BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2017
http://excel.fit.vutbr.cz

Systém pro analýzu dopravy

Pavel Hájek, Radek Vopálenský

Abstrakt
Cílem této práce je navrhnout a implementovat systém pro analýzu dopravy v robotickém operačním systému. Systém postupně detekuje a sleduje vozidla v obraze. Projíždějícím vozidlům je měřená aktuální rychlost a po opuštění ze scény je automobil klasifikován (značka a typ). Pro měření rychlosti je potřeba nejdříve zkalibrovat video. Tento článek popisuje návrh systému a řešení jednotlivých problémů. Tento systém je vyvíjen autory článku na základě spojení jejich diplomových prací.

Klíčová slova: doprava — detekce — sledování objektů — kalibrace — klasifikace — rychlost — Docker — ROS — FFmpeg — OpenCV

Přiložené materiály: *N/A*

1. Úvod

V dnešní době je získávání a využívání informací ze silničního provozu velmi často probírané téma. Při analýze a řešení situací v dopravě považujeme využití počítačového vidění 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.

Tento systém se dá využít pro řízení křižovatek, kdy jsou semafory řízeny podle aktuální dopravy. Systém je popsán v následujících kapitolách.

2. Existující metody pro zjištění rychlosti a klasifikaci automobilů

V této kapitole jsou popsány některé práce zabývající se měřením rychlosti nebo klasifikací automobilů.

Obecně se dá říci, že pro měření rychlosti je nejdůležitější částí těchto systémů proces získání parametrů kamery - její kalibrace. Práce se dají rozdělit do dvou skupin - používající zkalibrovanou kameru a používající nezkalibrovanou kameru. V prvním případě systém již předem zná přesnou polohu kamery - její výšku nad vozovkou, horizontální vzdálenost od středu pruhu (případně vozovky) a úhly natočení ve všech osách, ve formě například *KRT* [1]. V opačném případě systémy musí kalibrační údaje získat z videa a to buď plně automaticky [2], nebo s pomocí uživatele [3][4].

Pro klasifikaci není nutné získávat parametry kamery, jinak je ale třeba pro měření rychlosti i klasifikaci nejprve automobily ve videu detekovat a sledovat. Systémy se tedy liší ve způsobu detekce automobilů - založené na detekci příznaků [5][6][7], modelu pozadí [4][3][8] atd. - a ve způsobu trackování vozidel - kalmanův filter [8], KLT tracker [4], prosté přiřazení podle nejnižší ceny, atd.

Jednotlivé námi použité metody jsou rozebrány dále.

3. Existující systémy pro měření rychlosti automobilů

L.Grammatikopoulos et al., 2005

Systém [1] navržený pány Grammatikopoulem, Karrasem a Petsou v roce 2005 plně automaticky počítá rychlost vozidel. Počítání rychlosti je zde založeno na měření 1D vzdálenosti mezi výskyty vozidel na rektifikovaných snímcích pomocí homografní matice. Pro nalezení této matice zavedli autoři různá omezení týkající se polohy kamery.

Autoři detekují vozidla pomocí modelu pozadí a na jednotlivých snímcích detekují bloby. Během je-

jich trackování systém zároveň počítá aktuální rychlost vozidel a to pomocí ujeté vzdálenosti od posledního výskytu. Výsledek je v relativních jednotkách, ty se přepočítají na klasické jednotky srovnáním nějaké známé vzdálenosti v obraze.

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

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

I.Sina et al., 2013

Z uvedených prací vybočuje I.Sina et al., 2013 [9], kteří navrhli metodu pro počítání vozidel a odhad jejich rychlosti při špatných světelných podmínkách, zejména ve tmě. Detekci automobilů zjednodušili na detekci jejich světlometů. Centroidy těchto blobů je třeba spárovat, k tomu autoři používají kombinaci metod *area-centroid-difference* a *normalized-cross-correlation*.

Pro trackování využívají velmi jednoduchou metodu - pokud je vzdálenost mezi detekovanými automobily na následujících snímcích menší než stanovený práh, jsou tato vozidla prohlášeno za jedno. Pro zjištění rychlosti vozidla autoři navrhují dvě metody - pomocí euklidovské vzdálenosti a s využitím modelu dírkové kamery. V práci jsou diskutovány a vyhodnoceny obě.

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

D.C.Luvizon et al., 2016

Tento tým v práci *A Video-Based System for Vehicle Speed Measurement in Urban Roadways* [4] z roku 2016 používá model popředí pro detekci jedoucích automobilů, autoři navrhli několik vylepšení pro jeho získání. V popředí systém pomocí Sobelova operátoru hledá registrační značku, na které jsou nalezeny *good features to track* [10] a ty jsou trackovány KLT trackerem [11][12]. Vyhledání trackovatelných příznaků se děje pouze jednou při prvním nalezení každého automobilu.

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

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

4. Existující řešení pro klasifikaci automobilů

N. Ch. Mithun et al., 2012

V tomto článku autoři popisují detekci a klasifikaci druhů automobilů [13]. Na rozdíl od klasifikace automobilů na tovární značky a typu vozidla, tata práce klasifikuje vozidla pouze na jejich druh (osobní automobil, autobus, kamion, atd.) a počítá četnost jednotlivých kategorií, které projedou v určité části silnice, kterou zabýrá kamera. Autoři použily v systému popsaném v tomto článku Hughovu transformaci a pro klasifikování vozidel kNN klasifikátor.

H. Emami et al., 2014

Problematiku klasifikace automobilů (značka, typ) popisuje tento článek [14]. Autoři zde popisují systém, který detekuje pouze zadní část automobilů, za pomocí zjištění 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 280 obrázků obsahující automobily a dosáhl úspěšnosti 96%.

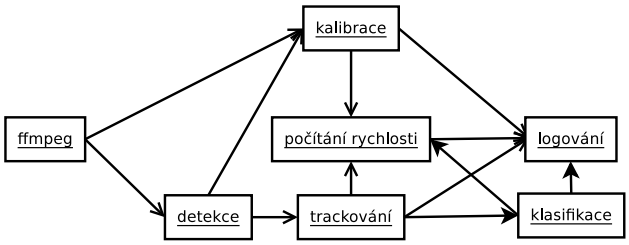
5. Návrh systému pro analýzu dopravy

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

Výsledná aplikace ze vstupu, kterým bude URL video souboru, vypíše rychlost automobilů, které musí ve streamu detekovat a sledovat. Pro tento účel je potřeba zkalibrovat kameru. Dále, vzhledem k tomu, že vzdálenosti a pak i rychlosti jsou získávány v nějakých relativních jednotkách, je třeba získat měřítko mezi těmito relativními jednotkami a

skutečnými veličinami (např. km/h). Kromě měření rychlosti aplikace klasifikuje vozidla, která projela záběrem kamery. Pomocí klasifikace je k jednotlivým automobilům přiřazena jejich tovární značka a typ.

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



Obrázek 1. Očekávané schéma bloků a jejich vzájemné komunikace

Obrázek 1 znázorňuje hierarchii jednotlivých částí systému a schéma jejich komunikace.

Celý systém je rozdělen do následujících částí - ROS balíčků:

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

5.1 Robotický operační systém

Hlavním cílem robotického operačního systému (dále jen ROS) je umožnit vývojářům rychlé a snadné vytváření modulů pro aplikace na společné platformě. ROS byl zvolen kvůli jeho podpoře škálovatelnosti aplikace na jednotlivé bloky a mechanismy, které umožňují komunikaci mezi nimi Bloky se v ROSu nazývají *nody*. Mechanismy pro komunikaci mezi *nody* jsou *topic* nebo *service*.

Tabulka 1. Způsob komunikace a sdílená data

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

5.1.1 Topic

Na *topicu* jsou uchovávána data. *Nody* mohou na zadaný *topic* buď to zapisovat nebo z něj číst potřebné informace.

5.1.2 Service

Service funguje na principu dotaz-odpověď. To znamená, že server přijme požadavek od klienta a *node* podporující požadovanou akci provede. Klient následně obdrží odpověď.

5.2 Komunikace

Systém pro komunikaci balíčků využívá mechanismy ROSu - zejména topicy. K tomuto účelu je třeba definovat několik nových datových typů. Definice těchto nových typů budou umístěny v hlavním balíčku - *traffic*.

5.3 Čtení videa

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

5.4 Detekce

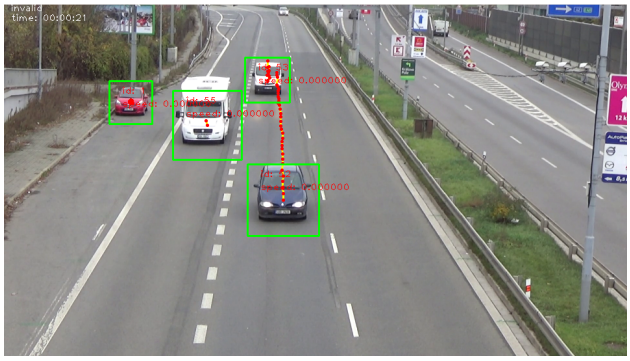
Detekce automobilů je v tomto systému založena na extrakci Haarových příznaků - využívá Viola-Jones detektor, který používá trénovací algoritmus AdaBoost, jehož cílem je sestavení kaskády dílčích klasifikátorů, které pak během rozhodování hlasují, zda-li aktuální okno obsahuje hledaný objekt - v tomto případě automobil. Pro zrychlení činnosti dílčích klasifikátorů

detektor používá tzv. integrální obraz [15]. Základním vstupem detektoru je natrénovaná kaskáda pro vyhledávaní objektů. Detektor pak na každém nově příchozím snímku vyhledá automobily a přes topic je pošle trackeru (viz dále). Kromě jejich polohy zpráva obsahuje i výřez automobilu (variabilní velikosti), který může být použit v dalších činnostech.

5.5 Tracking

Nalezené automobily (resp. středy jejich bounding boxů) jsou předány Kalmanovu filtru [16], který přijatou zprávu zkopíruje na výstupní topic s přiřazenými ID. V případě, že automobil není detekován, ale stále by se měl nacházet ve snímku, sledovací část nepošle jeho odhadovanou polohu.

Tracker si během své činnosti udržuje informace o aktuálních automobilech ve scéně, s příchodem dalších detekcí porovná odhady dalších poloh všech automobilů s právě příchozími detekcemi a pomocí *Hungarian algoritmu* [17] (sloužící k párování na základě nejnižší ceny) přiřadí tyto příchozí detekce k jednotlivým automobilům. Pokud by cena byla příliš vysoká (nad stanovený práh), tracker bude s danou detekcí nakládat jako s prvním výskytem nového automobilu. Po tom, co jsou všechny detekce správně přiřazeny, tracker vypočítá pomocí Kalmanova filtru odhady nových poloh, které budou v dalším cyklu opět porovnávány s příchozími detekcemi.



Obrázek 2. Detekované automobily na snímku formou bounding boxů (zeleně). Obsah těchto boxů je poslán na topic spolu s údaji o poloze. Červené tečky spojené žlutou čárou značí dřívější výskyty daného vozidla.

5.6 Kalibrace

Vstupem kalibrace jsou snímky videa a výstupem jsou kalibrační informace složené ze třech úběžníků (vanishing points - VP), ohniskové vzdálenosti a dále informace, zda je tato informace validní.

Pro výběr vhodných regionů (patřící automobilům) si kalibrační blok udržuje model pozadí, po jehož odečtení se získají oblasti patřící automobilům. Tento mechanismus je zpřesněn o bounding boxy automobilů, za účelem minimalizace šumu. Během implementace se ukázalo, že toto opatření zrychluje nalezení prvního VP.

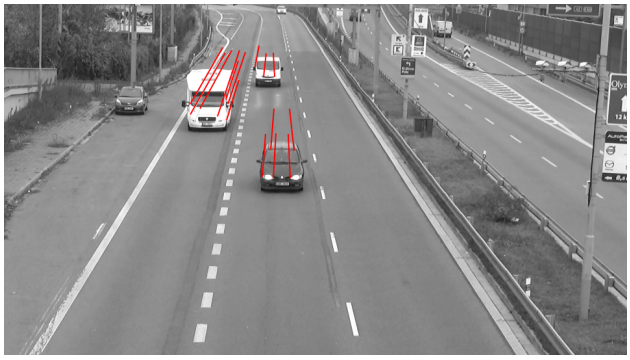
Dále je potřeba zavést podmínky, kdy lze kalibrační informace prohlásit za validní a tím v podstatě kalibraci ukončit. Ty jsou popsány v následujících kapitolách.

Kalibrace využívá nástroj *diamond space*[2][18], který je založen na Houghově transformaci, pro mapování celého nekonečného systému $\mathscr{R}^2$ do nějakého konečného souřadného systému. Při tomto mapování je kaskádově použit paralelní souřadný systém, tedy takový systém, kde jsou obě osy rovnoběžné.

5.6.1 Detekce prvního VP

Z obrazu popředí se vyberou příznaky vhodné ke trackování (good features to track [10]) (obrázek 3) a následně se pomocí KLT trackeru sledují (obrázek 3). Je třeba zajistit, aby mezi dvěma snímky byla dostatečná mezera a auta tak ujela dostatečnou vzdálenost. Snímky, které by přišly moc brzy, se jednoduše zahodí. Kalibrační část tedy pro snímky v časech t_1 a t_2 získá množinu dvojic $[x_{t_1}, y_{t_1}]$ a $[x_{t_2}, y_{t_2}]$, které tvoří přímky (obrázek 6), které se předají *diamond space*. Před předáním je vhodné odstranit zjevně špatná přiřazení - zejména s příliš krátkou nebo příliš dlouhou vzdáleností mezi určujícími body.

Obrázek 3. Nahoře snímek s vyhledanými trackovatelnými příznaky (zeleně, good features to track), dole odpovídající body na následujícím snímku vyhledané pomocí KLT trackeru (červeně)



Obrázek 4. Získané úseky přímek, které hlasují v diamond space.

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

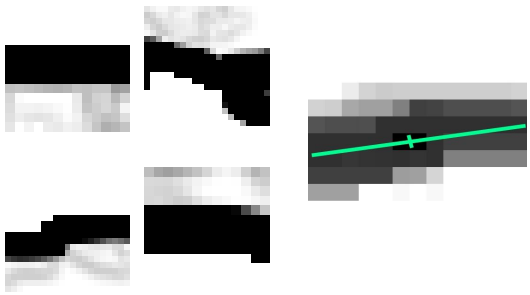
5.6.2 Detekce druhého VP

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

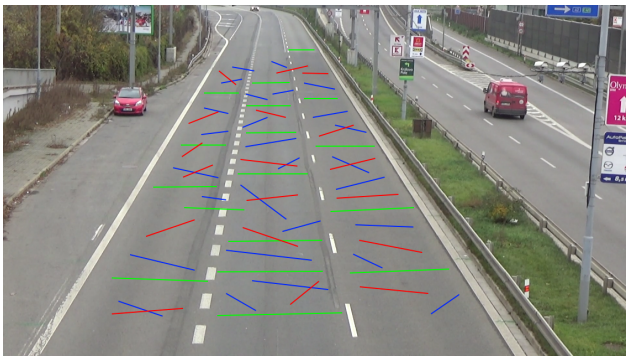
Jako velmi efektivní optimalizace se ukázalo zapojení okolí bodů do výpočtu směru hrany. Ideální velikost okolí se pohybuje okolo $9 \times 9px$, kdy příslušný bod se nachází přesně uprostřed. S nižšími hodnotami optimalizace ztrácí na přesnosti, s vyššími se pak zvyšuje riziko rušení ostatními hranami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

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

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



Obrázek 5. Vlevo několik segmentů čar, jednotlivé pixely tvoří magnitudy, středy obrázků tvoří body s vysokou magnitudou v rámci jednoho snímku. Pro každý z těchto segmentů jsou spočítány jeho vlastní hodnoty a vlastní vektory (zeleně na pravém obrázku), po nastřádání dostatečného počtu segmentů je určité procento těch s nejvyššími vlastními hodnotami vloženo do diamond space.



Obrázek 6. Vlastní vektory okolí bodů s vysokou magnitudou, které hlasují v diamond space. Červeně jsou vektory, jejichž vlastní hodnoty nejsou dostatečně velké, tudíž nebyly vloženy do diamond space. Zelené jsou ty, které již v diamond space hlasují a přibližně směřují k druhému VP. Modře jsou vyznačeny vektory hlasující v diamond space, ale nesměřují ke druhému VP.

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

5.6.3 Výpočet třetího VP a fokální vzdálenosti

Poloha třetího VP v obrazu se získá na základě rovnice 6. Pro získání reálné poloha třetího VP W' je třeba mít detekované předchozí dva VP - body U a V v následujících rovnicích.

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

$$f = \sqrt{-(U-P)(V-P)} \tag{1}$$

$$U' = [U_x, U_y, f] \tag{2}$$

$$V' = [V_x, V_y, f] \tag{3}$$

$$P' = [P_x, P_y, 0] \tag{4}$$

$$W' = (U' - P') \times (V'P') \tag{5}$$

$$W = \left[\frac{W'_x}{W'_z} f + P_x, \frac{W'_y}{W'_z} f + P_y\right] \tag{6}$$

5.7 Odhad rychlosti

Pro jakékoliv měření rychlosti jsou důležité dvě informace - ujetá vzdálenost a čas ujetí této vzdálenosti. Zatímco čas lze snadno získat z informací o snímcích, se vzdáleností je to těžší. Pro získání ujeté vzdálenosti je třeba nejprve získat polohu automobilu v reálném světě, protože měření pouze ze souřadnic v obrazu není možné kvůli perspektivnímu zkreslení. Je tedy třeba najít průsečík přímky danou dvěma body - polohou kamery a polohou v obraze - s rovinou, kterou je povrch vozovky. Normála této roviny je zároveň vektor směřující ke třetímu VP. Bohužel je neznámý druhý parametr roviny - její vzdálenost od kamery. Je zřejmé, že kamery bývají umístěny v různých polohách vůči silnici, kterou sledují, a přímé změření tohoto parametru občas nemusí být vůbec možné. Kvůli tomuto nedostatku systém vrací vzdálenosti v nějakých relativních jednotkách, které později budou díky měřítku převedeny na klasické jednotky délky. Získání vzdálenosti roviny od kamery (a tím i zmiňovaného měřítka) zatím není implementováno, nicméně bude založené na porovnávání řádově desítek výřezů automobilů klasifikovaných jako Škoda Octavia se známou konvexní obálkou tohoto automobilu.

Měření rychlosti pracuje ve dvou režimech:

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

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

5.8 Vypočítání vzdálenosti

Aplikace se snaží body v obraze promítnout na známou rovinu vozovky. Toho je dosaženo pomocí vztahu (pro vstupní bod $p = [p_x, p_y]^T$) 12, tento vztah předpokládá, že souřadnice jednoho z bodů na přímce leží v počátku, v tomto případě se jedná o kameru, proto je třeba od detekovaných souřadnic odečíst principal point P_p 10.

$$P_p = [pp_x, pp_y]^T \tag{7}$$

$$n = \frac{W}{\|W\|} \tag{8}$$

$$\rho = [n^T, \delta]^T \tag{9}$$

$$\bar{p} = [p_x - pp_x, p_y - pp_y, f]^T \tag{10}$$

$$s = -\frac{\delta}{[\bar{p}^T, 0].\rho} \tag{11}$$

$$P = s\bar{p} \tag{12}$$

5.9 Klasifikace

Klasifikaci je řešena pomocí konvolučních neuronových sítí [19]. Pro implementaci využit Keras, což je vysokoúrovňová knihovna pro práci s neuronovými sítěmi. Potom, 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). Nyní hlavní uzel převezme seznam výřezů tohoto automobilu z *topicu*, na který zapisuje tracker. Protože je doba průjezdu automobilů ve videu proměnlivá, je počet snímků zredukován defaultně na 10, kde je vypočítán počet vynechaných snímků 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 zjištěný počet snímků (minimálně 4) 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 5 jeho výřezů), jsou pro klasifikaci použity výřezy dva a to první a poslední. Po redukci seznamu výřezů je tento seznam poslán do klasifikačního *nodu* (vždy pouze seznam pro jeden automobil), který jednotlivé výřezy postupně posílá na vstup klasifikátoru pomocí *service*. 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.

Použitý model je rozdělen do 106 tříd, kde každá třída reprezentuje jednotlivé automobily. Index třídy s největší hodnotou je nejpravděpodobnější výsledek. Nejvyšší hodnoty se pohybují v intervalu 85 - 95 % a ostatní třídy mají většinou pravděpodobnost v rozsahu desetin procent. Model je zaměřen spíše na automobily nejčastěji se vyskytující v České republice, proto mají největší zastoupení automobily značky Škoda, Volkswagen nebo Ford. Model je trénován na školních data setech, proto je vhodné systém testovat na školních videích, kde se objevují podobné automobily. Pro tato videa je úspěšnost klasifikátoru pomocí tohoto modelu nejvyšší. Úspěšnost tedy klesá při videích, která obsahují málo rozšířené automobily nebo automobily, které nejsou určeny pro český trh.

6. Vyhodnocení

Systém byl testován na školních testovacích sadách. Při experimentech se systémem jsme došli k následujícím zjištěním.

Úspěšnost detektoru se na testovacích videích pohybuje v průměru kolem 88.33%.

Protože je činnost trackeru závislá na správné činnosti detektoru, pohybuje se jeho úpěšnost v podobném intervalu. Úspěšnost trackeru je tedy 85.14 %.

Natrénovaný model pro klasifikaci automobilů v průběhu experimentů přiřadil v 80.09% správnou tovární značku a typ vozidla.

Pro testování kalibrace jsme využili anotace zmíněných sad.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%, průměr pak 12%. Vliv této chyby na výsledky měření rychlosti zatím není otestován.

7. Závěr

Tento článek se zabývá systémem pro analýzu dopravy. Systém zde byl popsán hlavně z pohledu návrhu.

Pro implementaci jsou použity existující metody jednotlivých bloků v návrhu systému. Tyto metody bylo třeba převést a zprovoznit na platformě ROS. Tedy implementaci jednotlivých metod jsme předělali do samostatných *nodů*. Následně pomocí *topiců* a *service* jsme zajistili mezi *nody* komunikaci a získali požadované výsledky.

Budoucí práce na systému budou spočívat v optimalizaci a zpřesnění systému, aby byl co nejpřesnější a nejspolehlivější, a také co nejrychlejší pro real-time analýzu FullHD videí.

Literatura

[1] Lazaros Grammatikopoulos, George Karras, and Elli Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pages 332–338, 2005.

[2] M. Dubska, A. Herout, R. Juranek, and J. Sochor. Fully automatic roadside camera calibration for traffic surveillance. *Intelligent Transportation Systems, IEEE Transactions on*, 16(3):1162–1171, June 2015.

[3] Hardy Santosa Sundoro and Agus Harjoko. Vehicle counting and vehicle speed measurement based on video processing. *Journal of Theoretical and Applied Information Technology*, 84(2):233, 2016.

[4] Diogo Carbonera Luvizon, Bogdan Tomoyuki Nassu, and Rodrigo Minetto. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[5] Chi-Chen Raxle Wang and Jenn-Jier James Lien. Automatic vehicle detection using local features—a statistical approach. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):83–96, 2008.

[6] Farhad Mohamad Kazemi, Saeed Samadi, Hamid Reza Poorreza, and Mohamad-R Akbarzadeh-T. Vehicle recognition using curvelet transform and svm. In *Information Technology, 2007. ITNG'07. Fourth International Conference on*, pages 516–521. IEEE, 2007.

[7] TM Deng and B Li. A detection method of traffic parameters based on epi. *Procedia Engineering*, 29:3054–3059, 2012.

[8] Jakub Sochor. 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.

[9] I Sina, A Wibisono, A Nurhadiyatna, B Hardjono, W Jatmiko, and Petrus Mursanto. Vehicle counting and speed measurement using headlight detection. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, pages 149–154. IEEE, 2013.

[10] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[11] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[12] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. 1991.

[13] Niluthpol Chowdhury Mithun, Nafi Ur Rashid, and S. M. Mahbubur Rahman. Detection and classification of vehicles from video using multiple time-spatial images. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 13(3):1215–1225, September 2012.

[14] Hajar Emami, Mahmood Fathi, and Kaamran Raahemifar. Real time vehicle make and model recognition based on hierarchical classification. *International Journal of Machine Learning and Computing*, 4(2):142–145, April 2014.

[15] Paul Viola and J. Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2014.

[16] Greg Welch and Gary Bishop. An introduction to the kalman filter [online]. department of computer science, university of north carolina at chapel hill, 2006, July 24, 2006 [cit.2016-12-20].

[17] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[18] Markéta Dubská and Adam Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *Proceedings of BMVC 2013*, pages 1–10. The British Machine Vision Association and Society for Pattern Recognition, 2013.

[19] A Karpathy. onvolutional neural networks for visual recognition [online], 2015, [cit.2017-03-06].