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

DETEKCE, SLEDOVÁNÍ A KLASIFIKACE AUTOMOBILŮ

DETECTION, TRACKING AND CLASSIFICATION OF VEHICLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. RADEK VOPÁLENSKÝ**
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: **Vopálenský Radek, Bc.**
Obor: Inteligentní systémy
Téma: **Detekce, sledování a klasifikace automobilů**
Detection, Tracking and Classification of Vehicles
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte metody pro detekci, sledování a rozpoznání automobilů.
2. Pořiďte vhodný dataset (nahrávky dopravy za různých podmínek) a vytvořte jeho anotaci (bounding boxy automobilů, jejich orientace vůči kameře, typ,...).
3. Experimentujte s detekcí sledováním a klasifikací.
4. Vyhodnoťte výsledky.
5. Vytvořte 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

Cílem této diplomové práce je navrhnout a implementovat v jazyce C++ systém pro detekci, sledování a klasifikaci automobilů ze streamů nebo záznamů dopravních kamer. Systém běží na platformě robotického operačního systému a využívá knihovny OpenCV, FFmpeg, TensorFlow a Keras. Pro detekci je využit kaskádový klasifikátor, pro sledování Kalmanův filtr a pro klasifikaci konvoluční neuronová síť. Úspěšnost detekce je 91.93 %, sledování 81.94 % a klasifikace 63.72 %. Tento systém je součástí komplexního systému, který umí navíc kalibrovat video a měřit rychlost automobilů. Výsledný systém je možné využívat pro analýzu dopravy.

Abstract

The aim of this master thesis is to design and implementation in language C++ a system for the detection, tracking and classification of vehicles from streams or records from traffic cameras. The system runs on the platform Robot Operating System and uses the OpenCV, FFmpeg, TensorFlow and Keras libraries. For detection is used cascade classifier, for tracking Kalman filter and for classification of the convolutional neural network. Success rate for detection is 91.93 %, tracking 81.94 % and classification 63.72 %. This system is part of a comprehensive system, that can moreover calibrate video and measure of vehicles speed. The resulting system can be used for traffic analysis.

Klíčová slova

doprava, detekce, sledování, klasifikace, zpracování obrazu, ROS, kaskádový klasifikátor, Kalmanův filtr, konvoluční neuronová síť

Keywords

traffic, detection, tracking, classification, image processing, ROS, cascade classifier, Kalman filter, convolutional neural network

Citace

VOPÁLENSKÝ, Radek. *Detekce, sledování a klasifikace automobilů.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Juránek Roman.

Detekce, sledování a klasifikace automobilů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Romana Juránka, Ph.D.

.....
Radek Vopálenský
23. května 2017

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce Ing. Romanu Juránkovi, Ph.D. za odborné vedení, za čas, který mi věnoval a rady, které mi pomohly práci vést správným směrem.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá získáváním a využíváním informací v silničním provozu, což je v dnešní době velmi často probírané téma. Při analýze a řešení situací v dopravě je využití počítačového vidění považováno za jednu z nejlepších metod. Informace o aktuální dopravní situaci je potřeba získávat například pro sbírání dat potřebných ke zpracovávání analýz a statistik, pro efektivní ovládání dopravních signalizačních zařízení, pro výpočet tras v GPS navigacích, pro detekci nebezpečných situací na silnicích včetně dopravních nehod atd. Získávání těchto informací je v současnosti důležité hlavně proto, že se stále zvětšuje počet automobilů a tím houstne doprava na silnicích. Proto jsou vyvíjeny různé systémy, které mají za úkol zefektivnit dopravu a zlepšit bezpečnostní situaci tím, že budou regulovat a řídit dopravu podle aktuální situace na komunikacích.

Ke sběru dat lze využít více technologií, například kamery, radary nebo senzory. Kamery se nejčastěji používají pro sledování dopravní situace. Videozáznamy pořízené pomocí kamer jsou následně zpracovávané a analyzované. Získané výstupy se využívají pro zefektivnění a zrychlení dopravy. Tyto kamery mohou být umístěné staticky vedle nebo nad komunikací nebo se používají mobilní kamery umístěné přímo v automobilech. Další možností jsou mobilní kamery nesené dronem. Pro sběr dat se dále využívají také senzory zabudované ve vozovce nebo nad vozovkou, pomocí kterých lze například zjišťovat počty projíždějících automobilů, případně jejich hmotnost apod. Radary se zase využívají pro měření rychlosti vozidel na komunikacích.

Nejvíce informací lze získat z videozáznamů, protože automobily jsou během průjezdu sledovaným úsekem snímány celou dobu kamerou. Po vyhodnocení těchto záznamů je potom možné analyzovat a následně využívat získaná data. Pomocí klasifikace je možné zjistit o jaký druh vozidla se jedná (osobní, nákladní, autobus nebo kamion), u jednotlivých automobilů určit tovární značku a typ nebo rozpoznat konkrétní SPZ. Dále lze z videozáznamů určit aktuální rychlost automobilů nebo např. počet automobilů jedoucích v jednotlivých pruzích na komunikaci pro monitorování hustoty dopravy.

Toto zadání diplomové práce bylo zvoleno proto, že zadaná problematika je v současné době velmi aktuální a výsledný systém je využitelný v praxi. Tento systém běží na platformě robotického operačního systému, který je detailněji popsán v sekci 3.2. Úkolem systému je získávání informací o vozidlech, která projíždějí v záběru dopravní kamery a následné určení jejich tovární značky a modelu. Tato data mohou být získaná buď ze streamu nebo ze záznamu. Vozidlo je nejprve programem detekováno, během pohybu v rámci záběru kamery sledováno a následně je pomocí klasifikátoru určena jeho značka a typ.

Systém vytvořený v rámci této práce je součástí komplexního systému, který je využíván pro analýzu dopravy. Druhou část systému vytvořil Pavel Hájek v rámci své diplomové

práce. Společně vytvořený systém dokáže sledovat jednotlivé automobily během průjezdu v záběru kamery, zkalibrovat kameru a poté, co vozidla opustí kamerou snímaný prostor, určí jejich tovární značku, konkrétní typ a průměrnou rychlost.

Tato práce je rozdělena do dvou stěžejních kapitol a kapitoly, která se věnuje testování systému, experimentům a jejich výsledkům. Kapitola 2 popisuje v první části existující řešení a v její druhé části je potom popsáno a vysvětleno rozdělení a základní popis nejvyužívanějších metod pro detekci, sledování a klasifikaci. V kapitole 3 je nejprve popsána analýza zadaného problému, následuje popis potřebných nástrojů pro správné fungování výsledného systému. Největší část této kapitoly tvoří návrh systému, popis hierarchie systému rozděleného na bloky, které jsou následně jednotlivě a podrobně popsány.

Dále jsou v kapitole 4 uvedeny a popsány výsledky testování a experimentů provedené na jednotlivých částech systému. Následuje závěrečná kapitola 5, kde je shrnuta autorova celoroční práce na vývoji tohoto systému.

Kapitola 2

Existující metody pro získávání informací o objektech z videa

V této kapitole jsou nejprve uvedené postřehy získané studiem odborných článků o řešení systémů podobných tomuto zadání. Následně v jednotlivých sekcích jsou popsané existující metody pro řešení jednotlivých podproblémů, které jsou nezbytné pro správnou funkčnost celého systému.

2.1 Existující řešení

Existuje mnoho řešení podobných tomuto zadání. Nejčastěji jsou v odborných článcích zmiňovaná řešení, která snímají scénu ze statických kamer umístěných například u komunikací ve městech, popřípadě na dálnicích a měří nebo kontrolují rychlost, Re-identifikaci, SPZ automobilů nebo rozeznávají druh vozidla. V následujících dvou podkapitolách jsou okrajově zmíněny články, které se zabývají řešením podobných problémů.

N. Ch. Mithun et al., 2012

Tento článek [14] objasňuje zadanou problematiku o detekci a klasifikaci druhů automobilů. Na rozdíl od této práce program popsaný ve článku nevyhodnocuje konkrétní tovární značku a model automobilu, ale rozlišuje pouze druh vozidla (osobní automobil, autobus, kamion, atd.) a počítá četnost výskytů vozidel těchto jednotlivých kategorií. Autoři použili pro funkčnost systému Hughovu transformaci a pro klasifikování vozidel kNN klasifikátor.

Pomocí experimentů autoři zjistili, že systém počítá četnost vozidel s 97% úspěšností a klasifikuje s úspěšností mezi 88 – 91 %.

H. Emami et al., 2014

Problematiku klasifikace automobilů (značka, typ) popisuje článek [7]. Autoři zde popisují systém, který detekuje pouze zadní část automobilů pomocí určení polohy jejich SPZ a zadních světel. Klasifikaci automobilů provádí systém hierarchicky, kdy nejprve přiřadí automobilu tovární značku a až později mu přiřadí jeho typ.

Jejich systém byl testován na vzorku 280 obrázků zachycujících zadní část automobilů a dosáhl úspěšnosti 96 %.

Shrnutí článků

Ve všech článcích řešících problémy vyjmenované v předchozích podsekcích však lze najít společné rysy. Než je možné začít ať už s klasifikací (automobilů nebo SPZ) nebo například s měřením rychlosti, musí se nejprve automobil v obraze detekovat a pak v době, kdy se objevuje v záběru kamery sledovat. Tento krok je důležitý proto, aby mohl program určit, zda se jedná o jedno konkrétní vozidlo. V opačném případě by v následujících snímcích sice bylo vozidlo detekováno, ale bylo by mu vždy přiděleno nové označení (ID) a pro analýzu dopravy by tyto informace byly nepoužitelné. Pro získání dalších informací o detekci a sledování automobilů, které jsou důležité pro klasifikaci, je vhodný například článek [25], který pochází přímo z této fakulty.

Proces získávání informací o vozidlech (značku a typ) z videa lze tedy rozdělit na tři části:

- **Detekci** – proces vyhledávání automobilů v obraze/videu.
- **Sledování** – proces lokalizace konkrétního pohybujícího se automobilu na všech po sobě jdoucích snímcích videa, dokud automobil neodjede ze záběru.
- **Klasifikaci** – proces určení tovární značky a typu jednotlivých automobilů, které byly detekovány a sledovány v obraze/videu.

Dále je z článků zřejmé, že se existující systémy liší hlavně ve využití různých metod pro jednotlivé části systému. Autoři používají metody, které odpovídají jejich požadavkům. Pro detekci automobilů se využívají metody založené na pohybu nebo na vzhledu, které se dále rozdělují do dalších podkategorií. Pro sledování automobilů se využívají metody, které sledují body, siluety nebo sledují objekty pomocí jádra. Opět se tyto metody dají dále rozdělit. Pro klasifikaci jsou využívány například klasifikátor SVM (support vector machines), AdaBoost (adaptive boosting) nebo klasifikace pomocí konvolučních neuronových sítí.

Cílem druhé části této kapitoly je rozčlenit nejčastěji používané metody, ať pro detekci, sledování nebo klasifikaci objektů, do jednotlivých kategorií.

2.2 Detekce objektů

Detekce objektů v obraze je jeden ze základních úkonů v počítačové grafice. Aby bylo možné objekt z obrazu (videa) sledovat, klasifikovat, analyzovat atd. je třeba ho vždy nejdříve detekovat.

V současné době jsou vyvíjeny stále nové metody pro detekci objektů v obraze, takže těchto metod existuje mnoho. Z tohoto důvodu jsou zde popsané pouze metody zajímavé svým přístupem. Publikace [23] rozděluje tyto metody do dvou hlavních skupin a to na metody založené na pohybu a metody založené na vzhledu. Metody založené na snímání objektů během pohybu potřebují pro detekci objektu sekvenci po sobě jdoucích snímků z videa. Oproti tomu metodám založeným na rozpoznávání objektů podle vzhledu stačí jeden obraz. Zde se objekty detekují podle rozložení pixelů a zvolené reprezentace. Výstupem všech detektorů je seznam souřadnic detekovaných objektů v jednotlivých snímcích.

Metody založené na pohybu

Do této skupiny detektorů patří například rozdílová metoda, detekce na základě pohybu významných bodů a metoda odečítání pozadí. Tyto tři metody jsou používány nejčastěji.

Problémem těchto detektorů může být například znehybnění objektu, tedy zastavení vozidla, kdy hledaný objekt splyne s pozadím. Další problémy mohou nastat při pohybu kamery, protože pro tyto metody je optimální neměnné pozadí.

Rozdílová metoda

Pohybující se objekty jsou detekovány pomocí rozdílu mezi dvěma do sobě jdoucími snímky videa. Výpočet je lehce implementovatelný, jednoduchý a snadno adaptovatelný na různá prostředí. Nevýhodou této metody je chybná detekce pomalu se pohybujících objektů, kde může dojít k rozdělení na více segmentů nebo mohou detekce zaniknout. Další nevýhodou je, že metoda není úplně přesná, protože je problematické získat celý obrys pohybujícího se objektu [16].

Metoda odečítání pozadí

Video sekvence může být rozdělená na popředí (foreground) a pozadí (background). Popředí se skládá ze zájmových objektů a v pozadí jsou objekty nepotřebné k detekci. Proto se odstraní pozadí a zůstane pouze popředí, které obsahuje objekty, které mají být detekované. K tomuto úkonu se musí zjistit referenční model pozadí, který může být buď implicitně daný nebo se vypočítá. Lze využít rekurzivní nebo nerekurzivní výpočet.

Rekurzivní výpočet rekurzivně aktualizuje model pozadí na základě každého snímku, proto nepotřebuje buffer. Díky tomuto přístupu může mít i snímek z dávné minulosti vliv na aktuální model pozadí. Nevýhoda je, že každá chyba v pozadí může přetrvávat velmi dlouhou dobu.

Nerekurzivní výpočet využívá princip posuvného okna na odhad pozadí. V bufferu je uložený určený počet předchozích snímků a aktuální pozadí je odhadnuté na základě změn konkrétního pixelu v průběhu času. Nevýhodou je případně veliká velikost bufferu. Oproti rekurzivnímu výpočtu má tedy větší paměťové nároky.

Nejdůležitější částí je tedy kvalitní tvorba modelu pozadí. Čím lepší spolehlivost, tím jsou lepší výsledky detekce [20].

Detekce na základě pohybu významných bodů

Princip této metody detekce spočívá v pohybu kamery. Skládá se ze dvou kroků: v prvním kroku se v referenčním obraze detekují významné body, kterým se v druhém kroku hledají odpovídající body v následujících snímcích. Body, které se vyskytují v dostatečném množství po sobě jdoucích snímků, jsou označené za objekty, které se pohybují zároveň s kamerou [22].

Metody založené na vzhledu

Tyto metody jsou v problematice detekce vozidel z videa využívané častěji, než metody založené na pohybu [22]. K těmto metodám patří detektory vzorů, detektory tvarů, bodové detektory, detektory detekující pomocí klasifikátoru a detektory detekující podle barvy. V následujících podsekcích jsou tyto detektory jednotlivě popsané.

Detektor vzorů

Na vstupu detektoru je vzor objektu a tento vzor se snaží detektor nalézt v obraze. Při úspěchu nalezne polohu části obrazu odpovídající vzoru. Pro porovnání lze využít dva principy,

buď na základě korelace mezi vzorem a vybranou částí obrazu nebo odčítáním vzoru od vybrané části obrazu. Varianta s korelací je odolnější vůči šumu nebo změně osvětlení, ale je výpočetně náročnější [19, 21].

Detektor tvarů

Tento detektor detekuje objekty v obraze pomocí tvarové analýzy a vyžaduje dobrou segmentaci obrazu. Obrys objektu je reprezentován konečnou množinou bodů, křivkami nebo regiony. Lze použít i zjednodušenou reprezentaci objektů, která ale musí obsahovat informace potřebné k opětovnému získání tvaru. Tato reprezentace se nazývá tvarový deskriptor. Tato metoda se využívá pro jednoduché scény. U složitých scén dochází k problémům kvůli časté ztrátě viditelnosti objektu [19, 21].

Bodové detektory

Princip tohoto detektoru spočívá v tom, že v obraze najde významné body, které mají ve svém okolí výraznou strukturu, kterou může například být textura. Výhodou této metody je, že je odolná proti globálním změnám obrazu, například když dojde ke změně osvětlení, změně měřítka nebo pozice kamery. Mezi zástupce této metody patří: SURF detektor, SIFT detektor nebo Moravcův detektor rohů [19, 21].

Detekce pomocí klasifikátoru

Jiný název této metody je strojové učení s učitelem, kde princip spočívá v tom, že je možné natrénovat klasifikátor, který následně s určitou pravděpodobností rozhoduje, zda se hledaný objekt v obraze nachází nebo ne. Jedná se tedy o vytvoření funkce z poskytnutých dat, která na každý vstup (objekt) mapuje požadovaný výstup (nachází-li se objekt v obraze, vrátí jeho polohu). Klasifikátor se musí natrénovat do dvou tříd: do jedné třídy patří pouze obrázky hledaného objektu a do druhé všechny ostatní. Třídy se musí pro správnou funkčnost navzájem vylučovat. Na vstupu při trénování musí být velké množství dat pro obě třídy, čili pozitivní i negativní prvky (obrázek 2.1) [19, 21].

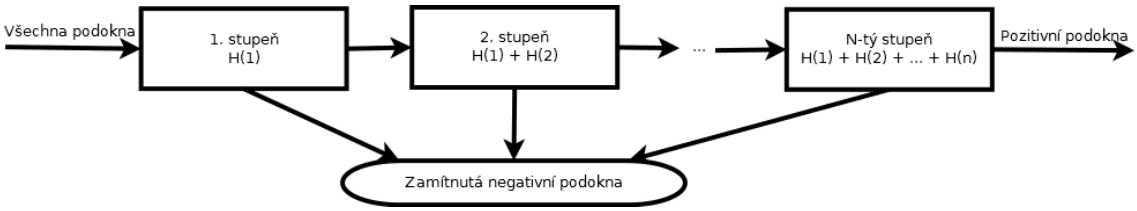
Dále je zde vysvětlen kaskádový klasifikátor, který patří mezi zástupce detektorů pomocí klasifikátorů a který je využit pro detekci automobilů v tomto systému. Další v dnešní době používané klasifikační techniky jsou popsány v podsekci 2.4.

Obrázek 2.1: Levé dva obrázky jsou pozitivní prvky, tedy prvky které obsahují hledaný objekt. Pravé dva obrázky potom představují prvky negativní, tedy prvky které obsahují buď pouze část hledaného objektu nebo jiné objekty.
[1]

[1] Obrázek 2.1 je získán z videa, pomocí kterého byl vyvíjen výsledný systém.

Kaskádový klasifikátor

Kaskádový klasifikátor se skládá z určitého počtu stupňů. Každý stupeň kaskády obsahuje určitý počet *slabých* klasifikátorů. Výsledkem je potom *silný* monolitický nelineární klasifikátor $H(x)$, který má svoji prahovou hodnotu P, podle které rozhoduje, zda aktuální podokno je pozitivní či negativní snímek. Účelem kaskády je urychlit dobu pro detekci hledaného objektu v obraze. Na obrázku 2.2 je znázorněn a vysvětlen princip zapojení klasifikátorů do kaskády. Trénování kaskády může probíhat metodou Viola-Jones [27].



Obrázek 2.2: Schéma principu kaskádového klasifikátoru. Princip spočívá v tom, že na vstupu je velké množství podoken, kdy každý stupeň kaskády pošle pozitivní podokna na další stupeň a zamítne určité množství negativních podoken. První stupeň kaskády obsahuje pouze malé množství klasifikátorů a měl by zamítnout co nejvíce negativních snímků. Následující stupně vždy obsahují všechny klasifikátory ze stupňů předešlých. Každý stupeň je natrénován tak, aby dosahoval určité procentuální hodnoty pozitivních detekcí i hodnoty falešně pozitivních detekcí. Účelem logicky je dosáhnout co největší hodnoty pozitivních detekcí a co nejmenší počet falešně pozitivních detekcí.

Detektory detekující podle barvy

Tato metoda detekuje objekt v obraze pomocí barevných vlastností obrazu. Mezi hlavní vlastnosti pro detekci patří schopnost odrazu od povrchu objektu a spektrální distribuce osvětlení. Nejvíce se využívají modely `L*a*b` nebo `L*u*v` kvůli neuniformnosti RGB modelu. Tento detektor je sice výpočetně nenáročný, ale není příliš efektivní [19, 21].

Shrnutí metod pro detekci

Kromě těchto výše zmíněných metod existují hybridní metody, které kombinují více způsobů a principů pro detekci objektů v obraze. Pro zrychlení a zefektivnění detektoru může programátor určit, ve které oblasti obrazu chce hledaný objekt detekovat. Například pokud se sledovaný jízdní pruh nachází v levé polovině scény, bude stačit, když detektor bude hledat objekty pouze v levé polovině obrazu. Tímto způsobem se také předejde nežádoucím nebo chybným detekcím. Nežádoucími detekcemi je myšleno například, že když je systém určen pro měření rychlosti, nebude detektor detekovat automobily, které stojí na parkovišti, které může být v záběru kamery.

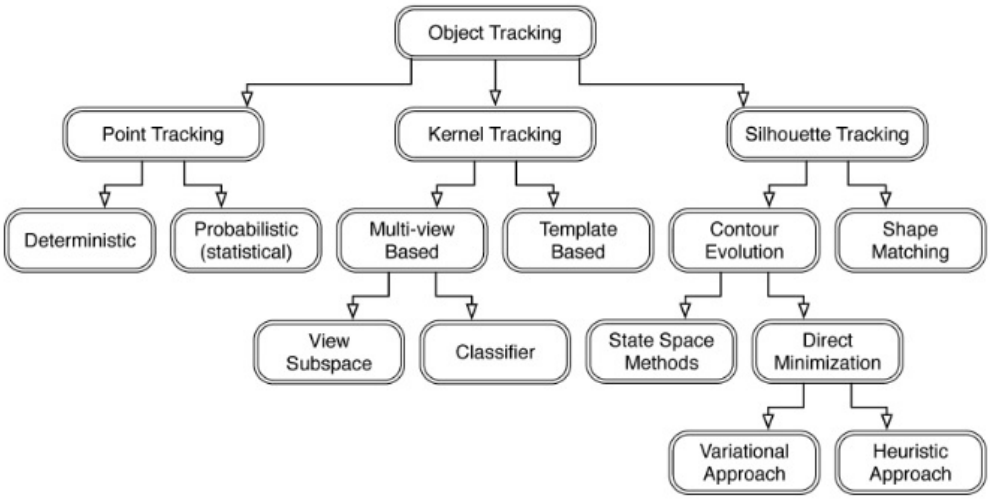
2.3 Sledování objektů

Cílem všech algoritmů zabývajících se sledováním objektu, je vygenerovat jeho trajektorii, která znázorňuje jeho pohyb ve videu. Fáze detekce a generování jeho trajektorie při sledování objektu mohou být spojené do jednoho kroku nebo rozdělené do dvou kroků. Při spojení do jednoho kroku systém detekuje objekty v každém snímku videa a pomocí sledovacích

metod se snaží správně propojit tyto detekce a tím vytvořit trajektorie pohybu jednotlivých objektů. Při rozdělení do dvou kroků je trajektorie získávána aktualizováním pozice objektu v aktuálním snímku na základě jeho předešlých pozic v již vyhodnocených snímcích.

Téměř všechny sledovací algoritmy předpokládají, že pohyb bude plynulý bez náhlých výkyvů a mezi jejich hlavní vlastnosti patří robustnost a jednoduchost. Robustností se myslí, že bude schopný sledovat objekt při chvilkové ztrátě objektu, změně natočení objektu a dalších často se vyskytujících nechtěných jevech. Jednoduchost značí nenáročnou implementaci.

Metody na sledování objektů lze rozdělit na: sledování bodu (point tracking), sledování pomocí jádra (kernel tracking) a sledování siluety (silhouette tracking). Podrobné rozdělení metod je znázorněné na obrázku 2.3 [30].



Obrázek 2.3: Detailní rozdělení metod pro sledování objektů.
[2]

Sledování bodu

Algoritmy pro sledování bodů se využívají při sledování objektu, který zabírá malou část snímku videa (např. hejna ptáků na obloze). Pro sledování větších objektů se využívá princip, kdy se objekt skládá z více bodů. Při používání těchto metod je třeba provádět detekci objektu na každém snímku videa. Metody na zjištění odpovídajících dvojic bodů lze rozdělit na deterministické a statistické (pravděpodobnostní).

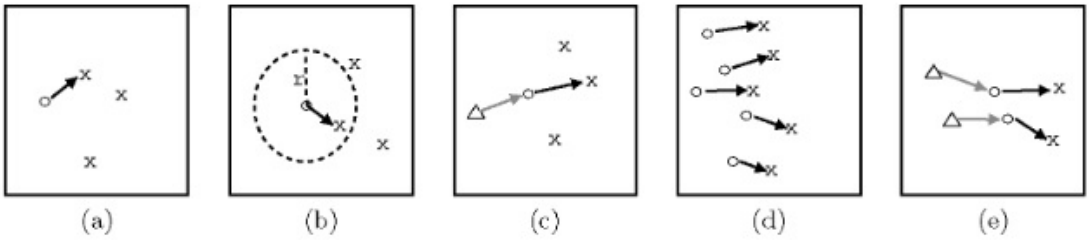
Deterministické metody definují cenu přiřazení každého objektu v předešlém snímku (snímek v čase $t-1$) ke sledovanému objektu v aktuálním snímku (snímek v čase t) pomocí těchto omezení:

- **Blízkost (Proximity)** – situace, kdy se pozice objektu z předešlého snímku ve snímku aktuálním příliš nezmění (obrázek 2.4 (a)).
- **Maximální rychlost (Maximum velocity)** – omezuje pomocí poloměru r vzdálenost, kterou může objekt urazit (obrázek 2.4 (b)).

[2] Obrázek 2.3 je převzat z: [30].

- **Malá změna rychlosti (Small velocity change)** – zaručuje plynulost pohybu objektu tím, že se mezi snímky rychlost a směr pohybu objektu příliš nemění (obrázek 2.4 (c)).
- **Společný pohyb objektů (Common motion)** – situace, kdy se sleduje více objektů najednou. Řeší se pomocí pravidla, kdy objekty zůstávají převážně v podobném sousedském rozestavení (obrázek 2.4 (d)).
- **Tuhost (Rigidity)** – předpokládá, že objekty reálného světa nemění svůj tvar a proto vzdálenost mezi dvěma body jednoho objektu zůstává neměnná (obrázek 2.4 (e)).
- **Blízká jednotnost (Proximal uniformity)** – kombinace blízkosti a malé změny rychlosti.

Tato omezení lze využít i u statistických metod.



Obrázek 2.4: Konstanty omezující pohyb bodu při vytváření trajektorie. Blízkost (a), maximální rychlost (b), malá změna rychlosti (c), společný pohyb (d), tuhost (e).
[3]

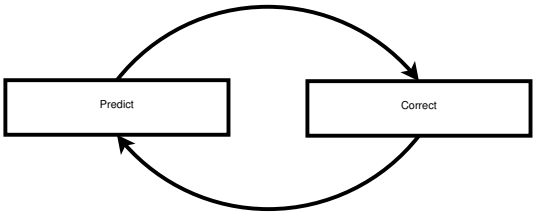
Statistické metody do sledování zanáší šum, aby simuloval klasický šum vzniklý snímáním kamery. Princip těchto metod je rozdělen do dvou modelů. První model popisuje vztah mezi předcházejícím stavem objektu s přidaným bílým šumem a novým aktuálním stavem. Druhý model zpracovává závislost aktuálního stavu se šumem měření. Klasickým představitelem statistických metod je Kalmanův filtr, který je popsán v následující sekci [30].

Kalmanův filtr

Jedná se o rekurzivní filtr používaný k odhadu lineárních dynamických jevů, kdy nejsou k dispozici přesné hodnoty nebo čas měření je delší než rychlost zobrazování. Stav systému je reprezentován vektorem reálných čísel a kovariancí mezi měřenými a odhadovanými hodnotami. Kalmanův filtr pracuje ve dvou základních stavech *Predict* a *Correct*, které se mezi sebou iterativně opakují (obrázek 2.5).

Vylepšením Kalmanova filtru je částicový filtr (particle filter), který se od Kalmanova filtru liší pouze v tom, že už nemá omezení na sledování pouze jednoho kandidáta na sledovaný objekt v aktuálním snímku [29].

[3] Obrázek 2.4 je převzat z: [30].



Obrázek 2.5: Princip Kalmanova filtru. Fáze *Predict* slouží k předpovědi pravděpodobné pozice objektu v následujícím snímku videa a aktualizaci nové chyby měření. Výsledkem této fáze je tedy první odhad polohy objektu pro další krok (snímek v čase $t + 1$). Druhou fází je fáze *Correct* zajišťující opravu predikce na základě měření, čili předpokládanou pozici objektu opraví na skutečnou pozici, kterou později (snímek v čase $t + 1$) zjistí detektor. Dále tato fáze aktualizuje takzvaný *Gain* Kalmanova filtru, který představuje odhad stavu a chybu měření. Pokud dojde k detekci dosud nesledovaného objektu, je vytvořen nový Kalmanův filtr pro tento objekt. A naopak pokud některý ze sledovaných předmětů nebyl na předem určeném počtu snímků detekován, Kalmanův filtr pro tento objekt zanikne a sledování tím skončí, neboť objekt zmizel ze záběru kamery.

Sledování pomocí jádra

Tyto metody jsou využívané pro sledování objektů, které jsou reprezentovány základními geometrickými primitivy, tedy obdélníkem nebo elipsou, které označují sledovaný předmět v obraze. Sledování pohybu se uskutečňuje na základě rotací, zkosení, posunutí nebo transformace daného objektu plynule snímek po snímku nebo hustotou toku. Metody založené na sledování pomocí jádra se dají rozdělit na metody používající modely se vzhledem založeným na šablonách a hustotě (Template and Density-Based Appearance Models) a na vícepohledové metody (Multiview Appearance Models).

Mezi metody využívající modely se vzhledem založeným na šablonách a hustotě patří Mean-Shift (podsekce 2.3) nebo KLT tracker. Nevýhodou těchto metod je, že se jejich modely generují za běhu programu, tedy pokud dojde k 3D rotaci objektu. Rotace mění sledovaný objekt v průběhu sledování a tyto metody potom selhávají.

Problém s měnícím se objektem během jeho sledování řeší vícepohledové metody, které před začátkem sledování natočí model objektu z více různých pohledů. Zástupcem těchto metod je například SVM tracker [30].

Mean-Shift tracking

Mean-Shift slouží pro sledování označené oblasti, ve které se nachází sledovaný objekt. V označené oblasti referenčního snímku je vytvořen a spočítán histogram barev, pomocí kterého je počítána zpětná projekce následujících snímků. Při projekci jsou zvýrazněné barvy vyskytující se v histogramu a potlačené barvy, které se v něm nenachází. Po provedení zpětné projekce je na každém snímku proveden tento algoritmus s počátkem v pozici objektu, kde se nacházel v předešlém snímku a zároveň počítá posuvy okna, které reprezentuje sledovaný objekt. Počítání snímků probíhá ve směru přibývající hustoty.

Výhodou Mean-Shiftu je rychlost, jednoduchost a stačí znát pouze předchozí pozici objektu. Mezi nevýhody patří, že je schopný sledovat pouze barevné předměty a při zakrytí objektu se ztrácí a sledování ukončí i když předmět z videa nezmizel [4].

Sledování siluety

Tyto metody se hodí pro sledování objektů složitějších tvarů, které nelze vhodně reprezentovat pomocí základních geometrických primitiv. Také se dají využívat pokud se má sledovat objekt s velkou přesností, například gesta ruky člověka. Mezi další výhody patří schopnost vypořádat se s dočasnou ztrátou sledovaného objektu nebo s rozdělením a následným spojením sledovaného předmětu. Princip spočívá v nalezení sledovaného předmětu na aktuálním snímku pomocí modelu, který byl vygenerován z předchozích snímků. Tento model je generován na základě tvaru siluety a je reprezentován pomocí histogramu barev, obrysů objektu nebo pomocí hran. Sledování siluety lze dále rozdělit na metody porovnání tvarů (shape matching) a na metody sledování obrysů (contour tracking).

V algoritmech založených na porovnání tvarů je hledání provedeno výpočtem podobnosti siluety objektu a jejího modelu s regiony v aktuálním snímku. Pro zjištění podobnosti siluet se využívá například Hausdorffova metrika [12], která zajistí porovnání dvou množin bodů. Dále lze využít pro výpočet podobnosti siluet detekce siluety na základě určení pozadí scény. Tento algoritmus pracuje tak, že ve snímku porovnává modely (detekce) s částmi siluety. Jedná se o podobný princip, jako u metod pro sledování bodu, s tím rozdílem, že zde se pracuje s modely místo s body. Modely jsou reprezentovány různými kombinacemi hran, obrysů, atd. nebo jsou ve formě rozložení pravděpodobnosti.

Princip metody sledování obrysů se od metody popsané v předchozím odstavci velmi liší. Zde spočívá princip v iterativním vyvíjení obrysu získaného v předchozím snímku tak, aby se co nejvíce podobal obrysu v aktuálním snímku. Podmínka je taková, aby se alespoň část objektu získaného z předchozího snímku zcela překrývala s částí předmětu v aktuálním snímku [30, 16].

Shrnutí metod pro sledování

Nevýhodou metod pro sledování objektů je, že jsou závislé na detektoru. V budoucnu se ovšem očekává spojení optického toku s informacemi o pohybu objektu ve snímcích, což by právě pomohlo vyřešit problém při výpadku detektoru.

2.4 Klasifikace objektů

Cílem klasifikace je navrhnout a vytvořit mechanismus, který na základě trénovacích dat nalezne určitou podobnost (vytvoří model) a po vytvoření modelu dokáže s co nejmenší možnou chybou rozdělit testovací data do modelem vytvořených tříd. Trénování klasifikátoru může probíhat pomocí metod, které buď využívají učitele, nebo které ho nepotřebují.

Výhodou metod trénování klasifikátoru s učitelem je, že mají k dispozici apriorní informaci o třídě. Cílem trénování je, aby klasifikační mechanismus pomocí trénovacích dat zobecnil model a díky tomu s co nejmenší chybou klasifikoval testovací data do tříd.

Naproti tomu metody trénování klasifikátorů bez učitele nemají apriorní informaci o původu dat. Proto se snaží najít mezi daty podobnost, například pomocí eukleidovské vzdálenosti. Data jsou tedy rozdělena podle podobnosti do skupin, které vlastně tvoří jednotlivé třídy.

V dnešní době existuje pro rozpoznání objektů mnoho metod, často se využívají konvoluční neuronové sítě nebo samostatné klasifikátory, mezi které patří AdaBoost či SVM.

Konvoluční neuronové sítě

Konvoluční neuronové sítě (*Convolutional Neural Networks* – CNN) jsou specifickým druhem neuronových sítí, které jsou určené pro zpracování dat s charakteristickou strukturou. V praxi jsou často využívané ke klasifikaci obrázkových dat ze snímků videa, například pro klasifikaci automobilů, které byly detekovány a sledovány v jednotlivých snímcích videa. A proto jsou konvoluční neuronové sítě využity v tomto systému pro klasifikaci automobilů.

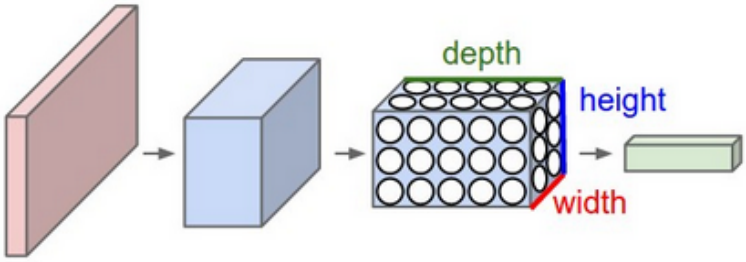
CNN mohou mít různou architekturu a kromě vstupní a výstupní vrstvy se skládají z dalších specifických vrstev. Mezi základní vrstvy patří:

- **Konvoluční vrstva** – je reprezentovaná obdélníkovou mřížkou neuronů a vyžaduje, aby její předcházející vrstva byla také reprezentována touto mřížkou, protože neurony dostávají na vstup data právě z obdélníkové podsekce předchozí vrstvy. Mřížek může být v konvoluční vrstvě více a každá dostává na vstup data od všech mřížek z předchozí vrstvy. Váhy jsou pro všechny neurony v této vrstvě stejné. Konvoluční vrstva je obrazem konvoluce předchozí vrstvy, kde váhy specifikují konvoluční filtr. Typicky tato vrstva obsahuje více konvolučních filtrů.

- **Vrstva poolingu** – v architektuře CNN se přidává za konvoluční vrstvu. Její účel spočívá v podvzorkování konvolučního obrazu na vstupu, tedy tato vrstva dostane na vstup obdélníkové bloky z konvoluční vrstvy a tyto bloky sloučí do jednoho výstupu, čili z n vstupů udělá jeden výstup. Pomocí této vrstvy se dá předcházet přeučení, protože pomocí této vrstvy se dá snižovat počet parametrů a množství výpočtů v síti. Pro *pooling* lze využít více metod, jako jsou například maximum nebo průměr. Nejčastěji se používá funkce pro maximum.

- **Klasifikační vrstva neboli plně propojená vrstva** – je nejvyšší vrstvou konvoluční neuronové sítě. Tato vrstva provádí vysokoúrovňové rozhodování a počet jejích neuronů se rovná počtu tříd, do kterých klasifikuje data ze vstupu. Na vstup této vrstvy jsou poslány všechny výstupy (tedy příznakové vektory vstupního obrazu) z předchozí vrstvy.

Na obrázku 2.6 je znázorněna základní struktura konvoluční neuronové sítě s tím, že tato síť obsahuje vrstvy s trojrozměrným uspořádáním neuronů. Každá vrstva převede trojrozměrný vstupní vektor na výstupní pomocí neuronových aktivací. Na výstupu konvoluční vrstvy je souhrn všech tříd a k nim přiřazené hodnoty pravděpodobnosti, kde tyto hodnoty říkají, ke které třídě vstupní objekt přiřadit. Souhrn těchto pravděpodobnostních hodnot musí být tedy logicky jedna. Pokud je konvoluční síť dobře natrénovaná, hodnoty s nejpravděpodobnější třídou jsou vyšší než 0,85, tedy je velká pravděpodobnost správného zařazení vstupního obrazu [9].



Obrázek 2.6: Vrstvy konvoluční neuronové sítě. V této síti reprezentuje červená vstupní vrstva obrázek, který bude klasifikovaný a výška se šířkou této vrstvy odpovídá rozlišení vstupního obrázku. Modrý kvádr reprezentuje skrytou vrstvu, kde její hloubku určuje počet barevných kanálů, v tomto případě je využit barevný model RGB, čili hloubka je rovna třem. Vrstvy obsahující proměnlivé počty příznakových map, kde příznaky mohou být různé kombinace barev na různých částech obrazu. Zelený kvádr znázorňuje konvoluční vrstvu. Na jejím výstupu je informace, kde ke každé třídě je přidělena pravděpodobnost jejího přiřazení ke vstupnímu obrazu.

[4]

Klasifikátor SVM

Klasifikátor SVM (support vector machines) je metoda strojového učení, kterou lze využít i pro klasifikaci objektů. Jelikož se jedná o metodu strojového učení, je pro jeho správnou funkčnost nezbytné nejprve ho natrénovat na trénovací množině dat. Tato metoda má za cíl oddělit dvě třídy objektů nadrovinou a to tak, aby mezi nimi byla co největší volná plocha. Tato plocha se spočítá jako vzdálenost této nadroviny od nejbližšího objektu dané třídy v n-dimenzionálním prostoru. Objekty nacházející se nejblíže specifikované rovině se nazývají *support vectors* a celá nadrovina je jimi definována pomocí vztahu 2.1. Díky tomuto principu nemůže dojít k *přetrénování* klasifikátoru [15].

$$g(x) = w^T x + w_0 = 0, \tag{2.1}$$

kde w je normálový vektor nadroviny a w_0 je skalární práh. Takto definovaných nadrovin může být nekonečně mnoho.

AdaBoost

Princip metody AdaBoost (adaptive boosting) je vytvoření klasifikátorů pomocí lineární kombinace *slabých* klasifikátorů, které mohou být reprezentovány pomocí rozhodovacího stromu nebo například perceptronem. Podmínkou je, aby jeho chyba byla menší než 0,5. V jednotlivých krocích je k výslednému klasifikátoru přidán *slabý* klasifikátor, který nejlépe klasifikuje špatně ohodnocená data. Klasifikátor vytvořený pomocí tohoto principu už však lineární není. Jeho přesnost je velmi vysoká a je označen jako *silný* klasifikátor.

Původně je AdaBoost navržen pro klasifikaci pouze do dvou tříd, později byla ovšem vyvinuta různá rozšíření, která dovolují klasifikovat do více tříd. Výhodou této metody je velká odolnost vůči *přetrénování* a poměrně snadná implementace. Nevýhodou pak je potřeba velkého množství dat pro trénovací proces [28].

[4] Obrázek 2.6 je převzat z: [9].

Kapitola 3

Systém pro detekci, sledování a klasifikaci automobilů

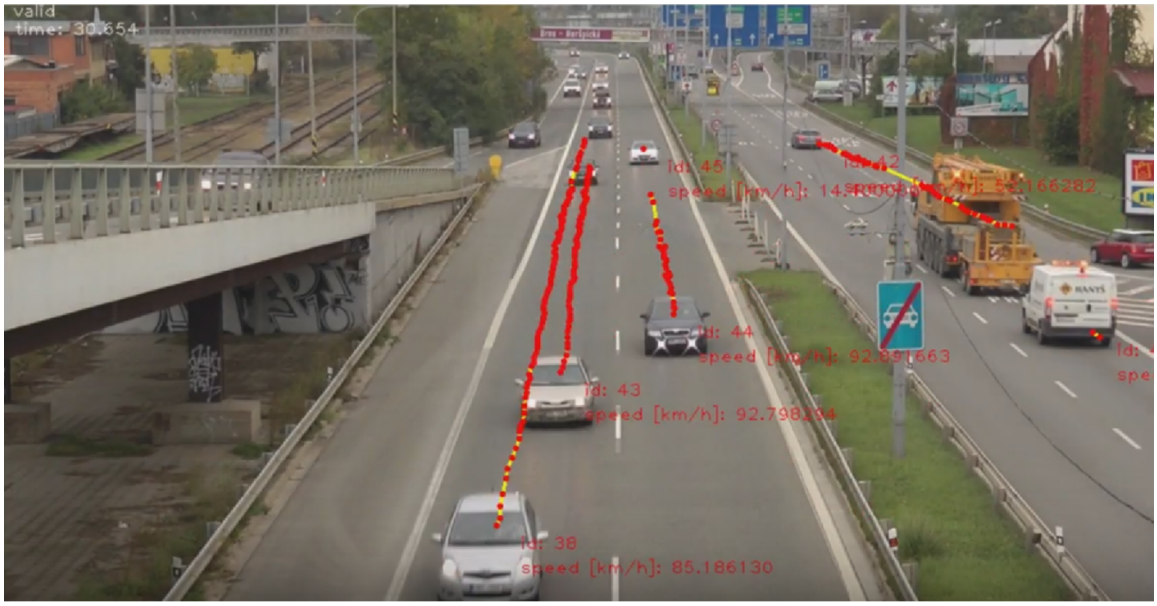
Zadáním této diplomové práce je navrhnout a implementovat systém pro detekci, sledování a klasifikaci vozidel. Systém je navržen tak, že sbírá data z video souborů, které jsou pořízeny statickými kamerami umístěnými nad dopravními komunikacemi. Informace získané z těchto kamer jsou potom zpracovávané v reálném čase nebo jsou záznamy z nich uložené pro budoucí zpracování. Systém, který byl vyvinut v rámci této práce, je součástí komplexního systému, který kromě klasifikace automobilů umí kalibrovat video a měřit rychlost projíždějících automobilů (aktuální i průměrnou). Druhou část implementace tohoto systému vypracoval v rámci diplomové práce s názvem *Odhad rychlosti automobilů ve videu* Pavel Hájek. Jedná se tedy o systém pro analýzu dopravy, který se dá využít pro řízení silničního provozu podle aktuální situace. Na vstupu je očekávané URL streamu nebo uloženého videa a na výstupu jsou publikovány informace o značce, typu a průměrné rychlosti jednotlivých vozidel. Aktuální rychlost je vykreslována přímo do vizualizace. Tato vizualizace není pro spuštění systému nutná, ale pokud je spuštěná, lze pomocí ní kontrolovat, jestli jsou vozidla správně detekována a sledována. Při problémech nebo nepřesnostech lze potom upravit nastavení parametrů v *config* souboru. Vizualizace výsledného funkčního systému je znázorněna na obrázku 3.1.

Tento systém je primárně určen pro videa, kde jsou automobily zaznamenávány zepředu. Video, na kterém byl systém vyvíjen, obsahuje i zřetelný záznam jízdního pruhu se záběry automobilů zezadu. Díky tomu je systém připraven i na detekci a sledování automobilů zezadu, kdy je potřeba pouze aktivovat sledování tohoto jízdního pruhu.

V této kapitole je popsána analýza zadaného problému, dále jsou zde zmíněny nástroje používané při vývoji systému a nástroje potřebné pro jeho správný chod. Ve druhé části této kapitoly je vysvětlen konkrétní návrh systému a je zde popsána hierarchie celého systému. Jednotlivé bloky schématu systému jsou popsány v samostatných sekcích. Podrobněji jsou popsány bloky, které jsou využity při vývoji systému pro detekci, sledování a klasifikaci automobilů a bloky, které jsou využité pro měření rychlosti a kalibraci kamery jsou pouze nastíněné. V popisech jednotlivých bloků jsou popsány i zajímavosti ohledně implementace.

3.1 Analýza problému

Jedním z cílů vývojářů zabývající se zefektivněním dopravy (Inteligentní dopravní systémy [17]) je, aby řidiči nemuseli trávit zbytečně dlouhý čas na křižovatkách nebo v kolonách



Obrázek 3.1: Ukázka vizualizace výsledného systému. Každému sledovanému automobilu je vykreslována trajektorie jeho dráhy. Dále každé vozidlo je označeno svým ID číslem a informací o aktuální rychlosti.

a aby se zvýšila bezpečnost. Příkladem takové situace je křižovatka řízená semafory, kde je přepínání barev nastaveno staticky na určitý časový interval. Tedy i když je na vozovce jediné vozidlo, řidič musí předem nastavený časový interval čekat na zelenou. Moderní systém pro analýzu dopravy by v této situaci řidiči, který je na křižovatce sám, sepnul zelenou v jeho směru jízdy okamžitě.

V současnosti, kdy jsou informační technologie na vysoké úrovni, se stále častěji využívají počítačové systémy, které jsou schopné na základě dat z kamer získat a využít více různých informací najednou. V minulosti bylo nutné pro získání každé informace použít jiný postup (např. rychlost bylo možné měřit pouze pomocí radaru [24] nebo efektivní řízení křižovatek na základě aktuální dopravní situace museli provádět policisté osobně).

Proces zpracování dat lze rozdělit na dvě části, na sběr dat (z kamery) a software (počítačový program, který provádí na základě získaných dat požadované úkony). Pro co nejlepší výsledky je důležitá jak kvalita získaných dat, tak i kvalita vytvořeného softwaru.

Analyzovat dopravu pomocí kamer je také výhodné z hlediska snadné instalace. Stačí nainstalovat kameru na místo, kde je žádoucí dopravní situaci sledovat a mít k dispozici funkční systém pro určenou problematiku. Další výhodou je, že získaná data mohou být zálohována na požadovanou dobu a následně je možné je využít pro zpracování různých statistik (např. počet projetých automobilů, nejčastější značky automobilů, nejrozšířenější barva vozidel, atd.) v určených časových intervalech (denně, týdně, měsíčně, atd.).

Pro získání kvalitních výsledků je rozhodující odpovídající kvalita záznamu (záleží na technických parametrech kamery) a zároveň i vhodné umístění kamery. Umístění kamer není omezeno pevně daným úhlem, ze kterého snímají vozovku, je tedy možné kamery umístit na budovy, mosty nebo jiné konstrukce. Důležité však je umístit kamery v dostatečné výšce nad komunikacemi tak, aby se snímaná vozidla při průjezdu sledovaným úsekem co nejméně vzájemně zakrývala. Při častém zakrývání vozidel by nebyly zakryté automobily zaneseny do statistik a výsledky by byly zkreslené.

Mezi další faktory, které ovlivňují správnou činnost systému patří meteorologické podmínky. Například silný sluneční svit může zastiňovat vozidla, nebo může záběry z kamer značně znepřehlednit hustý déšť nebo sněžení.

3.2 Nástroje

V této sekci jsou stručně popsány nástroje a knihovny, které jsou využity pro implementaci a správný chod celého systému. Je zde popsán Robotický operační systém, Docker a knihovny FFmpeg, OpenCV, TensorFlow a Keras.

Robotický operační systém

Hlavním cílem robotického operačního systému (ROS) je umožnit vývojářům rychlé a snadné vytváření modulů pro aplikace na společné platformě. Nabízí vývojářům hardwarovou abstrakci, ovladače zařízení, knihovny, vizualizéry, předávání zpráv mezi procesy a správu balíčků. Aplikace vytvářená v ROSu se skládá z více částí, které se nazývají *nody*. Jednotlivé *nody* jsou opakovaně použitelné pro další nové systémy. ROS je licencován pod open source na základě BSD licence a rozšířen po celém světě. Dále jsou zde popsány mechanismy ROSu, které jsou využity pro komunikaci systému. Informace o ROSu a jeho mechanismech byly čerpány z publikace [18].

Node

Každý *node* musí být pojmenovaný. *Node* je vlastně proces, který provádí výpočet. To znamená, že pomocí jednoduchých *nodů* je implementována část řešení aplikace, která je v ROS tvořena mnoha *nody*. Mnoho *nodů* pro různá využití je už vytvořeno, jsou volně dostupné a lze je použít pro vlastní řešení. Jednotlivé *nody* mezi sebou komunikují pomocí vzájemného zasílání zpráv na *topicy* nebo pomocí *service*.

Topic

Topic je pojmenovaný komunikační kanál mezi publisherem a subscriberem, přes který jednotlivé *nody* komunikují. Název *topicu* se používá k identifikaci obsahu zprávy. *Node* (publisher) vysílá zprávu tak, že zveřejní dotyčné informace na daný *topic*. *Node* (subscriber), který má zájem o určitý druh dat, se připojí na příslušný *topic* a z něho čte potřebná data. Na jediný *topic* může být zároveň připojeno více publisherů i subscriberů. Jeden *node* může zveřejnit a nebo se přihlásit k odběru z více *topiců*. Pro zjištění dostupných *topiců* slouží příkaz: `rostopic list`.

Message

Message je datová struktura, která reprezentuje *topic*. Tato datová struktura může obsahovat celé nebo desetinné číslo, pole, atd. Zprávy mohou také obsahovat libovolně vnořené struktury. Výpis zpráv z jednotlivých *topiců* lze vypsat pomocí příkazu: `rostopic echo jméno_tématu`.

Service

Service funguje na principu dotaz-odpověď. To znamená, že server přijme požadavek od klienta a *node* podporující tento požadovaný *service* ho provede. Klient následně obdrží odpověď. Struktura *service* je dána souborem typu *srv*. Pro zjištění dostupných *service* slouží příkaz: `rosservice list`.

Roslaunch

Roslaunch spouští předem definované *nody* s parametry, vlastními jmennými prostory a přemapováním *topiců* na vstupy/výstupy. Formát příkazu pro spuštění: `roslaunch adresář název_tématu.launch`.

Rosrun

Rosrun slouží k spouštění jednotlivých *nodů*, které jsou implementovány v jazyce Python. Formát příkazu pro spuštění: `rosrun adresář název_uzlu.py`.

Docker

Docker je určený primárně k usnadnění vývoje, údržby, dodávání a provozování aplikací pomocí aplikačních kontejnerů. Tedy umožňuje aplikaci izolovat v rámci kontejneru, společně se všemi jejími závislostmi, potřebnými knihovnami a programovým prostředím, od ostatních procesů na systému hostitele. Jedná se o open source platformu [13].

FFmpeg

FFmpeg je open source multimediální framework pod licencí LGPL nebo GPL, který umožňuje nahrávání, přehrávání, konverzi a streamování digitálního zvuku a obrazu. Jedná se o velmi rychlý audio/video konvertor. Jako zdroj pro *FFmpeg* může posloužit téměř cokoliv, od jednoduchého video souboru až po vysílání streamu. Knihovna obsahuje velké množství různých kodeků pro dekódování videa, jako jsou například `MPEG-4, H.264, AAC, FLAC`.

Princip činnosti spočívá v počátečním volání knihovny *libavformat* k načtení vstupních snímků a k získání dat pomocí demuxeru. Kódované pakety jsou potom poslané do dekodéru. Dekodér produkuje nekomprimované rámce (raw video, PCM audio, atd.), které mohou být zpracované filtrováním. Následně po filtrování jsou data poslaná do kodéru, kde jsou zakódovaná a posílaná jako kódované pakety. Nakonec jsou tyto pakety poslané na vstup muxeru, který je zapíše do výstupního souboru [3].

OpenCV

OpenCV (Open Source Computer Vision Library) je open source knihovna pod licencí BSD nabízející implementaci široké škály algoritmů z oboru počítačového vidění, zpracování obrazu a strojového učení. Vyřeší problematiku jako je filtrace a segmentace obrazu, rozpoznávání a sledování objektů, kalibrace kamer, spojování snímků, analýzy pohybu nebo 3D rekonstrukce scény. Tato knihovna byla navržena s důrazem na výpočet v reálném čase [2].

TensorFlow

TensorFlow je open source software knihovna pro strojovou inteligenci. Provádí numerické výpočty pomocí grafu toků dat. Uzly v grafu představují matematické operace a hrany

grafu reprezentují multidimenzionální datová pole neboli tenzory. Flexibilní architektura určitého prostředí umožňuje nasazení na jeden popřípadě více CPU nebo GPU. *TensorFlow* byl vyvinut vědci a inženýry pracujícími pro Google. Dnes je ovšem volně dostupný [1].

Keras

Keras je open source vysokoúrovňová knihovna pro práci s neuronovými sítěmi. Pro složité výpočty využívá *TensorFlow*, které bylo vyvinuto s důrazem na umožnění rychlého experimentování. Tato knihovna je schopná běžet bez problémů na CPU nebo na GPU. Mezi hlavní výhody patří snadná rozšiřitelnost, nové moduly lze bez problémů přidat ke stávajícímu projektu. Modul je chápán jako graf nebo sekvence grafů a měl by být krátký a jednoduchý [10].

3.3 Návrh systému

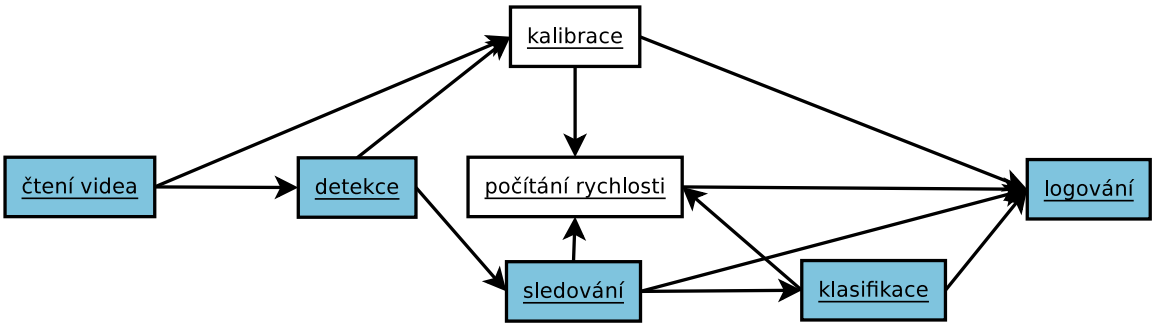
Aplikace běží na platformě ROS a je implementovaná v programovacím jazyce C++, který ROS plně podporuje. ROS jsem vybral vzhledem k tomu, že podporuje škálovatelnost aplikace na jednotlivé bloky a mechanismy (*topic* a *service*). Tyto mechanismy umožňují komunikaci mezi jednotlivými *nody*. ROS jsem také zvolil z důvodu jeho podpory pro práci s knihovnami *OpenCV, FFmpeg, TensorFlow* nebo *Keras*, které jsou využity pro implementaci tohoto systému. Vzhledem k náročné a mnohdy zdlouhavé instalaci ROSu, zejména na operační systémy, které nemají jeho plnou podporu, jsem zvolil použití ROSu v rámci *docker* kontejneru. *Docker* umožňuje snadnou instalaci, přenositelnost a znovupoužití jeho *obrazů*. Pomocí něj byl ROS velmi rychle zprovozněn a během vývoje systému fungoval bezproblémově. Ovšem použitý *docker image* měl k dispozici pouze knihovny *FFmpeg* a *OpenCV*. Knihovny *TensorFlow* a *Keras*, což jsou prerekvizity pro klasifikaci, musely být dodatečně doinstalovány.

Vstupem programu je URL streamu nebo video souboru uloženého na disku (povinný vstupní parametr) a pro kontrolu fungování programu je možné sledovat aktuální scénu pomocí vizualizace. Další vstupní parametry jsou určeny pro vyladění detektoru, trackeru a pro případné individuální nastavení systému. Na výstupu (v logovacím souboru ve formátu JSON) jsou potom uloženy získané informace o projíždějících vozidlech ve tvaru: id, čas průjezdu, tovární značka a model. Tento logovací soubor je výsledkem části systému, který klasifikuje sledovaná vozidla. Systém generuje ještě druhý logovací soubor, který uchovává informace potřebné k zjištění průměrné rychlosti vozidel.

Jak už bylo uvedeno na začátku kapitoly 2 v sekci Existující řešení, lze řešení tohoto systému rozčlenit na bloky pro detekci, sledování a klasifikaci vozidel. Protože byl systém pro klasifikaci sledovaných vozidel sloučen se systémem pro měření rychlosti do jednoho komplexního systému, jsou v návrhu řešení uvedeny i bloky pro kalibraci kamery a pro zjištění rychlosti. Pro správné fungování systému je nutný i blok pro otevření a překódování vstupního videa a blok pro logování výsledků. Na obrázku 3.2 je blokové schéma celého systému se správným vnitřním zapojením jednotlivých bloků a tím jsou znázorněny i komunikační kanály mezi jednotlivými *nody*.

Celý systém je tedy rozložen na následující ROS balíčky:

- **Hlavní uzel** – `traffic` – základní blok, který sdružuje potřebné informace od všech ostatních *nodů* a na jejich základě řídí celý systém, zajišťuje vizualizaci a logování výsledků.



Obrázek 3.2: Blokové schéma vnitřního zapojení celého systému. Jednotlivé bloky představují v ROSu *nody*. Světle modře označené bloky jsou ty, které jsou použity pro správný chod systému pro detekci, sledování a klasifikaci automobilů.

- **Čtení videa** – `traffic_ffmpeg` – otevře, čte a dekóduje vstupní video. Jeho vstupem je URL streamu nebo video souboru a výstupem potom jednotlivé snímky videa zapisované na zadaný *topic*.
- **Detekce automobilů** – `traffic_detection` – detekuje automobily v jednotlivých snímcích videa. Jeho vstupem jsou tedy snímky videa a jako výstup vrací informace o nalezených automobilech ve formě jejich bounding boxů, tj. rámeček, který označuje automobil. Všechny tyto informace jsou postupně publikovány na *topic*.
- **Sledování automobilů** – `traffic_tracking` – na vstupu tohoto bloku jsou informace z *topicu*, který publikuje výstup detektoru a na výstupu publikuje výsledek sledování automobilů. Tento balíček tedy doplňuje ID jednotlivých vozidel a tím spojuje více bouding boxů z různých snímků. Díky tomu je vytvořena správná trajektorie dráhy pohybu jednotlivých automobilů.
- **Klasifikace automobilů** – `traffic_classification` – na vstupu jsou snímky (výřezy) jednotlivých vozidel a na výstupu ID třídy, která reprezentuje tovární značku a model automobilu, jehož snímek byl na vstupu.
- **Kalibrace kamery** – `traffic_calibration` – na vstupu očekává stream nebo video soubor a na výstupu poskytuje kalibrační údaje ve formě vanishing points a údaj, jestli je výstup validní.
- **Měření rychlosti** – `traffic_speed` – z informací o pozici automobilu počítá jeho rychlost.

Komunikace

Systém využívá pro komunikaci mezi *nody* mechanismy ROSu, jednotlivé bloky mezi sebou komunikují pomocí *topicu* nebo mechanismu *service*. Pro tento účel je nutné definovat nové datové typy. Například aby bylo možné obrázek uložit na *topic*, musí být převeden z typu *Mat* na datový typ *sensor_msgs/Image*, který používá ROS. Definice těchto datových typů jsou umístěny v hlavním balíčku *traffic*. V tabulce 3.1 je zobrazen způsob komunikace a sdílení dat mezi jednotlivými bloky. Podrobněji je komunikace mezi *nody* popsána v sekcích pro jednotlivé bloky systému.

Tabulka 3.1: Popis komunikace bloků mezi sebou.

Informace (médium)	Položky
video (*topic*)	snímek pořadové číslo snímku čas snímku
kalibrace (*topic*)	tři 2D vanishing points třetí 3D vanishing point fokální vzdálenost validnost aktuální zprávy
detekce a sledování (*topic*)	ID automobilu střed jeho bounding boxu výřez automobilu
měření rychlosti (*service*)	ID automobilu první výskyt druhý výskyt časy výskytů
klasifikace (*service*)	seznam výřezů automobilu ID třídy modelu

3.4 Hlavní uzel

Hlavní uzel obsahuje *launch* soubory (spouštěcí soubory) a *config* soubory, pomocí kterých se nastavují před spuštěním programu všechny parametry nezbytné pro správný chod systému. Dále se zde nacházejí soubory typu *msg*, které reprezentují strukturu jednotlivých *topiců*, na kterých jsou sdružovány potřebné informace. Dalšími soubory, které jsou potřebné pro chod aplikace, jsou soubory typu *srv*. Tyto soubory reprezentují strukturu zpráv pro komunikační mechanismus *service*.

Kromě těchto specifických souborů, které se využívají pro implementaci na platformě ROS, se zde nachází klasický C++ soubor a jeho hlavičkový soubor. Pomocí těchto dvou souborů jsou ovládány všechny ROS balíčky znázorněné v blokovém schématu. Z těchto *nodů* získává postupně hlavní uzel potřebné informace a to pomocí mechanismů *topic* nebo *service*. Tento hlavní *node* je schopný komunikovat se všemi bloky systému, ostatní bloky komunikují pouze se svými sousedy.

Během běhu systému jsou získané informace ukládány do třídy *Car*. Tato třída obsahuje všechny potřebné informace o vozidlech: ID, bouding boxy jednotlivých detekovaných automobilů, výřezy všech vozidel z jednotlivých snímků videa, čas průjezdu, čísla snímků, kde se jednotlivá vozidla objevila, průměrnou a aktuální rychlost, značku a typ automobilů. Dále se zde nacházejí funkce, pomocí kterých se ukládají informace nebo naopak zjišťují hodnoty proměnných náležící do této třídy. Tato třída také obsahuje konstruktor a destruktor a funkce pro logování výsledků.

Dále je zde implementován *timer*, který každé dvě vteřiny zjišťuje, zda bylo dokončeno sledování jednoho nebo více automobilů. Pokud bylo dokončeno, jsou zkompletovány informace o daném vozidle, které jsou nutné pro určení průměrné rychlosti a pro proces klasifikace. Výstupem je potom seznam vozidel, která už opustila prostor snímaný kamerami a tato vozidla jsou následně klasifikována, respektive jim je vypočítána průměrná rychlost. Informace potřebné pro klasifikaci (ID a výřezy automobilu) jsou uloženy do fronty. Tam jsou postupně ukládány údaje o všech vozidlech, která už nejsou v záběru kamery.

Do fronty jsou informace o jednotlivých automobilech ukládány proto, že pro klasifikaci se nejprve musí načíst model, pomocí kterého jsou automobily zařazovány do jednotlivých tříd. K načtení modelu je potřeba určitý časový úsek, proto není k dispozici pro klasifikaci hned po spuštění systému. Ve chvíli, kdy je model načten, začíná proces klasifikace jednotlivých informací o vozidlech z fronty. Protože klasifikace je náročný proces, běží paralelně s hlavním uzlem v druhém vláknu.

Proces získávání informací o průměrné rychlosti běží v samostatném vláknu paralelně s hlavním uzlem a klasifikací. Stejně, jako v případě klasifikace, jsou údaje o jednotlivých vozidlech, která opustila zorné pole kamery, ukládány do vlastní fronty.

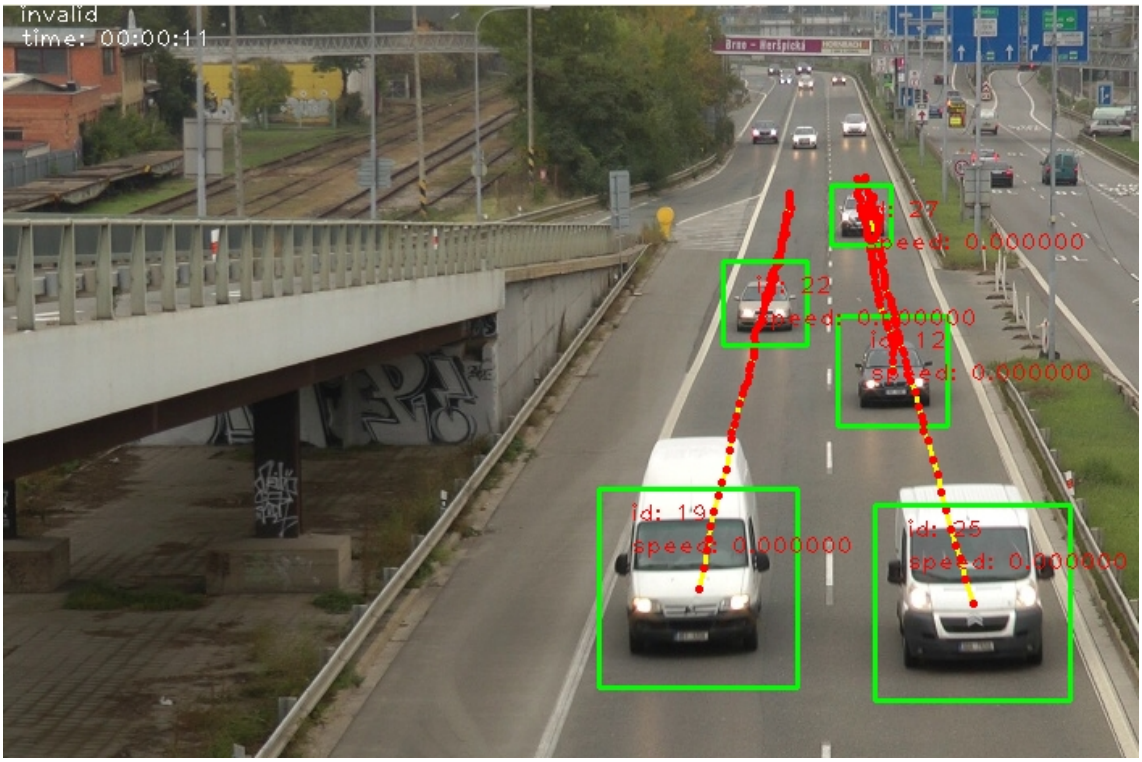
Díky rozdělení běhu systému na vlákna nedochází ke zpomalení, zejména vizualizace vstupního videa. Ta se při běhu systému na jednom vlákně prakticky nepohybovala a tím pádem nebylo možné takovou vizualizaci používat. Po přijetí posledního snímku videa do hlavního uzlu a jeho zpracování končí činnost všech *nodů* kromě klasifikace. Ta začala se zpožděním, proto dál probíhá a to i vzhledem k její větší výpočetní náročnosti. Hlavní uzel čeká, až se ukončí klasifikace a fronta údajů o neoklasifikovaných vozidlech bude prázdná. V tomto okamžiku hlavní uzel ukončí činnost celého systému.

Vizualizace

Hlavní uzel z dat o jednotlivých snímcích videa získávaných z *nodu* pro čtení videa vytváří vizualizaci vstupního videa. Do obrazu kromě snímků vstupního videa vykresluje i další informace. V levém horním rohu jsou zobrazeny informace o zkalibrování videa (valid x invalid) a časová stopa videa. Dále jsou detekované a sledované automobily označené zelenými rámečky, které reprezentují bounding boxy. V rámečku je také zobrazena informace o ID vozidla a jeho aktuální rychlosti. Poslední zobrazená informace je trajektorie jednotlivých vozidel (žlutá čára), kde červené tečky označují předešlé polohy automobilu. Ukázky vizualizací jsou na obrázcích: 3.1 pro celkový výsledný systém a 3.3 pro část systému, která pouze klasifikuje automobily. Obrázky vizualizací, na kterých jsou ilustrovány problémy, které nastávaly během implementace (screeny byly pořizovány během vývoje), se mírně liší od vizualizace výsledného systému. Při vývoji části systému pro klasifikaci byly spouštěny pouze potřebné *nody*, proto je v obrázcích, které zobrazují chyby při vývoji systému, špatně uvedena informace o aktuální rychlosti (pro všechny automobily 0.0) a kamera není zkalibrovaná (invalid). Dalším viditelným rozdílem mezi původní a výslednou verzí vizualizace systému jsou čtverečky označující automobily v obraze. Tyto čtverečky sloužily během vývoje pro kontrolu správné detekce a následné otestování úspěšnosti detektoru.

Konfigurační soubor

Konfigurační soubor, který je ve formátu YAML, obsahuje pro všechny *nody* definice názvu jejich komunikačních kanálů (*topic* nebo *service*) a vstupní parametry (URL videa, parametry pro detektor a tracker nebo cesty ke kaskádě a modelu pro klasifikaci, atd.). Dále se zde nastavuje umístění pro logovací soubory. Pomocí tohoto řídícího souboru lze přizpůsobit systém konkrétním požadavkům a nastavit, zda se využije celý systém nebo pouze jeho část, ať už část pro klasifikaci, měření rychlosti nebo kalibraci kamery.



Obrázek 3.3: Ukázka správné činnosti části systému, kdy systém pouze klasifikuje automobily, proto je pomocí vizualizace kontrolována pouze detekce a sledování jednotlivých automobilů.

3.5 Čtení videa

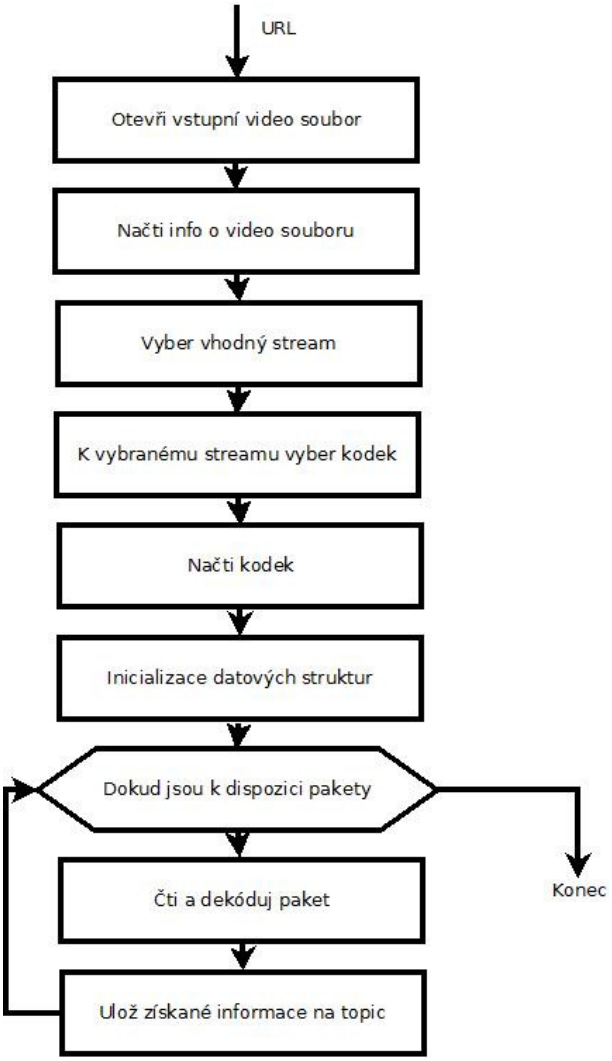
Pro implementaci otevření, čtení a dekódování vstupního videa je využita knihovna *FFmpeg*. Tato knihovna obsahuje všechny potřebné nástroje pro práci s videem a umožňuje otevřít nahraná videa i streamy. Výhodou knihovny *FFmpeg* je, že otevře jakékoliv video bez nutnosti parsovat zadané vstupní URL.

Na vstupu tedy je URL video souboru, což je vlastně hlavní vstupní parametr celého systému. Výstupem tohoto bloku je sekvence snímků vstupního videa, které jsou postupně zapisovány na zadaný *topic*.

Princip bloku pro čtení videa

Princip tohoto bloku je zobrazen na vývojovém diagramu (obrázek 3.4). Podrobněji je zde popsán pouze cyklus, kroky předcházející cyklu jsou snadno pochopitelné z vývojového diagramu. Jak je zřejmé z vývojového diagramu, tento blok postupně v cyklu čte a dekóduje jednotlivé rámce na snímky. Získaný snímek je následně převeden na datový typ *sensor_msgs/Image*, aby ho bylo možné uložit na zadaný *topic*. Poslední fází cyklu je uložení informací na *topic*. Ukládají se informace o ID snímku a jeho čase, ve kterém se tento snímek objevuje ve videu (v sekundách a milisekundách). A samozřejmě je uložen i samotný snímek.

Protože ROS má větší režii, co se týká zejména kopírování výřezů na *topicy*, pro videa, která mají vysokou hodnotou FPS, systém neběžel v real-time. Aby systém fungoval



Obrázek 3.4: Princip čtení vstupního videa pomocí knihovny *FFmpeg*.

co nejrychleji a odpovídal real-time, mohou být vynechávány některé pořízené snímky videa. To, že jsou snímky vynechané, není při vizualizaci zřejmé, protože testovaná videa mají vysokou hodnotu FPS, při které je vynechání menšího počtu snímků zanedbatelné. Počet vynechaných snímků (kladné celé číslo) lze přizpůsobit podle zadaných požadavků na systém. Při velkých hodnotách klesá přesnost systému a vizualizace nemusí být plynulá (automobily v obraze přeskakují), proto se tato hodnota nastavuje na malá čísla.

Také je v rámci procesu zrychlení půleno rozlišení vstupního videa, dokud není šířka nového rozlišení menší než 720 pixelů. Program dále i přes půlení rozlišení pracuje s původními hodnotami velikostí souřadnic videa.

Tento *node* běží pouze během otevření, čtení a dekódování videa. Po uložení posledního snímku na *topic* (pokud se nejedná o stream), je tento blok neaktivní, ovšem informace jím zapsané na *topic* jsou stále k dispozici.

3.6 Detekce automobilů

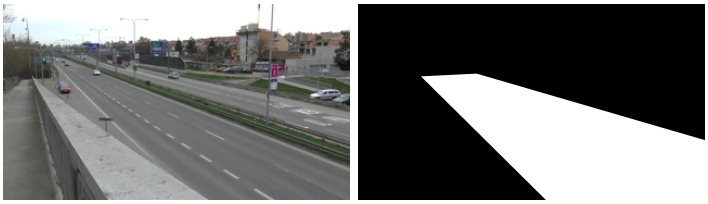
Aby bylo možné automobily sledovat, klasifikovat nebo jim měřit rychlost, musí být nejprve ve videu detekovány. Pro detekování automobilů byl zvolen kaskádový klasifikátor [27], který pro detekci potřebuje natrénovanou kaskádu.

Princip bloku pro detekování automobilů

Na vstupu tohoto bloku musí být k dispozici sekvence snímků vstupního videa. Tyto snímky *node* postupně čte z *topicu*, na který se snímky během čtení videa ukládají. Další podmínkou pro fungování detektoru je mít na vstupu natrénovanou kaskádu pro vyhledávání objektů (v tomto případě automobilů) ve snímcích videa.

Detektor tedy čte z *topicu* jednotlivé snímky videa a pro každý snímek pomocí kaskády vytvoří seznam nalezených automobilů a tyto informace v jednotlivých iteracích posílá přes tomu určený *topic* bloku systému, který obstarává sledování automobilů. Na tento *topic* se ukládá poloha jednotlivých automobilů ve formě bounding boxů, což je rámeček, který ohraničuje nalezené vozidlo ve videu. Bounding box obsahuje souřadnice levého horního rohu rámečku (X a Y) a informace o jeho velikosti (šířka a výška). Pomocí získaných bounding boxů nalezených automobilů jsou ze snímků videa získávány výřezy aktuálně detekovaných vozidel. Ty jsou dále ukládány do třídy *Car*, protože jsou v dalším průběhu systému využity pro klasifikaci.

Ke vstupnímu streamu nebo video souboru lze přidat do konfiguračního souboru i vstupní masku tohoto videa (obrázek 3.5), pomocí které se detektor rapidně zrychlí, neboť hledá automobily pouze tam, kde by se měly pohybovat, tedy pouze na silnicích a v jízdních pruzích a směrech, které mají být analyzované.



Obrázek 3.5: Levý obrázek zobrazuje screen videa a pravý obrázek masku tohoto videa. V případě přítomnosti masky vyhledává detektor automobily pouze v její bílé oblasti.

Před spuštěním systému je možné měnit parametry detektoru na základě použitých videí. Mezi parametry, které ovlivňují činnost detektoru, patří:

- **scale_factor** (double) – určuje, o kolik je velikost snímku snížena na obrazovém měřítku a pomocí toho určuje rychlost a důkladnost detekce.
- **min_neighbors** (int) – určuje, kolik sousedů má každý kandidát, čímž určuje přesnost a počet detekcí. Při nastavení tohoto parametru na hodnotu 4, která je v tomto systému používána, je detekce přesnější a počet detekcí menší, protože detekuje pouze objekty u kterých je vyšší pravděpodobnost, že se opravdu jedná o automobily.
- **min_size** a **max_size** (Size) – určují minimální a maximální rozměry detekovaných objektů. Tyto dva parametry jsou nepovinné. U parametru *min_size* testy ukázaly, že stačí nastavit minimální velikost hledaných objektů kolem 30 pixelů (tedy 30 x 30). Menší obrázky objektů jsou tak malé, že je detektor nerozpozná a případné automobily

detekuje, až když se přiblíží ke kameře. Tuto skutečnost ilustrují všechny screeny (například obrázky 3.3, 3.7 nebo 3.9), kde automobily, které se objevují v horní části obrazu nejsou detekovány, protože nedosáhly potřebné velikosti. Parametr *max_size* není v tomto systému nastaven.

Popis kaskády pro detekci

Pro získání kaskády pro detekování automobilů existuje více způsobů. Buď si musí uživatel natrénovat svoji kaskádu nebo může využít již natrénovanou kaskádu. Kaskáda je ve formátu XML. V této práci je použita fakultní již natrénovaná kaskáda.

Tato kaskáda je natrénovaná na fakultním datasetu *COD20K Dataset* [8]. Tento dataset obsahuje přibližně 10 000 trénovacích snímků s téměř 20 000 komentovaných anotací, kde se v jednotlivých snímcích nacházejí všechny automobily. Počet komentovaných anotací je dvojnásobně větší, než počet trénovacích snímků, protože každý snímek je zrcadlově převrácen, aby detektor uměl detekovat automobily ze všech stran. Dále dataset obsahuje přes 1 100 testovacích snímků. Kaskáda neumí detekovat motocykly.

Vzhledem k tomu, že byla k dispozici spolehlivá kaskáda, rozhodl jsem se detekovat vozidla právě pomocí detektoru, který detekuje objekty pomocí kaskády. Kaskádový klasifikátor je implementován knihovnou *OpenCV*[1], která je podporována platformou ROS. Což byl další důvod, proč zvolit tuto metodu.

Detektor umí detekovat automobily pomocí této dobře natrénované kaskády z více stran a ne pouze zepředu, jak je vidět na obrázku 3.3. Na ostatních obrázcích je prezentovaná správná detekce automobilů zepředu i zezadu.

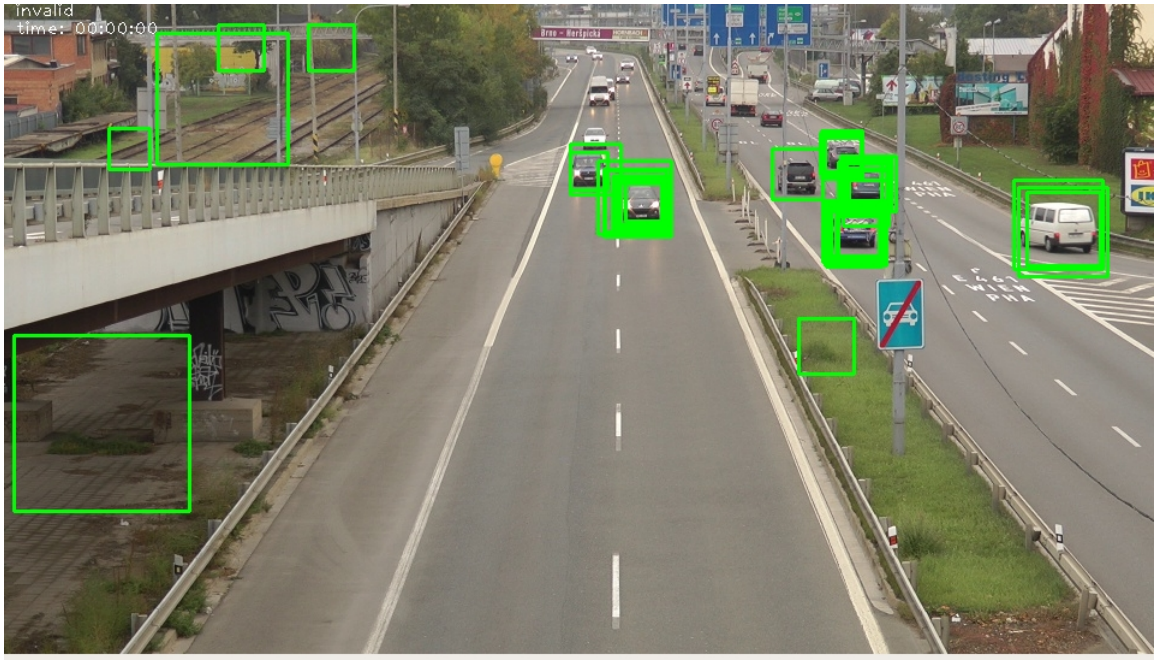
Odstranění chyb detektoru

Mezi hlavní chyby při práci detektoru, se kterými jsem se setkal, patří detekování jednoho automobilu v rámci jednoho snímku vícekrát (duplicity), detekování jiných objektů, než vozidel (false positive) a naopak nedetekování automobilu v aktuálním snímku.

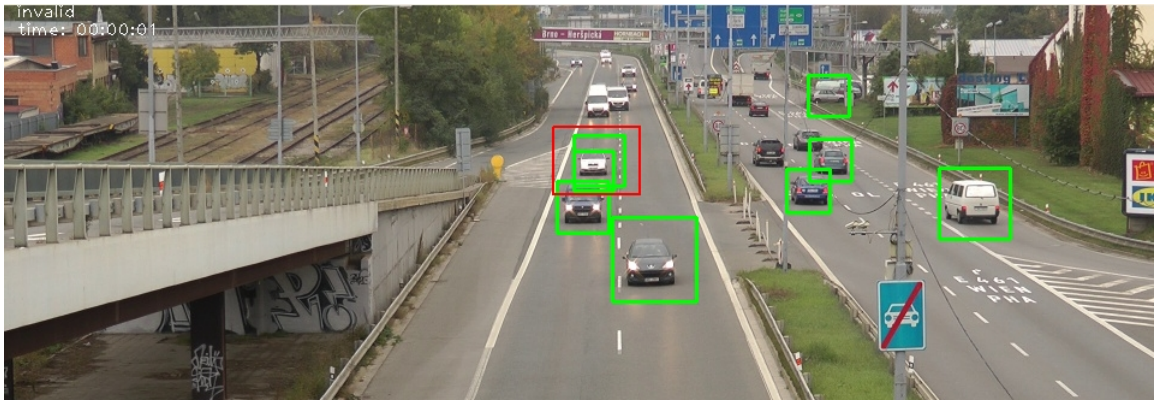
Problém s detekováním jiných objektů než vozidel (obrázek 3.6) je řešen tak, že pokud se detekce objevila nahodile pouze v jednom snímku a v dalších snímcích už ne, je tato detekce vypuštěna a dále se s ní nepracuje. Pokud je ovšem parametr detektoru konkrétně *min_neighbors* nastaven na vyšší hodnotu (v tomto systému defaultně na hodnotu 4), pak se tyto špatné detekce prakticky neobjevují.

Vícenásobná detekce jednoho automobilu na jednom snímku se tedy dá vyřešit správným nastavením parametru *min_neighbors*. I přes správné nastavení tohoto parametru ovšem nastávala situace, kdy jeden automobil byl detekován dvakrát. Tato situace se sice během testování na fakultních videích neobjevovala často, ale i tak bylo nutné tuto chybu odstranit, zejména kvůli správné funkčnosti sledování vozidel pomocí metody Kalmanovův filtr. Pro získání screenu s touto chybou byla snížena hodnota tohoto parametru tak, aby se tato chyba vyskytla častěji a nebyl problém získat její screen pro ilustraci problému. Tuto chybu (obrázek 3.7) je možné odstranit tak, že po získání seznamu detekovaných vozidel v jednom snímku systém tento seznam projde znovu a v případě, že se zde nachází podezřele stejné hodnoty souřadnic, je jedna z těchto duplicitních detekcí ze seznamu odstraněna. Pokud by dvojitá detekce nebyla odstraněna, byla by jednomu automobilu přidělena dvě identifikační čísla a automobil by byl potom i dvakrát sledován a klasifikován, což je nežádoucí stav.

[1] http://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html



Obrázek 3.6: Ukázka špatné funkčnosti detektoru, kdy byl nastaven parametr *min_neighbors* na hodnotu 0. Na tomto obrázku je dobře vidět, že při špatně nastavené hodnotě tohoto parametru detektor detekuje i objekty, které nejsou vozidla a zároveň detekuje jednotlivá vozidla v rámci jednoho snímku vícekrát.

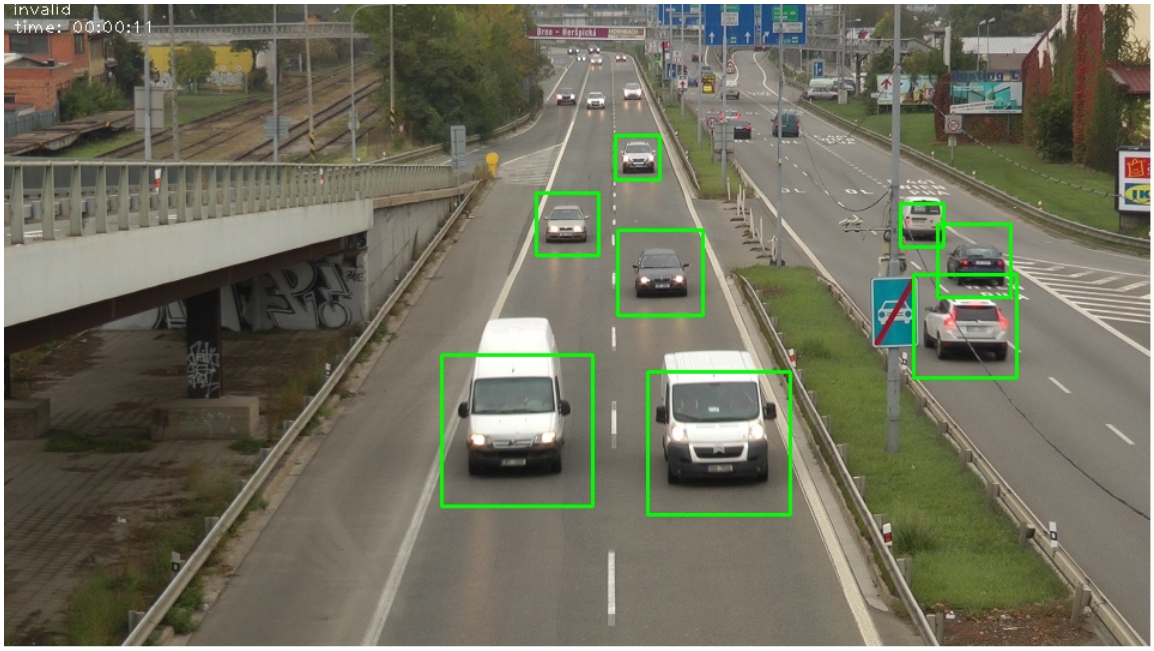


Obrázek 3.7: Ukázka dvojité detekce, kde červený obdélník označuje šedý automobil (Škoda Fabia), který byl detekován dvakrát v rámci jednoho snímku videa.

Poslední problém, kterým je nedetekování vozidla v aktuálním snímku, vyřeší *node* pro sledování automobilů. Pokud chybí aktuální informace (bounding box) ke konkrétnímu automobilu, který už je sledován, tento *node* polohu dopočítá. Činnost sledování automobilů je podrobněji popsána v následující podsekci o Sledování automobilů.

Po odstranění chyb, které byly popsány v předchozích odstavcích, se dá předpokládat správná činnost detektoru. Správně pracující detektor je znázorněn na snímku 3.8.

K nedetekování vozidla může dojít i jinak, než chybou detektoru. Nejčastější příčinou bývá zastínění hledaného objektu, tedy v případě, že mezi objektem a kamerou je další předmět. Může to být jiné vozidlo, sloup nebo například zábradlí. Tato chyba vzniká tím,



Obrázek 3.8: Ukázka správné funkčnosti detektoru, kde je každý automobil detekován pouze jednou a nenachází se zde ani žádné nahodilé detekce.

že detektor neumí detekovat částečně zastíněné automobily. O tom, které jízdní pruhy je vhodné sledovat a následně analyzovat s vysokou přesností je věnována sekce na konci této kapitoly.

Detektor detekuje automobily velmi rychle, protože pro zpracování využívá více jader a masku vstupního videa. Protože je detekce velmi rychlá a *node* pro čtení videa při takové rychlosti nestíhá ukládat všechny snímky videa, je po spuštění systému možné využít synchronizaci těchto dvou *nodů*. Z prvních detekcí se vypočte průměrná doba detekce a pomocí tohoto údaje jsou tyto dva *nody* synchronizovány. Při využití synchronizace je sice rychlost běhu systému pomalejší, ale trajektorie sledovaných automobilů jsou plynulejší a přesnější z důvodu většího počtu zpracovaných detekcí.

Úspěšnost detekce automobilů pomocí kaskády se pohybuje pod hranicí 92 %, což je podrobněji popsáno v kapitole 4. Detektor ale není závislý pouze na jediné kaskádě, proto lze bez problémů načíst do systému jinou kaskádu a používat ji.

3.7 Sledování automobilů

Dalším blokem tohoto systému je blok pro sledování automobilů. Pokud by se používal pouze detektor, program by při příchodu nových snímků neuměl rozlišit, který automobil už byl detekován v předchozích snímcích a který se objevil poprvé až v aktuálním snímku. A protože detektor detekuje jednotlivé automobily během jejich pohybu v zorném poli kamery vícekrát, při každé další detekci by určitý automobil dostal nové ID, což by zkreslovalo výsledky prováděných analýz. Proto se ve spolupráci s detektorem využívá tracker pro sledování objektů, v tomto případě vozidel. Tracker spolehlivě určí, který automobil už byl v předešlých snímcích identifikovaný a který automobil se objevil poprvé až v aktuálním snímku. Při použití bloku pro sledování automobilů jsou všechny projíždějící automobily

identifikované právě jednou a tím pádem každý automobil, který projel sledovaným prostorem, má mít přiřazené právě jedno ID, i když byl detektorem detekován na více snímcích.

Pro sledování automobilů jsem jako vhodnou metodu vyhodnotil Kalmanův filtr [29]. Kalmanův filtr byl zvolen proto, že není výpočetně náročný a navíc je tato metoda podporovaná výše zmíněnou knihovnou *OpenCV*.

Princip bloku pro sledování automobilů

Bouding boxy detekovaných automobilů jsou postupně ukládány na *topic*, ze kterého se tyto informace dostanou na vstup *nodu* pro sledování vozidel. Ten přijatou zprávu zkopíruje na výstupní *topic* s tím, že k jednotlivým bounding boxům a výřezům automobilů přiřadí ID. Přiřazením ID tedy spojuje více detekcí z jednotlivých předchozích snímků a tím sleduje jednotlivé automobily a vytváří trajektorie jejich pohybu.

Node, pomocí kterého je metoda Kalmanův filtr implementována, během své činnosti udržuje informace o aktuálních automobilech na scéně. Při příchodu nových detekcí z aktuálního snímku videa porovná odhady poloh jednotlivých vozidel, které vypočítal, s právě příchozími detekcemi a pomocí *Hungarian algoritmu* [11], který slouží k párování na základě nejnižší ceny, přiřadí tyto příchozí detekce k jednotlivým automobilům. Pokud by cena byla příliš vysoká (nad stanovený práh), znamená to, že tato detekce je nový automobil, který ještě není sledován. Tracker s danou detekcí nakládá jako s prvním výskytem nového automobilu a vytvoří nový Kalmanův filtr, který bude toto nově detekované vozidlo po dobu jeho průjezdu sledovaným úsekem sledovat. Po tom, co jsou všechny detekce správně přiřazeny, tracker vypočítá pomocí existujících Kalmanových filtrů odhady nových poloh, které budou v dalším cyklu opět porovnávány s příchozími detekcemi.

Při implementaci tohoto bloku byla upravena a rozšířena již existující implementace metody Kalmanův filtr[2] pro využití sledování vozidel. Tato implementace byla určená pro sledování náhodně se pohybujících kuliček v obraze. Dále byla implementace upravena a přizpůsobena pro správnou činnost a funkčnost na platformě ROS. Další úpravy byly aplikovány v rámci zpřesnění a odstranění chyb, které nastávaly. Řešení chyb, které se objevovaly během implementace, je popsáno níže.

Mezi parametry pro správné sledování automobilů patří:

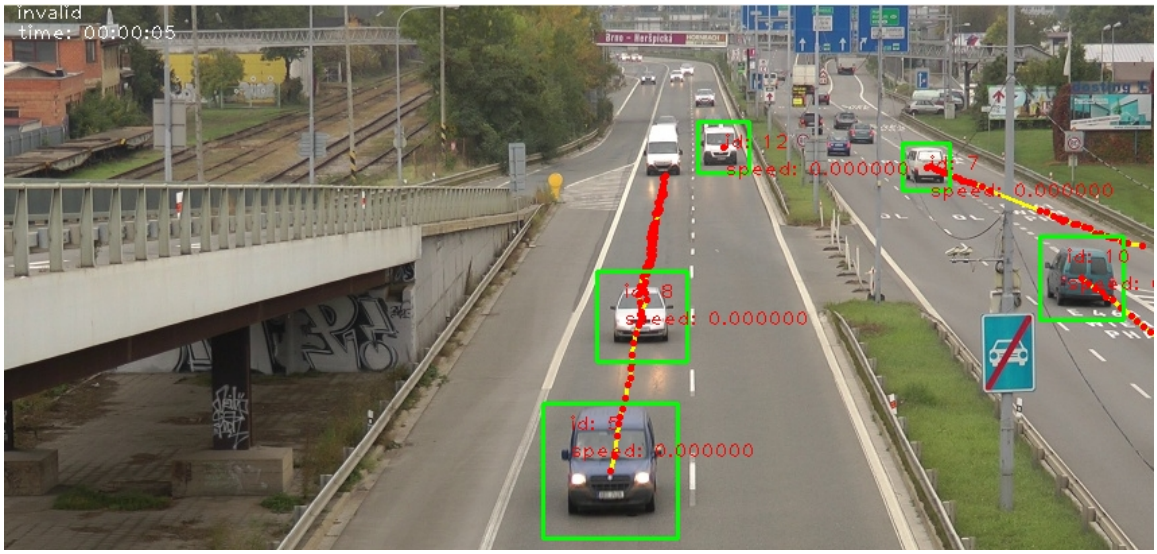
- **dt** (double) – neboli *delta_time* Kalmanova filtru,
- **accel_noise_mag** (double) – velikost akcelerace hluku Kalmanova filtru,
- **dist_thes** (double) – značí vzdálenost, o kterou se automobil může v rámci dvou po sobě jdoucích snímků posunout jakýmkoliv směrem (vzdálenost mezi detekcemi dvou po sobě jdoucích snímků),
- **max_trace_length** (int) – nastaví maximální počet snímků videa, po kterých je automaticky ukončeno sledování automobilu (maximální délka trajektorie),
- **max_allowed_skip_frame** (int) – nastaví maximální počet po sobě jdoucích snímků videa, ve kterých nemusí být automobil detekován, aby nebylo jeho sledování zrušeno.

Parametry detektoru a trackeru jsou jedny z nejdůležitějších, pomocí kterých lze systém vyladit, aby pracoval co nejpřesněji. Pomocí nich je možné nastavit co nejpřesnější detekci

[2] https://github.com/Smorodov/Multitarget-tracker

a následné sledování automobilů podle konkrétních požadavků bez nežádoucích jevů (například velké výkyvy trajektorií jednotlivých automobilů) v závislosti na testovaném videu. Protože systém podporuje více formátů vstupního videa, mohou mít spouštěná videa rozdílné rozlišení a další parametry.

Jak bylo uvedeno v podsekci o detekci, detektor nemusí vždy detekovat automobil na všech snímcích, ve kterých se vyskytuje. Jedním z parametrů, které ovlivňují práci metody Kalmanova filtru, je právě počet snímků jdoucích za sebou, na kterých nemusí být automobil detekovaný, aby ho tracker nepřestal sledovat. Tato situace může nastat například pokud je automobil na chvíli zastíněn jiným předmětem. Automobil je sice stále v obraze, ale díky stínění detektor nepozná, že se jedná o vozidlo. Zastiňovat automobily mohou například sloupy a tento případ a správná reakce trackeru na tuto situaci je na obrázku 3.9. Pokud je překročen počet po sobě jdoucích snímků, na kterých nemusí být automobil detekován, tracker automobil přestane sledovat a tím jeho Kalmanův filtr zaniká. Tato situace také nastává, když automobil odjede z prostoru snímaného kamerou, ovšem v tomto případě je konec sledování jednotlivých automobilů správným chováním. Tím je sledování ukončeno a tento automobil může být zařazen na konec fronty, ve které jsou uloženy informace potřebné pro klasifikaci jednotlivých dosud neoklasifikovaných automobilů.



Obrázek 3.9: Ukázka správné funkčnosti sledování automobilů pomocí metody Kalmanova filtru. Na obrázku je znázorněno i řešení chvilkového zastínění automobilu sloupem. Na trajektorii automobilu (ID 7) je vidět, že v okolí sloupu je pouze žlutá čára bez červených teček, které znázorňují jeho předešlé detekce.

Správné nastavení parametrů

Nastavení parametrů trackeru (popsaných v předchozích odstavcích) je důležité pro správné sledování vozidel. Nejvíce proměnný parametr je *dist_thes*, kde například záleží na rychlosti automobilů. Na dálnici mohou automobily za stejný časový úsek ujet několikrát větší vzdálenost, než automobily ve městě. Také záleží na úhlu, ze kterého kamera snímá dopravní scénu. Pokud tedy při práci se systémem nefunguje sledování vozidel a jeden automobil je označen více ID, znamená to, že vzdálenost *dist_thes* je nastavena na malou hodnotu. Tracker potom chybně vyhodnotí, že na dalším snímku se jedná o další vozidlo. Para-

metry *dt* a *accel_noise_mag* nebylo třeba oproti původní implementaci měnit. Parametr *max_trace_length* je třeba nastavit na větší číslo, na teoretickou hodnotu, která nemůže být překonána v rámci sledování automobilu, který by jel nejdelší možnou cestou. Při nastavení hodnoty parametru *max_allowed_skip_frame* je vhodné zvolit kompromis. Je vhodné, aby automobily v případě zastínění nepřestaly být sledované, dokud neodjedou ze záběru kamery. Nesmí se ale naopak zvolit moc vysoký počet vynechaných snímků, neboť by mohly jednotlivé trajektorie automobilů, které už opustily snímaný prostor, zůstávat dlouho v obraze. V takovém případě, hlavně pokud by projelo více automobilů těsně za sebou, by se vizualizace stala nepřehlednou.

Odstranění chyb trackeru

Největší chyba, kterou tracker dělal, nastávala v hustém silničním provozu, kdy automobily jezdily blízko sebe. V tomto případě docházelo k jevu, kdy byly jedním ID střídavě chybně označovány různé automobily. K této situaci dochází, protože tracker vyhledává nejbližší polohu nalezeného vozidla v následujícím snímku. Může tedy dojít k situaci, že bližší poloha, než poloha původně sledovaného automobilu v následujícím snímku, je poloha automobilu jedoucího vedle nebo za sledovaným automobilem, a proto tracker chybně spojí jednou trajektorií dva různé automobily. Tato situace je vyřešena úpravou implementace Kalmanova filtru tak, že během sledování jednotlivých automobilů systém kontroluje jejich směr v osách X a Y. Při přiřazování nových souřadnic potom tracker kontroluje, zda jsou přiřazeny souřadnice, které jsou v původním směru jízdy automobilu. Pokud nově přidělená detekce nepokračuje ve správném směru, je vyzkoušena hodnota předpokládaná Kalmanovým filtrem. Pokud předpovězená hodnota představuje pohyb automobilu správným směrem, je mu tato hodnota přiřazena. Jestliže ani tato hodnota není řešením, nebude k trajektorii v tento krok iterace přiřazen nový bod. Zároveň jsou případné nové polohy kontrolovány tak, aby původní část trajektorie a nová část trajektorie svíraly úhel okolo 180^o. Pomocí těchto kontrol nedochází k velkým výkyvům trajektorie.

Správná činnost trackeru je znázorněna na obrázku 3.9, kde je vidět, že trajektorie jsou téměř rovné a nemají žádné velké výkyvy. Zároveň je zde vidět i správné chování trackeru při zastínění vozidla překážkou, kvůli které detektor nemůže detekovat tento automobil v daném snímku. Tracker správně vyhodnotí, že automobil ještě neodjel ze zorného pole kamery a sleduje ho i nadále. Na následujících snímcích videa, kde už vozidlo není zastíněné, jsou opět vidět červené tečky, které znázorňují správné polohy bouding boxů sledovaného vozidla. Pokud se ovšem jedná o velkou překážku, je třeba zvětšit parametr *max_allowed_skip_frame*, nebo počítat s tím, že automobilu budou přiřazena dvě identifikační čísla (ID) a to v místě před překážkou a za ní.

Dále bylo naimplementováno nedetekování automobilů, které stojí na krajnici a nezasahují do aktuální dopravní situace. Tuto kontrolu podporuje využití případné masky vstupního videa. Detektor při použití masky ignoruje automobily, které by negativně ovlivnily výsledky systému.

Výsledná úspěšnost trackeru je 81.94 %, dosažené výsledky jsou podrobněji popsány v kapitole 4.

3.8 Klasifikace automobilů

Poté, co automobil odjede ze scény, je ukončeno jeho sledování a tím jsou pro tento automobil k dispozici všechny informace potřebné pro klasifikaci (ID vozidla a seznam jeho výřezů).

Klasifikace je řešena pomocí konvolučních neuronových sítí [9]. Pro jejich implementaci využívám *Keras*, což je vysokoúrovňová knihovna pro práci s těmito sítěmi. *Keras* jsem vybral z důvodu jeho jednoduché implementace a velké úspěšnosti dosahovaných výsledků. V tomto systému je pro klasifikování automobilů použit fakultní předem natrénovaný model.

Princip bloku pro klasifikaci

Ve chvíli, kdy sledované vozidlo opustí zorné pole kamery, hlavní uzel převezme informace tohoto automobilu (ID a seznam jeho výřezů) z *topicu*, na který zapisuje tracker. Následně jsou tyto informace uloženy do fronty. Protože je doba průjezdu automobilů ve videu proměnlivá, je počet snímků (výřezů automobilů) zredukován maximálně na deset tak, aby vybrané snímky reprezentovaly celou sledovanou dráhu vozidla. Zároveň jsou snímky vynechávány proto, že by bylo zbytečné klasifikovat výřezy ze dvou po sobě jdoucích snímků videa, kdy se obvykle vlastnosti výřezů téměř nezmění. Systém tedy postupně vynechává předem určený počet snímků (minimálně čtyři) ze seznamu výřezů a do redukovaného seznamu ukládá například každý pátý snímek (při minimální hodnotě vynechaných snímků). Pokud se automobil objeví v záběru kamery pouze na malou chvíli (je pořízeno méně než pět jeho výřezů), jsou pro klasifikaci použity výřezy dva a to první a poslední. Tyto defaultně nastavené hodnoty pro tento systém byly opět získány na základě experimentů popsaných v kapitole 4. Po redukci seznamu výřezů je tento seznam poslán do klasifikačního *nodu*, vždy pouze seznam pro jeden automobil, který jednotlivé výřezy postupně posílá na *node*, který obstarává klasifikaci. *Nody* pro klasifikaci komunikují pomocí mechanismu *service*. Před samotnou klasifikací je důležité každý výřez převést na konstantní velikost, kterou vyžaduje samotný klasifikátor. A protože ROS používá při komunikaci mezi *nody* barevný model BGR a klasifikátor pracuje s modelem RGB, je tedy potřeba získané výřezy převést z BGR do RGB. Klasifikátor na vstupu očekává 4D matici ve tvaru [N, width, height, M], kde **N** je počet snímků, které se posílají na vstup klasifikátoru současně, **width, height** je výška a šířka jednotlivých obrázků (klasifikátor očekává 224 x 224) a **M** je počet barevných kanálů (zde RGB). V tomto systému se vždy na vstup klasifikátoru posílají výřezy jednotlivě, tedy ve 4D matici s rozměry [1, 224, 224, 3]. Po vytvoření je tato matice poslána ke klasifikaci pomocí funkce *predict_on_batch (x)*, kde parametr x je právě tato 4D matice. Funkce *predict_on_batch ()* se nachází v knihovně *Keras*. Klasifikátor na základě natrénovaného modelu přiřadí pravděpodobnost shody ke každé třídě z modelu. Třída s největší pravděpodobností je vybrána jako vítěz a je uložena. Po skončení klasifikace všech výřezů určitého automobilu je na pole vítězných tříd aplikován modus. Nejčastější hodnota, která představuje index vítězné třídy, je poslána hlavnímu uzlu jako výsledek. Tento postup se opakuje jednotlivě pro každý automobil, který byl detekován ve vstupním videu.

Jelikož je *Keras* implementován výhradně v programovacím jazyce Python a ROS podporuje implementaci *nodů* jak v jazyce C++ tak jazyce Python, je *node* zajišťující klasifikaci implementován pomocí jazyka Python, protože při implementaci v jazyce C++ vznikají problémy. Při implementaci v jazyce Python nevznikaly žádné problémy a *TensorFlow* i *Keras* začaly velmi rychle komunikovat s ROSem a v podstatě hned mohla být zahájena implementace samotného klasifikátoru. Základní verze *TensorFlow* má podporu pouze CPU, pro rychlejší chod je třeba doinstalovat *TensorFlow* s podporou GPU.

Pro chod systému stačí *TensorFlow* s podporou CPU, ovšem pro plynulejší fungování je vhodné systém spouštět na výkonnějších počítačích s *TensorFlow* s podporou GPU.

Popis modelu pro klasifikaci

Natrénovaný model pro klasifikaci je stejně jako kaskáda pro detektor k dispozici na fakultě. Model je ve formátu H5. Použitý model je rozdělen do 106 tříd, kde každá třída reprezentuje konkrétní typy modelů jednotlivých továrních značek automobilů. Index třídy s největší hodnotou je i nejpravděpodobnějším výsledkem. Nejvyšší hodnoty se pohybují nejčastěji v intervalu 85 – 95 % a ostatní třídy mají pravděpodobnost v rozsahu desetin procent. Model je zaměřen na automobily nejčastěji se vyskytující v České republice, proto mají největší zastoupení automobily značky Škoda, Volkswagen nebo Ford, kdy u natrénovaných značek automobilů rozeznává rozdíl mezi verzemi combi, hatchback nebo sedan. Dále rozpoznává například u značek Škoda první, druhou nebo třetí generaci. Model je trénován na fakultních datasetech, proto je vhodné systém testovat na fakultních videích, kde se objevují podobné automobily. Pokud model klasifikuje pouze typy automobilů, na které je natrénovaný, dosahuje úspěšnost klasifikace nad hranici 80 %. Přesné výsledky testování jsou popsány a zhodnoceny v kapitole 4. Úspěšnost klesá při použití videí, kde se vyskytuje velké množství automobilů, na které není model natrénovaný. Může se jednat o v České republice málo rozšířené typy automobilů nebo automobily, které nejsou určeny pro český trh.

3.9 Kalibrace kamery

Vzhledem k tomu, že vzdálenosti a pak i rychlosti jsou získávány v relativních jednotkách, je třeba vypočítat měřítko mezi těmito relativními jednotkami a skutečnými veličinami, aby údaj o rychlosti mohl být uveden ve standardní veličině pro rychlost (v České republice km/h). Kalibrace kamery je potřebná pouze pro část pro měření rychlosti, při využití systému pouze pro klasifikaci automobilů potřebná není. Proto je tento blok, který zajišťuje kalibraci, v této práci popsán pouze okrajově a zjednodušeně.

Vstupem kalibrace jsou snímky videa a výstupem kalibrační informace složené ze tří úběžníků (vanishing points – VP), ohniskové vzdálenosti a informace, jestli kalibrace proběhla správně a získané informace jsou validní.

Pro výběr vhodných regionů (patřící vozidlům) si kalibrační blok udržuje model pozadí, po jehož odečtení získá oblasti patřící automobilům. Tento mechanismus je zpřesněn o bounding boxy automobilů za účelem minimalizace šumu.

Kalibrace využívá nástroj *diamond space* [6, 5], který je založen na Houghově transformaci pro mapování celého nekonečného systému $\mathcal{R}^2$ do vybraného konečného souřadného systému. Při tomto mapování je kaskádově použit paralelní souřadný systém, kde jsou obě osy rovnoběžné.

3.10 Měření rychlosti

Stejně, jako blok pro kalibraci, tak i blok pro měření rychlosti není potřebný pro systém, který pouze detekuje, sleduje a klasifikuje automobily, je popsán tedy také stručněji.

Pro měření rychlosti jsou nutné dvě informace a to ujetá vzdálenost a čas, za který automobil překonal tuto vzdálenost. Údaj o čase, za který vozidlo překoná určitou vzdálenost, lze získat z časové stopy jednotlivých snímků videa. Pro získání údaje o ujeté vzdálenosti je třeba nejprve získat informace o poloze automobilu v reálném světě, protože měření pouze ze souřadnic v obraze není možné kvůli perspektivnímu zkreslení. Proto se musí najít průsečík přímky daný dvěma body (poloha kamery a poloha roviny, která představuje

povrch vozovky). Normála této roviny je zároveň vektor směřující ke třetímu VP. Bohužel není známý druhý parametr roviny (vzdálenost od kamery). Protože kamery bývají umístěny v různých polohách vůči silnici, kterou sledují, nemusí být vždy přímé změření tohoto parametru možné. Kvůli tomuto nedostatku bude systém vracet vzdálenosti v relativních jednotkách, které jsou následně díky měřítku převedeny na klasické jednotky délky. Získání vzdálenosti roviny od kamery (a tím i zmiňovaného měřítka) je založené na porovnávání řádově desítek výřezů automobilů klasifikovaných jako Škoda Octavia se známou konvexní obálkou tohoto automobilu.

Měření rychlosti se dělí na určení:

- **okamžité rychlosti** – rychlost se počítá z posledních dvou detekcí, slouží jen pro vizualizační účely,
- **průměrné rychlosti** – všechny body se aproximují přímkou, případně parabolou, na níž se spočítá ujetá vzdálenost.

Průměrná rychlost je vypočítaná vždy poté, co daný automobil opustí zorné pole kamery. Okamžitou rychlost systém počítá vždy s příchodem nové detekce. Údaje o aktuální rychlosti jsou postupně vykreslovány do vizualizace k jednotlivým vozidlům.

3.11 Logování výsledků

Posledním celkem blokového schématu (obrázek 3.2) je logování. Tento blok jako jediný nepředstavuje samostatný *node*, ale postará se o něj hlavní uzel, který má k dispozici všechny informace o běhu systému.

Po skončení klasifikace a měření rychlosti zbývá uložit dosažené výsledky do logovacího souboru, který je ve formátu JSON. Logovací funkce jsou volány z destruktoru třídy *Car*. Formát logování pro jednotlivé automobily pro klasifikační část je: ID automobilu, čas průjezdu, značka a typ vozidla. Formát logování pro část systému pro měření rychlosti je: ID vozidla, výčet snímků, na kterých se automobil objevil, souřadnice bounding boxu (X a Y) a na konci souboru jsou uloženy kalibrační údaje získané kalibrací kamery. Všechny potřebné informace pro logování jsou získány ze třídy *Car*. Ovšem informace o výsledku klasifikace je v této třídě uložena jako číselný údaj (*integer*), proto se může toto číslo (identifikátor třídy klasifikátoru) pomocí připravené funkce převést na *string*, který reprezentuje tovární značku a model vozidla. Informace o typu automobilu je do logovacího souboru zapsána ve formě *stringu* pro lepší kontrolu výsledku klasifikace. Mimo logovacího souboru jsou do složky *logs* uloženy všechny výřezy vozidel, která se objevila v záběru vstupního videa. Snímky jednotlivých automobilů jsou roztříděné do jednotlivých složek. Složky jsou pojmenované podle ID automobilů a jednotlivé výřezy jsou vždy pojmenované číslicemi vzestupně od nuly. Výřezy se ukládají do třídy *Car*, jakmile automobil opustí zorné pole kamery, z výřezů se vyberou kandidáti na klasifikaci a ostatní výřezy jsou ze třídy smazány. Před smazáním ze třídy jsou všechny výřezy uložené na disk. Velké množství výřezů (ty, které nejsou určeny ke klasifikaci) je tedy rychle mazáno pro efektivnější práci s pamětí.

Umístění logovacího souboru je primárně nastavené do složky s cestou */data/logs*, toto umístění se ovšem může libovolně měnit. To samé platí pro uložení výřezů automobilů.

Výběr správného jízdního pruhu pro úspěšnou analýzu

Ze screenu (obrázek 3.1) je zřejmé, že systém není vždy schopen ze záběrů kamerového systému správně analyzovat celou scénu. Proto je velmi důležité vybrat na základě vlastností záběru, kterou část je systém schopen sledovat a analyzovat s velkou přesností. Během implementace a testování systému na tomto konkrétním videu byly získány tyto závěry.

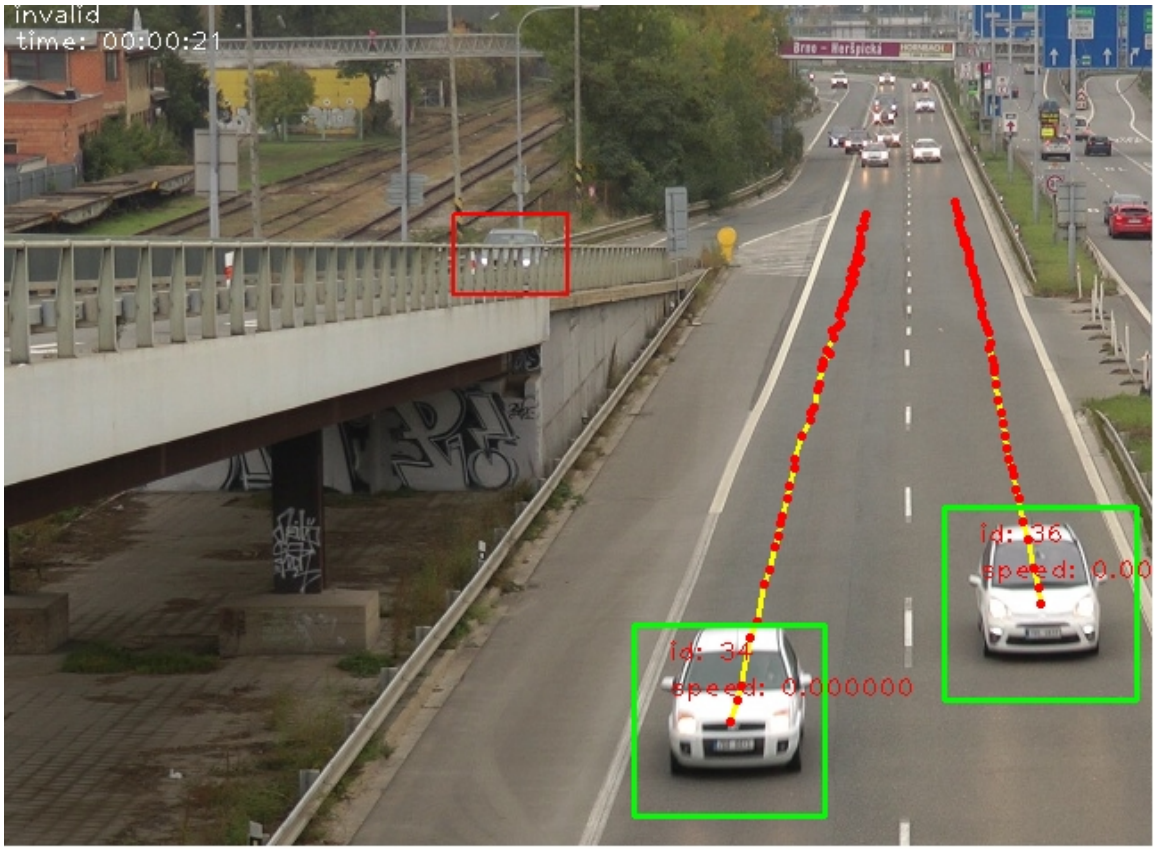
Pokud je pro sledování vybraný směr jízdy vozidel jedoucích směrem ke kameře, systém je schopný bezproblémově fungovat. V tomto směru nebrání kameře ve výhledu žádná dopravní značka, návěstí ani sloup, nebo jiná překážka. Při umístění kamery v optimální výšce se tu vzájemně nezastiňují ani automobily pohybující se jedním směrem.

Pokud je ovšem pro sledování a následnou analýzu vybrán směr opačný (pro automobily pohybující se směrem od kamery) nejsou získané výsledky tak přesné jako v předchozím případě. Mezi kamerou a projíždějícími automobily se vyskytuje více překážek v podobě sloupů, svodidel apod., které tato vozidla zastiňují na proměnlivě dlouhé časové úseky. Tento problém v případě zastínění na kratší dobu, kdy objekt není defaultně detekován maximálně deset snímků za sebou, řeší tracker (obrázek 3.9). Ale i přes to nemusí v podobných případech systém pracovat bezchybně. Navíc mohou nastat zvláštní případy stínění, které systém nedokáže eliminovat, například když se automobil pohybuje příliš pomalu, nebo pokud za sloupem zastaví v koloně a než se znovu rozjede, může být zpracované velké množství snímků videa a tracker ho potom vyhodnotí jako další automobil.

Na videu je vidět záběr vozidla v odbočovacím pruhu, je úplně vlevo na nájezdu na most (obrázek 3.10). Bohužel v tomto směru stíní kameře výhled svodidla a to znemožňuje detektoru detekovat automobily v tomto prostoru. Proto pro tento jízdní pruh systém nezískal žádné informace.

Je tedy důležité pro získaný záběr určit, který jízdní pruh lze bez problémů sledovat. Proto například u videa (obrázek 3.1) je pro bezproblémovou analýzu dopravní situace vhodné zvolit pouze jeden směr jízdy, tj. automobily pohybující se směrem ke kameře. Pokud je možné akceptovat analýzu s drobnými chybami (auto označeno více ID, atd.), mohou být použity pro analýzu i záběry automobilů pohybujících se v pravém jízdním pruhu (automobily jedoucí směrem od kamery).

Na dopravních komunikacích s více jízdními pruhy ve více směrech je ideálním řešením použít pro každý směr jízdy jednu kameru a analyzovat jednotlivé směry samostatně. Dále je také z tohoto popisu možných problémů zřejmé, jak důležité je pečlivě vybrat místo, kde bude kamera umístěna, protože pouze kvalitní záznamy je možné dále bez problémů využívat pro analýzu dopravy.



Obrázek 3.10: Ukázka silničního pruhu, ve kterém není možné analyzovat dopravu. Červený rámeček zde označuje automobil, který nemůže být detekován, protože je zakrytý svodidly.

Kapitola 4

Experimenty a vyhodnocení

Tato kapitola shrnuje všechny provedené experimenty. Nejprve bylo nutné vyladit parametry detektoru a trackeru. Následně mohl být testován systém jako celek.

Výsledný systém byl testován na setu fakultních videích *BrnoCompSpeed* [26]. Dataset se skládá ze šesti sezení, která byla pořízena na různých čtyřproudových dopravních komunikacích, kde jsou primárně sledované automobily jedoucí ve dvou pruzích směrem ke kameře. Každé sezení obsahuje tři videa, kde je kamera umístěna z levé nebo pravé strany vozovky a nebo přímo nad ní. Ukázky testovacích videí jsou znázorněny na obrázcích 4.1 – 4.6. Parametry videí jsou uvedeny v tabulce 4.1. V součtu všech videí se v tomto datasetu vyskytuje 20 865 automobilů.

Pro experimenty musela být zvolena ruční kontrola z důvodu chybějících anotací pro detekci a klasifikaci. Protože byly výsledky kontrolovány ručně a tento postup tedy byl časově náročný, byly zvoleny kratší testovací úseky videa. Experimenty vždy začínaly v náhodně zvoleném čase videa a jedná se o přibližně deset minut dlouhé testovací úseky. Tyto náhodné úseky byly zvoleny z toho důvodu, že alespoň dvě videa ze stejného sezení začínala stejnou dopravní situací. Takto byla funkčnost systému vyzkoušena na větším počtu dopravních situací.

Dostupná anotace k testovaným videím obsahovala pouze test na úspěšnost bloku pro sledování vozidel. Výsledky sledování automobilů byly testovány také ručně a potom jejich úspěšnost potom byla porovnána s anotacemi.

Tabulka 4.1: Parametry testovacích videí.

Formát	Rozlišení	FPS	Přibližná délka
avi	1920x1080	100	60 minut

Obrázek 4.1: Ukázka Session1.

Obrázek 4.2: Ukázka Session2.

Obrázek 4.3: Ukázka Session3.

Obrázek 4.4: Ukázka Session4.

Obrázek 4.5: Ukázka Session5.

Obrázek 4.6: Ukázka Session6.

4.1 Detekce automobilů

Automobily jsou detekovány pomocí kaskády detektoru. Pro detekování vozidel ze snímku videa se volá funkce *detectMultiScale ()*, kde je potřeba správně nastavit dva stěžejní parametry a to *scale_factor* a *min_neighbors*. Pro správné a co nejpřesnější detekování byla provedena řada experimentů, při kterých bylo vyzkoušeno více kombinací hodnot těchto parametrů na všech testovaných videích.

Pro testování a vyhodnocení správného nastavení parametrů detektoru byly použity tyto možnosti chování detektoru:

- **FN** (false negative) – vyjadřuje počet nedetekovaných vozidel. Za **FN** jsou považované i detekce automobilů, které jsou na rozdíl od ostatních detekovány kratší dráhu, což není vhodné pro klasifikaci. Pro klasifikaci jsou totiž potřebné snímky z celé doby průjezdu trasy a pokud je detekovaná trasa oproti skutečné například poloviční, získané výřezy jsou velmi podobné a pro klasifikaci nemají takový význam, jako snímky, které zastupují celou trasu. Za **FN** jsou dále považované detekce, kde automobil na určitý časový úsek přestává být detekován. Započítané nejsou detekce automobilů, které byly zakryty překážkou, to není chyba detektoru.

- **TP** (true positive) – vyjadřuje počet správně detekovaných automobilů. Tedy automobily jsou detekovány po celou trasu, bez jakýchkoliv delších výpadků, které by ani správně fungující tracker nezahladil do jedné trajektorie.

- **FP** (false positive) – vyjadřuje počet chybně detekovaných objektů. Jedná se o objekty, které jsou podobné automobilům, ale o automobily se nejedná.

Parametr *scale_factor* se v systémech, které byly nalezené a studované, pohyboval kolem hodnoty jedna, proto byly v experimentech použity hodnoty v intervalu 1.0 – 1.2. Při těchto hodnotách je detekce dostatečně důkladná. Větší hodnoty tohoto parametru zajistí detektoru vyšší rychlost, ale úspěšnost detektoru při detekci vozidel je v tomto případě nižší. Druhý parametr detektoru *min_neighbors* je třeba nastavit na vyšší hodnotu než je nula. Výsledek při nastavení tohoto parametru na hodnotu nula je zobrazen na obrázku 3.6, kde je na první pohled zřejmé, že takto nastavený detektor nelze použít.

Nejlepší výsledky testování detektoru jsou zobrazeny v tabulce 4.2. Z důvodu velkého množství testovaných kombinací nastavení parametrů je zde uvedena pouze tabulka s nejlepšími výsledky. Tyto výsledky byly pořízené při následujícím nastavení parametrů: $scale\_factor = 1.1$ a $min\_neighbors = 4$. Při této kombinaci dosahoval detektor na testovací sadě průměrné úspěšnosti 91.93 %. Úspěšnost detektoru snižovala zejména velká vozidla (nákladní vozidla, kamiony), proto je úspěšnost lehce proměnlivá v závislosti na počtu výskytu těchto vozidel v daném testovaném úseku videa. Ve výsledcích testů není zohledněn výskyt motocyklů, které detektor nedetekuje. Jak je vidět z tabulky, pro každé video se úspěšnost detekce mění v závislosti na konkrétních podmínkách dopravních situací v testovacích videích.

4.2 Sledování automobilů

Stejně jako detektor bylo nutné správně vyladit pomocí parametrů metodu pro sledování automobilů. Mezi nejdůležitější parametry trackeru patří *dist_thes* a *max_allowed_skip_frame* (význam těchto parametrů je popsán v kapitole 3 v sekci Sledování automobilů). Jediný proměnlivý parametr je *dist_thes*, proto není u všech testovaných videí tato hodnota stejná.

Vyladění trackeru probíhalo formou experimentů, kde byly testovány různé hodnoty parametrů až do vyladění jejich nejvhodnější kombinace. Testy spočívaly ve vyhodnocování počtu automobilů správně sledovaných celou dobu jejich pohybu v záběru kamery a počtu automobilů sledovaných špatně (jednotlivá vozidla byla sledována vícekrát nebo bylo jednou trajektorií propojeno více automobilů). V tabulce 4.3 jsou zobrazeny nejúspěšnější testy jednotlivých videí i s výslednou úspěšností.

Tabulka 4.2: Výsledky úspěšnosti detekce při nejvhodnější kombinaci parametrů: *scale_factor* = 1.1 a *min_neighbors* = 4.

video	$\sum$	FN	TP	FP	Úspěšnost
Session1_center	184	18	166	0	90.21 %
Session1_left	120	12	108	0	90.00 %
Session1_right	174	22	152	0	87.35 %
Session2_center	300	36	264	0	88.00 %
Session2_left	326	30	296	0	90.79 %
Session2_right	336	34	302	0	89.88 %
Session3_center	56	4	52	0	92.85 %
Session3_left	58	4	54	0	93.10 %
Session3_right	62	2	60	0	96.77 %
Session4_center	244	11	223	0	95.49 %
Session4_left	239	14	225	0	94.14 %
Session4_right	276	14	262	0	94.92 %
Session5_center	420	16	404	0	96.19 %
Session5_left	392	40	352	0	89,79 %
Session5_right	428	26	402	0	93.92 %
Session6_center	205	18	187	0	91.21 %
Session6_left	215	24	191	0	88.83 %
Session6_right	208	18	190	0	91.34 %

Zkratky v tabulce 4.3 mají následující význam:

- **anm** – *accel_noise_mag*,
- **dth** – *dist_thes*,
- **masf** – *max_allowed_skip_frame*,
- **mtl** – *max_trace_length*.

Z tabulky je vidět, že při slabém provozu je úspěšnost Kalmanova filtru podobná úspěšnosti detektoru, při husté dopravě tracker dosahuje horších výsledků. Detektoru složitější dopravní situace problémy nedělá. Nejčastěji dochází k přiřazování jednoho ID více vozidlům, takže úspěšnost je potom nižší, než úspěšnost detektoru. Největší problémy při sledování vozidel vznikaly u záznamů ze strany (left, right) při větší hustotě dopravy, protože občas docházelo ke spojení dvou automobilů do jednoho ID. K tomu dochází proto, že se zvětšují vzdálenosti mezi jednotlivými detekcemi sledovaného automobilu. Na těchto záznamech automobil urazí větší vzdálenost za stejný čas, než na záznamech pořízených kamerou, která snímá automobily výhradně zepředu (center). Problémy se spojením dvou vozidel do jednoho ID občas nastávají, když dva automobily jedou blízko sebe ve stejném směru jak v ose X tak v ose Y. K této chybě nedojde, pokud se sníží vzdálenost mezi jednotlivými detekcemi. Chyba, kdy byly automobily označeny více ID čísly, se vyskytovala častěji, než chyba kdy byla dvě vozidla spojena jednou trajektorií. Proto bylo vyhodnoceno jako vhodnější řešení nastavit větší vzdálenost mezi dvěma detekcemi, protože v tomto případě byl získán větší počet správně sledovaných automobilů po celou dobu jejich přítomnosti v záběru. Tímto se zvětšila celková úspěšnost metody Kalmanův filtr.

Tabulka 4.3: Úspěšnost sledování automobilů v jednotlivých videích. Úspěšnost je proměnlivá v závislosti na dopravní situaci.

Video	dt	anm	dth	masf	mtl	$\sum$	Počet správně sledovaných vozidel	Počet špatně sledovaných vozidel	Úspěšnost
Session1_center	0.2	0.1	200.0	10	2000	184	157	27	85.32 %
Session1_left	0.2	0.1	400.0	10	2000	120	98	22	81.66 %
Session1_right	0.2	0.1	400.0	10	2000	174	144	30	82.75 %
Session2_center	0.2	0.1	200.0	10	2000	300	252	48	84.00 %
Session2_left	0.2	0.1	400.0	10	2000	326	267	59	81.90 %
Session2_right	0.2	0.1	400.0	10	2000	336	266	70	79.16 %
Session3_center	0.2	0.1	200.0	10	2000	56	50	6	89.28 %
Session3_left	0.2	0.1	400.0	10	2000	58	51	7	87.93 %
Session3_right	0.2	0.1	400.0	10	2000	62	54	8	87.09 %
Session4_center	0.2	0.1	200.0	10	2000	244	214	30	87.70 %
Session4_left	0.2	0.1	400.0	10	2000	239	212	27	88.70 %
Session4_right	0.2	0.1	400.0	10	2000	276	225	51	81.52 %
Session5_center	0.2	0.1	200.0	10	2000	420	352	68	83.80 %
Session5_left	0.2	0.1	400.0	10	2000	392	311	81	79.33 %
Session5_right	0.2	0.1	400.0	10	2000	428	353	75	82.47 %
Session6_center	0.2	0.1	200.0	10	2000	205	165	40	80.48 %
Session6_left	0.2	0.1	400.0	10	2000	215	170	45	79.06 %
Session6_right	0.2	0.1	400.0	10	2000	208	169	39	81.25 %

Chyba spojení dvou automobilů jednou trajektorií vzniká i v důsledku toho, že *node traffic_ffmpeg* v rámci zrychlení běhu systému vynechává několik snímků vstupního videa. Z tohoto je patrné, že při menším počtu zpracovaných snímků je pořízeno méně detekcí a jsou větší vzdálenosti mezi jednotlivými detekcemi. Potom Kalmanův filtr musí mít zvětšené hodnoty parametru, který určuje maximální vzdálenost mezi dvěma detekcemi.

Jako nejlepší kombinace hodnot parametrů byla pro testovaná videa vyhodnocena tato: $dt = 0.2$, $accel\_noise\_mag = 0.1$, $max\_allowed\_skip\_frame = 10$, $max\_trace\_length =$ 2000 a $dist\_thes =$ se pohyboval v intervalu 200 – 400. Pro videa *center* byla použita hodnota 200.0 a pro videa *left a right* hodnota 400.0. Přesné hodnoty jednotlivých parametrů u testovaných videí jsou také uvedeny v tabulce 4.3.

Při takto nastavených parametrech dosahoval blok pro sledování automobilů v ručně provedených testech úspěšnost v průměru 83.52 %. Pomocí anotací byla získaná úspěšnost tohoto bloku při sledování všech automobilů ve všech testovacích videích 81.94 %. Tato hodnota úspěšnosti je tedy o zhruba 1.5 % nižší, než hodnota získaná ručními testy, protože anotace jsou přesnější a odhalí více chyb. Tento rozdíl vznikl tím, že pomocí anotacích jsou testována celá videa a ručními testy pouze jejich části. Rozdíl mezi úspěšnostmi byl minimální, proto lze tyto ruční experimenty uznat za validní. Úspěšnost bloku pro sledování vozidel je stanovena na 81.94 % a je přibližně o 10 % nižší, než úspěšnost detektoru.

4.3 Klasifikace automobilů

Pro klasifikaci bylo třeba určit, kolik výřezů je vhodné posílat na vstup klasifikátoru pro jednotlivé automobily. Dále bylo třeba stanovit minimální vzdálenost výřezů mezi snímky videa. Správné nastavení těchto parametrů zajistí vysokou úspěšnost klasifikátoru a zamezí zbytečně dlouhé době klasifikování jednotlivých automobilů, která by nastala v případě klasifikování všech získaných výřezů pro jednotlivá vozidla. Úkolem tedy bylo zjistit, kolik snímků a s jakým rozestupem je ideální uložit pro to, aby se klasifikovalo co nejméně obrázků při zachování vysoké úspěšnosti.

Po sérii experimentů byla určena jako vyhovující vzdálenost mezi výřezy na čtyři snímky videa, tedy každý pátý výřez se klasifikuje. Až při tomto rozestupu se začaly odlišovat pravděpodobnosti jednotlivých tříd modelu. Jako maximální počet výřezů pro každý automobil byla zvolena hodnota deset. Při klasifikování více než deseti výřezů pro jednotlivé automobily testy ve drtivé většině ukázaly, že více výřezů určení výsledné třídy, která představuje tovární značku a typ automobilu, nezmění. Pokud se automobil neobjevil na více než pěti snímcích vstupního videa, je klasifikován první a poslední výřez tohoto vozidla. Při detekování vozidla na více než padesáti snímcích (deset výřezů, kdy se klasifikuje každý pátý) je mezera mezi výřezy konstantně zvětšena tak, aby bylo klasifikováno deset výřezů a pomocí nich byla reprezentována celá dráha vozidla.

Po získání těchto hodnot mohla být otestována klasifikace na všech testovacích videích s následující úspěšností (tabulka 4.4). Pokud byly započítávány pouze automobily, na které byl použitý model natrénován, dosahovala úspěšnost 80.09 %. Tato hodnota byla uvedena u natrénovaného modelu a testy, které byly provedeny pouze na automobilech na které je model natrénován, tuto hodnotu potvrdily. Při testování se ale ve videích vyskytují i automobily, na které není model natrénován. V případě většího výskytu těchto vozidel úspěšnost výrazně klesla pod zmiňovaných 80 %. Do testů nebyla započítávána nákladní vozidla nebo kamiony, úspěšnost je počítána pouze na osobní automobily. Natrénovaný model pro klasifikaci automobilů tedy v průběhu experimentů na testovací sadě přiřadil v průměru s 63.72% úspěšností správnou tovární značku a typ vozidla.

V dalších testech byl model testován pouze na tovární značku Škoda, kde model automobilům přiřazoval správnou třídu téměř bezchybně.

Výsledky testů druhé části systému

V této sekci jsou velmi stručně zmíněny výsledky experimentů části systému, která nebyla implementována a testována v rámci této práce.

Pro testování kalibrace byla využita anotace zmíněných videí. Vyhodnocení funguje tak, že se porovnávají dvojice vzdáleností naměřených v reálu s dvojicemi odpovídajících vzdáleností, které testovací skript získal promítnutím těchto bodů z roviny obrazu do 3D pomocí kalibračních informací. Celkový medián těchto odchylek činil 6.13 %.

Při testování měření rychlosti bylo zjišťováno o kolik km/h se získané hodnoty o rychlosti jednotlivých automobilů liší od skutečných hodnot uvedených v anotacích. Výsledkem je celkový medián 5.31 km/h. Při výpočtu tohoto výsledku byla vynechána tři videa, u kterých dosahoval systém nejhorších výsledků.

Tabulka 4.4: Úspěšnost klasifikace osobních automobilů v jednotlivých videích.

Video	$\sum$	Počet správně klasifikovaných automobilů	Počet špatně klasifikovaných automobilů	Úspěšnost
Session1_center	160	96	64	60.00 %
Session1_left	110	75	35	68.18 %
Session1_right	164	114	50	69.51 %
Session2_center	288	197	91	68.40 %
Session2_left	310	191	119	61.61 %
Session2_right	304	205	99	67.43 %
Session3_center	48	32	16	66.66 %
Session3_left	48	30	18	62.50 %
Session3_right	52	32	20	61.53 %
Session4_center	205	130	75	63.41 %
Session4_left	205	128	77	62.43 %
Session4_right	228	138	90	60.52 %
Session5_center	396	253	143	63.88 %
Session5_left	360	225	135	62.50 %
Session5_right	408	252	156	61.76 %
Session6_center	189	116	73	61.37 %
Session6_left	183	115	68	62.84 %
Session6_right	176	110	66	62.50 %

4.4 Shrnutí experimentů

Z tabulek je zřejmé, že v podobných časových úsecích je rozdílný počet automobilů v závislosti na hustotě provozu.

V některých videích jsou výsledky horší a těžko rekonstruovatelné vzhledem k režii operačního systému a ROSu. Je to z důvodu náročného běhu, pokud jsou aktivní všechny části systému. Problém při testování bloku pro sledování automobilů byl, že detektor v ojedinělých případech nestíhal posílat včas souřadnice bounding boxů trackeru a ten potom nestihl sledovat určitá vozidla nebo spojil jednou trajektorií více vozidel. A pokud je menší úspěšnost sledování vozidel (zejména když spojí dva automobily jedním identifikačním číslem), přirozeně tím i mírně klesá úspěšnost klasifikátoru. Implementace bloku pro sledování automobilů byla otestována i mimo ROS (C++ a OpenCV), kde jsou souřadnice bounding boxů vždy hned k dispozici trackeru a jeho úspěšnost byla průměrně o 4 % vyšší.

Z testů klasifikátoru vyplývá, že model nejlépe funguje při klasifikování vozidel tovární značky Škoda, dále s úspěšností nad 80 % klasifikuje pouze automobily, na které je model natrénovaný. S průměrnou úspěšností 63.72 % pak oklasifikoval testovací sadu videí, kde se vyskytovaly a byly do výsledků započítány automobily, na které není model natrénovaný.

Systém byl otestován na všech šesti sezení testovacích fakultních videí, kde se ukázalo, že je stabilní a dosahuje dobré úspěšnosti. Během experimentů bylo také zjištěno, že systém zvládne zpracovat padesát snímků za vteřinu. Tato rychlost je závislá na rychlosti činnosti detektoru, který běží na více vláknech. Pokud je ovšem zapnutá synchronizace

nodů pro čtení videa a pro detekci, klesá hodnota zpracovaných snímků na hodnotu mezi deset až patnáct snímků za vteřinu.

Systém se může využívat i pro videa s úplně jinými vlastnostmi. V případě většího snížení úspěšnosti systému je třeba přenastavit parametry detektoru a trackeru na hodnoty, které budou vyhovovat konkrétním videím.

Kapitola 5

Závěr

Cílem této práce bylo navrhnout, vytvořit a otestovat systém pro detekci, sledování a klasifikaci automobilů. Práce byla zahájena studiem stěžejních bodů zadání této diplomové práce, popsaných v kapitole 2. Po nastudování všech materiálů citovaných v této práci mohla začít tvorba návrhu hierarchie systému.

Pro tento typ systému, který je vhodné rozdělit do více bloků, se nabízelo využití ROSu, který má výbornou podporu škálovatelnosti bloků (*nodů*) a podporuje práci se všemi knihovnami, které byly využity při implementaci systému. Protože jsou všechny informace získané ze všech uzlů uchovávané na jednom místě, je velmi jednoduché systém rozšířit, například o měření velikosti rozestupů mezi automobily.

Implementace práce byla pomocí návrhu řešení rozdělena do pěti bloků, mezi které patří tři stěžejní bloky a to bloky pro detekci, sledování a klasifikaci automobilů. Dále byly použity bloky pro čtení videa a logování výsledků. Pro detekci byl využit kaskádový klasifikátor, který využívá natrénovanou fakultní kaskádu. Pro sledování automobilů byla zvolena metoda Kalmanův filtr, kterému při přiřazování detekcí k jednotlivým sledovaným automobilům pomáhá Hungarian algoritmus. Klasifikaci provádí konvoluční neuronová síť, která klasifikuje automobily pomocí natrénovaného modelu. Úkolem této práce bylo vlastně spojit existující metody pro detekci, sledování a klasifikaci do výsledného systému, přizpůsobit tyto metody na platformu ROS a odstranit co nejvíce nedostatků, aby výsledný systém dosahoval co nejlepších výsledků. Jednotlivé metody byly implementovány do samostatných *nodů* a pomocí mechanismů *topic* a *service* byla zajištěna mezi *nody* bezproblémová komunikace.

Výsledný systém pro klasifikaci automobilů je součástí komplexního systému, který kromě detekce, sledování a klasifikace automobilů je schopen navíc kalibrovat kameru a měřit automobilům aktuální a průměrnou rychlost. Může samozřejmě fungovat také samostatně.

V experimentech bylo ověřeno, že systém je stabilní a dosahuje dobrých výsledků. Úspěšnost detektoru se na testovacích videích pohybuje v průměru kolem 91.93 % a úspěšnost trackeru 81.94 %. Natrénovaný model pro klasifikaci přiřadil v 63.72 % správnou tovární značku a typ vozidla. Systém je schopný zpracovat až padesát snímků za vteřinu.

Systém je sice primárně implementován na detekci, sledování a klasifikaci automobilů zepředu, pokud je však k dispozici kvalitní záznam, správně tyto funkce provádí i pro automobily snímané zezadu. Pokud by uživatel chtěl systém využít například pro americký trh, stačí načíst kaskádu pro detekci a model pro klasifikaci vozidel vyskytujících se na zpracovávaných video souborech. Systém tedy není závislý pouze na jedné kaskádě a jednom klasifikačním modelu, v případě budoucího použití v praxi by bylo nutné je pravidelně aktualizovat v závislosti na stále se rozšiřujícím počtu typů vozidel.

Další vývoj tohoto systému bude spočívat zejména v optimalizaci a zpřesnění systému, zvláště pak v ladění trackeru při hustém provozu, aby se jeho úspěšnost přiblížila úspěšnosti detektoru. Také je cílem natrénovat model, pomocí kterého bude klasifikátor schopen správně klasifikovat více typů automobilů, čímž by vzrostla úspěšnost klasifikace.

Autoři se s tímto systémem zúčastnili Studentské konference inovací, technologií a vědy v IT Excel@FIT 2017.

Literatura

[1] Abadi, M.; Barham, P.; Chen, J.; aj.: TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[3] Cheng, Y.; Liu, Q.; Zhao, C.; aj.: Design and Implementation of Mediaplayer Based on FFmpeg. *Software Engineering and Knowledge Engineering: Theory and Practice*, 2012: s. 867–874.

[4] Comaniciu, D.; Ramesh, V.; Meer, P.: Kernel-Based Object Tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, ročník 25, č. 5, May 2003: s. 564–577.

[5] 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.

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

[7] Emami, H.; Fathi, M.; Raahemifar, K.: Real Time Vehicle Make and Model Recognition Based on Hierarchical Classification. *International Journal of Machine Learning and Computing*, ročník 4, č. 2, April 2014: s. 142–145, ISSN 2010-3700.

[8] Juránek, R.; Herout, A.; Dubská, M.; aj.: Real-Time Pose Estimation Piggybacked on Object Detection. In *ICCV*, 2015.

[9] Karpathy, A.: Convolutional Neural Networks for Visual Recognition. *Unversity Lecture*, 2016.

[10] Ketkar, N.: Introduction to Keras. In *Deep Learning with Python*, Springer, 2017, s. 95–109.

[11] Kuhn, H. W.: The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, ročník 2, č. 1–2, 1955: s. 83–97.

[12] Kůrková, V.: Fraktální geometrie. *Pokroky matematiky, fyziky a astronomie*, ročník 34, č. 5, 1989: s. 267–277.

[13] Matthias, K.; Kane, S. P.: *Docker: Up and Running*. O'Reilly Media, 2015, ISBN 978-1-491-91757-2.

[14] Mithun, N. C.; Rashid, N. U.; Rahman, S. M. M.: Detection and Classification of Vehicles From Video Using Multiple Time-Spatial Images. *IEEE Transactions on Intelligent Transportation Systems*, ročník 13, č. 3, September 2012: s. 1215–1225, ISSN 1524-9050.

[15] Parasuraman, K.; Subin, P.: SVM Based License Plate Recognition System. *International Conference on Computational Intelligence and Computing Research*, 2010.

[16] Parekh, H. S.; Thakore, D. G.; Jaliya, U. K.: A Survey on Object Detection and Tracking Methods. *International Journal of Innovative Research in Computer and Communication Engineering*, ročník 2, č. 2, 2014: s. 2970–2979.

[17] Přibyl, P.; Svítek, M.: *Inteligentní dopravní systémy*. BEN-technická literatura, 2001.

[18] Quigley, M.; Conley, K.; Gerkey, B.; aj.: ROS: an Open-Source Robot Operating System. In *ICRA Workshop on Open Source Software*, ročník 3, Kobe, 2009, str. 5.

[19] Ragland, K.; Tharcis, P.: Survey on Object Detection, Classification and Tracking Methods. *International Journal of Engineering Research and Technology (IJERT)*, ročník 3, č. 11, November 2014: s. 622–628.

[20] Savić, N.; Junghans, M.; Krstić, M.: Traffic Data Collection Using Tire Pressure Monitoring System. In *International Conference on Transport Systems Telematics*, Springer, 2014, s. 19–28.

[21] Shantaiya, S.; Verma, K.; Mehta, K.: Survey on Approaches of Object Detection. *International Journal of Computer Applications*, ročník 65, č. 18, March 2013: s. 14–20.

[22] Sivaraman, S.; Trivedi, M.: Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis. *IEEE Transactions on Intelligent Transportation Systems*, ročník 14, č. 4, December 2013: s. 1773–1795, ISSN 1524-9050.

[23] Sivaraman, S.; Trivedi, M.: A Review of Recent Developments in Vision-Based Vehicle Detection. *Intelligent Vehicles Symposiu (IV)*, ročník 4, June 2013: s. 310–315, ISSN 1931-0587.

[24] Skolnik, M. I.: Introduction to Radar. *Radar Handbook*, ročník 2, 1962.

[25] 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.

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

[27] Viola, P.; Jones, J. M.: Robust Real-Time Face Detection. *International Journal of Computer Vision*, ročník 57, č. 2, May 2014: s. 137–154, ISSN 1573-1405.

[28] Viola, P.; Jones, M.: Rapid Object Detection Using a Boosted Cascade of Simple Features. *Accepted Conference on Computer Vision and Pattern Recognition*, 2001.

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

[30] Yilmaz, A.; Javed, O.; Shah, M.: Object Tracking: A Survey. *ACM Comput. Surv.*, ročník 38, December 2006: s. 1773–1795, ISSN 0360-0300.

Přílohy

Příloha A

Obsah DVD

Přiložené DVD obsahuje technickou zprávu ve formátu PDF a zdrojové soubory ve formátu LATEX, ze kterých byla zpráva vytvořena. Dále jsou zde uloženy zdrojové soubory výsledného systému, kde složky představují jednotlivé *nody* a soubor README, ve kterém je popsáno, jak se systém spouští. DVD obsahuje také video, které prezentuje vizuální funkčnost systému.