VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA ZÁZNAMOV BEZPEČNOSTNÝCH KAMIER

ANALYSIS OF SURVEILLANCE CAMERA RECORDINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ŠÁRKA ŠČAVNICKÁ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Studentka: **Ščavnická Šárka**
Program: Informační technologie
Název: **Analýza záznamů bezpečnostních kamer**
Analysis of Surveillance Camera Recordings
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte aktuální přístupy ke zpracování a analýze záznamů bezpečnostních kamer.
2. Zpracujte přehled dostupných datových sad vhodných pro vyhodnocování kvality analýzy záznamů.
3. Navrhněte a realizujte systém pro rozpoznávání a prohledávání záznamů bezpečnostních kamer a řazení výsledků podle relevance vzhledem k zadanému dotazu.
4. Vyhodnoťte výsledky systému na shromážděných datech.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- Dle domluvy s vedoucím

Pro udělení zápočtu za první semestr je požadováno:

- Funkční prototyp řešení

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

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Táto práca sa zaoberá systémami na analyzovanie záznamov z bezpečnostných kamier. Cieľom bolo vytvorenie funkčného riešenia, ktoré bude analyzovať záznamy a odpovedať na dotazy od užívateľa. Vytvorený systém kombinuje algoritmus YOLO pre detekciu objektov a DeepSORT na ich následné sledovanie. Celkovo obsahuje päť modelov, ktoré detekujú špecifické situácie. Jednotlivé modely dosiahli počas testovania rôznej úspešnosti, pričom najnižšia úspešnosť bola 58 % a išlo o model detekcie vystúpenia z auta. Najvyššiu úspešnosť, 83 %, dosiahol model na detekciu stretnutia medzi dvomi ľudí.

Abstract

This thesis deals with the systems for analyzing records from security cameras. It aims to create a functional solution that analyzes records and answers questions from the user. The created system combines the YOLO algorithm for object detection and DeepSORT for their subsequent tracking. It contains five models that detect specific situations. Individual models achieved varying degrees of success during testing, with the lowest success rate being 58 % for the "getting out of car" situation. The highest success rate, 83 %, was obtained by a model for detecting a meeting between two people.

Kľúčové slová

vyhľadávanie v záznamoch bezpčnostných kamier, detekcia objektov, sledovanie objektov, homografia, problémy spojené so sledovaní objektov, matica zámen

Keywords

surveillance video search, object detection, object tracking, homography, object tracking problems, confusion matrix

Citácia

ŠČAVNICKÁ, Šárka. *Analýza záznamov bezpečnostných kamier*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Analýza záznamov bezpečnostných kamier

Prehlásenie

Prehlasujem, že som túto bakalárskou prácu vypracovala samostatne pod vedením pána Doc. RNDr. Pavla Smrža Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Šárka Ščavnická
12. mája 2021

Poďakovanie

Chcela by som sa poďakovať vedúcemu mojej práce, pánovi Doc. RNDr. Pavel Smrž, Ph.D., za jeho rady a vedenie počas celej bakalárskej práce. Taktiež by som sa chcela poďakovať mojej rodine, za ich podporu a pomoc pri testovaní tejto práce.

Obsah

1 Úvod **3**

2 Spracovanie záznamov bezpečnostných kamier **5**
2.1 Video analýza 5
2.2 Schéma systémov spracovávajúcich záznamy z bezpečnostných kamier . . . 6
2.3 Predstavenie systémov na prehľadávanie videí 8
2.3.1 IBM Intelligent Video Analytics 8
2.3.2 Axxonsoft 9
2.3.3 Camio 10

3 Počítačové videnie a homografia **12**
3.1 Počítačové videnie 12
3.1.1 Čo je homografia? 12
3.2 Perspektívna projekcia 13

4 Detekcia objektov **16**
4.1 Úvod do detekcie objektov 16
4.2 Tradičné metódy 17
4.2.1 Viola Jones detectors 17
4.2.2 Detekcia objektov založená na histograme orientovaných gradientov 19
4.3 Metódy založené na hlbokom učení (*deep learning*) 22
4.3.1 Faster RCNN 22
4.3.2 YOLO 23

5 Metódy na porovnávanie kvality algoritmov **25**
5.1 Matica zámen 25
5.2 ROC krivky 27

6 Sledovanie objektov **28**
6.1 Metódy založené na sledovaní bodov 29
6.1.1 Kalman Filter 29
6.2 Kernel tracking 30
6.2.1 Mean-shift 30
6.3 Metódy založené na sledovaní siluet 31
6.4 DeepSORT algoritmus 32
6.5 Problémy sledovania objektov 32

7 Implementácia **34**

7.1 Prehľad systému 35
7.2 Nástroje použité pri implementácii 35
7.3 Architektúra 37
7.3.1 Serverová časť aplikácie 37
7.3.2 Klientska časť aplikácie 42

8 Dosiahnuté výsledky **44**
8.1 Testovanie funkcí systému 44
8.2 Testovanie za pomoci testrov 48
8.3 Budúci vývoj 48

9 Záver **49**

Literatúra **50**

A Testovací protokl **56**

B Plagát **58**

Kapitola 1

Úvod

Neustály vývoj moderných technológii priniesol revolúciu v zaisťovaní verejného ale aj súkromného bezpečia a poriadku. Jednou z technológii, ktoré výrazne zmenili náš každodenný život, boli bezpečnostné kamery. Začiatky používania bezpečnostných kamier boli ťažké, nakoľko ich nebolo veľa a záznamy častokrát neboli dostatočne kvalitné. Avšak postupom času, ako narastala ich kvalita a aj využívanie záznamov pri objasňovaní trestných činov – ako bolo napríklad objasnenie tragickej vraždy dvojročného Jamesa Bulgera z roku 1993 pomocou bezpečnostnej kamery umiestnenej v Londýnskom supermarkete – začal ich počet drasticky stúpať. Tento rast od deväťdesiatych rokov dodnes neskončil.

Ceduľa na začiatku mesta alebo dediny, informujúca nás o tom, že je daná oblasť monitorovaná kamerovým záznamom sa stala bežnou súčasťou našich životov. Už ani nevnímame, z ktorých všetkých strán na nás tieto kamery smerujú. Prešli ste s autom na červenú? Myslíte si, že tam nebol policajt a tak je to v poriadku? Do schránky vám príde pokuta. Vykradne niekto obchod? Večer už fotka lupiča z bezpečnostnej kamery beží vo všetkých zprávach. Bezpečnostné kamery vidia všetko a neboja sa to ukázať bezpečnostným zložkám.

Spojením polície s bezpečnostnými kamerami vznikol silný tím, ktorí pomáha objasňovať prípady po celom svete. Jedným z najvýznamnejších úspechov tohto spojenia, bolo identifikovania páchateľov atentátu na bostonský maratón v roku 2013. Už tri dni po atentáte boli zverejnené zábery dvoch podozrivých osôb. Medzi ďalšie trestné činy, ktoré boli objasnené pomocou bezpečnostných kamier patria únosy, vraždy či krádeže.

S nárastom množstva záznamov vyprodukovaných bezpečnostnými kamerami je nutnosťou zefektívniť spôsob ich analýzy. Najdlhšie sa na analýzu používajú ľudia, ktorí musia detailne prechádzať každý záznam a skúmať jednotlivé udalosti. Existujú ale aj záznamy, na ktorých sa nič nedeje a preto je vhodné používať systémy na analýzu videí, ktoré dokážu nájsť len ich významne časti, ktoré ďalej môže skúmať polícia.

Každý systém má svoje chyby a výnimkou nie sú ani systémy spracovávajúce záznamy z bezpečnostných kamier. Kvalitu týchto systémov môžeme hodnotiť podľa rýchlosti analýzy a jej prestnosti. Presnosť analýzy ovplyvňuje spôsob akým implementujeme náš systém, používame neurónové siete, aby sa systém naučil rozpoznávať určité udalosti, alebo máme dopredu definované modely situácii, ktoré chceme nájsť na záznamoch. Výber spôsobu sa odvíja od účelu a nárokov na daný systém. Cieľom tejto práce je preskúmať tieto rôzne metódy a vysvetliť procesy, ktoré vytvárajú systémy spracovávajúce videozáznamy. Ako aj následne navrhnúť a vytvoriť vlastný systém.

Väčšina ľudí sa domnieva, že tieto technológie im nezasahujú do života, avšak nie je tomu tak. Bežne sa s týmito systémami stretávame napríklad v galériách naších mobilných zariadení. Automaticky vytvorené zložky, ktoré roztrieďujú všetky fotky a videá v našom

telefóne do jednoduchých kategórii, podľa toho či sa na obrázku nachádza zviera, alebo významná pamiatka, už považujeme za samozrejmosť. Málokto už vie, že ide o proces detekcie objektov, o ktorej si môžete prečítať v kapitole číslo 4.

Telefóny nie sú jedinými modernými zariadeniami, ktoré spracovávajú obraz. Autá sú inteligentnejšie s každou novou generáciou modelu. To isté sa dá povedať i o veľkom množstve iných domácich spotrebičov. I preto je dôležité postarať sa o to, aby sa moderné prístroje mohli pozerať na svet svojimi vlastnými očami. Kvôli tomu sa musel vyriešiť problém perspektívnej projekcie v moderných systémoch. Veci, ktoré sú od nás vzdialené vyzerajú menšie než keď sú blízko nás, pričom vieme, že reálne rozmery objektu sa nemenia. Perspektívna transformácia umožňuje systémom zvládať tento problém a v kapitole číslo 3 sa môžete dozvedieť, ako tento náročný proces funguje.

Pri analýze videí je taktiež dôležitá schopnosť sledovania objektov naprieč celým videozáznamom. Nestačí len identifikovať o aký objekt ide, ale aj to, ako sa pohyboval, či aké činnosti vykonával. Schopnosť kvalitne sledovať objekty a predvídať ich pohyb je veľmi ťažká a dôležitá, nakoľko rozhoduje to o kvalite výsledného systému. Preto mu bola venovaná jedna časť tejto práce, konkrétne ide o 6 kapitolu.

V praktickej časti tejto práce je implementovaný systéme, u ktorého sú použité technológie, ako je napríklad detekcia objektu, perspektívna transformácia a sledovanie detekovaných objektov. Systém je schopný vyhľadávať na videozáznamoch rôzne udalosti a skladá sa z piatich modelov. U jednotlivých modelov je možné zúžiť rozsah vyhľadávaní vďaka atribútom, ktoré môže užívateľ ľubovoľne meniť. Medzi tieto atribúty patrí určenie farby objektu, alebo vzdialenosti medzi objektami. Ako príklad modelu, ktorý bol v systéme implementovaný, je model na detekciu situácie, kedy je taška vymenená medzi dvomi osobami. Tento model môže polícia používať napríklad pri identifikácii situácii kedy došlo k odcudzeniu tašky.

Kapitola 2

Spracovanie záznamov bezpečnostných kamier

Prvá časť tejto práce skúma existujúce systémy používané na vyhľadávanie v záznamoch z bezpečnostných kamier.

2.1 Video analýza

Video dohľad je proces analyzujúci video sekvencie. Ide o aktívnu oblasť počítačového videnia, ktorá pracuje s veľmi veľkým množstvo dát. Existujú tri typy aktivít pri video monitoringu [45]:

- manuálne monitorovanie
- poloautonómne monitorovanie
- úplne autonómne monitorovanie

Manuálna analýza videí vyžaduje prítomnosť človeka, ktorý zábery skúma. Takéto systémy sú v súčasnosti stále široko používané, i keď sa stretávame aj so systémami, kde systémy pracujú úplne autonómne [45]. Pod takýmto plne autonómnym systémom rozumieme systém, ktorého vstupom je sekvencia záberov zaznamenaných na mieste, kde sa vykonáva dohľad a neexistuje žiaden ľudský zásah, vtedy systém vykonáva úlohy na nízkej úrovni, ako je napríklad detekcia a sledovanie pohybu, ako aj úlohy na vysokej úrovni rozhodovania, ako je detekcia abnormálnych udalostí a rozpoznávanie gest. Premostením tých dvoch prístupov sú poloautonnóme systémy, pri ktorých video analýza zahŕňa určitú časť systému, ale s významným ľudským zásahom. Typickým príkladom sú systémy, ktoré vykonávajú jednoduchú detekciu pohybu a iba v prípade významného pohybu je video odoslané na analýzu expertom.[45]

Bezpečnostné záznamy sa používajú dvomi hlavnými spôsobmi [39]:

- sledovanie hrozieb v reálnom čase
- vyhľadávanie udalostí, ktoré nás zaujímajú

Skúmanie v reálnom čase sa používa na ohraničenom území, ako je napríklad letisko, s cieľom zabrániť verejnému ohrozeniu. Pri vyšetrovaní sa používajú videá, ktoré zahŕňajú väčšiu geografickú oblasť [39]. Systémy spracovávajúce takéto záznamy pracujú s veľkým

množstvom dát, čo je časovo náročný proces. Presnosť a rýchlosť týchto systémov je kľúčová k ochrane jedincov. Na druhú stranu sa video analýza používa aj na analýzu historických údajov k získaniu štatistík. Táto úloha forenznej analýzy dokáže zistiť trendy a vzorce, ktoré zodpovedajú rôzne otázky, ako napríklad, v ktorom čase je najväčšie vyťaženie, aká cieľová veková skupina a ďalšie.[7]

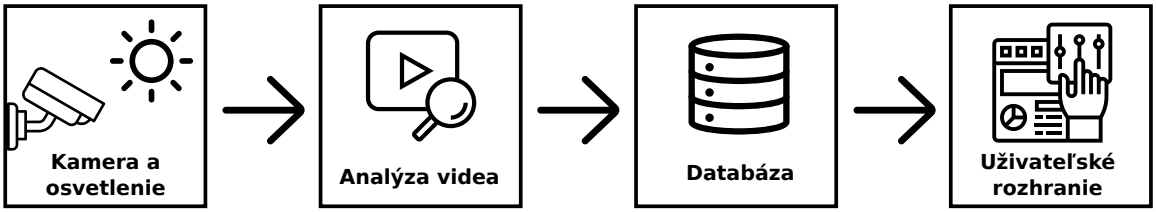
Jednou zo základných výziev počítačového videnia je rozlíšiť, aké fyzické objekty sa nachádzajú na jednej snímke od jednej kamery a ako sa pohybujú na sekvencii záberov z danej kamery. Tieto činnosti sa označujú ako detekcia (kapitola 4) a sledovanie objektov (kapitola 6).[24]

Konečnou fázou monitorovacích systémov je snaha o pochopenie správania sledovaných objektov a interakcie medzi objektami a prostredím. Bežným prístupom je používanie preddefinovaných aktivít, ktoré dodržiavajú niektoré základné pravidlá, ktoré sa systémy učia počas tréningovej fázy [24]. Počas klasifikácie sa buď vyhľadávajú akcie, ktoré odpovedajú naučeným modelom alebo anomálie, ktoré nezapadajú do naučených modelov. Opustený kus batožiny alebo osoby, ktorá sa pohybujú v oblasti s obmedzeným prístupom sú typické príklady anomálnych udalostí.[24][27]

Celková schopnosť automaticky analyzovať videozáznamy a extrahovať z nich objekty, detekovať pohyb týchto objektov ako aj detekovanie udalostí a vykonávanie behaviorálnej analýzy sa označuje ako video analýza.[24]

2.2 Schéma systémov spracovávajúcich záznamy z bezpečnostných kamier

Systémy na sledovanie videozáznamov sa líšia spôsobom akým fungujú, a zložitosťou povolených dotazov. I tak je ale možné predstaviť všeobecnú schému, podľa ktorej tieto systémy fungujú. Môžeme nájsť systémy, ktoré sú schopné len detekovať rôzne akcie, napríklad pohyb v skúmanej oblasti, pričom ale objekt zostáva neznámy, cez systémy, ktoré sú schopné nájsť situácie, keď sa osoba presunula z jedného miesta na druhé. Až nakoniec skončíme u systémov, ktoré sú schopné nájsť konkrétnu osobu, ktorá zodpovedá zadaným parametrom, ktorá sa pohybovala v konkrétnej oblasti so špecifickým smerom pohybu.[24][27]



Obr. 2.1: Všeobecná schéma systému spracovávajúceho záznamy

Prvý významný prvok systému sa skladá zo symbiózy osvetlenia a kamery. Aby ľudia videli a boli videní, je potrebné dostatočné osvetlenie. Bezpečnostné osvetlenie sa používa na zvýšenie viditeľnosti okolo obvodových línii, citlivých miest a taktiež v nočných hodinách. Pri osvetlení vstupov je dôležité, aby osvetlenie bolo dostatočné na zaistenie identifikácie osôb a vozidiel, ktoré vstupujú alebo opúšťajú monitorovanú oblasť [21]. Monitorovaná oblasť je vytvorená buď pomocou jednej kamery alebo agregáciou zorného poľa prepojených kamier [24]. Kamery by mali mať dostatočné rozlíšenie – HD 1080p alebo jeho ekvivalent a taktiež by mali poskytovať farebné zábery, aby sa maximalizoval rozsah detekcie [21].

Dôležitú úlohu zohráva aj kodek používaný kamerou. Väčšina systémov vytvára obrovské množstvo nahrávok. Z tohto dôvodu je nutné nájsť rovnováhu medzi kompresným pomerom kodeku a výslednou kvalitou záznamu. V súčasnosti sa v kamerách najčastejšie používa kodek H.264. Proces kódovania H.264 je rozdelený do troch častí, ktoré zahŕňajú predikciu, transformáciu a kódovanie toku videa. V porovnaní s inými štandardmi kódovania videa, ako sú MPEG-2 a MPEG-4, poskytuje H.256 vyšší kompresný pomer dát, Ďalej poskytuje videozáznamy vo vysokej kvalite a tiež zlepšuje adaptabilitu siete.[49][42][29]

Záznam z kamier vedie do kontrolnej miestnosti, kde prebieha ich analýza, ako aj archivácii záznamov. Video analyzátor je ďalšou časťou systému a zaobstaráva detekciu a následné sledovanie objektov [24]. Ak sa na zábere detekuje objekt, je možné extrahovať aj ďalšie informácie o objekte, ako napríklad identifikácie konkrétnej osoby. Všetky takto extrahované informácie, ktoré sa zistili počas analýzy videa sa uložia do databázy získaných informácii.[24][40]

Databáza tvorí ďalšiu dôležitú časť každého systému. Databáza používaná na ukladanie video záznamov by mala poskytnúť dostatočne veľkú kapacitu na to, aby sa zaistilo nepretržité, dvadsaťštyrihodinové snímanie monitorovanej oblasti, ako aj archiváciu najmenej jednej plnej sady o veľkosti 28 dní a ďalej minimálnu medzipamäť o veľkosti 10%. Nižšie je uvedená všeobecná rovnica, ktorá pomáha pri odhadovaní požadovanej veľkosti databázy na uchovávanie záznamov.[21]

$$ASR = \left(\frac{Size \times fps \times C \times Hours \times 3,600}{1,000,000} \right) \tag{2.1}$$

Kde [21]:

- **ASR** predstavuje približné požiadavky na úložisko v GB
- **Size** určuje veľkosť obrazu v kB
- **fps** vyjadruje počet snímkov za jednu sekundu
- premenná **C** predstavuje počet kamier, ktoré sa nachádzajú v danom systéme
- **Hours** označuje celkový počet hodín, počas ktorých systém zbieral záznamy po dobu trvania 24 hodinovej periódy
- premená **TR** v dňoch určuje dobu počas, ktorej chceme archivovať záznamy

Existuje široká škála databáz, ktoré sa používajú na ukladanie extrahovaných informácii zo záznamov. Typ databázy, ktorú systém používa, odráža spôsob prehľadávania databázy a záznamov v nej. Medzi systémy spracúvajúce záznamy z bezpečnostných kamier je možné naraziť na systémy, ktoré používajú relačné databázy[40] ako aj systémy, ktoré používajú bežné textové súbory.[28]

Poslednou časťou každého systému je užívateľské rozhranie používané na komunikáciu so systémom. Užívateľ používa rozhranie na zadávanie a špecifikovanie udalostí a objektov, ktoré sa pomocou systému snaží užívateľ nájsť. Rozhrania majú rôznu podobu a líšia sa spôsobom použitia ako aj tým komu je systém určený. Bežne sa vyskytujú systémy, ktoré používajú dotazovací jazyk [49], ale aj systémy, ktoré umožňujú užívateľovi kresliť do jednotlivých záberov a tak špecifikovať rôzne udalosti. Taktiež sa využíva aj kombinácia týchto dvoch prístupov.[1]

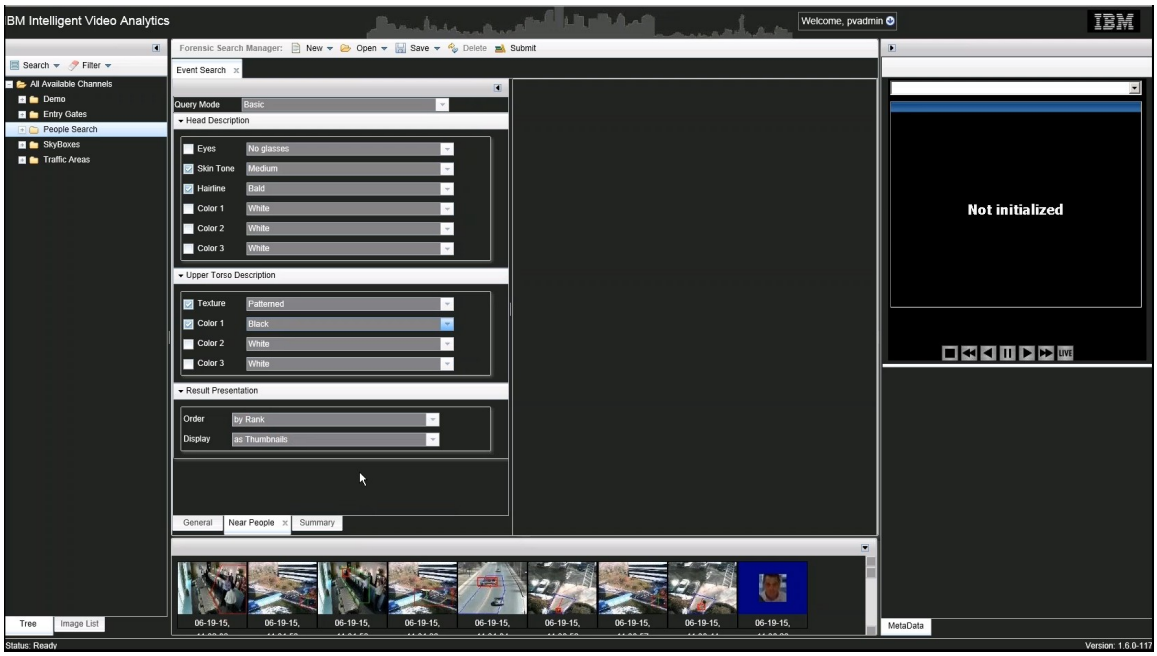
2.3 Predstavenie systémov na prehľadávanie videí

Táto časť nám predstavuje systémy, ktoré sú v súčasnej dobe používané za účelom spracovávania záznamov. Systémy ponúkajú unikátne prístupy k riešeniu problematiky spracovania videí. Líšia sa napríklad prácou s databázou ako aj spôsobom, akým užívateľ zadáva svoje požiadavky na vyhľadávanie.

2.3.1 IBM Intelligent Video Analytics

Systém od IBM[9] pomáha zvyšovať bezpečnosť a efektivitu správy infraštruktúry pomocou premeny digitálneho videa na cenné informácie. Systém generuje bohatú sadu metadát popisujúcich pohybujúce sa objekty v reálnom čase alebo zo zaznamenaného videa. Pri analýze videí v reálnom čase, je potrebné dopredu definovať pravidlá, na ktoré má systém reagovať a v prípade ak preddefinované správanie nastane, vyvolá sa okamžité varovanie o prebiehajúcom incidente. Software dokáže identifikovať narušenie obvodu, opustené objekty, odstránenie objektu, pohyb osôb a aktivitu vozidla.[17]

Pri analyzovaní minulých udalostí na videozáznamoch, systém dokáže vytvoriť index, ktorý je možné prehľadať, analyzovať a korelovať za pár sekúnd. Pri vyhľadávaní je možné špecifikovať rôzne atribúty hľadaných objektov, ako je napríklad špecifikovanie veku hľadanej osoby, taktiež aj špecifické črty osoby, napríklad či hľadaná osoba má vlasy alebo nie. Systém je taktiež schopný rozoznať osobu s okuliarmi od osoby bez okuliarí. Taktiež je možné definovať hornú časť tela, ktorá slúži na upresnenie oblečenia, ktoré hľadaná osoba nosí. Spôsob zadávania týchto atribútov je možné vidieť na obrázku 2.3.[10]

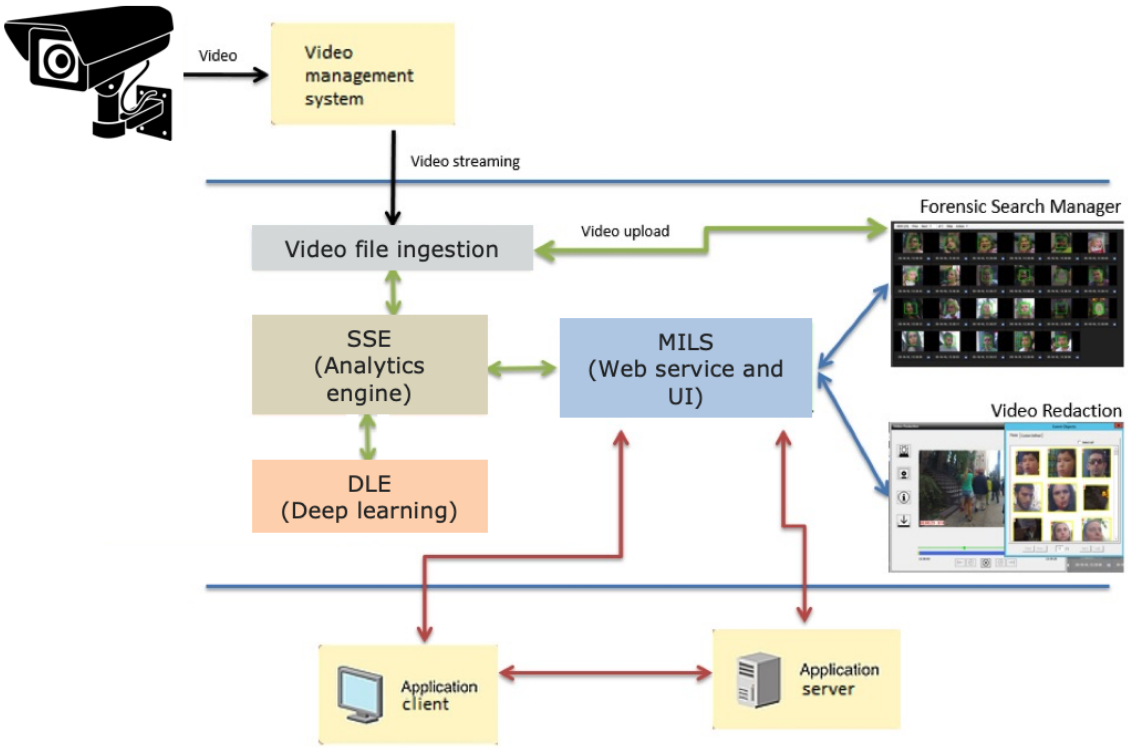


Obr. 2.2: Príklad programu od IBM. Prevzaté z [10]

Analytický rámec produktu IBM sa skladá z middleware pre veľkoplošné sledovanie (MILS), inteligentného sledovacieho modulu (SSE) a modulu využívajúceho technológiu umelých neurónových sietí (DLE).

Video dáta a metaúdaje pre výstrahy v reálnom čase a zaznamenané videá, prechádzajú systémom v tomto poradí [13]:

1. Video sa buď nahrá ako súbor, ktorý sa následne uloží, alebo sa video streamuje zo systému na správu videa (VMS) a až následne sa uloží
2. SSE prijíma video
3. SSE používa DLE na detekciu objektov a analýzu atribútov
4. Následne SSE vygeneruje metadáta a odošle ich na server
5. MILS indexuje a ukladá metaúdaje
6. Klient operátora buď prehľadá metadáta, alebo príjme výstrahy od MILS
7. Pomocou klienta operátora je možné sledovať zaznamenané video

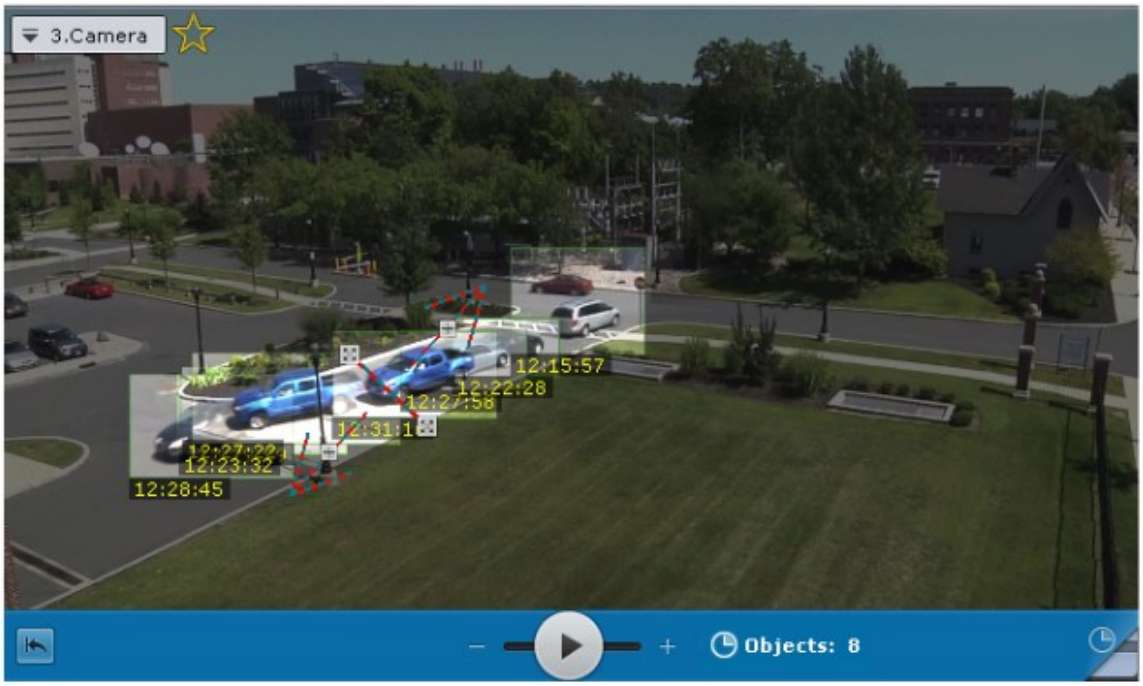


Obr. 2.3: Architektúra systému od IBM. Prevzaté z [13]

2.3.2 Axxonsoft

Software spoločnosti Axxonsoft[1] ponúka detekciu udalostí v reálnom čase s možnosťou zasielania notifikácii ako aj možnosť spracovávať záznamy v režime off-line. Pri spracovávaní záberov v reálnom čase je možné detekovať, prekročenie dopredu definovaných liniek v monitorovanej oblasti, ako aj vstup do monitorovanej zóny, rôzne podozrivé aktivity v tejto zóne, ako je napríklad opustenie nejakého predmetu. Pri definovaní zóny je dôležité jasne

vyznačiť všetky vstupné a výstupné časti oblasti, ktoré je možné zakresliť priamo do snímku videa, pričom pri každej takejto čiare je dôležité jasne určiť smer, ktorý určuje akým spôsobom je možné danú čiaru využívať. Tj. či ide o vstup alebo výstup. Systém podobným spôsobom ponúka aj zúžené vyhľadávanie na jednoznačne vyznačené oblasti, na ktorých je možné určovať, koľko ľudí sa v danej oblasti nachádza, alebo časti videa, v ktorých sa objekt pohyboval z jednej vyznačenej oblasti do druhej [3]. Toto je možné vidieť na obrázku 2.4.



Obr. 2.4: Príklad programu od Axxonsoft. Prevzaté z [3]

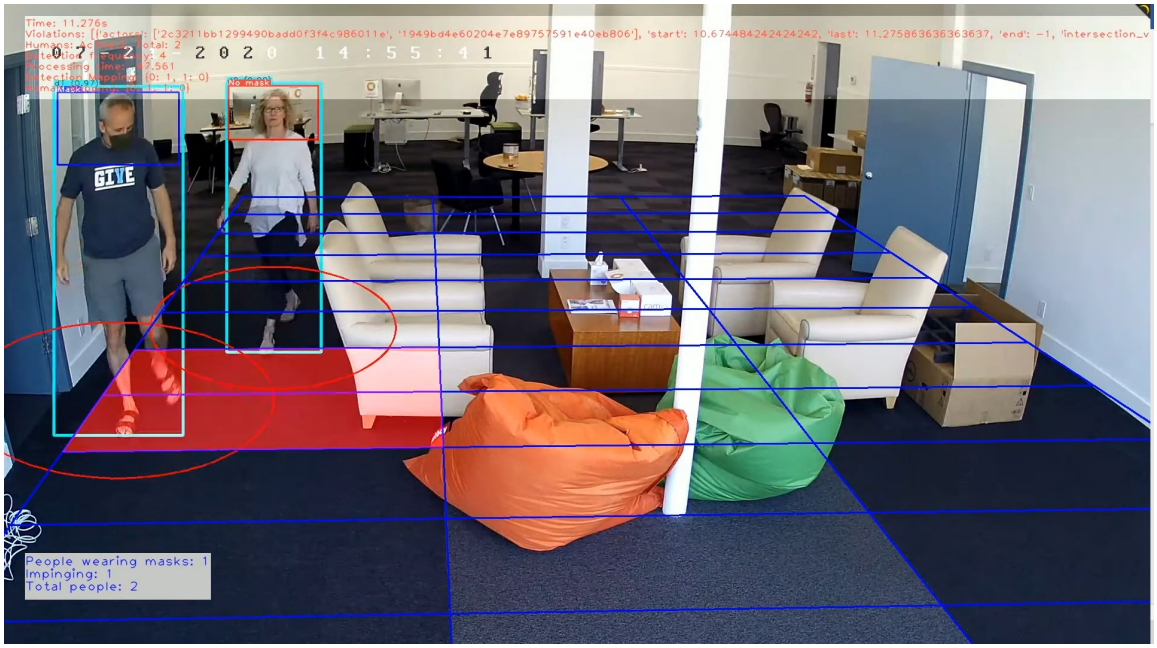
Za zmienku taktiež stojí aj funkcia umožňujúca užívateľovi zobraziť všetky objekty, ktoré spĺňali zadané kritéria a boli zaznamenané v rôznych časoch. Pri kliknutí na konkrétny objekt sa zobrazí časť videa, na ktorej sa objekt nachádzal. Spojenie tejto funkcie spolu s definíciou vstupného a výstupného miesta je možné vidieť na obrázku 2.4.[3]

2.3.3 Camio

Produkt spoločnosti Camio [14] sa odlišuje od dvoch už zmienených systémov - dôvodom je jeho užívateľské rozhranie a spôsob fungovania. Užívatelia používajú svoje bežné kamery, ktoré posielajú zachytený obsah do claudového úložiska [15]. Software dokáže premieňať štandardné 2D kamery na 3D senzory bez nutnosti inštalácie špecializovaných zariadení pre infračervené alebo stereoskopické videnie. Camio túto premenu zabezpečuje tým, že segmentuje a sleduje ľudí pri pohybe po 3D mriežke, ktorá definuje oblasti pred a za každým vchodom. [15]

Software je známy aj vďaka svojej schopnosti detekovať, či do stráženej oblasti vstúpi osoba bez toho aby na vstupe použila bezpečnostnú kartu, ktorá je potrebná na otvorenie dverí. K tomu používa vytvorenú 3D mriežku ako aj funkciu, ktorá sleduje, či sa počet načítaných kariet zhoduje s množstvom ľudí, ktorí vstúpili do budovy [19]. Ak systém zistí, že vstúpilo viac ľudí než bolo použitých kariet, pošle upozornenie vybranej osobe, ktorá situáciu skontroluje. V roku 2020 sa spoločnosť zamerala na detekciu bezpečných

rozostupov a tiež rozpoznávanie masiek na tvárach ľudí, čím sa snažia zabrániť šíreniu ochorenia covid19 na pracoviskách, alebo na miestach s vysokou frekvenciou výskytu ľudí. [16]



Obr. 2.5: Príklad programu od spoločnosti Camio. Prevzaté z [2]

Kapitola 3

Počítačové videnie a homografia

Odhad homografie je dôležitým krokom v mnohých algoritmoch počítačového videnia. Video zobrazuje dvojrozmerný obraz, ktorý však zachytáva trojrozmerný priestor. Pri perspektívnej projekcii dochádza k tomu, že všetky čiary sa nakoniec zbiehajú v jednom bode. V tejto kapitole je vysvetlené, ako funguje perspektívna transformácia a ako sa používa.

3.1 Počítačové videnie

Počítačové videnie je disciplína, ktorá sa snaží pomocou technických prostriedkov napodobniť ľudské videnie. Ľudia vnímajú svet ako trojrozmernú rovinu, vďaka čomu jednoducho vnímajú vzdialenosti, tvary objektov a dokážu pracovať s osvetlením a tieňmi. Počítačové videnie predstavuje súbor rôznych techník, pomocou ktorých sa snažíme napodobniť ľudské videnie v počítačovom svete. Jednou z techník, ktorá sa používa s týmto cieľom je transformácia obrazu. Na riešenie problému neskresleného premietania na plochu je vhodné použiť transformáciu pomocou homografie obrazu do roviny.

Počítačové videnie má široké použitie. Medzi príklady použitia patrí napríklad detekcia slepého uhla v aute, taktiež analýza okolia automobilov pri ich pohybe, ktoré tým napríklad detekujú prekročenie čiar na vozovke, ako aj pri analýze videozáznamov, pomocou ktorého sa ľahšie určujú vzdialenosti medzi autami. Tento spôsob taktiež využívajú kamery na zisťovanie rýchlosti a vzdialenosti medzi vozidlami, ako je popísané v práci od Markéty Dubskej viz [34].

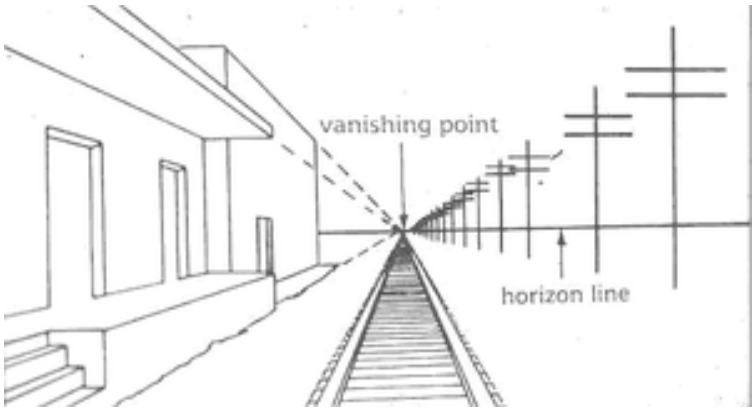
3.1.1 Čo je homografia?

Homografia, homogénna reprezentácia bodu, ktorý leží v rovine projektovanej roviny, sa dá podľa Dubrofského práce [33] definovať, ako trojrozmerný vektoro $v = (x_1, x_2, x_3)$. Tento vektor reprezentuje dvojrozmerný bod označený ako (x,y), kde x je podiel medzi x_1 a x_3, y je možné získať podielom x_2 a x_3.

Homografia je bijektívné zobrazenie bodov a priamok v projektovanej rovine, pričom mapovanie roviny R do seba sama, je projektivita len ak existuje *non-singular 3x3* matica M, pre každý bod z roviny R, ktorý je reprezentovaný vektorom v, platí, že tento vektor je mapovaný ako Mx. Takže, ak chceme vypočítať homografiu, ktorá bude mapovať každý bod x_i do jemu zodpovedajúcemu x_i‘, je potrebné vypočítať jeho maticu homografie o veľkosti *3x3*.

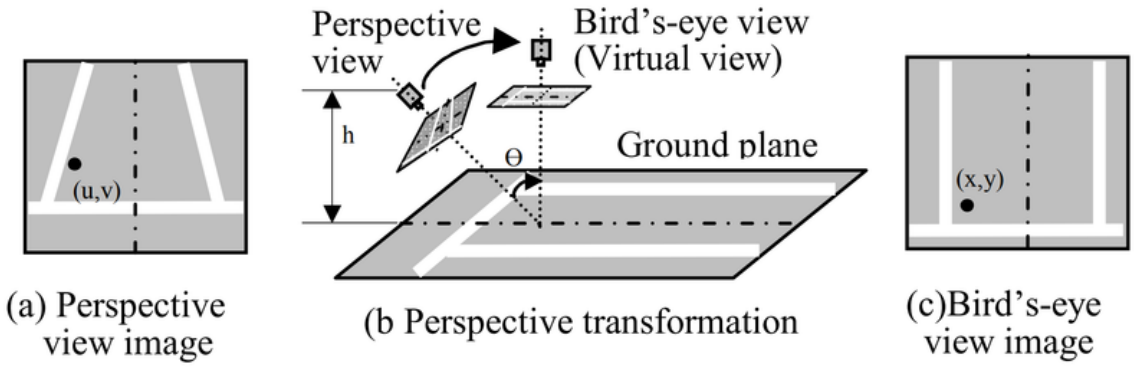
3.2 Perspektívna projekcia

Projektívna transformácia je koncept používaný v projektívnej geometrii na popísanie toho, ako sa sada geometrických objektov mapuje na inú sadu geometrický objektov v nejakom projektívnom priestore [65]. Perspektívna projekcia sa taktiež označuje ako bodová projekcia a to z toho dôvodu, že sa všetky línie obrazu nakoniec stretávajú v jednom bode, anglicky označovaného ako vanishing point. Cieľom perspektívnej projekcie teda je premietnutie objektov scény do dvojrozmerného záberu kamery tak, aby sa čo najlepšie zobrazil aj tretí rozmer. [48]



Obr. 3.1: Vanishing point v priestore. Prevzaté z [12]

Jedným z príkladov použitia perspektívnej transformácie je práca s kamerou na vozidle, ktorá sa dá aplikovať na akúkoľvek inú kameru, ktorú používame na získavanie záznamov, ktoré ďalej analyzujú systémy. Kamera na vozidle má vo všeobecnosti výrazný perspektívny efekt, ako je možné vidieť na obrázku 3.2 časti a. Z dôvodu efektu perspektívy vodič nemôže správne vnímať vzdialenosť na monitore. Pre systém sa pokročilé spracovanie obrazu a analýza stáva zložitou úlohou. V dôsledku toho je možné vidieť na obrázku 3.2 časti b, že perspektívna transformácia je nevyhnutná. V časti c obrázku 3.2, je možné vidieť pôvodný obrázok prenesený do vtáčej perspektívy a teda ide o perspektívnu projekciu pôvodného obrázku. [51]



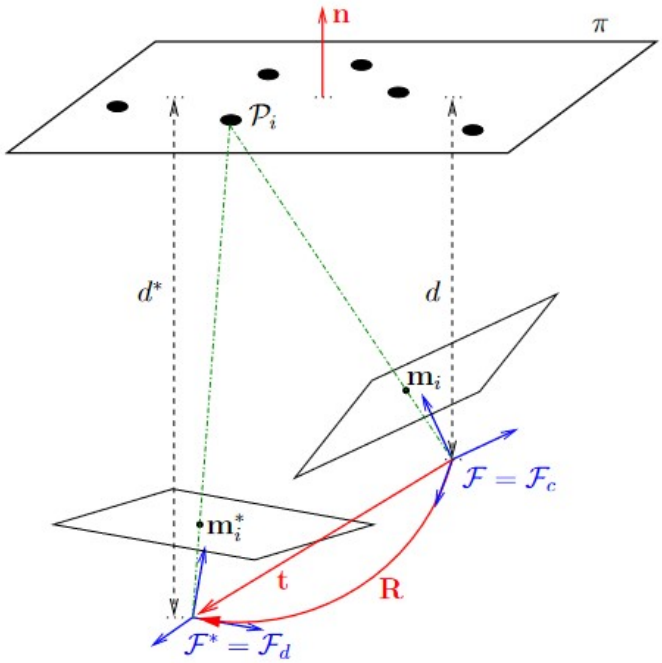
Obr. 3.2: Ilustrácia perspektívnej transformácie na parkovisku. Prevzaté z [51]

Vzťah medzi pixelmi (x, y) snímku z vtáčej perspektívy a pixelov (u, v) pôvodného obrázku je možné formulovať pomocou homografickej matice o veľkosti 3x3, kde x = x´/w´ a y = y´/w´ ako na rovnici 3.1 [51].

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.1}$$

Ak sa vezmú do úvahy dva rôzne zábery kamery, ktoré sú označené ako požadovaný (F*) a aktuálny (F), ktoré je možné vidieť na obrázku 3.3, predpokladá sa, že absolútny rámec sa zhoduje s rámcom referenčnej kamery F*. Za predpokladu, že kamera pozoruje rovinný objekt zo sady skladajúcich sa z n-tých 3D bodov, plochy P s karteziánskymi súradnicami [52]:

$$P = (X, Y, Z) \tag{3.2}$$



Obr. 3.3: Požadované a aktuálne zábery kamery a príslušné notácie. Prevzaté z [52]

Tieto body môžu buď predstavovať náš požadovaný rám kamery alebo ten aktuálny. Homogénna transformačná matica prevádzajúca tie súradnice 3D bodu z požadovaného rámca na aktuálny je [52]:

$${}^{c}T_d = \begin{bmatrix} Rd & t_d \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

Kde ${}^{c}\mathrm{R_d}$ je rotačná matica a prekladový vektor je ${}^{c}\mathrm{t_d}$. Kamera sníma obraz daného objektu z požadovanej a aj aktuálnej konfigurácie. Toto zoskupenie zahŕňa premietanie 3D bodov do roviny, čo zabezpečí aby mali rovnakú hĺbku od zodpovedajúcej kamery. Normalizované projektívne súradnice každého bodu pre požadovaný a aktuálny rámec kamery sa potom budú označovať ako [52]:

$$m* = (x*, y*, 1) =^{d} m; \tag{3.4}$$

$$m = (x, y, 1) =^{c} m \tag{3.5}$$

Nakoniec sa pomocou následujúcej transformácie získavajú homogénne koordináty každého bodu obrazu, p = (u, v, 1), vyjadrené v pixeloch [52]:

$$p = Km \tag{3.6}$$

Jednou z možností získania matice perspektívnej transformácie je použitie knižnice OpenCV, ktorá ponúka rôzne funkcie vykonávajúce geometrické transformácie 2D snímkov. Tieto funkcie nemenia obsah obrázku, ale deformujú mriežku pixelov a mapujú túto deformovanú mriežku na cieľový snímok. Na perspektívnu transformáciu potrebujeme transformačnú maticu 3x3. Na nájdenie tejto transformačnej matice potrebujeme štyri body na vstupnom obrázku a zodpovedajúce body na výstupnom zábere. Medzi týmito štyrmi bodmi by tri z nich nemali byť kolineárne. Následne môže byť nájdená perspektívna matica pomocou funkcie“cv2.getPerspectiveTransform“. [6]

Kapitola 4

Detekcia objektov

Detekcia objektov tvorí základ každého systému na analýzu videí alebo obrázkov. Ide o jeden z najzásadnejších a najnáročnejších problémov počítačového videnia. V tejto kapitole je vysvetlený základný princíp detekcie objektov a taktiež sú tu predstavené základné prístupy k riešeniu tohto problému.

4.1 Úvod do detekcie objektov

Detekcia objektov sa zaoberá detekciou inštancii vizuálnych objektov konkrétnej triedy a zároveň lokalizuje tieto objekty v digitálnych obrazoch [66]. Medzi základné triedy detekovaných objektov sa dá zaradiť druh zvieraťa, človek alebo automobil. Cieľom odvetvia zaoberajúceho sa detekciou objektov je vyvinúť výpočtové modely a techniky, ktoré poskytujú jednu z najzákladnejších informácii potrebných pre aplikácie používajúce počítačové videnie. Tým je zodpovedanie otázky: aké objekty sa nachádzajú na zázname a aká je poloha tohto objektu? [66]

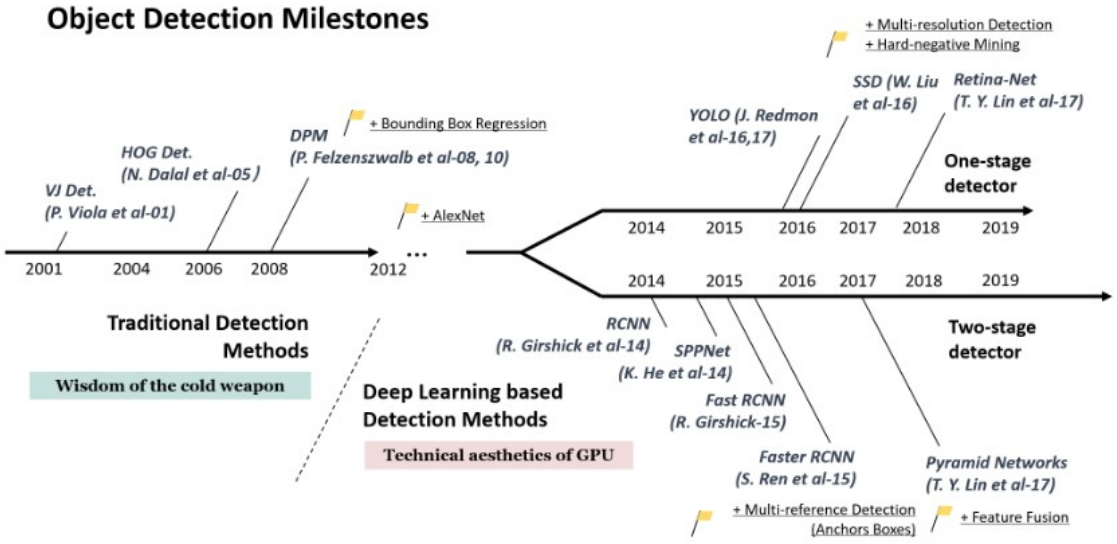
Image classification analyzuje numerické vlastnosti rôznych obrazových prvkov a organizuje údaje do predefinovaných kategórií [37]. Vo všeobecnosti sa dá rozdeliť do dvoch kategórii [8]:

- Klasifikácia založená na pixeloch (*Pixel-based classification*)
- Klasifikácia založená na objektoch (*Object based classification*)

Pixely sú základné stavebné jednotky záberov videa a analýza týchto pixelov je hlavným spôsobom klasifikácie obrázkov. Klasifikačné algoritmy môžu použiť spektrálne informácie v rámci jednotlivých pixelov, alebo preskúmať priestorové informácie, teda blízke pixely spolu so spektrálnymi informáciami [8]. Metódy klasifikácie založené na pixeloch využívajú len spektrálne informácie, medzi ktoré patrí napríklad intenzita pixelov. Na druhej strane metódy založené na klasifikácii objektov zohľadňujú spektrálne, ako aj priestorové informácie. [8]

Podľa prístupu k detekcii objektov môžeme systémy rozdeliť do dvoch prúdov – všeobecná detekcia objektov a aplikácie na detekciu [66]. Všeobecná detekcia preskúmava metódy detekcie rôznych typov objektov v rámci jedného snímku, čím simuluje ľudské pozorovanie a vnímanie prostredia. Naopak aplikácie na detekciu sa zameriavajú na detekciu nie plne špecifikovaných scenárov, ako je napríklad detekcia tváre, používanú v moderných telefónoch, či na detekciu textu a ďalšie. [66]

Rozvoj *deep learning* techník sa dotkol aj detekcie objektov a zvýšil záujem o túto oblasť výskumu. Detekcia objektov sa v súčasnosti používa v mnohých aplikáciách a odvetviach, medzi ktoré môžeme zaradiť autonómne riadenie, pohyb robotov, ale aj oblasť patológie a analyzovania záznamov bezpečnostných kamier [23][66]. Obrázok 4.1 zobrazuje vývoj tejto oblasti v chronologickom poradí. V nasledujúcich podkapitolách sú vysvetlené najvýznamnejšie modely na detekciu objektov.



Obr. 4.1: Historický vývoj algoritmov na detekciu objektov. Prevzaté z [66]

4.2 Tradičné metódy

Väčšina prvých algoritmov na detekciu objektov bola postavená na ručne navrhnutých funkciách. Bolo tomu tak kvôli nedostatku efektívnej obrazovej reprezentácie, kvôli čomu ľudia nemali inú možnosť, len navrhnúť sofistikované funkcie a rôzne zrýchľovacie procesy, aby naplno využili obmedzené výpočtové zdroje.

4.2.1 Viola Jones detectors

V roku 2001 sa P. Viole a M. Jonesovi prvýkrát podarila detekcia ľudských tvári v reálnom čase bez toho aby boli použité nejaké obmedzenia, ako napríklad segmentácia farby pleti [66]. Najvýraznejším rozdielom tohto systému bolo, že bol schopný detekovať tváre mimoriadne rýchlo. Ak pracoval na obrázkoch o veľkosti 384 x 288 pixelov, tváre boli detekované rýchlosťou pätnásť záberov za sekundu na konvenčnom procesore Intel Pentium III s frekvenciou 700 MHz [61].

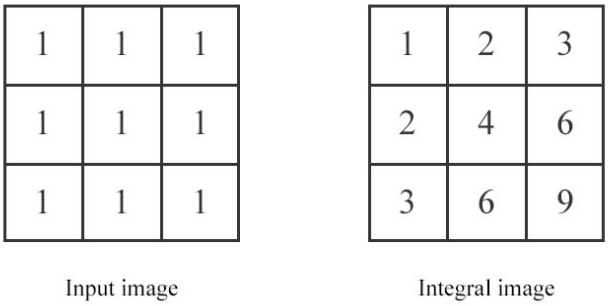
V tej istej dobe používali ostatné systémy za účelom dosiahnutia vysokých frekvencii záberov, pomocné informácie, ako boli napríklad rozdiely obrázkov vo video sekvenciách alebo farba pixelu pri farebných obrázkoch. Systém od Violy a Jonesa zaisťoval vysoké obnovovacie frekvencie iba s informáciami z jedného obrázku, ktorý navyše bol v šedých odtieňoch [61].

Viola Jones detektor používa jeden z najpriamejších spôsobov detekcie a to takzvané posuvné okná, čo znamená, že je nutné prejsť všetkými možnými umiestneniami a mierkami

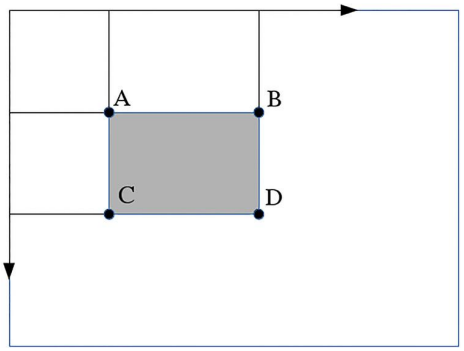
v obraze aby sa zistilo, či nejaké okno obsahuje ľudskú tvár [66]. Detektor Viola Jones zrýchlil proces tejto detekcie pomocou začlenenia troch techník [66][61]:

- Technika integrovaného obrazu (*integral image*)
- Technika výberu vlastnosti (*feature selection*)
- Techniky detekčnej kaskády (*detection cascades*)

Technika integrovaného obrazu je výpočtovou metódou na urýchlenie procesu filtrovania alebo konvolúcie políčka, algoritmus Viola Jones používa Haarovu vlnku ako znakovú reprezentáciu obrazu [66] [61]. Integrovaný obraz je možné vypočítať z obrázka pomocou niekoľkých operácii na každom jednom pixeli. Tento obraz je veľmi podobný tabuľke stredných oblastí používanej v počítačovej grafike na mapovanie textúr [61]. Pri technike výberu vlastností bol použitý algoritmus AdaBoost, ktorý vyberá malé množstvo dôležitých funkcii, ktoré sú užitočné na detekciu tváre [61]. V algoritme bola zavedená viacstupňová detekčná paradigma, tiež známa pod názvom detekčná kaskáda. Využíva sa s cieľom zníženia výpočtovej réžie algoritmu, tým že sa vynaložili menšie výpočty na okná, ktoré sú na pozadí, ale väčšie výpočty na okná, ktoré potencionálne obsahovali tvár. [61][66]



Obr. 4.2: Premena vstupného obrazu na integrovaný obraz. Prevzaté z [11]



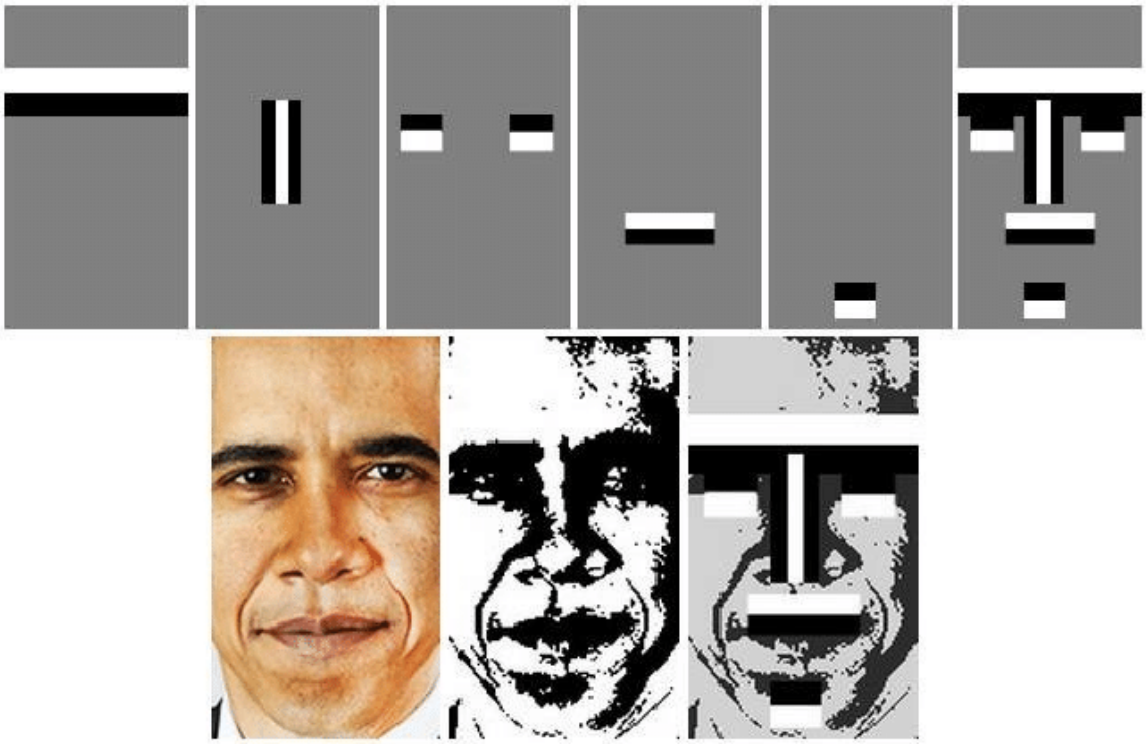
Obr. 4.3: Výpočet sumy. Prevzaté z [18]

Prvým krokom algoritmu Viola Jones je premeniť vstupný obrázok na integrovaný obraz. Tento proces je vytvorený tak, že každý pixel je rovný sume všetkých pixelov, ktoré sú nad ním a na ľavej strane od neho [44]. Tento proces je demonštrovaný na obrázku 4.2.

Tento proces umožňuje následne spočítať sumu všetkých pixelov vo vnútri vybraného štvorca za použitia iba štyroch hodnôt. Tieto hodnoty sú pixely integrovaného obrazu, ktoré sa zhodujú okrajmi so štvorcami zo vstupného obrázka. Príklad je možné vidieť na obrázku 4.3 Výsledná suma štvorca sa vypočíta nasledovný spôsobom [44]:

$$Sum = D - (B + C) + A \tag{4.1}$$

Následne algoritmus Viola Jones analyzuje časti obrazu, skladajúce sa z dvoch alebo viacerých takto získaných štvorcov. Rôzne vyjadrenie dôležitých vlastností ľudskej tváre je možné vidieť na obrázku 4.4. Každá vlastnosť je reprezentovaná jednou hodnotou, ktorá je získaná odpočítaním sumy bieleho štvorca od sumy čierneho štvorca [44].



Obr. 4.4: Príklad fungovania algoritmu Viola Jones[5]

4.2.2 Detekcia objektov založená na histograme orientovaných gradientov

Táto metóda bola pôvodne navrhnutá v roku 2005 N. Dalalom a B. Triggsom [31]. Histogram orientovaných gradientov, ďalej už len HOG, výrazne zlepšil SIFT (*scale-invariant feature transform*). Na vyváženie *feature invariance*, ako je napríklad mierka alebo osvetlenie, a tiež na vyváženie nelinearity, pri rozlišovaní rôznych kategórii objektov je deskriptor HOG navrhnutý tak, aby sa počítalo na hustej mriežke rovnomerne rozmiestnených mriežok buniek [66]. HOG je v súčasnosti základom mnohých detektorov objektov, ale primárne bol vytvorený na detekciu chodcov.

HOG deskriptor predstavuje obrazový vektor pomocou RGB farieb pixelov. Vektor je následne odoslaný klasifikátorovi, ktorý rozhodne, či obrázok obsahuje objekt alebo nie.

Dôležitou časťou algoritmu je rozdelenie obrazu na 8 x 8 blokov pomocou histogramu orientovaných gradientov [31]. Pre každý pixel v bloku sa následne vypočítajú dve hodnoty – smer a veľkosť. Na výpočet týchto hodnôt je potrebné porozumieť pojmu pixelový gradient.

Vektor gradientu obrazu je definovaný ako metrika pre každý jednotlivý pixel, ktorý obsahuje zmeny farieb pixelov v osi x aj v osi y. Definícia je v súlade s gradientom spojitej funkcie viacerých premenných, ktorá je vektorom parciálnych derivácii všetkých premenných. Predpokladajme, že funkcia f(x, y) zaznamená farbu pixelu v mieste (x, y), vektor gradientu pixelu je definovaný ako [62]:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{bmatrix} \tag{4.2}$$

Kde [62]:

- ∂f / ∂x je parciálna derivácia v smere x, ktorá je vypočítaná ako rozdiel farieb medzi susednými pixelmi naľavo a napravo od cieľa, f(x+1, y) - f(x-1, y)
- ∂f / ∂y je parciálna derivácia v smere y osi, meraná ako farebný rozdiel medzi susednými pixelmi nad a pod cieľom, f(x, y+1) - f(x, y-1)

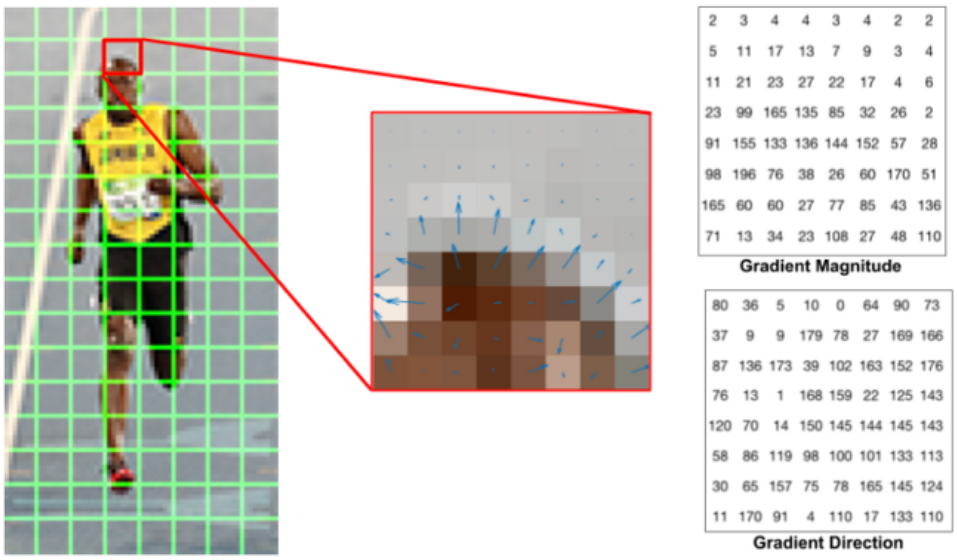
Dva dôležité atribúty gradientu obrazu sú [62]:

- Magnitúda:

$$g = \sqrt{g_x^2 + g_y^2} \tag{4.3}$$

- Smer: arkustangens pomeru medzi parciálnymi deriváciami v dvoch smeroch

$$\theta = \arctan(g_y / g_x) \tag{4.4}$$



Obr. 4.5: Obrázok s príslušnými gradientovými veľkosťami a smermi[53]

Ďalším krokom je zaistenie základnej nelinearity deskriptora. Každý pixel počíta takzvaný vážený hlas, ktorý sa zhromažďuje do *orientation bin* cez miestne priestorové oblasti, ktoré označujeme bunkami. Bunky môžu byť buď obdĺžnikové alebo radiálne. [31]

Následne je blok reprezentovaný jedným vektorom, ktorý je vytvorený súčtom magnitúdy všetkých vektorov s určitým smerom. Smer pixelov je v rozmedziach od 0 po 160 stupňov v prírastkoch po 20-ich stupňoch [31][53]. Na obrázku 4.5 je príklad ako môže vyzerať obrázok s príslušnými gradientovými veľkosťami a smermi. Je možné vidieť, že šípky sa zväčšujú podľa magnitúdy.

Poslednou časťou algoritmu je podporný vektorový stroj (SVM), ktorý je trénovaný tak, že vezme získaný vektor a SVM rozhodne či je alebo nie je objekt v obraze [31].

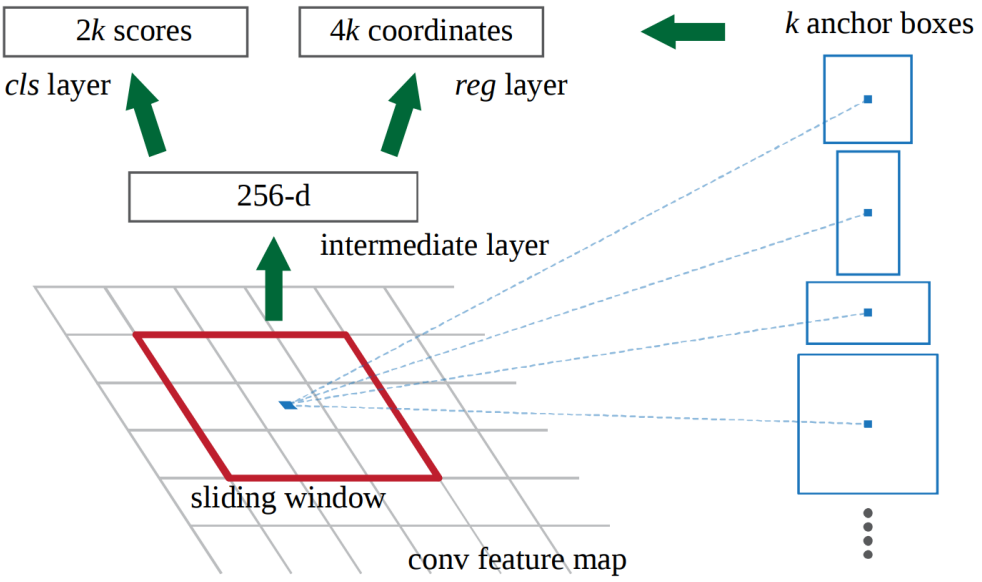
4.3 Metódy založené na hlbokom učení (*deep learning*)

V roku 2012 svet zažil nebývalý záujem o konvolučné neurónové siete [22]. Pretože hlboká konvolučná sieť je schopná sa naučiť robustné a vysoko postavené reprezentácie animácii, bolo prirodzeným krokom skúmať jej využitie pri detekcii objektov. Hlboké učenie umožňuje rozdeliť detekciu objektov na dva významné smery a to na dvojstupňové detekcie a jednostupňové detekcie. V tejto podkapitole sa zameriame na dvojstupňovú detekciu (Faster RCNN) a jednostupňovú detekciu (YOLO).

4.3.1 Faster RCNN

V roku 2015 S. Renet v krátkej dobe po Fast RCNN navrhol Faster RCNN detektor [57]. Faster RCNN je prvý end-to-end detektor implementovaný pomocou hlbokého učenia, ako aj prvý detektor, ktorý bol schopný detekcie skoro v reálnom čase [66]. Hlavným prínosom tohto algoritmu bolo zavedenie *Region Proposal Network* (RPN). Od R-CNN po Faster RCNN bola postupne väčšina jednotlivých blokov systému na detekciu objektov integrovaná do zjednoteného komplexného *learning frameworku*.

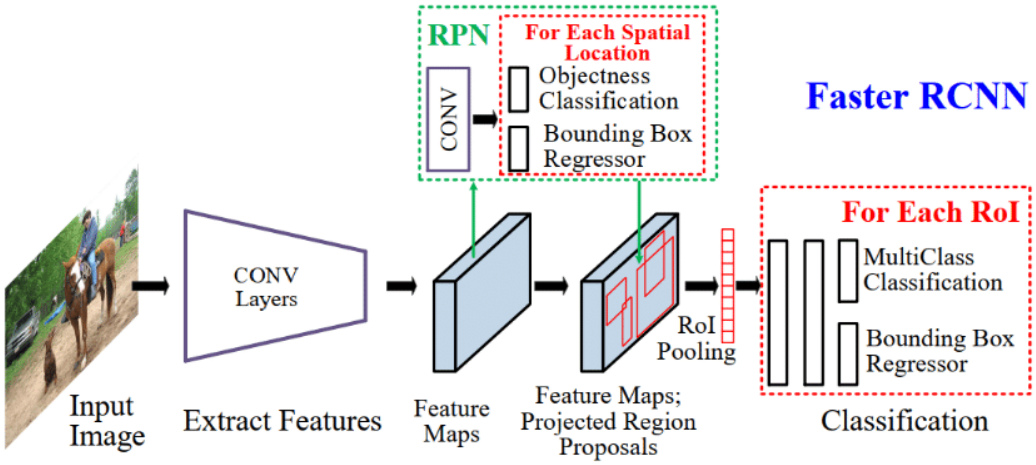
Systém sa skladá z dvoch modulov. Prvým modulom je hlboká plne konvolučná sieť, ktorá navrhuje potencionálne regióny, druhým modulom je detektor Fast R-CNN, ktorý používa navrhnuté potencionálne regióny [57].



Obr. 4.6: Príklad fungovania sliding window u Faster RCNN. Prevzaté z[57]

RPN na vstupe berie obrázok o ľubovoľnej veľkosti a na výstup posiela sadu obdĺžnikových okien, ktoré obsahujú potencionálny objekt. Každý z týchto boxov obsahuje aj skóre – *objectness score*, ktoré určuje členstvo k niektorej z tried objektov oproti pozadiu [57]. Na vygenerovanie regiónov posúvame malú sieť po *convolutionnal feature map*. Sieť berie ako vstup priestorové okno o veľkosti n x n zo vstupnej *convolutionnal feature* mapy. Táto funkcia ďalej vedie do dvoch plne prepojených vrstiev. Prvou vrstvou je *box-regresion* vrstva, na obrázku 4.6 označená ako *reg layer*, a druhá vrstva je pomenovaná ako *box-classification* vrstva, na obrázku 4.6 označená ako *cls layer*. [57]

Po RPN získavame navrhnuté oblasti, ktoré majú rôzne veľkosti, čo znamená, že majú rôzne veľké *CNN features* mapy. Keďže nie je jednoduché vytvoriť efektívnu štruktúru pre prácu na *features* o rôznych veľkostiach, tak sa používa *region of interest pooling* (ROI pooling) [57][36], ktorý zmenšuje mapy na rovnakú veľkosť, ktorá je následné poslaná do detektora.



Obr. 4.7: Architektúra Faster RCNN. Prevzaté z [38]

4.3.2 YOLO

YOLO bola navrhnutá v roku 2015 Josephom Redmonom, bol to prvý jednostupňový detektor v ére hlbokého učenia [56]. Systém je veľmi rýchly– v základe dokáže spracovať 45 obrázkov za sekundu a najrýchlejšie dokáže spracovať obraz rýchlosťou 150 obrázkov za sekundu. To umožňuje systému YOLO spracovávať streamované videá v reálnom čase s latenciou kratšou než 25 milisekúnd. [56]

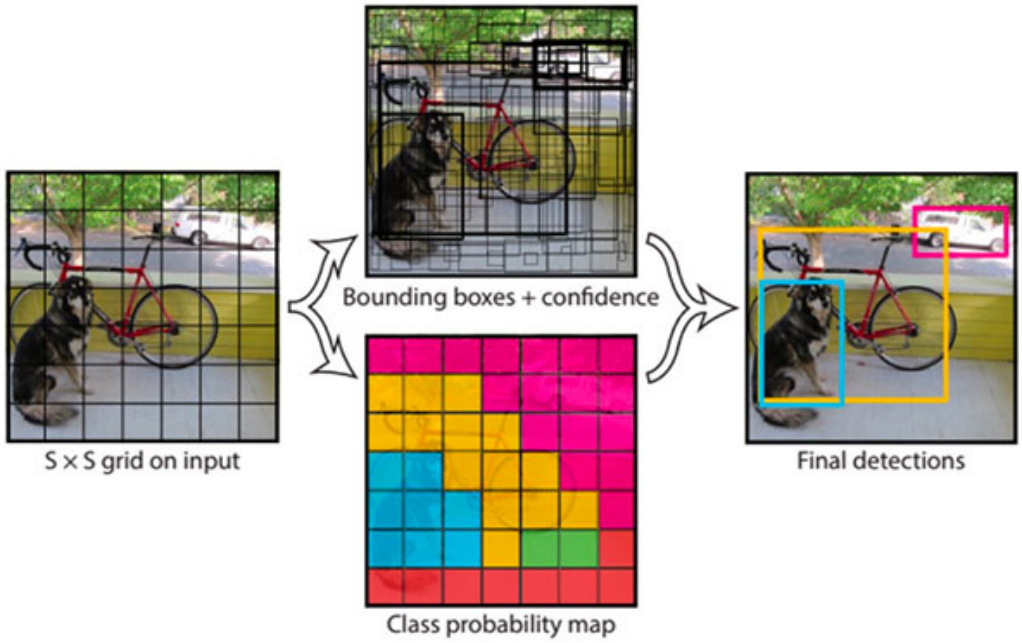
Od metódy Faster RCNN sa líši tým, že nepoužíva rovnaký detekčný prístup, ale na celý obraz používa jednu neurónovú sieť. Sieť rozdeľuje obraz na oblasti a predpovedá *bounding* boxy a pravdepodobnosti pre každý región súčasne. Avšak YOLO zaostáva v presnosti, keďže má problém detekovať menšie objekty. [56]

Prvým krokom systému je rozdelenie vstupného obrazu na mriežku o rozmeroch S *x* S. Ak sa stred objektu nachádza vo vnútri bunky, táto bunka je zodpovedná za detekciu tohto objektu. Každá bunka mriežky predpovedá *bounding* boxy a *confidence score* daných boxov [56]. *Confidence* skóre reflektuje ako moc si je model istý, že daný box obsahuje objekt a tiež s akou presnosťou bol box predpovedaný. Formálne je *confidence* skóre definované ako vzorec 4.5. Ak bunka neobsahuje žiaden objekt, skóre je rovné nula. V opačnom prípade je cieľom aby sa skóre rovnalo IOU (*intersection over union*) medzi predpovedaným boxom a *ground truth* [56].

$$Pr(Object) * IOU_{pred}^{truth} \tag{4.5}$$

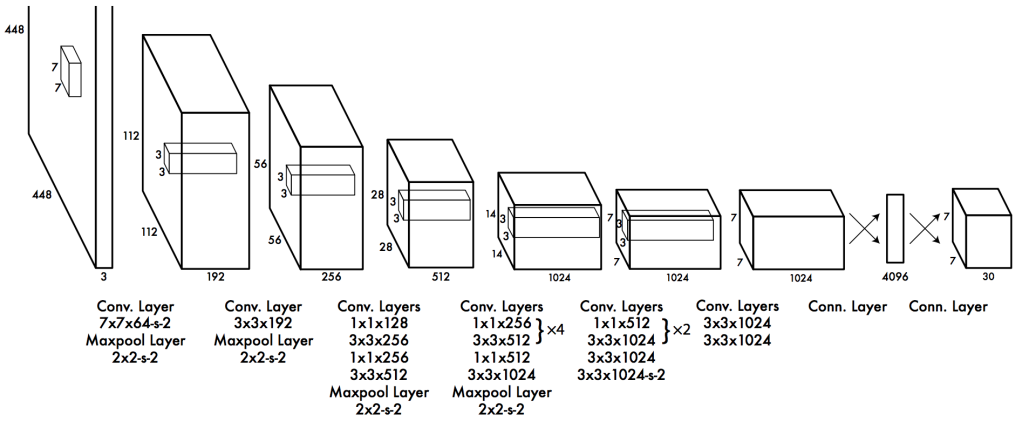
Bounding box pozostáva z 5 údajov: x, y, w, h a *confidence score*. Súradnice (x,y) reprezentujú stred boxu, w reprezentuje šírku boxu, h predstavuje výšku. Každá bunka taktiež predpovedá pravdepodobnosti tried, vzorec 4.6, pričom sa predpovedá len jedna skupina pravdepodobnosti triedy na bunku mriežky bez ohľadu na to, koľko obsahuje *bounding* boxov. [56]

$$Pr(Class_i|Object) \tag{4.6}$$



Obr. 4.8: Kroky potrebné na detekovanie objektov v YOLO. Prevzaté z [56]

Architektúra siete bola inšpirovaná modelom GoogLeNet, určeným na klasifikáciu obrázkov viz. [59]. Sieť modelu YOLO sa skladá z 24 konvolučných vrstiev nasledovaných dvomi úplne prepojenými vrstvami. Narozdiel od GoogLeNet však nepoužíva inception moduly, ale používa 1 x 1 redukčné vrstvy, po ktorých následujú 3 x 3 konvolučné vrstvy, podobne ako v práci od Lin a kolektívu [50]. Celú architektúru siete je možné vidieť na obrázku 4.9. [56]



Obr. 4.9: Architektúra systému YOLO [56]

Kapitola 5

Metódy na porovnávanie kvality algoritmov

Táto časť práce predstavuje rôzne metriky, ktoré sa používajú na zisťovanie kvality a úspešnosti algoritmov. Viac o jednotlivých nástrojoch na porovnávanie v práci [26].

Hodnotenie algoritmov na detekciu objektov

Pri algoritmoch na detekovanie objektov získavame na výstup súradnice daného objektu v zábere analyzovaného videa, ktoré sú vyjadrené pomocou *bounding boxov*, ktoré majú rôzne podoby. Existujú napríklad v tvare polygónu ako aj štvorca, ktorý je definovaný za pomoci súradníc jeho dvoch protiľahlých rohov.

Pri hodnotení detekčných algoritmov je dôležité určiť IOU (*Intersection Over Union*), ktorý sa vypočíta podľa vzorca na 5.1. Kde oblasť prekrytia (*area of overlap*) reprezentuje prekrytie medzi *bounding boxom* získaného z algoritmu a *bounding box* získaný z tréningových dát. Oblasť únie (*area of union*) je zjednotenou oblasťou oboch bounding boxov [54].

$$IOU = \frac{area\,of\,overlap}{area\,of\,union} \tag{5.1}$$

5.1 Matica zámen

Najčastejšie sa na analýzu kvality algoritmov používa takzvaná matica zámen, v origináli *confusion matrix*, z ktorej sa vyvodzujú ďalšie vzťahy, ktoré sa používajú na posudzovanie presnosti. Matica sa skladá zo štyroch premenných:

- TP (*true positive*) vyjadruje počet správnych predpovedí, že príklad je pozitívny, čo znamená, že trieda je správne označená ako pozitívna
- FN (*false negative*) je počet nesprávnych predpovedí, že príklad je negatívny, takže pozitívna trieda je nesprávne označená ako negatívna
- FP (*false positive*) je počet nesprávnych predpovedí, že príklad je pozitívny, čo znamená, že negatívna trieda je nesprávane označená ako pozitívna
- TN (*true negative*) vyjadruje počet správnych predpovedí, že príklad je negatívny a tým pádom je správne identifikovaná ako negatívna

Matica zámen následne vyzerá takto:

Tabuľka 5.1: Matica zámen

	Predpovedaná trieda		
Skutočná trieda		Trieda = Áno	Trieda = Nie
	Trieda = Áno	TP	FN
	Trieda = Nie	FP	TN

Najpoužívanejším parametrom, ktorý je možné získať z tejto matice je **accuracy**, ktorá je definovaná na vzorci 5.3. Ak existuje algoritmus, ktorý obsahuje 10 000 vzorkov, pričom trieda *true posity* (TP) obsahuje 9990 a *true negative* 10, výsledná presnosť algoritmu je 99,9%.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

Precision vyjadruje pomer celkového počtu správne klasifikovaných pozitívnych príkladov k celkovému počtu predefinovaných pozitívnych príkladov. Ukazuje správnosť dosiahnutú pri pozitívnej predikcii.

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

Medzi ďalšie parametre, ktoré je možné získať pomocou matice zámen patrí tiež **špecifickosť** (R, *recall*), ktorá je definovaná ako proporcia negatívnych ohodnotení oproti všetkým negatívnym.

$$recall = \frac{TN}{FP + TN} \tag{5.4}$$

Ďalším parametrom je takzvaná **pozitívna predpovedajúca hodnota** (PPV), ktorá vyjadruje pomer medzi správne predpovedanými oproti všetkým pozitívne predpovedaným:

$$PPV = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{5.5}$$

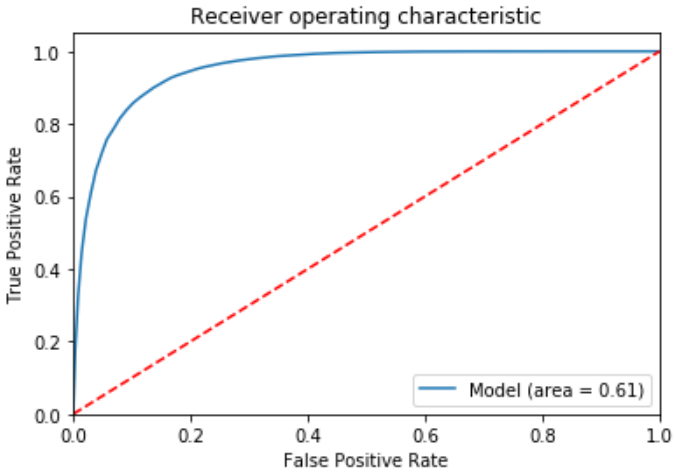
Mattewov korelačný koeficient (MCC), taktiež známy ako phi koeficient, môže byť vypočítaný priamo z matice zámen, vďaka čomu sa používa ako jedna z najlepších hodnôt pre vzájomné porovnanie výsledkov. Výpočet tohto koeficientu pomocou matice zámen, je možné vidieť na vzorci 5.6. Pričom je potrebné dať pozor, aby niektorá z hodnôt v deliteli nespôsobila, že jeden z menovateľov bude rovný 0. Ak sa vyskytne tento problém, odporúča sa nastaviť hodnota súčtu v menovateľovi na 1.

$$MCC = \frac{N_{TP} \times N_{TN} - N_{FP} \times N_{FN}}{\sqrt{(N_{TP} + N_{FP})(N_{TP} + N_{FN})(N_{TN} + N_{FP})(N_{TN} + N_{FN})}} \tag{5.6}$$

Pri nízkom počte vzoriek nemusí byť členenie na TP, FN, TN, FP vhodné, pretože sa zanedbáva s akou pravdepodobnosťou sa daná trieda predpovedá. V tom prípade je lepšie použiť napríklad metódu ROC krivky.

5.2 ROC krivky

Ide o grafické znázornenie pomeru medzi *true positive* a *false positiv* v závislosti na miere istoty, ktorú naučený model uvádza pri klasifikácii do dvoch tried. Na obrázku 5.1 je znázornená ROC krivka, pričom krivka, ktorá má pod sebou najväčšiu oblasť je najlepšou klasifikáciou. Červená krivka predstavuje náhodné hádanie, pri ktorom je četnosť FP voči TP 50:50, ak sa krivka dostane pod túto krivku, znamená to, že model predpovedá opačné situácie, než ako je skutočnosťou.



Obr. 5.1: Príklad ROC krivky Prevzaté z [35]

Z parametrov ROC krivky je taktiež možné odvodiť funkciu AUC (Area Under Curve), teda plochu pod krivkou. Ak sa hodnota tejto plochy rovná 1 znamená to, že ide o ideálnu mieru pravdepodobnosti

Kapitola 6

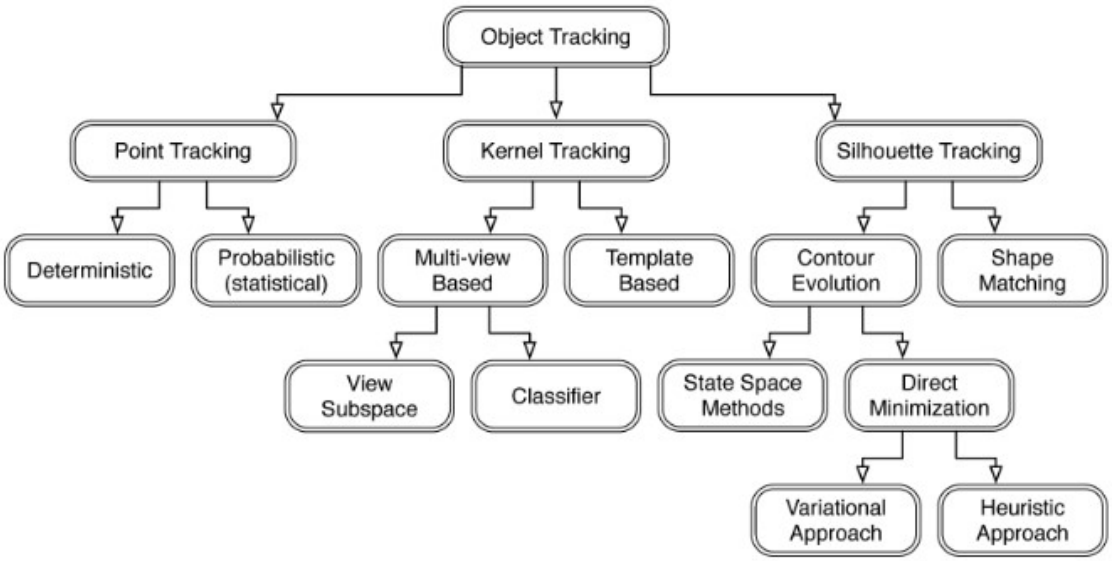
Sledovanie objektov

Cieľom algoritmov používaných na sledovanie objektov je generovať trajektóriu pohybu objektu v čase, tak že sa identifikuje jeho poloha na každom zábere videa. Pričom najväčší doraz sa kladie na výkon a presnosť sledovania. V tejto kapitole je vysvetlený základ systémov na sledovanie objektov, ako aj ťažkosti s ktorými sa tieto systémy stretávajú.

Algoritmy na sledovanie objektov je možné rozdeliť do troch základných kategórii a to [47]:

- Sledovanie bodov (*point tracking*)
- Kernel tracking
- Sledovanie siluet (*silhouette tracking*)

Tieto metódy sa ďalej rozdeľujú do ďalších presnejších skupín, ktoré je možné vidieť na obrázku 6.1.



Obr. 6.1: Rozdelenie algoritmov používaných na sledovanie objektov. Prevzaté z [64]

6.1 Metódy založené na sledovaní bodov

Túto metódu je možné v jednej vete charakterizovať ako korešpondenciu detekovaných bodov, ktoré sú predstavené pomocou bodov naprieč snímkami. Bodová korešpondencia je zložitá hlavne v situáciách, kedy sú objekty nesprávne detekované, alebo sa jedná o oklúziu, či vstupy a výstupy jednotlivých objektov.

Tieto metódy sa ďalej delia na štatistické a deterministické metódy [64]. Rozpoznávanie bodov je možné vykonať pomocou prahovania pri identifikácii týchto bodov [60]. Medzi najznámejšie modely, ktoré používajú túto techniku patrí Kalmanov filter, GOA tracker a ďalšie.

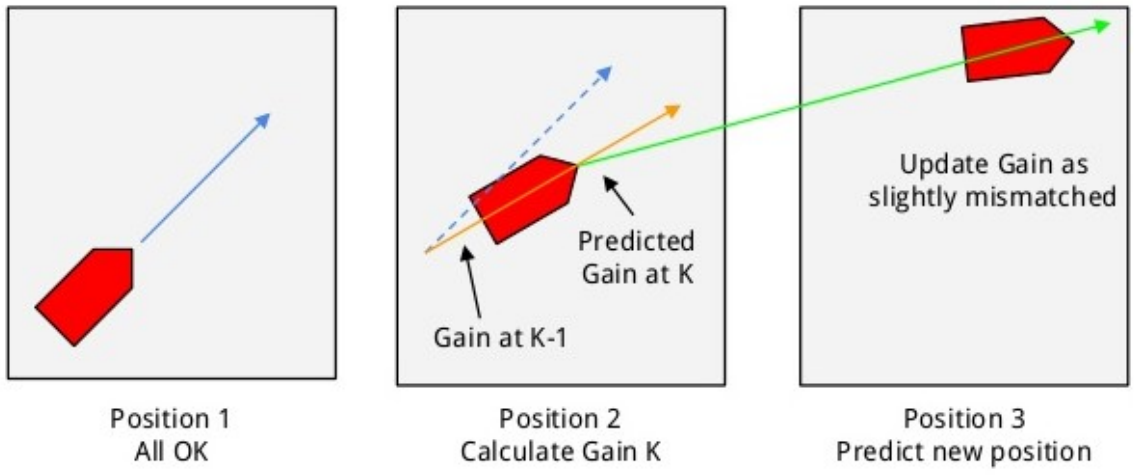
6.1.1 Kalman Filter

V roku 1960 Kalman publikoval prácu popisujúcu rekurzívne riešenie problému lineárneho filtrovania diskrétnych dát [46]. Kalmanov filter je možné použiť na systémy, ktoré obsahujú dve charakteristiky a to [63]:

- Systém je lineárny
- Aditívnym šumom je biely gaussovský šum

Každý aktualizovaný odhad stavu kalmanového filtru sa počíta z predchádzajúceho odhadu a nových vstupných údajov. Z tohto dôvodu je nutné uložiť iba predchádzajúci odhad. Vďaka tomuto rekurzívnemu prístupu je Kalmanov filter efektívnejší než systémy, ktoré na výpočet potrebujú všetky predchádzajúce hodnoty [58].

Rovnako ako Bayesianov filter aj Kalmanov filter sa skladá z dvoch krokov, a to z predikcie a aktualizácie. Tieto dva kroky sa vykonávajú pri každej iterácii.



Obr. 6.2: Príklad fungovania Kalmanova filtru. Prevzaté z [41]

V predikčnom kroku sa vygeneruje odhad označený ako *priori state* $\overline{x}_k$. V aktualizačnom korku sa zistí odhad $\hat{x}_k$, ktorý reprezentuje *posteriori state* [63]. Pretože odhadom je Gaussovo rozloženie, Kalmanov filter sleduje odhadovaný priemer a kovarianciu stavov, aby bolo možné sledovať odhad distribúcie.

Pre jednotlivé iterácie Kalmanovho filtru sú definované tieto rovnice [63]:

- Odhad:

$$\bar{\mu}_k = A_k \hat{\mu}_{k-1} + B_k u_k \tag{6.1}$$

$$\bar{\sum}_k = A\hat{\sum}_{k-1} A^T + R_k \tag{6.2}$$

- Aktualizácia:

$$K_k = \bar{\sum}_k H_k^T (H_k \bar{\sum}_k H_k^T + Q_k)^{-1} \tag{6.3}$$

$$\hat{\mu}_k = \bar{\sum}_k + K_k(z_k - H_k \bar{\sum}_k) \tag{6.4}$$

$$\hat{\sum}_k = (I - K_k H_k) \bar{\sum}_k \tag{6.5}$$

6.2 Kernel tracking

Kernel tracking sa zvyčajne vykonáva výpočtom trajektórie objektu, ktorý je reprezentovaný ako jednoduchá objektová oblasť naprieč videom. Pohyb objektov je zvyčajne vo forme parametrického pohybu, ako je napríklad translačný, konformý alebo afinitný pohyb. Taktiež je možné použiť husté tokové pole vypočítané pomocou nasledujúcich záberoch videa [64].

Algoritmy založené na *kernel tracking* sa líšia najmä z hľadiska reprezentácie vzhľadu, počtu sledovaných objektov a metódy použitej na odhad pohybu sledovaného objektu. *Kernel tracking* sa delí na dve podkategórie a to na metódy založené na šablóne (*Mean-shift*) a metódy založené na *multiview (SVM tracker)* [64].

6.2.1 Mean-shift

Mean-shift algoritmus s použitím na sledovania objektov bol prvýkrát formulovaný v práci od Comanicia a Meera viz. [30]. Mean-shift tracking pracuje s vybraným objektom na zábere *n* a potencionálnym objektom na zábere *n+1*. Pre účely sledovania je model definovaný ako distribúcia farebnej hustoty daného objektu. Model objektu sa odhaduje z diskrétnej hustoty farebného histogramu objektu [55].

Pravdepodobnosť podľa vzorca x.x, popisuje pravdepodobnosť, že určitá farba patriaca objektu so stredom $\hat{x}$ môže byť vyjadrená ako pravdepodobnosť javu $u = 1...m$ vo vybranom modely [55].

$$q_u = C \sum_{i=1}^{N} k(||\frac{x_i - \hat{x}}{h}||^2)\delta[b(x_i) - u] \tag{6.6}$$

Kde:

- δ je impulzná funkcia
- h je šírka pásma
- N je počet pixelov vybraného modelu
- C je prevrátená hodnota súčtu hodnôt funkcie jadra *k(z)*

Jadro K spolu s jadrovou funkciou *k(z)*, zvyšuje spoľahlivosť odhadu hustoty. Dôvodom je to, že pixely, ktoré sú ďalej od stredu majú menšiu váhu, takže tieto pixely príliš neovplyvňujú odhad hustoty [55].

Kľúčovou časťou algoritmu mean-shift, používaného na sledovanie objektov, je výpočet posunu od pozície starého objektu $\hat{x}$ k novej pozícii $\hat{x}_{new} = \hat{x} + \Delta x$. Výpočet je realizovaný pomocou odhadu stredného vektora posunu [55].

$$\Delta x = \frac{\sum_i K(x_i - \hat{x})\omega(x_i)(x_i - \hat{x})}{\sum_i K(x_i - \hat{x})\omega(x_i)} \tag{6.7}$$

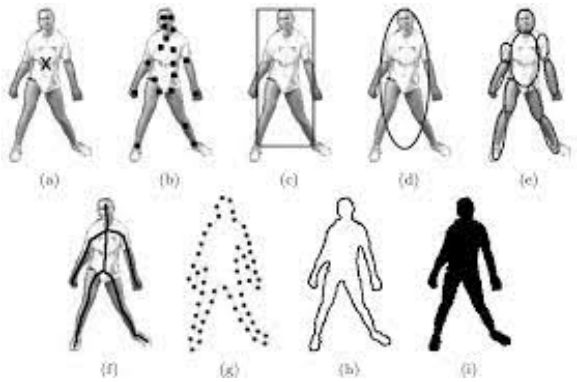
Pričom $\omega(\mathrm{x_i})$ predstavuje váhu x_i, ktorá je definovaná pomocou vzorca:

$$\omega(x_i) = \sum_{u=1}^{m} \delta[b(x_i) - u]\sqrt{\frac{q_u(\hat{x})}{p_u(\hat{x}_{new}}} \tag{6.8}$$

6.3 Metódy založené na sledovaní siluet

Objekty, ktoré sa skladajú z komplexných tvarov ako je napríklad pri ľuďoch ruka, alebo hlava, nedokážeme jasne popísať jednoduchými geometrickými tvarmi. Algoritmy na sledovanie objektov používajúce metódy sledovanie siluet sa snažia definovať, aj takto zložité tvary pomocou jednoduchých geometrických útvarou, s cieľom vytvoriť odpovedajúce modely sledovaných objektov.

Cieľom je nájsť oblasť objektu na každom zábere videa. Táto oblasť sa zisťuje za pomoci objektového modelu, ktorý je vygenerovaný prostredníctvom predchádzajúcich záberov. Model môže mať formu buďto farebného histogramu, hrán objektu alebo obrysu objektu [64]. Príklady reprezentácie objektov je možné vidieť na obrázku 6.3.



Obr. 6.3: Príklad reprezentácie objektov Prevzaté z [64]

Algoritmy používajúce metódu sledovania siluet sa dajú ďalej rozdeliť do dvoch podskupín. Prvú podskupinu tvoria algoritmy pracujúce pomocou priraďovania tvaru, ako je napríklad algoritmus Hausdorf. Druhá skupina používa techniku sledovania kontúr, do tejto skupiny patrí State space models. [64]

Pri metóde sledovanie kontúr sa na každom záber vyberie hranica sledovaného objektu. Následne sa vypočíta sa obrys z predchádzajúceho záberu a jeho nová pozícia v súčasnom zábere videa. Na konkrétne sledovanie sa používajú dva prístupy. Prvým je stavový model priestoru, ktorý špecifikuje tvar a pohyb obrysu. Druhým prístupom je použitie gradientového zostupu, vďaka čomu je algoritmus schopný sledovať väčšie množstvo objektov s rôznym tvarom [47].

Metóda priraďovanie tvarov je veľmi podobná metóde sledovania kontúr. Na každom zábere je potrebné nájsť a identifikovať siluetu, na ktorú sa následne použije zhoda tva-

rov. Pomocou tejto techniky sa hľadá zhoda kontúr alebo siluety v dvoch po sebe idúcich záberoch videa. Táto metóda zvláda oklúzie pomocou použitia Houghovej transformácie a sledovania vždy len jedného objektu [47].

6.4 DeepSORT algoritmus

DeepSORT je algoritmus na sledovanie objektov, ktorý je založený na základoch jeho predchodcu a to algoritmu SORT (*Simple Online and Realtime Tracking*) [25]. DeepSORT algoritmus sa skladá zo štyroch hlavných častí [32]:

1. Prvou časťou je odhad trasy, ktorý bol súčasťou už pôvodného algoritmu SORT a používa Kalmanov filter na predpovedanie polohy bounding boxu. Obsahuje osem premenných a to *u, v, a, h, u', v', a', h'*, kde (u, v) predstavujú stred bounding boxu, a je pomer strán a hs predstavuje výšku obrazu. Predikcia je založená na rýchlosti pohybu objektu pri predošlých detekciách. Odhad trasy je metrika, ktorá používa vzdialenosť IoU (intersection over union) medzi bounding boxom a potencionálnym novým bounding boxom.

2. Druhým krokom je použitie deskriptoru vzhľadu. Pri tomto kroku sa extrahujú informácie ohľadom vzhľadu objektu. Informácie sú extrahované pomocou CNN tak aby, vlastnosti z jednej triedy objektu boli podobné a naopak vlastnosti z opačných tried sa líšili.

3. Časť označovaná ako združovač údajov, priraďuje ku stope objektu bounding boxy. Tento proces sa vykonáva pomocou metrík získaných z prvého a druhého kroku. Každá existujúca stopa predstavuje identitu objektu, teda jeho ID.

4. Poslednou časťou je track handing, ktorý spravuje stopu objektu. Ak detekovaný objekt nie je možné spojiť so stopou na zábere x, umiestni sa tento bounding box do predbežnej stopy. Algoritmus sa následne snaží spojiť túto predbežnú stopu s inými stopami v nasledujúcich záberoch videa. Ak sa nájde vhodná stopa, stopa sa aktualizuje. V opačnom prípade je predbežná stopa odstránená, čím sa zvyšuje efektívnosť pôvodného SORT algoritmu.

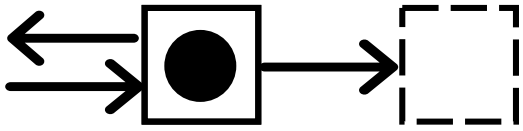
6.5 Problémy sledovania objektov

Sledovanie objektov na záznamoch videa naberá v súčasnosti na popularite, vďaka čomu došlo k obrovskému pokroku. Stále sa však existujú problémy, ktoré sťažujú proces analýzy týchto záznamov. Medzi najčastejšie problémy patria [43]:

- Nečakaný pohyb objektu
- Hijacking problem
- Zmena osvetlenia alebo zmena vzhľadu pozadia
- Problém centralizácie

Nečakaný pohyb objektu nastáva vtedy, ak objekt náhle zmení svoj smer.Kkonkrétnym príkladom je napríklad drifting problém, pri ktorom sa objekt nečakane začne pohybovať

opačným smerom. Dôsledkom je, že objektu je priradené nové unikátne číslo. Od tejto chvíle je sledované nové ID a objekt je vnímaný ako nový objekt, ktorý sa predtým na videu ešte nevyskytol.



Obr. 6.4: Nákres drifting problému

Ak sa dva objekty podobnej farby ocitnú v tesnej blízkosti je možné, že jeden tracker jedného objektu preskočí na druhý objekt. Tento problém sa označuje ako hijacking problem. Tento problém zapríčiňuje to, že jeden objekt je určitú dobu sledovaný ako dva rôzne objekty, čo skresľuje výsledky sledovania. Taktiež je možné, že sa jeden objekt prestane sledovať úplne.

Obr. 6.5: Nákres hijacking problému

Každá zmena počasia a osvetlenia scény ovplyvňuje kvalitu systémov na sledovanie objektov. Najlepšie podmienky sú, ak je jasno a dajú sa jasne rozoznať objekty, prípadne ak je zlé počasie, sneží, prší alebo je hmla, presnosť sa zmenšuje a to hlavne pri predmetoch, ktoré nemajú výrazne farby, ktoré by ich jednoznačne odlíšili od pozadia.

Ak sa dva objekty ocitnú v jeden okamžik na jednom mieste dochádza k problému centralizácie, kedy program na sledovanie prestáva sledovať správne objekty, potom čo sa tieto objekty rozdelia. Na obrázku 7.2 je nákres situácie, pri ktorej dochádza k problému centralizácie. Na obrázku je možné vidieť situáciu kedy sa objekty ocitli na jednom mieste a čiastočne sa zakrývajú a následná zmena trackera po ich rozdelení.

Obr. 6.6: Nákres problému centralizácie

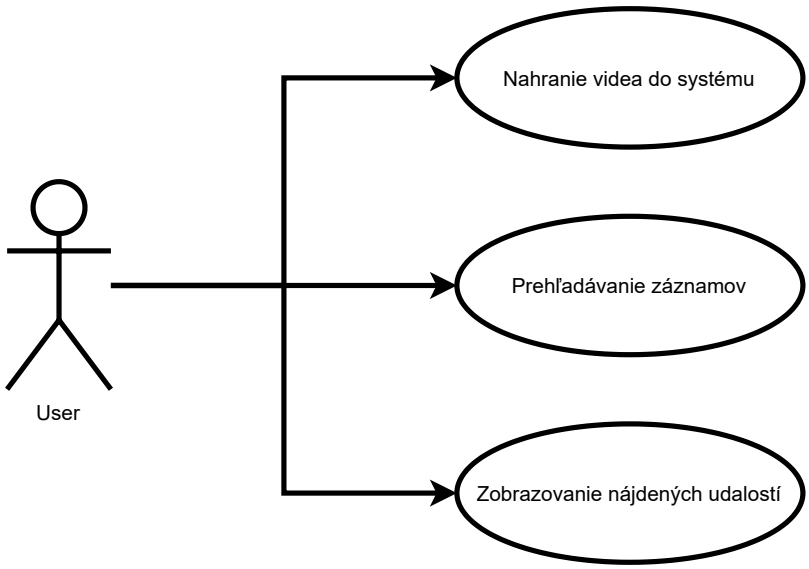
Kapitola 7

Implementácia

Táto kapitola sa venuje popisu programu, ktorý tvorí praktickú časť tejto práce. Tento program umožňuje užívateľovi analyzovať záznamy z bezpečnostných kamier pomocou dotazovania sa na rôzne udalosti. Cieľom bolo vytvoriť program, ktorý umožní efektívne analyzovanie záznamov z bezpečnostných kamier. Zároveň sa navrhol tak aby bolo ovládanie konečného systému jednoduché a intuitívne.

Systém je navrhnutý ako offline analyzátor bezpečnostných záznamov. Užívateľ je schopný do systému nahrať video, ktoré je následne analyzované. V priebehu analýzy sú detekované objekty zo záznamu, následne sú tieto objekty sledované počas celej doby trvania videa. Pri analýze videa sa taktiež zisťuje farba detekovaného objektu.

Užívateľské rozhranie je implementované ako webová aplikácia, ktorú užívateľ používa na uľahčenie práce so systémom. Celý výpočet sa odohráva na serverovej časti systému a následne je užívateľ schopný komunikovať so systémom pomocou API je. Aplikácia je navrhnutá tak, aby fungovala v tandeme s analyzátorom videa. Kvôli vzájomnej tesnej integrácii všetkých komponentov systému je dostupná len na zariadeniach s týmto analyzátorom, aj keď je vytvorená za použitia webových technológii.



Obr. 7.1: Use case diagram webovej aplikácie

Pri každom spustení analýzy nahratého záznamu sa užívateľovi zobrazí prvý záber videa. Užívateľ následne na zábere vyznačí ROI (region of interest), pomocou ktorého sa vypočíta matica perspektívnej projekcie, podľa postupu vysvetleného v kapitole 3. Pri analyzovaní dotazov od užívateľa sa prechádzajú všetky záznamy ktoré boli nahraté do systému. Užívateľ si z jednoduchých komponent skladá príkaz, ktorý bude vyhľadávaný. Na vytvorenie vyhľadávacieho dotazu má užívateľ k dispozícii tri stavebné bloky:

1. Objekt
2. Atribút objektu
3. Relácia medzi objektami

Po analyzovaní všetkých záznamov je užívateľovi k dispozícii tabuľka pomocou, ktorej si môže zobraziť jednotlivé časti videa, na ktorých bola detekovaná vyhľadávaná udalosť. Zobrazený snímok obsahuje názov videa ako aj časovú značku.

7.1 Prehľad systému

Ako už bolo spomenuté, systém sa skladá z dvoch častí a to z časti klienta a serveru. Serverová časť je zodpovedná za analýzu videí, získavania odpovedí na dotazy od užívateľa a taktiež na odosielanie odpovedí klientovi. Klientova časť systému prijíma príkazy užívateľa, verifikuje správnosť dotazov a zobrazuje odpovede od serveru.

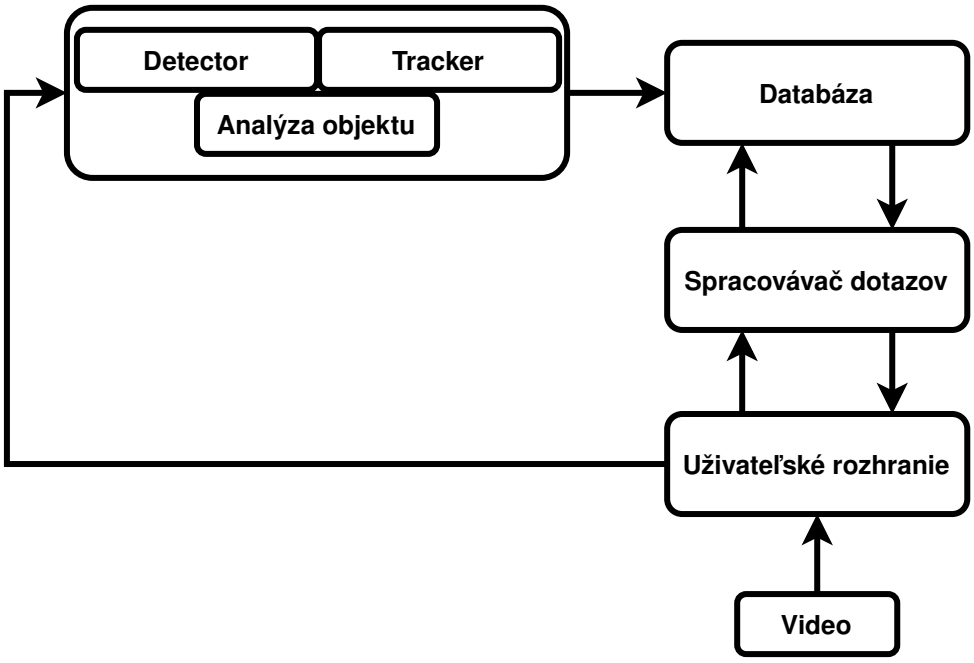
Potom, ako sa nahrá video do systému a je vypočítaná matica perspektívnej projekcie (kapitola 3), prejde sa k analýze nahraného videa, a teda k detekcií objektov (kapitola 4) v súčinnosti so sledovačom objektov (kapitola 6) extrahujú informácie o jednotlivých objektoch na zázname. Extrahované informácie o objektoch sa ukladajú do databázy. Pri každom detekovanom objekte sa taktiež vykonávajú špeciálne analýzy, ktorých cieľom je zistenie farby objektov, ako aj to, či k danému objektu prislúchajú ďalšie objekty zo záberu videa.

Databáza sa následne prehľadáva podľa podmienok zadaných užívateľom. Z databázy sa vyberajú objekty, ktoré odpovedajú jednotlivým zadaným parametrom. Ak objekty spĺňajú podmienky celého dotazu zadaného užívateľom, výsledok sa uloží ako jeden záznam do jsonu. Výsledný json sa užívateľovi zobrazí v podobe tabuľky ako výsledok vyhľadávania. Užívateľ si sám vyberie, ktorý výsledok si chce zobraziť a po vybratí konkrétneho záznamu sa mu zobrazí fotka, ktorá obsahuje orezanú časť záberu, na ktorom nastala hľadaná udalosť.

7.2 Nástroje použité pri implementácii

Na implementáciu serverovej časti systému bol použitý programovací jazyk **Python**. Tento jazyk bol vybraný z dôvodu, že podporuje veľké množstvo knižníc, medzi ktoré patrí aj knižnica OpenCV. Taktiež bol vybraný kvôli jeho univerzálnosti, vďaka čomu je ľahko prispôsobiteľný pre rôzne potreby systému.

Na vytvorenie užívateľského rozhrania bol použitý javascriptový *framework* **Vue.js**. Vue CLI (command line interface) ponúka množstvo užitočných funkcií, ktoré vylepšujú vývoj pomocou frameworku Vue, ale taktiež umožňuje generovať a predkonfigurovať jednostránkové aplikácie. Tieto jednostránkové aplikácie je možné vytvoriť pomocou príkazu *vue create*, alebo pomocou webového GUI, *frameworku* Vue.js, ktoré je možné spustiť príkazom *vue ui*. Použitie užívateľského rozhrania proces vytvorenia jednostránkovej aplikácie výrazne zjednodušuje a urýchľuje.



Obr. 7.2: Prehľad systému

Node.js je *open-source, cross-platform Javascript runtime* prostredie, ktorý sa skladá z engine V8, ktorý je zabalený v C++ programe. Rýchlo vytvára škálovateľné sieťové aplikácie. Node.js používa neblokujúci I/O model, ktorý je založený na udalostiach, vďaka čomu je ideálny pre dátovo náročné aplikácie, ktoré bežia v reálnom čase.

Na vývoj webového serveru bol použitý framework **Express**, ktorý bol vytvorený pre **Node.js** na tvorbu webových aplikácii. Zjednodušuje vytváranie webových serverov a používanie robustných API (*application programming interface*). Následné vytvorenie webového serveru je jednoduché a zmestí sa len na dva kroky. ako je vidieť tejto časti kódu:

```
const express = require('express');
const app = express();
```

Výpis 7.1: Príklad vytvorenia webového serveru pomocou frameworku Express

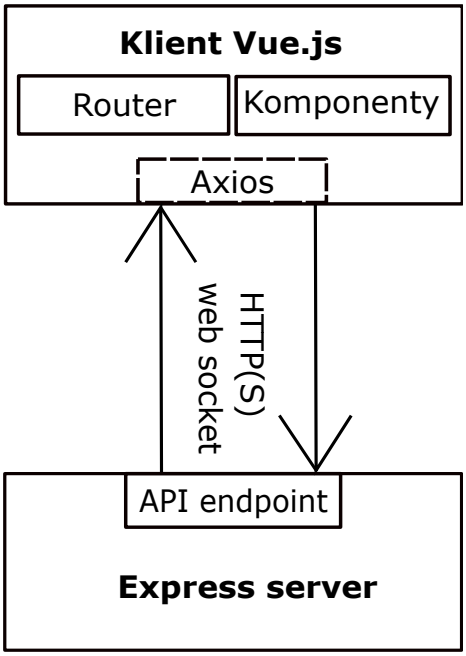
Jednotlivé komponenty užívateľského rozhrania sú vytvorené pomocou frameworku **Vuetify**. Vuerify je kompletný UI *framework* postavený na Vue.js. a je vyvíjaný podľa špecifikácie Material Design. Každá komponenta tohto *frameworku* je modulárna, výkonná a reagujúca. Vďaka čomu je celý proces vývoja užívateľského rozhrania jednoduchý a rýchly.

Na prácu s databázou bol použitý **SQLite**, samostatná súborová databáza SQL. SQLite je C knižnica, s ktorou je možné pracovať aj v Pythone, bez nutnosti inštalácie ďalšieho softwaru. Na prácu s SQLite je možné v Pythone použiť modulu sqlite3, ktorý systému poskytuje rozhranie SQL.

Ďalším dôležitým prvkom systému je open source knižnica **OpenCV**, ktorá obsahuje algoritmy používané na počítačové videnie a strojové učenie. V systéme sa používa na získavanie matice perspektívnej projekcie, podľa kapitoly 3. Taktiež sa používa na získavanie informácii o analyzovanom videu, ako je napríklad kodek videa, dĺžka ako aj na manipuláciu s videom a jeho zábermi pri zmene farebnej škály videa pre lepšiu analýzu, alebo orezanie jednotlivých záberov.

Knižnica OpenCV sa taktiež použitá v implementácii algoritmu na sledovanie objektov[1]. Implementácia používa algoritmus YOLO (zo sekcie 4.8) na detekciu objektov, ktorá ďalej tieto objekty posiela algoritmu na ich sledovanie. Algoritmus na sledovanie objektov vo videu je implementovaný pomocou DeepSORT algoritmu (zo sekcie 6.4). Na zrýchlenie analýzy bola použitá GPU, ktorá výrazne zrýchľuje analýzu videa oproti použitiu analyzátora len za pomoci CPU.

TensorFlow je open-source knižnica používaná pri strojovomu učení a je ďalším prvkom systému. Používa sa pri trénovaní hlbokých neurónových sietí. Predtrénované YOLO weights konvertuje na TensorFlow model. Pomocou tohto modelu sa realizujú detekcie objektov na záznamoch.



Obr. 7.3: Schéma komunikácie medzi klientom a serverom

7.3 Architektúra

Táto časť sa zaoberá architektúrou aplikácie, jeho serverovej ako aj klientskej časti.

7.3.1 Serverová časť aplikácie

Serverová časť systému sa skladá z piatich hlavných častí:

- Funkcia na získavanie perspektívnej matice
- Analýza videa a analýza detekovaných objektov
- Databáza systému
- Spracovanie príkazov od užívateľa

[1] https://github.com/theAIGuysCode/yolov4-deepsort

- Webový server

Ako už bolo zmienené, prvou časťou je funkcia na získavanie perspektívnej matice. Po načítaní videa sa užívateľovi zobrazí prvý záber záznamu videa, na ktorom používateľ následne pomocou myši zadá šesť bodov. Prvé štyri body vytvoria obdĺžnikovú oblasť na základnej rovine, ktorá predstavuje oblasť záujmu ROI. Konečný výpočet matice je realizovaný pomocou knižnice OpenCV.

```
src = np.float32(np.array(points[:4]))
dst = np.float32([[0, H], [W, H], [W, 0], [0, 0]])
matrix = cv2.getPerspectiveTransform(src, dst)
```

Výpis 7.2: Príklad výpočtu perspektívnej matice pomocou knižnice OpenCV

Ďalšou serverovou časťou je analýza záznamov. V tejto časti programu sa detekujú jednotlivé objekty na videu a následne sa skúma ich pohybovanie počas celého videa. Okolo každého detekovaného objektu sa vytvorí bounding box s unikátnym identifikátorom.

Do databázy *annotated_video_data.db* sa ukladajú získané informácie z analyzovaných videí o jednotlivých objektoch. Priklad tabuľky je vidieť na 7.1

Tabuľka 7.1: Príklad jedného záznamu v databáze

Index	5
File_name	video_file.mp4
Frame_number	10
ID	5
Class	person
BBox_xmin	147
BBox_ymin	382
BBox_xmax	313
BBox_ymax	819
Color	Unknow
Human_action	Unknow
Related_objects	[('bag', '3')]

Ak bol detekovaný objekt typu *car* alebo *bag*, zavolá sa funkcia *get_average_color*. Táto funkcia prechádza celú oblasť detekovaného obrazu, ktorá je ohraničená pomocou jeho bounding boxu, ktorý bol zmenšený o 50%. Oblasť bola zmenšená z dôvodu odstránenia časti bounding boxu, ktoré obsahujú okolie detekovaného objektu. Týmto spôsobom sa zabezpečí analýza stredovej časti detekovaného objektu. Pomocou všetkých pixelov v danej oblasti sa získava priemerná farba objektu. Funkcia vracia trojicu obsahujúcu RGB (*Red Green Blue*) hodnotu získanej farby.

```
for y in range(y_min, y_max):
    for x in range(x_min, x_max):
        r += frame[y,x,2]
        g += frame[y,x,1]
        b += frame[y,x,0]
        count += 1
```

Výpis 7.3: Výpočet priemernej RGB hodnoty všetkých pixelov vo vybranej oblasti

Za účelom priradzovania druhu farby k jeho RGB hodnote bol vytvorený dataset, v ktorom sú základné farby ako je žltá, modrá, červená, ale aj zložené typy farieb, ako je napríklad žlto-zelená. Ak je objekt typu *person*, farba je označená ako *Unknown*. Dataset obsahuje RGB hodnoty jednotlivých farieb, ktoré sa prehľadávajú pomocou algoritmu K najbližších susedov (*K Nearest-Neighbor*). K nami vypočítanej RGB hodnote objektu sa priradí typ farby, ku ktorej mal v datasete najbližie.

Štvrtou časťou je analyzátor príkazov od užívateľa. Program podľa dotazov, ktoré užívateľ zadal do GUI prehľadáva databázu a vyhodnocuje, či jednotlivé situácie nastali. Táto časť sa skladá z piatich hlavných funkcii, pričom každá z týchto funkcií predstavuje jeden model, ktorý bol pre túto aplikáciu vyvinutý. V programe sú implementované tieto modely:

- Model sociálneho dištancovania sa medzi dvomi osobami
- Model stretnutia dvoch ľudí
- Model výmeny tašky
- Model vystúpenia z auta
- Model nastúpenia do auta

Model sociálneho dištancovania medzi ľuďmi slúži na kontrolu dodržiavania bezpečných vzdialeností. Bezpečnostná vzdialenosť je predvolená na dva metre, avšak užívateľ dokáže túto vzdialenosť zmeniť pomocou užívateľského rozhrania. Vzdialenosť sa dá ľubovoľne zmeniť v rozmedzí od 0 do 10 metrov, pričom posun je možné realizovať po 0.5 metra. Príklad porušenia bezpečnej vzdialenosti medzi ľudmi je možné vidieť na obrázku 7.4.

Obr. 7.4: Príklad porušenia bezpečnej vzdialenosti medzi dvomi osobami

Druhým modelom je stretnutie dvoch ľudí, pri ktorom rovnako ako pri modele sociálneho dištancovania medzi ľuďmi je dopredu zvolená vzdialenosť, ktorú systém používa, ak nebola definovaná iná vzdialenosť. V tomto prípade je vzdialenosť prednastavená na 0.5 metra, pričom užívateľ ju môže podľa potreby zväčšiť alebo zmenšiť, v rovnakom rozmedzí ako tomu bolo pri predchádzajúcom modeli.

Tento model má okrem vzdialenostnej podmienky aj časovú podmienku. Na to, aby bola pozitívne detekovaná situácia stretnutia medzi dvomi ľuďmi je potrebné, aby stretnutie trvalo viac než 2 sekundy. Táto doba bola zvolená, pretože sa predpokladá, že za túto dobu sa dvaja okoloidúci ľudia stihnú vyhnúť, ale zároveň je časová doba dostatočne krátka na to, aby zaznamenala aj krátke stretnutie trvajúce iba pár sekúnd. Príklad stretnutia

dvoch ľudí je možné vidieť na obrázku 7.5. Po splnení oboch podmienok, systém pridá informácie do jsonu.

Obr. 7.5: Príklad stretnutia dvoch ľudí

Tretí je model výmeny tašky, ktorý slúži na detekciu situácie, kedy taška zmenila svojho majiteľa. Pod pojmom zmena majiteľa sa predpokladá situácia, pri ktorej je taška na začiatku u jednej osoby a následne v priebehu videa túto tašku preberie iná osoba. Pričom taška sa už po zbytok záznamu neobjaví u pôvodnej osoby.

Keďže medzi taškou a osobou je zvyčajne veľmi malá vzdialenosť, ktorá nejde presne identifikovať, používa sa v tomto modeli položka z databázy nazvaná ako *Related_objects*. *Related_objects* sa napĺňa počas analýzy videa a obsahuje objekty, ktoré majú so skúmaným objektom spoločnú časť bounding boxu, pričom pre objekt typu *person* sa skúmajú len objekty typu *bag*. Na obrázku 7.6 je možné vidieť situáciu, pri ktorej taška zmenila svojho majiteľa.

Obr. 7.6: Príklad výmeny tašky

Ďalším modelom je vystúpenie z auta. V tomto modeli je taktiež použitá vzdialenostná podmienka, ktorá je definovaná na vzdialenosť jeden a pol metra. Túto podmienku, na rozdiel od vzdialenostnej podmienky u modelu stretnutia dvoch ľudí, nemôže užívateľ prispôsobiť. Na druhú stranu užívateľ môže nastaviť farbu vozidla, čím sa zúži rozsah skúmaných objektov.

Ďalšou podmienkou je, že objekt triedy *person*, ktorý spĺňa vzdialenostnú podmienku, nebol prítomný na videu, predtým než bola splnená vzdialenostná podmienka. Ak sú splnené obe podmienky ide o vystúpenie z auta. Na obrázku 7.7 je možné vidieť, ako osoba vystupuje z auta.

Obr. 7.7: Príklad vystúpenia z auta

Posledným implementovaným modelom je nastúpenie do auta. Tento model, rovnako ako model vystúpenia z auta, obsahuje vzdialenostnú podmienku, ktorú užívateľ nemôže meniť a je nastavená na jeden a pol metra. Ďalšou podmienkou je, že skúmaný objekt triedy *person* sa po splnení podmienky už nevyskytuje na videu. Vďaka tejto podmienke sa zabezpečí, že výsledkom je posledná detekcia, na ktorej bola osoba v blízkosti auta. Tento výsledok je možné vidieť na obrázku 7.8.

Obr. 7.8: Príklad výsledku modelu: nastúpenie do auta

Poslednou serverovou časťou tejto práce je webový server. Pomocou webového serveru sa realizuje spúšťanie jednotlivých častí. Spúšťanie sa realizuje zavolaním daného pythonového súboru priamo z príkazovej riadky pomocou modulu *child_process*.

Modul *child_process* nám umožňuje prístup k funkciám operačného systému prostredníctvom spustenia ľubovoľného príkazu vo vnútri podradeného procesu. S podprocesom pracujeme rovnako, ako by sme pracovali v operačnom systéme Linux.

Existujú štyri rôzne spôsoby, ako vytvoriť podradený proces. V tejto aplikácii bola použitá funkcia *child_process.spawn()*. Funkcia spawn spúšťa príkaz v novom procese a taktiež sa dá použiť na predávanie argumentov. Na nasledovnej časti kódu je možné vidieť zavolanie funkcie *spawn*, ktorá spúšťa program na analyzovanie video záznamov a predáva mu argumenty.

```
let arg = [‘object_tracker_new.py‘, --video, data/video/${req.body.name}
const { spawn } = require(’child_process’);
const analizer = spawn(process.env.HOME +
```

```
'/anaconda3/envs/yolov4-gpu/bin/python3',
arg, { stdio: ['pipe', 'pipe', 'pipe'] });
```

Výpis 7.4: Príklad zavolania pythonového skriptu

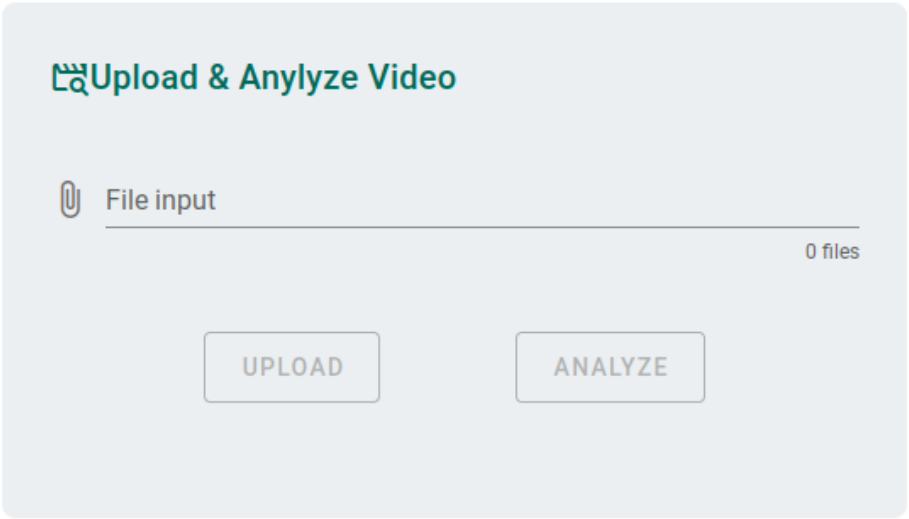
7.3.2 Klientska časť aplikácie

Užívateľské rozhranie sa skladá z jednej jednostránkovej aplikácie, na ktorej sa následne menia len jej komponenty. Prvá stránka slúži na zadávanie dotazov od užívateľa, podľa ktorých sa prehľadáva databáza. V tejto časti sa taktiež vykonáva kontrola, ktorá zabraňuje odoslaniu zlých argumentov na serverovú časť aplikácie. Kontrola je zabezpečená použitím vlastnosti *disabled*, ktorá nedovolí používať prvky stránky, pre ktoré nie sú splnené pravidlá. Rovnako je znemožnené používať aj tlačidlo *Search*, ktoré slúži na vyhľadávanie dotazov.

Obr. 7.9: Príklad prvku bez vlastnosti disabled

Obr. 7.10: Príklad prvku s použitím vlastnosti disabled

Potom ako užívateľ zadá správne všetky argumenty, je mu umožnené použiť tlačidlo, ktoré pošle informácie o zadanom dotaze na serverovú časť. Klientska časť komunikuje so serverovou časťou prostredníctvom HTTP protokolu použitím knižnice *Axios* a konkrátne príkazom *post()*. Táto knižnica je podporovaná vo všetkých prehliadačoch. *Axios* taktiež dokáže automaticky transformovať dáta do JSONu čím sa značne uľahčí práca.



Obr. 7.11: Formulár na nahrávanie videozáznamu.

Ďalšia stránka slúži na nahrávanie videí na server a taktiež na následné spúšťanie analýzy záznamu. Záznam videa sa na server nahráva s pôvodným názvom, čo uľahčuje užívateľovi prácu s následne analyzovanými záznamami. Proces analýzy videa nie je možné spustiť pred tým, než užívateľ úspešne nahral video na server. Na nahrávanie videa bola použitá metóda *axios.post()* a príklad nahrávania na server je možné vidieť na následujúcej ukážke kódu:

```
submitFile(){
    let formData = new FormData();
    formData.append('file', this.file);
    try {
        axios.post( 'http://localhost:8001/upload', formData);
        this.message = "File has been uploaded";
        this.error = false;
        this.upload_dis = true;
        } catch(err) {
            this.message = "Something went wrong";
            this.error = true;
            }
  },
```

Výpis 7.5: Časť kódu popisujúca odosielanie dotazov od užívateľa na server

Kapitola 8

Dosiahnuté výsledky

Testovanie systému bolo rozdelené na tri časti. Prvá časť sa zameriavala na testovanie programu, kedy sa skúmala schopnosť systému správne rozpoznávať jednotlivé farby objektov. Druhá časť vyhodnocovala kvalitu jednotlivých modelov na rozpoznávanie rôznych situácii. Poslednou, treťou časťou, bolo testovanie systému na dobrovoľníkoch, ktorí hodnotili vizuálnu stránku systému a jednoduchosť ovládania. Na túto časť testovania boli oslovení štyria dobrovoľníci.

Za účelom testovania implementovaných modelov boli vytvorené špeciálne videá, ktoré obsahujú implementované situácie s rôznymi elementami, ktoré analýzu videa a následné zodpovedanie dotazov značne sťažujú. Príkladom takýchto elementov je napríklad použitie tašky, ktorá splývala s oblečením pozorovanej osoby, alebo väčšie množstvo ľudí, ktorí čiastočne zacláňali výhľadu na testovanú situáciu. Okrem vlastných videí, boli na testovanie použité aj videá z datasetov dostupných na internete, ako je napríklad VIRAL video dataset [20], alebo záznamy vytvorené pomocou vysielania Brnenských dopravných kamier [4].

8.1 Testovanie funkcí systému

Pri testovaní funkcie, ktorá zaradzovala farby jednotlivých objektov do farebných skupín, boli dôležité tri faktory. Prvým a aj najvýznamnejším faktorom je veľkosť použitého datasetu. Dataset použitý v tejto práci obsahoval 276 rôznych farieb a 19 farebných skupín. Čím viac farieb dataset obsahoval, tým väčšia bola úspešnosť priradenia farby do odpovedajúcej skupiny. Ďalším faktorom ovplyvňujúcim kvalitu funkcie, bola intenzita skúmanej farby. Výrazná a sýta farba sa ľahšie priradzovala do skupiny, než svetlé, pastelové odtiene, ktorých odtiene sa výrazne podobali na ďalšie svetlé odtiene inej farby. Posledný faktor je subjektívny a ide teda o schopnosť, ako farbu vníma osoba, ktorá sa na ňu pozerá. Z tohto dôvodu, nebolo možné určiť percentuálnu úspešnosť funkcie, pretože bez názvov jednotlivých farieb nedokážem jednoznačne identifikovať, do ktorej z devätnástich farebných skupín, skúmaná farba patrí.

Ako ďalšie boli testované jednotlivé modely. Pri testovaní sa používali metriky, ktoré sú vysvetlené v kapitole 5. Model, ktorý kontroval dodržiavanie bezpečných vzdialeností medzi dvomi ľuďmi nebolo možné jednoznačne overiť. Dôvodom bola absencia informácií o reálnych vzdialenostiach medzi ľuďmi na videozáznamoch. Zo subjektívneho pohľadu bol tento model dostatočne úspešný.

Na nasledujúcej tabuľke 8.1 sa nachádzajú výsledky testovania. Tabuľka obsahuje štyri stĺpce. V prvom sa nachádza názov modelu, ktorý bol testovaný. Druhý stĺpec obsahuje

percentuálne vyjadrenie úspešnosti jednotlivých modelov, pri ktorom sa brali do úvahy len videá, na ktorých sa daná situácia udiala. Tretí stĺpec vyjadruje celkovú úspešnosť modelu, ktorá bola vypočítaná podľa vzorca 5.3. Jednotlivé percentá boli zaokrúhlené na celé čísla pre lepšiu reprezentáciu. Posledný stĺpec obsahuje celkový počet použitých testovacích videí pre jednotlivé modely.

Tabuľka 8.1: Súhrn výsledkov testovania

Model	**Úspešnosť modelu**	**Celková úspešnosť**	**Počet videí**
Nastupovanie do auta	100%	58%	17
Vystúpenie z auta	100%	65%	17
Stretnutie ľudí (0,5 metra)	64%	78%	18
Stretnutie ľudí (1 meter)	73%	83%	18
Výmena tašky	67%	75%	12

Model zaisťujúci rozpoznávanie situácie, kedy osoba nastupuje do auta dosiahol pri testovaní situácii kedy došlo k reálnemu nastúpení do auta úspešnosť 100%. Celkovú úspešnosť modelu bola 58%. Táto úspešnosť bola vypočítaná pomocou matice zámen z kapitoly 5, ktorá po naplnení hodnotami vyzerala nasledovne:

Tabuľka 8.2: Matica zámen pre model nastúpenia do auta

TP = 6	FN = 0
FP = 7	TN = 4

Výsledná 58 percentná úspešnosť bola spôsobená tým, že za nastúpenie boli označené aj situácie, pri ktorých k tejto udalosti nedošlo (FP), tento prípad počas testovania nastal až sedemkrát, ako je možné vidieť v tabuľke.

Negatívne prípady boli napríklad, že osoba tesne prechádzala okolo auta, ktoré bolo tesne pri hrane záberu, takže sa stalo, že osoba následne vyšla zo záberu, ako aj keď osoba obišla auto koldokola takže osoba bola snímaná z rôznych strán, a taktiež bola z časti zakrývaná inými objektami. Pri tejto situácii došlo k udalosti, keď užívateľ zmenil počas pohybu okolo auta svoje identifikačné číslo. Predchádzajúce identifikačné číslo sa už ďalej na videu nevyskytovalo a preto bola situácia nesprávne vyhodnotená ako *false positive*.

Model na vystupovanie z auta mal počas testovania o niečo lepšie výsledky než model nastupovania do auta. Jeho matica zámen je zobrazená v tabuľke 8.3. Pomocou vzorca 5.3 na výpočet *accuracy* bola získaná výsledná úspešnosť vo výške 65%.

Tabuľka 8.3: Matica zámen pre model vystúpenia z auta

TP = 6	FN = 0
FP = 6	TN = 5

Rovnako ako pri modely na nastupovanie do auta boli správne detekované všetky situácie, pri ktorých človek vystúpil z auta. Problémy sa vyskytli pri videách, ktoré nemali spĺňať podmienky modelu. Tieto videa sa skladali zo situácii, kde sa osoba pohybovala len pri aute, alebo vedľa neho len stála, alebo osoba obišla celé auto. Taktiež sa situácie menili podľa farby oblečenia jednotlivých osôb, kedy ak farba oblečenia osoby, ktorá sa pohybovala v blízkosti auta, bola blízka farbe auta, vyskytol sa problém centralizácie, ktorý bol

popísaný v časti 6.5, kedy sa vymenil tracker medzi autom a osobou. Na konkrétnom zázname, kde sa vyskytol tento problém, nedošlo k nastúpeniu a ani vystúpeniu z auta, ale táto situácie sa môže vyskytnúť aj v budúcnosti, kde môže negatívne ovplyvniť výsledky týchto dvoch modelov.

Obr. 8.1: Príklad problému centralizácie

Príkladom situácie, ktorá bola identifikovaná ako *false positive* bolo, ak sa vozidlo nachádzalo na kraji kamerového záznamu. Následne sa osoba objavila v blízkosti auta a zároveň to bol jej prvý výskyt na skúmanom zázname a spĺňala všetky podmienky tohto modelu a preto bola nesprávne označená, ako úspešná detekcia vystúpenia z auta. Ďalším prípadom, kedy sa objavil nesprávny výsledok, bolo spôsobené priradením nového identifikačného čísla objektu, ktorý už bol na zázname identifikovaný. Tento problém sa taktiež vyskytol aj pri modeli nastúpenia do auta.

Pri testovaní modelu stretnutia dvoch ľudí sa testovali dve situácie, a to model v predvolených nastaveniach, teda so vzdialenosťou nastavenou na menšiu, alebo rovnú 0.5 metra a model s nastavenou vzdialenosťou na menšiu, alebo rovnú jeden meter. Maticu zámen pre prvú variantu je možné vidieť v tabuľke 8.4 a vypočítaná celková úspešnosť vyhľadávania bola 78% a úspešnosť správnej detekcie modelových stretnutí bola 64%. Na druhej strane, pri zväčšení vzdialenosti sa úspešnosť v obidvoch prípadoch zvýšila a to pri celkovej úspešnosti na 83%, čo bolo spôsobené žvýšením situácii, ktoré boli priradené do *true positive* a znížením *false negative* ako je možné vidieť v tabuľke 8.5. Úspešnosť modelu detekovať stretnutia do vzdialenosti menšej, alebo rovnej 1 meter sa zvýšila na 73%, viz. tabuľka 8.1.

Tabuľka 8.4: Matica zámen pre model stretnutia so vzdialenosťou menšou, rovnou 0.5 metra

TP = 7	FN = 4
FP = 0	TN = 7

Tabuľka 8.5: Matica zámen pre model stretnutia so vzdialenosťou menšou, rovnou 1 meter

TP = 8	FN = 3
FP = 0	TN = 7

Z týchto testov sa dá usúdiť, že sa tak stalo pretože počiatočná hodnota, ktorá bola nastavená na pol metra, nie je dostatočná. Dôvodom prečo môže byť to, že nie všetky stretnutia sú realizované na takúto vzdialenosť, ktorá môže byť označená za dôvernú.

Posledným testovaným modelom bola výmena tašky medzi dvomi osobami. Bol to jediný testovací model, ktorý zaplnil všetky štyri premenné zo vzorca 5.3, jeho matica zámen je zobrazená v tabuľke 8.6. Celková úspešnosť modelu bola 75%. Negatívne videá obsahovali rôzne situácie, pri ktorých napríklad došlo k objatiu medzi ľuďmi, takže taška bola v blízkosti druhej osoby, ako aj situácie, kedy osoba s taškou prudko mávala s taškou a zároveň okolo prechádzala druhá osoba. Obe situácie model úspešne zaradil ako *true negative*.

Schopnosť správneho detekovania výmeny tašky medzi dvomi osobami závisela od rôznych okolností, čo spôsobilo, že model bol úspešný na 67%. Jedným z výrazných elementov bolo aj to či sa osoba snažila schovať tašku pred kamerou alebo nie. Taktiež, ak taška splývala s oblečením niektorej z osôb, úspešnosť správneho detekovania tašky bola značne zhoršená. Na druhú stranu bol model schopný úspešnej detekcie aj v prípade, ak človek držal tašku na druhej strane, než ako smerovala kamera a teda bránil kamere vo výhľade na tašku svojím vlastným telom.

Tabuľka 8.6: Matica zámen pre model stretnutia so vzdialenosťou menšou, rovnou 1 meter

TP = 4	FN = 2
FP = 1	TN = 5

Za spomenutie taktiež stojí prípad, kedy sa na jednom mieste stretli dve osoby, ktorých oblečenie bolo v tmavých farbách, pričom jedna z nich mala tašku. Následne došlo k problému centralizácie, kedy jedna osoba na seba zobrala tracker tašky. Táto situácia bola následne nesprávne identifikovaná ako *false positive*.

Obr. 8.2: Príklad problému centralizácie

8.2 Testovanie za pomoci testrov

Testovanie použiteľnosti systému, jeho vzhľadu a jednoduchosti ovládania, bolo vykonané za pomoci štyroch dobrovoľníkov[1]. Každému dobrovoľníkovi bolo vysvetlené fungovanie systému ako aj prečo je dôležité použiť maticu homografie, ako funguje detekcie a následne sledovanie objektov.

Po odprezentovaní všetkých dôležitých informácii bol tester ponechaný, aby sám pozrel výslednú aplikáciu a naučil sa s ňou pracovať, pričom testrovi nebolo vysvetlené ovládanie aplikácie. Po uplynutí desiatich minút, tester zodpovedal prvú časť otázok, ktoré je možné vidieť na testovacom protokole v časti X.

V druhej časti mali testeri vykonať tri akcie:

- Nahratie videa a spustenie analýzy videa
- Vyhľadať udalosť: osoba vystúpil z červeného auta
- Vyhľadať udalosť: stretnutie dvoch ľudí na vzdialenosť 1 metra

Ku jednotlivým úlohám, ktoré mali splniť počas druhej časti testovania sa pri prvej úlohe, teda nahratí a následnej analýze videa, objavili výhrady, že by uvítali tlačidlo na zrušenie procesu analýzy, v prípade ak nahrali nesprávne video. Pri zadávaní dotazov ocenili postupné odomykanie jednotlivých blokov, ktoré boli jedným z faktorov prečo im systém prišiel jednoduchý na ovládanie. Priemerné bodové ohodnotenie zadávania dotazov dosiahlo hodnoty 2.25, pričom 1 bolo veľmi ľahké a 10 veľmi ťažké.

8.3 Budúci vývoj

Na zlepšenie rozpoznávania farby objektu, by bolo dobré rozšíriť dataset z doterajších 276 farieb aspoň na 400, pričom každý typ farby by mal aspoň 21 zástupcov. Vďaka tomu by sa znížilo riziko zlých detekcii. Aby sa zlepšila dostupnosť aplikácie, bolo by vhodné aby webové rozhranie bolo možné použiť aj z iných zariadení, než na ktorom sa nachádza serverová časť.

Aby sa zvýšila úspešnosť modelov vystupovania a nastupovania do auta, bolo by dobré pridanie detekcie dverí auta, najvhodnejšie by bolo ak by sa detekovalo otvorenie a zavretie dverí na aute. Týmto by sa odstránili nesprávne detekcie, ak užívateľ len prešiel okolo auta v prípade keď bolo auto na okraji záznamu.

V prípade modelu výmeny tašky, by bolo dobré pridať model, ktorý by detekoval ak taška zmizne potom, čo okolo osoby prešla iná osoba, vďaka čomu by sa detekovali aj situácie, kedy sa druhá osoba snažila schovať tašku pred bezpečnostnou kamerou.

[1] Prvý tester: 26 rokov, pokročilé znalosti práce s počítačom. Druhý tester: 57 rokov, mierne pokročilý. Tretí tester: 52 rokov, základné znalosti. Štvrtý tester: 21 rokov, pokročilé znalosti.

Kapitola 9

Záver

Jedným z cieľov tejto práce bolo pochopenie fungovania systémov používaných na analyzovanie videí z bezpečnostných kamier. Poznať skreslenie obrazu je nutnosťou každého systému spracovávajúceho videozáznamy, preto bolo dôležité pochopiť, ako sa pracuje s perspektívnou projekciou v počítačovom videní a ako sa získava matica homografie.

Žiaden systém by nebol úplný bez časti na detekciu a sledovanie objektov naprieč videom. Každej z týchto tém bola venovaná samostatná kapitola. Najdôležitejšie časti týchto kapitol sú tie, ktoré popisujú algoritmus YOLO a DeepSORT, ktorý tvoria jadro implementovaného systému.

V druhej časti tejto práce, boli použité teoretické znalosti z predchádzajúcej časti, pomocou ktorých bol následne vytvorený systém na analyzovanie záznamov. Výsledný systém pracuje so vzdialenosťou medzi jednotlivými objektami, ako aj s ich farbou a obsahuje päť modelov. Dotazy sa skladajú pomocou relácie medzi dvomi objektami. Jednotlivé objekty môžu byť ďalej bližšie definované pomocou ich farby. Taktiež je možné modifikovať vzdialenosti medzi skúmanými objektami.

Počas testovania sa ukázalo, že jednotlivé modely dosahujú rôzne úspešnosti. Testovanie modelov sa skladalo z dvoch častí, kedy sa najprv otestoval model na situáciách, kde došlo k hľadanej udalosti a následne boli k pôvodným testovacím videám pridané videá, ktorých úlohou bolo zmiasť tieto modely. Pri prvej časti testovania boli najúspešnejšie modely nastupovania a vystupovania z auta, kedy oba modeli dosiahli 100% úspešnosť. Avšak v druhej časti testovania dosahovali najnižších výsledkov. Model na vystupovanie z auta dosiahol 64% úspešnosť a model na nastupovanie do auta dokonca len 58% úspešnosť. Najlepšie počas testovania dopadol model na detekciu stretnutia medzi ľuďmi pri vzdialenosti jeden meter, kedy dokázal detekovať 73% situácii v prvej časti testovanie a jeho celková úspešnosť dosiahla 83% percent. Ako som však ukázala tieto nižšie percentá boli spôsobené určitými nedokonalosťami modelov, ktoré sa dajú v ďalších častiach vývoja. Napríklad pridaním detekcie otvorených a zatvorených dverí na aute by sa výrazne znížil počet nesprávnych detekcií modelov na vystupovanie a nastupovanie do auta.

Literatúra

[1] *Axxon Intellect's Video Analytics*. Dostupné z: https://www.axxonsoft.com/products/axxon_enterprise/video_analytics/.

[2] *Camio Social Distancing and Mask Detection - office floor plane*. Dostupné z: https://www.youtube.com/watch?v=OOlsxtUwMB8.

[3] *Documentation*. Dostupné z: https://doc.axxonsoft.com/confluence/display/ASdoc/Documentation.

[4] *Dopravní informační centrum*. Dostupné z: https://www.doprava-brno.cz/.

[5] *Features*. Dostupné z: https://miro.medium.com/max/644/1*SL7QfuqVAY70m-zO41RHuw.png.

[6] *Geometric Transformations of Images*. Dostupné z: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html.

[7] *A Guide to Video Analytics:: Applications and Opportunities*. Dostupné z: https://tryolabs.com/resources/video-analytics-guide/.

[8] *How Does Image Classification Work? mm*. Dostupné z: https://www.unite.ai/how-does-image-classification-work/.

[9] *IBM Intelligent Video Analytics documentation*. Dostupné z: https://www.ibm.com/docs/en/iva.

[10] *IBM's Intelligent Video Analytics packaged with Operation Center - Georgia Dome Demo*. Dostupné z: https://www.youtube.com/watch?v=yuSiKYuLV84&list=PLTfX2BbPgFZSYeg8BzJjqDvsm1SXkNPAy&index=16.

[11] *Integral image*. Dostupné z: https://media.vlpt.us/images/to2915ny/post/e749e0a1-2555-479a-af70-730f0461de91/image.png.

[12] *Perspective Drawing Unit*. Dostupné z: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ-RromiPUJqHhI7T5OLi1mPm8ZGH5bGiRswAVdGjPqDQOOKnpSmx9rXd3PKppJnZg_ti4&usqp=CAU.

[13] *Product architecture*. Dostupné z: https://www.ibm.com/docs/en/iva/3.0.0?topic=overview-product-architecture.

[14] *Put your cameras to work*. Dostupné z: https://www.camio.com/.

[15] *Real-time search. Long-term storage.* Dostupné z: https://www.camio.com/features.

[16] *Social Distancing & Mask Detection helps your workplace reopen safely.* Dostupné z: https://www.camio.com/sd.

[17] *Solution overview.* Dostupné z: https://www.ibm.com/docs/en/iva/3.0.0?topic=solution-overview.

[18] *Sum of grey rectangle.* Dostupné z: https://content.iospress.com/media/ida/2017/21-5/ida-21-5-ida163114/ida-21-ida163114-g003.jpg?width=755.

[19] *Tailgating Detection.* Dostupné z: https://www.camio.com/tailgating.

[20] *The VIRAT Video Dataset.* Dostupné z: https://viratdata.org/.

[21] *VIDEO SURVEILLANCE SYSTEM (VSS): STANDARD FOR BUILDINGS.* 1.0. Singapore, 2013.

[22] ImageNet classification with deep convolutional neural networks. *Communications of the ACM.* 2017-05-24, zv. 60, č. 6, s. 84–90. DOI: 10.1145/3065386. ISSN 0001-0782. Dostupné z: https://dl.acm.org/doi/10.1145/3065386.

[23] Abraham, T., Todd, A., Orringer, D. A. a Levenson, R. Chapter 7 - Applications of artificial intelligence for image enhancement in pathology. In: Cohen, S., ed. *Artificial Intelligence and Deep Learning in Pathology.* Elsevier, 2021, s. 119–148. DOI: https://doi.org/10.1016/B978-0-323-67538-3.00007-5. ISBN 978-0-323-67538-3. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780323675383000075.

[24] Adams, A. A. a Ferryman, J. M. The future of video analytics for surveillance and its ethical implications. *Security Journal.* 2015, zv. 28, č. 3, s. 272–289. DOI: 10.1057/sj.2012.48. ISSN 0955-1662. Dostupné z: http://link.springer.com/10.1057/sj.2012.48.

[25] Bewley, A., Ge, Z., Ott, L., Ramos, F. a Upcroft, B. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP).* IEEE. 2016, s. 3464–3468. DOI: 10.1109/ICIP.2016.7533003. Dostupné z: http://ieeexplore.ieee.org/document/7533003/.

[26] Burget, R. *Teoretická informatika.* Brno, 2013. Skripta. Vysoké učení technické v Brne.

[27] Castañón, G., Saligrama, V., Caron, A. L. a Jodoin, P. Real-Time Activity Search of Surveillance Video. In: *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance.* Sep. 2012, s. 246–251. DOI: 10.1109/AVSS.2012.58.

[28] Choe, T. E., Lee, M. W., Guo, F., Taylor, G., Yu, L. et al. Semantic video event search for surveillance video. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops).* IEEE. 2011, s. 1963–1970. DOI: 10.1109/ICCVW.2011.6130489. Dostupné z: http://ieeexplore.ieee.org/document/6130489/.

[29] Chu, D., Jiang, C. hua, Hao, Z. bo a Jiang, W. The Design and Implementation of Video Surveillance System Based on H.264, SIP, RTP/RTCP and RTSP. *2013 Sixth International Symposium on Computational Intelligence and Design*. IEEE. 2013, s. 39–43. DOI: 10.1109/ISCID.2013.124. Dostupné z: http://ieeexplore.ieee.org/document/6804823/.

[30] Comaniciu, D., Ramesh, V. a Meer, P. Real-time tracking of non-rigid objects using mean shift. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*. IEEE Comput. Soc. 2000, s. 142–149. DOI: 10.1109/CVPR.2000.854761. Dostupné z: http://ieeexplore.ieee.org/document/854761/.

[31] Dalal, N. a Triggs, B. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE. 2005, s. 886–893. DOI: 10.1109/CVPR.2005.177. Dostupné z: http://ieeexplore.ieee.org/document/1467360/.

[32] Dang, T. L., Nguyen, G. T. a Cao, T. OBJECT TRACKING USING IMPROVED DEEP_SORT_YOLOV3 ARCHITECTURE. *ICIC Express Letters*. 2020, zv. 14, č. 10, s. 961–969. Dostupné z: http://www.icicel.org/ell/contents/2020/10/el-14-10-04.pdf.

[33] Dubrofsky, E. *Homography Estimation*. Vancouver, 2009. Dizertačná práca. THE UNIVERSITY OF BRITISH COLUMBIA.

[34] Dubská, M., Sochor, J. a Herout, A. *Automatic Camera Calibrationfor Traffic Understanding*. Brno, 2014. Dizertačná práca. Brno University of Technology. Dostupné z: https://www.fit.vutbr.cz/~herout/papers/2014-BMVC-VehicleBoxes.pdf.

[35] England, A. *Zmatek matice a ROC křivka*. 2018. Dostupné z: https://i.stack.imgur.com/zoVvs.png.

[36] Gao, H. *Faster R-CNN Explained*. 27.09. 2017. Dostupné z: https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8.

[37] Gavali, P. a Banu, J. S. Chapter 6 - Deep Convolutional Neural Network for Image Classification on CUDA Platform. In: Sangaiah, A. K., ed. *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Academic Press, 2019, s. 99–122. DOI: https://doi.org/10.1016/B978-0-12-816718-2.00013-0. ISBN 978-0-12-816718-2. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780128167182000130.

[38] Ghoury, S., Sungur, C. a Durdu, A. Real-Time Diseases Detection of Grape and Grape Leaves using Faster R-CNN and SSD MobileNet Architectures. *International Conference on Advanced Technologies, Computer Engineering and Science*. 2019, s. 39–44. Dostupné z: https://www.researchgate.net/publication/334987612_Real-Time_Diseases_Detection_of_Grape_and_Grape_Leaves_using_Faster_R-CNN_and_SSD_MobileNet_Architectures.

[39] Hampapur, A., Brown, L., Feris, R., Senior, A., Chiao-Fe Shu et al. Searching surveillance video. In: *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. Sep. 2007, s. 75–80. DOI: 10.1109/AVSS.2007.4425289.

[40] Hampapur, A., Brown, L., Feris, R., Senior, A., Shu, C.-F. et al. Searching surveillance video. *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*. IEEE. 2007, s. 75–80. DOI: 10.1109/AVSS.2007.4425289. Dostupné z: http://ieeexplore.ieee.org/document/4425289/.

[41] Haque, M. *Multi Object Tracking*. Rajshahi University of Engineering and Technology, 2015. Dostupné z: https://image.slidesharecdn.com/103001cse820p1-160606065212/95/multi-object-tracking-presentation-1-id-103001-16-638.jpg?cb=1465196340.

[42] Iqbal, A., Arif, F. a Minallah, N. Analyzing impact of video codec, encapsulation methods and streaming protocols on the quality of video streaming. In: *Eighth International Conference on Digital Information Management (ICDIM 2013)*. Sep. 2013, s. 182–186.

[43] Islam, M. Z., Islam, M. S. a Rana, M. S. Problem Analysis of Multiple Object Tracking System: A Critical Review. *IJARCCE*. 2015-11-30, zv. 4, č. 11, s. 374–377. DOI: 10.17148/IJARCCE.2015.41183. ISSN 22781021. Dostupné z: http://ijarcce.com/upload/2015/november-15/IJARCCE83.pdf.

[44] Jensen, O. H. Implementing the Viola-Jones Face Detection Algorithm. Kongens Lyngby: Technical University of Denmark. 2008, s. 1–36. ISSN 1601-233X. Dostupné z: https://www.yumpu.com/en/document/view/18418164/implementing-the-viola-jones-face-detection-algorithm.

[45] Joshi, K. A. a Thakore, D. G. A Survey on Moving Object Detection and Tracking in Video Surveillance System. *International Journal of Soft Computing and Engineering*. 2012, zv. 2, č. 3, s. 44–48. ISSN 2231-2307. Dostupné z: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.645.7492&rep=rep1&type=pdf.

[46] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. 1960-03-01, zv. 82, č. 1, s. 35–45. DOI: 10.1115/1.3662552. ISSN 0021-9223. Dostupné z: https://asmedigitalcollection.asme.org/fluidsengineering/article/82/1/35/397706/A-New-Approach-to-Linear-Filtering-and-Prediction.

[47] Kothiya, S. V. a Mistree, K. B. A Review on Real Time Object Tracking in Video Sequences. 2015. Dostupné z: https://www.researchgate.net/profile/Kinjal-Mistree-2/publication/308673323_A_review_on_real_time_object_tracking_in_video_sequences/links/5c8a5b27a6fdcc381754005c/A-review-on-real-time-object-tracking-in-video-sequences.pdf.

[48] Kováč, O. *Úvod do číslicového spracovania obrazov*. Máj 2020. ISBN 978-80-553-3521-6.

[49] Le, T.-L., Thonnat, M., Boucher, A. a Brémond, F. A Query Language Combining Object Features and Semantic Events for Surveillance Video Retrieval. *Advances in Multimedia Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008, s. 307–317. DOI: 10.1007/978-3-540-77409-9_29. Dostupné z: http://link.springer.com/10.1007/978-3-540-77409-9{_}29.

[50] Lin, M., Chen, Q. a Yan, S. Network In Network. 2013, s. 1–10. Dostupné z: https://arxiv.org/abs/1312.4400v3.

[51] Luo, L.-B., Koh, I.-S., Min, K.-Y., Wang, J. a Chong, J.-W. Low-cost implementation of bird's-eye view system for camera-on-vehicle. *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*. IEEE. 2010, s. 311–312. DOI: 10.1109/ICCE.2010.5418845. Dostupné z: http://ieeexplore.ieee.org/document/5418845/.

[52] Malis, E. a Vargas, M. *Deeper understanding of the homography decomposition for vision-based control*. Research Report RR-6303. INRIA, 2007. 90 s. Dostupné z: https://hal.inria.fr/inria-00174036.

[53] Mittal, K. *A Gentle Introduction Into The Histogram Of Oriented Gradients*. Dostupné z: https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa.

[54] Padilla, R., Netto, S. L. a Silva, E. A. B. da. A Survey on Performance Metrics for Object-Detection Algorithms. *A Survey on Performance Metrics for Object-Detection Algorithms*. 2020, s. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130. Dostupné z: https://ieeexplore.ieee.org/document/9145130.

[55] Quast, K. a Kaup, A. Scale and shape adaptive mean shift object tracking in video sequences. *European Signal Processing Conference*. IEEE. 2009, zv. 17, s. 1513–1517. Dostupné z: https://ieeexplore.ieee.org/abstract/document/7077601.

[56] Redmon, J., Divvala, S., Girshick, R. a Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, s. 779–788.

[57] Ren, S., He, K., Girshick, R. a Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017-6-1, zv. 39, č. 6, s. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/7485869/.

[58] Simon, H. Kalman Filters. In: *Kalman Filtering and Neural Networks*. John Wiley and Sons, Ltd, 2001, kap. 1, s. 1–21. DOI: https://doi.org/10.1002/0471221546.ch1. ISBN 9780471221548. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471221546.ch1.

[59] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. et al. Going Deeper With Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, s. 1–9.

[60] Vasuhi, S., Vijayakumar, M. a Vaidehi, V. Real time multiple human tracking using Kalman Filter. *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*. IEEE. 2015, s. 1–6. DOI: 10.1109/ICSCN.2015.7219902. Dostupné z: http://ieeexplore.ieee.org/document/7219902/.

[61] Viola, P. a Jones, M. J. Robust Real-Time Face Detection. *International Journal of Computer Vision*. 2004, zv. 57, č. 2, s. 137–154. DOI: 10.1023/B:VISI.0000013087.49260.fb. ISSN 0920-5691. Dostupné z: http://link.springer.com/10.1023/B:VISI.0000013087.49260.fb.

[62] Weng, L. *Object Detection for Dummies Part 1:: Gradient Vector, HOG, and SS*. Dostupné z: https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html.

[63] Yi, D. Kalman Filter. 1977, s. 1–7. Dostupné z: http://info.dqyi.org/files/lecture/note/KalmanFilter.pdf.

[64] Yilmaz, A., Javed, O. a Shah, M. Object tracking. *ACM Computing Surveys*. 2006-12-25, zv. 38, č. 4, s. 1–44. DOI: 10.1145/1177352.1177355. ISSN 0360-0300. Dostupné z: https://dl.acm.org/doi/10.1145/1177352.1177355.

[65] Zhang, Z. Estimating Projective Transformation Matrix: (Collineation, Homography). *Microsoft Research Techical Report MSR-TR-2010-63*. s. 1–10. Dostupné z: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-63.pdf.

[66] Zou, Z., Shi, Z., Guo, Y. a Ye, J. Object Detection in 20 Years: A Survey. 2019, s. 1–39. Dostupné z: https://arxiv.org/abs/1905.05055.

Príloha A

Testovací protokl

Základné informácie o testrovi

☐ Vek

☐ Úroveň počítačových znalostí

- Základné
- Mierne pokročilé
- Pokročilé

Prvá sada otázok, po 10 minutách na zoznámenie so systémom

☐ Ako na vás pôsobí systém?

☐ Myslíte si, že aplikácia obsahuje dostatočné množstvo častí?

☐ Myslíte si, že viete systém ovládať?

Praktická časť testovania

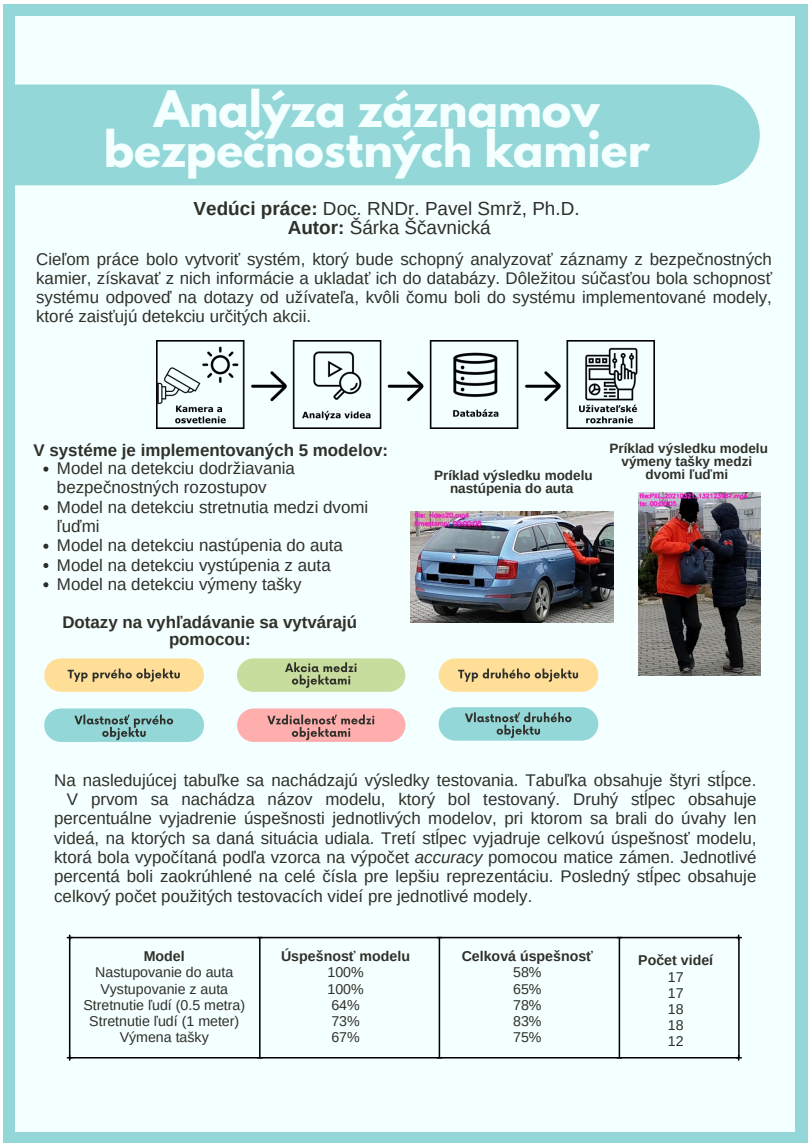
1. Vložte video a spustite jeho analýzu
   - ☐ Aké sú vaše dojmy z procesu vloženia a analýzi videa?
   - ☐ Mali ste problém so zadávaním ROI (Region of interest)
2. Vyhľadajte udalosť, kedy došlo k stretnutiu dvoch ľudí na vzdialenosť menšiu rovnú jednému metru
   - ☐ Aké sú vaše dojmy z procesu vloženia a analýzi videa?
   - ☐ Ste spokojný so spôsobom zobrazovania výsledkov?
3. Vyhľadajte udalosť, kedy osoba vystúpila z červeného auta
   - ☐ Aké sú vaše dojmy z procesu vloženia a analýzi videa?
   - ☐ Ste spokojný so spôsobom zobrazovania výsledkov?

Záverečná časť testovania

- ☐ Ako hodnotíte ovládanie systému na stupnici od 1 po 10, kde jedna je veľmi ľahké a 10 je veľmi ťažké?
- ☐ Ako ťažké pre vás bolo pochopenie zadávania dotazov? Použite stupnicu z predchádzajúcej otázky.
- ☐ Aký je celkový dojem z uživateľského rozhrania? Prosím použite stupnicu od 1 po 10 kde 1 je veľmi dobré a 10 je veľmi zlé.
- ☐ Zmenili by ste niečo?

Príloha B

Plagát



Obr. B.1: Plagát prezentujúci túto prácu