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

SLEDOVANIE POHYBLIVÝCH OBJEKTOV VO VIDEU

TRACKING OF MOVING OBJECTS IN VIDEO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JÁN FOLENTA**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání bakalářské práce

Student: **Folenta Ján**
Program: Informační technologie
Název: **Sledování pohyblivých objektů ve videu**
Tracking of Moving Objects in Video
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou sledování objektů ve videu, s dostupnými datovými sadami a s metrikami pro hodnocení algoritmů sledování objektů.
2. Vyberte vhodnou metodu a implementujte ji.
3. Experimentujte s implementovanou metodou v různých variantách a na různých datových sadách.
4. Navrhněte vhodné aplikace zvolené metody a demonstrujte její použitelnost na vhodných datech.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 a 4.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 5. listopadu 2019

Abstrakt

Táto bakalárska práca sa zaoberá problematikou detekcie, sledovania a počítania vozidiel pohybujúcich v jednotlivých smeroch vo videu. Pre riešenie problému sú využité moderné techniky detekcie a sledovania objektov využívajúce konvolučné neurónové siete. Cieľom práce je dosiahnutie čo najväčšej presnosti počítania vozidiel pri zachovaní spracovania videozáznamov v reálnom čase. Problémy implementovanej metódy pre detekciu a sledovanie sú riešené analýzou a prácou s trajektóriami vozidiel. S úspešnosťou 90,94% a s celkovým skóre 0,8829 sa táto práca zúčastnila súťaže AI City Challenge 2020, kde sa umiestnila na konečnom 6. mieste.

Abstract

This bachelor thesis deals with the issue of detection, tracking and counting vehicles in different directions in video. To deal with this problem, modern techniques of object detection and tracking using convolutional neural networks are used. The goal of this work is to achieve highest possible accuracy of vehicle counting while maintaining the processing of video recordings in real-time. The problems of the implemented method for detection and tracking are solved by analyzing and working with the trajectories of vehicles. With accuracy of 90,94% and total score of 0,8829, this work participated in AI City Challenge 2020, where it placed 6th.

Kľúčové slová

detekcia objektov, sledovanie objektov, CNN, YOLOv3, Deep SORT, AI City Challenge, trajektórie

Keywords

object detection, object tracking, CNN, YOLOv3, Deep SORT, AI City Challenge, trajectories

Citácia

FOLENTA, Ján. *Sledovanie pohyblivých objektov vo videu.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Sledovanie pohyblivých objektov vo videu

Prehlásenie

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

.....
Ján Folenta
28. mája 2020

Poďakovanie

Touto formou by som sa chcel poďakovať pánovi prof. Ing. Adamovi Heroutovi Ph.D., za jeho cenné rady a odborné vedenie práce, pánovi Ing. Jakubovi Špaňhelovi za spoluprácu a pomoc pri vykonávaní experimentov a svojej rodine za neustálu podporu.

Obsah

Kapitola 1

Úvod

Táto práca sa zaoberá detekciou a sledovaním pohybujúcich sa objektov na scénach zaznamenaných zo statickej kamery. V prvých fázach bola práca zameraná na sledovanie ľudí a vozidiel, neskôr, po konzultácii s vedúcim tejto bakalárskej práce pánom prof. Ing. Adamom Heroutom Ph.D., sme sa rozhodli, že sa práca prostredníctvom výskumnej skupiny Graph@FIT zúčastní súťaže AI City Challenge 2020 a z toho dôvodu je práca neskôr zameraná najmä na sledovanie vozidiel.

Cieľom riešenej úlohy v tejto súťaži je navrhnúť a implementovať systém pre sledovanie a počítanie osobných a nákladných vozidiel pohybujúcich sa v jednotlivých smeroch. Výsledný systém by mohol mať využitie v analýze dopravy a mohol by pomôcť k lepšiemu plánovaniu časovania križovatiek, prípadne k výberu vhodnej stratégie pre zmiernenie dopravného zaťaženia. V systéme pre sledovanie a analýzu dopravy je dôležitá okrem presnosti aj rýchlosť spracovania. V ideálnom prípade by mal takýto systém dosahovať spracovanie v reálnom čase, preto je dôležité vybrať pre detekciu a sledovanie metódu, ktorá bude zohľadňovať obidva aspekty. Z toho dôvodu je pre riešenie problému použitý detektor YOLOv3 v kombinácii so sledovacou metódou Deep SORT. Aj napriek prudkému vývoju v posledných rokoch majú súčasné systémy pre detekciu a sledovanie objektov nedostatky. Medzi najznámejšie problémy patrí prekrytie objektu na dlhší čas, zámena identity alebo nepresnosť detektora. Hlavnou myšlienkou tejto práce je riešiť spomínané problémy na základe trajektórií sledovaných vozidiel. Pomocou trajektórií vozidiel, ktoré reprezentujú predchádzajúci pohyb vozidiel je možné napríklad predpovedať ďalší pohyb vozidla, re-identifikovať vozidlo, ktoré bolo prekryté na dlhší čas a tiež predpovedať pohyb vozidla predtým, ako bolo prvýkrát detegované.

V kapitole 2 sú popísané súčasné metódy, ktoré sa zaoberajú problematikou detekcie a sledovania objektov používajúce konvolučné neurónové siete. Návrh systému na počítanie vozidiel, výber metód pre detekciu a sledovanie a popis dátových sád je uvedený v kapitole 3. Práca v spomínanej súťaži AI City Challenge 2020 s prispením Ing. Jakuba Špaňhela dosiahla celkové skóre 0,8829 a umiestnila sa na konečnom 6. mieste. Bližšie informácie o implementácii systému, jednotlivých experimentoch a výsledkoch sú popísané v kapitole 4.

Kapitola 2

Detekcia a sledovanie objektov s použitím neurónových sietí

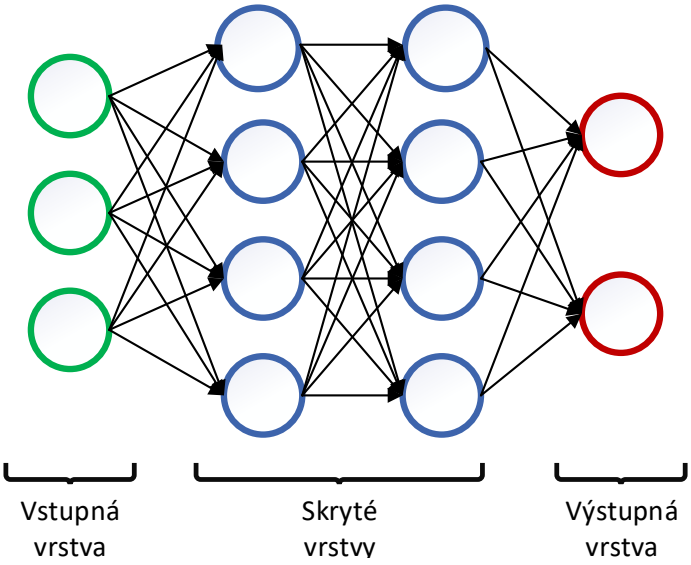
V posledných rokoch dosiahla detekcia objektov značný pokrok najmä vďaka využívaniu neurónových sietí. Cieľom tejto práce je využiť moderné techniky detekcie a sledovania objektov využívajúce konvolučné neurónové siete. Súčasné techniky detekcie objektov je možné rozdeliť na dve základné kategórie, a to na detektory založené na regiónoch a jednokrokové detektory. Z obidvoch kategórii budú popísané princípy ich najznámejších predstaviteľov. V prvom rade však bude vysvetlené, čo sú neurónové siete a aký je ich princíp, pričom dôraz bude kladený na konvolučné neurónové siete. V závere kapitoly budú popísané metódy pre sledovanie objektov.

2.1 Neurónové siete

Neurónová sieť je matematický model inšpirovaný biologickými neurónovými sieťami, v ktorých si jednotlivé neuróny medzi sebou vymieňajú informácie. Neurónové siete majú širokú škálu použitia a vo vybraných úlohách dosahujú relatívne vysokú úspešnosť. Svoje použitie našli aj v počítačovom videní, kde sú neurónové siete používané pre riešenie zložitejších úloh, akými sú napríklad detekcia obrazu alebo klasifikácia objektov.

Klasická neurónová sieť [2] je zložená z troch typov vrstiev – zo vstupnej vrstvy, z jednej alebo viacerých skrytých vrstiev a z výstupnej vrstvy. Vstupná vrstva prijíma vstupy v rôznych formátoch, akými sú napríklad pixely obrazu, čísla alebo zvuk. Skrytá vrstva zodpovedá za spracovanie dát, matematické výpočty a podobne. V neurónovej sieti nie je nutná prítomnosť skrytých vrstiev a taká sieť potom obsahuje iba jednu vstupnú a jednu výstupnú vrstvu. V prípade, že sieť obsahuje aspoň jednu skrytú vrstvu, hovoríme o hlbokej neurónovej sieti (Deep Neural Network). Na konci sa nachádza jedna výstupná vrstva, ktorá generuje požadovaný výstup. Všeobecný model neurónovej siete je zobrazený na obrázku 2.1.

Každá vrstva obsahuje určitý počet neurónov, ktoré sú prepojené s neurónmi z predchádzajúcej vrstvy. Príklad neurónu je zobrazený na obrázku 2.2. Vstupom neurónu je vektor signálov, pričom každý signál má svoju váhu, ktorá vyjadruje, aký vplyv má daný signál na výstup neurónu. K súčtu signálov vynásobených ich váhami je pripočítaný posun, nazývaný tiež aj bias, a výsledok je následne argumentom aktivačnej funkcie. Matematicky je možné neurón vyjadriť vzťahom



Obr. 2.1: Všeobecný model neurónovej siete, ktorý je zložený zo vstupnej vrstvy, dvoch skrytých vrstiev a výstupnej vrstvy.

$$y_k = \varphi\left(\sum_{i=1}^{N} x_i w_i + b\right) \tag{2.1}$$

kde x_i je vstupný signál, w_i je váha vstupného signálu, b označuje bias a φ je aktivačná funkcia.

Základným princípom aktivačnej funkcie je zavedenie nelinearity do výpočtu siete. Poznáme viacero rôznych aktivačných funkcií. Medzi tie najznámejšie patrí sigmoid, ReLU alebo pri klasifikácii objektov často používaný softmax.

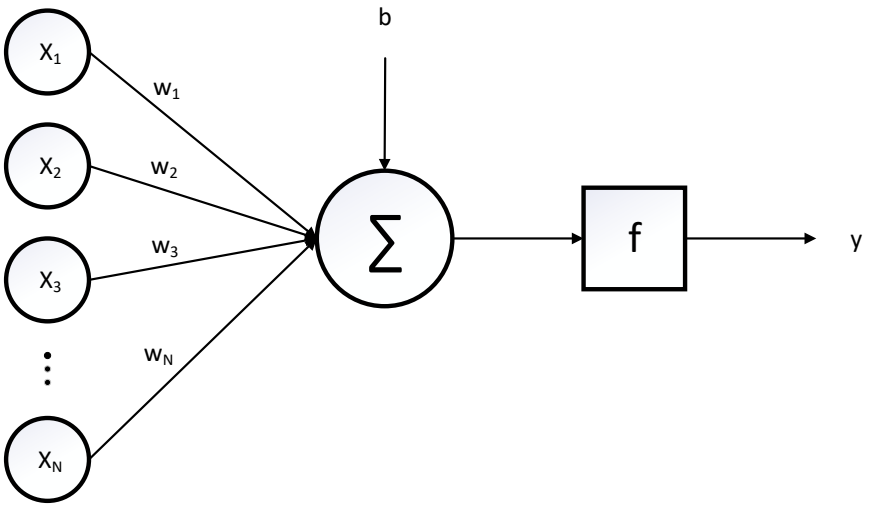
2.1.1 Konvolučné neurónové siete

Pre spracovanie obrazu či videa sa používajú konvolučné neurónové siete. Konvolučné siete [23] sú schopné zachytávať priestorové závislosti v obraze aplikovaním vhodných filtrov. Úlohou konvolučnej siete je transformovať obraz do formy, ktorá je ľahšia pre spracovanie bez straty príznakov obrazu, ktoré sú dôležité pre dobrú predpoveď.

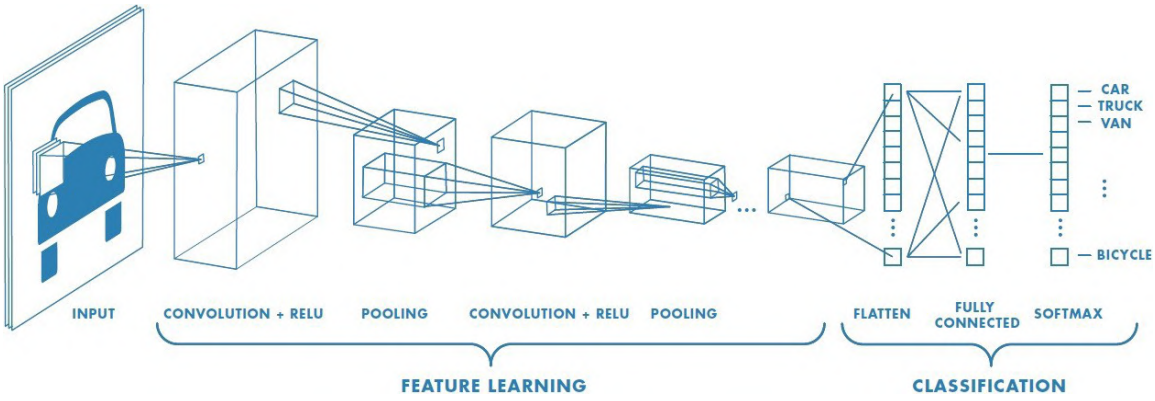
Konvolučná neurónová sieť [2] je zložená z jednej alebo viacerých konvolučných vrstiev, medzi ktorými sa nachádzajú združovacie vrstvy a na konci siete zvyčajne nasleduje jedna alebo viaceré plne prepojené vrstvy. Príklad konvolučnej neurónovej siete je možné vidieť na obrázku 2.3. Oproti klasickej neurónovej sieti obsahujú konvolučné siete menej prepojení a parametrov, čo umožňuje jednoduchšie trénovanie siete. Klasickým vstupom siete je obrázok, ktorý má 3 dimenzie – výšku, šírku a hĺbku, kde hĺbka je zvyčajne reprezentovaná troma farebnými kanálmi RGB.

Konvolučná vrstva

Základom konvolučných neurónových sietí sú konvolučné vrstvy, ktorých úlohou je extrahovať príznaky obrazu. Konvolučné vrstvy pozostávajú zo sady konvolučných filtrov nazývaných tiež aj kernel. Filter je reprezentovaný ako matica menších rozmerov, zvyčajne 3×3



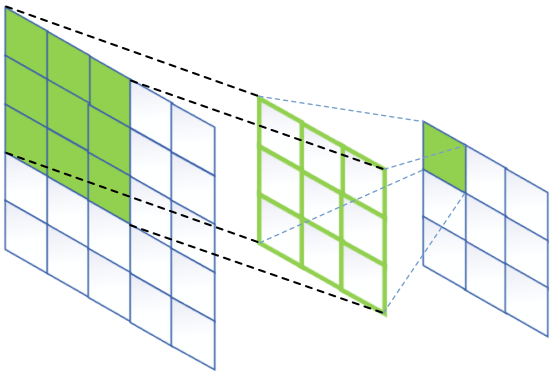
Obr. 2.2: Všeobecný model neurónu.



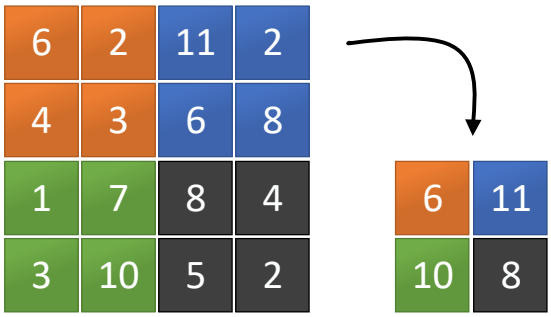
Obr. 2.3: Príklad konvolučnej neurónovej siete klasifikujúcej objekt zo vstupného obrázka. Prevzaté z [23].

alebo 5×5, a musí prechádzať všetkými kanálmi vstupného obrazu, preto má zvyčajne hĺbku 3. Každý filter je následne posúvaný po matici vstupného obrazu. Časť matice, nad ktorou sa práve filter nachádza, je vynásobená maticou filtra a je vyprodukovaná jedna hodnota. Grafické znázornenie je zobrazené na obrázku 2.4. Po prejdení celej matice obrazu je vytvorená mapa príznakov, ktorá je zároveň aj výstupom konvolučnej vrstvy. Každý filter produkuje samostatnú mapu príznakov, čo znamená, že konvolučná vrstva produkuje typicky sadu máp príznakov.

Konvolučná neurónová sieť zvyčajne pozostáva z viacerých konvolučných vrstiev. Prvé z nich sú zodpovedné za extrakciu príznakov nižšej úrovne, akými sú napríklad farba alebo hrany. Posledné konvolučné vrstvy extrahujú príznaky vyššej úrovne, ako rozpoznanie tried či oblastí.



Obr. 2.4: Ukážka konvolúcie vstupnej matice konvolučným filtrom o veľkosti 3×3.



Obr. 2.5: Ukážka výpočtu max pooling. Z každého okna 2×2 je vybraná najväčšia hodnota.

Združovacia vrstva

Združovacia vrstva, tiež nazývaná aj subsampling, znižuje priestorovú veľkosť mapy príznakov, pričom si ponecháva dôležité informácie. Zvyčajne sa nachádza medzi konvolučnými vrstvami. Hlavným cieľom vrstvy je redukovať množstvo parametrov siete, čím sa znižuje aj výpočtová náročnosť požadovaná pre spracovanie celého obrazu. Redukcia prebieha najčastejšie pomocou max pooling alebo average pooling. V praxi dosahuje najlepšie výsledky max pooling (obrázok 2.5), pri ktorom je definované okno o veľkosti $S \times S$ a z daného okna je vybraná najvyššia hodnota. Pri average pooling je z hodnôt v okne vypočítaná priemerná hodnota. Združovacia vrstva nemení hĺbku siete.

Plne prepojená vrstva

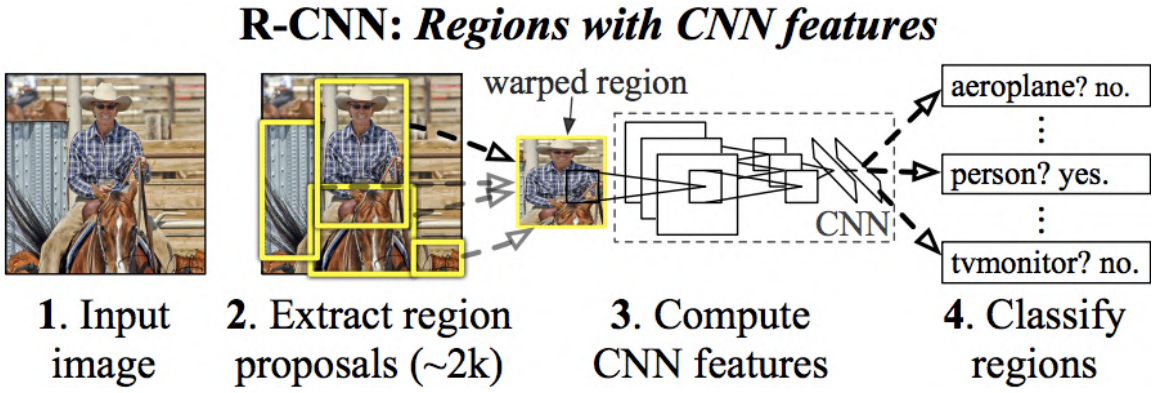
Na konci konvolučných neurónových sietí sa nachádza niekoľko po sebe idúcich plne prepojených vrstiev, ktoré sú spravidla zodpovedné za klasifikáciu objektov v obraze. Neuróny v tejto vrstve sú prepojené so všetkými neurónmi v predchádzajúcej vrstve. Vstupom plne prepojených vrstiev je jednodimenzionálny vektor príznakov a počet výstupov je rovný počtu tried, ktoré má daná konvolučná neurónová sieť klasifikovať.

2.2 Detektory založené na regiónoch

Činnosť detektorov založených na regiónoch je možné rozdeliť do dvoch fáz. V prvej fáze sa vygenerujú návrhy regiónov (region proposals), v ktorých je predpoklad, že sa vyskytuje objekt. V druhej fáze sa následne pomocou konvolučnej neurónovej siete detegujú a klasifikujú objekty v daných regiónoch. Systémy založené na tejto metóde sa vyznačujú relatívne vysokou presnosťou, avšak sú výpočtovo náročné, pretože je nutné prechádzať každý vygenerovaný región.

2.2.1 R-CNN

Systém R-CNN [8] je prvým známym systémom využívajúcim pre detekciu neurónové siete. Celý systém je zložený z troch základných modulov. Schému systému je možné vidieť na obrázku 2.6.



Obr. 2.6: Schéma systému R-CNN. Obrázok je prevzatý z [8].

Prvý modul generuje návrhy regiónov. Existuje viacero metód generovania návrhov regiónov, ktoré je možné v R-CNN použiť. V poskytnutom systéme je použitá metóda selektívneho vyhľadávania (selective search). Selektívne vyhľadávanie [25] je algoritmus, ktorý skúma obrázok na základe okien rôznych veľkostí a snaží sa zgrupovať susedné pixely podobné svojou farbou, textúrou alebo intenzitou za účelom nájdenia objektu. Výsledkom selektívneho vyhľadávania je približne 2000 regiónov.

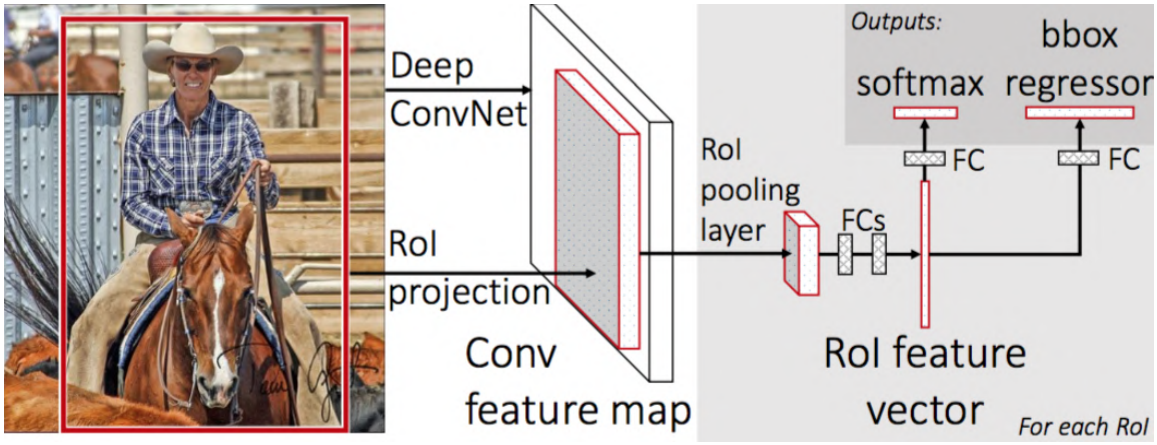
Druhý modul je veľká konvolučná neurónová sieť, ktorej výstupom pre každý región je vektor príznakov (feature vector) pevnej dĺžky. R-CNN používa neurónovú sieť AlexNet [12]. Každý región je upravený na veľkosť s rozlíšením 227×227 pixelov a poslaný do konvolučnej neurónovej siete, ktorá vyextrahuje 4096-dimenzionálny vektor príznakov.

Posledným modulom je sada lineárnych SVM (Support Vector Machine). SVM [6] je klasifikačný algoritmus, ktorého cieľom je nájsť hyperrovinu v N-rozmernom priestore (kde N je počet príznakov), ktorý jednoznačne klasifikuje dátové body. Pre každú triedu sa použitím SVM natrénovaného pre danú triedu ohodnotí vektor príznakov, ktorý je výstupom konvolučnej neurónovej siete. Po získaní všetkých ohodnotení regiónov je pre každú triedu nezávisle aplikované potlačenie nemaxím, pri ktorom sa z ohraničujúcich obdĺžnikov, ktoré sa prekrývajú, ponechá tá s najväčším skóre a vyprodukujú sa konečné detekcie.

R-CNN dosahuje presnosť 66% mAP, avšak je schopný spracovať iba 0,03 snímky za sekundu.

2.2.2 Fast R-CNN

Hlavným nedostatkom detektora R-CNN je jeho rýchlosť, keďže konvolučná neurónová sieť je spúšťaná nad každým z 2000 regiónov, a jeden obrázok sa tak spracováva približne 47 sekúnd. Preto sa autori snažili vylepšiť danú metódu a predstavili metódu Fast R-CNN [7], ktorá je 213-krát rýchlejšia ako metóda R-CNN a tiež o niečo presnejšia. Miesto siete AlexNet je v tejto metóde použitá sieť VGG16 [11]. Vstupom siete VGG16 je celý obrázok a sada návrhov objektov. Sieť najprv spracuje celý obrázok niekoľkými konvolučnými vrstvami a vyprodukuje mapu príznakov. Pomocou techniky nazývanej Region of Interest Pooling, ktorá zdieľa vyprodukovanú mapu príznakov, je z danej mapy príznakov vybraná príslušná oblasť a z nej je následne vyextrahovaný vektor príznakov pevnej dĺžky. Každý vektor príznakov potom prechádza sekvenciou plne prepojených vrstiev, ktoré sa nakoniec vetvia do dvoch súrodeneckých výstupných vrstiev. Prvá vrstva odhaduje pravdepodobnosti



Obr. 2.7: Schéma systému Fast R-CNN. Obrázok je prevzatý z [7].

výskytu tried objektov v daných regiónoch. Druhá vrstva produkuje štyri hodnoty, ktoré kódujú pozície ohraničujúcich obdĺžnikov pre jednotlivé objekty.

V metóde Fast R-CNN teda odpadáva nutnosť púšťať konvolučnú neurónovú sieť nad každým regiónom, čo sa podpísalo v znížení času potrebného pre spracovanie obrázka na 0,3 sekundy. Úspešnosť na dátovej sade Pascal VOC 2012 bola oproti metóde R-CNN zvýšená o 4%.

2.2.3 Faster R-CNN

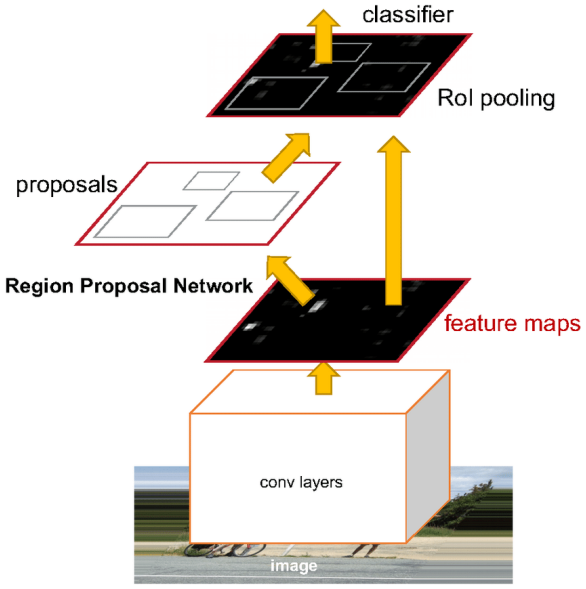
Ďalšou modifikáciou a vylepšením vyššie spomínaných metód je metóda Faster R-CNN [22]. V predchádzajúcich metódach sa pre návrh regiónov používala metóda selektívneho vyhľadávania, ktorá nie je príliš rýchla a nájdenie regiónov pre každý snímok v implementácii na CPU trvá 2 sekundy. Základnou myšlienkou nového systému je navrhovať regióny pomocou hlbokej neurónovej siete, ktorá bude zdieľať konvolučné vrstvy so sieťou pre detekciu, a tým sa nielen zvýši rýchlosť celého systému, ale spôsobí to aj zlepšenie v reprezentácii príznakov.

Systém je teda možné rozdeliť na dva moduly. Prvý modul je hlboká konvolučná neurónová sieť, ktorá navrhuje regióny, v ktorých by sa mohol nachádzať objekt, a druhým modulom je samotný Fast R-CNN detektor popísaný vyššie, ktorý využíva dané regióny pre detekciu objektov. Celý systém tak tvorí jednu sieť. Vstupom siete pre návrh regiónov je obrázok akejkoľvek veľkosti a jej výstupom je sada ohraničujúcich obdĺžnikov vrátane ohodnotenia jednotlivých tried objektov, ktoré sa v daných obdĺžnikoch môžu nachádzať.

Čas potrebný pre nájdenie regiónov sa teda znížil z 2 sekúnd na snímku, ako tomu bolo pri Fast R-CNN, na 10 milisekúnd na snímku. Z rodiny detektorov založených na regiónoch dosahuje metóda Faster R-CNN najvyššiu presnosť a rovnako aj najvyššiu rýchlosť.

2.3 Jednokrokové detektory

Jednokrokové metódy založené na regresii mapujú pixely obrazov priamo na súradnice ohraničujúcich obdĺžnikov a pravdepodobnosti zaradenia objektov do jednotlivých tried. Na rozdiel od metód založených na regiónoch pozostávajúcich z viacerých častí, ktoré je zvyčajne nutné trénovať samostatne, jednokrokové metódy je možné trénovať ako celok. Vďaka tomu, že jednokrokovým metódam stačí na detegovanie a lokalizáciu objektov je-



Obr. 2.8: Architektura systému Faster R-CNN. Prevzaté z [22].

den prechod sieťou, dosahujú tieto metódy vyššiu rýchlosť spracovania obrazu ako metódy založené na regiónoch, pričom ich presnosť je porovnateľná.

2.3.1 You only look once – YOLO

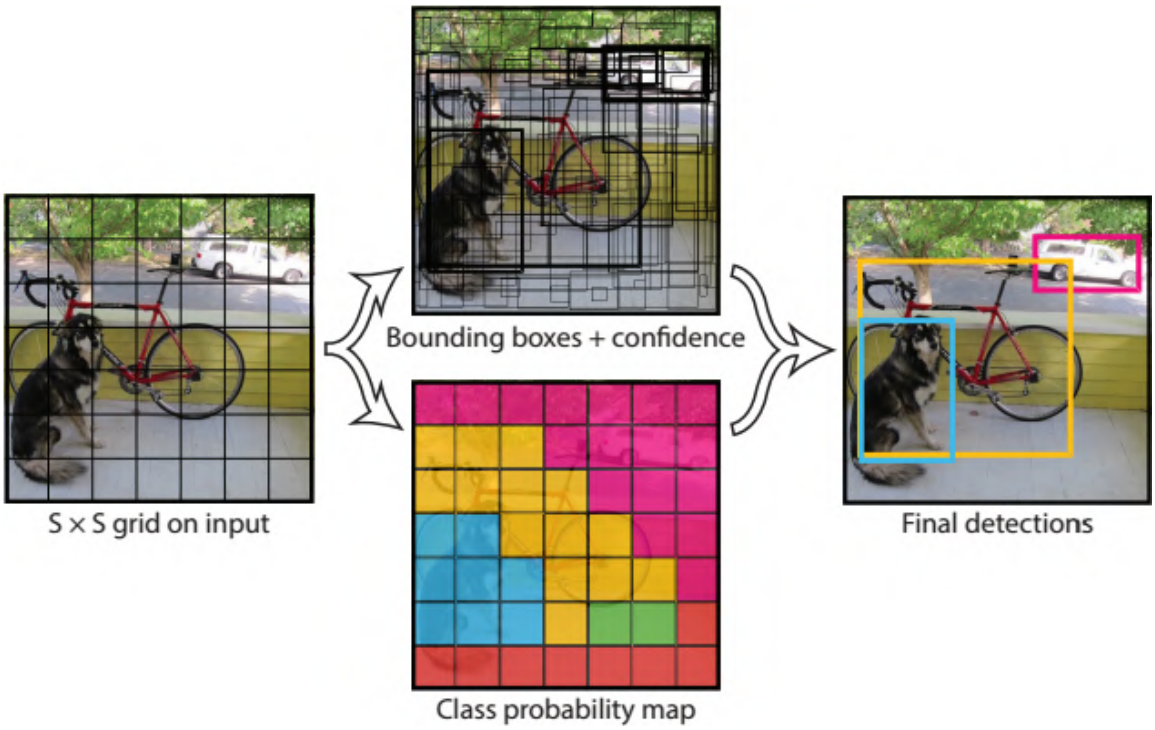
Systém YOLO [19] tvorí jedna konvolučná sieť, ktorá predpovedá ohraničujúce obdĺžniky a pravdepodobnosť triedy objektu, ktorá sa nachádza v danom obdĺžniku. Vstupný obrázok je najprv rozdelený na mriežku o veľkosti $S \times S$. V prípade, že stred objektu spadá do bunky danej mriežky, je táto bunka zodpovedná za detekciu objektu. Každá bunka mriežky predpovedá niekoľko ohraničujúcich obdĺžnikov a tiež skóre spoľahlivosti pre dané obdĺžniky, ktoré predstavuje mieru istoty, že obdĺžnik obsahuje objekt. V každej bunke sú tiež predpovedané pravdepodobnosti, do akej triedy objekt v danej bunke patrí. Tieto pravdepodobnosti sú však podmienené tým, či sa v danej bunke nachádza objekt.

Skutočnosť, že YOLO spracúva celé obrázky má niekoľko výhod. Vďaka tomu, že nie je nutné komplexné zreťazené spracovanie (pipeline), je YOLO detektor veľmi rýchly. Na rozdiel od posuvného okna a metód založených na návrhoch regiónov, YOLO vidí počas trénovacieho a testovacieho času celý obraz, čo umožňuje implicitne kódovať kontextové informácie o triedach a ich vzhľade. V porovnaní s Fast R-CNN sa YOLO dopúšťa o polovicu menej chýb, čo sa týka detegovania pozadia.

Síce YOLO rýchlo deteguje objekt, má ale problémy pri presnom určení polohy niektorých objektov, najmä tých malých. Na GPU beží rýchlosťou 150 snímok za sekundu, čo ho radí medzi najrýchlejšie metódy.

YOLOv2

S cieľom zvýšiť presnosť systému YOLO pri zachovaní jeho rýchlosti bola predstavená modifikácia systému YOLO, nazývaná YOLOv2 [20]. Jedným z vylepšení nového systému je zavedenie normalizácie dávok (batch normalization). Normalizácia dávok vedie k význam-



Obr. 2.9: Schéma práce detekčnej metódy YOLO. Obrázok bol prevzatý z [19].

nému zlepšeniu konvergencie a zároveň je odstránená potreba ďalších foriem regularizácie. Pridaním normalizácie dávok do všetkých konvolučných vrstiev siete YOLO bola zlepšená mAP o 2%.

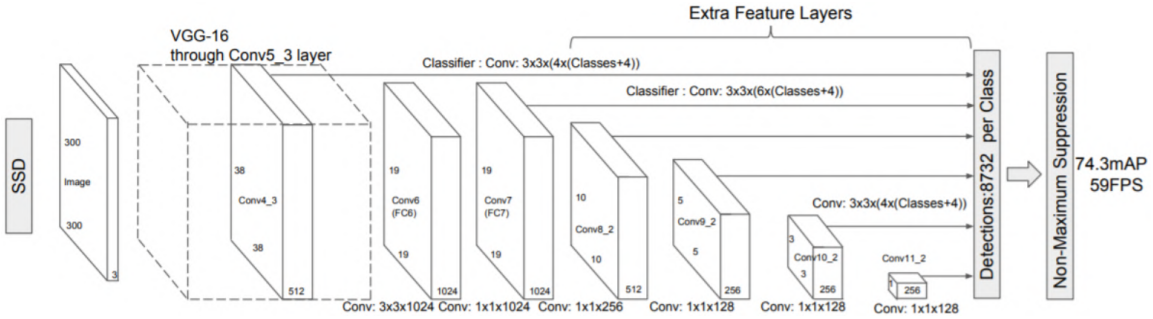
Ďalším vylepšením je použitie klasifikátora s vysokým rozlíšením. Väčšina klasifikátorov detekčných systémov pracuje so vstupnými obrázkami s rozlíšením nižším ako 256×256. YOLOv2 zvýšilo rozlíšenie z pôvodných 224×224, ako tomu bolo pri YOLO, na 448×448. Vyladenie klasifikačnej siete na rozlíšenie 448×448 pre 10 epoch systému ImageNet poskytuje čas prispôsobiť filtre siete, aby mohli pracovať efektívnejšie so vstupnými obrázkami s vyšším rozlíšením. To vylepšuje mAP o takmer 4% oproti systému YOLO.

Vďaka ďalším modifikáciám bola zvýšená mAP systému na 78,6% pri rýchlosti 40 snímok za sekundu.

YOLOv3

Zatiaľ najnovšou verziou systému YOLO je systém YOLOv3 [21]. Jednou z modifikácii je vylepšenie siete používanej pre extrakciu príznakov. YOLOv3 používa sieť Darknet-53, ktorá obsahuje 53 konvolučných vrstiev, čo je takmer 3-krát viac, ako pri sieti Darknet-19, ktorú používa predchádzajúca verzia YOLOv2. Zväčšenie siete síce spôsobilo menší úpadok rýchlosti, ale presnosť siete sa, naopak, zvýšila a v porovnaní so sieťami ResNet-101 a ResNet-152 dosahuje podobnú presnosť, pričom je takmer dvakrát rýchlejšia.

Klasifikátor softmax je v tejto verzii nahradený nezávislými logistickými klasifikátormi. Softmax predpokladá, že jednotlivé triedy objektov sa vzájomne vylučujú, a tým pádom objekt v ohraničujúcom obdĺžniku môže patriť iba do jednej triedy. YOLOv3 umožňuje objektu vďaka použitiu nezávislých logistických klasifikátorov spadať do viacerých tried



Obr. 2.10: Schéma konvolučnej neurónovej siete používanej v systéme SSD. Prevzaté z [15].

zároveň, a teda detegovaný objekt môže patriť, napríklad, do triedy „Osoba“ rovnako ako do triedy „Muž“.

Ohraničujúce obdĺžniky sú predpovedané v troch rôznych mierkach. Príznaky z týchto mierok následne YOLOv3 extrahuje s použitím konceptu, podobnému pyramídovej sieti príznakov (feature pyramid networks). To umožňuje získať zmysluplnejšie sémantické informácie zo skorších máp príznakov. Oproti predchádzajúcim verziám je YOLOv3 úspešnejší v detegovaní malých objektov.

2.3.2 Single Shot MultiBox – SSD

Ďalším detektorom, ktorému stačí pre detegovanie objektov jeden prechod je Single Shot MultiBox detektor (SSD) [15]. SSD je založený na neurónovej sieti, ktorá produkuje pevný počet ohraničujúcich obdĺžnikov vrátane predpovedí tried objektov, ktoré sa nachádzajú v daných obdĺžnikoch.

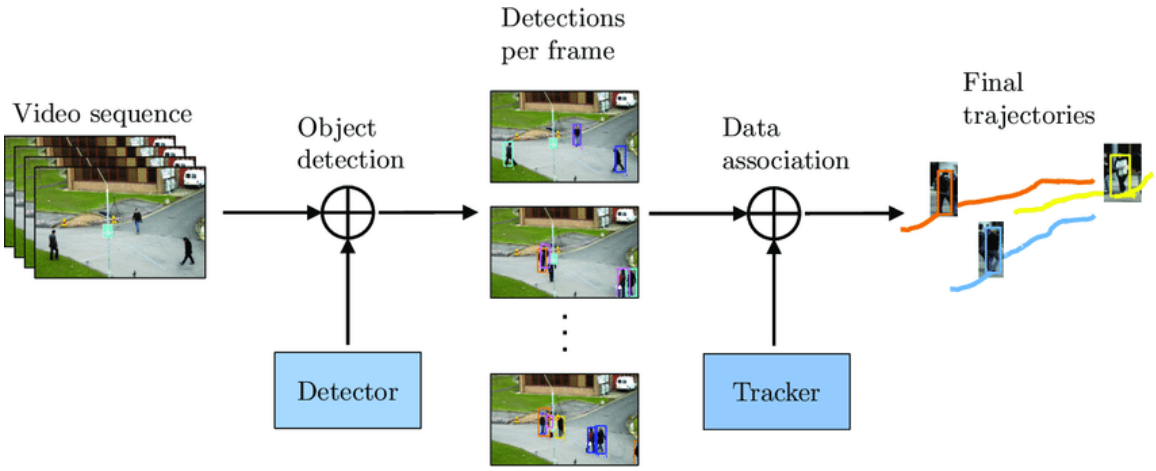
SSD používa ako základ neurónovú sieť VGG16 bez posledných plne prepojených vrstiev. Miesto nich sú pridané konvolučné vrstvy príznakov, ktoré sú postupne zmenšované a umožňujú predpovede detekcií vo viacerých mierkach. Na mapy príznakov sú postupne aplikované malé konvolučné filtre a vďaka tomu dosahuje SSD vysokú presnosť aj pre relatívne malé objekty. Každá pridaná vrstva použitím konvolučných filtrov produkuje pevný počet ohraničujúcich obdĺžnikov. Spolu tak SSD produkuje až 8732 ohraničujúcich obdĺžnikov pre každú triedu (pre porovnanie, YOLO ich produkuje 98), čo napomáha zvýšeniu presnosti detektora. Posledným krokom je potlačenie nemaxím (non-maximum suppression), po ktorom sa vyprodukujú konečné detekcie. Architektúra systému SSD je zobrazená na obrázku 2.10.

Pri vstupnom obrázku s rozlíšením 300×300 dosahuje detektor SSD presnosť 74,3% pri rýchlosti 59 snímok za sekundu.

2.4 Sledovanie objektov

Cieľom sledovania objektov je lokalizovať polohu jednotlivých objektov na po sebe idúcich snímkach videa a zostrojiť trajektórie ich pohybu. Počiatočným detekciám v prvom snímku videa sa priradí jedinečný identifikátor (identita) a je vytvorená trajektória objektu. V každom ďalšom snímku sú nové detekcie asociované s existujúcimi trajektóriami objektov, prípadne pre nepriradené detekcie sú vytvorené nové trajektórie.

Vďaka vývoju v posledných rokoch a používaniu konvolučných neurónových sietí dosiahla detekcia objektov výrazný pokrok. Vďaka tejto skutočnosti je v súčasnosti sledovanie



Obr. 2.11: Štruktúra systému používajúceho metodológiu sledovania na základe detekcií. Obrázok bol prevzatý z [13].

na základe detekcií (tracking by detection) obľúbenou metodológiou pre sledovanie objektov. Je to metodológia, v ktorej pre každú snímku najprv detektor deteguje a lokalizuje objekty. Sledovacia metóda následne predpovie pozíciu objektu na základe predchádzajúceho stavu a spojí predpovedané informácie o objektoch s výsledkami detekcií. Schéma metódy je zobrazená na obrázku 2.11.

Súčasné metódy sledovania objektov čelia niekoľkým známym problémom. S kratšími prekrytiami objektov iným objektom (occlusion) si súčasné metódy sledovania objektov vedia poradiť, avšak pri dlhších prekrytiach sa identita sledovaného objektu stratí a pri znovuobjavení objektu na snímke videa je objektu priradená nová identita (nový identifikátor). Ďalším problémom je zámena identity. Tento problém nastáva, keď sú objekty blízko seba alebo sa navzájom prekryjú a ich trajektóriám je priradená nesprávna detekcia. Presnosť súčasných metód pre sledovanie objektov tiež výrazne závisí aj od úspešnosti detekčných metód, a preto nedostatky detekčných metód ovplyvňujú aj výkon samotnej metódy pre sledovanie objektov.

2.4.1 Simple Online and Realtime Tracking – SORT

SORT [1] je jednou z najznámejších metód pre sledovanie objektov. Ako už aj zo samotného názvu vyplýva, svojou rýchlosťou umožňuje spracovanie v reálnom čase. Ostatné robustnejšie metódy, ktoré sa snažia riešiť hraničné situácie a viaceré nedostatky spojené s detekciami, vyžadujú väčší výpočtový výkon, a tým pádom je limitované použitie takých metód pre real-time aplikácie. Cieľom metódy SORT tak nie je riešiť podobné problémy, ale využiť efektívne a spoľahlivé asociácie medzi jednotlivými snímkami videa, čo by umožnilo tejto metóde pracovať v reálnom čase.

Táto metóda nepoužíva vizuálne vlastnosti detegovaných objektov a pre odhad pohybu a asociáciu dát sú používané iba ohraničujúce obdĺžniky detegovaných objektov. SORT používa konštantný lineárny model rýchlosti pre aproximáciu polohy objektu medzi jednotlivými snímkami videa, ktorý nezávisí od polohy kamery a ostatných sledovaných objektov. Stav každého objektu je vyjadrený ako

$$x = [u, v, s, r, \tilde{u}, \tilde{v}, \tilde{s}]^T \tag{2.2}$$

kde u a v reprezentujú súradnice stredu objektu, s označuje mierku a r označuje pomer strán ohraničujúceho obdĺžnika. Keď je trajektórii priradená nová detekcia, ohraničujúci obdĺžnik detekcie je použitý na aktualizáciu stavu objektu, o ktorú sa stará Kalmanov filter. Pri chýbajúcej detekcii je aktuálny stav objektu predpovedaný iba na základe modelu rýchlosti.

Nové detekcie sú k existujúcim trajektóriám priraďované pomocou Maďarského algoritmu. Na základe prekrytia medzi predpovedným ohraničujúcim obdĺžnikom a všetkými novými ohraničujúcimi obdĺžnikmi je vypočítaná matica cien priradenia (association cost matrix) a pomocou danej matice Maďarský algoritmus priradí novú detekciu k existujúcej trajektórii. Prekrytie týchto obdĺžnikov však musí byť väčšie ako určená hranica $\mathrm{IOU}_{\mathrm{min}}$.

Pri vstupe objektu do snímky videa je potrebné objektu vytvoriť novú identitu a pri jeho výstupe vymazať starú. Keď nová detekcia nie je priradená žiadnej existujúcej trajektórii, je vytvorená nová trajektória. Takto novo vytvorená trajektória je však ešte stále nepotvrdená a potvrdenou sa stane až vtedy, keď sú danej trajektórii priradené detekcie v niekoľkých po sebe idúcich snímkach videa. To zabraňuje vzniku false positive prípadov. V prípade, že v T_l po sebe idúcich snímkach nie je trajektórii priradená nová detekcia, je táto trajektória vymazaná.

2.4.2 Deep SORT

Vylepšením metódy SORT je metóda Deep SORT [26]. Vzhľadom na to, že SORT používa jednoduchú asociačnú metriku na základe prekrytia ohraničujúcich obdĺžnikov, ktorá je presná najmä vtedy, keď je neistota pohybu nízka, táto metóda si nevie dobre poradiť s dlhšími prekrytiami objektov alebo zámenami identity. Cieľom metódy Deep SORT bolo nahradiť túto metriku viac informovanou metrikou, ktorá by kombinovala pohybové a výzorové vlastnosti objektov. Aplikovaním konvolučnej neurónovej siete schopnej rozlišovať ľudí, bola zväčšená robustnosť systému proti chybám, pričom systém ostal jednoduchý, efektívny a aplikovateľný pre aplikácie, ktoré potrebujú bežať v reálnom čase.

Nové detekcie sú k existujúcim trajektóriám priraďované stále pomocou Maďarského algoritmu, avšak je zmenená asociačná metrika, ktorá je teraz zložená s dvoch rozdielnych metrík. Prvou z nich je asociácia na základe Mahalabinosovej vzdialenosti, ktorá je zameraná na pohybové vlastnosti objektu. Prahovaním Mahalabinosovej vzdialenosti na mieru istoty 95% je možné vylúčiť nepravdepodobné asociácie. Táto metrika je využiteľná hlavne vtedy, keď je neistota pohybu nízka. Neočakávané pohyby kamery môžu narušiť použiteľnosť metriky. Z toho dôvodu sa používa v kombinácii s druhou metrikou, ktorá je zameraná na výzorové vlastnosti objektov. Pre každý ohraničujúci obdĺžnik je pomocou konvolučnej neurónovej siete vypočítaný deskriptor vzhľadu a následne je hľadaná najmenšia kosínusová vzdialenosť medzi vektormi príznakov vzhľadu (appearance feature vector) existujúcej trajektórie a novej detekcie. Obidve metriky sú skombinované do jednej asociačnej metriky, ktorú je možné zapísať v tvare

$$D = \lambda D_m + (1 - \lambda) D_a \tag{2.3}$$

kde D_m značí Mahalabinosovu vzdialenosť, D_a značí kosínusovú vzdialenosť medzi vektormi príznakov vzhľadu a λ je parameter, ktorý umožňuje kontrolovať mieru jednej alebo druhej metriky vo výslednej asociačnej metrike.

2.4.3 Person of Interest – POI

POI [28] je jednoduchý online sledovací algoritmus využívajúci konvolučné neurónové siete. Podobne ako SORT a Deep SORT, aj POI používa Kalmanov filter pre predpoveď pohybu objektu.

Asociácia medzi existujúcimi trajektóriami a novými detekciami je vykonávaná pomocou Kuhn-Munkresovho algoritmu. Za účelom vytvorenia matice afinít, potrebnej pre Kuhn-Munkresov algoritmus, je nutné vypočítať afinitu medzi existujúcimi trajektóriami a novými detekciami, ktorá je kombináciou afinity pohybu, afinity tvaru a afinity výzoru. Pre výpočet afinity výzoru je použitá konvolučná neurónová sieť podobná sieti GoogLeNet [24]. Výstupom poslednej plne prepojenej vrstvy siete je 128-dimenzionálny vektor príznakov. Medzi danými vektormi príznakov je nakoniec vypočítaná kosínusová vzdialenosť, ktorá je použitá pre určenie afinity vzhľadu. Čím vyššia je hodnota afinity, tým väčšia je pravdepodobnosť, že sa jedná o rovnaký objekt.

V algoritme POI sú trajektórie rozdelené na dve kategórie – na trajektórie s vysokou kvalitou a trajektórie s nízkou kvalitou. Kvalita trajektórie označuje, ako dobre je objekt sledovaný a berie do úvahy dĺžku trajektórie a všetky úspešné asociácie danej trajektórie. Keďže Kuhn-Munkresov algoritmus môže zlyhať, keď niektoré detekcie chýbajú, najprv sú s detekciami asociované trajektórie s vyššou kvalitou, a neskôr tie s nižšou.

Kapitola 3

Návrh systému pre detekciu a sledovanie vozidiel

V tejto kapitole bude popísaná súťaž AI City Challenge 2020 a konkrétna úloha, ktorú táto práca rieši. Ďalej budú popísané existujúce dátové sady, vrátane dátových sád, na ktorých boli vykonané experimenty a vyhodnotenia. Nasledovať bude popis návrhu systému, vrámci ktorého budú porovnané jednotlivé metódy pre detekciu a sledovanie objektov a bude zdôvodnený výber konkrétnych metód pre riešenie problému.

3.1 AI City Challenge 2020

AI City Challenge [18] je medzinárodná súťaž zameriavajúca sa na úlohy súvisiace s počítačovým videním. Tento rok boli zadané 4 úlohy a táto bakalárska práca sa zameriava na riešenie jednej z nich. Úlohou bolo počítať osobné a nákladné vozidlá pohybujúce sa v jednotlivých predom definovaných smeroch. Pre trénovanie a vyhodnotenie boli poskytnuté videozáznamy viacerých scén zaznamenané statickou kamerou spolu s popisom jednotlivých smerov, ktoré bolo nutné sledovať. Riešenie daného problému môže pomôcť inžinierom lepšie porozumieť požiadavkám na dopravu v jednotlivých smeroch a môže pomôcť k zlepšeniu plánovania časovania križovatiek, prípadne je možné na základe výsledku uplatniť iné stratégie zmiernenia dopravného preťaženia. Aby výsledok mal čo najväčšiu praktickú hodnotu, súťažiaci sa mali zamerať ako na presnosť, tak aj na efektivitu riešenia.

3.2 Dátové sady

Medzi všeobecne známe dátové sady používané na trénovanie a vyhodnotenie detekčných systémov patrí napríklad Pascal VOC alebo MS COCO. Pascal VOC [5] pozostáva z 11 530 obrázkov, ktoré obsahujú spolu 27 450 anotovaných objektov dvadsiatich rôznych kategórií. Dáta sú rozdelené na polovicu, pričom jedna polovica slúži na trénovanie a druhá na testovanie. V súčasnosti sa pre vyhodnotenie detekčných metód viac používa dátová sada MS COCO [14]. Jedná sa o rozsiahlu dátovú sadu obsahujúcu vyše 1,5 milióna objektov osemdesiatich rôznych kategórií, ktoré sa nachádzajú na vyše 330 000 obrázkoch.

Medzi dátové sady používané pre sledovanie objektov patrí napríklad TrackingNet [17]. Je to široká dátová sada zložená z viac ako 30 000 videí, s viac ako štrnástimi miliónmi anotovanými ohraničujúcimi obdĺžnikmi. Známejšia dátová sada, ktoré sa používa aj na vyhodnotenie a porovnanie sledovacích algoritmov je MOT. Konkrétne MOT16 [16] sa

zameriava na sledovanie ľudí a zachytáva 14 videosekvencií v rôznych klimatických podmienkach, s rôznym zahustením scén a s rôznymi pohybmi kamery. Celková dĺžka videí činí niečo vyše 7 a pol minút a videá sú zachytené v snímkovej frekvencii 15-30 FPS, v rozlíšení od 640×480 do 1920×1080 pixelov. V tejto kapitole však budú bližšie popísané dátové sady, na ktorých boli vykonané experimenty a vyhodnotenia.

3.2.1 Vlastná dátová sada

Za účelom vykonania prvotných experimentov s detekciou a sledovaním objektov bola vytvorená vlastná dátová sada. Jedná sa o videozáznamy zachytené z kamery mobilného telefónu. Videá boli natáčané v rozlíšení 1920×1080 so snímkovou frekvenciou 30 snímok za sekundu. Na jednotlivých videozáznamoch sú zachytené pohybujúce sa osoby a vozidlá a pri zriaďovaní dátovej sady bol kladený dôraz na rozmanitosť zahustenia scén. Celá dátová sada pozostáva spolu z 10 videí a celková dĺžka všetkých videí činí približne 59 minút. Príklady je možné vidieť na obrázku 3.1.

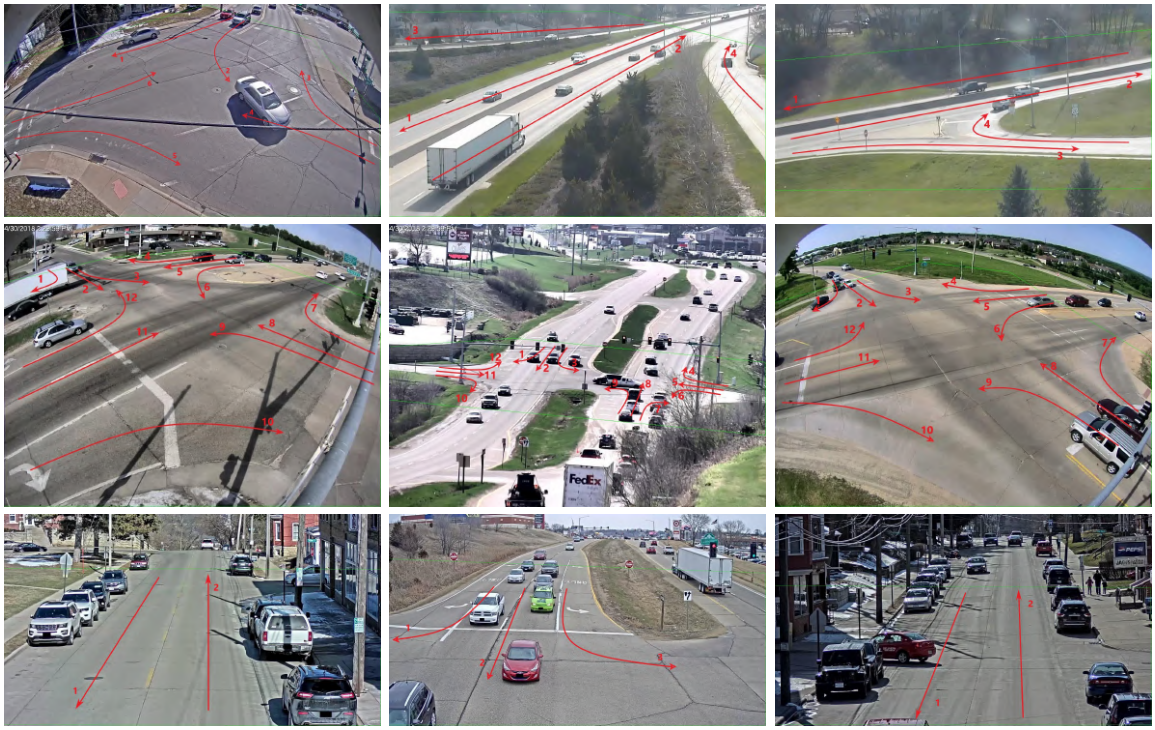
Obr. 3.1: Na obrázkoch sú znázornené príklady pohľadov kamery z vlastnej dátovej sady.

3.2.2 Dátová sada AIC 2020

Dátová sada poskytnutá pre riešenie úlohy obsahuje 31 videoklipov zachytených z dvadsiatich rôznych pohľadov kamery. Súčasťou každého pohľadu kamery je dokument obsahujúci popis regiónu záujmu a popis pohybu. Dátová sada zachytáva rôzne svetelné a klimatické podmienky, ako napríklad dážď, sneženie alebo svitanie. Počet sledovaných smerov naprieč videami nie je rovnaký a mení sa od 2 sledovaných smerov po 12 sledovaných smerov. Dĺžka jednotlivých videoklipov je variabilná, zvyčajne majú však 5 alebo 30 minút, a celková dĺžka všetkých videoklipov činí približne 5 hodín. Videá majú rozlíšenie od 960×960 do 2560×1920 a sú zachytené snímkovou frekvenciou 10 snímok za sekundu. Na obrázku 3.2 je možné vidieť ukážky z dátovej sady.

3.3 Návrh systému

Pred zahájením súťaže AI City Challenge 2020 bolo cieľom tejto práce vytvoriť systém, ktorý by na základe sledovania objektov vytvoril analýzu pohybu objektov na scéne zaznamenanej zo statickej kamery. Za účelom zoznámenia sa s prácou a nedostatkami vybranej detekčnej metódy v kombinácii s vybranou metódou pre sledovanie objektov bolo so systémom v prvých fázach práce experimentované na vlastnej dátovej sade. Už po prvých experimentoch boli viditeľné nedostatky hlavne v detekcii objektov a časté boli aj chyby spôsobené prekrytím objektov. Hlavnou myšlienkou riešenia tejto práce bolo do určitej miery eliminovať dané problémy analýzou a prácou s trajektóriami objektov. Tým pádom



Obr. 3.2: Na obrázku sú znázornené príklady pohľadov kamery z dátovej sady AI City Challenge 2020 spolu s vyznačenými smermi pohybu, v ktorých je nutné počítať vozidlá.

výstup vybranej metódy pre sledovanie objektov nebol konečným krokom, ako to býva u väčšiny systémov, ale bol to len akýsi medzikrok pred vlastným spracovaním trajektórií.

Návrhom riešenia daného problému bolo zaznamenávať si doplňujúce údaje o jednotlivých sledovaných objektoch, akými sú napríklad jednotlivé body trajektórie, vzdialenosť medzi poslednými bodmi trajektórie, uhol pohybu objektov alebo stav sledovaných objektov. Tieto údaje bolo následne možné použiť pre predpovedanie ďalšieho pohybu objektu v prípade, že nebol detegovaný, alebo spojiť trajektórie objektu v prípade, že bol objekt znovu detegovaný pod inou identitou.

Po zahájení súťaže AI City Challenge 2020 boli využité nápady z predchádzajúcej práce, ktoré boli prispôsobené na riešenie konkrétnej úlohy. Návrh systému pre detekciu, sledovanie a počítanie vozidiel pohybujúcich sa v určitých smeroch spolu s konkrétnym riešením vyššie spomenutých problémov je popísaný v ďalšej časti práce.

Štruktúra systému

Činnosť navrhovaného systému je možné rozdeliť na niekoľko samostatných častí, ktoré sú postupne prepojené.

- **Načítanie potrebných údajov** – V tejto časti je nutné načítať vstupné video, prípadne iné potrebné súbory a pripraviť ich na spracovanie.
- **Detekcia objektov** – Po načítaní snímky videa bude na danú snímku aplikovaná vybraná detekčná metóda, ktorá deteguje a lokalizuje objekty určené pre sledovanie. Výstupom tejto časti budú ohraničujúce obdĺžniky.

- **Sledovanie objektov** – Ohraničujúce obdĺžniky budú asociované s existujúcimi trajektóriami sledovaných objektov pomocou vybranej metódy pre sledovanie objektov. Pre nové objekty budú vytvorené nové trajektórie a trajektórie vozidiel, ktoré sa už neobjavujú na snímke, budú odstránené. Výstupom tejto časti budú základné informácie o jednotlivých trajektóriách.
- **Spracovanie trajektórií** – Trajektórie z predchádzajúcej časti budú rozšírené o doplňujúce informácie. Pre každý sledovaný objekt budú uložené doplňujúce informácie týkajúce a ich pohybu a stavu, ktoré budú viesť k odstráneniu niektorých známych chýb sledovania objektov, akými sú nedostatky detektorov alebo dlhšie prekrytia objektov.
- **Postprocessing** – Posledná časť systému sa bude venovať finálnemu spracovaniu, ako je zakreslenie trajektórii na výstup, prípadne zápis potrebných údajov do súborov.

3.4 Výber metód pre detekciu a sledovanie

Z hľadiska dosiahnutia čo najväčšej presnosti systému pri spracovaní v rozumnej rýchlosti (ideálne pri spracovaní v reálnom čase) je výber vhodných metód pre detekciu a sledovanie objektov veľmi dôležitý.

3.4.1 Výber detekčnej metódy

Keďže v systéme bude použitá metodológia sledovania na základe detekcií, je dôležité vybrať vhodnú metódu pre detekciu, ktorá bude zohľadňovať presnosť aj rýchlosť detekcií. Rýchlosť detekčných metód sa určuje počtom spracovaných snímok za sekundu (FPS) a na presnosť sa zvyčajne používa metrika Mean Average Precision (mAP).

Mean Average Precision – mAP

Pred vysvetlením, na akom princípe funguje metrika mAP [10], je dôležité vysvetliť si nasledujúce pojmy.

Precision (presnosť) je počítaná ako pomer správne detegovaných objektov voči všetkým detegovaným objektom. Presnosť je možné vyjadriť vzťahom

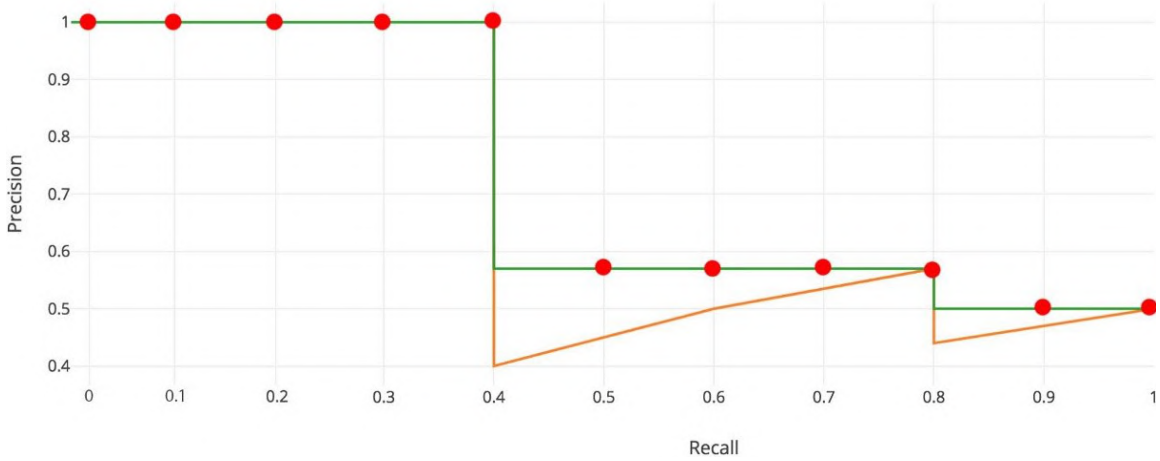
$$P = \frac{TP}{TP + FP} \tag{3.1}$$

kde TP (true positive) znázorňuje počet objektov, ktoré boli detegované správne a FP (false positive) znázorňuje počet objektov, ktoré boli detegované nesprávne alebo nemali byť detegované vôbec.

Recall (citlivosť) je počítaná ako pomer správne detegovaných objektov voči skutočnému počtu objektov, ktoré mali byť detegované. To je možné vyjadriť ako

$$R = \frac{TP}{TP + FN} \tag{3.2}$$

kde TP (true positive) znázorňuje počet objektov, ktoré boli detegované správne a FN (false negative) znázorňuje počet objektov, ktoré detegované mali byť, ale neboli.



Obr. 3.3: Zobrazenie interpolovanej precision-recall krivky. Obrázok bol prevzatý z [10].

Intersect over Union (IoU) vyjadruje mieru prekrytia dvoch obdĺžnikov. Jeden obdĺžnik vyjadruje výsledok detekcie a druhý obdĺžnik vyjadruje skutočné ohraničenie objektu (ground truth). IoU je počítaný ako

$$\text{IoU} = \frac{R_1 \cap R_2}{R_1 \cup R_2} \tag{3.3}$$

kde R_1 a R_2 znázorňujú plochu ohraničujúcich obdĺžnikov. V prípade, že IoU je väčšie ako určitá hranica, je detekcia považovaná za správnu (true positive). V opačnom prípade je detekcia považovaná za chybnú (false postive). Hranica IoU sa líši od používanej metriky. Pascal VOC používa hranicu 0,5 IoU, zatiaľ čo MS COCO používa viaceré hranice IoU od 0,05 až po 0,95.

Pred výpočtom mAP je potrebné si vypočítať AP (average precision) pre každú triedu. Všeobecne je AP definovaná ako plocha pod precision-recall krivkou. Konkrétny výpočet AP sa však naprieč dátovými sadami mierne líši.

Pre výpočet hodnoty AP je zostrojená precision-recall krivka, ktorá je následne vyhladená jej interpoláciou. To je dosiahnuté nahradením hodnoty precision pre každú hodnotu recall maximálnou hodnotou precision, ktorú dosahujú hodnoty recall napravo od aktuálnej hodnoty. Graficky je interpolácia znázornená na obrázku 3.3. Interpoláciou je dosiahnuté odstránenie zákmitov spôsobené malými odchýlkami v klasifikácii detekcií. Matematicky je možné interpolovanú precision-recall krivku vyjadriť vzťahom

$$p_{int}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{3.4}$$

Pascal VOC2007 používa 11-bodovú interpoláciu, kde sú hodnoty recall rozdelené na 11 častí (0,0.1,0.2,...,0.9,1). AP je následne určená hodnotami precision-recall krivky pre dané hodnoty recall, čo je možné vyjadriť vzťahom

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} p_{int}(r) \tag{3.5}$$

Celková hodnota mAP je vypočítaná ako priemer AP naprieč všetkými triedami. Pre niektoré dátové sady, ako napríklad MS COCO, sú pojmy mAP a AP zameniteľné.

Porovnanie metód pre detekciu

Porovnanie presnosti a rýchlosti jednotlivých detekčných metód popísaných v kapitole 2 na dátovej sade Pascal VOC 2007 je možné vidieť v tabuľke 3.1.

Metóda	**Presnosť mAP (%)**	**Rýchlosť (FPS)**
R-CNN	66,0	0,03
Fast R-CNN	66,9	0,6
Faster R-CNN(VGG16)	73,2	9,1
Faster R-CNN(ResNet101)	**83,8**	0,4
YOLO	63,4	45
SSD300	74,3	**46**
SSD512	76,8	19
YOLOv2(544)	78,6	40
YOLOv3(544)	79,6	–

Tabuľka 3.1: Porovnanie presnosti a rýchlosti detekčných metód na dátovej sade Pascal VOC. Údaje boli prebrané z [30, 29].

Z tabuľky 3.1 je vidieť, že najvyššiu presnosť dosahuje detektor Faster R-CNN, ktorý používa ResNet101 ako konvolučnú sieť pre nájdenie regiónov, avšak daná metóda dosahuje rýchlosť spracovania iba 0,4 snímok za sekundu, čo je pre zadanú úlohu nevyhovujúce. V rýchlosti najlepšie obstál detektor SSD so vstupným obrázok s rozlíšením 300×300, ktorý dosahuje presnosť 74,3% mAP pri rýchlosti 46 snímok za sekundu. YOLOv2 dosahuje vyššiu presnosť s vyšším rozlíšením vstupného obrázka ako SSD pri porovnateľnej rýchlosti. Ešte vyššiu presnosť ako YOLOv2 dosahuje YOLOv3, avšak pri tomto detektore chýba údaj ohľadom rýchlosti spracovania na dátovej sade Pascal VOC 2007.

Následujúca tabuľka 3.2 obsahuje porovnanie verzií YOLOv2 a YOLOv3 vrátane ich odľahčených verzií na dátovej sade MS COCO.

Metóda	**Presnosť mAP (%)**	**Rýchlosť (FPS)**
YOLOv2	48,1	40
YOLOv2-Tiny	23,7	**244**
YOLOv3	**57,9**	20
YOLOv3-Tiny	33,1	220

Tabuľka 3.2: Porovnanie presnosti a rýchlosti detekčných metód YOLOv2 a YOLOv3 (vrátane ich odľahčených verzií) na dátovej sade MS COCO. Údaje boli prebrané z [27, 9].

Najrýchlejšou metódou je odľahčená verzia systému YOLOv2 – YOLOv2-Tiny, ktorá umožňuje spracovanie pri rýchlosti až 244 snímok za sekundu, avšak za cenu nízkej presnosti. Čo sa týka presnosti, je na tom najlepšie metóda YOLOv3, ktorá dosahuje o 9,8% vyššiu presnosť ako YOLOv2. Aj napriek tomu, že je o polovicu pomalší, systém YOLOv3 dosahuje spracovanie v rýchlosti 20 snímok za sekundu, čo som považoval pre túto prácu za dostačujúce, a preto bude v práci použitá práve táto metóda.

3.4.2 Výber sledovacej metódy

Ďalším krokom bol výber vhodnej metódy pre sledovanie objektov. Aby bolo možné jednotlivé metódy vyhodnotiť a porovnať medzi sebou, existujú na to viaceré metriky vyhodnote-

nia. Medzi najznámejšie a najpoužívanejšie metriky pre vyhodnotenie metód pre sledovanie objektov patria metriky MOTA a MOTP [16].

MOTA (Multiple Object Tracking Accuracy) kombinuje tri zdroje chýb. Zodpovedá za nájdenie false positive prípadov, nezachytených prípadov (misses) a prípadov, pri ktorých došlo k nesprávnemu spojeniu (mismatch) vo všetkých snímkach videa. MOTA je možné zapísať vzťahom

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{3.6}$$

kde m_t znázorňuje počet nezachytených prípadov, fp_t počet false positive prípadov, mme_t počet nesprávne spojených prípadov za čas t a g reprezentuje počet všetkých objektov.

MOTP (Multiple Object Tracking Precision) popisuje presnosť určenia polohy medzi správne sledovanými objektmi (true positive prípadmi) a všetkými objektmi, ktoré mali byť sledované (ground truth). Je možné ho vyjadriť vzťahom

$$\text{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t} \tag{3.7}$$

kde c_t značí počet spojení v snímke t a $d_{i,t}$ je prekrytie ohraničujúceho obdĺžnika cieľa i s priradeným skutočným (ground truth) objektom.

Porovnanie metód pre sledovanie objektov

Porovnanie metód pre sledovanie objektov popísaných v podkapitole 2.4 vyhodnotených na dátovej sade MOT16 je zobrazené v tabuľke 3.3.

Metóda	MOTA	MOTP	Rýchlosť (Hz)
SORT	33,4	72,1	**260**
Deep SORT	61,4	79,1	40
POI	**66,1**	**79,5**	10

Tabuľka 3.3: Porovnanie MOTA, MOTP a rýchlosti metód pre sledovanie objektov na dátovej sade MOT16. Údaje boli prebrané z [27, 9].

Metóda SORT síce je spomedzi porovnávaných metód najrýchlejšia, pričom dosahuje rýchlosť až 260 Hz, ale podľa metriky MOTA je takmer o polovicu menej presná, ako metódy Deep SORT a POI. Metóda POI vedie v metrikách MOTA a MOTP, avšak dosahuje rýchlosť iba 10 Hz, čo nezaručuje spracovanie v reálnom čase. Deep SORT dosahuje o necelých 5% menšiu hodnotu v metrike MOTA ako POI, ale pri rýchlosti 40 Hz je 4-krát rýchlejší ako POI, čím dosahuje spracovanie v reálnom čase, a preto som sa rozhodol v tejto práci použiť práve túto metódu.

3.5 Spracovanie trajektórií

Po asociácii nových detekcií s existujúcimi trajektóriami, za ktorú je zodpovedná metóda pre sledovanie objektov, nasleduje vlastné spracovanie trajektórií. Cieľom tohto kroku je redukovať chyby spôsobené chybnými detekciami a nedostatkami sledovacej metódy, čím by sa následne zvýšila presnosť systému.

Ukladané informácie o pohybujúcom sa vozidle

U každého sledovaného vozidla je ukladaných niekoľko údajov, ktoré rozširujú výstup poskytnutý sledovacou metódou Deep SORT, ako napríklad body trajektórie, stav vozidla, rýchlosť pohybu, smer pohybu a iné (bližšie popísané v sekcii 4.2). Tieto údaje pomáhajú získať lepší prehľad o pohybe vozidla a môžu byť použité pri predpovedi ďalšieho pohybu vozidla, pri re-identifikácii vozidla a podobne.

Nájdenie vstupných a výstupných plôch

Nájdenie vstupnej a výstupnej plochy, cez ktoré vozidlo vstúpilo do regiónu záujmu, prípadne z neho vystúpilo, je základom pre určenie konkrétneho smeru, v ktorom sa vozidlo pohybovalo.

Pri prvom výskyte vozidla v regióne záujmu je potrebné zistiť, či sa vozidlo nachádza na jednej z preddefinovaných vstupných plôch. Spôsob definovania vstupných a výstupných plôch bude bližšie popísaný v podkapitole 4.2. V prípade, že sa vozidlo nachádza na nejakej vstupnej ploche, identifikátor danej plochy je uložený medzi údaje vozidla. Opačný prípad indikuje, že vozidlo bolo prvýkrát detegované na mieste blížiacom sa k stredu regiónu zájmu. V takom prípade sa systém po opustení vozidla zo scény snaží dohľadať, akou vstupnou plochou mohlo vozidlo vstúpiť do regiónu záujmu. Na základe prvých bodov jeho trajektórie je predpovedaný pohyb vozidla pred tým, ako bolo prvýkrát detegované. Môže ale nastať situácia, že vozidlo sa na začiatku svojej existencie v regióne záujmu nepohybuje, a preto je v takom prípade nájdená najbližšia vstupná plocha k prvému bodu trajektórie vozidla.

Pokým je vozidlo v regióne záujmu, pri každom ďalšom bode jeho trajektórie sa skúma, či sa daný bod nachádza na niektorej z výstupných plôch. V momente nájdenia výstupnej plochy sa identifikátor danej plochy uloží medzi údaje vozidla. Pri situácii, kedy vozidlo opustilo región záujmu a nebola mu nájdená žiadna výstupná plocha (môže nastať pri nepresnej predpovedi pohybu vozidla), sa hľadá výstupná plocha v určitej blízkosti posledného bodu trajektórie vozidla. Keď sa v takej blízkosti nenachádza žiadna výstupná plocha, identifikátor výstupnej plochy ostane prázdny, a tým pádom vozidlo nebude započítané do žiadneho smeru.

Predikcia ďalších bodov trajektórie

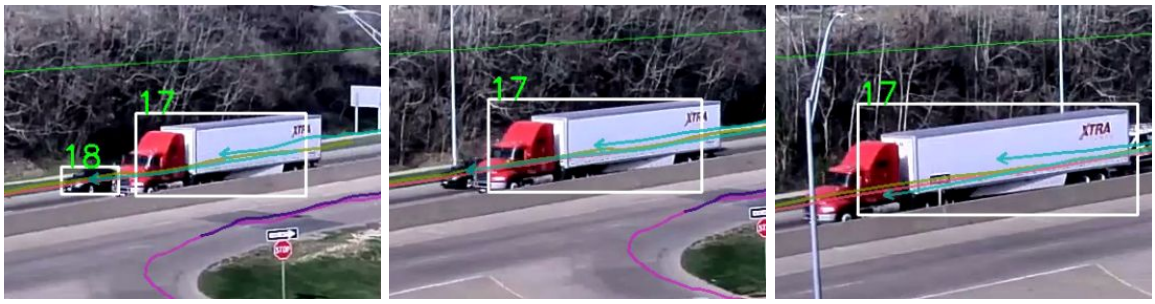
V prípade, že vozidlo už nie je detegované, ale stále sa objavuje na snímke videa (ešte neopustilo región záujmu), ďalšie body jeho trajektórie sú predpovedané. Vozidlo však musí spĺňať určité podmienky, aby mohol byť jeho ďalší pohyb predpovedaný. Trajektória vozidla musí pozostávať minimálne z troch bodov. Vozidlo musí byť v pohybe, to znamená, že vzdialenosť medzi poslednými bodmi trajektórie musí byť väčšia ako určitá hranica. Ďalšou podmienkou je určitá konzistentnosť v smere pohybu vozidla, čiže jednotlivé uhly pohybu medzi poslednými bodmi trajektórie sa nemôžu výrazne líšiť. V prípade, že tieto podmienky nie sú splnené, ďalší pohyb vozidla predpovedaný nie je. Implementované sú 2 spôsoby predpovede ďalšieho pohybu.

Prvý spôsob je použitý vtedy, keď sa celá trajektória vozidla nachádza iba na jednej ceste. Cesta je definovaná polygónom v preddefinovanom textovom súbore k danému pohľadu kamery. V takom prípade je ďalší bod trajektórie predpovedaný na základe vzdialenosti medzi poslednými dvoma bodmi trajektórie smerom k výstupnej ploche, ktorá sa nachádza na konci danej cesty.

V situácii, keď sa celá trajektória vozidla nachádza na viacerých cestách, sa vypočíta priemerný uhol medzi poslednými piatimi bodmi a euklidovská vzdialenosť medzi poslednými dvomi bodmi trajektórie. Na základe vypočítanej vzdialenosti a uhla sa predpovie ďalší bod trajektórie. Matematicky je možné predpoveď ďalšieho bodu x_n zapísať ako

$$x_n = x_i + d(x_i, x_{i-1}) \cdot \frac{1}{4} \sum_{j=0}^{4} a(x_{i-j}, x_{i-j-1}) \tag{3.8}$$

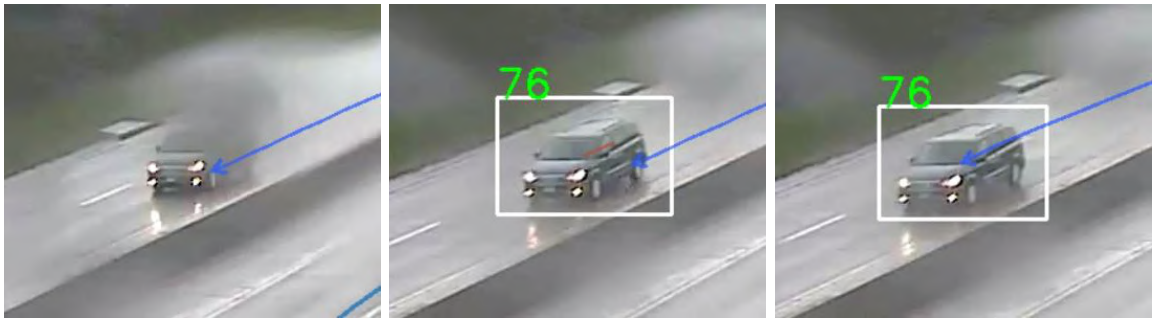
kde x_i znázorňuje posledný bod trajektórie, funkcia d vypočíta vzdialenosť medzi dvoma bodmi a funkcia a vypočíta uhol medzi dvoma bodmi. Grafická ukážka predpovede trajektórie je zobrazená na obrázku 3.4.



Obr. 3.4: Ukážka predpovede ďalších bodov trajektórie. Na ľavom obrázku je možné vidieť 2 sledované vozidlá, vrátane ich trajektórií. Stredný obrázok zobrazuje situáciu o pár snímok neskôr, kde vozidlo s identifikátorom 18 už nebolo detegované. O pár snímok neskôr už vozidlo nie je viditeľné, pretože ho prekrylo nákladné vozidlo, ale trajektória zakrytého vozidla je stále predpovedaná.

Spájanie trajektórií

Pred predpoveďou ďalšieho pohybu vozidla sa systém snaží vyriešiť problém dlhšieho prekrytia objektu alebo nedostatky detektora spájaním trajektórií. V prípade, že vozidlo nie je viditeľné na snímku videa určitý čas a následne sa znovu objaví na snímku, začne svoju vlastnú novú trajektóriu, aj keď stará trajektória vozidla je stále predpovedaná. To spôsobuje problém, že jedno vozidlo má dve rôzne trajektórie. Pre vysporiadanie sa s takou situáciou sa systém snaží spojiť také trajektórie. Pre každú predpovedanú trajektóriu vozidla je snaha nájsť novo vzniknutú trajektóriu pozostávajúcu z dvoch alebo troch bodov, ktorá sa nachádza v blízkosti posledného bodu predpovedanej trajektórie. Ak je taká trajektória nájdená, je následne porovnaný aj smer pohybu obidvoch trajektórií. V prípade, že aj umiestnenie aj smer pohybu sú podobné, sú tieto dve trajektórie spojené. Grafické znázornenie je zobrazené na obrázku 3.5. Táto metóda môže viesť k zámene identity medzi vozidlami, ale pri pohybe rovnakým smerom to nie je problém.



Obr. 3.5: Ukážka spojenia dvoch trajektórií. Na ľavom obrázku nebolo vozidlo detegované kvôli dažďovej kvapke na zábere kamery a jeho trajektória je s malou nepresnosťou predpovedaná. Na ďalšej snímke bolo vozidlo znovu detegované, ale už pod novou identitou a začalo novú červenú trajektóriu. Na poslednom obrázku systém rozoznal, že sa v blízkosti modrej trajektórie nachádza nová trajektória, ktorá sa pohybuje podobným smerom a tieto trajektórie sú následne spojené.

Kapitola 4

Implementácia systému a vyhodnotenie

V úvode kapitoly budú popísané použité nástroje vybrané pre riešenie problému. V ďalšej časti budú popísané konkrétne riešenia problémov vrátane popisu vstupných a výstupných dát. Na konci kapitoly budú popísané jednotlivé experimenty so systémom spolu s vyhodnotením systému na dátových sadách.

4.1 Použité technológie

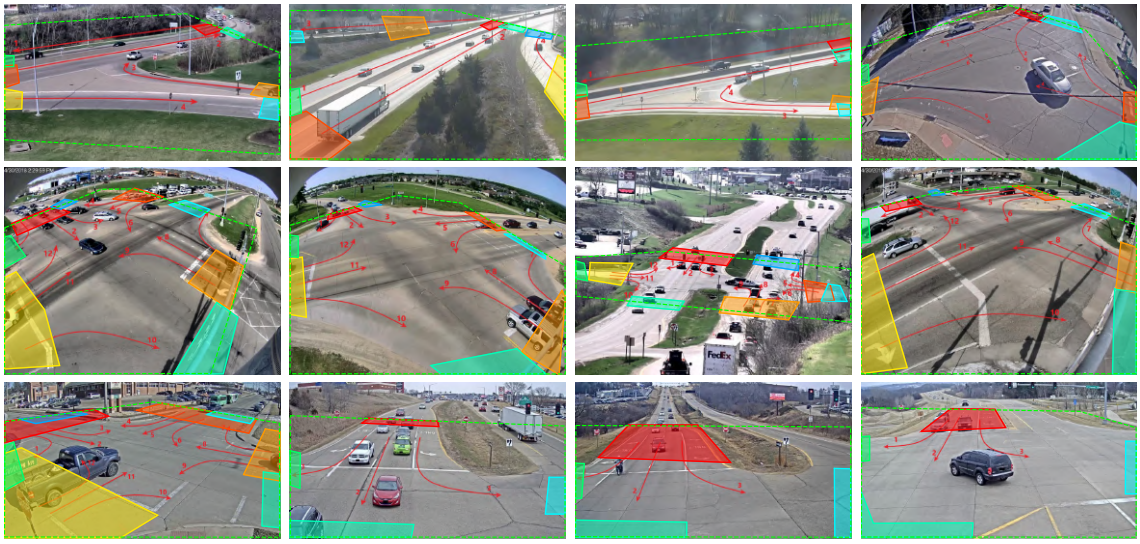
Pre implementáciu systému som sa rozhodol použiť jazyk Python vo verzii 3.7. Z dôvodu dosiahnutia maximálnej rýchlosti spracovania je na výpočty použitá aj grafická karta GPU. Aby bolo možné používať na výpočty aj grafickú kartu, je potrebné mať nainštalovaných niekoľko balíkov. Jedným z nich je CUDA. CUDA [3] je paralelná počítačová platforma a počítačový model vyvinutý spoločnosťou NVIDIA, ktorý umožňuje všeobecné výpočty na GPU. Ďalšou potrebnou súčasťou je knižnica cuDNN (NVIDIA CUDA Deep Neural Network) [4]. Jedná sa o knižnicu primitív pre hlboké neurónové siete s akceleráciou GPU, ktorá poskytuje vysoko ladené implementácie pre štandardné rutiny, ako sú dopredné a spätné konvolúcie, združovanie, normalizácie a aktivačné vrstvy. Medzi ďalšie potrebné balíčky patrí mimo iných aj tensorflow-gpu a knižnica OpenCV.

4.2 Implementácia systému

Ako bolo spomenuté v podkapitole 3.4, pre riešenie úlohy je na detekciu a lokalizáciu objektov vybraná metóda YOLOv3 a pre sledovanie objektov metóda Deep SORT. Základ riešenia tvorí open-source projekt deep_sort_yolov3[1], ktorý prepája spomenuté metódy pre detekciu a sledovanie objektov. Výstupom prebraného systému bolo video s vyznačenými ohraničujúcimi obdĺžnikmi detegovaných objektov, vrátane zobrazenia identifikátorov daných objektov.

V nasledujúcej časti bude popísané, ako som sa snažil vyriešiť spomenuté problémy, a na akom princípe funguje vytvorený systém pre detekciu, sledovanie a počítanie vozidiel v jednotlivých smeroch.

[1] https://github.com/Qidian213/deep_sort_yolov3



Obr. 4.1: Príklady obrázkov z dátovej sady s vyznačením vstupných a výstupných plôch pre jednotlivé pohľady kamier.

Označenie parametrov pre kamery

Pre každý pohľad kamery bol vytvorený textový súbor, v ktorom boli vyznačené pomocné údaje, potrebné pre správne počítanie vozidiel. Je potrebné si vyznačiť:

- Vstupné plochy – Súradnice polygónu.
- Výstupné plochy – Súradnice polygónu.
- Miesta, kde sa budú zobrazovať počty prejdených vozidiel v danom smere – Súradnice bodu.
- Dvojice „ID vstupnej plochy – ID výstupnej plochy“, ktoré reprezentujú, odkiaľ a kam sa majú vozidlá pohybovať, aby boli započítané v danom smere.
- Jednotlivé cesty pre každý smer – Súradnice polygónu.

Príklad označených vstupných a výstupných plôch pre jednotlivé pohľady kamery je možné vidieť na obrázku 4.1.

Načítanie potrebných údajov

Na začiatku systému je nutné načítať potrebné súbory. Pomocou funkcie `VideoCapture` je načítané vstupné video vo formáte *mp4*, z ktorého sú neskôr sekvenčne vyberané snímky určené pre spracovanie.

Medzi ďalšie vstupné súbory patrí textový súbor s bližšími špecifikáciami pre konkrétny pohľad kamery popísaný vyššie a textový súbor popisujúci región záujmu, ktorý je súčasťou poskytnutej dátovej sady.

Detekcia objektov

Každá snímka je následne vstupom funkcie `detect_image`, ktorá deteguje a lokalizuje objekty. Detektor bol upravený na detekciu dvoch tried objektov, konkrétne na detekciu triedy „car“ a triedy „truck“. Pri počiatočnom znížení veľkosti snímky na pevné rozlíšenie 416×416 pixelov mala detekčná metóda problém s detegovaním menších objektov. Po ponechaní vstupného rozlíšenia snímky sa mierne znížila rýchlosť spracovania, ale detekcie boli do značnej miery presnejšie.

Aj napriek tomu detektor občas nebol schopný detegovať niektoré objekty, a preto bol znížený prah detekcie z pôvodných 0,5 na 0,25. Zníženie prahu detekcie pre osobné vozidlá vykazovalo lepšie výsledky, avšak po znížení prahu detekcie pre nákladné vozidlá vznikalo viacero false positive prípadov, a preto je v systéme použitý iný prah detekcie pre osobné vozidlá a iný pre nákladné vozidlá. Pre nákladné vozidlá je použitý prah 0,6.

Keďže detektor YOLOv3 povoľuje objektu spadať do viacerých tried súčasne, vyskytoval sa problém, že vozidlo spadalo do triedy „car“ rovnako ako do triedy „truck“. Preto sa pre ohraničujúce obdĺžniky nákladných vozidiel hľadajú ohraničujúce obdĺžniky osobných vozidiel, ktoré sa s ním prekrývajú aspoň na 50%, a je ponechaný iba jeden ohraničujúci obdĺžnik. Posledným krokom detekcie objektov je potlačenie nemaxím.

Asociácia s trajektóriami

Prácu metódy pre sledovanie objektov Deep SORT je možné rozdeliť do dvoch častí. V prvej časti prostredníctvom funkcie `predict` je na základe predchádzajúceho stavu objektov predpovedaný pomocou Kalmanovho filtra ďalší ich stav. Výsledné stavy sú následne prostredníctvom funkcie `update` asociované s detekciami, ktoré sú výstupom detekčnej metódy, a sú rozšírené trajektórie vozidiel (tracks), prípadne pre nezaradené detekcie sú vytvorené nové trajektórie. Asociácie sú realizované pomocou Maďarského algoritmu.

Spracovanie trajektórií

Trajektórie, ktoré sú výstupom metódy pre sledovanie objektov, sú rozšírené o ďalšie informácie. Medzi najzákladnejšie informácie patrí:

- trajectory – Obsahuje body trajektórie. Za bod trajektórie je braný stred ohraničujúceho obdĺžnika.
- length – Určuje počet bodov trajektórie.
- state – Je rozlišovaných 5 stavov objektov, a to *detected*, *predicted*, *redetected*, *assigned* a *finished*.
- distance – Určuje vzdialenosť medzi poslednými dvoma bodmi trajektórie.
- angle – Určuje uhol pohybu medzi poslednými dvoma bodmi trajektórie.
- TTL – Určuje životnosť objektu.
- type – Určuje typ vozidla (osobné alebo nákladné).
- startID – Určuje identifikátor vstupnej plochy, ktorou vozidlo vstúpilo do regiónu zájmu.

- endID – Určuje identifikátor výstupnej plochy, ktorou vozidlo vystúpilo z regiónu zájmu.
- road – Určuje identifikátor cesty, na ktorej sa vozidlo nachádza.
- include – V prípade, že bolo vozidlo re-identifikované, je do „include“ uložený jeho nový identifikátor.

Jednotlivé sledované vozidlá sú identifikovateľné na základe identifikátora, ktorý je výstupom metódy Deep SORT. Po celý čas, kedy je vozidlo detegované je v stave *detected*. V prípade, že v niektorom snímku vozidlo detegované nebolo, prejde do stavu *predicted*, kedy je ďalší jeho pohyb predpovedaný a pri každom ďalšom snímku sa snaží nájsť trajektóriu vozidla, s ktorou by sa mohol spojiť. Keď takú trajektóriu nájde, prejde do stavu *redetected*, predpovedané body jeho trajektórie sú vymazané a sú nahradené bodmi trajektórie vozidla s novou identitou (jedná sa o to isté vozidlo, ale keďže bolo vozidlo detegované po nejakom čase, nadobudlo novú identitu). Vozidlo s novou identitou prejde do stavu *assigned* a v každom ďalšom snímku sú body jeho trajektórie priradené príslušnému vozidlu so starou identitou, ktoré je v stave *redetected*.

V momente, kedy vozidlo opustí región záujmu, prejde do stavu *finished* a tým pádom už nie je ďalej spracovávané (nie sú pridávané body trajektórie, nie je predpovedaný ďalší pohyb a podobne). V každom snímku sa následne dekrementuje hodnota TTL, ktorá určuje, ako dlho sa bude vykresľovať trajektória daného vozidla. V prípade, že TTL nadobudne hodnotu 0, trajektória vozidla sa už ďalej nebude vykresľovať.

Postprocessing

Systém poskytuje viacero výstupov. Poskytuje textový súbor v špecifickom formáte potrebný pre vyhodnotenie v súťaži AI City Challenge 2020, kde pre každé započítané vozidlo je na samostatnom riadku uvedený identifikátor videa, číslo snímky, kedy bolo vozidlo započítané, identifikátor smeru pohybu a typ vozidla. Jednotlivé údaje sú od seba oddelené medzerou. Užívateľ však má možnosť prostredníctvom výstupu sledovať spracovávané videozáznamy, prípadne si výsledné video uložiť vo formáte *AVI*. Vo výstupe môže užívateľ vidieť jednotlivé trajektórie vozidiel vrátane smeru ich pohybu, ktorý je vyznačený šípkou, a tiež počty osobných a nákladných vozidiel pohybujúcich sa v jednotlivých smeroch. Aby bolo možné trajektórie od seba jednoducho rozlíšiť, každej je priradená náhodná farba, a po zmiznutí vozidla zo scény zmizne po chvíli aj jeho trajektória. Príklad výstupu je zobrazený na obrázku 4.2.

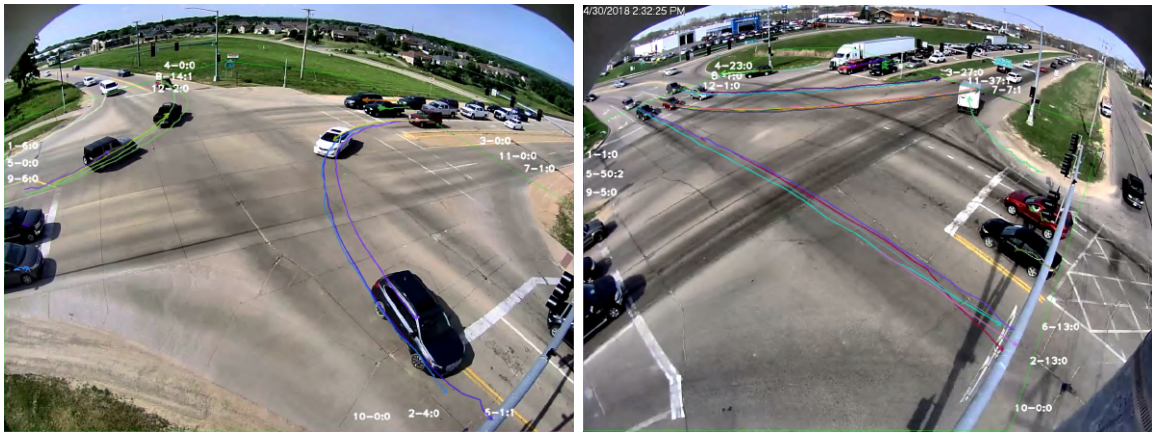
4.3 Metrika vyhodnotenia

AI City Challenge používa svoju vlastnú metriku pre vyhodnotenie systémov pre počítanie vozidiel pohybujúcich sa v jednotlivých smeroch [18]. Celkové skóre je vypočítané ako vážená kombinácia medzi účinnosťou ($S1_{\text{efficiency}}$) a efektívnosťou ($S1_{\text{effectiveness}}$) riešenia.

$$S1 = \alpha S1_{\text{efficiency}} + \beta S1_{\text{effectiveness}} \tag{4.1}$$

kde $\alpha = 0.3$, $\beta = 0.7$.

Efektivita riešenia $S1_{\text{efficiency}}$ je počítaná na základe celkového času t potrebného na spracovanie úlohy, ktorý je vynásobený základným faktorom efektivity b normalizovaným



Obr. 4.2: Na obrázku sú zobrazené príklady výstupu systému, na ktorých sú rozdielnymi farbami vykreslené trajektórie vozidiel, vrátane počítadiel vozidiel v jednotlivých smeroch.

v rozsahu 0 až 5-krát celkový čas videí T. Základný faktor efektivity sa meria na stroji, na ktorom sa vykonávajú experimenty a znázorňuje výpočtovú výkonnosť stroja.

$$S1_{\text{efficiency}} = \max\left(0, 1 - \frac{t \cdot b}{T}\right) \tag{4.2}$$

Účinnosť riešenia $S1_{\text{effectiveness}}$ je počítaná ako vážený priemer normalizovaných vážených stredných štvorcových chybových skóre nwRMSE pre všetky videá, smery pohybu a triedy vozidiel v dátovej sade s proporciálnymi váhami, ktoré sú založené na počte vozidiel danej triedy v smere pohybu. Každé video je rozdelené do k segmentov a do úvahy sa berú kumulatívne počty vozidiel od začiatku videa do konca každého segmentu. Skóre nwRMSE je vážená stredná štvorcová chyba wRMSE medzi predpokladaným a skutočným počtom vozidiel, ktorá je normalizovaná skutočným počtom vozidiel daného typu pohybujúcich sa daným smerom. V prípade, že skóre wRMSE je väčšie ako skutočný počet vozidiel, je do skóre nwRMSE priradená hodnota 0. V opačnom prípade je do skóre nwRMSE priradená hodnota (1-wRMSE/počet vozidiel).

$$wRMSE = \sqrt{\sum_{i=1}^{k} w_i(\hat{x}_i - x_i)^2}, \tag{4.3}$$

$$\text{kde } w_i = \frac{i}{\sum_{j=1}^{k} j} = \frac{2i}{k(k+1)} \tag{4.4}$$

Vlastné vyhodnotenie

Keďže vyššie spomenutá metrika vyhodnotenia sa používa na evalvačnom serveri súťaže AI City Challenge 2020 a každý súťažiaci má obmedzený počet vyhodnotení na danom serveri (maximálne 10 pokusov), vytvoril som si vlastný jednoduchý program pre vyhodnotenie úspešnosti riešenia.

Jedná sa o program, ktorý na základe textového výstupu systému vypočíta percento úspešnosti a určí počet true positive, false positive a false negative prípadov. True positive prípady označujú správne započítané vozidlá, false positive prípady označujú vozidlá, ktoré boli započítané nesprávne, a false negative sú prípady, ktoré nastanú v situácii, keď vozidlo

nebolo započítané. Vstupom programu sú 2 textové súbory – textový súbor s anotáciami a textový súbor, ktorý je výstupom systému pre počítanie vozidiel. Oba textové súbory majú formát požadovaný pre vyhodnotenie súťaže popísaný vyššie. Pre účely vyhodnotenia bolo nutné anotovať poskytnutú dátovú sadu. Dátová sada bola anotovaná ručne s podporou jednoduchého programu, ktorý umožňoval načítať video, zobraziť aktuálnu snímku spolu s číslom snímky, a na základe dvoch kláves umožňoval posúvať sa po snímkach dopredu alebo dozadu. Anotovaných bolo prvých 3000 snímok z každého videa, čo spolu činí 4537 anotácií. Program pre vyhodnotenie načíta riadok z textového súboru s anotáciami, získa z neho potrebné informácie, a snaží sa nájsť v textovom súbore riadok s rovnakým identifikátorom videa, rovnakým smerom pohybu, rovnakým typom vozidla a s podobným číslom snímky v tolerancii ±100 snímok. Celková úspešnosť je následne vypočítaná odčítaním počtu chýb z celkového počtu anotácií. Keďže je použitá iná metrika ako pri oficiálnom vyhodnotení na evalvačnom serveri, výsledky sa mierne líšia. Porovnanie celkovej úspešnosti oboch používaných metrík je možné vidieť v tabuľke 4.1. Jedným z dôvodov, prečo dosahuje vlastná metrika nižšiu úspešnosť, môže byť skutočnosť, že vyhodnotených je prvých 3000 snímok každého videa a najmä v prvých snímkach je pomerne časté, že vozidla nie sú započítané, čo spôsobuje nárast false negative prípadov.

Metrika vyhodnotenia	**Úspešnosť (%)**
AIC 2020	90,94
Vlastná	85,81

Tabuľka 4.1: Porovnanie úspešnosti systému na rozdielnych metrikách vyhodnotenia, používaných pre vyhodnotenie systému.

4.4 Experimenty

Prvé vyhodnotenie na vyhodnocovacom serveri súťaže AI City Challenge ukázalo s úspešnosťou 0,8993 sľubné výsledky. Menej potešujúca však bola efektivita, ktorá dosiahla skóre 0. Bolo to spôsobené tým, že spracovanie bolo vykonávané na stroji s málo výkonnou grafickou kartou NVIDIA GeForce 940MX a preto spracovanie trvalo veľmi dlhý čas. Po prechode na stroj s grafickou kartou NVIDIA RTX 2080 Ti, bola rýchlosť výrazne vyššia a preto v druhom vyhodnotení dosiahla efektivita skóre 0,6725. Aj keď celkové skóre po druhom vyhodnotení dosahovalo skóre 0,8129, stále existoval priestor na zlepšenie.

Napriek tomu, že prah detekcie pre osobné vozidlá bol nastavený na 0,25, stále dochádzalo k pomerne veľkému množstvu false negative prípadov, kedy vozidlo nebolo detegované. Z toho dôvodu bolo experimentované s nižšími prahmi detekcie, konkrétne s prahmi 0,20 a 0,15 a pre porovnanie aj s prahom 0,30. Výsledky experimentov s rozdielnymi prahmi detekcie je možné vidieť v tabuľke 4.2. Z experimentov je vidieť jasný trend, kedy pri znižovaní prahu detekcie klesá počet false negative prípadov, ale naopak, rastie počet false positive prípadov. Najlepšiu úspešnosť vykazovalo riešenie s prahom 0,20.

Pri niektorých záberoch sa stávali prípady, kedy sa vozidlo pohybovalo príliš rýchlo, a teda detektor vozidlo detegoval, ale sledovacia metóda už dané vozidlo nepriradila k predchádzajúcej detekcii daného vozidla, a preto na každom snímku malo vozidlo inú identitu (iný identifikátor). Po vyladení parametrov pozície a rýchlosti v Kalmanovom filtri sledovača Deep SORT bol daný problém do výraznej miery odstránený. Spomínané zmeny zvýšili skóre účinnosti riešenia na 0,9077.

Prah detekcie	TP	FP	FN	Úspešnosť (%)
0,15	**4100**	248	**402**	85,67
0,20	4096	228	416	**85,81**
0,25	4078	227	421	85,72
0,30	4046	**214**	451	85,24

Tabuľka 4.2: Porovnanie úspešnosti systému na vlastnej metrike vyhodnotenia pri použití rôznych prahov pre detekciu osobných vozidiel.

Ďalší experiment súvisel so znížením tolerancie pre nájdenie výstupnej plochy. Po znížení tolerancie zo 100 pixelov na 50 pixelov skóre účinnosti mierne narástlo, a to na 0,9079.

Experimentované bolo tiež s prahom detekcie pre nákladné vozidlá. Výsledky, zobrazené v tabuľke 4.3, sa pre použité prahy líšili len mierne. Najlepšiu úspešnosť dosahovalo riešenie s prahom 0,5, avšak pri vyhodnotení na evalvačnom serveri (zobrazené v tabuľke 4.4) vykazoval mierne lepšie výsledky prah 0,6.

Prah detekcie	TP	FP	FN	Úspešnosť (%)
0,50	**4112**	241	**402**	**85,83**
0,60	4096	228	416	85,81
0,70	4088	**226**	427	85,60

Tabuľka 4.3: Porovnanie úspešnosti systému na vlastnej metrike vyhodnotenia pri použití rôznych prahov pre detekciu nákladných vozidiel.

Posledné experimenty sa týkali zvýšenia rýchlosti spracovania vykonané Ing. Jakubom Špaňhelom, ktorému sa podarilo zvýšiť skóre efektivity na 0,8212. Celkové skóre po poslednom experimente dosahovalo hodnotu 0,8829, čo je zároveň aj konečným výsledkom. Vyhodnotenia jednotlivých experimentov je možné vidieť v tabuľke 4.4.

Rýchlosť spracovania

Experimenty boli vykonávané na stroji s grafickou kartou NVIDIA GeForce RTX 2080 Ti. Pred vylepšením rýchlosti systému Ing. Jakubom Špaňhelom dosahovala rýchlosť spracovania priemerne 7,37 snímok za sekundu a približne 299 minút videozáznamov bolo spracovaných za 420 minút. Po modifikácii dosahovala rýchlosť priemerne 13,76 snímok za sekundu

Pokus	Prah detekcie osobné/nákladné	Tolerancia výstup. plochy	mwRMSE	Účinnosť	Efektivita	Skóre
001	0.25 / 0.60	100 px	4.8219	0.8993	0.0000	0.6295
002	0.25 / 0.60	100 px	5.5757	0.8820	0.6725	0.8192
003	0.20 / 0.60	100 px	4.4035	0.9077	0.6776	0.8387
004	0.20 / 0.60	100 px	4.4063	0.9076	0.6666	0.8353
005	0.20 / 0.60	50 px	4.4324	0.9079	0.6669	0.8356
006	0.20 / 0.50	50 px	4.4161	0.9076	0.6660	0.8352
007	0.20 / 0.70	50 px	5.5757	0.8820	0.6671	0.8176
008	0.20 / 0.60	50 px	4.3616	0.9094	0.7868	0.8726
009	**0.20 / 0.60**	**50 px**	**4.3616**	**0.9094**	**0.8212**	**0.8829**

Tabuľka 4.4: Zhrnutie jednotlivých experimentov, ktoré boli vyhodnotené na evalvačnom serveri súťaže.

Poradie	ID tímu	Meno tímu	Skóre
1	99	Everest	0.9389
2	110	CSAI	0.9346
3	92	INF	0.9292
4	26	Orange-Control	0.8936
5	22	psl2020	0.8852
6	**74**	**GRAPH@FIT BUT**	**0.8829**
7	6	KISTI	0.8540
8	119	PES	0.8254
9	80	HCMUS	0.8064
10	65	BUPT-MCPRL	0.7933
11	40	Insight-DCU	0.7785
12	70	CUIP	0.6922
13	75	Albany_NCCU	0.3116

Tabuľka 4.5: Umiestnenie najlepších tímov súťaže AI City Challenge.

a spracovanie videozáznamov trvalo 225 minút, čo znamená, že systém umožňuje spracovanie reálnom čase. Keďže zobrazenie výstupu v čase spracovania a zápis do súboru sú náročné operácie, tak v prípade zvolenia týchto možností, je rýchlosť spracovania nižšia.

Umiestnenie

Každému účastníkovi súťaže bolo poskytnutých 10 pokusov vyhodnotenia riešenia na vyhodnocovacom serveri súťaže AI City Challenge 2020. Do výsledného rebríčka bol uložený najúspešnejší pokus každého tímu. Riešenie tejto práce spolu s prispením pána Ing. Jakuba Špaňhela, ktorý zvýšil rýchlosť celého systému, sa celkovo umiestnilo na peknom 6. mieste. Celkový rebríček najlepšie umiestnených tímov je možné vidieť v tabuľke 4.5.

4.5 Zhodnotenie a budúci vývoj

Systém s úspešnosťou približne 90,94% poskytuje smerodajný pohľad na vyťaženosť zaznamenanej cestnej premávky. Výsledný systém môže pomôcť analyzovať premávku najmä na križovatkách, čo môže dopomôcť k lepšej organizácii celkovej premávky, prípadne k lepšiemu časovaniu križovatiek. Táto skutočnosť môže viesť k zmierneniu preťaženia premávky na konkrétnych križovatkách. Systém navyše po určitých úpravách umožňuje dosahovať rýchlosť spracovania v reálnom čase, a preto si myslím, že výsledný systém s miernymi úpravami je možné použiť v praxi.

Pomerne veľa zo vzniknutých chýb sa týkalo detekcií. Preto ako najväčší priestor na zlepšenie vidím práve v tejto problematike. Vzhľadom na veľký pokrok v problematike počítačového videnia a obzvlášť detekcie objektov v posledných rokoch sa dá očakávať, že tento pokrok bude pokračovať ďalej a v najbližšej dobe sa bude zväčšovať presnosť detektorov a rovnako aj ich rýchlosť. Lepšie výsledky by tiež mohli byť dosiahnuté pri natrénovaní modelu YOLO alebo inej detekčnej metódy na dátovej sade zachytávajucej najmä vozidlá. Aj napriek tomu, že v sledovacej metóde Deep SORT bola konvolučná neurónová sieť, ktorá bola používaná ako extraktor príznakov, natrénovaná na širokej dátovej sade zaoberajúcej sa sledovaním ľudí, pracovala táto metóda spoľahlivo aj pri sledovaní vozidiel. Natrénovanie tejto siete na vhodnejšej dátovej sade sledujúcej vozidlá by taktiež mohlo mierne zvýšiť celkovú úspešnosť systému.

Kapitola 5

Záver

Cieľom tejto práce bolo vytvoriť systém pre automatickú detekciu, sledovanie a počítanie pohybujúcich sa osobných a nákladných vozidiel v jednotlivých smeroch vo videu. V úvode práce boli popísané súčasné metódy riešenia problematiky detekcie a sledovania objektov založené na konvolučných neurónových sieťach. V ďalšej časti boli porovnané jednotlivé metódy pre detekciu a sledovanie objektov, a následne boli vybrané konkrétne metódy spolu so zdôvodnením výberu. Pre riešenie úlohy bola použitá detekčná metóda YOLOv3 v kombinácii so sledovacou metódou Deep SORT. Práca bola zameraná na riešenie úlohy súťaže AI City Challenge 2020, ktorej sa aj prostredníctvom výskumnej skupiny Graph@FIT zúčastnila.

Problémy detekčnej a sledovacej metódy boli riešené vlastným spracovaním trajektórií vozidiel. Pre každé vozidlo boli ukladané doplňujúce informácie o jeho pohybe, ktoré boli v prípade potreby použité pre predpoveď ďalšieho pohybu alebo spojenie trajektórií vozidla, ktoré bolo re-identifikované pod novou identitou.

Pre účely prvotného experimentovania so systémom bola zriadená dátová sada zachytávajúca pohyb osôb a automobilov, ktorá bude dostupná pre ďalšie použitie. Dátová sada poskytnutá pre riešenie súťaže bola ručne anotovaná a následne bol vytvorený jednoduchý program slúžiaci pre vyhodnotenie úspešnosti riešenia, ktorý porovnal vytvorený textový súbor s anotáciami a textový súbor, ktorý je výstupom systému.

Výsledný systém dosiahol na základe metriky vyhodnotenia súťaže AI City Challenge úspešnosť 90,94%, pričom pred modifikáciou rýchlosti Ing. Jakubom Špaňhelom dokázala na grafickej karte NVIDIA GeForce RTX 2080 Ti spracovávať videozáznamy s rýchlosťou priemerne 7,37 snímok za sekundu. S celkovým skóre 0,8829 obsadila práca v súťaži konečné 6. miesto.

Literatúra

[1] Bewley, A., Ge, Z., Ott, L., Ramos, F. a Upcroft, B. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, s. 3645–3649.

[2] *CS231n Convolutional Neural Networks for Visual Recognition* [online]. [cit. 2020-05-08]. Dostupné z: https://cs231n.github.io/convolutional-networks/.

[3] *CUDA Zone* [online]. 2020 [cit. 2020-04-25]. Dostupné z: https://developer.nvidia.com/cuda-zone.

[4] *NVIDIA cuDNN* [online]. 2020 [cit. 2020-04-25]. Dostupné z: https://developer.nvidia.com/cudnn.

[5] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. a Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*. jún 2010, zv. 88, č. 2, s. 303–338.

[6] Gandhi, R. *Support Vector Machine — Introduction to Machine Learning Algorithms* [online]. 2018 [cit. 2020-04-28]. Dostupné z: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[7] Girshick, R. B. Fast R-CNN. *ICCV*. 2015.

[8] Girshick, R. B., Donahue, J., Darrell, T. a Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*. 2014.

[9] Huang, R., Pedoeem, J. a Chen, C. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, s. 2503–2510.

[10] Hui, J. *MAP (mean Average Precision) for Object Detection* [online]. 2018 [cit. 2020-04-30]. Dostupné z: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a311737.

[11] Karen Simonyan, A. Z. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*. 2015.

[12] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classifiation with Deep Convolutional Neural Networks. *NIPS*. 2012.

[13] Leal Taixé, L. Multiple object tracking with context awareness. *CoRR*. 2014.

[14] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B. et al. Microsoft COCO: Common Objects in Context. *CoRR*. 2014, abs/1405.0312.

[15] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. et al. SSD: Single Shot MultiBox Detector. *ECCV*. 2016.

[16] Milan, A., Leal-Taixé, L., Reid, I. D., Roth, S. a Schindler, K. MOT16: A Benchmark for Multi-Object Tracking. *CoRR*. 2016, abs/1603.00831.

[17] Müller, M., Bibi, A., Giancola, S., Al-Subaihi, S. a Ghanem, B. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. *CoRR*. 2018, abs/1803.10794.

[18] Naphade, M., Wang, S., Anastasiu, D., Tang, Z., Chang, M.-C. et al. The 4th AI City Challenge. *CVPR*. 2020.

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

[20] Redmon, J. a Farhadi, A. YOLO9000: Better, Faster, Stronger. *CoRR*. 2016.

[21] Redmon, J. a Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR*. 2018.

[22] Ren, S., He, K., Girshick, R. a Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE PAMI*. 2016.

[23] Saha, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way* [online]. 2018 [cit. 2020-05-09]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[24] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E. et al. Going Deeper with Convolutions. *CoRR*. 2014, abs/1409.4842.

[25] Uijlings, J. R. R., Sande, K. E. A. van de, Gevers, T. a Smeulders, A. W. M. Selective Search for Object Recognition. *International Journal of Computer Vision*. 2013.

[26] Wojke, N., Bewley, A. a Paulus, D. Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, s. 3464–3468.

[27] *YOLO: Real-Time Object Detection* [online]. 2020 [cit. 2020-04-25]. Dostupné z: https://pjreddie.com/darknet/yolo/.

[28] Yu, F., Li, W., Li, Q., Liu, Y., Shi, X. et al. POI: Multiple Object Tracking with High Performance Detection and Appearance Feature. *CoRR*. 2016, abs/1610.06136.

[29] Zhao, Y., Barnes, N., Chen, B., Westermann, R., Kong, X. et al. *Image and Graphics 10th International Conference, ICIG 2019, Beijing, China, August 23–25, 2019, Proceedings, Part I: 10th International Conference, ICIG 2019, Beijing, China, August 23–25, 2019, Proceedings, Part I*. Január 2019. ISBN 978-3-030-34119-0.

[30] Zhao, Z., Zheng, P., Xu, S. a Wu, X. Object Detection with Deep Learning: A Review. *CoRR*. 2018.