanie závislosti, rozdelenia a korelácie medzi vybranými hodnotami. Parametre použité pre graf boli natočenie sledovaného objektu, prekrytie referenčného a sledovacieho okna. Farba poslúžila na odlíšenie dvoch rýchlosti (bola použitá jedna vzdialenosť od objektu, jeden SA, jedna pozícia svetla a jedna intenzita svetla). Vynesené body boli čiastočne priehľadné čo umožnilo ukázať aj hodnoty, ktoré boli pre dané meranie najčastejšie. Takýto graf bolo možné vytvoriť s každého excel súboru.



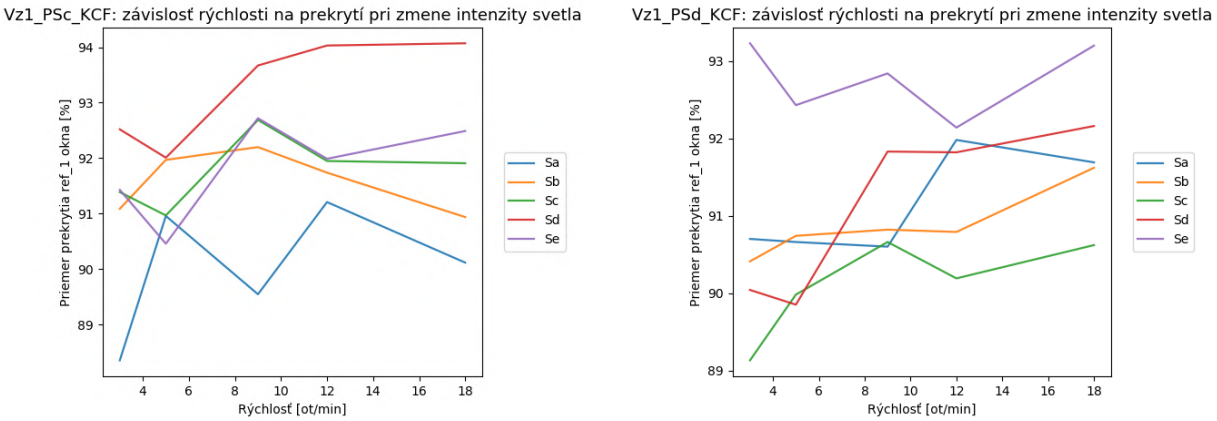
(a) Spracovaný graf Vz1_Sa_PSa_B (b) Spracovaný graf Vz2_Sa_PSa_B

Obr. 5.6: Pair plot graf

Na diagonále sa nachádzali tri grafy, ktoré ukazujú rozdelenie hodnôt jedného parametru. S využitím farebného odlíšenia pre rôzne rýchlosti, grafy ukázali ako dobre medzi sebou vybrané parametre korelujú. Body na grafe obsahujú všetky hodnoty pre daný parameter (podmienka úspešného sledovanie nebola aplikovaná). Pre vynášanie dát, ktoré obsahovali chybu (podkap. 5.5), sa obidve sady hodnôt pre rýchlosti zmenšili tak, aby natočenie sledovaného objektu (pair plot grafe- ľavý horný roh) pre obidve rýchlosti skončilo v rovnakej hodnote.

Závislosť rýchlosti na priemernom prekrytí referenčného okna pri zmene intenzity svetla

Líniový graf (spaghetti plot) zobrazoval závislosť rýchlosti na priemernej hodnote prekrytia referenčného okna pre danú rýchlosť (obr. 5.7). Kedy hodnoty intenzity zobrané z tab. 5.1 boli označené ako Sa, Sb, Sc, Sd, Se.



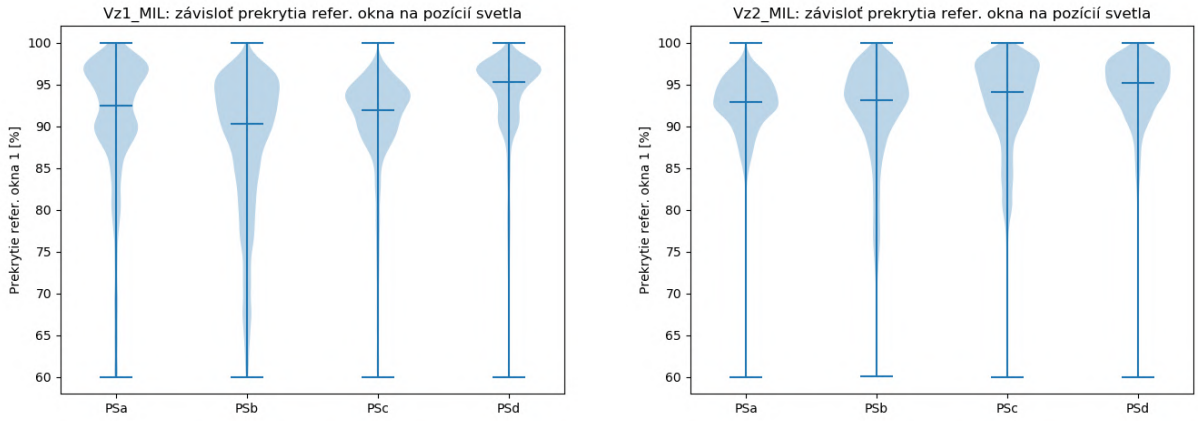
(a) Spracovaný graf Vz1_PSc_KCF (b) Spracovaný graf Vz1_PSd_KCF

Obr. 5.7: Líniový graf

Z grafu vychádzal výsledok ako intenzita svetla vplývala na prekrytie referenčného okna pri zvolenej vzdialenosti, pozície svetla a SA. Priemerná hodnota prekrytia referenčného okna na grafe bola vypočítaná zo všetkých hodnôt danej rýchlosti (podmienka úspešného sledovanie nebola aplikovaná). Ak hodnoty pre intenzitu obsahovali chybu (podkap. 5.5), priebeh bol vykreslený iba po rýchlosť, ktorá nemala žiadne porušené v dáta.

Závislosť prekrytia referenčného okna na pozícií svetla/celková úspešnosť SA pre danú vzdialenosť

Violin grafy (obr. 5.8) ukázali hustotu pravdepodobnosti a celkové rozdelenie použitých dát. Tieto grafy boli použité na porovnanie úspešnosti medzi jednotlivými polohami svetla (na grafe: PSa, PSb, PSc, PSd), kde hodnoty boli zobrané zo všetky intenzít svetla a z nájdených kritických rýchlostí pre danú vzdialenosť H a na celkové porovnanie úspešnosti SA (pri obidvoch bola použitá podmienka úspešnosti ≥ 60 %). Modrá čiara v strede grafu predstavovala medián a horná a dolná noha grafu bola najväčšia a najnižšia hodnota použitých dát. Vďaka grafickému zobrazeniu rozdelenia, bolo možné nájsť hodnoty (na grafe vrcholy) ktoré SA najviac dosahoval.



(a) Spracovaný graf Vz1_MIL

(b) Spracovaný graf Vz2_MIL

Obr. 5.8: Violit graf- porovnanie pozícií svetla

Výsledky pre SA Boosting

SA Boosting dosahoval počas celého testovania veľmi dobré pozitívne výsledky (nad 60 % prekrytia), či už bola zmenená rýchlosť, vzdialenosť, pozícia svetla, intenzita svetla alebo natočenie objektu. V prvej a druhej vzdialenosti boli vidieť najvýraznejšie zmeny pri rýchlosti 26 ot/min, sledovanie však bolo stále veľmi dobré. Pre tretiu vzdialenosť bola najkritickejšia rýchlosť okolo 18 ot/min, kedy bol vidieť značný posun vynesených hodnôt a zväčšený celkový rozsah.

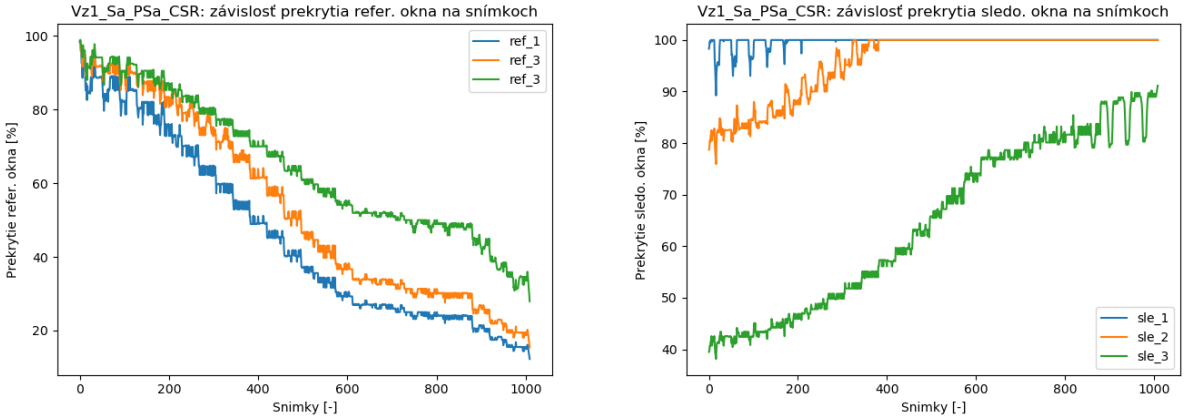
Všetky intenzity svetla sa v prvej vzdialenosti menili v rovnakom rozsahu, v druhej vzdialenosti bola najvýraznejšia zmena pre polohu svetla z vrchu a z boku od 23 ot/min pre všetky intenzity svetla. Pre tretiu vzdialenosť nastal pokles v prekrytí referenčného okna pre všetky intenzity svetla v každej polohe a to okolo rýchlosti 12 ot/min. Pri porovnaní všetkých nameraných dát (rýchlosti a intenzity svetla), samotná poloha svetla ovplyvnila prekrytie referenčného okna iba minimálne pre druhú a tretiu vzdialenosť a v prvej vzdialenosti bola poloha z vrchu horšia oproti ostatným.

Výsledky pre SA CSRT

Všetky úspešné priebehy sledovania pomocou SA CSRT boli veľmi kontinuálne a po filtrácií (podkap. 5.5.1) ich priebeh bol skoro úplne hladký (bez oscilácie). Pre CSRT nebola nájdená kritická rýchlosť v žiadnej zo vzdialeností, pri ktorej by sledovanie spadlo na nízke hodnoty (vrátane rýchlosti 26 ot/min). Mierne horšie hodnoty prekrytie však vznikli pri vyšších hodnotách rýchlostí a nižšej intenzite svetla (intenzita 50 - 100 lux). Zaujímavosťou bolo, že niektoré problémové priebehy pri nižších intenzitách pre druhú vzdialenosť neboli problémové v tretej vzdialenosti. Tento SA počas priebehu menil svoje sledovacie okno na základe zmeny natočenia objektu (zmenšovanie- obr. 5.9a). Pri porovnaní na obr. 5.9 bolo ukázané, že SA mení svoju veľkosť okna a tak stráca celkové prekrytie najväčšieho referenčného okna, ale prekrytie menších sledovacích okien sa zvyšuje pretože. Sledovanie tak bolo zachované.

Intenzita pre prvú vzdialenosť bola vo všetkých prípadoch v rovnakom rozsahu pre všetky rýchlosti. Pre druhú vzdialenosť intenzita b (600 - 700 lux) v pozícií svetla z predu poklesla pre rýchlosť 20 ot/min. Pri vykreslení toho ako pozícia svetla vplývala

na prekrytie referenčného okna bola podmienka úspešnosti zanedbaná kvôli tomu, že SA CSRT dosahoval dobrého sledovania aj keď prekrytie okna bolo nižšie. Do grafu tak boli zahrnuté všetky hodnoty, pre prvú vzdialenosti bol medián pre všetky polohy svetla v rovnakej hodnote, pre druhú a tretiu vzdialenosť bol medián najnižšie pre polohu svetla z boku.



(a) Priebehy referenčných okien

(b) Priebehy okien SA

Obr. 5.9: Ukážka priebehu SA CSRT

Výsledky pre SA KCF

Sledovanie pre SA KCF bolo pre priebeh referenčného aj sledovacieho okna rovnaké. Kritická hodnota vzdialenosti bola v prvej a druhej vzdialenosti medzi rýchlosťami 20 - 26 ot/min. Kedy pri rýchlosti 26 ot/min bol najväčší výskyt neúspešných sledovaní, spôsobených aj veľkým uhol natočenia objektu aj nízkou intenzitou svetla. Pre tretiu vzdialenosť bola hodnota kritickej rýchlosti 12 - 18 ot/min už bolo sledovanie neúspešné.

Na grafoch priemerného prekrytia referenčného okna pri zmene intenzity sa hodnoty u prvej vzdialenosti držali na rovnakej úrovni okrem najnižšej intenzity svetla (50 lux). U druhej a tretej vzdialenosti postupne klesali všetky priebehy. Hodnota mediánu prekrytia na grafe porovnávajúcom pozície svetla v prvej vzdialenosti dosiahla pre všetky pozície hodnotu okolo 92 %, pre druhú a tretiu vzdialenosť bola hodnota z predu o niečo horšia ako ostatné pozície.

Výsledky pre SA MedianFlow

Priebeh prekrytia, ktorý bol získaný po aplikácií SA MedianFlow nebol tak plynulý ako predchádzajúce SA, bol skôr nepravidelný, menej presný a skoro nezávislý na rýchlosti. Už pri rýchlosti 3 ot/min nastala situácia, kedy hodnota prekrytia referenčného okna rástla (do 100 %) a hodnota prekrytia sledovacieho okna klesala, čo značilo, že sledovacieho okna sa zväčšovalo okolo referenčného. Približná kritická hodnota rýchlosti bola pre prvú vzdialenosť 18 - 20 ot/min, pre druhú vzdialenosť 9 - 12 ot/min a pre tretiu vzdialenosť okolo rýchlosti 9 ot/min.

Skoro všetky priebehy intenzít každej polohy svetla postupne klesli na hodnotu okolo 20 %. Hodnota mediánu prekrytia na grafe porovnávajúcom pozície svetla vo všetkých vzdialenosti dosiahla pre všetky pozície hodnotu okolo 95 %, ale postupne pre každú

vzdialenosť sa zväčšil počet pozícií svetla ktoré mali svoju dolnú nohu nad hodnotou 60 % (1. vzd- z predu, 2. vzd- z predu, z boku, 3. vzd- z predu, z vrchu, z boku).

Výsledky pre SA MIL

U SA MIL bol priebeh referenčného a sledovacieho okna skoro identický, nebol však tak hladký ako napríklad SA CSRT, ale skôr viac osciloval medzi blízkymi hodnotami (bolo vidieť aj po filtrácií). Pre prvú vzdialenosť bola kritickou hodnotou rýchlosť 18 ot/min, kedy bolo sledovanie už neúspešné, pre druhú a tretiu vzdialenosť to bola hodnota rýchlosti 9 - 12 ot/min.

Všetky intenzity svetla začali postupne klesať pri rýchlosti 12 - 18 ot/min. Hodnota mediánu prekrytia na grafe porovnávajúcom pozície svetla vo všetkých vzdialenosti dosiahla pre všetky pozície hodnotu medzi 82 - 92 %.

Výsledky pre SA MOSSE

Referenčné aj sledovanie okno SA MOSSE dosiahlo počas sledovania rovnakého priebehu. Pre prvú vzdialenosť predstavovala rýchlosť 23 - 26 ot/min hodnotu, kedy začalo sledovanie mierne klesať a priebeh mal viacej oscilácií. Pre druhú vzdialenosť bola kritickou rýchlosťou 12 - 18 ot/min a pre tretiu vzdialenosť rýchlosť 5 - 9 ot/min.

Priebehy intenzít boli pre všetky polohy svetla každej vzdialenosti klesajúce. Hodnota mediánu prekrytia na grafe porovnávajúcom pozície svetla vo všetkých vzdialenosti dosiahla pre všetky pozície hodnotu približne 82 - 95 %, ale v druhej a tretej vzdialenosti boli pozície svetla, ktoré mali svoju dolnú nohu nad hodnotou 60 % (2. vzd- z vrchu, zo zadu 3. vzd- zo zadu). SA MOSSE pre niektoré náhodné pozície a rýchlosti padol a stratil sledovaný objekt od prvého snímku a nebol schopný ho nájsť na obraze. Táto chyba môže byť spôsobená horšou implementáciou algoritmu do OpenCV alebo neschopnosťou sledovať objekt pri nastavených parametroch.

Výsledky pre SA TLD

Hodnoty prekrytia referenčného a sledovacieho okno získané po aplikácií SA TLD nemali rovnaké priebehy. Ich hodnoty prekrytia značia, že sledovacie okno sa zväčšovalo a zmenšovalo čo spôsobovali skákanie percentuálneho prekrytia. Pre výsledky prvej a druhej vzdialenosti nebolo možné určiť kritickú rýchlosť, kedy sledovanie prestávalo byť spoľahlivé, keďže vynesené hodnoty mali veľký rozsah hodnou (napr.: od 20 do 100 %). Videá spracované SA TLD pre tretiu vzdialenosť dosiahli najlepší výsledok v podobe priebehu prekrytia referenčného a sledovacieho okna zo všetkých vzdialeností. Kritickou rýchlosťou bola hodnota 18 ot/min pre túto vzdialenosť.

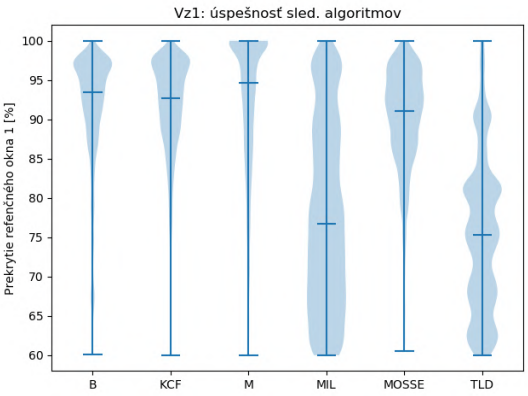
Pre graf pre prvú a druhú vzdialenosť, ktorý zobrazoval intenzity, sa všetky priebehy nachádzali v rovnakom rozsahu. Pre tretiu intenzitu boli grafy príliš ovplyvnené chybou s hárdveru. Hodnota mediánu prekrytia na grafe porovnávajúcom pozície svetla v prvej a druhej vzdialenosti dosiahla pre všetky pozície hodnotu približne 75 %. Pre tretiu pozíciu bol medián pre polohu svetla zo zadu posunutý pod hodnotu 70 %.

Celkové vyhodnotenie úspešnosti SA

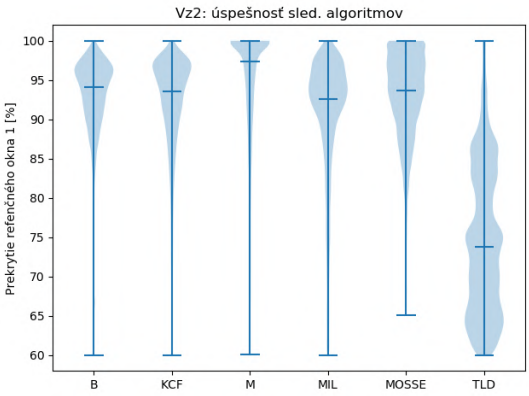
Grafy celkovej úspešnosti SA pre každú vzdialenosť zahrňovali pre daný SA všetky pozície svetla, všetky intenzity svetla, natočenia a bol vybraný taký rozsah pre rýchlosti, ktorý obsahoval kritické rýchlosti pre danú vzdialenosť. Keďže pri použití celého rozsahu rýchlostí by boli grafy mierne skreslené, kvôli množstve pozitívnych alebo negatívnych výsledkov, ktoré boli získané z nižších alebo z vyšších rýchlostí. Pre prvú vzdialenosť bol vybraný rozsah rýchlostí 18 - 23 ot/min, pre druhú vzdialenosť 9 - 18 ot/min a pre tretiu vzdialenosť 5 - 18 ot/min (obr. 5.10). Pre SA CSRT (obr. 5.10d) bola zanedbaná podmienka úspešnosti, keďže SA zmenšoval sledovacie okno podľa toho ako sa menil uhol natočenie sledovaného objektu.

Z obr. 5.10 bolo vidieť, že pre prvú vzdialenosť mal najlepší výsledok SA Boosting, KCF, MedianFlow a CSRT, ktorého medián bol na hodnote 80 %, ale celkové rozloženie bolo najväčšie pre hodnoty 80 - 100 %. Nasledoval SA MOSSE a ako posledné skončili SA MIL A TLD. Pre druhú a tretiu vzdialenosť boli najúspešnejšie SA rovnaké ako pre prvú vzdialenosť. Najhorší opäť skončil SA TLD, ale MIL a MOSSE bol v druhej a tretej vzdialenosti na rovnakej úrovni ako SA Boosting, KCF.

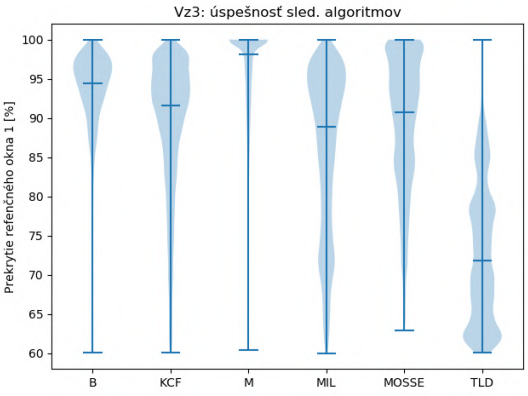
Z celkového testovania pri zadaných podmienkach najúspešnejšie boli SA MedianFlow, Boosting, KCF a CSRT. SA TLD bol v každej vzdialenosti nameraný ako najhorší pri zadaných parametroch.



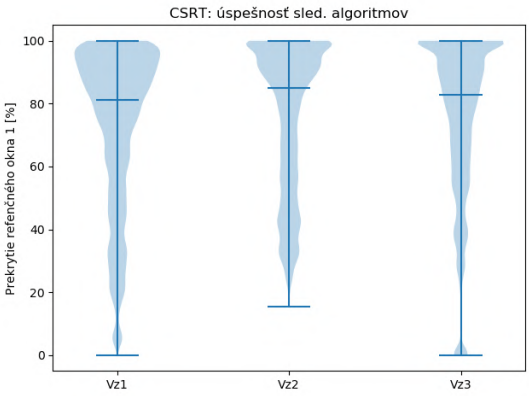
(a) Úspešnosť SA pre prvú vzdialenosť



(b) Úspešnosť SA pre druhú vzdialenosť



(c) Úspešnosť SA pre tretiu vzdialenosť



(d) CSRT: uspešnosť SA

Obr. 5.10: Úspešnosť SA pre každú vzdialenosť

6. Závěr

V práci bolo za úlohou zoznámenie sa z knižnicou OpenCV, vytvoriť sériu pohybujúcich sa obrazov z definovanou rýchlosťou a tu následne analyzovať pomocou SA s ohľadom na motion blur fekt.

Prvý cieľ práce bol splnený v kap. 2, kde boli rozobrané metódy počítačového videnia, ktoré boli použité na vypracovanie tohto zadania. V kapitole 4 bola predstavená myšlienka realizácie testovacieho zariadenia, ktoré poslúžilo na vytvorenie datasetu videí potrebných na evaluáciu. V kapitole boli predstavené predpoklady a potreby, ktoré bolo nutné aby zariadenie spĺňalo. Následne bola vysvetlené ovládanie a samotný návrh testovacieho zariadenia.

Kapitola 5 sa zaoberala vysvetlením priebehu testovania, úspešnosti sledovania a analýze výsledkov. Pri prvotnej analýze dát bola objavená chyba, spôsobená hardvérovou časťou (krokový motor alebo kamera). Táto chyba bolo z datasetu vylúčená aby sa zamedzilo znehodnoteniu celkovej evaluácie. Spracované videá pomocou SA vytvorili obrovské množstvo dát, ktoré nebolo možné zobraziť dostatočne pochopiteľne na jednom grafe. Preto väčšina grafov nebola vložená do práce, ale pomocou skriptu ich bolo možné vygenerovať Na samotné vyhodnotenie bolo zvolených niekoľko grafov, každý zobrazoval závislosť určitých testovaných parametrov a pomohol vyhodnotiť výsledky. Z výsledkov vyplynulo, že parameter, ktorý najviac ovplyvňoval SA bola rýchlosť. Nasledovali poloha svetla a intenzita svetla.

Z celkových výsledkov boli najlepšie SA MedianFlow, Boosting, KCF a CSRT. Z toho KCF a CSRT podľa výsledkov pre samotné algoritmy, vyšli ako z najpresnejším priebehom sledovania. Celkovo sa dá usúdiť, že SA obsiahnuté v knižnici OpenCV boli primerane odolné voči vytvorenému motion blur efektu. Všetky ciele práce sa tak podarilo splniť.

Pre pochopenie a nájdenie ešte presnejších hraníc SA voči motion blur efektu by bolo potrebné vykonať ďalšie testovanie. Pred samotným testovaním by sa bolo treba zamerať na podrobnejší rozbor použitého hardvéru aby sa predišlo chybe v datase, ktorá neúmyselne vznikla v tejto práci. Samotné testovanie by mohlo byť na videách z reálne prostredia (obraz zachytení kamerou na drone), aby boli výsledky čo najviac priblížené realite.

7. Zoznam použitých skratiek a symbolov

px	pixel
mm	milimeter
A	ampér
V	volt
Nm	Newton meter
TZ	Testovacie zariadenie
SA	Sledovacie algoritmy

8. Zoznam príloh

1. Priečinok: Excel_subor
2. skript: Nahravanie_video.py
3. skript: Data_video.py
4. skript: Spracovanie.py

Literatúra

[1] HARNISCHMACHER, Cyril. *The Complete Guide to Macro and Close-Up Photography*. Santa Barbara: Rocky Nook, 2016. ISBN 978-1-68198-052-2.

[2] Shutter speed. *Wikipedia.org* [online]. 2020-05-31 [cit. 2020-06-04]. Dostupné z: https://en.wikipedia.org/wiki/Shutter_speed

[3] Motion Blur. *Shutterstock.com* [online]. c2003-2019 [cit. 2020-06-04]. Dostupné z: https://www.shutterstock.com/support/article/what-is-motion-blur

[4] KEYSER, Celine. *Python Programming Language & Applications*. 2012. ISBN 9788132317586.

[5] The Python Wiki. *Wiki.python.org* [online]. [cit. 2020-06-04]. Dostupné z: wiki.python.org/moin/

[6] The Zen of Python. *Python.org* [online]. c2001-2020 [cit. 2020-06-04]. Dostupné z: https://www.python.org/dev/peps/pep-0020/

[7] RODAS DE PAZ, Alejandro. *Tkinter GUI application development cookbook*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78862-230-1.

[8] Tkinter. *Python.org* [online]. 2020-05-30 [cit. 2020-06-04]. Dostupné z: https://wiki.python.org/moin/TkInter

[9] Numpy: About us. *Numpy.org* [online]. c2019-2020 [cit. 2020-06-04]. Dostupné z: https://numpy.org/about/

[10] MILLER, Curtis. *Hands-On Data Analysis with NumPy and Pandas*. 2. Packt Publishing, 2018 [cit. 2020-06-04]. ISBN 978-1789530797.

[11] Pandas: package overview. *Pandas.pydata.org* [online]. c2008-2014 [cit. 2020-06-04]. Dostupné z: https://pandas.pydata.org/docs/getting_started/overview.html

[12] YIM, Aldrin, Claire CHUNG a Allen YU. *Matplotlib for Python Developers*. 2. Packt Publishing, 2018 [cit. 2020-06-04]. ISBN 978-1788625173.

[13] *Matlpotlib* [online]. c2002-2012 [cit. 2020-06-04]. Dostupné z: https://matplotlib.org/index.html

[14] OpenCV: About. *Opencv.org* [online]. c2020 [cit. 2020-06-04]. Dostupné z: https://opencv.org/about/

[15] OpenCV: modules. *Docs.opencv.org* [online]. [cit. 2020-06-04]. Dostupné z: https://docs.opencv.org/master/index.html

[16] JOSHI, Prateek, David MILLÁN ESCRIVÁ a Vinícius GODOY. *OpenCV By Example*. Packt Publishing, 2016 [cit. 2020-06-04]. ISBN 978-1785280948.

[17] VILLÁN, Alberto Fernández. *Mastering OpenCV 4 with Python*. Packt Publishing, 2019 [cit. 2020-06-04]. ISBN 978-1789344912.

[18] Pixel. *Wikipedia.org* [online]. 2020-05-31 [cit. 2020-06-04]. Dostupné z: https://en.wikipedia.org/wiki/Pixel

[19] DADHICH, Abhinav. *Practical Computer Vision*. 2. Packt Publishing, 2018 [cit. 2020-06-04]. ISBN 978-1788297684.

[20] OpenCV Logo. *Wikimedia.org* [online]. 2020-05-31 [cit. 2020-06-04]. Dostupné z: https://commons.wikimedia.org/wiki/File:OpenCV_Logo_with_text_svg_version.svg

[21] Introduction to OpenCV. *Opencv.org* [online]. 2020-06-04 [cit. 2020-06-04]. Dostupné z: https://docs.opencv.org/trunk/df/d65/tutorial_table_of_content_introduction.html

[22] OpenCV-Python Tutorials. *Opencv.org* [online]. 2020-06-04 [cit. 2020-06-04]. Dostupné z: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

[23] Erosion. *Homepages.inf.ed.ac.uk* [online]. c2003 [cit. 2020-06-04]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm

[24] Dilation. *Homepages.inf.ed.ac.uk* [online]. c2003 [cit. 2020-06-04]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm

[25] MALLICK, Satya. Object Tracking using OpenCV. *Learnopencv.com* [online]. c2020 [cit. 2020-06-04]. Dostupné z: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

[26] GRABNER, Helmut, Michael GRABNER a Horst BISCHOF. *Real-time tracking via on-line boosting*. In: BMVC. 2006. DOI: 10.5244/C.20.6.

[27] BABENKO, Boris, Ming-Hsuan YANG a Serge BELONGIE. *Visual tracking with online multiple instance learning*. In: CVPR. 2009. DOI: 10.1109/CVPR.2009.5206737.

[28] HENRIQUES, João F, Rui CASEIRO, Pedro MARTINS a Jorge BATISTA. *High-Speed Tracking with Kernelized Correlation Filters*. In: IEEE TPAMI. 2014. DOI: 10.1109/TPAMI.2014.2345390.

[29] KALAL, Zdenek, Krystian MIKOLAJCZYK a Jiří MATAS. *Forward-backward error: Automatic detection of tracking failures*. In: Pattern Recognition (ICPR). 2010. DOI: 10.1109/ICPR.2010.675.

[30] LUKEŽIČ, Alan, Tomáš VOJÍŘ, Jiří MATAS a Matej KRISTAN. *Discriminative correlation filter tracker with channel and spatial reliability*. International Journal of Computer Vision. 2018. DOI: 10.1007/s11263-017-1061-3.

[31] BOLME, David S, J. Ross BEVERIDGE, Bruce A DRAPER a Man Lui YUI. *Visual object tracking using adaptive correlation filters*. In: CVPR. 2010. DOI: 10.1109/CVPR.2010.5539960.

[32] KALAL, Zdenek, Krystian MIKOLAJCZYK a Jiří MATAS. *Tracking-learning-detection*. In: IEEE TPAMI. 2012. DOI: 10.1109/TPAMI.2011.239.

[33] HELD, David, Sebastian THRUN a Silvio SAVARESE. *Learning to Track at 100 FPS with Deep Regression Networks*. In: ECCV. 2016. DOI: 10.1007/978-3-319-46448-0_45.

[34] JANKU, Peter, Karel KOPLIK, Tomas DULIK a Istvan SZABO. *Comparison of tracking algorithms implemented in OpenCV*. In: MATEC Web Conf. Volume 76. 2016. DOI: https://doi.org/10.1051/matecconf/20167604031

[35] LEHTOLA, Ville, Heikki HUTTUNEN, Francois CHRISTOPHE a Tommi MIKKONEN. *Evaluation of Visual Tracking Algorithms for Embedded Devices*. In: SCIA. 2017. DOI: 10.1007/978-3-319-59126-1_8.

[36] RYDLO, Pavel. *KROKOVÉ MOTORY A JEJICH ŘÍZENÍ: Studijní texty* [online]. Liberec, 2000 [cit. 2020-05-31]. Dostupné z: http://cis.wz.cz/elz/krok2.pdf

[37] WOODFORD, Chris. Stepper motors. *Explainthatstuff.com* [online]. [cit. 2020-06-04]. Dostupné z: https://www.explainthatstuff.com/how-stepper-motors-work.html

[38] HRABOVCOVÁ, Valéria, Ladislav JANOUŠEK, Pavol RAFAJDUS a Miroslav LIČKO. *Moderné elektrické stroje*. Žilina: EDIS, 2001. ISBN 80-7100-809-5

[39] Krokový motor. *Monolithicpower.com* [online]. [cit. 2020-06-04]. Dostupné z: https://media.monolithicpower.com/wysiwyg/3_11.png

[40] Camera. *Media.4rgos.it* [online]. [cit. 2020-06-04]. Dostupné z: https://media.4rgos.it/i/Argos/8365534_R_Z001A?w=750&h=440&qlt=70

[41] Krokový motor Nema 17. *Distrelec.sk* [online]. [cit. 2020-06-04]. Dostupné z: https://www.distrelec.sk/Web/WebShopImages/landscape_large/0-/01/30097220-01.jpg

[42] Arduino introduction. *Arduino.cc* [online]. c2020 [cit. 2020-06-04]. Dostupné z: https://www.arduino.cc/en/guide/introduction

[43] The Pxhere. *Pxhere.com* [online]. [cit. 2020-06-04]. Dostupné z: https://pxhere.com/en/photo/1158161

Zoznam obrázkov

Zoznam tabuliek