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

ODHAD RYCHLOSTI AUTOMOBILU VE VIDEU

VEHICLE SPEED ESTIMATION FROM VIDEO

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PAVEL HÁJEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ROMAN JURÁNEK, Ph.D.
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Hájek Pavel, Bc.**
Obor: Inteligentní systémy
Téma: **Odhad rychlosti automobilů ve videu**
Vehicle Speed Estimation from Video
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte metody pro detekci, sledování automobilů.
2. Najděte nebo pořiďte vhodný dataset pro měření ruchlosti.
3. Navrhněte a implementujte metodu pro měření rychlosti ze statické, libovolně umístěné kamery.
4. Experimentujte s navrženou metodou a vyhodnoťte její výsledky.
5. Vytvožte prezentační materiály.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Bod 1.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Juránek Roman, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá návrhem a tvorbou aplikace pro odhadování rychlosti automobilů jak ze záznamu kamery, tak realtime ze streamu. Práce popisuje kalibraci kamery, detekci a trackování automobilů a odhad jejich rychlosti a věnuje se robotickému operačnímu systému pro nějž je určená. Vytvořená aplikace používá knihovnu OpenCV k většině úkonů, pro přístup k videu používá knihovnu FFmpeg. Výsledky může vypsat na terminál, do souboru nebo je například dále publikovat v rámci ROSu. Aplikace je napsaná v jazyce C++, některé části v jazyce Python.

Abstract

This master's thesis describes design and development of an application for estimation of vehicle speed from both recorded video file and from camera stream. It explains the procecess of a camera calibration, vehicle detection and tracking and describes the robot operating system as a target platform. The application uses library OpenCV for most of tasks, to access video application uses a FFmpeg library. Results can be printed to the terminal window, they can also be logged to a file or published in ROS. Application is written in C++ language, some parts in Python.

Klíčová slova

Docker, ROS, OpenCV, FFmpeg, doprava, rychlost, kalibrace

Keywords

Docker, ROS, OpenCV, FFmpeg, traffic, speed, calibration

Citace

HÁJEK, Pavel. *Odhad rychlosti automobilu ve videu.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Juránek Roman.

Odhad rychlosti automobilu ve videu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Romana Juránka Ph.D. Uvedl jsem všechny literární prameny a publikace, z nichž jsem čerpal.

. . . . . .
Pavel Hájek
22. května 2017

Poděkování

Rád bych poděkoval vedoucímu Ing. Romanu Juránkovi PhD. za vedení práce, rady, trpělivost a shovívavost a přítelkyni za podporu a korekci tohoto textu.

Obsah

1 Úvod **3**

2 Existující systémy pro měření rychlosti **4**
2.1 Okamžitě změřená rychlost . . . 4
2.1.1 Radar . . . 4
2.1.2 LIDAR . . . 6
2.1.3 IR senzory . . . 7
2.1.4 Senzory ve vozovce . . . 8
2.1.5 Kamera . . . 9
2.2 Průměrná rychlost . . . 9

3 Metody používané pro zjištění rychlosti automobilu **11**
3.1 Existující metody . . . 11
3.2 Detekce objektů . . . 13
3.2.1 Model pozadí . . . 13
3.2.2 Metody založené na extrakci příznaků . . . 14
3.2.3 Metody založené na hledání vzorů pohybu . . . 15
3.3 Tracking detekovaných objektů . . . 16
3.3.1 Point tracking . . . 16
3.3.2 Kernel tracking . . . 18
3.3.3 Adaptivní trackování . . . 18
3.4 Kalibrace kamery . . . 19
3.4.1 Metody založené na čarách rozdělující pruhy vozovky . . . 19
3.4.2 Metody založené na pohybu automobilů . . . 20
3.4.3 Manuální kalibrace . . . 20

4 Návrh systému pro měření rychlosti automobilů **22**
4.1 Docker . . . 23
4.2 Robot Operating System . . . 23
4.2.1 ROS balíčky . . . 24
4.2.2 ROS Topics . . . 24
4.2.3 ROS service . . . 25
4.3 Komunikace částí . . . 25
4.3.1 Video . . . 25
4.3.2 Kalibrace . . . 25
4.3.3 Detekované automobily . . . 26
4.3.4 Měření rychlosti . . . 26
4.3.5 Rychlost detekce . . . 26

4.4 Čtení videa . . . 27
4.5 Kalibrace . . . 27
4.5.1 Diamond space . . . 28
4.5.2 Detekce prvního VP . . . 29
4.5.3 Detekce druhého VP . . . 32
4.5.4 Výpočet třetího VP a fokální vzdálenosti . . . 33
4.6 Detekce . . . 34
4.7 Tracking . . . 34
4.8 Klasifikace . . . 35
4.9 Odhad rychlosti . . . 35
4.9.1 Vypočítání vzdálenosti . . . 36
4.9.2 Získání měřítka . . . 37
4.10 Implementace . . . 37
4.10.1 Ovládání aplikace . . . 37
4.10.2 Změna rozlišení videa . . . 39
4.10.3 Hlavní uzel . . . 39
4.10.4 Detekční uzel . . . 39

5 Testování a experimenty **40**
5.1 Dataset . . . 40
5.2 Detekce . . . 40
5.2.1 Tracking . . . 42
5.3 Klasifikace . . . 43
5.4 Kalibrace . . . 43
5.5 Měření vzdáleností . . . 45
5.6 Měření rychlosti . . . 46

6 Závěr **48**

Literatura **49**

Přílohy **54**

A Instalace **55**
A.1 Nový Docker image . . . 55

Kapitola 1

Úvod

V dnešní době jsou CCTV kamery instalovány téměř na každém kroku a mnoho z nich monitoruje i dopravu. Analýza dopravy, zvláště v dnešní době, kdy je ve světě extrémní množství aut, je velmi perspektivní a užitečný obor. Zároveň je analýza dopravy zajímavým oborem počítačového vidění. Díky obyčejné kameře lze získat ohromné množství informací o dopravě - rychlost jednotlivých automobilů, jejich typ, SPZ a s vhodnými klasifikátory by šel nepochybně určit i přesný model auta.

Hlavním cílem této práce je vypočítat rychlost automobilů. K tomu bude potřeba nejprve zkalibrovat používanou kameru, dále detekovat automobily v obraze a sledovat je. Z historie pohybu automobilů bude možné získat jejich rychlost v relativních jednotkách, tu pak bude potřeba přepočíst na nějakou standardní jednotku. Výsledky této práce mohou být použity například v oboru inteligentních křižovatek v chytrých městech nebo na dálnicích pro zjištění plynulosti dopravy.

Jako platformu tato práce využívá robotický operační systém (ROS), který je detailněji popsán v kapitole 4.2. ROS sám navíc poběží jako Docker kontejner.

Tato zpráva nejprve popisuje z teoretického hlediska metody počítačového vidění pro analýzu, které tato práce využívá. V kapitola 4 je detailně popsán návrh celého systému a jsou zde popsány některé klíčové části implementace.

Tato práce byla přijata na studentskou konferenci Excel@FIT 2017.

Kapitola 2

Existující systémy pro měření rychlosti

Měření rychlosti bylo dříve synonymem pro detekci překročení rychlostního limitu, nicméně dnes v době rozšiřování inteligentního řízení dopravy se lze často setkat i s měřiči rychlosti, které informují řídící jednotku nějakého inteligentního prvku o rychlosti a četnosti přijíždějících vozidel. Rychlost a počet automobilů lze také využít pro nejrůznější statistiky o provozu a tím i k různým optimalizacím atp.

Dnes už většina zařízení napříč různými typy obsahuje malý, ale plnohodnotný počítač, který umožňuje alespoň částečně vzdálenou údržbu. K té je také zapotřebí, aby zařízení byla připojena k internetu nebo do jiné sítě. Vzhledem k vestavěnému počítači není těžké zařízení připojit např. na WiFi nebo ke GSM síti.

Zařízení fungující jako detektory překročení rychlosti bývají navíc opatřeny kamerou, která bývá umístěna tak, aby mohla vyfotografovat SPZ automobilu ideálně i s řidičem. Zařízení pak do obrazu vloží informace o času přestupku, detekované rychlosti a často i paritní data pro ověření integrity.

Detektory rychlosti lze rozdělit do dvou skupin – ty, které změří okamžitou rychlost a ty, které změří průměrnou rychlost na určitém úseku. Byť i získání okamžité rychlosti má charakter průměrného měření, měřený úsek je tak krátký (maximálně několik desítek metrů), že lze hovořit o měření okamžité rychlosti. Oproti tomu úseky průměrného měření mohou být dlouhé i desítky kilometrů.

2.1 Okamžitě změřená rychlost

Tento druh rychlostních kamer změří přesnou rychlost pohybujícího se objektu v jednom časovém okamžiku. Velkou výhodou těchto měřících zařízení je to, že nemusí být pouze statické, ale díky své často malé velikosti mohou být snadno přenosné a mohou být uloženy např. do automobilů či na trojnožku. V následujících podkapitolách jsou detailněji popsány nejobvyklejší systémy pro změření okamžité rychlosti.

2.1.1 Radar

Klasické dopravní radary využívají Dopplerova jevu. Dvě hlavní části tvoří vysílač a přijímač. Vysílač vysílá směrem k měřenému objektu impulzy v mikrovlnném pásmu (frekvence se pohybují v řádu jednotek MHz až desítek GHz na základě očekávané vzdálenosti měřiče a měřeného objektu), ty se od něj odrazí a přijímač je opět zachytí. Díky tomu, že se cílový

Obrázek 2.1: Nalevo radar RAMER 7M ukrytý v masce chladiče policejní Škody Octavie[2], napravo stejný typ – RAMER 7 – pouze ve statickém provedení na trojnožce[3].

objekt pohybuje, frekvence odražených vln se od dopadnutých liší díky Dopplerově efektu. V praxi se používají zařízení s vyšší frekvencí, řádově alespoň 10GHz, neboť na nižších frekvencích může docházet k rušení ze strany radarů a např. nechtěnému otevření/zavření cizí garáže, interferenci se senzory automatického otevírání dveří a dalšími zařízeními, která pracují na stejných kmitočtech. Použití těchto frekvencí omezuje dosah radarů na přibližně sto metrů. Tato vzdálenost se se zvyšující se frekvencí dále snižuje.

Většina typů radarů zvládne změřit rychlost jak přibližujícího se, tak oddalujícího se objektu.

Rychlost měřeného objektu v lze vypočítat pomocí vztahu 2.1, kde f je frekvence vyslaných vln a Δf je rozdíl frekvencí vyslaného a přijatého signálu. Tyto radary předpokládají, že rychlost automobilů nebo obecně měřených objektů je výrazně nižší než rychlost světla c, což je v dopravě zaručeno.

$$v = \frac{\Delta f}{f}\frac{c}{2} \tag{2.1}$$

Co se týče mikrovlnných radarů používá Policie České republiky (PČR) výhradně radary RAMER výrobce Ramet Kunovice. Tento výrobce dlouhodobě spolupracuje s PČR a dodává jak statická zařízení (např. RAMER 7CCD-U, RAMER 7 M-V a RAMER 7 M-S), tak mobilní (RAMER AD 9 C, RAMER 7M, RAMER VB a RAMER 7F)[1]. Všechny tyto typy vysílají vlny na frekvenci 34GHz, nebo 34,4GHz. U PČR se tyto radary vyskytují nejčastěji v provedení tzv. "bradavic"– transcieverů zabudovaných do masky automobilu, resp. v provedení na trojnožce či plně zabudované pro stacionární typy.

Proti těmto typům radarů se lze bránit dvěma způsoby – pasivním a aktivním antiradarem. Aktivní antiradar, který je v České republice nelegální [4], aktivně ruší dopravní radar vysíláním šumu na frekvenci radaru, čímž zahltí přijímač, který pak nedokáže identifikovat "správné"vlny odražené od automobilu.

[1]Více detailů lze nalézt na http://www.ramet.as/policejni-radary

[3]Obrázek převzat z https://www.antiradary.net/img/radary/radar-ramer-3.jpg

[3]Obrázek převzat z http://powertuning.hu/images/stories/cikk_kepek/radarok_europaban/radar/ramer7%20szerk.jpg, upraveno

[4]Dle zákona 361/2000 Sb., § 3 odst. 6

Obrázek 2.2: Na obrázcích se nachazí typy LTI 20-20 UltraLyte[7](vlevo) a ProLaser III[8], které běžně používá PČR

Oproti tomu pasivní antiradar pouze detekuje radary – poslouchá na běžných frekvencích používaných vysílači radarů a v případě detekování signálu pouze upozorní řidiče.

Výhodou této metody měření rychlosti je, že je velmi přesná – dobře zkalibrované zařízení za ideálních povětrnostních podmínek změří rychlost automobilu s přesností do $1km/h$. Nevýhodou je vysoká pořizovací cena těchto měřičů (např. zařízení RAMER 10C stojí 800 000Kč[5]).

2.1.2 LIDAR

LIDAR (akronym "light detection and ranging") je zařízení, které funguje podobně jako radar (viz výše). Tato zařízení nevyužívají Dopplerův efekt, místo toho přímočaře změří časový úsek mezi vysláním světelného signálu a dopadem jeho odrazu, čímž získá aktuální vzdálenost automobilu a senzoru. Opakováním těchto měření v pravidelných intervalech lze změřit aktuální rychlost. Tento způsob má spíše povahu měření průměrné rychlosti, nicméně vzhledem k tomu, že jednotlivé měření se provádějí s frekvencí až 100Hz, je naměřená rychlost považována za okamžitou rychlost.

Typické zařízení tohoto druhu emituje pulzy trvající 30ns vlnové délky 905nm o výkonu 50mW. Vyzařovací úhel činí přibližně 3mrad.

Největší výhodou LIDARU oproti radarům je to, že vyslaný laserový paprsek dopadne na jediný cílový bod, kdežto signál vyslaný radarem má vyzařovací rozsah větší (do 10°). To znemožňuje zejména vícecestné šíření signálu. Další výhodou oproti klasickým radarům je jejich cena – měřič LTI 20-20 UltraLyte, který mj. používá PČR, stojí 3 032.44 USD[6].

Tento druh vybavení se u nás začal objevovat poměrně nedávno – v roce 2008.

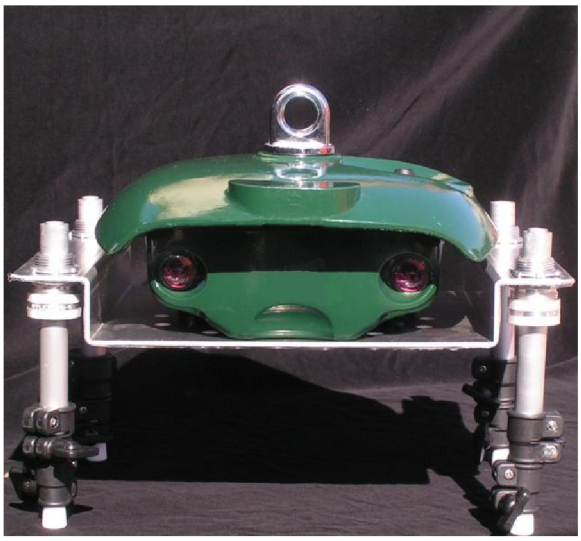
I proti LIDARům se lze bránit pasivními a aktivními rušičkami. Pasivní však v tomto případě nemusí být stoprocentně úspěšné v detekci kvůli těžší identifikaci příslušného světelného signálu.

[5]Zdroj: http://www.rozhlas.cz/pardubice/zpravodajstvi/_zprava/spickovy-policejni-radar-za-850-tisic-ztrati-se-v-davu-a-zmeri-i-protijedouci-auta--1427542

[6]Více modelů výrobce na https://purchasing.idaho.gov/pdf/contracts/Police%20Radar/Laser%20Tech%20Pricing%20and%20Ordering%20Info.doc

[8]Převzato z: https://www.antiradary.net/img/radary/laser-lti-4.jpg, více detailů na stránce výrobce: http://www.lasertech.com/Micro-Digi-Cam.aspx?s=1

[8]Převzato z: http://www.cesbrod.cz/media/obrazky/radarmp.jpg



Obrázek 2.3: Přijímající zařízení v přenosném provedení, obvyklejší bývají zařízení napevno přibudovaná, zejména kvůli citlivosti na vibrace, otřesy a posunutí[9].

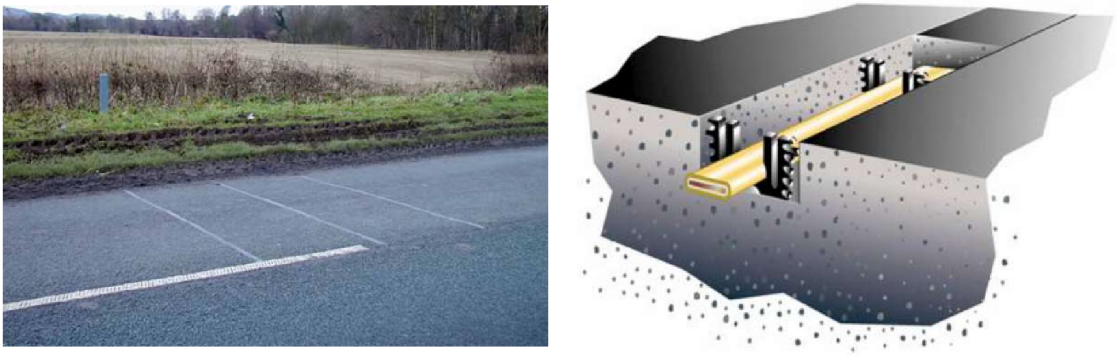
Aktivní rušičky jsou u nás a ve většině Evropské unie zákonem zakázané. Tyto systémy detekují LIDARy, po dopadu laserového paprsku na měřené vozidlo začne rušička ve zlomku vteřiny (dle předprogramovaného algoritmu) vysílat pulsy proti měřiči rychlosti. Zároveň je řidiči signalizován poplach. Laserový měřič není schopen změřit rychlost vozidla a měřící laserové zařízení není schopno vyhodnotit rychlost vozidla. Tyto aktivní rušičky lze u nás nicméně provozovat.

2.1.3 IR senzory

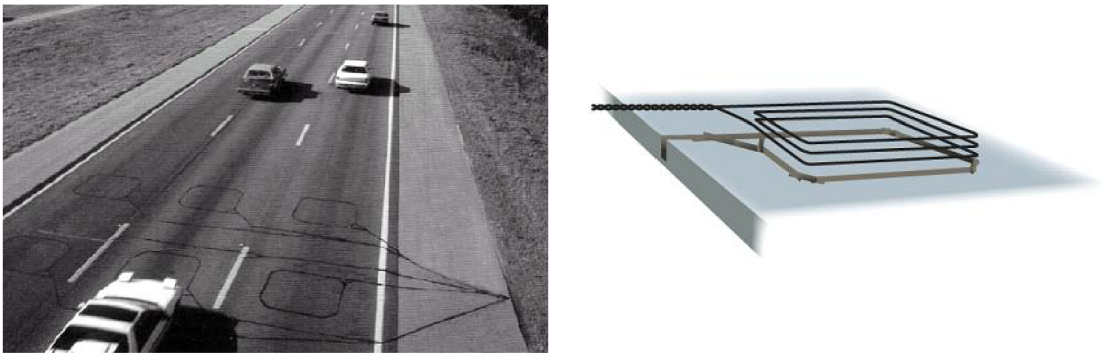
Infračervené zaznamenávače dopravy – The Infra-Red Traffic Logger (TIRTL) – jsou multifunkční senzory, které se typicky ve dvojicích umisťují po stranách jízdního pruhu. Jsou to multifunkční zařízení, která se dají použít k detekci a počítání automobilů, měření jejich rychlosti, detekci projetí na červenou, klasifikace typu auta atd. Tato zařízení byla vyvinuta v Austrálii jako sekundární senzor inteligentních křižovatek vedle klasických kamer. Postupně se začaly rozšiřovat např. do USA, západní Evropy atd., nicméně v České republice se v hojné míře nepoužívají.

Tento systém se skládá ze dvou zařízení umístěných po stranách jízdního pruhu kolmo na směr dopravy. Jedno zařízení je vysílačem, druhé přijímačem. Vysílač vysílá většinou dva rovnoběžné světelné paprsky a přijímač detekuje jejich přerušení – systém tedy funguje na principu dvou světelných závor. Vzhledem k tomu, že paprsky jsou rovnoběžné a je známá vzdálenost mezi nimi, lze z časového rozdílu přerušení jednotlivých paprsků dopočítat rychlost objektu, jež paprsky přerušil. Dále lze získat informaci o délce přerušení, čímž lze v kombinaci s informací o rychlosti získat délku vozidla a tím i přibližně jeho druh (osobní, nákladní).

[9]Model *TIRTL* výrobce Injaz National General Enterprises L.L.C. Převzato z: https://upload.wikimedia.org/wikipedia/commons/thumb/a/ac/TIRTL_portstand.jpg/220px-TIRTL_portstand.jpg



Obrázek 2.4: Na obrázku vlevo jsou patrné šedé krycí elastické prvky napříč jízdním pruhem, které přenáší tlak na senzor pod nimi[10]. Na pravém obrázku je znázorněn způsob uložení piezo senzoru ve vozovce[11]. Senzory se nachází obvykle okolo 3cm pod povrchem.



Obrázek 2.5: Na levém obrázku je patrné uložení několika dvojic indukčních smyček za účelem měření rychlosti[12]. Je patrné, že jednotlivé senzory jsou poměrně velké. Na pravém obrázku je zobrazena klasická indukční smyčka[13].

2.1.4 Senzory ve vozovce

Detektory uložené v povrchu vozovky využívají buď piezoelektrického jevu, nebo elektromagnetické indukce. Oba tyto na váhu citlivé detektory detekují projetí automobilu a informují o tom řídící jednotku. Těchto senzorů je pak ve vozovce uloženo více se známou vzdáleností a řídící jednotka vypočte rychlost vozidla.

Piezoelektrický senzor využívá schopnosti některých krystalů generovat elektrické napětí při jejich deformaci, tento efekt se nazývá piezoelektrický jev. Tyto senzory generují napětí, které je přímo úměrné síle jeho stlačení.

Indukční dopravní senzor se skládá z jedné nebo několika smyček vodiče ve vozovce a kontrolní jednotky. Řídící jednotka vysílá energii do smyčky s rezonanční frekvencí (10kHz–200kHz v závislosti na provedení). Cívka ve vozovce se pak chová jako rezonanční obvod – při průjezdu vozidlo narušením elektromagnetického pole senzoru dá vzniknout vířivým proudům, které mění indukčnost cívky. Tato změna je detekována řídící jednotkou.

[11] Převzato z: https://www.speedcamerasuk.com/images/speed-camera-types/ds2-mobile-speed-camera-location.jpg

[11] Převzato z: http://www.te.com/content/dam/te-com/images/sensors/global/products/IMG_TrafficSensors_RoadTrax_BL.jpg

Výhodou piezoelektrických senzorů ve srovnání s indukčními je jejich velikost – jednak mohou být umístěny dočasně na povrch vozovky bez nutnosti stavebního zásahu a jednak při jejich vkládání do již hotové cesty není zapotřebí tak velký zásah. Při instalaci piezoelektrického senzoru stačí vytvořit několik málo centimetrů širokou a hlubokou drážku, zatímco při instalaci indukční smyčky je potřeba úplně odstranit horní vrstvu vozovky o velikosti jednotek metrů čtverečních. Nevýhodou pak je nestálá odezva výstupního napětí, která se liší zejména při různých teplotách povrchu.

Jak piozoelektrické, tak indukční senzory se používají spíše jako klasické detektory, jejich užití je dnes běžné zejména u inteligentních křižovatek. Jako rychlostní měřiče se v České republice nepoužívají, naopak zcela běžné jsou například ve Velké Británii.

2.1.5 Kamera

Dopravní kamery, ve smyslu klasických video kamer, se (zatím) nepoužívají jako vynucovací prostředek pro dodržování rychlostních limitů, nicméně v posledních přibližně pěti letech se stále častěji objevují tato měřící zařízení buď jako součást inteligentních řídících systémů, nebo samostatně – například za účelem sběru statistických dat apod.

Typická sekvence kroků pro spočítání rychlosti automobilu je nejprve detekce vozů a s ní související jejich trackování. Tím systém získá pro každý automobil množinu bodů v obraze, které odpovídají jeho detekovaným polohám. Následně je potřeba provést rekonstrukci scény, čímž jsou získány původní souřadnice automobilů v reálném světě. Tuto rekonstrukci lze samozřejmě provádět za běhu, jak budou přicházet jednotlivé tracky – zde záleží na konkrétní implementaci. Pro úspěšnou a přesnou rekonstrukci – a tím i přesné výsledky měření – je potřeba znát informace o poloze kamery – její kalibraci.

Existující aplikace bývají buďto implementovány jako software běžící na řídícím serveru nebo jako dedikovaný hardware, který může být přímo součástí kamery, nebo plně externí.

Výhodou měření rychlosti automobilů touto metodou je zejména nízká cena, protože stačí pouze kamera a případně počítač s vhodným softwarem, což jsou oproti statisícovým radarům levné položky. Další výhodou je i snadná rozšiřitelnost použitého softwaru pro mnoho dalších úkonů, mj. detekce jízdních pruhů, detekce špatným směrem jedoucích automobilů, klasifikace typu a značky vozidel a další. Pro video kamery hovoří i snadnost jejich instalace – není potřeba zasahovat do vozovky a ani do její blízkosti jako u senzorů ve vozovce a klasických radarů.

2.2 Průměrná rychlost

Nejobvyklejší způsob změření průměrné rychlosti je za použití úsekového radaru. Toto zařízení se skládá ze dvou částí, typicky kamer, umístěných určitou vzdálenost od sebe (řádově od stovek metrů po jednotky kilometrů). První kamera detekuje automobil, extrahuje jeho poznávací značku a uloží ji spolu s časem projetí. Druhá se pak při rozpoznání nějaké registrační značky podívá do své databáze, vyhledá čas projetí okolo první kamery a z rozdílu časů vypočítá průměrnou rychlost.

Kritickou částí těchto systémů je správné rozpoznání SPZ. Rozpoznavač musí dosahovat vysoké přesnosti a zároveň musí být robustní vůči různým druhům SPZ pro různé státy.

[13] Převzato z: https://ops.fhwa.dot.gov/freewaymgmt/publications/frwy_mgmt_handbook/images/fig15-1.jpg

[13] Převzato z: http://support.diamondtraffic.com/knowledgemanager/images/loop_wire_installed_cross_s.gif

V případě měření pro statistické účely lze pro vyhledávání v databázi první kamery povolit například chybovost jednoho znaku – pokud se nalezená SPZ bude lišit o jeden znak, bude se považovat za totožnou.

Kapitola 3

Metody používané pro zjištění rychlosti automobilu

Cílem této práce je implementovat metodu, která z proudu dat z kamery dokáže spočítat rychlosti zachycených automobilů. Tato video kamera může být jakéhokoliv typu, metoda musí být robustní vůči jejímu rozlišení, počtu snímků za vteřinu a dalším parametrům. Kamera se předpokládá být statická, nad vozovkou (na přesné výšce nezáleží) a úhel mezi směrem pohledu kamery a vozovkou by neměl překročit hodnotu 45°.

V následujících kapitolách jsou nejprve popsány existující práce, které se snaží vypočítat rychlost jedoucího automobilu, dále pak jednotlivé metody počítačové grafiky používané při řešení tohoto problému.

3.1 Existující metody

V této kapitole jsou popsány některé práce zabývající se měřením rychlosti automobilů. Obecně se dá říci, že nejdůležitější částí těchto systémů je proces získání parametrů kamery – její kalibrace. Práce se dají rozdělit do dvou skupin – používající zkalibrovanou kameru a používající nezkalibrovanou kameru. V prvním případě systém již předem zná přesnou polohu kamery – její výšku nad vozovkou, horizontální vzdálenost od středu pruhu (případně vozovky) a úhly natočení ve všech osách, ve formě například *KRT* [20]. V opačném případě systémy musí kalibrační údaje získat z videa a to buď plně automaticky [17], nebo s pomocí uživatele [56][34].

Dále se systémy liší ve způsobu detekce automobilů – založené na detekci příznaků [59][27][14], modelu pozadí [34][56][52] atd. – a ve způsobu trackování vozidel – Kalmanův filter [52], KLT tracker [34], prosté přiřazení podle nejnižší ceny, ...

Jednotlivé metody jsou detailněji rozebrány dále.

L.Grammatikopoulos et al., 2005

Systém [20] navržený pány Grammatikopoulem, Karrasem a Petsou v roce 2005 plně automaticky počítá rychlost vozidel. Počítání rychlosti je zde založeno na měření 1D vzdálenosti mezi výskyty vozidel na rektifikovaných snímcích. Pro korektní rektifikaci musí systém získat polohu úběžníku připadajícímu ke směru jízdy vozidel. Toho je docíleno extrakcí přibližně svislých hran pomocí Canny detektoru. Tyto hrany, protažené do nekonečna, hlasováním určí polohu úběžníku. Autoři předpokládají orientaci kamery přesně nad vozovkou tak, že úhel mezi směrem jejího pohledu a směrem dopravy je nulový. Díky tomuto předpokladu

pak lze tvrdit, že úběžník, který připadá k vodorovnému směru kolmému ke směru dopravy se nachází v nekonečnu. Rektifikace probíhá pomocí homografní matice, jejím výsledkem je ekvivalentní pohled na scénu shora pomocí paralelní projekce.

Autoři detekují vozidla pomocí modelu pozadí, které pak morfologicky uzavřou a na jednotlivých snímcích detekují bloby. Bloby jsou pak trackovány pomocí normalizované křížové korelace. Během tohoto procesu systém zároveň počítá aktuální rychlost vozidel a to pomocí ujeté vzdálenosti od posledního výskytu. Výsledek je v relativních jednotkách, ty se přepočítají na klasické jednotky srovnáním nějaké známé vzdálenosti v obraze (autoři používají šířku jízdního pruhu).

Autoři systém testovali na vlastním datasetu složeném z několika videí. Získali referenční údaje díky dvaceti automobilům se známou rychlostí změřenou pomocí GPS. Z publikované výsledků vyplývá, že přesnost měření byla přibližně $\pm 3km/h$.

Slabým místem tohoto systému je potřeba znát přesně alespoň jednu reálnou vzdálenost v obraze.

I.Sina et al., 2013

Z uvedených prací vybočuje I.Sina et al., 2013 [47], kteří navrhli metodu pro počítání vozidel a odhad jejich rychlosti při špatných světelných podmínkách, zejména ve tmě. Detekci automobilů zjednodušili na detekci jejich světlometů – blobů na binárním snímku. Centroidy těchto blobů je třeba spárovat, k tomu autoři používají kombinaci metod *area-centroid-difference* a *normalized-cross-correlation*. První z nich používá jako kritéria vzdálenosti centroidů a jejich velikost, druhá pak symetrii blobů.

Pro trackování využívají velmi jednoduchou metodu – pokud je vzdálenost mezi detekovanými automobily na následujících snímcích menší než stanovený práh, jsou tato vozidla prohlášeno za jedno. Nevýhod této trackovací metody je více, například si nedokáže poradit s tím, pokud se automobil "schová"za jiný a není tedy určitý čas detekován.

Pro zjištění rychlosti vozidla autoři navrhují dvě metody – pomocí euklidovské vzdálenosti a s využitím modelu dírkové kamery. První z nich v podstatě násobí ujetou vzdálenost v pixelech kalibračním koeficientem, čímž je získána ujetá vzdálenost. Je zřejmé, že tato metoda není přesná, chyby měření rychlosti se pohybují okolo až $11km/h$.

Oproti tomu při využití modelu dírkové kamery systém spočítá reálnou vzdálenost automobilu od kamery v metrech. K tomu je potřeba znát *KRT* kalibraci kamery, kterou je potřeba systému manuálně zadat.

Experimenty byly provedeny na jejich vlastním datasetu, během jehož získávání několikrát projelo auto, jehož rychlost byla známá. Tím autoři získali koeficient pro získání skutečné rychlosti z relativní. Prezentovaná průměrná chyba činí $3.325km/h$.

D.C.Luvizon et al., 2016

Tento tým v práci *A Video-Based System for Vehicle Speed Measurement in Urban Roadways* [34] z roku 2016 používá model popředí pro detekci jedoucích automobilů, autoři navrhli několik vylepšení pro jeho získání. Obraz modelu popředí je rozdělen na několik částí, z nichž každá odpovídá jednomu vozidlu. V těchto částech systém hledá registrační značku.

Při hledání registrační značky v oblastech zájmu využívají autoři poznatku Zheng et al., že okolní oblasti značky mívají dlouhé horizontální hrany a malé množství náhodného šumu. Tyto hrany autoři získají pomocí Sobelova operátoru, filtrují je na základě délek (omezení minimální i maximální délky pevně definovanými hodnotami) a zbylé hrany uloží do binárního obrazu. Na něj je aplikována dilatace (s jádrem 1×7). Následně jsou v obrázku hledány

bounding boxy stanovených rozměrů a v jejich obsazích je vyhledána skutečná registrační značka, na které jsou nalezeny *good features to track* [46] a ty jsou trackovány. Vyhledání trackovatelných příznaků se děje pouze jednou při prvním nalezení každého automobilu.

Pro trackování příznaků autoři používají Kanade-Lucas-Tomasi (KLT) tracker ([32] a [58]).

Autoři této práce kalibrují kameru pomoci čtyř známých bodů na vozovce, čímž získají homografní matici H. Během trackování automobilů systém získá vektory pohybu jednotlivých vozidel v pixelech, které jsou pomocí matice H převedeny na metry. Nevýhodou tohoto přístupu je nutnost ručního měření přímo v terénu.

Tým autorů prováděl experimenty na datasetu dvaceti FullHD videí s 30.15 FPS. Z výsledků vyplývá, že chyba u 96% analyzovaných rychlostí byla v intervalu $(-3; 2)$ km/h od příslušných skutečných rychlostí.

3.2 Detekce objektů

V systémech, které analyzují dopravní situaci, je detekce objektů první činnost, která je pro každý snímek provedena. I v dnešní době je detekce pohybujících se objektů v dynamické scéně stále velký problém a množství vědeckých prací se jím zabývá.

Obecně se tyto metody dle způsobu činnosti dají rozdělit na metody:

1. založené na modelu pozadí,
2. založené na extrakci příznaků a
3. metody založené na porovnávání snímků a hledání pohybu.

3.2.1 Model pozadí

Proces extrakce pohybujícího se popředí z uloženého statického obrazu modelu stálého pozadí se nazývá segmentace popředí. Finální obraz popředí je získán prahováním rozdílu daného snímku a obrazu pozadí.

Tento způsob detekce pohybujících se objektů je jeden z nejčastějších nejen v oblasti analýzy dopravy. Jeho největší nevýhodou je snížená schopnost adaptace na jakékoliv změny v pozadí, ať už je to změna osvětlení či například pohyb stromů díky větru. Proto je tato metoda hojně vylepšována různými způsoby.

Parametrické metody

Velké zlepšení přinesly parametrické techniky, například zavedení Gaussova rozložení pravděpodobnosti pro každý pixel. Tyto techniky unimodální pravděpodobnostní funkce hustoty pro modelování každého z pixelů a během chodu systému upravují na základě příchozích snímku hodnoty rozdělení. Gaussova funkce každého z pixelů v čase t je dána středem μ_t a rozptylem σ_t. Pixely jsou pak klasifikovány pomocí vztahu 3.1 jako součást dynamického popředí, případně vztahem 3.2 jako součást pozadí.

$$\frac{|I_t - \mu_t|}{\sigma_t} > k \tag{3.1}$$

$$\frac{|I_t - \mu_t|}{\sigma_t} < k \tag{3.2}$$

V těchto vztazích značí I_t intenzitu daného pixelu a k je práh. Čím je hodnota k nižší, tím se zvyšuje pravděpodobnost chybné identifikace pozadí coby popředí a naopak, čím je vyšší, může docházet k ignorování částí popředí.

Dalším vylepšením je filtrování mediánem – jednotlivé pixely modelu pozadí jsou určeny mediánem několika pixelů ze starších snímků, což ale zavádí nutnost ukládat tyto snímky do bufferu. Toto vylepšení je založeno na předpokladu, že jednotlivé pixely se během času budou měnit pouze o malé hodnoty, čímž je umožněna určitá benevolence na například změny osvětlení.

Lepší výsledky za cenu složitějšího výpočtu přináší aplikace GMM (Gaussian mixed model). Každý pixel je v tomto případě modelován několika Gaussovými funkcemi. Nevýhodou tohoto vylepšení je jeho nízká výpočetní rychlost, horší adaptace na změny a v neposlední řadě i nižší schopnost správně identifikovat objekty pohybující se různými směry a rychlostmi v jedné sekvenci snímků. Nicméně byly publikovány práce, které zlepšují jak rychlost výpočtu, tak schopnost adaptace. Tato metoda dokáže také správně rozpoznat a vyřadit z modelu popředí stín objektu.

Neparametrické metody

Existují nicméně i metody, které nepoužívají funkce hustoty pravděpodobnosti – nejsou parametrické. Tyto techniky používají historii pixelů k vytvoření pravděpodobnostní reprezentace pozorování pomocí omezené historie snímků.

Mezi tyto metody patří KDE (Kernel Density Estimation), kde pravděpodobnost, že zkoumaný pixel je součástí pozadí, je odhadnuta interpolační metodou Parzenova okna.

3.2.2 Metody založené na extrakci příznaků

Tato množina metod používá ke své činnosti vizuální informace – barvy, tvary, textury. Na rozdíl od metod používajících model pozadí zde není podmínka pohybu detekovaných objektů ve scéně.

Dřívější práce používaly pro detekci symetrii a hranové příznaky [23][21]. Dnes se používají komplexnější a robustnější metody pro popis příznaků, které často umožňují nejen přímou detekci, ale i klasifikaci. Mezi nejčastější používané nástroje patří Scale Invariant Feature Transformation (SIFT), speeded up Robust Features (SURF), Histogram of Oriented Gradient (HOG) a Haarovy příznaky.

SIFT

SIFT algoritmus slouží k nalezení a popisu lokálních příznaků. Metodu je potřeba naučit na trénovací množině, může být použita i jako klasifikátor. SIFT se skládá ze čtyř hlavních kroků:

1. detekce extrémů uvnitř scale-space (3D reprezentace obrazu o několika vrstvách, které značí zvyšující se měřítko původního obrazu),
2. postupné zpřesňování polohy významných bodů,
3. přiřazení orientace významným bodům a
4. sestavení deskriptoru významných bodů.

Vylepšení této metody spočívají v modifikaci deskriptorů zvyšováním jejich abstrakce. SIFT se v analýze dopravy dá použít jako detektor jakýchkoliv objektů (od celých vozidel po jejich části [62][41][22]), detektor vhodných příznaků k trackování [19][15] nebo klasifikátor vozidel [35].

HOG

Histogram orientovaných gradientů je deskriptor příznaků používaný pro detekci objektů, vychází z toho, že objekt ve scéně lze popsat na základě gradientů hran. Původně byl HOG navrhnut pro detekci chodců.

V mnoha pracích byly použity symetričnosti v HOG spolu s klasickým histogramem gradientů pro detekce vozidel. V kombinaci s touto metodou popisu příznaků se velmi často používá SVM (Support Vector Machines) – metoda strojového učení pro klasifikaci [55][30][57][51]. Rozšířením této metody bylo docíleno zjištění orientace automobilu v prostoru [61].

Nevýhodou HOG je náročnost jeho výpočtu, nicméně v posledních letech se doba výpočtu zrychlila díky zapojení GPU.

Haarovy příznaky

Haarovy příznaky se dají rozdělit na hranové, čárové a středové. Jsou to v podstatě černobílé obdélníky lišící se uspořádáním černé a bílé. Tyto obdélníky se rozloží přes aktuálně klasifikovanou část obrazu, následně se zvlášť sečtou části obrazu ležící pod bílou a černou částí obdélníku a sumy se od sebe odečtou a normalizují.

Detektory používající Harrovy příznaky (např. Viola-Jones detektor) používají trénovací algoritmus AdaBoost, jehož cílem je sestavení kaskády dílčích klasifikátorů, které pak během rozhodování hlasují.

Tyto příznaky bývají často používány při detekci automobilů díky své rychlosti a přesnosti [55][40][13][48][49].

Detekování auta coby kolekce komponent

Tyto relativně nové techniky, na rozdíl od výše uvedených, nakládají s automobilem coby s několika spojenými částmi. Většinou dělí vozidlo na přední část (čelní sklo, maska, kapota), boční čast (dveře, kola) a zadní část (zadní sklo). Jednotlivé části jsou detekovány na základě jejich velikosti, tvaru a hran, poté jsou na základě analýzy prostorových vztahů identifikována vozidla.

B.F.Lin et al. [29] používají k detekci kombinaci SURF a hranových příznaků, kde jsou jednotlivé části identifikovány detekcí klíčových bodů. Zhang et al. [62] detekují auta jako kolekce komponent za použití SIFT příznaků a skryté CRF (Conditional Random Field) klasifikace. V jiné práci zaměřené na realtime detekci automobilů [50] autoři zvlášť detekují přední část a zadní část vozidel, které jsou potom spojeny na základě prostorových omezení pomocí SVM.

3.2.3 Metody založené na hledání vzorů pohybu

Aktuálně stále častějším problémem je detekce vozidel ve tmě [12][47], systémy k tomu často používají detekci a tracking založený na známých naučených vzorech chování společně se

segmentací obrazu a hledání vizuálních vzorů pro detekci předních, resp. koncových světel [12].

V případě, že kamera je uložená v jedoucím automobilu, se často používá optical flow – hledání vzorů pohybu objektů. Tato technika, která může být doplněná například o detekci na základě symetrie, se používá mj. pro detekci v obraze perzistentních bodů – pravděpodobných okolních automobilů jedoucích stejným směrem a podobnou rychlostí [5] [24].

3.3 Tracking detekovaných objektů

Trackování vozidel se děje za účelem předpovídání polohy automobilu na následujícím snímku, přiřazení stejných detekovaných vozidel na různých snímcích. Díky trackování vozidla zná systém celou jeho trajektorii a body výskytu v obrazech. V oblasti analýzy dopravy se tracking používá ke zjištění dynamických atributů automobilů, zejména rychlost a směr pohybu. Většina trackovacích algoritmů používá podobný druh přiřazení – pokud je vzdálenost mezi dvěma následujícími výskyty nízká, lze tyto dva výskyty považovat za jedno vozidlo.

Před trackováním vozidla musí být zajištěna unikátní reprezentace všech detekovaných vozidel. Používají se reprezentace pomocí regionu s vozidlem, kontury (typicky uzavřené venkovní), modelu vozidla, příznakového vektoru (vhodný zejména pro trackování oblastí s malou velikostí) nebo v případě detekce založené na modelu pozadí se nabízí trackování centroidů nalezených blobů. Každý z tohoto způsobu popisu automobilu se hodí pro jiný tracker.

3.3.1 Point tracking

Trackování bodů lze popsat jako hledání korespondencí mezi nalezenými objekty, které jsou reprezentovány body napříč snímky. Tento druh trackingu se dělí na deterministický a statistický. V případě point trackingu je vyžadováno, aby externí detektor nacházel body v každém snímku.

Deterministické

Tento druh definuje cenu přiřazení každého z objektů snímku $t-1$ k objektům snímku t, přičemž se snaží, aby tato cena byla co nejmenší. Na cenu přiřazení mají vliv následující omezení:

- **proximita** – očekává se, že lokace objektů v jejich snímcích se nebude příliš lišit,
- **maximální rychlost** – stanovuje maximální vzdálenost mezi přiřazovanými objekty a
- **nízké změny rychlosti** – předpokládá se, že ani rychlost, ani směr pohybu se nebudou zásadně měnit.

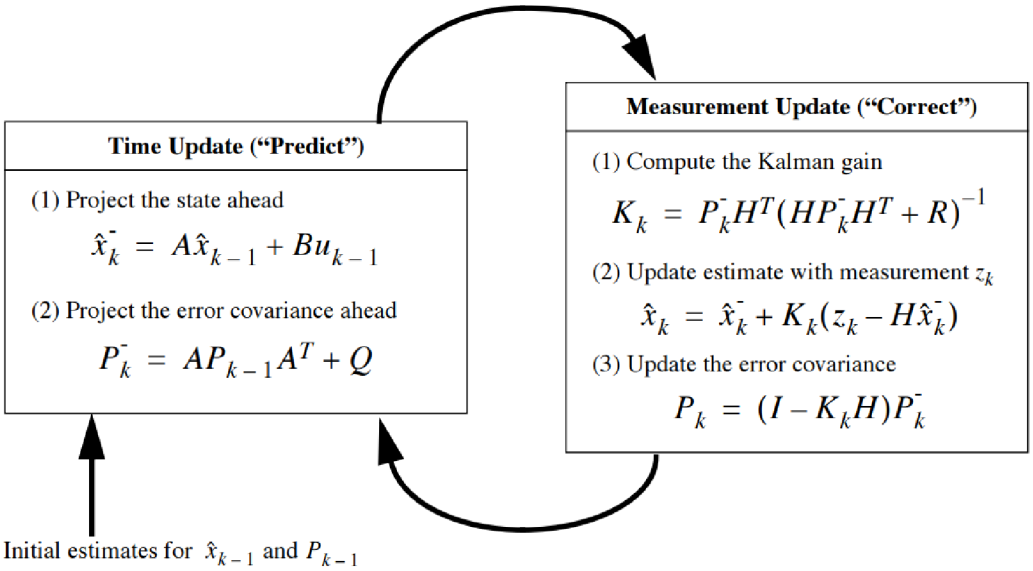
Statistické

Statistické metody berou v úvahu nepřesnosti měření a modelu při odhadu stavu objektu. Používají stavovou reprezentaci k modelování vlastností objektů jako směr, rychlost a akcelerace.

Kalmanův filtr Jeden z nejpoužívanějších nástrojů pro předpovídání pohybu objektu z historie je Kalmanův filtr. Za Kalmanovým filtrem stojí Rudolf Emil Kálmán, americký matematik a elektroinženýr původem z Maďarska, který jej představil roku 1960. Jedná se o nástroj, který zvládne z historie poloh automobilu ve snímcích určit očekávanou následující polohu. Kalmanův filtr má nicméně širší spektrum využití. Obecně se snaží o aproximaci diskrétního procesu definovaného lineární stochastickou diferenciální rovnicí[60]:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

Matice A a B určují vztah nového výstupu na předchozím výstupu x_{k-1}, respektive na předchozím vstupu u_{k-1}.



Obrázek 3.1: Princip Kalmanova filtru[1]

Tento algoritmus pracuje ve dvou fázích – *predict* a *correct*. Fáze *predict* slouží k předpovědi následujícího stavu, zatímco fáze *correct* slouží k aktualizaci modelu skutečnými hodnotami.

Kalmanův filtr pro trackování používá mnoho prací [37][38][53]. Jeho nevýhodou v tomto oboru je předpoklad linearity a normálně rozložená šumová charakteristika, což ovšem odstraňuje modifikace – rozšířený Kalmanův filtr (EKF) [31].

Částicový filtr Tato metoda byla představena v roce 1993 a od této doby se objevila řada různých podob tohoto algoritmu. Částicové filtry jsou zobecněním Kalmanových filtrů. Jejich základní ideou je použití množiny náhodných vzorků s přiřazenými vahami a odhad nové polohy na základě této pravděpodobnosti.

Částicový filtr představuje implementaci Bayesova filtru vycházející ze simulace Monte Carlo metody. Na rozdíl od Kalmanova filtru odhaduje novou polohu prvků na základě náhodného vzorku předchozích částic (předchozích poloh). Výhodou použití vzorků k odhadu nové pozice je, že neklademe žádné omezení na tvar hustoty pravděpodobnosti výskytu objektu v obraze. Vzorků musí být dostatečné množství, podle teorie Monte Carlo lze náhodná

[1] Obrázek převzat z: [60]

pravděpodobnostní veličina popsat mj. dostatečným množstvím náhodných vzorků. V každém dalším stavu jsou jednotlivým částicím přiřazovány váhy na základě shody odhadované reprezentace dané částice se skutečnou hodnotou, které odpovídají pravděpodobnosti výskytů daných objektu na daném místě v aktuálním snímku.

Stejně jako Kalmanův filtr bývají částicové filtry často používány pro trackování vozidel, případně bodů obecně[8][39][36].

3.3.2 Kernel tracking

Tento druh trackingu typicky dostane původní bod jako nějakou miniaturní oblast, kterou se pak snaží najít v novém snímku.

KLT

Kanade-Lucas-Tomasi feature tracker je rychlý algoritmus k hledání zadaných bodů na snímcích. Tato metoda je založena na dvou pracích – Lucas-Kanade [33] algoritmu pro určení optického toku, ze které vychází hlavní část metody, a Tomasi-Kanade [58] metodě pro získání co nejvhodnějších příkazů k trackování.

Často se používá pro tracking pouze práce Lucase a Kanadeho [6]. Díky tomu, že využívá stanovené okolí sledovaného bodu, je metoda velmi přesná a méně náchylná vůči šumu v obrazu. Její nevýhodou pak je pevně stanovená velikost okna, ve kterém jsou obrazy bodů hledány. Tento nedostatek byl odstraněn zavedením pyramidového schématu [9].

Pánové Shi a Tomasi navrhli dodatečné způsoby verifikace, že nalezené příznaky jsou validní [46].

Vstupem algoritmu je množina bodů, které je třeba vyhledat na dalším snímku, a další snímek. Celý algoritmus vychází z předpokladů:

- jas sledovaných bodů se jejich pohybem nemění,
- pohyb bodů mezi jednotlivými následujícími snímky je relativně malý a
- blízké body patřící jednomu objektu se pohybují po podobných trajektoriích.

Nevýhodou tohoto algoritmu je fakt, že pokud sledovaný objekt jako celek zmizí z obrazu, stále se snaží detekovat jeho význačné body a poměrně velké množství z nich chybně přiřadí na dalším snímku.

Tento způsob trackingu využívá mj. Kanhere et al. [26]. Saunier et al. [43] využívají KLT v kombinaci s Kalmanovým filtrem.

3.3.3 Adaptivní trackování

Adaptivní trackování se od výše zmíněných metod liší ve stupni závislosti na externím detektoru – v tomto případě od něj dostane pouze informaci a prvním výskytu sledovaného objektu. Dále se již o detekci daného objektu tyto metody starají sami. Tyto sledovací systémy se tedy skládají z více částí, kromě trackingu a detekce typicky obsahují i část pro aktualizaci modelu detektoru[42].

TLD

Tracking-learning-detection [25] se v současné době řadí mezi nejrobustnější a nejefektivnější metody sledování objektů. Algoritmus kromě samotného trackování používá techniky pro detekování objektů a učení. Jedná se o poměrně novou metodu trackování, kterou navrhl v rámci své dizertační práce Zdeněk Kálal v roce 2011. TLD umožňuje dlouhodobé sledování objektu, který se navíc může během trackování několikrát dostat za hranice obrazu, to vše bez nutnosti znalosti prostředí a s minimem znalostí o objektu samotném.

Tato metoda ve své základní podobě funguje tak, že nejprve na vstupu dostane bounding box objektu zájmu. Detektor TLD se naučí jeho vzhled a na dalších snímcích už objekt detekuje sám. Jako tracker tato metoda využívá Kanade-Lucas. Tracker je periodicky reinicializován modelem detektoru. Třetí část algoritmu – část učící – získává data z obou výše jmenovaných částí, porovnává je, zjišťuje chyby a také vybírá části snímku, na kterých se bude detektor doučovat a tedy zpřesňovat svůj model.

Tato metoda se nevyskytuje v systémech pro analýzu dopravy tak často jako předchozí jmenované, nicméně si začíná nacházet cestu i do tohoto oboru [10][63].

3.4 Kalibrace kamery

Jednou z nejdůležitější částí systémů pro přesnou analýzu dopravy je kalibrace kamery. Přesnost těchto systémů má přímou závislost na přesnosti kalibrace. V případě, že není známá dokonale přesná poloha kamery včetně natočení a měřítka scény, je třeba tyto informace získat z videa – zkalibrovat kameru. V praxi se využívají nejčastěji dva druhy kalibračních informací – matice KRT a úběžníky. Metody kalibrace kamery lze rozdělit dle několika kritérií:

- zda-li jsou schopné určit kalibrační údaje automaticky bez nutnosti interakce s uživatelem a
- dle způsobu získání kalibračních údajů.

Metody lze dále rozdělit dle způsobu reprezentace získaných kalibračních dat. V dopravní tematice se jako velmi vhodné jeví využití úběžníků. Tyto reprezentace lze rozdělit na:

- **pomocí jednoho úběžníku** – v tomto případě se typicky používá úběžník příslušející směru dopravy,
- **pomocí dvou úběžníků** – zde se k výše zmíněnému úběžníku přidává další – buď v reálu vertikální, nebo kolmý na směr dopravy a příslušející rovině vozovky a
- **pomocí třech úběžníků**.

Dalšími způsoby jsou matice KRT, homografní matice atp.

V následujících kapitolách jsou popsány různé kalibrační metody, které jsou rozděleny dle způsobu získání údajů.

3.4.1 Metody založené na čarách rozdělující pruhy vozovky

Tato skupina metod se snaží extrahovat dělící čáry mezi jízdními pruhy a po krajích. Z nich lze snadno jejich protažením získat úběžník, který leží ve směru dopravy. Dále se dá z informací o značení pruhů získat i měřítko scény – na základě průměrné šířky jízdního pásu, průměrné délky segmentu přerušované části apod.

Grammatikopoulos et al. [20] kalibrují kameru pomocí úběžníku ve směru dopravy. Toho je docíleno extrakcí přibližně svislých hran pomocí Canny detektoru. Tyto hrany, protažené do nekonečna, určí polohu úběžníku pomocí metody nejmenších čtverců. Autoři předpokládají orientaci kamery přesně nad vozovkou tak, že úhel mezi směrem jejího pohledu a směrem dopravy je nulový. Díky tomuto předpokladu pak lze tvrdit, že úběžník, který připadá k vodorovnému směru kolmému ke směru dopravy se nachází v nekonečnu. Oblast zájmu je rektifikována pomocí homografní matice, jejím výsledkem je ekvivalentní pohled na scénu shora pomocí paralelní projekce.

Cathy et al. [11] také získávají úběžník pomocí čar, podobně k tomu využívají metodu nejmenších čtverců. Pro detekování čar autoři nejdříve získají model pozadí, aplikací Sobelova operátoru získají gradienty a odstraní přibližně vodorovné hrany ($\pm 22.5°$). Na získané gradienty aplikují OTSU prahovaní, čímž zůstanou pouze přibližně horizontální hrany. Tyto hrany, protažené do nekonečna, aproximují metodou nejmenších čtverců hledaný úběžník. Autoři navíc získají měřítko scény na základě průměrné délky segmentu přerušované čáry, které porovnají s ve skutečnosti naměřenou délkou.

3.4.2 Metody založené na pohybu automobilů

Další možností je extrakce automobilů, jejich trackování a odhad polohy úběžníku ve směru dopravy na základě jejich cesty.

Dubská et al. [17] kalibruje kameru detekcí úběžníků, které definují rovinu vozovky. Autoři pomocí modelu pozadí získají pohybující se elementy. Na nich pro získání úběžníku ve směru dopravy detekují příznaky vhodné k trackování a na dalších snímcích je pomocí KLT trackeru vyhledají, čímž získají množinu vektorů, jejichž přímky pak hlasují v nástroji *DiamondSpace* (viz dále). Extrakce druhého úběžníku, kolmého na směr dopravy, také využívá model pozadí. Z popředí jsou detekovány hrany, které jsou filtrovány (přibližně vodorovné, nesměřující k prvnímu úběžníku), z nichž určité procento nejlepších opět hlasuje v nástroji *DiamondSpace*. Na základě těchto dvou úběžníku autoři dopočítají intrinsické a extrinsické parametry, ovšem za několika předpokladů (čtvercové pixely, *principal point* uprostřed obrazu, atd.). Pro hlasování čar autoři používají nástroj *DiamondSpace*, který je založený na kaskádě Houghových transformací a který mapuje nekonečný prostor R^2 do konečného prostoru, ve kterém jsou hledána maxima.

Schoepflin et al. [44] používají mapu aktivity, kterou aktualizují na základě pohybu popředí a ze které se dají extrahovat jednotlivé pruhy, respektive jejich dělící čáry. Místo, kde se tyto čáry, protažené do nekonečna, protnou, je považováno za úběžník ve směru dopravy. Druhý úběžník kolmý na směr dopravy je extrahován na základě protažených spodních hran automobilů. Pro získání měřítka pro měření rychlosti systém očekává jednu známou vzdálenost v reálu, z níž toto měřítko spočítá.

Dalším přístupem je detekce poznávacích značek a jejich tracking [18].

3.4.3 Manuální kalibrace

Poslední skupina metod je založená na manuálním zadání různých údajů. Může se jednat přímo o matice KRT nebo znalost několik bodů na povrchu vozovky včetně jejich vzdáleností atd.

[2] Obrázek převzat z: [45]

Obrázek 3.2: Mapa nejčastějšího pohybu, jsou zde patrné dělící čáry mezi pruhy[2]

D.C.Luvizon et al. [34] kalibrují kameru pomocí čtyř známých bodů na vozovce, čímž získají homografní matici H. Během trackování automobilů systém získá vektory pohybu jednotlivých vozidel v pixelech, které jsou pomocí matice H převedeny na metry.

Kapitola 4

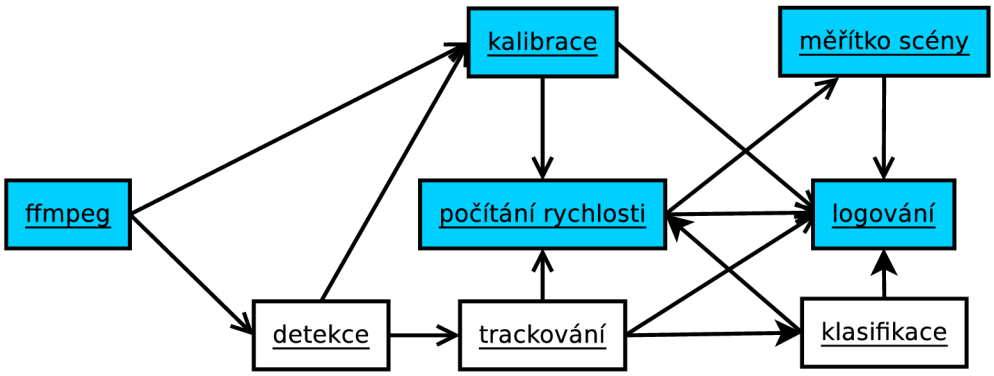
Návrh systému pro měření rychlosti automobilů

V této kapitole je popsána hierarchie celého systému a detailně jeho části. Cílem této práce je vybrat nejvhodnější metody pro dílčí úkony (detekce, kalibrace, ...) a zkombinovat je do funkčního systému. Na této práci spolupracuji s Radkem Vopálenským, jehož cílem je klasifikace vozidel. Jeho výsledky klasifikace v této práci využívám pro získání měřítka scény.

Aplikace poběží na vrstvě robotického operačního systému (ROS), který je popsán dále. ROS byl zvolen zejména kvůli výborné podpoře škálovatelnosti aplikace na jednotlivé bloky a mechanismům komunikace mezi nimi. Vzhledem ke složitosti ROSu a náročnosti jeho instalace na jakýkoliv jiný operační systém, než je pár podporovaných, bylo zvoleno použití ROSu v rámci Docker kontejneru. To umožňuje snadnou instalaci, přenášení a další činnosti s obrazy.

Výsledná aplikace ze vstupu, kterým je URL video souboru, vypisuje rychlost automobilů, které musí ve streamu detekovat a trackovat. Pro tento účel je potřeba zkalibrovat kameru. Dále, vzhledem k tomu, že vzdálenosti a pak i rychlosti jsou získávány v nějakých relativních jednotkách, je třeba získat měřítko mezi těmito relativními jednotkami a skutečnými veličinami (např. km/h).

Celý systém využívá služeb knihovny OpenCV, která je navíc solidně podporována ROSem.



Obrázek 4.1: Očekávané schéma bloků a jejich vzájemné komunikace

Obrázek 4.1 znázorňuje hierarchii jednotlivých částí systému a schéma jejich komunikace. Bloky, které jsou celé implementovány v rámci této práce, mají modré pozadí.

Celý systém je rozdělen do následujících částí – ROS balíčků:

- **traffic**, který sdružuje informace od všech ostatních *nodů* a v ucelené formě je vypisuje,
- **traffic_ffmpeg**, jehož vstupem je URL streamu / souboru s videem a jeho výstupem snímky videa zapisované na zadaný *topic* typu,
- **traffic_calibration**, který čeká na video stream a poskytuje kalibrační údaje ve formě vanishing pointů a údaj, zda-li je výstup validní,
- **traffic_detection**, který také čeká na snímky videa a dále vrací informace o nalezených automobilech ve formě jejich bounding boxů,
- **traffic_tracking**, který v podstatě k výstupu předchozího bloku přiřadí informace, o které auto se jedná, ve formě ID,
- **traffic_speed**, který z informací o pozicích automobilů počítá jejich rychlost a zároveň počítá měřítko scény a
- **traffic_classification**, který čeká na snímky (výřezy) jednotlivých automobilů, které postupně klasifikuje a vrací ID třídy, která reprezentuje jeho typ a značku.

4.1 Docker

Docker je open-source virtualizační nástroj určený primárně pro Linux. Oproti klasickým nástrojům pro virtualizace (Oracle VirtualBox, VMWare) se liší v použití tzv. kontejnerů, které využívají virtualizační a izolační nástroje, které jsou již zabudovány v linuxovém jádře. Díky tomuto není potřeba emulovat celý operační systém, což velmi snižuje systémové nároky.

Docker kontejner tedy obalí nějaký software kompletním systémem souborů tak, že z pohledu tohoto software běží, jakoby byl klasicky nainstalován. Na webu je rozsáhlá knihovna kontejnerů[1]. Docker se používá, pokud je příliš složité nebo zbytečné instalovat software na hostitelský PC, pro zvýšení bezpečnosti zejména webových služeb, atd.

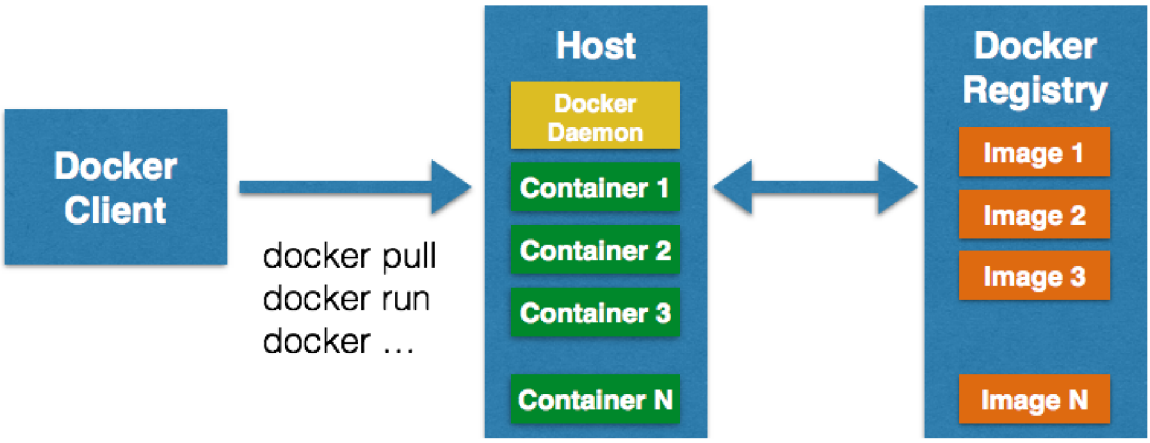
Zjednodušeně řešeno je Docker kontejner read-only, po spuštění kontejneru vznikne obraz – paměť, kterou používá k zápisu.

4.2 Robot Operating System

ROS je soubor knihoven a nástrojů fungující jako middleware prostředí pro vývoj programů pro práci s roboty. K tomuto účelu poskytuje mj. hardwarovou abstrakci, ovladače zařízení, podporu pro meziprocesovou komunikaci, simulační nástroje atd [1]. Umožňuje tedy roboty ovládat, číst data ze senzorů a ihned je zpracovávat, plánovat veškeré akce a mnoho dalšího. Programy pro ROS je možné psát v několika jazycích – např. C++, python, lisp, Java.

[1] https://hub.docker.com/explore/

[2] Obrázek převzat z: http://devopscube.com/what-is-docker/



Obrázek 4.2: Architektura Dockeru – uživatel si nejprve stáhne kontejnery z Docker Hubu, po jejich spuštění se vytvoří Docker Image. K jednomu kontejneru může uživatel vytvořit libovolné množství obrazů[2]

Pro práci s tímto middlewarem je nutné, aby běžel program *roscore*, který tvoří jeho jádro. Jednou z jeho nejdůležitějších funkcí je zajištění meziprocesové komunikace, na které je závislá většina programů.

Program využívající služeb ROSu se nazývá uzel (node) [3]. Tyto programy jsou součástí balíčků, s jejichž pomocí se i spouští. Jednotlivé uzly spolu pak komunikují. V ROSu existují dva mechanismy pro vzájemnou komunikaci procesů – topicy a služby. Topicy slouží pro jednosměrnou komunikaci, služby pak umožňují komunikaci typu dotaz–odpověď. V této práci jsou využity pouze topicy.

ROS lze nainstalovat na podporované OS dle návodů [1], pokud jej chce uživatel používat jinak než pro práci s roboty, nejjednodušší je si stáhnout Docker kontejner[3].

4.2.1 ROS balíčky

I když lze programy pro ROS distribuovat samostatně, například pouze jako zdrojový text, doporučuje se používat systém balíčků *catkin*, který nahrazuje předchozí systém *rosbuild*. Systém *catkin* je sada nástrojů sloužící pro vytváření, překlad a instalování balíčků, využívající nástroj pro automatický překlad CMake.

Pro snadné spouštění více programů a předávání parametrů je vhodná utilita *roslaunch*, využívající XML soubory pro specifikaci spouštěných programů a jejich parametrů.

4.2.2 ROS Topics

Topic je kanál mezi publisherem (tím, co na kanál zapisuje) a subscriberem (tím, co z kanálu čte). Topicy tedy slouží k jednosměrné komunikaci mezi několika uzly, kdy každý z nich před připojením deklaruje, jestli bude z topicu číst, nebo na něj zapisovat [4]. Komunikace probíhá formou výměny zpráv. Zpráva je jednoduchá datová struktura obsahující prvky různých datových typů, případně pole s nimi. Navíc mohou obsahovat i jednoduché datové struktury, vycházející z jazyka C[4] [2].

[3] http://wiki.ros.org/docker/Tutorials/Docker
[4] http://wiki.ros.org/Messages

Při připojování na topic je třeba specifikovat požadovaný typ zpráv. Ten je možné zvolit z předem definovaných[5], nebo si vytvořit nový pomocí utility *rosmsg*. Způsob připojení také závisí na tom, zda bude program z topicu číst, nebo na něj zapisovat. Při připojení na topic subscriber navíc specifikuje maximální délku fronty zpráv – kolik nevyzvednutých zpráv se uloží pro pozdější vyzvednutí. Pokud je tato fronta plná, další zprávy subscriber zahazuje.

4.2.3 ROS service

Dalším způsobem komunikace mezi procesy ROSu je využití služeb. Ty fungují systémem dotaz–odpověď. Opět platí omezení datových typů, které musí odpovídat, navíc může existovat pouze jeden démon odpovídající na dotazy pro každou službu. Formát dotazů a odpovědí se může lišit. Odpověď na dotaz by měla být vypočítána co nejrychleji, neboť volání servicu je blokující, při dlouhém čekání se můžou zahazovat data z topiců apod.

4.3 Komunikace částí

Systém pro komunikaci balíčků využívá mechanismy ROSu – zejména topicy. K tomuto účelu je třeba definovat několik nových datových typů. Definice těchto nových typů jsou umístěny v hlavním balíčku – *traffic*. V následujících podkapitolách jsou popsány všechny typy vyměňovaných informací.

4.3.1 Video

Jednotlivé snímky dekódované *ffmpeg* částí jsou posílány na *topic* spolu s informacemi o pořadovém číslu a přesném času snímku pro synchronizaci ostatních částí. Pokud nastane konec videa, část $ffmpeg$ nevyšle žádnou speciální zprávu, pouze se z *topicu* odhlásí – tuto událost mohou detekovat ostatní procesy.

4.3.2 Kalibrace

Kalibrační informace zapisuje odpovídající proces periodicky na zadaný *topic* a to jak během procesu jejich získávání, tak pokud jsou již známy. Zpráva obsahuje následující položky:

- tři úběžníky v rámci 2D prostoru obrazu,
- trojrozměrný vektor odpovídající směru k svislému úběžníku, který zároveň definuje normálu povrchu vozovky,
- fokální vzdálenost,
- *principal point* a
- informaci o tom, zda-li je příslušná zpráva validní.

V případě, že kalibrační proces již určil parametry kamery, bude je na *topic* zasílat s frekvencí $0.5HZ$. Až kalibrační část získá kalibrační údaje, odhlásí se z *topicu* s video streamem.

[5] http://wiki.ros.org/std_msgs, http://wiki.ros.org/sensor_msgs a další

4.3.3 Detekované automobily

Detekci a trackování automobilů mají na starost dvě různé části, které nicméně používají stejný formát zpráv. Ty se skládají z vektoru pro jednotlivé automobily popsané položkami:

- ID automobilu,
- místo jeho výskytu a jeho velikost a
- výřez snímku s automobilem.

Detektor posílá na svůj topic položky bez vyplněného ID. Tracker k těmto položkám přidá příslušné ID automobilu a pošle je pro pořádek na jiný topic.

4.3.4 Měření rychlosti

Měření rychlosti funguje ve dvou režimech – měření aktuální rychlosti mezi dvěma polohami pro získání přesné rychlosti a měření průměrné rychlosti v rámci snímku. První případ bude prováděn pouze pro vizualizační účely, druhý vždy když automobil opustí oblast obrazu.

Měření využívá ROS service – část, počítající rychlost čeká na dotaz a pošle odpověď. Formát dotazovací zprávy je následující:

- ID vozidla,
- číslo a čas prvního a druhého snímku, mezi kterými se určuje rychlost (v případě druhého režimu se jedná o časy prvního respektive posledního výskytu),
- poloha na prvním a druhém snímku (v případě druhého režimu nevyužito) a
- seznam bodů všech výskytů automobilu (v případě prvního režimu nevyužito).

Měřící část dostane seznam těchto dílčích zpráv.

Odpověď pak má následující formát (odpověď je ve skutečnosti opět tvořena seznamem těchto dílčích zpráv):

- ID automobilu,
- číslo pozdějšího snímku (v případě druhého režimu nevyužito),
- rychlost a
- seznam bodů se zpřesněnou polohou automobilu (v případě prvního režimu nevyužito).

4.3.5 Rychlost detekce

Pro případ, že nebude prioritou běh aplikace v reálném čase, detekční část vypisuje periodicky informace o průměrné době detekce. Této průměrné délce se pak dle specifikace uživatele může FFmpeg část přizpůsobit, čímž aplikace za určitých podmínek nebude schopna analyzovat video v reálném čase, na druhou stranu se dají očekávat přesnější výsledky, neboť detektor stihne analyzovat každý snímek.

4.4 Čtení videa

K otevření, čtení a dekódování vstupních videí aplikace používá knihovnu *FFmpeg*. Ta obsahuje veškeré potřebné nástroje a umožňuje otevřít různé druhy vstupů (nahrané video, stream) jednotně bez nutnosti parsování URL souboru (viz 4.4). Tyto úkonu má na starost část *traffic_ffmpeg*. Výstupem tohoto bloku je proud snímků videa, které jsou zapisovány na zadaný topic.

Tato část, vzhledem k výpočetně náročné detekci automobilů, může (dle nastavení uživatelem) poslouchat

FFmpeg

Projekt *FFmpeg* zaštiťuje skupinu knihoven a nástrojů pro práci s audiem a videem. Obsahuje prostředky pro nahrávání, streaming, konverzi formátů, změny velikosti, ořezávání a mnoho dalšího. Projekt je vyvíjen pod licencí LGPL a je dostupný v podobě zdrojových kódů nebo v přeložené podobě pro Windows, Linux i MacOS. Tento framework poskytuje nástroje pro běžné uživatele i vývojáře. Zřejmě nejznámějším nástrojem používajícím *FFmpeg* je VLC, dále Blender, GStreamer a mnoho dalších.

Knihovna obsahuje velkou škálu různých kodeků pro dekódování videa, mezi nejdůležitější patří `MPEG-4`, `H.264`, `AAC`, `FLAC`, atd.

Jádrem celého systému jsou knihovny pro činnosti popsané výše. Další nástroje projektu, se kterými běžný uživatel může přijít do styku, jsou založeny na těchto knihovnách. Nejdůležitější z nich jsou:

- **libavcodec** – obsahuje všechny podporované audio/video kodeky,
- **libavformat** – zahrnuje podporu pro práci s multimediálními kontejnery,
- **libavutil** – pomocná knihovna obsahující užitečné nástroje, datové typy a další, které jsou sdílené napříč všemi knihovnami,
- **libswscale** – knihovna umožňující editování formátu, velikosti a barvové reprezentace obrazu.

Z knihovny *FFmpeg* se v roce 2011 odštěpil projekt *libav*, jehož cílem je odstranit některé nedostatky zmíněné výše při snaze zachování vzájemné kompatibility.

4.5 Kalibrace

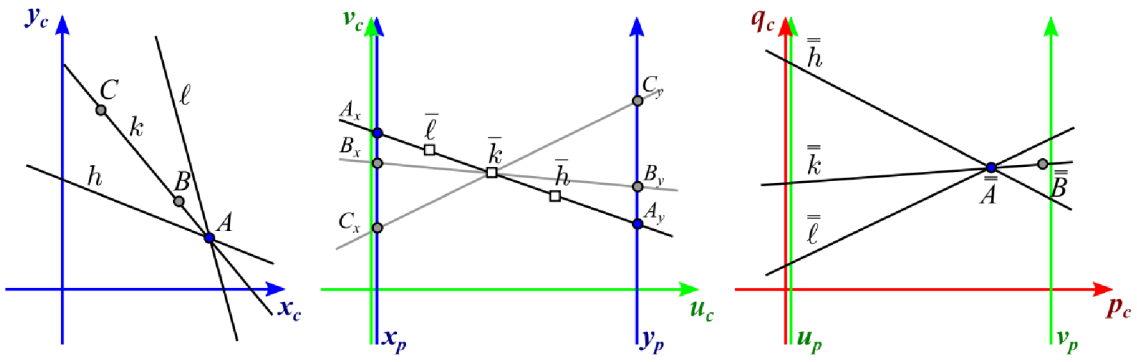
Vstupem kalibrace jsou snímky videa a výstupem jsou kalibrační informace složené ze třech VP, ohniskové vzdálenosti a dále informace, zda je tato informace validní.

Pro výběr vhodných regionů (patřící automobilům) si kalibrační blok udržuje model pozadí, po jehož odečtení se získají oblasti patřící automobilům.

Je potřeba zavést podmínky, kdy lze kalibrační informace prohlásit za validní a tím v podstatě kalibraci ukončit. Ty jsou popsány v následujících kapitolách. Dále je popsána Houghova transformace a nástroj *diamond space*, které slouží k získání průsečíku velkého množství přímek.

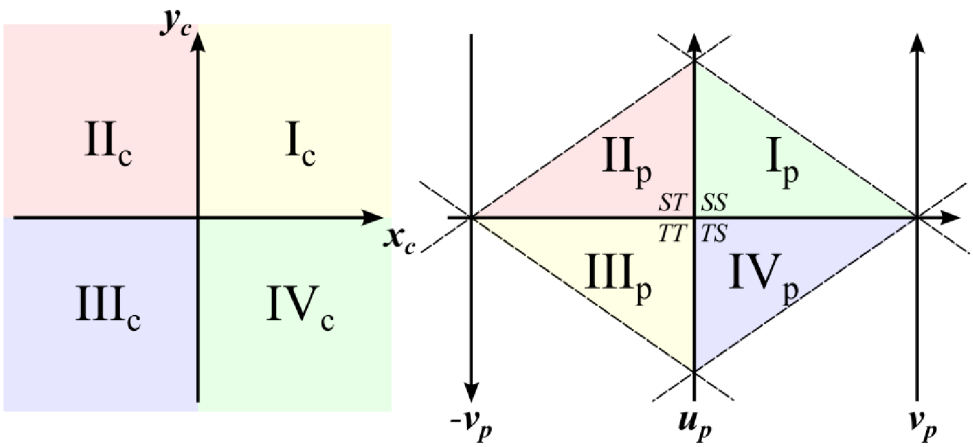
4.5.1 Diamond space

V této práci používám Houghovu transformaci pro detekci průsečíku velkého množství přímek. Houghova transformace [7] je metoda extrakce příznaků z obrazu za cílem nalezení objektů určité třídy pomocí hlasování.



Obrázek 4.3: Vlevo původní souřadný systém. Uprostřed systém po aplikaci první transformace – z přímek se staly body a naopak. Zde je zároveň definovaný nový systém (v_c, u_c), který je transformován na paralelní (vpravo). [6]

Metoda Diamond space (dále jen DS) se snaží transformovat celý nekonečný systém $\mathcal{R}^2$ do nějakého konečného souřadného systému. Při tomto mapování je použit paralelní souřadný systém, tedy takový systém, kde jsou obě osy rovnoběžné. V tomto paralelním systému jsou body zobrazeny jako přímky (úsečky) a přímky jako body. To je ilustrováno na obrázku 4.3, prostřední část. DS tuto transformaci provádí dvakrát (viz obr. 4.3, pravá část), pro každý kvadrant zvlášť. Právě toto způsobuje konečnost výsledného souřadného systému. Výsledné mapování kvadrantů je zobrazeno na obrázku 4.4.



Obrázek 4.4: Výsledek obou transformací, vlevo původní kartézský systém, vpravo systém po dvou transformacích. Body jsou zobrazeny v barevně vyznačeném kosočtverci, který má zároveň tvar diamantu – odtud diamond space[7]

[6] Obrázek převzat z: [17]
[7] Obrázek převzat z: [17]

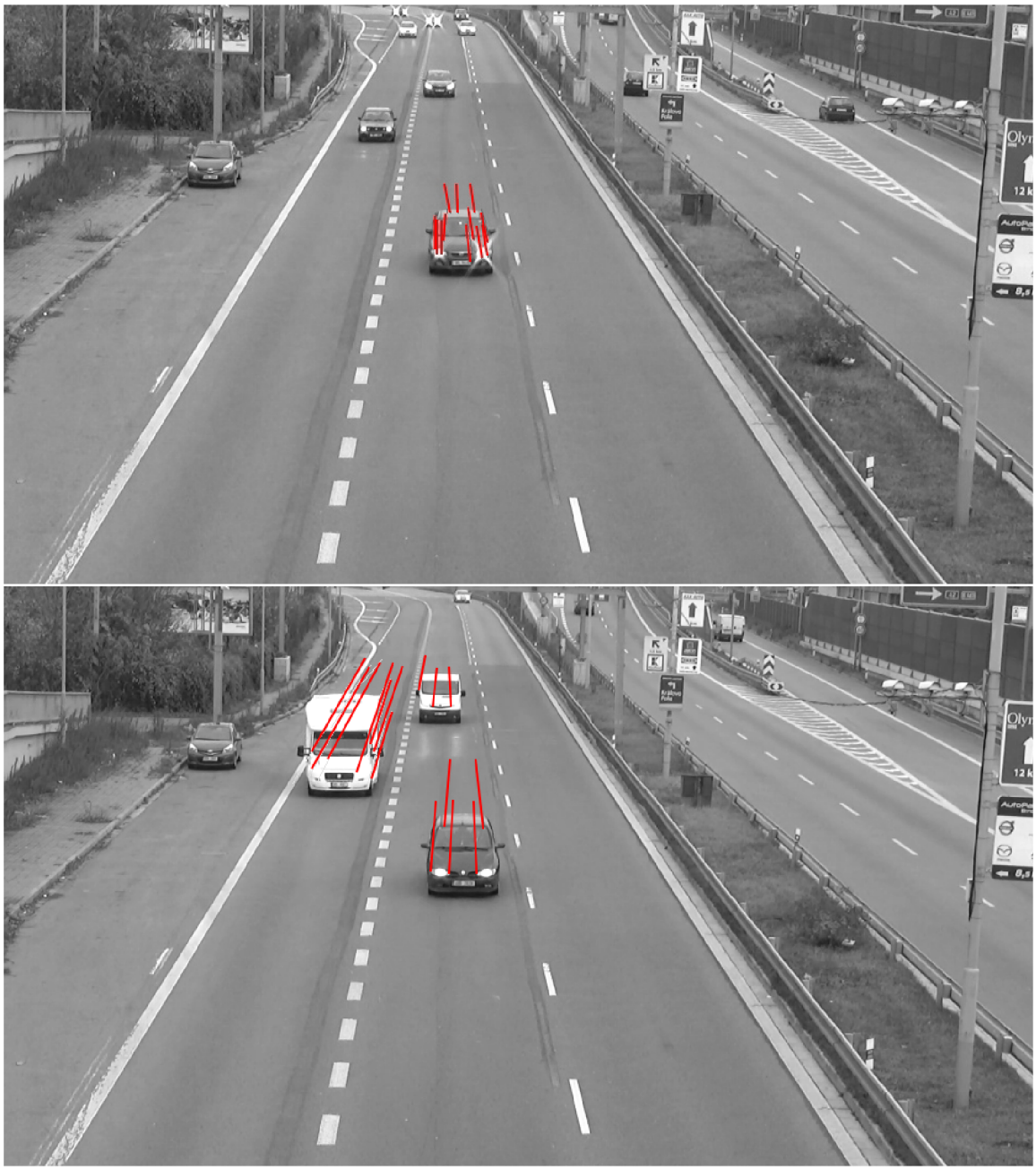
Přímky získané během kalibrace jsou transformovány na polypřímky (pomocí vztahů (10) z [16]). Tím je získáno tolik polypřímek, kolika kvadranty původní přímka prochází. Tyto polypřímky jsou vloženy do DS tak, že hodnoty jejich pixelů jsou inkrementovány. Tím polypřímky hlasují – průsečík nejvíce přímek je definován jako globální maximum z hodnot. Souřadnice maxima v DS jsou transformovány na původní souřadnice pomocí vztahu (2) z [17].

4.5.2 Detekce prvního VP

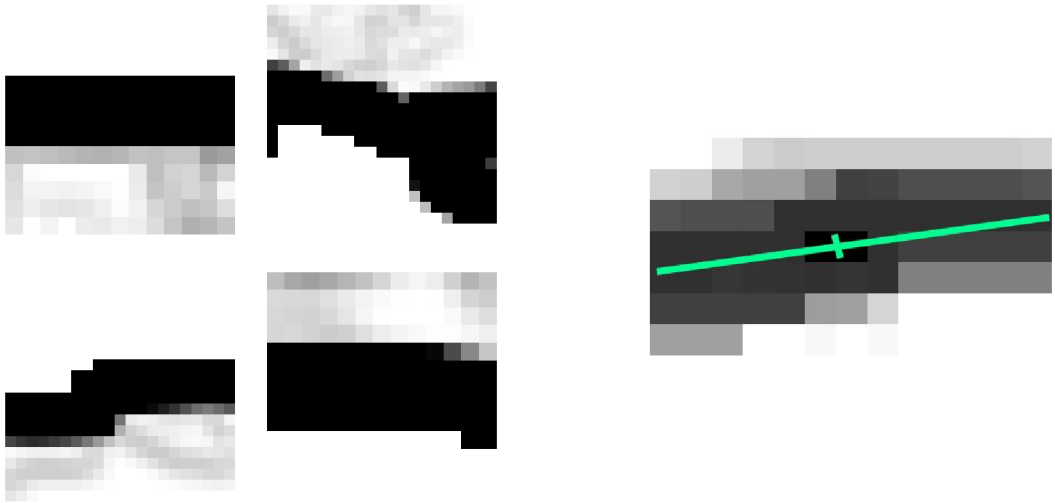
Z obrazu popředí se vyberou příznaky vhodné ke trackování (good features to track [46], obrázek 4.5) a následně se pomocí KLT trackeru sledují (obrázek 4.5). Je třeba zajistit, aby mezi dvěma snímky byla dostatečná mezera a auta tak ujela dostatečnou vzdálenost. Snímky, které by přišly moc brzy, se jednoduše zahodí. Kalibrační část tedy pro snímky v časech t_1 a t_2 získá množinu dvojic $[x_{t_1}, y_{t_1}]$ a $[x_{t_2}, y_{t_2}]$, které tvoří přímky (obrázek 4.8), které se předají *diamond space*. Před předáním je vhodné odstranit zjevně špatná přiřazení – zejména s příliš krátkou nebo příliš dlouhou vzdáleností mezi určujícími body.

Vzhledem k režii operačního systému a ROSu a případnému zahazování snímků s nevhodným časovým razítkem (popsáno výše) mohou být pro stejné video při několika spuštěních dosaženo různých výsledků, dá se nicméně očekávat, že tyto různé výsledky budou velmi blízko.

Obrázek 4.5: Nahoře snímek s vyhledanými trackovatelnými příznaky (zeleně, good features to track), dole odpovídající body na následujícím snímku vyhledané pomocí KLT trackeru (červeně)



Obrázek 4.6: Získané úseky přímek, které hlasují v diamond space. Zde je patrné, že část bodů na vrchní části automobilu na horním obrázku byla trackerem vyhledána špatně, proto přímky směřují špatným směrem.



Obrázek 4.7: Vlevo několik segmentů čar, jednotlivé pixely tvoří magnitudy, středy obrázků tvoří body s vysokou magnitudou v rámci jednoho snímku. Pro každý z těchto segmentů jsou spočítány jeho vlastní hodnoty a vlastní vektory (zeleně na pravém obrázku), po nastřádání dostatečného počtu segmentů je určité procento těch s nejvyššími vlastními hodnotami vloženo do diamond space.

Existuje několik způsobů ověření, zda ukončit hledání. První z nich může být stanovení časového intervalu, po který musí výsledky vracené modulem *diamond space* být dostatečně blízko sebe. Jiný způsob je stanovení počtu přímek, po jehož dosažení se hledání prvního VP zastaví nebo stanovit maximum, kterého musí globální maximum třídy *diamond space* dosáhnout. Já jsem zvolil omezení časem a intervalem, obě tyto hodnoty musí být dostatečně rozumně zvolené, aby vůbec k potvrzení VP došlo, nicméně aby řešení bylo dostatečně přesné.

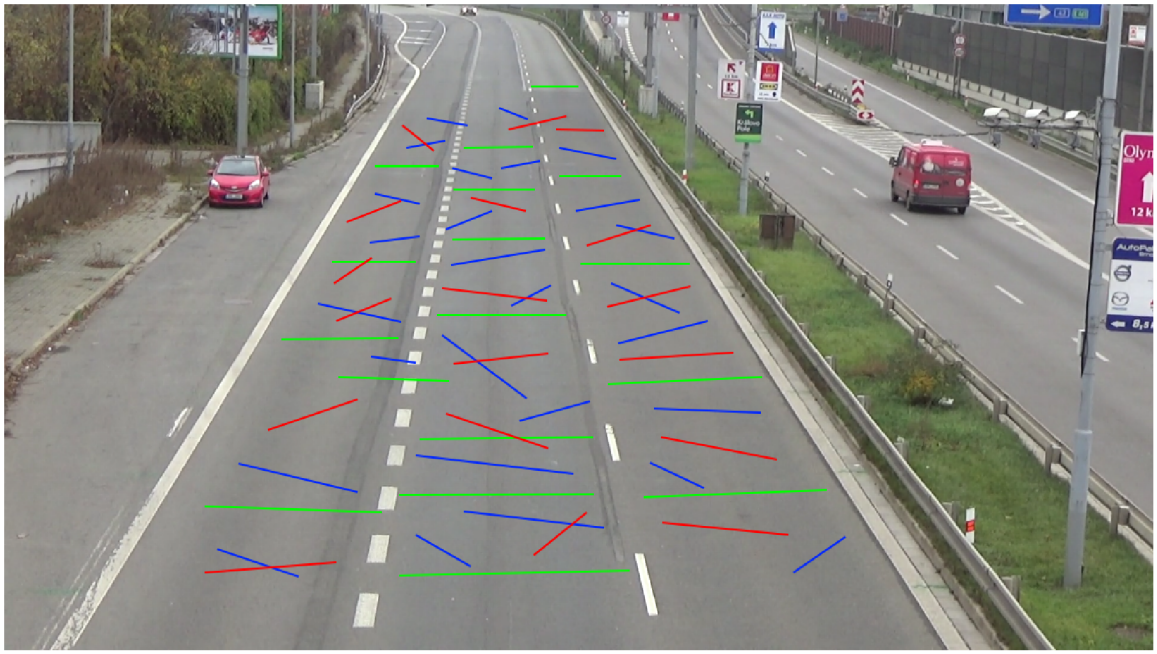
4.5.3 Detekce druhého VP

Tento úkon by měl být započat po skončení hledání prvního VP, neboť první VP omezuje oblast, ve které se druhý VP vyskytuje. Druhý VP je vyhledán na základě hran automobilů, které k němu směřují. K tomu je potřeba tyto hrany vyhledat, opět pouze v oblastech popředí. V této práci používám Sobelův operátor k získání hranového modelu. Díky Sobelovu operátoru jsou získány dva modely – pro hrany přibližně ve směru osy X a ve směru osy Y. Z těchto dvou modelů jsou spočteny gradienty a magnitudy. Hrany s nejvyšší magnitudou jsou použity dále. Tyto hrany je třeba filtrovat, je například zbytečné používat svislé hrany apod. Hrany tedy mohou mít úhel $< -45°; +45° >$.

Jako velmi efektivní optimalizace se ukázalo zapojení okolí bodů do výpočtu směru hrany. Ideální velikost okolí se pohybuje okolo $9 \times 9px$, kdy se příslušný bod nachází přesně uprostřed. S nižšími hodnotami optimalizace ztrácí na přesnosti, s vyššími se pak zvyšuje riziko rušení ostatními hranami atp. Pro každé okolí extrahovaných a přefiltrovaných hran (respektive jejich výchozích bodů) jsou spočítány vlastní vektory a vlastní čísla, které jsou ukládány. Až je nastřádáno dostatečné množství těchto hodnot, určité procento nejlepších podle vlastních čísel je vloženo do diamond space.

Diamond space je navíc maskován v závislosti na poloze prvního VP na základě:

- úhlu horizontu – úhlu, který svírá přímka tvořená prvním a druhým VP s přímkou vodorovnou,
- fokální vzdálenosti, která omezuje vzdálenost mezi prvním a druhým VP. Maximální a minimální fokální vzdálenost se získá z rozlišení snímků a stanoveného maximálního, respektive minimálního, horizontálního zorného úhlu kamery.



Obrázek 4.8: Vlastní vektory okolí bodů s vysokou magnitudou, které hlasují v diamond space. Červeně jsou vektory, jejichž vlastní hodnoty nejsou dostatečně velké, tudíž nebyly vloženy do diamond space. Zeleně a modře jsou vektory, které již v diamond space hlasují. Zelené jsou ty, které přibližně směřují k druhému VP.

Opět je třeba určit způsob zastavení hledání, je použit stejný způsob jako u prvního VP.

4.5.4 Výpočet třetího VP a fokální vzdálenosti

Poloha třetího VP v obrazu se získá na základě rovnice 4.6. Pro získání reálné polohy třetího VP W' je třeba mít detekované předchozí dva VP – body U a V v následujících rovnicích.

Po získání posledního VP a fokální vzdálenosti je kamera plně zkalibrována.

$$f = \sqrt{-(U-P)(V-P)} \tag{4.1}$$

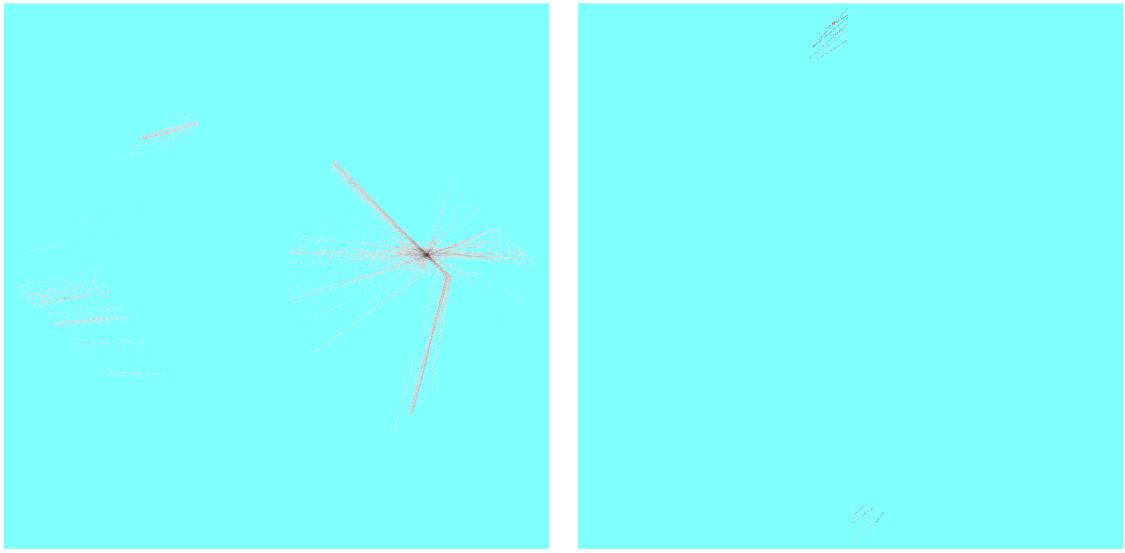
$$U' = [U_x, U_y, f] \tag{4.2}$$

$$V' = [V_x, V_y, f] \tag{4.3}$$

$$P' = [P_x, P_y, 0] \tag{4.4}$$

$$W' = (U' - P') \times (V'P') \tag{4.5}$$

$$W = \left[\frac{W'_x}{W'_z} f + P_x, \frac{W'_y}{W'_z} f + P_y\right] \tag{4.6}$$



Obrázek 4.9: Dimond space s naakumulovanými přímkami a označenými validními VP. Vlevo diamond space pro první VP, vpravo pro druhý. Oba prostory jsou maskované, druhý dle pravidel popsaných výše, v prvním platí omezení, že VP nesmí být pod úrovní obrazu (musí být buď ve snímku nebo nad ním). Pozadí obrázků bylo upraveno pro lepší viditelnost akumulovaných bodů. V tomto případě je druhý VP velmi vzdálený (souřadnice x se pohybuje okolo hodnoty 70000), proto jsou nejvyšší hodnoty v nejhornější části diamond space - blízko nekonečna.

4.6 Detekce

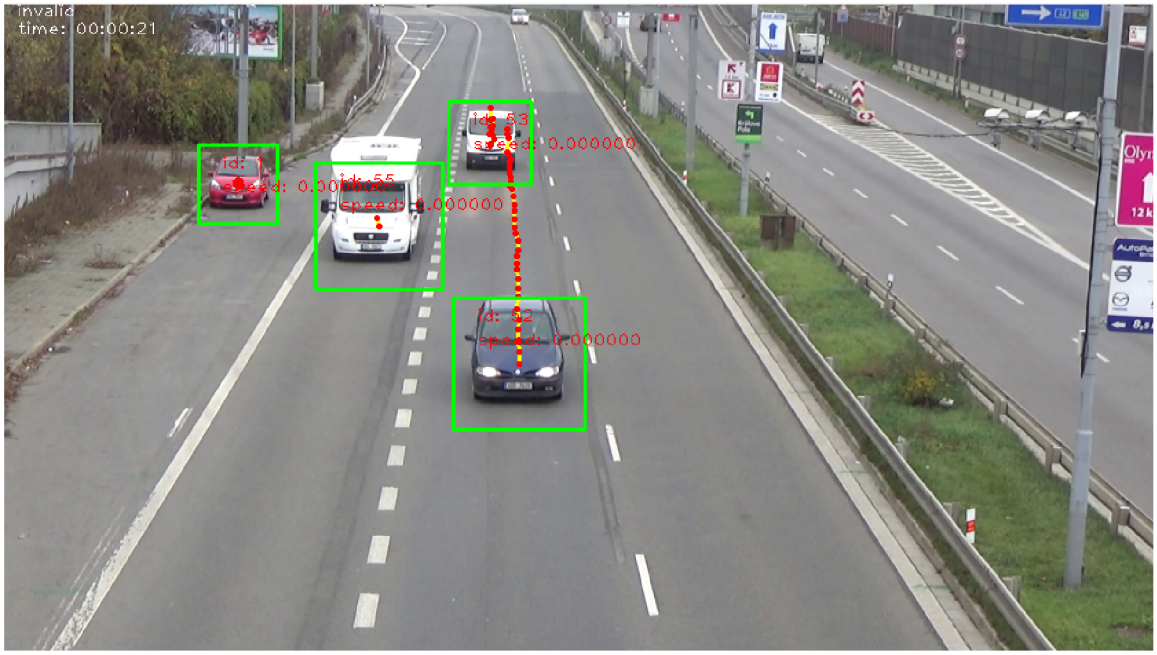
Detekce automobilů je v tomto systému založena na extrakci Haarových příznaků – využívá Viola-Jones detektor, který používá trénovací algoritmus AdaBoost, jehož cílem je sestavení kaskády dílčích klasifikátorů, které pak během rozhodování hlasují, zda-li aktuální okno obsahuje hledaný objekt – v tomto případě automobil. Pro zrychlení činnosti dílčích klasifikátorů detektor používá tzv. integrální obraz. Základním vstupem detektoru je natrénovaná kaskáda pro vyhledávaní objektů. Detektor pak na každém nově příchozím snímku vyhledá automobily a přes topic je pošle trackeru (viz dále). Kromě jejich polohy obsahuje zpráva i výřez automobilu (variabilní velikosti), který může být použit v dalších činnostech.

4.7 Tracking

Zdetekované automobily (resp. středy jejich bounding boxů) jsou předány Kalmanovu filtru, který přijatou zprávu zkopíruje na výstupní topic s přiřazenými ID. V případě, že automobil není detekován, ale stále by se měl nacházet ve snímku, sledovací část nepošle jeho odhadovanou polohu.

Tracker si během své činnosti udržuje informace o aktuálních automobilech ve scéně, s příchodem dalších detekcí porovná odhady dalších poloh všech automobilů s právě příchozími detekcemi a pomocí *Hungarian algoritmu* [28] (sloužící k párování na základě nejnižší ceny) přiřadí tyto příchozí detekce k jednotlivým automobilům. Pokud by cena byla příliš vysoká (nad stanovený práh), tracker bude s danou detekcí nakládat jako s prvním

výskytem nového automobilu. Po tom, co jsou všechny detekce správně přiřazeny, tracker vypočítá pomocí Kalmanova filtru odhady nových poloh, které budou v dalším cyklu opět porovnávány s příchozími detekcemi.



Obrázek 4.10: Detekované automobily na snímku formou bounding boxů (zeleně). Obsah těchto boxů je poslán na topic spolu s údaji o poloze. Červené tečky spojené žlutou čárou značí dřívější výskyty daného vozidla.

4.8 Klasifikace

Po odjetí automobilu ze scény je ukončeno jeho sledování a tím jsou pro tento automobil k dispozici všechny potřebné informace pro klasifikaci (id, seznam výřezů). Počet výřezů aplikace zredukuje, ve výchozím nastavení na deset, a to tak, aby mezi původními časy výřezů byly stejné rozestupy. Navíc platí omezení pro minimální časový rozestup mezi výřezy. Potom jsou snímky z redukovaného seznamu poslány na vstup klasifikátoru, který určí pravděpodobnosti shody ke každé třídě z modelu – po klasifikaci všech snímků ze seznamu je nalezen modus získaných hodnot, která reprezentuje značku a typ vozidla. Pro klasifikaci využíváme Keras, což je vysokoúrovňová knihovna pro práci s neuronovými sítěmi. Jednotlivé třídy modelu reprezentují vybrané automobily, které aplikace dokáže klasifikovat. Předtrénovaný model obsahuje sto šest tříd. Pravděpodobnost úspěšného určení značky a typu vozidla se většinou pohybuje v intervalu 85–95 %.

4.9 Odhad rychlosti

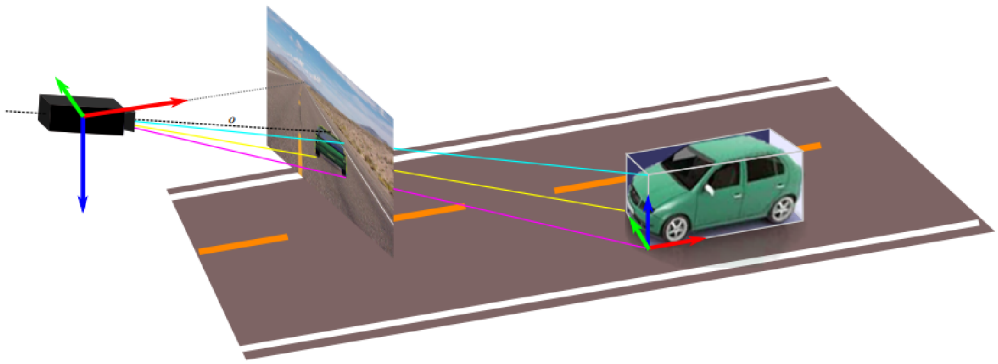
Pro jakékoliv měření rychlosti jsou důležité dvě informace – ujetá vzdálenost a čas ujetí této vzdálenosti. Zatímco čas lze snadno získat z informací o snímcích, se vzdáleností je to těžší. Pro získání ujeté vzdálenosti je třeba nejprve získat polohu automobilu v reálném světě, protože měření pouze ze souřadnic v obrazu není možné kvůli perspektivnímu zkreslení. Je

tedy třeba najít průsečík přímky danou dvěma body – polohou kamery a polohou v obraze – s rovinou, kterou je povrch vozovky. Normála této roviny je zároveň vektor směřující ke třetímu VP. Bohužel je neznámý druhý parametr roviny – její vzdálenost od kamery. Je zřejmé, že kamery bývají umístěny v různých polohách vůči silnici, kterou sledují, a přímé změření tohoto parametru občas nemusí být vůbec možné. Získání vzdálenosti roviny od kamery je popsáno dále. Kvůli tomuto nedostatku proces vrací vzdálenosti v nějakých relativních jednotkách, které později budou díky měřítku převedeny na klasické jednotky délky. Podobu scény ilustruje obrázek 4.11.

Měření rychlosti pracuje ve dvou režimech:

- okamžité – rychlost se počítá z první a poslední detekce, slouží jen pro vizualizační účely a
- průměrné – všechny body se aproximují přímkou případně parabolou, na níž se spočítá ujetá vzdálenost.

Průměrná rychlost se spočítá vždy po vyjetí daného automobilu z regionu obrazu, okamžitou rychlost systém počítá vždy s příchodem další detekce pouze tehdy, pokud si uživatel nechá vizualizovat aktuálně zpracovávaná data.



Obrázek 4.11: Konfigurace scény – kamera v bodě $[p_x, p_y, 0]$, principal point na souřadnicích $[p_x, p_y, f]$, povrch vozovky s normálou shodnou s vektorem ke 3.VP[8]

4.9.1 Vypočítání vzdálenosti

Aplikace se snaží dva body v obraze – X a Y – promítnout na známou rovinu vozovky na body X' a Y'. Toho je dosaženo pomocí vztahu (pro vstupní bod $p = [p_x, p_y]^T$) 4.12, tento vztah předpokládá, že souřadnice jednoho z bodů na přímce leží v počátku, v tomto případě se jedná o kameru, proto je třeba od detekovaných souřadnic odečíst principal point P_p 4.10.

$$P_p = [pp_x, pp_y]^T \tag{4.7}$$

$$n = \frac{W}{\|W\|} \tag{4.8}$$

$$\rho = [n^T, \delta]^T \tag{4.9}$$

[8]Obrázek převzat z: [53]

$$\bar{p} = [p_x - pp_x, p_y - pp_y, f]^T \tag{4.10}$$

$$s = -\frac{\delta}{[\bar{p}^T, 0].\rho} \tag{4.11}$$

$$P = s\bar{p} \tag{4.12}$$

4.9.2 Získání měřítka

Po tom, co aplikace nastřádá dostatečné množství výřezů automobilů určitého typu (v tomto případě Škoda Octavia), systém se pokusí zjistit měřítko scény. Mechanismus zjištění je založen na porovnávání těchto výřezů se známou konvexní obálkou patřící danému typu automobilu. K porovnávání je zapotřebí znát kalibrační údaje, proto systém musí pro zjištění měřítka počkat jednak na dostatečný počet snímků automobilů daného typu a na kalibrační údaje.

Vzhledem k tomu, že detektor nevrací bounding boxy dokonale obklopující automobil, bylo potřeba přidat zvláštní uzel, jenž si udržuje model pozadí a na jeho základě v kombinaci s údaji trackeru vrací přesné bounding boxy, které se již použijí pro získání měřítka scény.

Jako záložní způsob výpočtu měřítka aplikace obsahuje možnost zadat dva body v obraze a vzdálenost mezi odpovídajícími body v reálném světě (např. délka přerušované čáry).

4.10 Implementace

Celý systém je implementován v jazyce C++ za účelem maximalizování výkonu, některé části balíčku *traffic_speed* jsou napsány v jazyce Python. Celý systém je rozdělen do ROS balíčků popsaných na začátku této kapitoly. Pomocné soubory (zejména definice zpráv a služeb a spouštěcí launch soubory) jsou umístěny v hlavním balíčku *traffic*. Dále jsou zde umístěny soubory pro nastavení parametrů spouštění, které jsou ve formátu YAML. Vzhledem k tomu, že celý systém běží na ROSu a využívá jeho vestavěné možnosti meziprocesové komunikace, je snadno rozšiřitelný o další bloky, které mohou dále zpracovávat známá data.

Vstupem systému je zejména adresa video souboru nebo streamu, dále soubor s parametry jednotlivých bloků a také kaskáda pro detekování vozidel. Výstupem pak je JSON soubor obsahující informace o nalezených automobilech – jejich ID, pozice výskytu a celkovou rychlost a navíc kalibrační údaje. Uživatel si pomocí nástrojů jako *rostopic* může nechat zobrazovat průběžně zpracovávané informace, systém navíc umožňuje vizualizovat aktuální data a ukázat, co se s nimi právě děje.

4.10.1 Ovládání aplikace

Aplikaci uživatel spouští jediným launch souborem, který spustí všechny potřebné ROS uzly a zároveň načte do paměti konfiguraci uloženou ve speciálním YAML souboru. Pomocí tohoto konfiguračního souboru je jediná možnost, jak celému systému předat různé parametry. YAML soubor obsahuje v oddělených částech nastavení pro všechny části systému:

- traffic_ffmpeg – URL streamu a název výstupního topicu pro video,
- traffic_calibration – typ kalibrace (automatická, manuální),
- traffic_detection – cestu k natrénované kaskádě

a další (názvy ostatních topiců, služeb, debug parametry apod.).

```
traffic:
    roi:
        # Body oblasti zajmu, kazdy z~techto elementu obsahuje
        # podlelementy x a y
        tl,tr,bl,br:
        # Alternativa k~bodum vyse, cesta k~cernobile masce
        mask:
    # Vystupni JSON soubor, URL vstupniho souboru
    log_file,stream_url:
    # Jmeno topicu, na ktery se zapisuje stream
    stream:
        url: /data/video/video.avi # cesta k~video souboru / streamu
        topic: "/traffic/stream"
        wait_for_detector: true # ffmpeg cast ne/ceka na detektor
    calibration:
        display: true # zobrazit, co se aktualne deje
        # Podelementy vp1 a vp2; minimalni cas, po ktery se poloha VP
        # muze zmenit maximalne o~max_stable_move (px, nize),
        # pro druhy VP se jedna o~pocet vlozeni do diamond space
        stable_time:
        # Podlementy vp1 a vp2; pro vp1 hodnota v~pixelech, pro druhy
        # v~procentech
        max_stable_move:
        # 0 - automaticka kalibrace, 2 - rucni hodnoty
        type:
        # Podelementy x a y; pouzite pro typ 2 vyse
        vp1, vp2, pp:
        # Minimalni pocet naakumulovanych car pro VP1
        min_accumulated_lines: 100
        # Jakmile je nastradan pocet total_candidates,
        # pocet insert_candidates je vlozen do d.space
        vp2_insert_candidates, vp2_total_candidates:
    classification:
        classify: false
    detection:
        # URL natrenovane kaskady
        cascade:
        # Parametry funkce detectMultiScale
        scale_factor, min_neighbors:
    speed:
        measure: true
        # Vzdalenost roviny vozovky od kamery
        plane_dist:
        scale:
            # System se ne/snazi zjistit meritko sceny
            obtain: false
            manual:
                # vzdalenost v~realu mezi body p1 a p2
                distance:
                p1, p2: x, y
```

Kód 4.1: Část konfiguračního souboru systému s ukázkou nejdůležitějších nastavení.

4.10.2 Změna rozlišení videa

Při vývoji se ukázalo, že posílání snímků videa přes topic stojí poměrně hodně času, který je závislý na velikosti obrazu. Snížit tuto dobu se povedlo komprimací snímku (z BGR do JPEG), což sice stojí systém čas pro kódování a dekódování snímků, nicméně i tak přinese až přibližně čtyřnásobné zrychlení v případě FullHD videa.

Další zrychlení přineslo zavedení změny rozlišení videa, pokud je příliš velké. Část *traffic_ffmpeg* o tom informuje nastavením jedné položky zprávy na podíl původního a nového rozlišení. Všechny další části nicméně všechna data publikují v původním souřadnicovém systému, neboť prosté vynásobení podílem původního a nového rozlišení by mohlo vést k nepřesnostem (zejména v případě kalibračních údajů).

4.10.3 Hlavní uzel

Z pohledu implementace je nejzajímavější část systému uzel traffic, který seskupuje a ukládá data z ostatních uzlů. Automobily jsou symbolizovány třídou *Car*, na které si systém udržuje odkazy pomocí *boost::shared_ptr*. Systém si udržuje několik seznamů těchto chytrých ukazatelů (automobily aktuální ve videu, automobily čekající na klasifikaci, vhodné automobily pro získání měřítka). Při destrukci objektu *Car* (nastane při odstranění záznamu ze všech seznamů) dojde k zalogování objektu do zadaného výstupního streamu.

Systém si udržuje frontu také pro měření rychlosti, byť je měření na rozdíl od klasifikace velmi rychlé. Existence této fronty je zapříčiněná tím, že ze začátku systém nezná kalibrační údaje.

Klasifikaci, která trvá určitý čas, a měření rychlosti bylo třeba oddělit z hlavního cyklu systému do vlastních vláken, které periodicky (s frekvencí 10Hz) hlídají klasifikační frontu a případně předá automobil k oklasifikování či změření a pak jej z této fronty odstraní.

4.10.4 Detekční uzel

Vzhledem k tomu, že detekce pomocí kaskády trvá přibližně 55ms (na počítačové sestavě popsané v následující kapitole), může docházet u videí s vyšším FPS k přeskakování snímků. Za účelem minimalizace tohoto negativního jevu jsem použil nástroj ROSu *MultiThreadedSpinner*, který, pokud přijdou přes topic nová data, přičemž zpracovávání těch předchozích ještě nebylo dokončeno, vytvoří nové vlákno, které zpracuje nový snímek. Ve výchozím nastavení je maximální počet těchto případných vláken roven počtu vláken procesoru. V případě, že je již vytvořen maximální povolený počet vláken, jsou další příchozí zprávy zahazovány.

Tento uzel zároveň měří průměrnou dobu detekce t_d za účelem přizpůsobení frekvence publikování snímků. S počtem vláken procesoru c je pak požadovaná frekvence f ffmpeg části:

$$f = \frac{1}{t_d/c * 2} \tag{4.13}$$

Jmenovatel zlomku je násoben konstantou 2, aby pouze detektor nevyužíval všechen výpočetní výkon CPU.

Kapitola 5

Testování a experimenty

V této kapitole je popsáno vyhodnocení přesnosti jak jednotlivých částí, tak celého systému ve formě přesnosti získaných rychlostí. Dále je zde popsána rychlost zpracovávání videí systémem.

Systém byl otestován na setu videí *BrnoCompSpeed*[54], který je popsán dále.

Experimenty byly prováděny na počítači s CPU Intel Core i7 890 a 4GB RAM s nainstalovaným OS Linux Mint. Co se týče rychlosti zpracovávání, aplikace zvládne zpracovat až sto snímků za vteřinu. Tato hodnota je závislá pouze na rychlosti dekódování snímků, detektor automobilů analyzuje pouze například každý pátý snímek. Se zapnutou možností čekání na detektor se pak rychlost zpracovávání pohybuje mezi deseti až patnácti snímky za vteřinu. Bez zapnutého dekódování ve více jádrech by tato hodnota byla poloviční až třetinová.

5.1 Dataset

Sada videí se skládá ze šesti sezení po třech videích. Jednotlivá sezení byla pořízena na různých místech u čtyřproudových komunikací. Každé sezení se skládá ze třech videí – pohled z levé strany, zprostřed a z pravé strany.Videa jsou dlouhá přibližně hodinu, mají rozlišení FullHD a počet snímků za vteřinu je padesát. Použité kamery jsou Panasonic HC-X920, Panasonic HDC-SD90 a Sony Handycam HDR-PJ410. Celkový počet zachycených automobilů je 20 865.

Pro zjištění referenčních rychlostí byly použity dvě sestavy složené z PC, LIDARu a GPS přijímače, umístěné podél cesty ve většině případů 28 metrů od sebe. LIDARy, nastavené jako světelné závory, byly umístěny přesně kolmo vůči směru dopravy a zjišťovaly časy začátku a konce protnutí paprsku. Tyto časy byly synchronizovány pomocí GPS přijímačů.

Tento dataset se tedy skládá z osmnácti videí natočených v šesti lokacích, autoři poskytují referenční informace o vybraných automobilech (jejich rychlost) pro všechna videa, stejně jako kalibrační údaje a délky určitých příznaků ve videích.

5.2 Detekce

Pro vyhodnocení úspěšnosti detekce byly použity manuálně anotované části videí dlouhé deset minut – dohromady tedy 180 minut. V tomto případě se zaměřujeme pouze na validní nalezení automobilu v závislosti na parametrech detektoru. Posouzení středu objektu a jeho

Obrázek 5.1: Screenshoty ze šesti sessions, každé na jednom řádku. V levém sloupci jsou videa točená z levé strany, podobně v prostředním a pravém.

velikosti vrácené detektorem je závislé na dodané natrénované kaskádě, byť i tyto informace mají pro správné určení rychlosti a zejména měřítka vliv.

Video	Počet aut	FN	TP	Úspěšnost
session3_right	62	2	60	96.77%
session5_center	420	16	404	96.19%
session4_center	234	11	223	95.29%
session5_left	392	40	352	89.79%
session2_center	300	36	264	88.00%
session1_right	174	22	152	87.35%
Celkem	3598	280	3318	92.21%

Tabulka 5.1: Tři videa s nejlepšími a tři videa s nejhoršími výsledky detektoru vozidel. Celkové hodnoty platí pro všechna videa. Tabulka obsahuje srovnání celkového počtu automobilů s počtem neúspěšně detekovaných (ignorovaných). Automobil je zahrnut do skupiny false negative, pokud není během jeho výskytu vůbec detekován, nebo pokud je jeho získaná dráha ve vztahu k drahám správně detekovaných automobilů ve stejném pruhu přibližně třetinová a menší.

Těchto hodnot dosáhl detektor s nastavením *scale factor* = 1.1 a *min neighbors* = 4. Právě s těmito hodnotami je detektor nejúspěšnější. Je zřejmé, že detektor si nedokáže poradit se situacemi, kdy je malá část automobilu schovaná za např. sloupem osvětlení. Toto omezení, týkající se obecně metod založených na extrakci příznaků, je zohledněno v tabulce výše - takto nedetekované automobily nejsou počítány jako false negative. Během testování se také ukázalo, že detektor zvládne úspěšně detekovat:

- drtivou většinu osobních automobilů jak zepředu, tak zezadu,
- drtivou většinu dodávek a menších nákladních automobilů zepředu, o něco hůře pak zezadu a
- většinu velkých nákladních automobilů zepředu, zezadu si neporadí s neobvyklejšími automobily (např. popelářské auto).

5.2.1 Tracking

Vyhodnocení trackingu je také založeno na vizuální verifikaci historie poloh automobilu, kterou aplikace vykresluje. V případě sledování automobilů se velmi liší optimální hodnoty parametrů v závislosti zejména na úhlu pohledu kamery s vozovkou. Čím je tento úhel nižší (kamera se dívá do dáli), tím se zvyšují hodnoty pro maximální vzdálenost mezi detekcemi a s tím související počet snímků, kdy může být auto nedetekováno. Při nastavení optimálních hodnot dosahuje trackování vysoké úspěšnosti, během testování došlo párkrát k přechodu identit mezi automobily a to pouze v případě druhého nebo třetího výskytu, dál už tracker vrací konzistentní výsledky.

Dle testovací sady, která srovnává známé automobily s automobily detekovanými, se dosažená úspěšnost detekce a trackingu pohybuje v součtu pro všechna videa okolo hodnoty 83% – tolik procent anotovaných automobilů byl skript schopen spojit se získanými tracky.

5.3 Klasifikace

Zjištění procentuální úspěšnosti klasifikace probíhalo podobně jako detekce. V prvních deseti minutách byly manuálně vizuálně kontrolovány klasifikované třídy automobilů. Ne u všech automobilů jsme byli schopni se stoprocentní jistotou určit přesný typ, tyto případy jsou pro vyhodnocení klasifikátoru ignorovány. Celková úspěšnost klasifikace je 63.72 %.

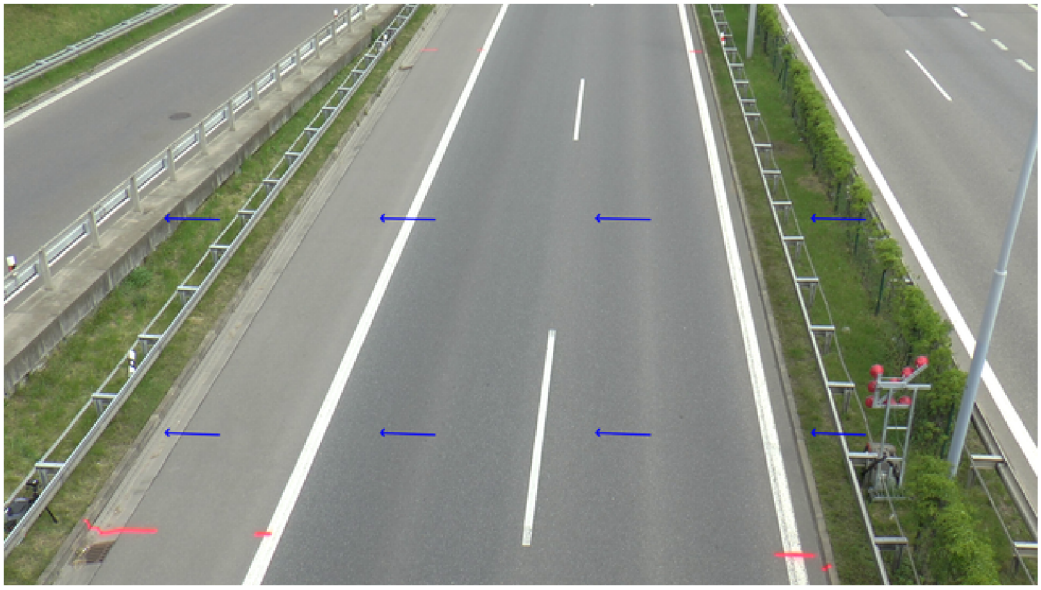
5.4 Kalibrace

Přesné změření rychlosti je nejvíce závislé na úspěšné kalibraci kamery. Pro vyhodnocení přesnosti kalibrace jsem použil skripty dodané k testovacímu datasetu[1]. Vyhodnocení úspěšnosti kalibrace je založeno na promítnutí vybraných bodů v obraze, u nichž jsou známy skutečné 3D souřadnice, pomocí kalibračních údajů (prvního a druhého úběžníku a principal pointu). Pro vyhodnocení se porovnávají poměry mezi všemi dvojicemi skutečných a promítnutých bodů. To je znázorněno ve vztazích 5.1 a 5.2 pro výpočet absolutní a relativní chyby, kde r_{gt} je poměr mezi dvěma dvojicemi skutečných bodů (ground truth) a r_m je poměr mezi odpovídající dvojicí promítnutých vzdáleností.

$$err = |r_{gt} - r_m| \tag{5.1}$$

$$err = |r_{gt} - r_m|/r_{gt} * 100\% \tag{5.2}$$

V tabulce 5.2 jsou prezentovány průměry a mediány jednotlivých videí pro tento systém.



Obrázek 5.2: Obrázek z videa *session1_center* se znázorněnými čarami, které míří ke druhému úběžníku.

Z výsledků a obrázků videí je patrné, že kalibrace je v součtu nejméně přesná pro středová videa, což způsobují dva důvody - jednak je sledovaný úsek cesty nejkratší a jednak se orientace v reálu vodorovných hran na jedoucích autech při pohledu zepředu mění mnohem méně než při pohledu zboku, což způsobuje vyšší náchylnost na nepřesnosti. Po manuálním

[1] https://github.com/JakubSochor/BrnoCompSpeed/

Video	Relativní průměr [%]	Relativní medián [%]
session1_left	1.46	1.45
session1_center	8.22	7.52
session1_right	2.58	2.58
session2_left	6.11	4.98
session2_center	4.19	4.84
session2_right	1.00	1.12
session3_left	14.64	8.81
session3_center	11.19	7.09
session3_right	15.11	3.95
session4_left	16.25	5.35
session4_center	27.19	8.51
session4_right	8.30	0.81
session5_left	7.17	1.47
session5_center	13.52	6.69
session5_right	9.99	4.47
session6_left	16.23	7.83
session6_center	18.37	3.82
session6_right	21.55	3.93
Celkové hodnoty	14.02	6.13

Tabulka 5.2: Relativní průměry a mediány odchylek kalibračních údajů pro videa z datasetu. Z vzájemných poloh hodnot lze vyvodit, že hustota distribuce odchylek je vyšší pro nízké odchylky. Naopak větší odchylky mají spíše charakter odlehlých hodnot. Toto je pravděpodobně způsobeno nepřesně detekovanou polohou druhého úběžníku (viz následující kapitola.)

srovnání získaných údajů s optimální kalibrací, kterou poskytují autoři datasetu, se ukázalo, že první úběžníky jsou detekovány poměrně přesně. Rozložení hodnot popsané v tabulce 5.2 je způsobeno nepřesným detekováním druhého VP, přičemž alespoň směr k němu u většiny videí souhlasí (patrné na obrázku 5.2), vyšší odchylky tedy způsobí nepřesné určení vzdálenosti druhého VP od středového bodu obrazu. Nepřesné určení druhého VP nicméně přináší chybně vypočítanou fokální vzdálenost, což je chyba, která se bude projevovat v dalších fázích. Tato chyba by šla eliminovat nastavením úhlu horizontu na velmi nízkou hodnotu, na druhou stranu systém musí být benevolentní vůči natočení kamery v ose souhlasné se směrem jízdy automobilů. Během testování byl maximální úhel horizontu nastaven na hodnotu $\frac{\pi}{48} = 3.75°$.

Tyto nepřesnosti má pravděpodobně na svědomí snížení rozlišení snímků, nicméně bez tohoto rozšíření by systém byl schopen zpracovat přibližně pouze pět FullHD snímků za vteřinu, což je nevyhovující. Snížení rozlišení snímku zároveň snižuje detailnost vyobrazení hran, které se používají k nalezení druhého VP.

Vliv chyby kalibrace kamery na měření rychlosti je popsán v dalších podkapitolách.

5.5 Měření vzdáleností

Vyhodnocení měření vzdáleností již využívá získané měřítko a skládá se ze dvou částí. Měření vzdáleností, které směřují k prvnímu VP a ostatní. Pro měření rychlosti je nejdůležitější přesné změření vzdáleností ve směru k prvnímu VP – tímto směrem jezdí automobily. Vyhodnocení probíhá porovnáváním známých vzdáleností na povrchu vozovky s odpovídajícími vzdálenostmi, které jsou získány promítnutím koncových bodů pomocí kalibračních údajů a vynásobením měřítkem. Testovací skripty navíc dokážou vypočítat měřítko takové, při jehož použití je odchylka vzdálenosti minimální, a tím rozlišit, do jaké míry jsou nepřesnosti způsobené špatnou kalibrací nebo špatným měřítkem. Jelikož systém nedokáže automaticky přesně určit měřítko scény (resp. získané měřítko je velmi nepřesné), pro vyhodnocení používám testovacími skripty vypočítané optimální měřítko. Výsledky v tabulce 5.3 jsou tedy nejlepší možné, jejich odchylky jsou způsobené pouze chybou kalibrace.

Při srovnání s chybami kalibrace vyplývá, že čím je odchylka kalibrace vyšší, tím se kromě celkově horších výsledků měření vzdáleností zvyšuje rozdíl mezi dvěma druhy měření – to je opět způsobeno nepřesně určenými druhými úběžníky. Z většiny vybočuje video *session1_left*, u něhož je přesnost měření všech vzdáleností dvojnásobně lepší než u vzdáleností směřujících pouze k prvnímu VP. Důvodem této výjimky je nepřesné určení prvního úběžníku.

	vše		směrem k prvnímu VP	
Video	průměr [m]	medián [m]	průměr [m]	medián [m]
session1_left	0.21	0.18	0.40	0.40
session1_center	0.34	0.09	0.09	0.09
session1_right	0.43	0.44	0.64	0.64
session2_left	0.34	0.09	0.12	0.09
session2_center	0.39	0.15	0.36	0.07
session2_right	0.06	0.07	0.05	0.03
session3_left	0.96	0.33	0.31	0.32
session3_center	0.61	0.22	0.21	0.17
session3_right	1.00	0.16	0.14	0.12
session4_left	1.11	0.22	0.20	0.18
session4_center	2.86	0.30	0.30	0.26
session4_right	0.45	0.03	0.03	0.03
session5_left	0.68	0.22	0.71	0.22
session5_center	1.00	0.26	0.35	0.15
session5_right	0.92	0.28	0.98	0.28
session6_left	1.35	0.24	0.17	0.08
session6_center	1.89	0.19	0.22	0.11
session6_right	2.46	0.18	0.10	0.08
Celkové hodnoty	1.03	0.19	0.29	0.16

Tabulka 5.3: Průměrné a mediánové odchylky měření všech vzdáleností a vzdáleností směřujících pouze k prvnímu VP. Téměř ve všech případech jsou vzdálenosti směrem k prvnímu VP přesnější, neboť právě první úběžník je ve většině případech určen přesně.

5.6 Měření rychlosti

Úspěšnost měření rychlosti závisí na přesnosti dvou údajů – časových záznamů u jednotlivých výskytů automobilů a ujeté vzdálenosti. Jelikož jsou u snímků informace o pořadovém čísle a jejich čase, odchylka změřené rychlosti od správné je způsobena pouze chybným změřením ujeté trasy. Vliv na správně změřenou délku trasy automobilu mohou mít dva faktory – kromě přesné kalibrace je to také úspěšnost trackování (zejména jestli nedochází k přeskakování identit mezi automobily).

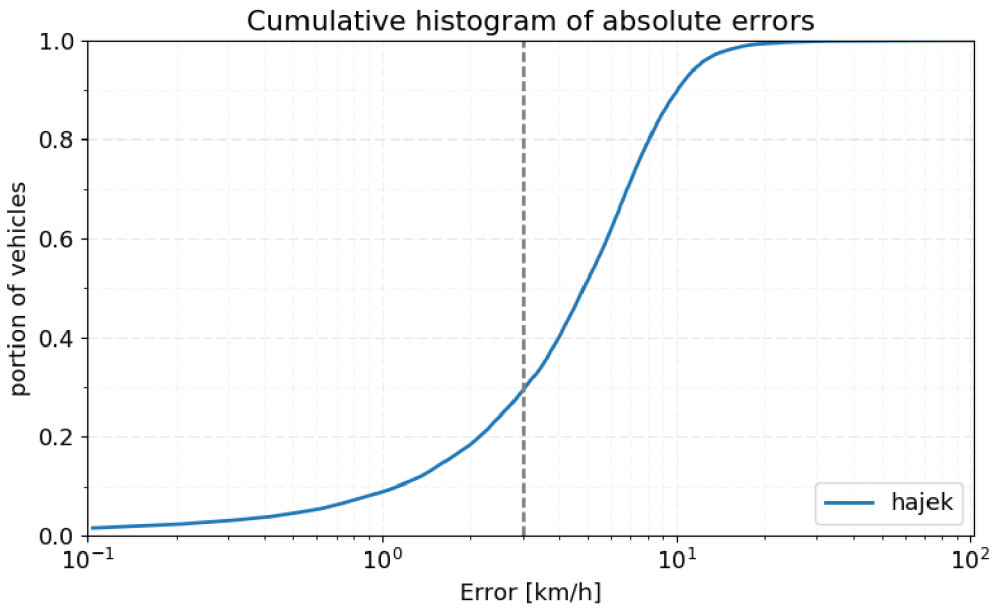
Video	Průměr odchylky [km/h]	Medián odchylky [km/h]
session1_left	5.70	4.83
session1_center	2.78	1.94
session1_right	5.85	4.94
session2_left	6.96	6.52
session2_center	10.66	9.91
session2_right	6.96	6.58
session3_left	3.21	2.03
session3_center	3.46	2.92
session3_right	6.29	6.16
session4_left	10.37	9.63
session4_center	10.02	9.18
session4_right	8.76	7.99
session5_left	3.10	2.56
session5_center	10.94	10.22
session5_right	13.52	12.09
session6_left	11.90	11.62
session6_center	8.13	7.52
session6_right	4.71	4.35
Celkové hodnoty	8.21	7.54
Celkové hodnoty*	6.04	5.31

Tabulka 5.4: Medián a průměr odchylek u jednotlivých videí a celkové hodnoty. Na posledním řádku jsou vyobrazeny celkové hodnoty s vynecháním třech nejhorších videí.

Mediány a průměry se při vynechání nejhorších videí pohybují okolo hodnot, které sice překračují zákonem stanovené maximální odchylku měření pro policejní měření rychlosti[2], nicméně pro účely získávání statistických údajů lze považovat přesnost měření za dostatečnou.

Ve většině případů chyba rychlosti reflektuje chybu měření vzdáleností k prvnímu VP, nicméně například kalibrace u videa *session2_left* se pohybuje mezi nejlepší, podobně měření vzdáleností, naopak měření rychlosti v tomto případě dosahuje podprůměrných výsledků. Lze usuzovat, že na vině je nedokonalá funkčnost trackeru – přeskakování identit.

[2] ±3km/h při naměřené rychlosti do 100 km/h a ±3 % nad 100 km/h



Obrázek 5.3: Distribuce rozložení chyby. Přibližně u jedné třetiny automobilů se chyba pohybuje pod 3km/h, u více než poloviny automobilů je pak rychlost určena s chybou do 5km/h.

Kapitola 6

Závěr

V této práci je popsán zejména návrh systému pro vypočítání rychlosti automobilu z videa a jeho vyhodnocení. Systém běží jako aplikace na vrstvě Robotického operačního systému (ROS), který je spouštěn v rámci docker kontejneru. Výsledkem celé práce tedy můžou být jednak jednotlivé ROS uzly (otevření videa, kalibrace kamery, ...) a jednak docker kontejner s předinstalovanými vytvořenými bloky.

Pro otevření videa (příp. streamu) aplikace používá knihovnu FFmpeg, kalibrační údaje jsou ve formě vanishing points. Dekódované snímky vstupují do kalibračního modulu a do detekčního modulu. Pro detekování vozidel je použit předtrénovaný Viola-Jones detektor, výsledky vrací ve formě bounding boxů. Automobily jsou trackovány pomocí Kalmanova filtru a jejich pozice jsou předávány modulu pro vypočítání rychlosti. Po odjetí automobilu z obrazu je klasifikován. Pokud je daný automobil rozpoznán jako Škoda Octavia, je předán modulu pro získání měřítka scény.

Aplikace využívá služeb knihovny OpenCV a je implementována v jazyce C++, dílčí skripty pak v Pythonu. Díky možnostem ROSu může být aplikace snadno rozšiřitelná o další bloky, které mohou například zaznamenávat různé statistické údaje apod. Hlavní uzel (*traffic*) lze snadno rozšířit o další výstupní topicy, například v případě, že by byl systém použit jako "oči"inteligentní křižovatky, může publikovat okamžitou rychlost a polohu přijíždějících automobilů za účelem maximální efektivity.

Z výsledků popsaných v předcházející kapitole vyplývá, že systém zvládne u čtvrtiny videí měřit rychlost s přesností pod 3km/h, což je zákonem definovaná hranice přesnosti policejních radarů. U poloviny videí je průměrná odchylka menší než 7km/h, chyba u většiny videí se pak pohybuje do 10km/h, medián celkové chyby systému se pohybuje okolo 7km/h, s vynecháním třech nejhorších videí pak klesne na 5km/h, což lze považovat za dostatečně přesné pro například "oči"chytré křižovatky nebo sběr statistických údajů.

Tento text nejprve popisuje existující systému sloužící stejnému účelu a existující metody pro dílčí úkony (kalibrace kamery, detekce a tracking automobilů atd.), dále je zde detailně rozebrán jak návrh celého systému jako celku, tak jednotlivé bloky a komunikace mezi nimi. Nakonec se text věnuje vyhodnocení přesnosti systému.

Literatura

[1] WWW stránky: About ROS. online, [cit. 5.3.2015].
URL http://www.ros.org/about-ros/

[2] WWW stránky: Messages. online, [cit. 6.3.2015].
URL http://wiki.ros.org/Messages

[3] WWW stránky: Nodes. online, [cit. 5.3.2015].
URL http://wiki.ros.org/Nodes/

[4] WWW stránky: Topics. online, [cit. 5.3.2015].
URL http://wiki.ros.org/Topics

[5] Alonso, J. D.; Vidal, E. R.; Rotter, A.; aj.: Lane-Change Decision Aid System Based on Motion-Driven Vehicle Tracking. *IEEE Transactions on Vehicular Technology*, ročník 57, č. 5, Sept 2008: s. 2736–2746, ISSN 0018-9545, doi:10.1109/TVT.2008.917220.

[6] Aslani, S.; Mahdavi-Nasab, H.: Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical, Electronics, Communication, Energy Science and Engineering*, ročník 7, č. 9, 2013: s. 789–793.

[7] Ballard, D. H.: Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, ročník 13, č. 2, 1981: s. 111–122.

[8] Bardet, F.; Chateau, T.: MCMC particle filter for real-time visual tracking of vehicles. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, IEEE, 2008, s. 539–544.

[9] Bouguet, J.-Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, ročník 5, č. 1-10, 2001: str. 4.

[10] Caraffi, C.; Vojíř, T.; Trefný, J.; aj.: A system for real-time detection and tracking of vehicles from a single car-mounted camera. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, Sept 2012, ISSN 2153-0009, s. 975–982, doi:10.1109/ITSC.2012.6338748.

[11] Cathey, F.; Dailey, D.: A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, IEEE, 2005, s. 777–782.

[12] Chen, Y.-L.; Wu, B.-F.; Fan, C.-J.: Real-time vision-based multiple vehicle detection and tracking for nighttime traffic surveillance. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, IEEE, 2009, s. 3352–3358.

[13] Cui, J.; Liu, F.; Li, Z.; aj.: Vehicle localisation using a single camera. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, IEEE, 2010, s. 871–876.

[14] Deng, T.; Li, B.: A detection method of traffic parameters based on EPI. *Procedia Engineering*, ročník 29, 2012: s. 3054–3059.

[15] Do, V.; Woo, D.: Multi-Resolution Estimation of Optical Flow on Vehicle Tracking under Unpredictable Environments. *Advanced Science and Technology Letters*, ročník 129, 2016: s. 32–36.

[16] Dubská, M.; Herout, A.: Real Projective Plane Mapping for Detection of Orthogonal Vanishing Points. In *Proceedings of BMVC 2013*, The British Machine Vision Association and Society for Pattern Recognition, 2013, s. 1–10.

[17] Dubska, M.; Herout, A.; Juranek, R.; aj.: Fully Automatic Roadside Camera Calibration for Traffic Surveillance. *Intelligent Transportation Systems, IEEE Transactions on*, ročník 16, č. 3, June 2015: s. 1162–1171, ISSN 1524-9050, doi:10.1109/TITS.2014.2352854.

[18] Filipiak, P.; Golenko, B.; Dolega, C.: NSGA-II Based Auto-Calibration of Automatic Number Plate Recognition Camera for Vehicle Speed Measurement. In *European Conference on the Applications of Evolutionary Computation*, Springer, 2016, s. 803–818.

[19] Gao, T.; Liu, Z.-g.; Gao, W.-c.; aj.: Moving vehicle tracking based on SIFT active particle choosing. In *International Conference on Neural Information Processing*, Springer, 2008, s. 695–702.

[20] Grammatikopoulos, L.; Karras, G.; Petsa, E.: Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, 2005, s. 332–338.

[21] Hilario, C.; Collado, J.; Armingol, J. M.; aj.: Pyramidal image analysis for vehicle detection. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, IEEE, 2005, s. 88–93.

[22] Ho, W. T.; Lim, H. W.; Tay, Y. H.: Two-stage license plate detection using gentle Adaboost and SIFT-SVM. In *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, IEEE, 2009, s. 109–114.

[23] Hoffmann, C.: Fusing multiple 2D visual features for vehicle detection. In *Intelligent Vehicles Symposium, 2006 IEEE*, IEEE, 2006, s. 406–411.

[24] Jazayeri, A.; Cai, H.; Zheng, J. Y.; aj.: Vehicle Detection and Tracking in Car Video Based on Motion Model. *IEEE Transactions on Intelligent Transportation Systems*, ročník 12, č. 2, June 2011: s. 583–595, ISSN 1524-9050, doi:10.1109/TITS.2011.2113340.

[25] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, ročník 34, č. 7, 2012: s. 1409–1422.

[26] Kanhere, N. K.; Pundlik, S. J.; Birchfield, S. T.: Vehicle segmentation and tracking from a low-angle off-axis camera. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, ročník 2, IEEE, 2005, s. 1152–1157.

[27] Kazemi, F. M.; Samadi, S.; Poorreza, H. R.; aj.: Vehicle recognition using curvelet transform and SVM. In *Information Technology, 2007. ITNG'07. Fourth International Conference on*, IEEE, 2007, s. 516–521.

[28] Kuhn, H. W.: The Hungarian method for the assignment problem. *Naval research logistics quarterly*, ročník 2, č. 1-2, 1955: s. 83–97.

[29] Lin, B.-F.; Chan, Y.-M.; Fu, L.-C.; aj.: Integrating appearance and edge features for sedan vehicle detection in the blind-spot area. *IEEE Transactions on Intelligent Transportation Systems*, ročník 13, č. 2, 2012: s. 737–747.

[30] Liu, W.; Wen, X.; Duan, B.; aj.: Rear vehicle detection and tracking for lane change assist. In *Intelligent Vehicles Symposium, 2007 IEEE*, IEEE, 2007, s. 252–257.

[31] Lou, J.; Yang, H.; Hu, W. M.; aj.: Visual vehicle tracking using an improved EKF. In *Proc. Asian Conf. Computer Vision*, 2002, s. 296–301.

[32] Lucas, B. D.; Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, s. 674–679.
URL http://dl.acm.org/citation.cfm?id=1623264.1623280

[33] Lucas, B. D.; Kanade, T.; aj.: An iterative image registration technique with an application to stereo vision. 1981.

[34] Luvizon, D. C.; Nassu, B. T.; Minetto, R.: A Video-Based System for Vehicle Speed Measurement in Urban Roadways. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[35] Ma, B.; Ban, X.; Wang, Y.; aj.: Recognition of blacklisted vehicle based on SIFT feature. In *Cloud Computing and Intelligence Systems (CCIS), 2016 4th International Conference on*, IEEE, 2016, s. 456–460.

[36] Mauthner, T.; Donoser, M.; Bischof, H.: Robust tracking of spatial related components. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, IEEE, 2008, s. 1–4.

[37] Messelodi, S.; Modena, C. M.; Zanin, M.: A computer vision system for the detection and classification of vehicles at urban road intersections. *Pattern analysis and applications*, ročník 8, č. 1-2, 2005: s. 17–31.

[38] Morris, B. T.; Trivedi, M. M.: Learning, modeling, and classification of vehicle track patterns from live video. *IEEE Transactions on Intelligent Transportation Systems*, ročník 9, č. 3, 2008: s. 425–437.

[39] Nummiaro, K.; Koller-Meier, E.; Van Gool, L.: An adaptive color-based particle filter. *Image and vision computing*, ročník 21, č. 1, 2003: s. 99–110.

[40] Ponsa, D.; López, A.; Lumbreras, F.; aj.: 3D vehicle sensor based on monocular vision. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, IEEE, 2005, s. 1096–1101.

[41] Psyllos, A. P.; Anagnostopoulos, C.-N. E.; Kayafas, E.: Vehicle logo recognition using a sift-based enhanced matching scheme. *IEEE Transactions on Intelligent Transportation Systems*, ročník 11, č. 2, 2010: s. 322–328.

[42] Salti, S.; Cavallaro, A.; Di Stefano, L.: Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Transactions on Image Processing*, ročník 21, č. 10, 2012: s. 4334–4348.

[43] Saunier, N.; Sayed, T.: A feature-based tracking algorithm for vehicles in intersections. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, IEEE, 2006, s. 59–59.

[44] Schoepflin, T. N.; Dailey, D. J.: Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, ročník 4, č. 2, 2003: s. 90–98.

[45] Schoepflin, T. N.; Dailey, D. J.: Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, ročník 4, č. 2, June 2003: s. 90–98, ISSN 1524-9050, doi:10.1109/TITS.2003.821213.

[46] Shi, J.; aj.: Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, IEEE, 1994, s. 593–600.

[47] Sina, I.; Wibisono, A.; Nurhadiyatna, A.; aj.: Vehicle counting and speed measurement using headlight detection. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, IEEE, 2013, s. 149–154.

[48] Sivaraman, S.; Trivedi, M. M.: Active learning based robust monocular vehicle detection for on-road safety systems. In *Intelligent Vehicles Symposium, 2009 IEEE*, IEEE, 2009, s. 399–404.

[49] Sivaraman, S.; Trivedi, M. M.: A General Active-Learning Framework for On-Road Vehicle Recognition and Tracking. *IEEE Transactions on Intelligent Transportation Systems*, ročník 11, č. 2, June 2010: s. 267–276, ISSN 1524-9050, doi:10.1109/TITS.2010.2040177.

[50] Sivaraman, S.; Trivedi, M. M.: Real-time vehicle detection using parts at intersections. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, IEEE, 2012, s. 1519–1524.

[51] Sivaraman, S.; Trivedi, M. M.: Active learning for on-road vehicle detection: A comparative study. *Machine vision and applications*, 2014: s. 1–13.

[52] Sochor, J.: Fully Automated Real-Time Vehicles Detection and Tracking with Lanes Analysis. In *Proceedings of The 18th Central European Seminar on Computer Graphics*, Technical University Wien, 2014, ISBN 978-3-9502533-3-7.

[53] Sochor, J.: *Traffic Analysis from Video*. Diplomová práce, Brno University of Technology, Faculty of Information Technology, 2014.

[54] Sochor, J.; Juránek, R.; Špaňhel, J.; aj.: BrnoCompSpeed: Review of Traffic Camera Calibration and Comprehensive Dataset for Monocular Speed Measurement. 2017, arXiv:1702.06441.

[55] Sun, Z.; Bebis, G.; Miller, R.: Monocular precrash vehicle detection: Features and classifiers. *IEEE transactions on image processing*, ročník 15, č. 7, 2006: s. 2019–2034.

[56] Sundoro, H. S.; Harjoko, A.: VEHICLE COUNTING AND VEHICLE SPEED MEASUREMENT BASED ON VIDEO PROCESSING. *Journal of Theoretical and Applied Information Technology*, ročník 84, č. 2, 2016: str. 233.

[57] Teoh, S. S.; Bräunl, T.: Symmetry-based monocular vehicle detection system. *Machine Vision and Applications*, ročník 23, č. 5, 2012: s. 831–842.

[58] Tomasi, C.; Kanade, T.: Detection and tracking of point features. 1991.

[59] Wang, C.-C. R.; Lien, J.-J. J.: Automatic vehicle detection using local features—A statistical approach. *IEEE Transactions on Intelligent Transportation Systems*, ročník 9, č. 1, 2008: s. 83–96.

[60] Welch, G.; Bishop, G.: An Introduction to the Kalman Filter. 1995.

[61] Yuan, Q.; Thangali, A.; Ablavsky, V.; aj.: Learning a Family of Detectors via Multiplicative Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 33, č. 3, March 2011: s. 514–530, ISSN 0162-8828, doi:10.1109/TPAMI.2010.117.

[62] Zhang, X.; Zheng, N.; He, Y.; aj.: Vehicle detection using an extended hidden random field model. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, IEEE, 2011, s. 1555–1559.

[63] Zou, B.; Liu, Y.; Zhou, X.: A Robust Vehicle Tracking for Real-time Traffic Surveillance. In *20th ITS World Congress*, 2013.

Přílohy

Příloha A

Instalace

A.1 Nový Docker image

1. stáhnout aktuální verzi kontejneru s ROSem: `docker pull ros`
2. spustit kontejner s podporou GUI, je vhodné přepínačem `-v` nastavit sdílené složky a spustit kontejner v privilegovaném režimu (kvůli pozdějšímu případnému mountování)
3. nainstalovat balíky nutné pro běh systému:
   - *cv_bridge* a *image_transport* (názvy balíků v repozitáři jsou ve formátu ros-<verze>-<balík> - např. *ros-kinetic-cv-bridge*)
   - FFmpeg knihovny (*libavutil*, *libavformat*, ...)
   - *keras* (verze 1.X.X), *tensorflow* a závislosti (*h5py*) - doporučuji pomocí aplikace `pip`