VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Michal Komloši

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

TRASOVÁNÍ POHYBUJÍCÍHO SE OBJEKTU V OBRAZOVÉ SCÉNĚ
TRACKING OF MOVING OBJECT IN VIDEO

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Michal Komloši

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jiří Přinosil, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Michal Komloši ***ID:*** 164307
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Trasování pohybujícího se objektu v obrazové scéně

POKYNY PRO VYPRACOVÁNÍ:

Proveďte analýzu stávajících metod pro sledování trajektorie pohybujícího se objektu v obrazové scéně. Na základě získaných znalostí navrhněte algoritmus pro sledování pohybu různých objektů, který bude vhodně kombinovat vybrané metody. Algoritmus dostane na vstupu informaci o počáteční pozici daného objektu v obraze, a poté již bude zcela autonomně zaznamenávat jeho pohyb napříč obrazovou scénou. Algoritmus by si měl poradit s dočasným překrytím nebo zmizením sledovaného objektu ze scény. Navržený algoritmus implementujte ve vybraném programovacím jazyce a ověřte jeho spolehlivost nad záznamy z reálného prostředí.

DOPORUČENÁ LITERATURA:

[1] GU, Irene a Zulfiqar KHAN. Online Learning and Robust Visual Tracking using Local Features and Global Appearances of Video Objects. Object Tracking. InTech, 2011, 89-118.

[2] SUGANDI, Budi, Hyoungseop KIM, Joo Kooi TAN a Seiji ISHIKAWA. Object Tracking Based on Color Information Employing Particle Filter Algorithm. Object Tracking. InTech, 2011, 69-88.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: Ing. Jiří Přinosil, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto diplomová práca sa zaoberá možnosťami trasovania pohybujúceho sa objektu v obraze. Výsledkom práce je navrhnutý a v programovacom jazyku C# implementovaný algoritmus, ktorý vylepšuje funkciu existujúceho algoritmu.

KĽÚČOVÉ SLOVÁ

Trasovanie, Video, Algoritmus, Detekcia

ABSTRACT

This master thesis deals with tracking the moving object in image. The result of the thesis is designed algorithm which is implemented in the programming language C#. This algorithm improves the functionallity of an existing tracking algorithm.

KEYWORDS

Tracking, Video, Algorithm, Detection

KOMLOŠI, Michal. *Trasování pohybujícího se objektu v obrazové scéně*. Brno, 2019, 49 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Jiří Přínosil, Ph.D.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovoú prácu na tému „Trasování pohybujícího se objektu v obrazové scéně“ vypracoval(a) samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce, ktorým je pán Ing. Jiří Přínosil, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Počítačová vizuálna detekcia a rozpoznávanie objektov v obraze je v poslednej dekáde jednou z najväčších výziev v oblasti informatiky. Ide o snahu, aby počítače boli schopné rozpoznať sledovanú scénu podobne, ako to dokáže ľudský mozog. Jej využitie je veľmi široké, či už v bezpečnostných systémoch (identifikácia oprávnených osôb a pod.), v priemysle alebo v medicínskej oblasti (detekcia tumorov v tele pacienta).Najmä v bezpečnostných systémoch je potom okrem detekcie, dôležité aj sledovanie trasy pohybu objektu a to napríklad v prípade potreby sledovania pohybu áut na diaľnici.

Táto diplomová práca sa venuje teoretickému popisu a následne porovnaniu niektorých vybraných algoritmov pre trasovanie pohybujúceho sa objektu v obraze.

Hodnotenie úspešnosti algoritmov prebieha porovnaním výsledných nameraných súradníc polohy sledovaného objektu, so súradnicami jeho skutočnej pozície.

Výsledkom práce je návrh algoritmu, ktorý sa snaží o zlepšenie vlastností jedného z testovaných algoritmov, a aplikácia, ktorá sleduje vyznačený objekt v obraze, pri čom využíva daný optimalizovaný algoritmus.

Samotná aplikácia je napísaná v programovacom jazyku C# a využíva knižnicu OpenCv resp. EmguCv. Obsahuje grafické užívateľské rozhranie, pomocou ktorého je možné zvoliť a prehrať ľubovolné video, v ktorom bude následne užívateľom vybraný objekt na sledovanie. Výber sledovaného objektu je možné urobiť pomocou kurzoru myši.

1 DETEKCIA OBJEKTOV V OBRAZE

1.1 Eliminácia pozadia

Eliminácia pozadia sa veľkou mierou podieľa na celkovej úspešnosti trasovacieho algoritmu. Pre správny návrh modelu pozadia je nutné myslieť na dynamicky sa meniace vlastnosti obrazu, ako je napríklad zmena osvetlenia scény a pod. Je teda potrebné vytvoriť model pozadia, ktorý sa bude vedieť adaptovať na rôzne zmeny parametrov obrazu.
V podkapitolách 1.1.1 až 1.1.4 sú vybrané a zjednodušene popísane niektoré algoritmy na elimináciu pozadia. [9]

1.1.1 Metóda Frame Difference

Jedná sa o najjednoduchšiu metódu eliminácie pozadia. Zakladá sa na odpočítaní aktuálneho snímku, od snímku, ktorý bol definovaný ako pozadie. Väčšinou sa ako snímok pozadia použije prvý snímok spracovávaného videa. Výsledný rozdiel pixelov snímkou T_Δ je potom porovnávaný s prahovou hodnotou T_s. Ak platí že $T_\Delta > T_s$, pixel patrí do popredia.
Keďže pri využívaní tejto metódy nedochádza k zmenám v modeli pozadia, nie je príliš vhodná pre použitie v reálnom čase. Jej výhodou je však nízka výpočtová náročnosť a teda rýchlosť. [2]

1.1.2 Metóda Approximate Median

Metóda Approximate Median sa zakladá na ukladaní predchádzajúcich snímkov, ktorých medián určuje model pozadia. Ako pozadie je teda definovaný pixel, ktorý sa nachádza v strede usporiadanej rady pixelov. Snímka popredia je potom získaná obdobne ako v predchádzajúcej metóde a teda odčítaním aktuálneho snímku od snímku pozadia. Súčasne dochádza k aproximácií mediánu zo vzájomného porovnania pozadia a aktuálneho snímku. Hodnota jasu pixelu na pozadí je zvýšená o jedna, ak je hodnota jasu pixelu aktuálneho snímku vyššia. Naopak, ak je táto hodnota nižšia, hodnota jasu na pozadí je znížená.
Táto metóda je pre použitie v reálnom čase vhodnejšia ako metóda Frame Difference, kedže model pozadia sa dynamicky mení v čase. Avšak jej nevýhodou je pamäťová náročnosť, pretože predchádzajúce snímky je potrebné ukladať. [2]

1.1.3 Metóda Mixture of Gaussians (MOG)

MOG využíva Gaussové rozdelenie pravdepodobnosti. Pre každý pixel všetkých povrchov, je určená rada funkcií hustoty pravdepodobnosti. Pre prípad jednokanálového obrazu v stupňoch šedi, vyzerá pravdepodobnostná funkcia nasledovne:

$$f(x) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{\frac{(x-\mu_n)^2}{2\sigma_n^2}} \tag{1.1}$$

x – je práve spracovávaný pixel snímku
μ_n – označuje strednú hodnotu $E(X)$ n-tého pozadia
σ_n – smerodajná odchýlka
σ_n^2 – rozptyl pravdepodobnosti $D(X)$ n-tého povrchu

Potom povrch ktorého krivka pravdepodobnosti je najvyššia predstavuje pozadie a krivka s nižšou hodnotou pravdepodobnosti môže definovať sledovaný objekt.
Táto metóda sa uplatňuje hlavne tam, kde dochádza k určitým periodickým zmenám na pozadí. Je taktiež vhodná pre použitie v reálnom čase vďaka priebežným aktualizáciám modelu pozadia. [2]

1.1.4 Samoorganizačné neurónové siete

Samoorganizačné neurónové siete fungujú na princípe automatického učenia bez učiteľa. Siete na základe určitých vzťahov reagujú na dáta na vstupe správnou hodnotou na výstupe.
Tieto siete sú koncipované ako dvojrozmerné usporiadanie neurónov, kde je každému pixelu priradená skupina $n * n$ neurónov. Pre každý uzol danej siete je vypočítaná funkcia vážených lineárnych kombinácií, ktoré sú na vstupe siete. Každý uzol možno opísať pomocou váhového vektora a kombinácia všetkých váhových vektorov potom definuje model pozadia. Počiatočný model pozadia je potom stanovený z prvého snímku vo videu, kde hodnoty jednotlivých váhových vektorov v $n * n$ maticiach korešpondujú s hodnotami príslušného pixelu v snímke.
Keď sa pixely ďalšieho snímku dostanú na vstup siete, dôjde k súťaži na základe euklidovských vzdialeností medzi vstupom a neurónmi siete. Neurón, ktorý má túto vzdialenosť najmenšiu, sa stáva víťazom. Súčasne je vytvorená snímka vzdialeností D^t. Postupným porovnávaním s hodnotou prahu ϵ je získaná maska popredia. Model pozadia je vzápätí určený aktualizáciou váhových vektorov určených z predchádzajúcej snímky.

Táto metóda je z dôvodu výpočtovej náročnosti vhodná na použitie v reálnom čase, len vo videu do určitého rozlíšenia. [14]

1.2 Región

Región môžeme definovať ako ohraničené okolie sledovaného objektu, získané predchádzajúcim spracovaním. Región býva označený spravidla štvoruholníkom a reprezentovaný jeho súradnicami, čo umožňuje nižšiu výpočtovú náročnosť. Takéto označenie objektu však presne nekopíruje jeho tvar, čo môže byť nežiadúce v určitých typoch aplikácií.

1.3 Model

Pod pojmom model môžeme v tomto kontexte chápať objekt, ktorý bol vytvorený na základe predchádzajúcich vedomostí o objektoch, ktoré sa vyskytujú v obraze. Modely sú najčastejšie vytvárané manuálne.
Pre sledovanie pohybu osôb sa používa analýza syntézou. Najprv je z predchádzajúceho pohybu predikovaná základná poloha postavy pre nasledujúcu snímku. Následne je predikovaný model syntetizovaný a porovnaný so skutočnými dátami, čím je určená podobnosť medzi týmito dátami. Tohoto výsledku sa dá v závislosti na použitom algoritme dosiahnuť buď rekurzívne, alebo pomocou vzorkovacích techník, až kým nie je nájdená správna poloha postavy a model môže byť aktualizovaný. Definíciu modelu pre prvý záber je nutné riešiť individuálne.

1.4 Príznaky

V algoritmoch používajúcich príznaky, sú z obrazu vyberané segmenty, ktoré su zlúčené do vysokoúrovňových príznakov a príznaky sú potom priraďované medzi snímky. Príznakové algoritmy je možné rozdeliť do týchto kategórií:

- lokálne (segmenty čiar, kriviek...)
- globálne (plocha, farba...)
- závislostné (napr. geometrické vzťahy medzi príznakmi)

Algoritmy týchto kategórií môžu byť medzi sebou vhodne kombinované v záujme zlepšenia výsledku.
Príznakové algoritmy je možné používať v reálnom čase vďaka ich dobre adaptácií v dvojrozmernom obraze. Avšak úspešnosť detekcie založenej na príznakoch je pomerne nízka. [3]

2 PREDIKCIA POHYBU A TRASOVANIE DETEKOVANÝCH OBJEKTOV V OBRAZE

2.1 Algoritmus BOOSTING

Tento sledovač je založený na online verzií algoritmu AdaBoost, ktorý využíva interne HAAR kaskádu. Tento klasifikátor sa musí učiť za behu na základe pozitívnych a negatívnych príkladoch objektu.
Počiatočný ohraničujúci región predaný užívateľom alebo detekčným algoritmom sa považuje za pozitívny príklad sledovaného objektu. Ostatné pixely, mimo označenú oblasť, sú považované za pozadie. Pri ďalšom snímku klasifikátor vypočíta skóre pre všetky pixely v blízkosti predchádzajúceho umiestnenia objektu. Nové umiestnenie objektu je miesto, kde je skóre maximálne. Takto vzniká ďalší pozitívny príklad objektu pre klasifikátor a ten sa následne aktualizuje.[12]
Algoritmus BOOSTING je pomerne zastaralý a jeho výkonnosť je v porovnaní s novšími sledovačmi iba podpriemerná.[11]

2.2 Algoritmus MIL

Princíp tohoto sledovača je podobný, ako u vyššie opisovaného algoritmu BOOSTING.
Najväčší rozdiel je v tom, čo sledovač považuje za ďalší pozitívny príklad pre učenie. Kým algoritmus BOOSTING berie ako pozitívny príklad len momentálnu polohu objektu, MIL hľadá ďalšie pozitívne príklady aj v úzkom susedstve tejto polohy. Takýto prístup sa na prvý pohľad nemusí javiť ako správny, keďže na niektorých pozitívnych príkladoch nemusí byť sledovaný objekt presne v strede vybranej oblasti. MIL tracker však nerozdeľuje príklady na pozitívne a negatívne ale rozdeľuje ich do pozitívnych a negatívnych „vreciek“. Nie všetky príklady v pozitívnom vrecku sú potom naozaj pozitívne. Stačí ak je pozitívny iba jeden z nich. Ak sledovač MIL neoznačil polohu objektu presne, je stále veľmi veľká šanca že v pozitívnom vrecku sa nachádza aj presný obrázok na základe ktorého sa klasifikátor aktualizuje.[12]
Výkonnosť tohoto sledovača je dobrá a jeho chybovosť je menšia ako u BOOSTING sledovača. Nevýhodou je že zlyhanie sledovania nie je spoľahlivo hlásené užívateľovi.[11]

2.3 Algoritmus KCF

Sledovač KCF (Kernelized Correlation Filters), stavia na myšlienkach predchádzajúcich dvoch algoritmov. Tento sledovač využíva fakt, že niekoľko pozitívnych príkladov, ktoré používa algoritmus MIL, má veľké navzájom sa prekrývajúce plochy. Tieto dáta vedú k určitým matematickým vlastnostiam, ktoré potom KCF sledovač využíva tak, aby bolo samotné sledovanie rýchlejšie a presnejšie.
Použitie tohoto sledovača je odporúčané pre väčšinu aplikácií. Jeho nevýhodou je že sa nedokáže uzdraviť z úplnej oklúzie.[12] [11]

2.4 Algoritmus TLD

TLD - Tracking, learning and detection. Ako už samotný názov napovedá, tento sledovač rozdeľuje proces sledovania do troch úloh:

1. sledovanie
2. učenie
3. detekciu

Algoritmus sleduje objekt snímok za snímkom a detektor lokalizuje všetky výstupy, ktoré boli doposiaľ sledované a v prípade potreby sledovač opraví. Z chýb sa detektor aktualizuje (učí) aby sa podobným chybám v budúcnosti vyvaroval.[12]
Výsledkom je však to, že označenie regiónu, teda výstup sledovača, má tendenciu trochu skákať. Napríklad v scéne kde algoritmus sleduje chodca, má tento sledovač tendenciu preskočiť na iného chodca ak sa tento v scéne nachádza tiež.
Výhodou TLD sledovača je, že najlepšie sleduje zmeny veľkosti sledovaného objektu v obraze.[11]

2.5 Algoritmus MEDIANFLOW

Tento sledovač sleduje trajektóriu objektu dopredu aj dozadu a meria rozdiel medzi týmito trajektóriami, čo mu umožňuje spoľahlivo určiť skutočný smer pohybu objektu v sekvencií obrázkov, ale aj detektovať zlyhanie.
Sledovač funguje najlepšie v prípade, ak je pohyb malý a predvídateľný.
Jednoznačne najväčšou výhodou je spoľahlivá detekcia zlyhania algoritmu, ktorá je užívateľovi hlásená, na rozdiel od ostatných algoritmov, ktoré pokračujú aj keď sledovanie zlyhalo.[12] [11]

2.6 Kalmanov filter

Umožňuje predpovedať polohu a upresniť ju na základe súčastného merania. Bol vyvinutý vedcom Rudolfom Kalmanom, pre potreby filtrácie šumu z elektrických signálovm v rokoch 1960-1961. Svoje uplatnenie však našiel v širokom spektre odvetví (letectvo - autopilot).
Celý princíp Kalmanovho algoritmu je možné vysvetliť tak, že ide o spriemerovanie nepresností nameraných hodnôt od odhadovaných hodnôt, teda pravdepodobnosti, že nameraná hodnota je správna. T.j. k odhadu najbližších nasledujúcich polôh objektu, dochádza na základe priebehu predchádzajúcich chýb, resp. na základe ich rozptylu.
Predikcia nasledujúceho stavu prebieha podľa vzťahu:

$$x_k^- = Ax_{k-1} + \omega_{k-1} \tag{2.1}$$

Odhad odchylky podľa vzťahu:

$$P_k^- = AP_{k-1}A^T + Q \tag{2.2}$$

$\boldsymbol{x_k^-}$ – predikovaný stav filtra pre súčasný stav $\boldsymbol{k}$
$\boldsymbol{P_k^-}$ – odhad chyby pre momentálny stav
$\boldsymbol{A}$ – $\boldsymbol{n \times n}$ prechodová matica označujúca vzťah medzi predošlým a súčasným stavom systému
$\boldsymbol{\omega_{k-1}}$ – hodnota procesného šumu
$\boldsymbol{Q}$ – kovariačná matica procesného šumu

2.7 Sledovanie verzus detekcia

Sledovacie algoritmy sú väčšinou rýchlejšie ako detekčné. Dôvod je celkom jednoduchý. Ak už bol objekt v scéne detekovaný,sú známe informácie o jeho polohe v predchádzajúcom snímku, smere jeho pohybu a jeho rýchlosti. Dobrý sledovací algoritmus tieto nazhromaždené dáta využije, zatiaľ čo detekčný algoritmus začína vždy prakticky on nuly.
Účinné sledovacie systémy často výhody oboch prístupov kombinujú a pri použití sledovacieho algoritmu, vykonávajú detekciu na každom n-tom snímku.

3 NÁSTROJE POUŽITÉ NA POROVNANIE SLEDOVACÍCH ALGORITMOV

3.1 Microsoft Visual Studio 2017

Visual Studio je najpopulárnejším vývojovým prostredím pre aplikácie, spustiteľné na operačných systémoch Windows, od firmy Miscrosoft. Toto vývojové prostredie ponúka širokú podporu programovacích jazykov a technológií platformy „.NET“. Používa sa pre vývoj všetkých druhov aplikácií, od konzolových cez desktopové až po webové aplikácie a služby. Editor kódu vo Visual Studiu podporuje tzv. IntelliSense, ktorý je schopný detekovať chyby a samostatne navrhnúť opravu chýb v kóde, čo veľmi uľahčuje prácacu programátorovi. Testovanie aplikácií za behu, zase ponúka vstavaný debugger, ktorý pracuje ako na úrovni stroja tak na úrovni kódu. [15]

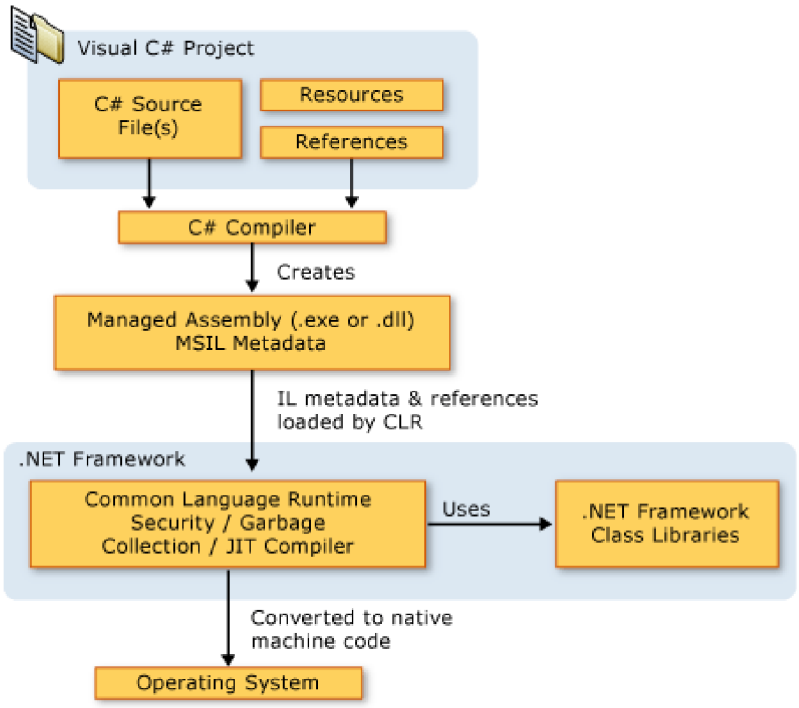
3.2 Programovací jazyck C#

Jedná sa o elegantný a typovo bezpečný objektovo orientovaný programovací jazyk, pomocou ktorého je možné na platforme .Net vyvárať rôzne typy aplikácií, od konzolových až po webové. Syntax jazyka je pomerne jednoduchá a ľahko osvojiteľná pre každého, kto už niekedy programoval v jazykoch ako JAVA alebo C++.

Jazyk C# zjednodušuje veľa zložitostí jazyka C++ a to ako po stránke syntaxe, tak aj napríklad pri práci s pamäťou. Pre uvoľňovanie pamäti nie sú potrebné destruktory. Tento proces je riešený automaticky pomocou tzv. garbage collector-u.
Zdrojový kód je kompilovaný do tzv. intermediate language (IL). Tento skompilovaný kód je potom spolu s ďalšími zdrojmi (napr. bitové mapy), uložený na disku v spustiteľnom súbore, typicky s koncovkou .exe alebo .dll, ktorý nazývame „assembly“. [8]
Priebeh vytvorenia spustiteľného súboru môžeme vidieť na obrázku 3.1

3.3 OpenCv

OpenCV je knižnica zameraná hlavne na prácu s obrazom v reálnom čase a na počítačové videnie. Knižnica je šíriteľná zadarmo ako pre komerčné, tak pre akademické účely. Priamo použiteľná je v jazyku C/C++, s generátorom rozhrania SWIG tiež v jazykom Python, Java alebo Octave. [10] Pre iné programovacie jazyky (C#, Perl...) existujú rôzne wrappre, cez ktoré je možné v kóde volať funkcionalitu OpenCv.



Obr. 3.1: Priebeh kompilácie jazyka c# a jeho vzťah s .net Frameworkom (obr. prevzatý od [8])

3.3.1 EmguCv

EmguCv je wrapper knižnice OpenCv ktorý sprostredkúva metódy a objekty tejto knižnice pre prostredie .NET, teda pre jazyky ako je C#, Visual Basic, Visual C++, IronPython a pod. [4]

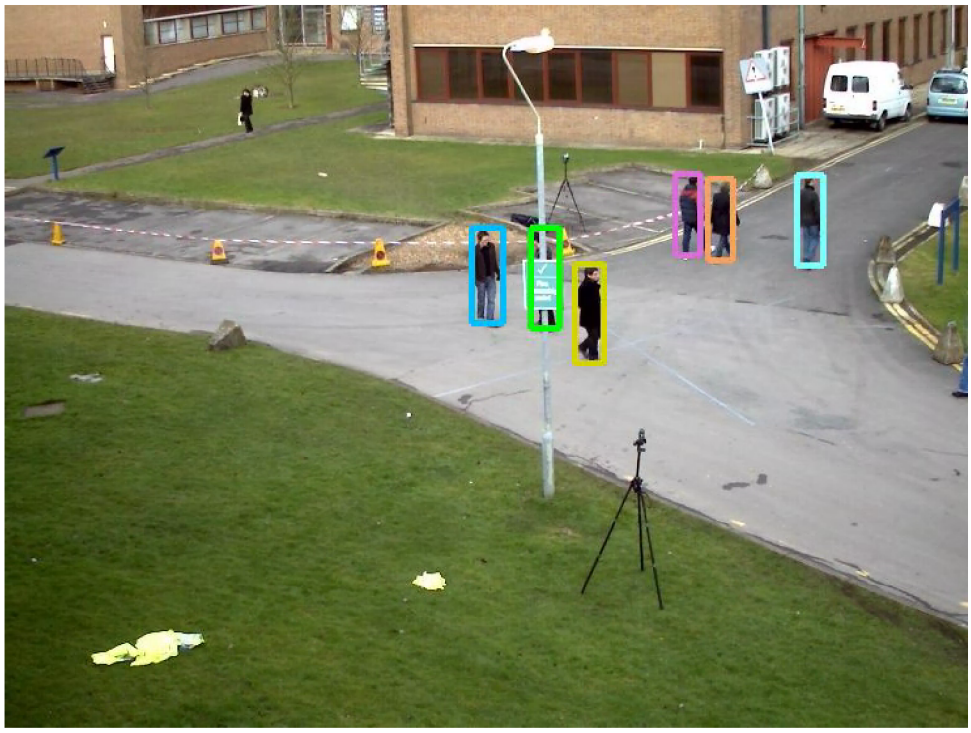
3.4 Accord Net

Accord Net Framework je volne dostupná knižnica algoritmov pre platformu .NET. Obsahuje metódy určené na prácu s obrazom alebo zvukom, a jeho spracovávanie. Knižnica je napísaná v jazyku C# a jej obsah je veľmi dobre zdokumentovaný spolu s užitočnými príkladmi. [6], [1]
Okrem iného ponúka funkcionalitu Kalmanového filtru, ktorá bola použitá pre riešenie tejto diplomovej práce

4 POROVNANIE ÚSPEŠNOSTI VYBRANÝCH ALGORITMOV

4.1 Skutočná pozícia

Skutočná pozícia (ang. Ground truth) je termín používaný v štatistike a strojovom učení, ktorý predstavuje niečo, ako kontrolu výsledkov a presnosti strojového učenia oproti skutočnému svetu. [13]
Tento termím bol prevzatý z metrológie, kde sa ako groud truth označujú informácie získané priamo na určitom mieste.
Pri testovaní úspešnosti jednotlivých sledovacích algoritmov bolo použité voľne dostupné video, stiahnuteľné z internetovej stránky `https://motchallenge.net/vis/PETS09-S2L1`. Stránka motchallenge sa zaoberá spravodlivým hodnotením algoritmov sledovania objektov v obraze.
K použitému videu s názvom „PETS09-S2L1“, je na tejto stránke dostupná skutočná pozícia, ako vo vizuálnej, tak aj v textovej podobe.



Obr. 4.1: Video reprezentujúce skutočnú pozíciu

Ako je vidieť na obrázku 4.1, vo vizuálnej podobe skutočnú pozíciu predstavuje jedinečné vyznačenie každého sledovaného objektu (v tomto prípade osoby), po celú dobu jeho prítomnosti v obraze.

Tak ako bolo spomenuté vyššie, skutočná pozícia pre toto testovacie video je dostupné aj v textovej forme, kde poskytuje ešte presnejšie informácie.

```
1,9,499,158,31.03,75.17,1,-4.1554,-7.3591,0
1,15,258,219,32.913,88.702,1,-11.306,-5.5995,0
1,19,633,242,42.338,81.074,1,-9.0323,-12.587,0
2,9,497,158,31.03,75.17,1,-4.1744,-7.3156,0
2,15,263,218,32.774,88.505,1,-11.21,-5.6516,0
2,19,627,242,42.16,81.387,1,-9.0791,-12.489,0
3,9,495,159,31.03,75.17,1,-4.2787,-7.314,0
3,15,268,216,32.605,88.263,1,-11.06,-5.6643,0
3,19,619,242,41.938,81.766,1,-9.1492,-12.342,0
4,9,492,160,31.03,75.17,1,-4.3919,-7.2909,0
4,15,275,215,32.419,87.999,1,-10.891,-5.7142,0
4,19,610,242,41.683,82.175,1,-9.2726,-12.214,0
5,9,489,161,31.03,75.17,1,-4.5045,-7.2679,0
5,15,281,214,32.227,87.725,1,-10.784,-5.7856,0
5,19,601,242,41.39,82.581,1,-9.3428,-12.067,0
```

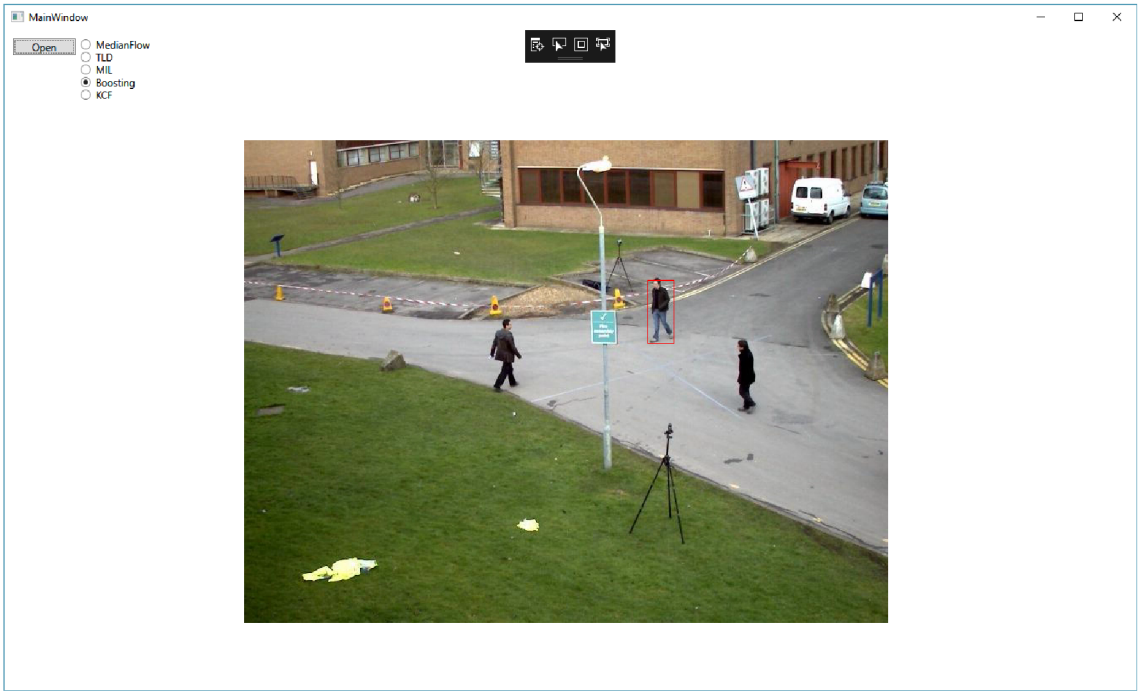
Obr. 4.2: Ukážka skutočnej pozície v textovej forme

Dôležitých je najmä prvých šesť čísel, ktoré sú oddelené čiarkou. Prvé číslo označuje poradové číslo snímky videa. Snímky sú číslované je od **1** až po $\boldsymbol{n}$, kde $\boldsymbol{n}$ predstavuje celkový počet snímkov videa. Druhé číslo predstavuje identifikáciu sledovaného objektu (jeho jedinečné ID). Ďalšie dve čísla, sú súradnice X a Y regiónu (štvoruholníka), ktorým je objekt počas sledovania označený. Ďalej nasleduje číslo, ktoré definuje šírku tohoto štvoruholníka, a šieste číslo vyjadruje jeho výšku.

Ak sa teda pozrieme na obrázok 4.2, kde sa nachádza ukážka groud truth v textovej forme, môžeme sa z prvého riadku dozvedieť, že na prvej snímke sa objekt s ID = 9, nachádza v regióne so súradnicami X = 499, Y = 158 a veľkosťou 31,03 x 75,17 pixelu.
Úlohou hodnoteného sledovacieho algoritmu je dosiahnuť výsledky, ktoré budú čo možno najviac odpoveď skutočne pozícií.

4.2 Aplikácia pre porovnanie algoritmov

Pre účel testovania úspešnosti sledovacích algoritmov, bola v rámci vypracovania zadania tejto práce vytvorená jednoduchá aplikáciu v jazyku C#. Ide o aplikáciu typu WPF, pomocou ktorej je možné prehrávať video a sledovať pohyb určitého objektu pomocou vybraného trackeru z knižnice OpenCv, resp. EmguCv.



Obr. 4.3: Prostredie aplikácie pre testovanie sledovacích algoritmov

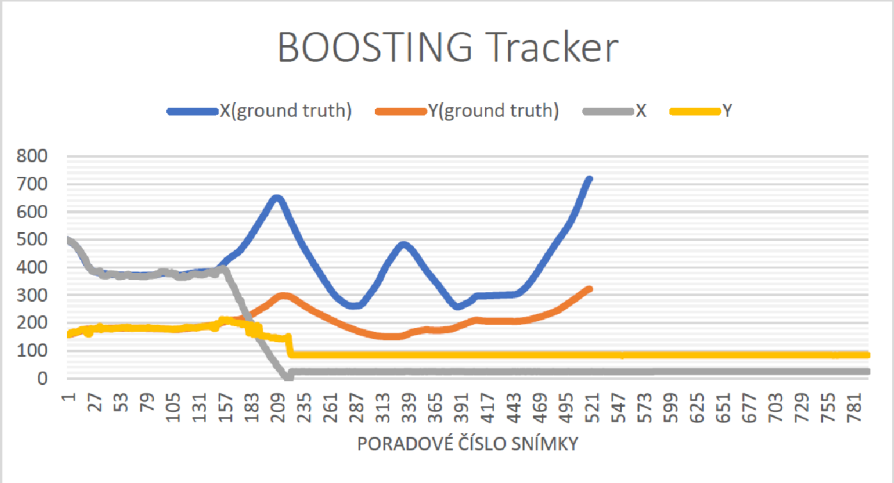
Prostredie testovacej aplikácie je veľmi jednoduché. V ľavom hornom rohu sa nachádza tlačítko „Open“ pomocou ktorého, užívateľ vyberie testovacie video. Ďalej je hneď na pravo od tlačítka možnosť vybrať testovaný algoritmus. Video je následne prehrané približne v prostriedku užívateľského rozhrania (viď obrázok 4.3).
Priamo v zdrojovom kóde aplikácie je zadaná počiatočná pozícia sledovaného objektu, ktorá je prevzatá zo skutočnej pozície a v obraze vyznačená štvoruholníkom červenej farby. Pozícia tohoto štvorholníku, je vo všetkých ďalších snímkoch videa výsledkom testovaného algoritmu. Okrem vyznačenia tohoto výsledku priamo v UI aplikácie, je ďalším krokom jeho zaznamenanie do textového súboru, ktorého formát je podobný ako formát textovej skutočnej pozície (číslo snímku, ID, X, Y, šírka, výška), s rozdielom, že ako oddeľovací znak je namiesto čiarky použitý tabulátor, z dôvodu jednoduchšieho vloženia výsledkov z textového súboru do programu MS Excel. Výsledkom práce tohoto programu je teda textový súbor, určený na porovnanie so skutočnou pozíciou.

4.3 Výsledky porovnania

Pre porovnávanie úspešnosti sledovacích algoritmov, bol zo skutočnej pozície vybraný objekt s ID = 9. Počiatočná pozícia a veľkosť regiónu, v ktorom sa objekt (v tomto prípade osoba) nachádzala na prvom snímku videa, bola zadaná do inicializačnej metódy daného algoritmu.

4.3.1 Algoritmus Boosting

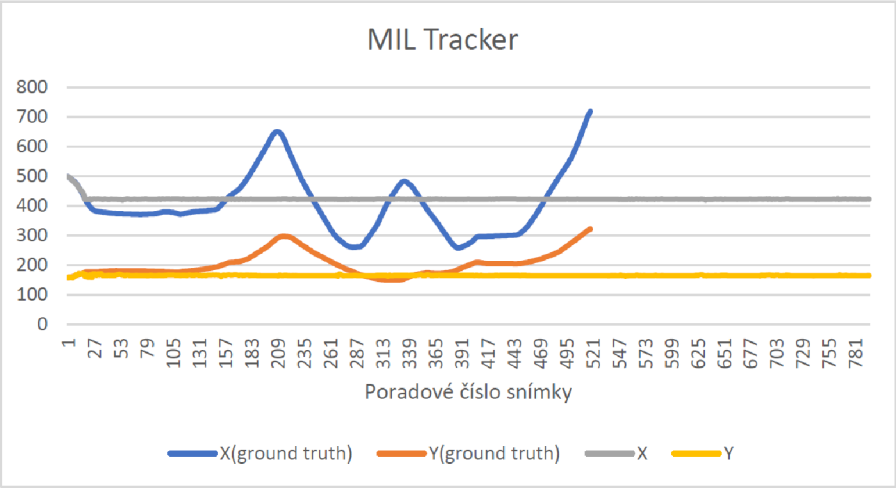
Boosting tracker si v teste neviedol vôbec zle. Aj keď ide o najstarší z testovaných algoritmov, sledovaného objektu sa dokázal držať v porovnaní s ostatnými veľmi dlho, a to aj v prípadoch kedy stála sledovaná osoba z veľkej časti skrytá za iným objektom. Sledovač zlyhal až v momente, kedy v blízkosti sledovanej osoby prechádzala iná osoba, ktorú potom začal algoritmus sledovať. Táto osoba z obrazu po chvíli zmizla a v tom momente sa algoritmus Boosting chytil na jeden zo statických objektov vo videu, ktorého sa potom držal až do konca.
Z grafu na obrázku 4.4, môžeme vidieť že až po snímku s poradovým číslom 160, sú súradnice X,Y algoritmu a skutočnej pozície takmer rovnaké. Potom došlo k zlyhaniu a sledovaniu inej osoby asi po snímku 210. Od tohoto miesta už algoritmus sledoval spomínaný statický objekt, a teda súradnice X a Y sa už nemenili.
Veľkou nevýhodou tohoto algoritmu je, že zlyhanie sledovania nevie spoľahlivo detektovať.



Obr. 4.4: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu BOOSTING so skutočnou pozíciou

4.3.2 Algoritmus MIL

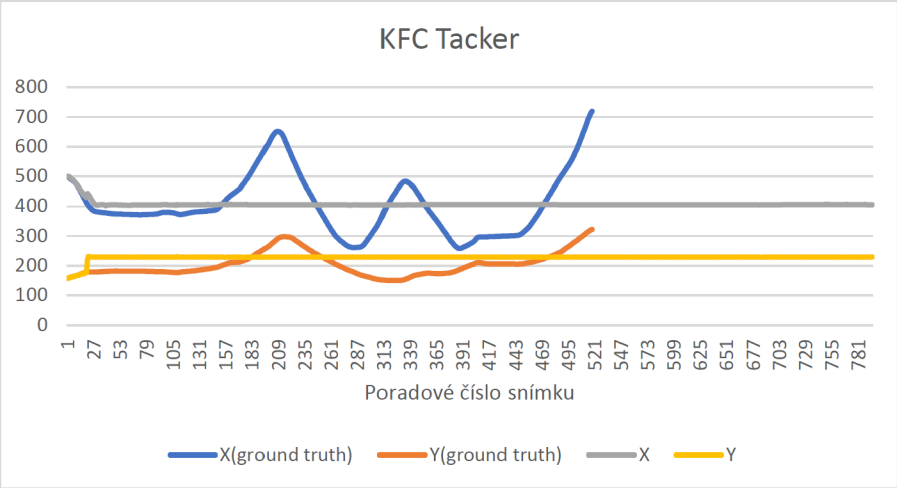
Algoritmus MIL v teste dopadol horšie ako predošlí algoritmus a to vo všetkých ohľadoch. Sledovať cieľový objekt sa mu darilo iba na začiatku videa, približne prvých 30 snímkov. V momente kedy bol sledovaný objekt v obraze prekrytý iným objektom, algoritmus okamžite zlyhal. Toto zlyhanie však ani neodhalil a po zvyšok videa zostalo sledovanie zaseknuté na jednom bode.
Okrem toho bol algoritmus MIL pomerne pomalý a náročný na výpočtový výkon.



Obr. 4.5: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu MIL so skutočnou pozíciou

4.3.3 Algoritmus KCF

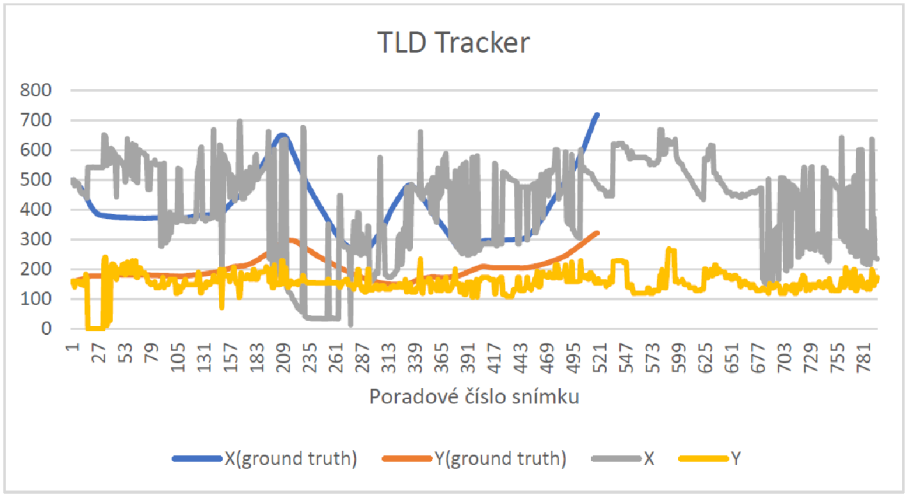
Tento algoritmus s ohľadom na väčšinu sledovaných parametrov dopadol podobne ako algoritmus MIL. V okamihu kedy vo videu sledovaná osoba prejde poza lampu pouličného osvetlenia, algoritmus zlyhá, čo však neodhalí a pokračuje v sledovaní nesprávneho objektu.
Na obrázkoch 4.5 a 4.6 je vidieť, že výsledné súradnice pre MIL a KCF sú takmer rovnaké. V porovnaní s algoritmom MIL bol KCF menej náročný na výpočtový výkon.



Obr. 4.6: Porovnanie výsledných súradníc X a Y z trasovacieho algroritmu KCF so skutočnou pozíciou

4.3.4 Algoritmus TLD

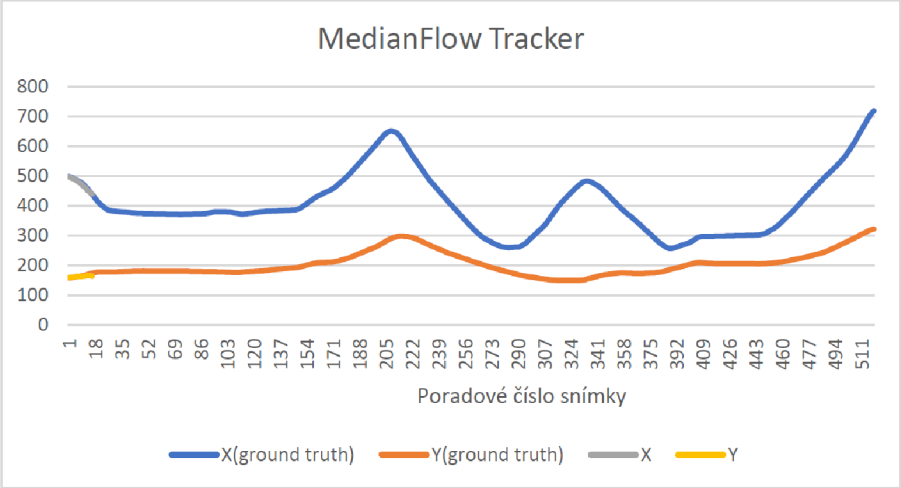
Správanie algoritmu TLD bolo úplne iné, ako všetkých ostatných algoritmov. Tracker v podstate neustále preskakoval z jedného objektu na iný a v danej scéne, kde sa nachádzalo veľa podobných objektov sa javil ako absolútne nepoužiteľný. Ako jediný z testovaných algoritmov výraznejšie menil veľkosť vyznačeného regiónu v obraze. Bol však pomerne náročný na výpočtový výkon.



Obr. 4.7: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu TLD so skutočnou pozíciou

4.3.5 Algoritmus MedianFlow

MedianFlow dokázal sledovať cieľový objekt taktiež len veľmi krátky čas, ale na rozdiel od iných algoritmov svoje zlyhanie spoľahlivo odhalil, a oznámil užívateľovi. Túto vlastnosť môžeme považovať za veľmi veľkú výhodu, pretože, ak algoritmus v trasovaní objektu zlyhá, je možné jednoducho vykonať detekciu cieľového objektu v obraze znovu, a pokračovať v sledovaní.
Algoritmus MedianFlow bol v porovnaní s ostatnými algoritmami pomerne výpočtovo nenáročný.



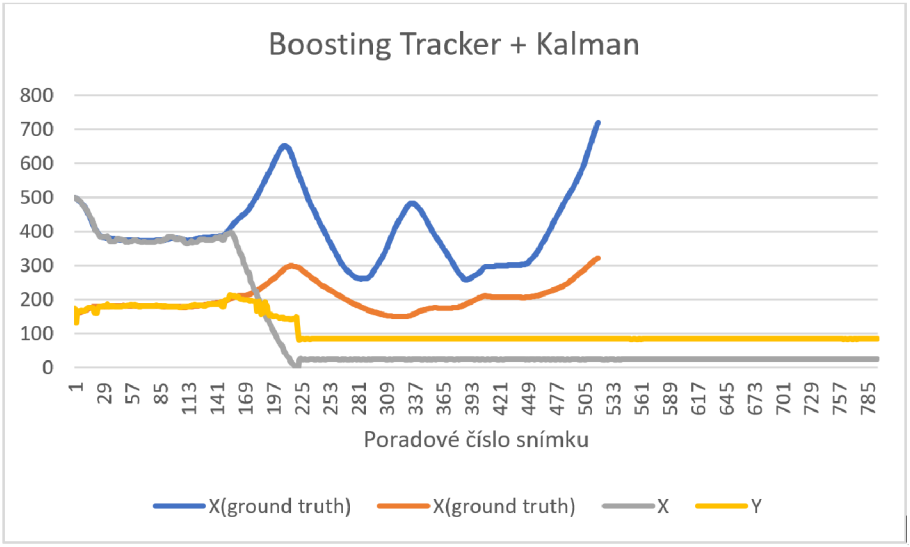
Obr. 4.8: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu MedianFlow so skutočnou pozíciou

4.4 Test s použitím Kalmanového filtra

Jenou z možností ako vylepšiť výsledky sledovacích algoritmov v porovnaní so skutočnou pozíciou, bolo použiť Kalmanov filter. Ako už bolo povedané Kalmanov filter sa používa na odhad budúcej polohy objektu na základe spriemerovania predchádzajúcich odchyliek od očakávanej hodnoty. S ohľadom na predchádzajúce merania, bol predpoklad, že zapojenie Kalmanového filtra, by malo byť prínosom najmä pre sledovací algoritmus TLD, ktorého priebeh súradníc, X resp. Y, bol veľmi skokový. Meranie a porovnanie nameraných výsledkov s Ground Truth bolo však vykonané pre všetky testované algoritmy. V nasledujúcich podkapitolách sú teda prezentované výsledky týchto meraní.

4.4.1 Algoritmus Boosting a Kalmanov filter

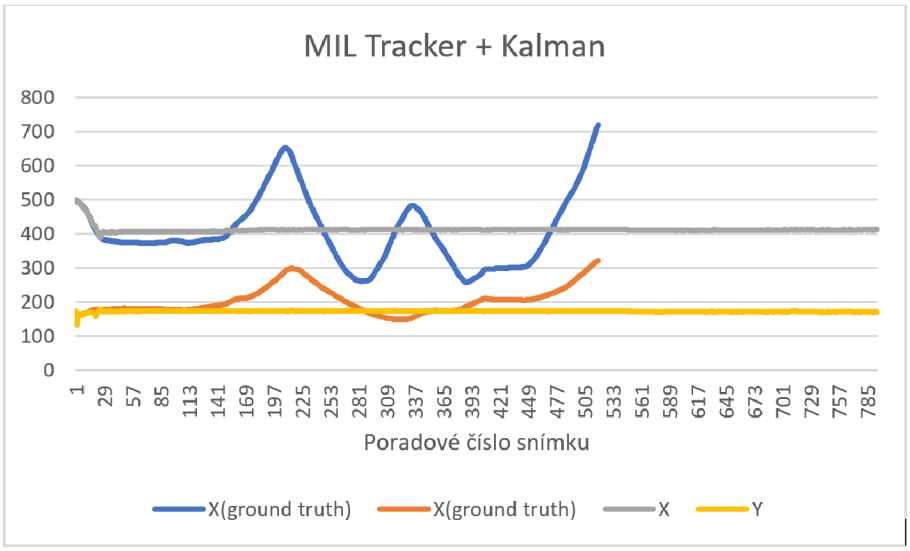
Testovanie sledovacích algoritmov v spolupráci s Kalmanovým filtrom prebehlo v rovnakom poradí ako testovanie bez jeho použitia, a teda prvé hodnoty boli získané pre algoritmus Boosting.
Na obrázku 4.9 je možné vidieť, že v Kalmanov filter nepriniesol žiadnu výraznejšiu zmenu priebehu súradníc X a Y. Niektoré z nameraných hodnôt boli mierne presnejšie, iné však zasa menej presné. Kalmanov filter sa teda v tomto meraní nejavil ako prínosný pre celkovú úspešnosť algoritmu v porovnaní so skutočnou hodnotou.



Obr. 4.9: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu Boosting a Kalmanového filtru, so skutočnou pozíciou

4.4.2 Algoritmus MIL a Kalmanov filter

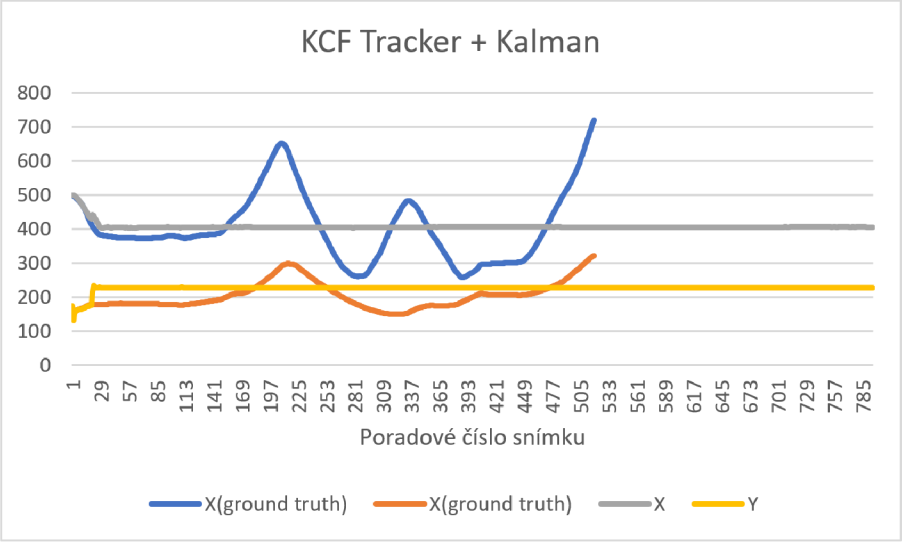
Ďalším algoritmom, ktorý podstúpil „vylepšenie“ Kalmanovým filtrom, bol sledovací algoritmus MIL.
Ako je možné vidieť na obrázku 4.10, výsledok opäť nebol veľmi povzbudivý. Na základe vykonaného merania, je možné tvrdiť, že Kalmanov filter nemal pre daný algoritmus v podstate žiadny prínos.



Obr. 4.10: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu MIL a Kalmanového filtru, so skutočnou pozíciou

4.4.3 Algoritmus KCF a Kalmanov filter

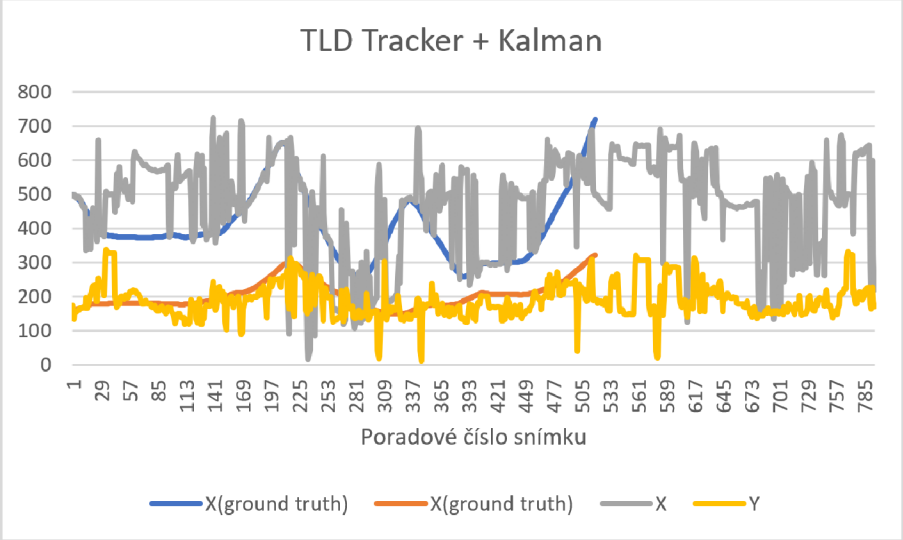
Spolupráca algoritmu KCF a Kalmanovho filtru taktiež (ako v predchádzajúcich prípadoch) nepriniesla výsledky, ktoré by bolo možné z akéhokoľvek pohľadu pokladať za prínos.



Obr. 4.11: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu KCF a Kalmanového filtru, so skutočnou pozíciou

4.4.4 Algoritmus TLD a Kalmanov filter

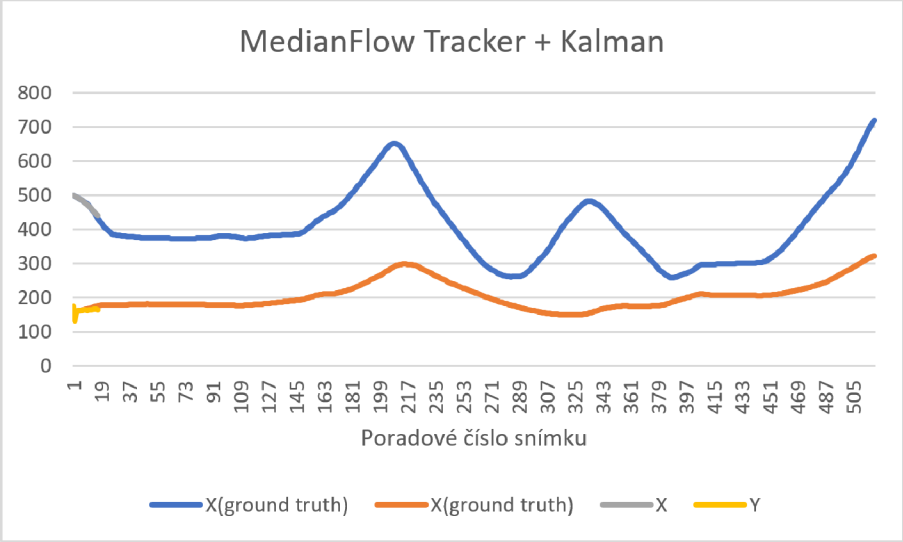
Na základe predchádzajúceho testovania algoritmu TLD a vedomostí o fungovaní Kalmanového filtra, sa jeho použitie, na korekciu výsledkov merania daného algoritmu, javilo ako potencionálne prínosné.
Výsledok ale nenaplnil vysoké očakávania. Je možné konštatovať, že došlo k zlepšeniu, nie však k tak veľkému aby bolo možné označil Kalmanov filter za výraznejší prínos pre úspešnost algoritmu.



Obr. 4.12: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu TLD a Kalmanového filtru, so skutočnou pozíciou

4.4.5 Algoritmus MedianFlow a Kalmanov filter

Pri testovaní spolupráce algoritmu MefianFlow a Kalmanového filtru, boli očakávania naplnené.
Keďže algoritmus svoju prácu ukončí v momente ako stratí sledovaný objekt, nebolo možné predpokladať, že použitie Kalmanového filtru si v tejto aplikácií nájde veľké využitie.



Obr. 4.13: Porovnanie výsledných súradníc X a Y z trasovacieho algoritmu MedianFlow a Kalmanového filtru, so skutočnou pozíciou

4.4.6 Celkové zhodnotenie aplikácie Kalmanového filtru

Na základe meraní v tejto kapitole diplomovej práce, možeme konštatovať, že použitie Kalmanového filtru spolu s testovanými sledovacími algoritmami, malo pre úspešnosť daného algoritmu len malý, alebo žiadny prínos.
Takéto výsledky bolo však možné očakávať na základe princípu fungovania jednotlivých algoritmov. Výnimkou je snáť iba algoritmus TLD, u ktorého bola očakávaná vyššia miera zlepšenia.

5 NÁVRH OPTIMALIZÁCIE ALGORITMU MEDIANFLOW

Ako už bolo spomenuté v predchádzajúcich kapitolách, algoritmus MedianFlow poskytuje jeho užívateľovi informáciu o zlyhaní sledovania objektu veľmi spoľahlivo. Táto vlastnosť, nám poskytuje dobrý priestor na jeho optimalizáciu, pretože v prípade ak algoritmus zlyhá, môžeme detekčným algoritmom sledovaný objekt znovu nájsť a trasovanie obnoviť.
Problémom takéhoto prístupu je však fakt na ktorý poukázal aj test algoritmu TLD, v tretej kapitole tejto práce. Ak počas trasovania objektu dochádza k jeho opätovnej detekcií, hrozí že detektor vyhodnotí ako cieľ objekt, ktorý je sledovanému objektu podobný. V scéne kde je napríklad viac podobných, pohybujúcich sa ľudí je takáto chybná detekcia veľmi pravdepodobná.
Navrhnuté riešenie je nasledujúce.:
Algoritmus si bude pamätať vždy poslednú, úspešne označenú polohu sledovaného objektu. Ak v ďalšej snímke trasovanie zlyhá, detekčný algoritmus vyhľadá novú polohu daného objektu. Ak bude pozícia výsledku detekčného algoritmu príliš vzdialená poslednej známej polohe objektu, tak sa namiesto tohoto výsledku použije práve predošlá poloha.
Popis navrhnutého optimalizovaného algoritmu je na obrázku 5.1

5.1 Ukážka implementácie algoritmu

```
public Image<Bgr, byte> Detect
(VideoCapture capture, Image<Gray, Byte> image_object, ref Rectangle rectangle)
{
    var imageFrame = capture.QueryFrame()?.ToImage<Bgr, Byte>();
    prev = rectangle;

    if (imageFrame == null)
        return null;

    if (!trackingOn || image_object == null)
        return imageFrame;

    do
    {
        var grayframe = imageFrame.Convert<Gray, byte>();
        if (!trackerInitialized)
        {
            FFTService.Instance.DetectObject(grayframe, image_object, ref rectangle);
            if (!prev.IsEmpty && (Math.Abs(prev.Location.X - rectangle.Location.X) > 90
            || Math.Abs(prev.Location.Y - rectangle.Location.Y) > 90))
            {
                rectangle = prev;
            }

            if (tracker == null)
            {
                tracker = new Tracker("MEDIANFLOW");
                trackerInitialized = tracker.Init(grayframe.Mat, rectangle);
            }
            return imageFrame;
        }
        trackerInitialized = tracker.Update(grayframe.Mat, out rectangle);
    }
    while (!trackerInitialized);

    return imageFrame;
}
```

Vyššie sa nachádza úryvok kódu, resp. metóda, ktorá vykonáva práve zmienenú funkcionalitu, teda akési vylepšenie práce sledovacieho algoritmu MedianFlow.
Ako môžeme vidieť, metóda vracia späť do hlavného vlákna aplikácie dve hodnoty. V prvom rade vráti vždy nasledujúci obrázok z práve prehrávaného videa, priamo pomocou návratovej hodnoty typu `Image<Bgr, byte>`. Ďalej, pomocou referenčnej hodnoty je možné získať výsledný obdĺžnik (ang. rectangle), ktorý slúži pre označenie sledovaného objektu v obraze.

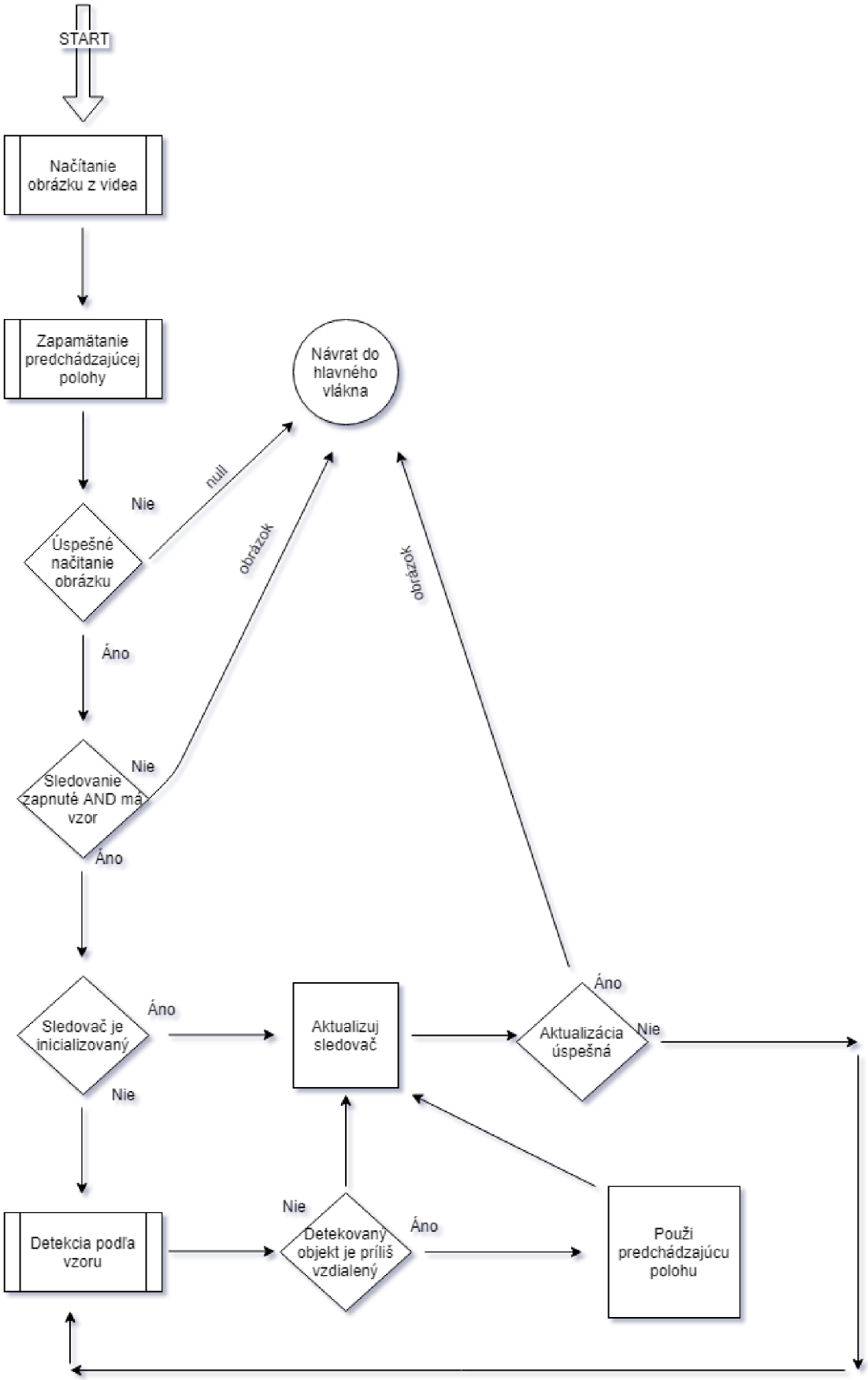
Priebeh metódy v aplikácií je nasledovný:

- Metóda je vyvolaná pre každý „Tick“ časovača (ang. timer), ktorý je nastavený v hlavnom vlákne aplikácie s periódou 40 milisekúnd
- Do metódy sú predané vstupné hodnoty, t.j.:
    1. **parameter `VideoCapture capture`**, ktorý obsahuje dáta, prehrávaného videa
    2. **parameter `Image<Gray, Byte> image_object`**, ktorý obsahuje vzor objektu ktorý má byť sledovaný. V tomto prípade ide o výrez z obrázku videa, na základe akcie užívateľa pomocou kurzoru myši
    3. **referenčný parameter `ref Rectangle rectangle`**, ktorý obsahuje dáta polohy úspešne detekovaného resp. sledovaného objektu. Aby bola hodnota tohoto parametru prenesená do hlavného vlákla aplikácie, je po-

trebné využitie kľúčového slova `ref`, pretože objekt typu `Rectangle` nie je referenčný typ (jedná sa o štruktúru „`struct`")

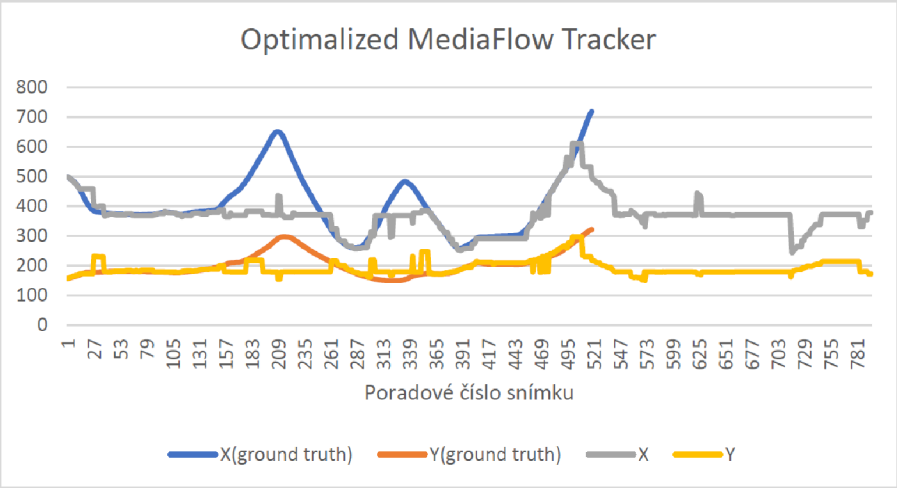
5.1.1 Detailný popis fungovania metódy „Detect“, krok za krokom

V prvom kroku si metóda, pomocou parametru `capture`, zistí nasledujúci obrázok videa. Ďalším krokom je zapamätanie si poslednej polohy sledovaného objektu do premennej `prev`.
Nasleduje podmienka, ktorá kontroluje, či sa podarilo načítať ďaľší obrázok z videa. Ak nie, metóda končí a vracia `null`. Následne metóda skontroluje či je sledovanie zapnuté a zároveň či bol vybratý nejaký vzor. Ak nie, znamená to že užívateľ zatiaľ nevybral žiadny objekt v obraze, a teda nie je potrebné ďalej pokračovať na algoritmus sledovania, či detekcie ale stačí vrátiť nezmenený obrázok.
Po prechode týmito eliminačnými podmienkami, vstupuje kód do cyklu `do while`, ktorý je iterovaný až do momentu úspešnej inicializácie resp. úspešnej aktualizácie pozície použitého sledovača.
V prvej časti cyklu sa vykoná pokus o detekciu vzoru v obrazovej scéne pomocou datekčného servisu FFT. Ak je detekovaný objekt príliš vzdialený od prechádzajúcej detekcie, resp. v prvom behu metódy, príliš vzdialený od pôvodne vyznačeného objektu, použije sa predchádzajúca rep. pôvodná poloha.
Ďalej prebehne kontrola referencie premenne `tracker`. Ak zatiaľ neukazuje na inicializovaný objekt sledovača, resp. ak ukazuje na hodnotu `null`, do premenne priradíme ukázateľ na novú instanciu sledovača, ktorý zároveň inicializujeme pomocou prvej vybranej alebo detekovanej hodnoty.
Po inicializácií sledovača daný cyklus a aj celá metóda končia a do hlavného vlákna je vrátený súčasný obrázok.
Pri každom ďalšom volaní metódy je cyklus `do while` vykonaný iba jeden krát v prípade úspechu sledovača. Ak aktualizácia polohy pomocou sledovača neprebehne úspešne, cyklus sa vykoná znovu, s pokusom detekovať novú polohu sledovaného objektu.



Obr. 5.1: Vývojový diagram navrhnutého algoritmu

5.2 Výsledky porovnania optimalizovaného algoritmu so skutočnou pozíciou

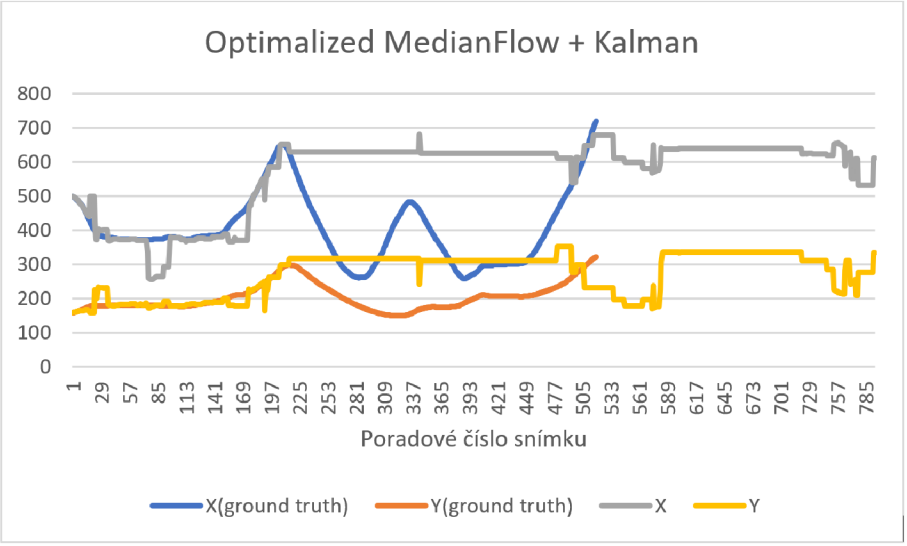


Obr. 5.2: Porovnanie výsledných súradníc X a Y z optimalizovaného trasovacieho algoritmu MedianFlow so skutočnou pozíciou

Optimalizovaný algoritmus bol testovaný voči skutočnej pozícií rovnako, ako všetky algoritmy v predchádzajúcej kapitole. Na obrázku 5.2 je vidieť, že optimalizovaný algoritmus si viedol podstatne lepšie ako pôvodný. Jeho nevýhodou je však častá potreba detekcie, ktorá je pomerne náročná na výpočtový výkon. Ďalšou nevýhodou je, že v momente, keď sa už sledovaný objekt v obraze nenachádza, algoritmus začne spravidla trasovať iný objekt.

5.2.1 Pridanie Kalmanového filtru

Rovnako ako v prípade ostatných sledovacích algoritmov, aj v tomto prípade bolo vykonané meranie aj s použitím Kalmanového filtru.
Výsledný efekt však nebol pre sledovač prínosom, ako je to možné vidieť na obrázku 5.3.



Obr. 5.3: Porovnanie výsledných súradníc X a Y z optimalizovaného trasovacieho algoritmu MedianFlow, pri použití Kalmanového filtru, so skutočnou pozíciou

5.3 Aplikácia využívajúca optimalizovaný sledovací algoritmus

Súčasťou prílohy k tejto práci je zdrojový kód aplikácie, implementovanej v programovacom jazyku C#, ktorá aktívne využíva vyššie popisovaný, optimalizovaný algoritmus na sledovanie objektov vo videu.
Pri implementácií bola použitá knižnica EmguCV vo verzií *3.2*.
Rozhranie aplikácie je veľmi jednoduché. Obsahuje iba tlačítko na vybratie súboru z disku a plochu, na ktorej sú zobrazené jednotlivé snímky prehrávaného videa (viď obrázok 5.4).

Obr. 5.4: Aplikácia využívajúca optimalizovaný algoritmus

Po stlačení tlačítka „Open“ a výbere videa, sa video automaticky začne prehrávať. Užívateľ môže pomocou kliknutia a podržania ľavého tlačítka myši označiť ľubovolný objekt v prehrávanom videu. Tento objekt bude následne sledovaný po celú dobu prehrávania videa. Pre uľahčenie práce s programom, je v momente stlačenia ľavého tlačítka myši prehrávanie pozastavené. Video v prehrávaní pokračuje automaticky po uvoľnení tlačítka.
Táto aplikácia bola vyvíjaná a testovaná vo vývojovom prostredí Visual Studio 2017. Pre úspešnú kompiláciu zdrojového kódu, ktorý je súčasťou prílohy, je potrebné úspešné stiahnutie externých knižníc. Visaul Studio vykoná ich stiahnutie automaticky z tzv. nuget-u.

6 ZÁVER

Táto diplomová práca bola zameraná na súčasné možnosti trasovania pohybujúcich sa objektov vo videu.
Popisuje funcionalitu sledovacích algoritmov:

- Algoritmus BOOSTING
- Algoritmus MIL
- Algoritmus KCF
- Algoritmus TLD
- Algoritmus MEDIANFLOW

Vybrané sledovacie algoritmy boli najprv teoreticky popísané, následne implementované pomocou knižnice EmguCV a nakoniec aj otestované voči skutočnej pozícií. Ďalej bol vykonaný pokus v ktorom sledovacie algoritmy mali spolupracovať s Kalmanovým filtrom. Kalmanov filter však nebol výrazne prínosný, čo je možné tvrdiť na základe výsledkov merania.
Po vzhliadnutí na všetky nadobudnuté informáce bol navrhnutý, implementovaný a otestovaný algoritmus, ktorý sa snaží o zlepšenie vlastností jedného z testovaných algoritmov. Výsledkom tejto práce je potom funkčná aplikácia typu WPF, implementovaná v programovacom jazyku C#, ktorá daný algoritmus aktívne používa.
V prílohe k tejto práci sa nachádza časť nameraných dát. Kompletné dáta z vykonaného merania, ako aj spustiteľný zdrojový kód aplikácie je súčasťou priloženého CD.

LITERATÚRA

[1] *accort - net* [online]. [cit. 23. 04. 2019]. Dostupné z URL: <https://github.com/accord-net/framework/wiki/Getting-started>

[2] ALDHAHERI, Asim R. a Eran A. EDIRISINGHE: *Detection and Classification of a Moving Object in a Video Stream* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <http://www.seekdl.org/conferences_page_papers.php?confid=115>

[3] ELENA ŠIKUDOVÁ, ZUZANA ČERNEKOVÁ, WANDA BENEŠOVÁ, ZUZANA HALADOVÁ, JÚLIA KUCEROVÁ: *Počítačové videnie, detekcia a rozpoznávanie objektov* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <http://sccg.sk/~cernekova/Pocitacove_videnie.pdf>

[4] EMGU CV: *EmguCv Main Page* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <http://www.emgu.com/wiki/index.php/Main_Page>

[5] *Forward-Backward Error: Automatic Detection of Tracking Failures* [online]. [cit. 20. 04. 2019]. Dostupné z URL: <http://kahlan.eps.surrey.ac.uk/featurespace/tld/Publications/2010_icpr.pdf>

[6] *Framework modules* [online]. [cit. 21. 04. 2019]. Dostupné z URL: <http://accord-framework.net/docs/html/R_Project_Accord_NET.htm9>

[7] *How a Kalman filter works, in pictures* [online]. [cit. 25. 04. 2019]. Dostupné z URL: <https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>

[8] *Introduction to the C# Language and the .NET Framework* [online]. 20. 07. 2015 [cit. 29. 04. 2019]. Dostupné z URL: <https://docs.microsoft.com/cs-cz/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>

[9] Lee, Hasup, HyungSeok Kim, Jee–In Kim: *Background Subtraction Using Background Sets With Image- and Color-Space Reduction. In: IEEE Transactions on Multimedia* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <http://ieeexplore.ieee.org/document/7524018/>

[10] OPEN CV: *opencv.org* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <https://opencv.org/>

[11] OpenCV 3.0.0 documentation: *Tracker Algorithms* [online]. [cit. 5. 12. 2018]. Dostupné z URL: <https://docs.opencv.org/3.0-beta/modules/tracking/doc/tracker_algorithms.html>

[12] SATYA MALLICK: *Object Tracking using OpenCV (C++/Python)* [online]. 13. 2. 2017 [cit. 5. 12. 2018]. Dostupné z URL: <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/>

[13] JAMES KOBIELUS: *The Ground Truth in Agile Machine Learning* [online]. 13. 6. 2014 [cit. 6. 12. 2018]. Dostupné z URL: <http://www.ibmbigdatahub.com/blog/ground-truth-agile-machine-learning>

[14] Kvasnička, Beňušková, Pospíchal, Farkaš, Tiňo, Kráľ: *Úvod do teórie neurónových sietí* [online]. [cit. 23. 04. 2019]. Dostupné z URL: <https://encyklopediapoznania.sk/data/eknihy/informatika/uvod_do_teorie_neuronovych_sieti.pdf>

[15] *Welcome to the Visual Studio IDE* [online]. 19. 03. 2019 [cit. 20. 04. 2019]. Dostupné z URL: <https://docs.microsoft.com/cs-cz/visualstudio/get-started/visual-studio-ide?view=vs-2019>

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

MOG	Mixture of Gaussians
WPF	Windows Presentation Foundation
CD	Compact Disc
ang.	anglicky

ZOZNAM PRÍLOH

A SKUTOČNÁ POZÍCIA A NAMERANÉ DÁTA PRE JEDNOTLIVÉ SLEDOVACIE ALGORITMY

Nasledujúce prílohy zobrazujú ukážku nameraných dát pre optimalizovaný sledovací algoritmus a skutočnú pozíciu (Ground Truth). Ide o prvých 100 meraných obrázkov. Kompletné dáta z meraní je možné nájsť na priloženom CD.

A.1 Skutočná pozícia objektu s ID = 9 (prvých 100 obrázkov)

GROUND TRUTH					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
1	9	499	158	31,10	75,17
2	9	497	158	31,03	75,17
3	9	495	159	31,03	75,17
4	9	492	160	31,03	75,17
5	9	489	161	31,03	75,17
6	9	487	162	31,03	75,17
7	9	484	163	31,03	75,17
8	9	481	163	31,03	75,17
9	9	478	165	31,03	75,17
10	9	474	166	31,03	75,17
11	9	469	167	31,03	75,17
12	9	463	168	31,03	75,17
13	9	458	170	31,03	75,17
14	9	451	171	31,03	75,17
15	9	445	173	31,03	75,17
16	9	438	174	31,03	75,17
17	9	432	175	31,03	75,17
18	9	425	176	31,03	75,17
19	9	418	177	31,03	75,17
20	9	412	178	31,03	75,17
21	9	407	178	31,03	75,17
22	9	402	178	31,03	75,17
23	9	398	178	31,03	75,17
24	9	394	178	31,03	75,17
25	9	390	178	31,03	75,17

GROUND TRUTH					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
26	9	387	178	31,03	75,18
27	9	385	178	31,03	75,19
28	9	384	178	31,03	75,20
29	9	383	178	31,03	75,21
30	9	382	178	31,03	75,23
31	9	382	178	31,03	75,24
32	9	381	178	31,03	75,26
33	9	381	179	31,03	75,27
34	9	380	179	31,03	75,29
35	9	380	179	31,03	75,31
36	9	379	179	31,03	75,32
37	9	379	180	31,03	75,34
38	9	379	180	31,03	75,35
39	9	378	180	31,03	75,37
40	9	378	180	31,03	75,39
41	9	377	180	31,03	75,40
42	9	377	181	31,03	75,42
43	9	376	181	31,03	75,43
44	9	376	181	31,03	75,45
45	9	375	181	31,03	75,47
46	9	375	181	31,03	75,49
47	9	375	181	31,03	75,52
48	9	375	181	31,03	75,55
49	9	374	182	31,03	75,59
50	9	374	181	31,03	75,63
51	9	374	181	31,03	75,67
52	9	374	181	31,03	75,71
53	9	374	181	31,03	75,75
54	9	374	181	31,03	75,79
55	9	374	181	31,03	75,83
56	9	373	181	31,03	75,87
57	9	373	181	31,03	75,91
58	9	373	181	31,03	75,95
59	9	373	181	31,03	75,99
60	9	373	181	31,03	76,04
61	9	373	181	31,03	76,08
62	9	373	181	31,03	76,12

GROUND TRUTH					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
63	9	373	181	31,03	76,16
64	9	372	181	31,03	76,20
65	9	372	181	31,03	76,24
66	9	372	181	31,03	76,28
67	9	372	181	31,03	76,32
68	9	372	181	31,03	76,36
69	9	372	181	31,03	76,40
70	9	372	181	31,03	76,44
71	9	372	181	31,03	76,48
72	9	371	181	31,02	76,52
73	9	371	181	31,01	76,56
74	9	371	181	30,99	76,60
75	9	372	181	30,97	76,64
76	9	372	181	30,94	76,69
77	9	372	181	30,90	76,73
78	9	372	181	30,86	76,77
79	9	372	180	30,83	76,81
80	9	372	180	30,79	76,85
81	9	373	180	30,75	76,89
82	9	373	180	30,72	76,94
83	9	373	180	30,68	76,98
84	9	373	180	30,64	77,02
85	9	373	180	30,61	77,06
86	9	373	180	30,57	77,10
87	9	374	179	30,53	77,14
88	9	374	179	30,49	77,19
89	9	374	179	30,43	77,23
90	9	375	179	30,36	77,27
91	9	376	179	30,27	77,32
92	9	377	179	30,14	77,36
93	9	378	179	30,00	77,41
94	9	379	179	29,85	77,46
95	9	380	179	29,69	77,50
96	9	380	179	29,54	77,55
97	9	380	178	29,38	77,60
98	9	380	178	29,22	77,65
99	9	380	178	29,06	77,69

GROUND TRUTH					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
100	9	380	178	28,91	77,74

A.2 Namerané hodnoty pre navrhnutý optimalizovaný sledovač (prvých 100 obrázkov)

Optimalized MedianFlow					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
1	9	499	158	31	75
2	9	497	159	31	75
3	9	496	160	31	75
4	9	492	162	31	75
5	9	489	162	31	75
6	9	486	164	31	75
7	9	483	166	31	75
8	9	478	165	31	75
9	9	477	168	31	75
10	9	473	167	31	75
11	9	466	169	31	75
12	9	464	171	31	75
13	9	459	174	31	75
14	9	459	174	31	75
15	9	459	174	31	75
16	9	459	174	31	75
17	9	459	174	31	75
18	9	459	174	31	75
19	9	459	174	31	75
20	9	459	174	31	75
21	9	459	174	31	75
22	9	459	174	31	75
23	9	459	174	31	75
24	9	459	174	31	75
25	9	459	174	31	75
26	9	459	174	31	75
27	9	403	233	31	75
28	9	402	232	31	75
29	9	400	232	31	75

Optimalized MedianFlow					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
30	9	399	232	31	75
31	9	399	231	31	75
32	9	400	231	31	75
33	9	401	230	31	75
34	9	401	231	31	75
35	9	401	231	31	75
36	9	401	231	31	75
37	9	369	182	31	75
38	9	368	181	31	75
39	9	369	181	31	75
40	9	370	180	31	75
41	9	371	179	31	75
42	9	373	179	31	75
43	9	374	179	31	75
44	9	374	179	31	75
45	9	374	180	31	75
46	9	374	181	31	75
47	9	373	181	31	75
48	9	372	182	31	75
49	9	371	182	31	75
50	9	371	182	31	75
51	9	371	182	31	75
52	9	371	182	31	75
53	9	371	182	31	75
54	9	371	182	31	75
55	9	371	182	31	75
56	9	371	182	31	75
57	9	371	182	31	75
58	9	371	182	31	75
59	9	375	185	31	75
60	9	375	185	31	75
61	9	375	185	31	75
62	9	375	185	31	75
63	9	375	185	31	75
64	9	375	185	31	75
65	9	370	181	31	75
66	9	370	181	31	75

Optimalized MedianFlow					
FrameNr	**ID**	**X**	**Y**	**Width**	**Height**
67	9	369	180	31	75
68	9	369	182	31	75
69	9	369	185	31	75
70	9	369	182	31	75
71	9	369	181	31	75
72	9	369	180	31	75
73	9	369	181	31	75
74	9	369	186	31	75
75	9	369	186	31	75
76	9	369	186	31	75
77	9	369	186	31	75
78	9	369	186	31	75
79	9	369	186	31	75
80	9	369	186	31	75
81	9	369	186	31	75
82	9	369	186	31	75
83	9	369	186	31	75
84	9	369	186	31	75
85	9	369	186	31	75
86	9	371	181	31	75
87	9	372	180	31	75
88	9	372	180	31	75
89	9	374	180	31	75
90	9	375	180	31	75
91	9	375	180	31	75
92	9	375	180	31	75
93	9	375	180	31	75
94	9	375	180	31	75
95	9	375	180	31	75
96	9	375	180	31	75
97	9	383	179	31	75
98	9	382	179	31	75
99	9	381	179	31	75
100	9	378	179	31	75

B OBSAH PRILOŽENÉHO CD

1. TrackingTest.xlsx - namerané hodnoty
2. Zdrojový kód výslednej aplikácie