VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

VIZUÁLNÍ SYSTÉM PRO DETEKCI OBSAZENOSTI PARKOVIŠTĚ POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ

VISUAL CAR-DETECTION ON THE PARKING LOTS USING DEEP NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. VÁCLAV STRÁNSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19747/2016/xstran16

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Stránský Václav, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Vizuální systém pro detekci obsazenosti parkoviště pomocí hlubokých neuronových sítí**
Visual Car-Detection on the Parking Lots Using Deep Neural Networks
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s principy detekce vozidel v obraze u statických kamerových systémů, zejména pak s metodami modelování pozadí a strojového učení (hluboké neuronové sítě).
2. Navrhněte systém pro vizuální detekci obsazenosti parkoviště pro více-kamerový systém s možností obrazového překryvu mezi kamerami. Zaměřte se na robustnost systému vůči přirozeným intenzitním změnám v obraze (mraky, stmívání, umělé osvětlení, změna počasí apod.).
3. Navržený systém implementujte a testujte na poskytnuté sadě reálných záznamů.
4. Zhodnoťte přesnost a robustnost navrženého systému. Uveďte možnosti dalšího vylepšení.

Literatura:

- KAEHLER, Adrian a Gary BRADSKI. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2015. ISBN 978-1-4919-3799-0.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

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: **Rozman Jaroslav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Počet automobilů neustále roste a jejich parkování se čím dál více komplikuje. Ve městech proto začala vznikat inteligentní parkoviště. Tato práce se zabývá návrhem a implementací robustního systému pro analýzu obsazenosti parkoviště z kamerových záznamů. Systém analyzuje jednotlivá parkovací místa ze záznamů z více-kamerového systému s možností překryvu mezi kamerami. Aplikace je navržena a implementována v Robotickém operačním systému (ROS) a její jádro se skládá ze dvou oddělených klasifikátorů. Úspěšnější, avšak pomalejší, je klasifikace pomocí hluboké neuronové sítě. Rychlou interakci řeší méně přesný klasifikátor pohybu s modelem pozadí. Systém je schopen fungovat v reálném čase, a to na grafické kartě i na procesoru. Úspěšnost systému na testovací datové sadě z reálného provozu jednoho parkoviště přesahuje 95 %.

Abstract

The concept of smart cities is inherently connected with efficient parking solutions based on the knowledge of individual parking space occupancy. The subject of this paper is the design and implementation of a robust system for analysing parking space occupancy from a multi-camera system with the possibility of visual overlap between cameras. The system is designed and implemented in Robot Operating System (ROS) and its core consists of two separate classifiers. The more successful, however, a slower option is detection by a deep neural network. A quick interaction is provided by a less accurate classifier of movement with a background model. The system is capable of working in real time on a graphic card as well as on a processor. The success rate of the system on a testing data set from real operation exceeds 95 %.

Klíčová slova

Počítačové vidění, detekce automobilů, hluboké neuronové sítě, strojové učení, GoogLeNet, model pozadí, Robotický operační systém

Keywords

Computer vision, car detection, deep neural networks, machine learning, GoogLeNet, background model, Robot Operating System

Citace

STRÁNSKÝ, Václav. *Vizuální systém pro detekci obsazenosti parkoviště pomocí hlubokých neuronových sítí.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

Vizuální systém pro detekci obsazenosti parkoviště pomocí hlubokých neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana, Ph.D. Práce má firemní zadání, a proto byla průběžně konzultována s Ing. Davidem Hermanem. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Václav Stránský
23. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Jaroslavu Rozmanovi, Ph.D. za odborné vedení práce, přínosné nápady a rychlou komunikaci. Dále děkuji Ing. Davidu Hermanovi za vedení technické stránky práce, umožnění pracovat na diplomové práci v rámci firmy RCE Systems, zprostředkování testovacích záznamů a pomoc s články pro konference *Excel@FIT 2017* a *SCCG 2017*.

Obsah

1 Úvod **3**
- 1.1 Cíle práce . . . 4
- 1.2 Obsah práce . . . 4

2 Principy detekce vozidel v obraze **6**
- 2.1 Metody modelování pozadí . . . 7
- 2.2 Metody strojového učení . . . 11
  - 2.2.1 Neuronové sítě . . . 11
  - 2.2.2 Rozdělení neuronových sítí . . . 12
  - 2.2.3 Hluboké neuronové sítě . . . 14
  - 2.2.4 Učení hluboké neuronové sítě . . . 14
  - 2.2.5 Existující dostupné architektury . . . 15
  - 2.2.6 Nástroje pro práci s neuronovými sítěmi . . . 19
- 2.3 Shrnutí . . . 20

3 Návrh systému **21**
- 3.1 Struktura a popis ROS uzlů . . . 21
  - 3.1.1 Zpracování snímků z kamery . . . 23
  - 3.1.2 Filtrace snímků . . . 23
  - 3.1.3 Transformace snímků . . . 24
  - 3.1.4 Lokální klasifikátory . . . 26
  - 3.1.5 Globální klasifikátor . . . 28
  - 3.1.6 Prezentace výsledků . . . 29
- 3.2 Klasifikace obsazenosti místa . . . 30
- 3.3 Sloučení výsledků klasifikátorů . . . 31
- 3.4 Shrnutí . . . 32

4 Implementace aplikace **33**
- 4.1 Použité softwarové nástroje . . . 33
  - 4.1.1 ROS . . . 34
  - 4.1.2 OpenCV . . . 35
  - 4.1.3 Qt . . . 36
- 4.2 Klasifikace obsazenosti místa . . . 36
  - 4.2.1 Klasifikace pomocí hluboké neuronové sítě . . . 36
  - 4.2.2 Klasifikace pomocí modelu pozadí . . . 37
- 4.3 Označení parkovacích míst . . . 38
- 4.4 Použitý hardware . . . 39
- 4.5 Shrnutí . . . 40

5 Testování **41**
5.1 Vliv deformací na úspěšnost 42
5.2 Porovnání existujících architektur 42
5.3 Vliv zmenšení vstupů na úspěšnost 43
5.4 Úspěšnost během různých podmínek 44
5.5 Časté chyby a jejich řešení 47
5.6 Shrnutí 48

6 Závěr **49**
6.1 Následující vývoj 50

Literatura **51**

A Obsah CD **55**

B Podrobný model systému **56**

C Článek pro konferenci *Excel@FIT 2017* **58**

D Poster pro konferenci *Excel@FIT 2017* **66**

E Článek pro konferenci *SCCG 2017* **68**

Kapitola 1

Úvod

Počítačové vidění se v dnešní době stalo rychle se rozvíjející oblastí informatiky, bez níž by řada aplikací nemohla existovat. Kromě průmyslového užití lidé začali počítačové vidění využívat i v soukromém životě. Výhodou se stávají mnohé volně dostupné knihovny, které sdružují základní algoritmy a funkce a jsou již v počítačovém vidění běžné. Díky rostoucí poptávce a snadnějšímu vývoji je počítačové vidění stále více podporováno a vznikají nejrůznější oblasti k jeho využití.

Jednou z oblastí, do které počítačové vidění také proniklo, se bezpochyby stalo monitorování dopravy. Existuje celá řada spolehlivých detektorů poznávacích značek, detekce a klasifikace dopravních značek, čítače provozu nebo detekce kolon. Tyto aplikace mají dvě základní využití. První z nich je získávání informací o provozu, na základě kterých může provozovatel upravit vozovku pro plynulejší provoz, měřit rychlost sledovaných aut a podobně. Druhým využitím je snaha ulehčit řidiči řízení a zpříjemnit mu tak jízdu.

S neustále rostoucím počtem automobilů vznikají nepříjemné potíže s parkováním. Ačkoliv se na parkovišti nacházejí stále volná parkovací místa, pro řidiče bývá komplikované tato místa na rozlehlém parkovišti nalézt. Řešením mohou být aplikace, které označují obsazenost jednotlivých parkovacích míst a prostřednictvím informativních tabulí či jiné vizualizační metody informují řidiče o pozici volných míst. V současné době jsou využívané systémy tvořeny senzory, které jsou založené na indukčních cívkách, mikrovlnách [15], magnetickém poli [43][45] nebo ultrazvuku [44][33]. Jelikož každé parkovací místo vyžaduje vlastní fyzický senzor, instalace systému i následná údržba je poměrně náročná. Pokud by chtěl provozovatel změnit rozložení parkovacích míst, je třeba přesunout i příslušné senzory. Tyto metody a jejich srovnání jsem blíže popsal v mé bakalářské práci [36].

V posledních letech proto začaly vznikat aplikace detekující obsazenost parkovacích míst za použití počítačového vidění. Oproti senzorovým systémům je výrazně jednodušší instalace, změna rozložení míst i údržba. Kamerový systém navíc umožňuje živý náhled na scénu a s ním spojené rozšířené využití (rozpoznání typu vozidla, barvy aj.). Úspěšností se ale zatím se senzorovými systémy nemohou měřit. Monitorování kamerami přináší i řadu nevýhod, jako jsou legislativní problémy a nutnost informovat uživatele parkoviště. Obtížnější je pak detekce za nepříznivého počasí nebo v noci při nedostatečném umělém osvětlení. Pokud cizí objekt zastíní výhled kamery nebo se kamera odkloní (např. vlivem větru), může začít docházet k chybným detekcím.

Populární metodou rozpoznávání a umělé inteligence se v posledních letech staly neuronové sítě. Metody založené na rozpoznávání pomocí neuronových sítí dosahují velmi dobrých výsledků a jsou dostatečně obecné, aby mohly řešit nejrůznější problémy. Klasické neuronové sítě ovšem pro složitější úkoly vyžadují exponenciální počet neuronů, z nichž se sítě

skládají. Tento problém řeší hluboké neuronové sítě, které se, po vyřešení problémů s trénováním, staly velmi oblíbenou a rychle se rozvíjející oblastí. Detekce obsazenosti parkoviště je jedním ze složitějších úkolů, které mohou hluboké neuronové sítě vyřešit.

Systém popisovaný v této práci je navržen a implementován v Robotickém operačním systému [30], díky němuž je rozdělen do několika oddělených uzlů. Hlavními uzly jsou zpracování snímků z kamery, filtrace snímků, transformace snímků, lokální klasifikátory, globální klasifikátor a prezentace výsledků. Snímky jsou filtrovány na základě porovnání histogramů správných a chybných snímků. Pro lepší detekci jsou ze snímku odstraněny nežádoucí transformace, radiální distorze[1] a perspektiva[2]. Předpřipravený snímek je použit pro klasifikaci obsazenosti parkovacích míst pomocí dvou klasifikátorů. Hlavní důraz je kladen na klasifikaci pomocí strojového učení, ke kterému je využita hluboká konvoluční neuronová síť. Detekci příjezdu a odjezdu doplňuje klasifikátor založený na modelu pozadí. Výsledky obou klasifikátorů jsou sloučeny a zveřejněny. Pokud systém obsahuje více kamer, jsou snímky z každé kamery zpracovávány nezávisle, výsledky dílčích klasifikátorů se sloučí všechny najednou.

Představované řešení je zajímavé svou vysokou úspěšností, jednodušší instalací a schopností klasifikovat obsazenost stovek parkovacích míst v reálném čase. Systém není bezchybný, v některých případech dochází k chybné klasifikaci. Takových případů je však méně než 5 % a s postupným vývojem aplikace a zvětšováním trénovací sady jich stále ubývá. Tato metoda nejčastěji selhává při zhoršené viditelnosti (v noci, při orosení kamery), při změnách počasí (déšť, sníh, ostré stíny) a v případech extrémní okluze, kdy je automobil téměř nebo úplně zakryt větším objektem (nákladní vůz).

1.1 Cíle práce

Práce navazuje na mou bakalářskou práci, danou problematiku však pojímá šířeji, návrh systému je robustnější a metody propracovanější. Především pak přidává rozšíření s neuronovými sítěmi. Cílem práce je seznámení se s principy detekce vozidel v obraze u statických kamerových systémů, především s metodami modelování pozadí a strojového učení. Na základě získaných informací a zkušeností bude navržen systém pro vizuální detekci obsazenosti parkoviště s více-kamerovým systémem s možností obrazového překryvu mezi kamerami. Systém by měl být co nejrobustnější vůči přirozeným změnám v obraze, jako je stmívání, mlha, déšť, sníh či umělé osvětlení. Výsledkem bude systém, který bude schopen označovat obsazenost parkovacích míst v reálném čase na skutečném parkovišti a zobrazovat získané výsledky.

1.2 Obsah práce

Kapitola 2 této práce popisuje existující metody detekce vozidel v obraze, zaměřuje se na metody modelování pozadí a metody strojového učení využívající neuronové sítě. Především pak popisuje hluboké neuronové sítě. Kapitola 3 je věnována návrhu systému, popisu jeho jednotlivých částí a řešení problémů spojených s těmito částmi. Druhá část této kapitoly popisuje výpočet obsazenosti parkovacího místa. Kapitola 4 přibližuje implementační detaily aplikace, softwarové a hardwarové nástroje, které se v rámci této práce využívají,

[1] Při radiální distorzi (zkreslení obrazu) není příčné zvětšení po celém poli obrazu stejné, čímž je porušena geometrická podobnost předmětu a jeho obrazu.

[2] Při perspektivním zobrazení se zobrazované předměty zdánlivě zmenšují a sbíhají.

a jejich integraci. V kapitole 5 jsou představeny a popsány provedené testy a prezentovány výsledky z tohoto testování. Kapitolou 6 bude tato diplomová práce uzavřena a shrnuta.

Kapitola 2

Principy detekce vozidel v obraze

S přibývajícím počtem aplikací detekujících vozidla v obraze roste i počet metod, jak správné detekce dosáhnout. Tyto metody se vzájemně značně liší a je třeba zvolit správnou metodu pro konkrétní aplikaci. Pro výběr nejlepší metody je třeba položit si několik otázek. Jsou záznamy pořízeny statickou, nebo pohybující se kamerou? Musí aplikace detekovat nejrůznější vozidla, nebo je zaměřená na konkrétní typ? Je třeba, aby aplikace zpracovávala video v reálném čase a s omezeným výpočetním výkonem, nebo na době trvání detekce nezáleží? Po zodpovězení těchto otázek je možné přistoupit k výběru samotné metody.

Principy detekce vozidel v obraze lze rozdělit do dvou obecných kategorií podle toho, zda je znám přesný nebo alespoň přibližný vzhled detekovaného vozidla. Informace o vzhledu vozidla je ve většině případů zprostředkována pomocí trénovací sady obsahující stovky až tisíce snímků vozidel a pozadí. Použitý algoritmus se na této sadě natrénuje, při samotné detekci pak hledá natrénovaný vzor v obraze. K těmto metodám se řadí AdaBoost (Adaptive Boosting), který sdružuje jednodušší detektory do jednoho robustnějšího a úspěšnějšího [17]. Detekce může být založena na nejrůznějších příznacích, často používanými jsou Haarovy příznaky [27]. Ukázku těchto příznaků pro automobil lze vidět na obrázku 2.2. Další možností je metoda K-Nearest Neighbor (K-nejbližších sousedů), jež sdružuje k nejbližších příznaků. Metoda Support vector machines (SVM) [29] zase rozděluje prostor příznaků pomocí lineárních klasifikátorů [18]. Důležitou částí metody je jádrová transformace, jež transformuje příznaky z neseparovatelné do separovatelné podoby. Pokud jsou v obraze objekty dobře separovatelné, lze detekovat vozidla pomocí segmentace obrazu [42]. Další metody používají histogram gradientů [18] nebo kaskádový detektor [26]. Výstup kaskádového detektoru je znázorněn na obrázku 2.2, ze kterého je zřejmé, že je detekci stále možné zlepšovat. Metoda ale detekuje automobily na celém snímku, nikoliv jen v označených výřezech parkovacích míst. V neposlední řadě do této skupiny patří již zmiňované neuronové sítě [9].

V případě, kdy vzhled detekovaného vozidla není znám, je příliš složitý či se výrazně liší pro jednotlivé případy, mohou být použity metody z druhé kategorie. Takové metody detekují vozidlo pouze na základě stručných informací, jejich cílem je oddělit samotné vozidlo od okolí. Tyto metody nevyžadují předtrénování a jejich úspěšnost je závislá na nastavení uživatelem. Jsou založené na informaci, která rozlišuje vozidlo od okolí. Pokud je známo, že se na vozidle vyskytuje větší množství výrazných hran než v okolí, je možné použít jeden z hranových detektorů [2]. Takové detektory mají výhodu, že jsou automobily detekovatelné z jediného snímku. Jejich úspěšnost však velmi závisí na okolí vozidla. Další metody jsou založené na informaci o pohybu automobilů. Nejjednodušším způsobem detekce pohyblivých objektů, a tedy i automobilů, je rozdílová metoda [40]. Funguje na jednoduchém odečtení

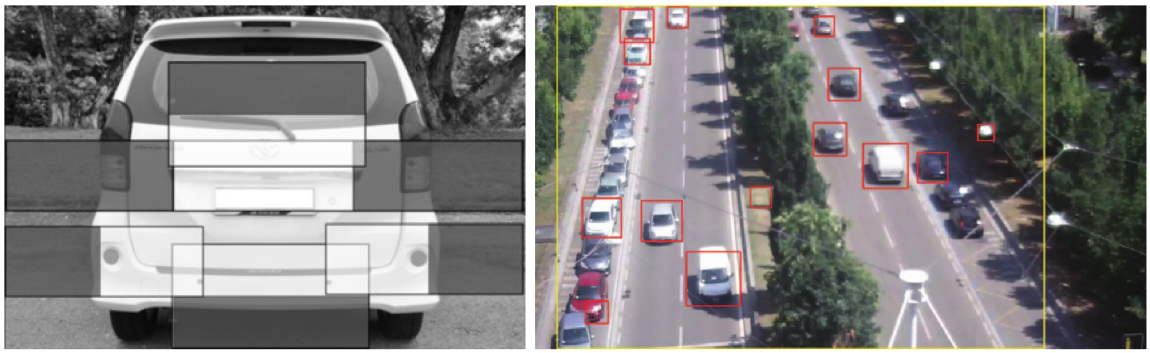
po sobě jdoucích snímků a její úspěšnost není příliš vysoká. Sofistikovanější metody používají model pozadí [26] nebo optický tok [40]. Optický tok hledá pozice stejných bodů na dvou různých snímcích a na základě rozdílů jejich pozic určuje pohyblivé objekty. Metody s modelem pozadí si vytvářejí model, který odpovídá pozadí bez pohybujících se objektů. S tímto pozadím jsou nové snímky srovnávány, detekované rozdíly odpovídají pohybujícím se objektům. Metodám s modelem pozadí se tato práce věnuje podrobněji.

Porovnání úspěšnosti jmenovaných metod je komplikované, neboť každý autor přístupu využil jinou testovací sadu. Autory udávaná úspěšnost těchto metod se pohybuje mezi 85 % a 95 %, datové sady ale většinou neobsahují okluze či zhoršenou viditelnost. Lepší úspěšnosti dosahují metody využívající hluboké neuronové sítě [3][22], jejichž úspěšnost přesahuje 95 %. Hlavním úskalím je rychlost klasifikace, kterou autoři uvádějí na 15 sekund na celé parkoviště s použitím grafické karty.

Obrázek 2.1: Označení parkovacích míst a automobilů v metodách s konvoluční neuronovou sítí. Převzato z [3], [22] a [9].

2.1 Metody modelování pozadí

Metody založené na modelování pozadí jsou velmi rozšířené především pro detekci pohybujících se objektů ve videu ze statické kamery [11]. Přitom podmínka statické kamery je zde nezbytně nutná. Pokud není k dispozici video ze statické kamery, video je možné stabilizo-



Obrázek 2.2: Ukázka Haarových příznaků detekujících automobil zezadu, převzato z [27], a výstup kaskádového detektoru automobilů, převzato z [26].

vat, většinou ale takovýto přístup zhoršuje výsledky detekce. Úspěšnost detekce pak záleží na míře pohybu kamery a změnách mezi jednotlivými snímky. Princip těchto metod spočívá v detekci pohybujících se objektů z rozdílu mezi aktuálním snímkem a snímkem referenčním, často nazývaným jako model pozadí (background model nebo background image) [28]. Model pozadí musí reprezentovat scénu bez jakýchkoliv pohybujících se objektů a musí být pravidelně aktualizován, aby se přizpůsobil měnícím se podmínkám [4]. Právě na tvorbě co nejpřesnějšího modelu pozadí závisí úspěšnost popisovaných metod. Pro jejich porovnání se hodnotí rychlost výpočtu, paměťové nároky a přesnost detekce. Dále jsou popsány základní metody modelování pozadí seřazené od jednodušších a rychlejších po složitější a přesnější. Ukázky výstupů vybraných metod lze vidět na obrázku 2.3.

Gaussův klouzavý průměr

Jedna z nejjednodušších metod počítá hodnotu každého pixelu nezávisle [39]. Hodnota každého pixelu modelu je spočítána z pixelů na stejných souřadnicích z posledních n snímků pomocí Gaussovy pravděpodobnostní funkce:

$$u_t = \alpha I_t + (1 - \alpha) u_{t-1} \tag{2.1}$$

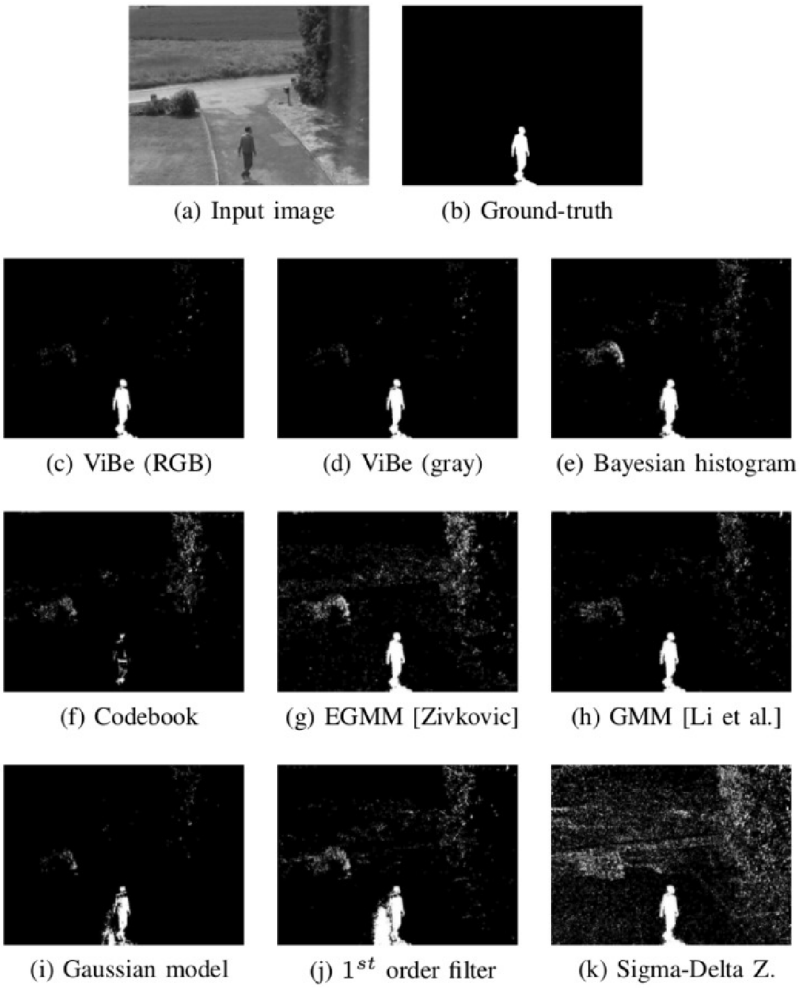
kde I_t je hodnota pixelu na aktuálním snímku, u_{t-1} předchozí průměr a α váha vyvažující stabilitu modelu a rychlou změnu. Tato úprava snižuje paměťové nároky, neboť není třeba vytvářet zásobník s posledními n hodnotami. Porovnáním rozdílu aktuální hodnoty pixelu s hodnotou pixelu v modelu s prahem lze klasifikovat pixel jako popředí nebo pozadí. Lepších výsledků je možné dosáhnout úpravou rovnice 2.1 o binární indikátor popředí M:

$$u_t = M u_{t-1} + (1 - M)(\alpha I_t + (1 - \alpha) u_{t-1}) \tag{2.2}$$

Model pozadí je aktualizován pouze v případě, že se jedná o pozadí (M=0). Pokud klasifikátor označí pixel jako popředí (M=1), do modelu je zkopírována předchozí hodnota.

Časový mediánový filtr

Dle různých autorů dosahují jiné formy průměru lepších výsledků. Jednou z nich je použití střední hodnoty z posledních n snímků jako hodnoty modelu pozadí [10]. Metoda je stabilnější, hlavní nevýhodou této metody je ale nutnost zásobníku pro nedávné hodnoty. Mediánový filtr také neobsahuje míru odchylky pro úpravu prahu.



Obrázek 2.3: Příklady metod modelujících pozadí, převzato z [4].

Směs Gaussových rozdělení

Ve většině případů jsou objekty, které se nacházejí delší dobu na stejné pozici, zahrnuty do modelu pozadí a pohybující se objekty klasifikovány jako popředí. V některých případech ale může docházet ke změnám v rámci pozadí rychleji, než se model stihne aktualizovat. Příkladem mohou být větve a listy stromů ve větru, déšť nebo sníh. V těchto případech má pozadí pro daný pixel více možných hodnot (např.: list, větev nebo stromem částečně zakrytá budova). Doposud popsané modely nejsou schopny tyto jevy zahrnout, proto existují modely s více hodnotami [35]. Pravděpodobnost hodnoty pixelu je možné vyjádřit směsí Gaussových rozložení:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t - \mu_{i,t}, \Sigma_{i,t}) \tag{2.3}$$

kde každé Gaussovo rozdělení odpovídá právě jednomu objektu v pozadí či popředí. Počet rozložení K je obvykle nastaven na 3 až 5 [28]. Pokud se nalezne shoda mezi novou hodnotou a známým rozložením, parametry tohoto rozložení jsou upraveny. Pokud nová hodnota neodpovídá žádnému známému rozložení, model s nejnižším ohodnocením je nahrazen za nový model s velmi nízkou váhou a velkou odchylkou.

Odhad hustoty jádra

Pravděpodobnostní funkce výskytu hodnot modelu může být aproximována pomocí histogramu [16]. Jelikož je ale počet vzorků limitován, histogram není přesný a může docházet k chybným detekcím. Histogram je možné nahradit odhadem hustoty jádra (Kernel Density Estimation – KDE), který je na rozdíl od histogramu plynulý. Model pozadí se pak získá jako suma Gaussových jader, která jsou vycentrována na nejčastější hodnoty x_i:

$$P(X_t) = \frac{1}{n} \sum_{i=1}^{n} \eta(x_t - x_i, \Sigma_t) \tag{2.4}$$

Tento model připomíná směs Gaussových rozdělení. V té ale každé Gaussovo rozdělení popisovalo hlavní model a bylo aktualizováno v čase. V případě odhadu hustoty jádra každé Gaussovo rozdělení popisuje pouze jeden vzorek a Σ_t je stejná pro všechna jádra.

Postupná aproximace hustoty jádra

Techniky založené na mean shift vektoru jsou pro segmentaci obrazu velmi úspěšné [28]. Jedná se o efektivní metody, které vyžadují minimální počáteční předpoklady. Nicméně je jejich běh vysoce náročný na čas, neboť jsou založeny na iterativním výpočtu. Proto není možné použít je pro tvorbu modelu pozadí na úrovni pixelů. Jednou z možností optimalizace tohoto algoritmu je použití mean shift vektoru pouze pro inicializaci modelu. Aktualizace modelu pozadí v reálném čase je pak realizována prostřednictvím jednoduché heuristiky.

Souběžný výskyt odchylek obrazu

Další metoda je založená na předpokladu, že se sousedící bloky pixelů pozadí mění v čase podobným způsobem [32]. Příkladem mohou být pohybující se listy na stromě. K chybám dochází na hranách objektů, neboť se pixely příslušící jiným objektům mění odlišným způsobem. Metoda nepracuje na úrovni jednotlivých pixelů, nýbrž zpracovává celé bloky $N \times N$ pixelů. Díky tomu je metoda rychlejší a stabilnější. Během učící fáze je ze sady vzorků spočten průměr, rozdíly mezi průměrem a samotnými vzorky se nazývají odchylky obrazu. S ohledem na průměr je spočítána $N^2 \times N^2$ kovarianční matice a aplikací transformace vlastním vektorem jsou dimenze obrazu redukovány z N^2 na K. Pro aktualizaci modelu je tuto fázi možné v časových intervalech opakovat. Při klasifikaci je pro každý blok u spočtena odpovídající vlastní odchylka obrazu z_u. Je nalezeno L nejbližších sousedů k z_u a spočítána lineární interpolace mezi z_u a odchylkami sousedů. Stejné interpolační koeficienty jsou aplikovány na aktuální blok b, čímž je získán odhad z'_b. Porovnáním z_b a z'_b je možné o bloku b rozhodnout, zda se nachází na pozadí (z_b a z'_b jsou podobné) nebo na popředí (z_b je výrazně odlišné od z'_b).

Vlastní pozadí

Poslední popisovaná metoda je taktéž založená na vlastních hodnotách dekomponovaného obrázku, v tomto případě se ale obraz nerozdělí na bloky, jako v předchozí metodě, nýbrž je s ním počítáno jako s celkem [28]. Během učící se fáze je získáno n vzorků celých obrázků, spočítán jejich průměr a odchylky jednotlivých obrázků. Dále je spočítána kovarianční matice a uloženo M nejlepších vlastních vektorů. Pokaždé, když je k dispozici nový obrázek I, je spočítána odchylka I':

$$I' = \Phi(I - \mu) \tag{2.5}$$

kde Φ je matice vlastních vektorů a μ je průměrný obrázek. I' je pak znovu aplikována na obrázek:

$$I'' = \Phi^T I' + \mu \tag{2.6}$$

Jelikož je vlastní prostor dobrým modelem pro statické části scény, ale nikoliv pro malé pohybující se objekty, I'' tyto objekty obsahovat nebude. Pro získání bodů příslušejících k popředí obrázku stačí odečíst I a I'' a porovnat s prahem, tedy:

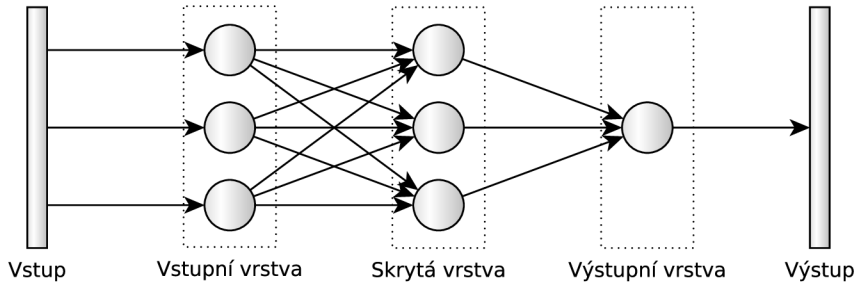
$$|I - I''| > T$$

2.2 Metody strojového učení

Strojové učení je podoblast umělé inteligence, která se zabývá algoritmy, pomocí nichž se počítačový systém může učit [25]. Učením je myšleno zlepšování vnitřního nastavení systému takovým způsobem, aby se jeho úspěšnost zvyšovala. Pro strojové učení se využívají algoritmy hlubokého učení (Deep Learning). Většinou se používá strojového učení v kombinaci s neuronovými sítěmi, které mají velký počet vrstev. Tyto sítě se nazývají Deep Neural Networks, což lze do českého jazyka přeložit jako hluboké nebo složité neuronové sítě.

2.2.1 Neuronové sítě

Každá neuronová síť se skládá z jedné vstupní vrstvy, jedné výstupní vrstvy a vrstev dalších, které se nazývají skryté, jak lze vidět na obrázku 2.4. Libovolný problém je možné vyřešit neuronovou sítí s jednou skrytou vrstvou. V takovém případě je ale zapotřebí mnoho neuronů, neboť většina funkcí vyžaduje exponenciální počet hradel. Ve většině případů tedy platí, že neuronová síť s více skrytými vrstvami dokáže abstrahovat problém do větších detailů a vyřešit ho v kratším čase.



Obrázek 2.4: Příklad dopředné neuronové sítě

Komplikací při používání neuronových sítí je jejich velká paměťová a výkonnostní náročnost. Trénování i testování sítí lze však snadno paralelizovat, proto se sítě v hojné míře trénují i testují na grafických kartách.

Jak již bylo řečeno, neuronové sítě se skládají z vrstev. Každá vrstva obsahuje určitý počet neuronů, základních bloků, které jsou vzájemně propojeny. Neuron má vždy jeden výstup a libovolný počet vstupů, ke kterým jsou vázány váhy. Tyto váhy násobí vstupy neuronů, jejich změnou je možné ovlivňovat výpočet sítě, čímž dochází k učení sítě. Kromě vah se při trénování sítě mění bias [20]. Jedná se o číslo, jež se přičítá k váhovému součtu vstupů. Váhový součet společně s biasem slouží jako vstup pro aktivační funkci [20], která aktivuje neuron. Aktivační funkce může být jednoduchá skoková funkce, dosahující lineární

aproximace, nebo nelineární funkce, s níž lze dosáhnout univerzální aproximace [13]. Často využívanou nelineární aktivační funkcí je sigmoida:

$$y = \frac{1}{e^{-x} + 1}, \tag{2.7}$$

která normalizuje výstup na intervalu $< 0; 1 >$. Na interval $< -1; 1 >$ normalizuje funkce hyperbolický tangens:

$$\tanh x = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.8}$$

O počátcích vzniku umělých neuronových sítí, jejich inspiraci a historii lze více zjistit například v knize *An Introduction to Neural Networks* [20].

2.2.2 Rozdělení neuronových sítí

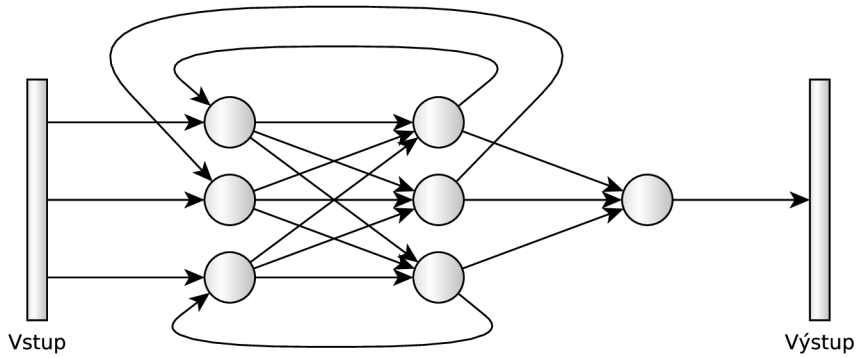
Neuronové sítě je možné dělit na základě nejrůznějších kritérií, zde jsou uvedeny tři nejčastější. Prvním z nich je rozdělení podle použité architektury. Druhé rozlišuje sítě podle způsobu, jakým jsou jednotlivé vrstvy propojeny. Třetí způsob se zaměřuje na způsob trénování neuronových sítí.

Podle architektury

Jedno z možných dělení neuronových sítí porovnává použitou architekturu. Existuje mnoho různých architektur, z nichž nejpoužívanější a nejjednodušší je dopředná neuronová síť (obrázek 2.4), další potom rekurentní síť.

Dopředné neuronové sítě: Neurony z jedné vrstvy mohou v dopředné neuronové síti záviset pouze na neuronech z vrstvy předchozí, v síti se proto nevyskytují žádné smyčky. Vyhodnocení takové sítě probíhá postupným výpočtem výstupů všech neuronů v každé vrstvě, první skrytou počínaje. Výstupem sítě je výstup poslední vrstvy.

Rekurentní neuronové sítě: Rekurentní neuronová síť je opět tvořena vrstvami neuronů. V případě rekurentní neuronové sítě však mohou vrstvy záviset i na vrstvách následujících, čímž v síti vznikají smyčky. Vyhodnocení rekurentní sítě je provedeno s opožděným vyhodnocením neuronů [7]. Příklad rekurentní sítě lze vidět na obrázku 2.5.



Obrázek 2.5: Příklad rekurentní neuronové sítě

Podle propojení vrstev

Další dělení rozlišuje neuronové sítě podle propojení jejich vrstev a rozlišuje dva základní typy. Prvním z nich jsou plně propojené sítě, v nichž je každý neuron z jedné vrstvy propojen s každým neuronem z vrstvy následující (příklad na obrázku 2.4). Druhým typem jsou sítě konvoluční. Speciálním případem jsou potom plně konvoluční neuronové sítě.

Plně propojené neuronové sítě: Příklad plně propojené neuronové sítě lze vidět na obrázku 2.4. Takováto síť se učí přesné pozice detekovaných příznaků v obraze, čímž komplikuje detekci objektů na různých pozicích na snímku. Plně propojenou neuronovou síť je příhodné využívat v případech, kdy se pozice objektů v obraze nemění. Příkladem může být rozpoznávání tváří, při kterém síť získá předpřipravený výřez obličeje.

Konvoluční neuronové sítě: Konvoluční neuronové sítě využívají pro zpracování dat konvoluci. Na rozdíl od plně propojených neuronových sítí sdílí váhové koeficienty celá vrstva, čímž se počet koeficientů výrazně redukuje. Koeficienty mají podobu konvolučních jader, což jsou matice, pomocí kterých se konvoluce provádí. U konvolučního operátoru, jak je v této síti nazývána celá konvoluční vrstva, se definuje velikost jádra, velikost kroku konvoluce, velikost rámce nul a počet výstupů. Příklad konvoluční sítě lze vidět na obrázku 2.8.

Konvoluce v konvolučním operátoru je definována jako váhový součet v části obrazu, kterou pokrývá konvoluční jádro. Pro velikost konvolučního jádra je vhodné volit liché číslo, aby byl právě jeden středový bod. Velikost kroku určuje posunutí jádra po obraze, velikost rámce nul přidává k obrazu pixely s nulovou hodnotou po všech okrajích. Počet výstupů je definován počtem konvolučních jader, z nichž každé jádro může zkoumat jiné příznaky. Výhoda konvolučních neuronových sítí spočívá v schopnosti učit se nezávisle na pozici a transformaci objektu na snímku.

Kromě konvolučních vrstev se v těchto sítích často používají tzv. pooling vrstvy (sdružovací), které zajišťují nelinearitu. Pooling vrstva podvzorkovává obraz pomocí nepřekrývajících se jader, čímž redukuje velikost obrazu. Nejmenším možným jádrem je jádro o velikosti 2×2, pro které se použije krok posunu velikosti 2. Výstupem může být průměr ze vstupních hodnot, medián či minimum, nejčastěji se však používá nejvyšší vstupní hodnota. Uváděné jádro zmenší vstupní obraz na poloviční rozměry.

Konvoluční neuronová síť je tvořena několika konvolučními vrstvami a několika pooling vrstvami. Platí pravidlo, že se pooling vrstva nachází až za vrstvou konvoluční, avšak může být vynechána. Na konci sítě se nachází plně propojená vrstva, která slouží jako klasifikátor. Pokud se jedná o plně konvoluční neuronovou síť, závěrečná plně propojená vrstva je vynechána a výstupem je podvzorkovaný vstupní obraz.

Podle způsobu trénování

Podle způsobu trénování lze sítě rozdělit na sítě s učením bez učitele, sítě s učením s učitelem a hybridní sítě.

Sítě s učením bez učitele: Neuronové sítě s učením bez učitele se snaží nalézt souvislosti mezi daty pouze na základě dat samotných bez dalších informací. Síť nemá informace o příslušnosti trénovacích dat do tříd. Nejrozšířenější modely jsou založeny na energii, typickým představitelem je ale síť, jejíž učení je založené na základním modelu autoenkodéru. Autoenkodér je dvouvrstvá neuronová síť se stejným počtem vstupů

a výstupů. První skrytá vrstva obsahuje menší počet neuronů než vrstva výstupní, proto jsou data sítí zobecněna a zakódována. Využití autoenkodéru pro natrénování neuronové sítě je popsáno dále.

Sítě s učením s učitelem: Neuronové sítě s učením s učitelem, neboli diskriminativní sítě, mají k dispozici informaci o příslušnosti trénovacích dat do tříd. Sítě s učením s učitelem se snaží nalézt souvislosti mezi těmito informacemi a samotnými daty.

Hybridní sítě: Hybridní sítě kombinují oba popisované přístupy. Cílová diskriminativní síť je optimalizována pomocí generativní složky. Síť, která je naučená metodou učení bez učitele, může poskytnout ideální inicializační pozici pro síť učenou s učitelem.

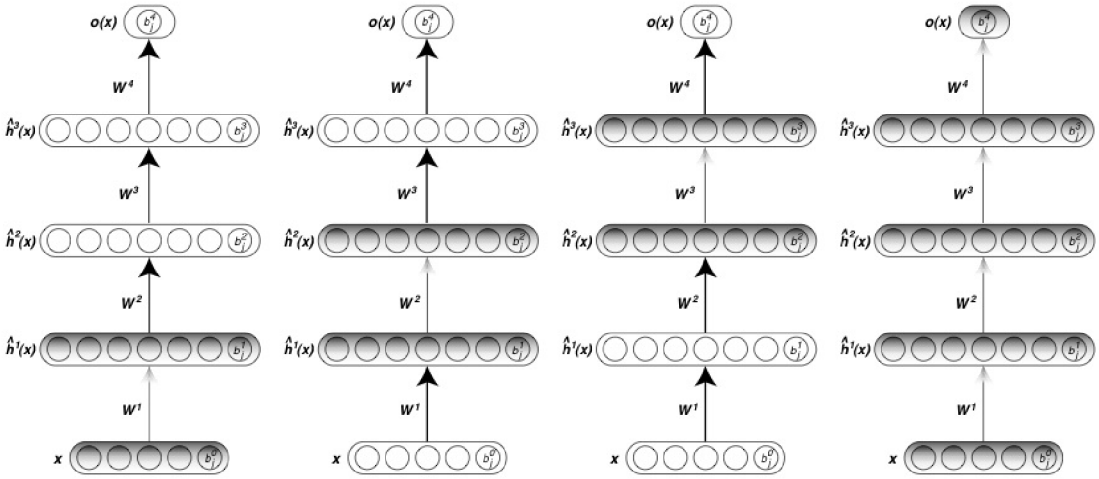
2.2.3 Hluboké neuronové sítě

Jednoduché neuronové sítě se skládají z několika málo vrstev, od jedné po čtyři skryté vrstvy. Takové sítě lze poměrně snadno trénovat, neobsahují příliš mnoho parametrů a jejich trénování i vyhodnocení časově není tolik náročné. Od pěti vrstev už je možné hovořit o hluboké neuronové síti, která s sebou přináší řadu komplikací. Kromě větší časové náročnosti se jedná především o problém s trénováním, jehož řešení je popsáno dále. Obecně však platí, že s přibývajícím počtem vrstev se zlepšují schopnosti sítě, která je schopna problém řešit komplexněji. Lze totiž síť vytvořit z více různých typů vrstev, zkoumajících odlišné příznaky, čímž síť získává nové užitelné informace sloužící k rozhodování.

2.2.4 Učení hluboké neuronové sítě

Pro učení neuronových sítí s malým počtem skrytých vrstev (maximálně čtyři) se nejčastěji používá algoritmus zpětné šíření chyby (*backpropagation*) [8]. Algoritmus hledá takovou konfiguraci sítě, pro kterou je odchylka (chyba) mezi požadovaným a reálným výstupem co nejmenší. Tato chyba je propagována od výstupní vrstvy směrem ke vstupní, zmenšení chyby je dosaženo úpravou vah jednotlivých neuronů. Tímto algoritmem jsou nejvíce měněny váhy ve vrstvě nejblíže vrstvě výstupní, s postupem ke vstupní vrstvě jsou náhodně inicializované váhy měněny čím dál méně. Pokud síť obsahuje více skrytých vrstev (pět a více), úprava konfigurace sítě je zastavena na lokálním minimu, které se od globálního minima může značně lišit. Takto natrénovaná síť nevykazuje příliš dobrých výsledků. První vrstvy sítě, které zajišťují nejhrubší síto, jsou totiž téměř náhodné. Z toho důvodu vědci hledali jiná řešení pro natrénování vícevrstvé neuronové sítě.

Jedno z možných řešení je chytré nastavení počáteční konfigurace namísto náhodné inicializace. Takové konfigurace lze dosáhnout předtrénováním jednotlivých vrstev metodou učení bez učitele, při které je každá skrytá vrstva sítě trénována odděleně. Předtrénování probíhá od vstupní vrstvy směrem k vrstvě výstupní. První vrstva je vhodnou metodou natrénována na základě vstupních hodnot z trénovací sady. Výstupy první vrstvy slouží jako vstupy pro natrénování druhé vrstvy, obdobně pak další vrstvy až k vrstvě výstupní. Parametry, získané tímto předtrénováním, slouží jako inicializační parametry pro samotné natrénování sítě, které může být realizované metodou *backpropagation*. Průběh předtrénování společně s konečným natrénováním neuronové sítě lze vidět na obrázku 2.6. U zvýrazněných vrstev probíhá v daném kroku trénování. Síť předtrénovaná touto metodou je rezistentnější vůči uváznutí v lokálním minimu a obecně vrací lepší výsledky.



Obrázek 2.6: Jednotlivé fáze předtrénování a finální natrénování neuronové sítě (obrázek přejat z [25])

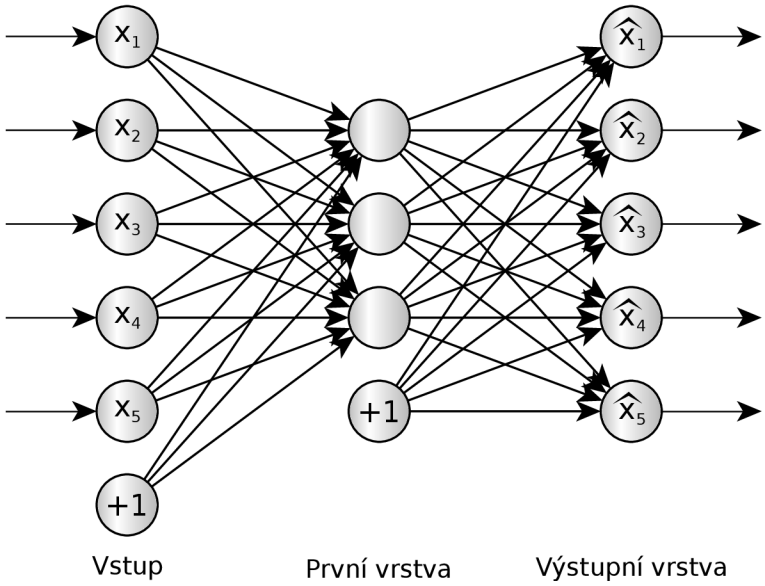
Předtrénování pomocí autoenkodéru

Síť je možné předtrénovat například pomocí dvouvrstvé neuronové sítě zvané autoenkodér. Počet vstupů je u autoenkodéru shodný s počtem výstupních neuronů, počet neuronů v první vrstvě je menší, viz obrázek 2.7. Díky tomu proběhne při trénování sítě zobecnění dat. Autoenkodér lze použít pro zakódování a opětovné dekódování dat či pro komprimaci. Samotné trénování probíhá následujícím způsobem. V prvním kroku jsou trénovány váhy mezi vstupem a první vrstvou. K tomu je vytvořena vrstva dočasných neuronů, jejichž počet je shodný s počtem vstupů. Tato vrstva je připojena za první skrytou vrstvu. Vzniklá struktura odpovídá struktuře autoenkodéru a může být natrénována metodou *backpropagation*. Po natrénování jsou dočasné neurony odstraněny. Získané výstupy se použijí jako vstupy v dalším kroku, ve kterém je opět vytvořena vrstva dočasných neuronů, umístěna za druhou skrytou vrstvu. Obdobně se postupuje pro všechny skryté vrstvy hluboké neuronové sítě.

2.2.5 Existující dostupné architektury

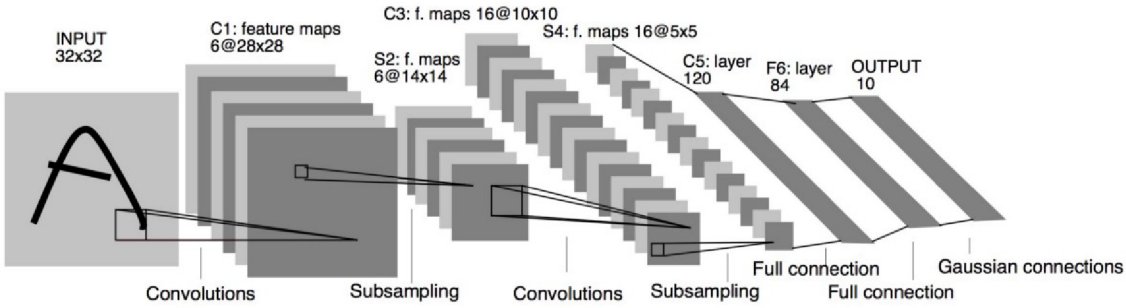
Značná část úspěšnosti neuronové sítě závisí na návrhu její architektury [6]. Jelikož se návrhem neuronových sítí zabývá stále více odborníků, vznikají i různé soutěže, které mají nové architektury porovnávat. Od roku 2010 je pořádána každoroční soutěž *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [31], kde výzkumné týmy předkládají programy, které klasifikují a detekují objekty na snímcích. Od roku 2010 výrazně roste úspěšnost klasifikace. V roce 2011 byla v rámci ILSVRC dobrá chybovost 25 %, v následujících letech se snížila na několik procent. Právě soutěž *ImageNet* motivovala vývojáře a výrazně přispěla k vytvoření úspěšnějších architektur.

Pro porovnání klasifikačních programů se nejčastěji využívají dvě metriky [6]. Pokud je testována přesná shoda referenčního řešení s nejlépe ohodnoceným výstupem testovaného programu, nazývá se tato metrika *Top-1 přesnost* (*Top-1 accuracy*). *Top-5 přesnost* (*Top-5 accuracy*) pak potvrzuje správnost klasifikace v případě, že se referenční řešení nachází mezi pěti nejlépe ohodnocenými výstupy. Na obrázku 2.13 jsou srovnány některé architektury metrikou *Top-1*. Následuje stručný popis některých existujících architektur účastnících se soutěže ILSVRC.



Obrázek 2.7: Schéma autoenkodéru

LeNet5: Jedna z prvních konvolučních hlubokých neuronových sítí vznikla v roce 1994 a byla pojmenována LeNet5. Zásadní pro ni byl pohled na to, že jsou obrazové prvky rozloženy po celém obraze. Konvoluce s parametry, které lze učit, je účinný způsob, jak tyto prvky extrahovat nezávisle na jejich pozici (obrázek 2.8). Jelikož v té době neexistovaly výkonné grafické karty, bylo trénování neuronových sítí velmi pomalé. Klíčovou vlastností proto bylo průběžné ukládání parametrů a výpočtů. Na rozdíl od plně propojených neuronových sítí nevyužívá LeNet5 samostatné hodnoty pixelů v první vrstvě, neboť by tím přišla o vzájemné vztahy mezi pixely.



Obrázek 2.8: Architektura konvoluční neuronové sítě LeNet5. Obrázek přejat z [12].

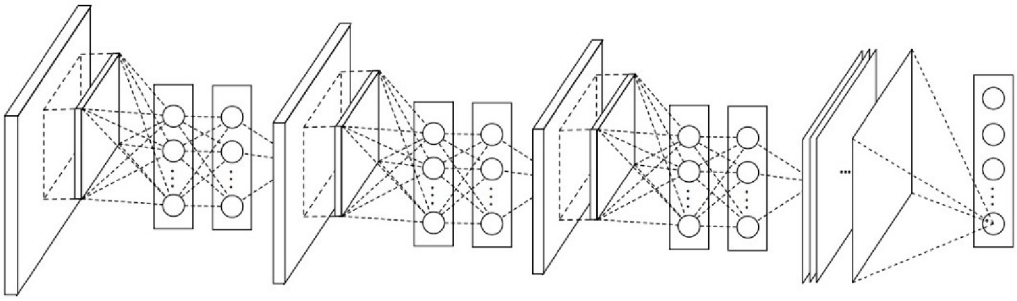
Dan Ciresan Net: Díky levnějším fotoaparátům a rozšíření mobilních telefonů s fotoaparáty začaly na přelomu tisíciletí vznikat objemné datové sady, na nichž bylo možné sítě trénovat. V roce 2010 byla Danem Ciresanem zveřejněna jedna z prvních implementací neuronové sítě na grafické kartě. Síť obsahovala dopřednou i zpětnou implementaci a podporovala až devět vrstev.

AlexNet: Hlubší a mnohem širší verzí předchozí sítě LeNet5 se v roce 2012 stala síť AlexNet, jež v tom samém roce vyhrála soutěž *ImageNet*. AlexNet může být použita na naučení mnohem složitějších objektů. Síť vylepšuje LeNet5 o pooling nejvyšší hod-

noty namísto průměrujícímu poolingu, využití grafických karet pro trénování a další. Úspěch sítě AlexNet započal malou revoluci v oblasti hlubokého učení, od této chvíle se konvoluční neuronové sítě staly hlavním předmětem zájmu.

VGG: Síť VGG z Oxfordu byla první sítí, v níž byly použity daleko menší filtry o velikosti 3×3 pixely. Takto malé filtry byly použity v každé konvoluční vrstvě a byly kombinovány jako sekvence konvolucí. Na první pohled síť VGG odporuje zásadám ze sítí LeNet a AlexNet o používání větších filtrů 9×9 nebo 11×11. V Oxfordu ale zjistili, že sekvence 3×3 konvolucí může napodobit větší filtry.

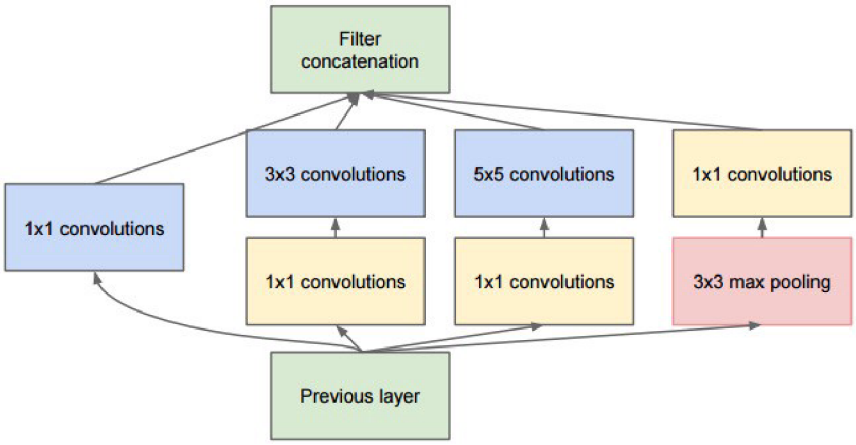
Network-in-network: Architektura sítě v síti používá prostorové MLP vrstvy po každé konvoluci, aby bylo možné lépe zkombinovat parametry před další vrstvou (obrázek 2.9). K tomu využívané filtry 1×1 mohou opět vzbuzovat pochybnosti vzhledem k původním principům sítě LeNet5. Ve skutečnosti však dokáží spojovat konvoluční prvky lépe než pouhé navazování konvolučních vrstev. Ve výsledku tedy používají daleko méně parametrů, neboť jsou sdíleny mezi všemi pixely těchto funkcí. Síla MLP může výrazně zvýšit úspěšnost jednotlivých konvolučních funkcí jejich kombinací do složitějších skupin. Síť v síti používá průměr v pooling vrstvě jako součást posledního klasifikátoru.



Obrázek 2.9: Schéma sítě v síti. Obrázek přejat z [12]

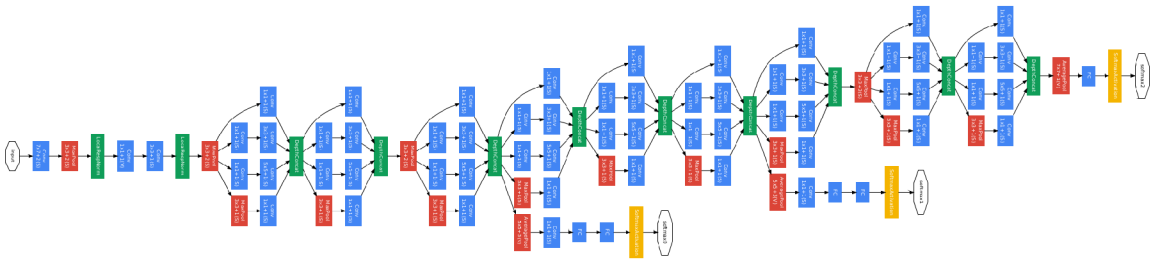
GoogLeNet a Inception: Vzhledem k narůstající úspěšnosti hlubokých neuronových sítí se velké firmy jako Google začaly zajímat o efektivní nasazení neuronových sítí na svých serverových farmách. Vývojář z Google Christian se proto snažil najít způsob, jak snížit výpočetní nároky hlubokých neuronových sítí se zachováním nejlepších možných výsledků, případně jak zachovat nároky a zvýšit úspěšnost. Se svým týmem vymysleli modul Inception, jenž je znázorněn na obrázku 2.10.

Tento modul obsahuje paralelní kombinaci filtrů 1×1, 3×3 a 5×5. Nejvýznamnější je ale použití konvolučních bloků 1×1, jenž snížily počet prvků před náročnými paralelními bloky. Toto úzké místo je označováno jako *bottleneck* a jeho funkci lze uvést na příkladu. Pokud má vrstva 256 vstupů, 256 výstupů a bude provádět konvoluci filtrem 3×3, provede celkem $256 \times 256 \times 3 \times 3 = 589824 \approx 590000$ operací. S použitím filtru 1×1 se provede konvoluce na snížení počtu vstupů např. na 64, tedy $256 \times 64 \times 1 \times 1 = 16384$. Poté je provedena konvoluce filtrem 3×3 s počtem operací $64 \times 64 \times 3 \times 3 = 36864$. Na závěr se výstupy převedou zpět $64 \times 256 \times 1 \times 1 = 16384$. Celkový počet operací je asi 70000, což je v porovnání s počtem operací 590000 skoro desetinásobná úspora. Jelikož mezi vstupními daty existuje korelace, s vhodnou kombinací filtrů se s touto redukcí neztrácí informace. V následujících letech představil



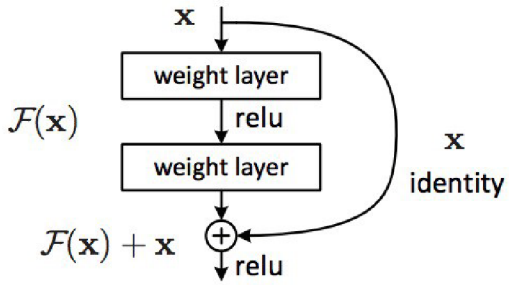
Obrázek 2.10: modul Inception sítě GoogLeNet. Přejato z [37].

jmenovaný tým další verze Inception V2, V3 a V4, jež se liší především v různých variantách po sobě jdoucích 3×3 filtrů.
Celá hluboká neuronová síť GoogLeNet se skládá z 22 vrstev, které obsahují parametry, respektive 27 včetně pooling vrstev. Schéma celé architektury lze vidět na obrázku 2.11.

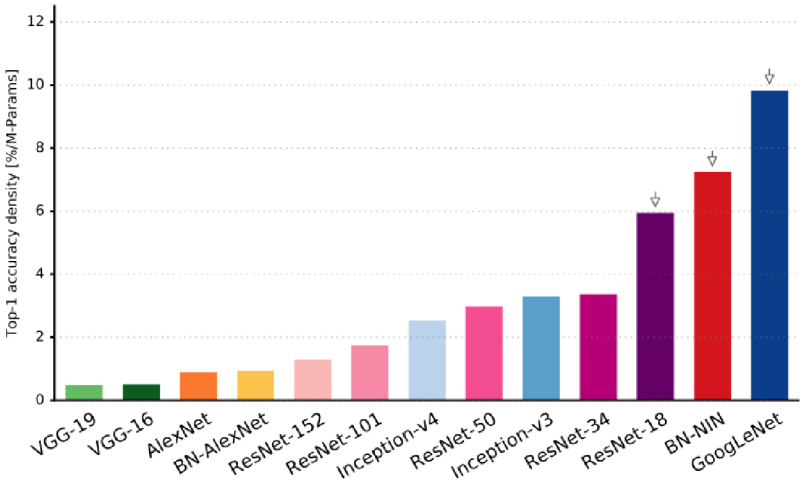


Obrázek 2.11: Schéma konvoluční hluboké neuronové sítě GoogLeNet. Modré vrstvy = konvoluční, červené vrstvy = pooling, žluté vrstvy = plně propojené, zelené vrstvy = ostatní. Obrázek přejat z [37]

ResNet: Architektura ResNet vznikla koncem roku 2015 a měla jednoduchou myšlenku. Výsledek po dvou vrstvách je sloučen s hodnotou před těmito vrstvami, jak lze vidět na obrázku 2.12. Myšlenka přeskočení jedné vrstvy již pochází z dřívější doby, nedosahovala však na rozdíl od přeskočení dvou vrstev výrazného zlepšení. Tyto dvě vrstvy už lze považovat za Síť v síti (Network-In-Network). ResNet je jednou z prvních architektur, která umožnila vytvořit a natrénovat neuronovou síť se stovkami vrstev. Sítě s větším počtem vrstev využívají *bottleneck* vrstvu podobnou jako v Inception. ResNet používá poměrně jednoduchou inicializační vrstvu následující konvolucí filtrem 7×7 a pooling vrstvou. Jako konečný klasifikátor využívá ResNet pooling vrstvu s vrstvou softmax.



Obrázek 2.12: Přeskočení dvou vrstev a následné sloučení v architektuře ResNet. Přejato z [12].



Obrázek 2.13: Srovnání úspěšnosti metrikou *Top-1* různých architektur. Přejato z [12].

2.2.6 Nástroje pro práci s neuronovými sítěmi

Jelikož je trénování i testování neuronových sítí výpočetně velmi náročné, je třeba optimalizovat možné algoritmy. Díky rostoucímu rozšíření neuronových sítí vznikly pro práci s nimi frameworky řešící mimo jiné optimalizace společných a často používaných algoritmů. Pro usnadnění práce je proto vhodné použít některý z nich. Následuje výčet a stručný popis nejpoužívanějších frameworků [19].

Tensorflow: Používaný open-source framework, především pro práci s data flow grafy. Práce s frameworkem je na poměrně nízké úrovni, uživatel musí napsat více kódu než u některých jiných frameworků. Tensorflow podporuje C++ a Python.

Theano: Jeden z nejstarších a stabilních frameworků. Podobně jako v Tensorflow je i programování v Theano na nízké úrovni. Framework bohužel nepodporuje více grafických jednotek a proto přechází do pozadí.

Keras: Novější framework s jasnou syntaxí a dobrou dokumentací. Programování v něm je na vyšší úrovni, Keras pracuje dle výběru nad frameworkem Tensorflow nebo Theano.

Lasagne: Další knihovna fungující na základě frameworku Theano, která se snaží o přívětivější uživatelské rozhraní v Pythonu. Knihovna byla svého času velmi rozšířena, nyní její používání ustupuje.

Caffe: Jeden z nejstarších a v současnosti nejpoužívanějších frameworků se nazývá BVLC Caffe. Jedná se o open source nástroj zaměřující se pouze na počítačové vidění. Nástroj je velmi optimalizovaný a robustní, není však tolik flexibilní. Disponuje rozhraním v C++, Pythonu, Matlabu a dalších. Umožňuje výpočty na procesoru (moduly v C++) i grafické kartě (moduly pro CUDA). V případě procesoru i grafické karty lze zvolit počet použitých jader. Pro návrh struktury je použito serializačního textového formátu *Protobuffer* od Google. Caffe má k dispozici velké množství typů vrstev, umožňuje ale i vlastní definici.

DSSTNE: Často přehlíženým frameworkem je DSSTNE (Destiny), který není určen pro výzkum ani vývoj, nýbrž pro nasazení do výroby. Destiny neumožňuje výběr mezi procesorem a grafickou kartou a není dostatečně popsán.

Torch: Framework používaný ve výzkumu pro Facebook odkoupil Google. Hlavním úskalím frameworku Torch je používání programovacího jazyka Lua.

Mxnet: Mxnet je knihovna podporující velké množství programovacích jazyků (Python, R, C++, Julia a další). Je využívaný společností Amazon a umožňuje měnit počet využívaných grafických jednotek.

DL4J: Dokumentace této knihovny je na velmi dobré úrovni. Knihovna je jednou z mála, která podporuje jazyky Java, Clojure a Scala.

Cognitive Toolkit: Posledním zmíněným frameworkem bude Cognitive Toolkit, dříve známý jako CNTK. Framework vyvíjí společnost Microsoft Research, mezi vývojáři ale není příliš populární.

2.3 Shrnutí

K detekci automobilů ve videu ze statické kamery je možné použít mnoho různých metod, jež se liší v úspěšnosti, robustnosti a náročnosti. Tato kapitola se podrobněji věnovala metodám s modelem pozadí a metodám strojového učení. Metody modelování pozadí lze použít ve videu ze statické kamery, detekovat mohou pouze pohyblivé objekty a to až v okamžiku, kdy má systém vytvořený model pozadí. Mohou však detekovat libovolné objekty, které se pohybují. Byly popsány různé existující metody s modelem pozadí, odlišné v úspěšnosti i náročnosti. Strojové učení s hlubokými neuronovými sítěmi může být použito na daleko širší škálu problémů. Byly představeny nejběžněji používané architektury neuronových sítí, odlišnosti pro hluboké neuronové sítě a jejich učení. Nejzajímavějšími v rámci této práce jsou hybridní hluboké neuronové sítě, u nichž se nejprve samostatně předtrénují jednotlivé skryté vrstvy a až poté se síť natrénuje jako celek. Z pohledu propojení vrstev byly podrobněji popsány konvoluční neuronové sítě vhodné pro obrazová data. Kapitola dále popsala způsob učení hluboké neuronové sítě a představila již vytvořené dostupné architektury. Závěrem shrnula dostupné nástroje pro práci s neuronovými sítěmi.

Kapitola 3

Návrh systému

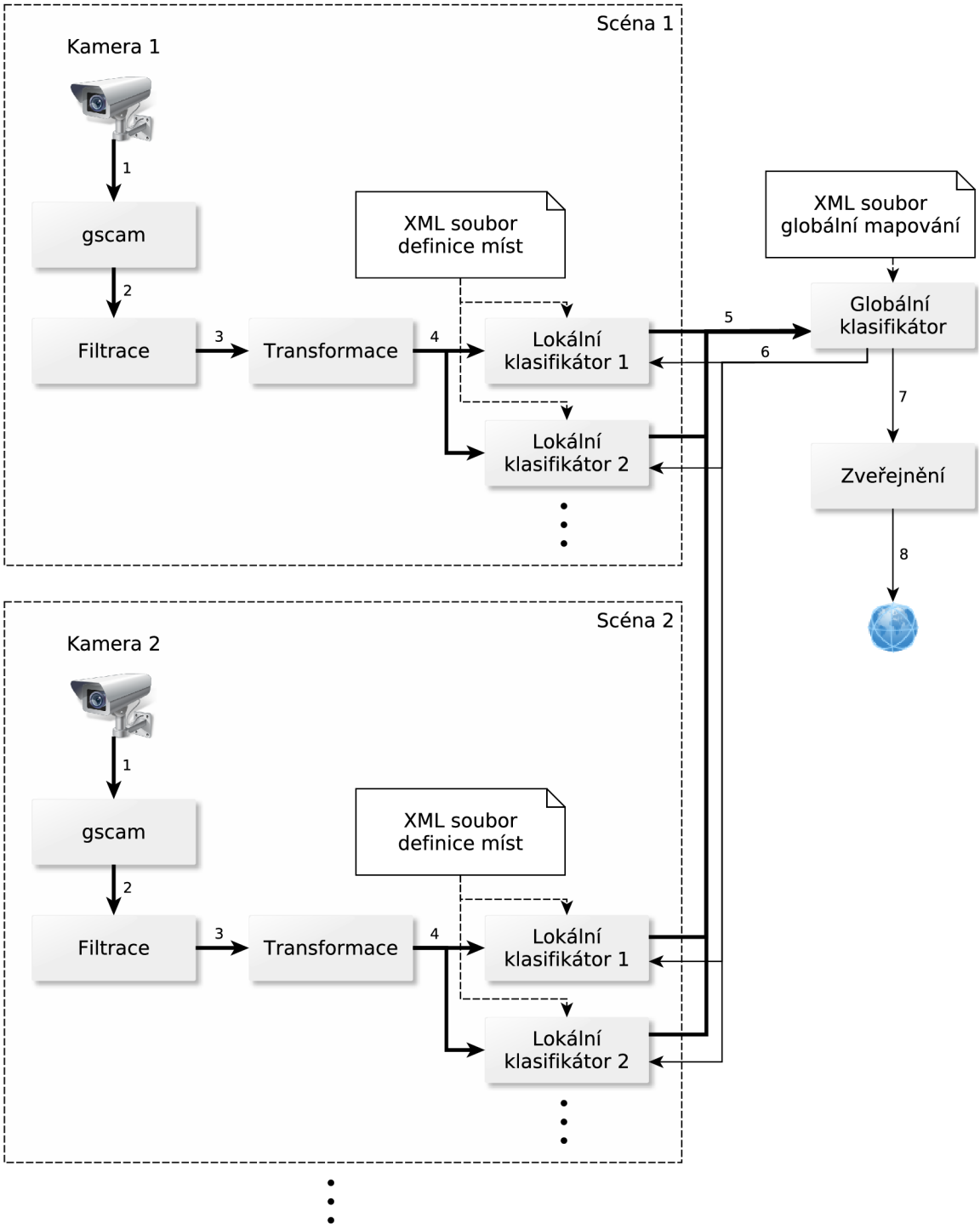
Jedním z hlavních úkolů této diplomové práce je navrhnout a implementovat aplikaci analyzující obsazenost parkoviště. Aplikace je navržena a vytvořena v prostředí Robotického operačního systému (zkráceně ROS). Zásadní výhodou ROSu v rámci této aplikace je možnost rozdělit program do samostatných uzlů (procesů), které spolu vzájemně komunikují pomocí zpráv (komunikace je založena na TCP/IP). To umožňuje restartování pouze jednoho uzlu v případě jeho selhání, jednoduchou tvorbu různých variant aplikace na základě požadavků, transparentnost, ladění pomocí odchytávání zpráv a další.

Při návrhu aplikace byl kladen důraz na běh v reálném čase, robustnost a znovupoužitelnost jednotlivých částí. Kromě samotné aplikace pro detekci je třeba vytvořit software pro označování parkovacích míst a pro anotaci trénovacích videí. Tyto podpůrné aplikace v této práci popisovány nejsou. Struktura definičního XML souboru pro definici parkovacích míst i XML souboru pro mapování lokálních identifikátorů na globální je přejata z již zmiňované bakalářské práce, zbytek systému je navržen nově. Celý návrh se zakládá na předpokladu, že uživatel ručně zadá pozice parkovacích míst, jejichž obsazenost je předmětem zkoumání. Vzhledem k faktu, že je toto označení třeba zadat pouze jednou při instalaci systému na parkoviště, nejedná se o výrazně omezující podmínku.

V této kapitole bude popsán návrh aplikace s využitím teoretických i praktických zkušeností. První část kapitoly popisuje strukturu navrženého systému a podrobněji se věnuje jeho jednotlivým uzlům a propojení mezi nimi. Druhá část představuje lokální klasifikátory a slučování jejich výsledků. V případě překrývajících se pohledů z dvou a více kamer pak systém slučuje výsledky ze všech lokálních klasifikátorů ze všech pohledů, které zkoumané místo obsahují. Kapitolu ukončuje návrh rozložení softwarových částí na hardwarových jednotkách.

3.1 Struktura a popis ROS uzlů

Na obrázku 3.1 je možné vidět návrh struktury celé aplikace. V levé části obrázku se nacházejí skupiny uzlů, vždy jedna skupina pro jednu kameru. Tento návrh umožňuje přidání libovolného počtu kamer bez nutnosti kompilace programu, pouze spuštěním příslušných uzlů. Výsledky z těchto skupin jsou dále zpracovány dohromady a konečné rozhodnutí o obsazenosti může být prezentováno. Podrobnější schéma systému je možné nalézt v příloze na obrázku B.1. Toto schémata je rozšířené o vizualizační uzly a uzly, které zpracovávají výsledná data. V této kapitule dále následuje podrobnější popis jednotlivých uzlů.



Obrázek 3.1: Návrh systému znázorňuje nejvýznamnější ROS uzly, které jsou propojeny pomocí témat (1: rtsp, 2: nezpracované snímky, 3: vybrané nejlepší snímky v určité frekvenci, 4: snímky bez deformací, 5: měkká rozhodnutí o obsazenosti, 6: dotazy pro zkontrolování určitého místa, 7: finální rozhodnutí o obsazenosti, 8: http)

3.1.1 Zpracování snímků z kamery

Uzel, který zpracovává snímky z kamery, se nazývá `Gscam`. Jedná se o již vytvořený uzel v ROSu, který zpracovává video z IP kamery vysílané pomocí RTSP. Uzel vysílá přijaté snímky pomocí již vytvořené zprávy včetně přídavných informací o obrázku a stavu kamery. Uzel navíc kontroluje funkčnost kamery a při jejím výpadku obnoví odesílání snímků. `Gscam` využívá *Gstreamer*, multimediální framework podobný DirectShow. Vzhledem k faktu, že je *Gstreamer* kompatibilní s téměř každým standardem pro nahrávání videa pod Linuxem, činí `Gscam` ROS kompatibilní s naprostou většinou dostupných webových kamer. Díky GStreameru umožňuje `Gscam` pokročilejší zpracování videa (vyvážení bílé barvy, ořezání atd.) i v případě levnějších kamer.

Při spuštění `Gscam` vyžaduje konfigurační řetězec zahrnující adresu kamery, formát obrazu, snímkovou frekvenci a další možné nastavení. Touto konfigurací je uzel možné napojit na libovolnou kameru bez nutnosti kompilace. Pokročilé nastavení pak upravuje výstup uzlu. Bohužel v oficiálním repozitáři ROSu je `Gscam` dostupný v nejvyšší verzi *Hydro Medusa*, bylo tedy třeba tento uzel ručně přeložit (pro použití v novějších verzích). Při této příležitosti byla do uzlu přidána rozšiřující funkcionalita spojená s kontrolou a odesíláním stavu kamery.

3.1.2 Filtrace snímků

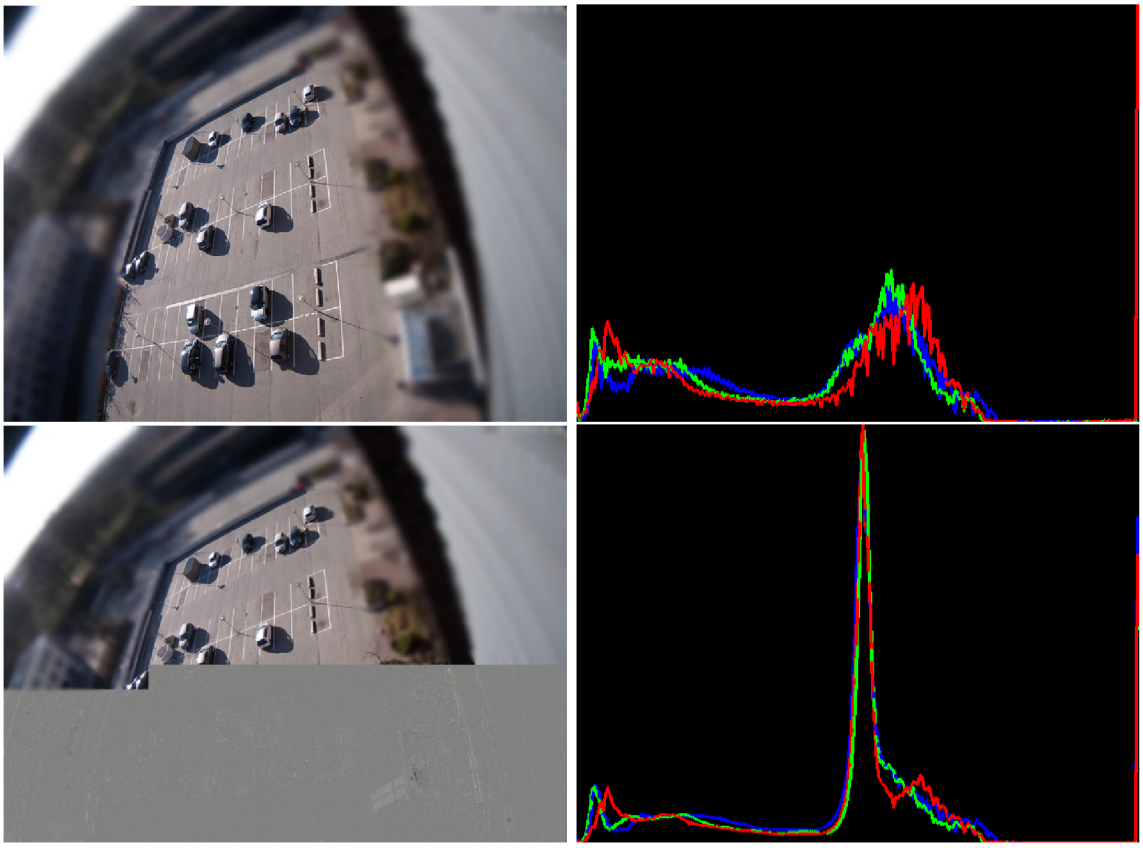
Snímky jsou z uzlu `Gscam` předány do dalšího uzlu k filtraci. Uzel `Image filter` slouží k libovolné filtraci příchozích snímků. V případě potřeby nižšího počtu snímků za vteřinu uzel vybere pouze některé příchozí snímky, které odesílá s nastavenou frekvencí. Výběr snímku může být čistě náhodný nebo se může jednat o složitější algoritmus porovnávající příchozí snímky. Tímto způsobem je možné vyřadit případné vadné snímky. V rámci tohoto uzlu je možné předpřipravit přicházející snímky způsobem totožným pro všechny následující klasifikátory. Příkladem je změna velikosti snímku pro snížení toku dat.

Navržený uzel filtruje snímky pomocí porovnávání histogramu příchozího snímku s průměrným histogramem předchozích snímků (obrázek 3.2). Pokud se histogramy liší, snímek není odeslán a vybírá se jiný snímek. Pokud jsou histogramy podobné, snímek se odešle a průměrný histogram je aktualizován. Práh rozhodující o přijetí může uživatel nastavit, v případě chybných snímků se však hodnoty výrazně liší, není proto nutné hledat přesnou hranici. Takto jsou odstraněny vadné snímky z kamery a přitom zachovány snímky s postupnou změnou (např. změna osvětlení). Uzel odesílá snímky s nastavenou frekvencí, přebytečné snímky zahazuje.

Pro porovnání histogramů existuje řada metod, jež lze rozdělit do tří základních skupin [36]. Tyto skupiny jsou zde stručně představeny s ohledem na konkrétní využití v popisované aplikaci.

Prostý rozdíl

První z nich porovnává pouze celkové velikosti histogramů a nezohledňuje rozložení dílů histogramů. Nazývá se *prostý rozdíl* a v případě stejně velkých snímků postrádá smysl. Vzhledem k tomu, že navrhovaná aplikace zpracovává neustále stejně velké snímky, není třeba se touto metodou dále zaobírat.



Obrázek 3.2: Správný a chybný snímek s odpovídajícími histogramy

Metody díl po dílu

Druhá skupina metod se označuje jako metody *díl po dílu* (*bin-by-bin*). Tyto metody již porovnávají části histogramu se stejnými indexy, nezohledňují však posunutí geometricky podobných histogramů. Budou-li se lišit hodnoty na stejných indexech, metody již nezohlední hodnoty sousedních indexů a histogramy budou vykazovat velkou rozdílnost. Příkladem těchto metod jsou *Minkowského vzdálenost*, *Bhattacharyya vzdálenost*, *statistika* χ^2 a *divergence Kullback-Leibler*.

Metody křížení dílů

Třetí skupina metod porovnává každou část histogramu se všemi ostatními částmi s různými indexy, při výpočtu rozdílu histogramů zohledňuje topografickou vzdálenost dílů. Tyto metody jsou výpočetně náročnější, ale dosahují globálně lepších výsledků než metody z předchozích skupin. Do této skupiny patří metody *kvadratická vzdálenost*, *kumulativní vzdálenost* a *statistika Kolmogorov-Smirnov*.

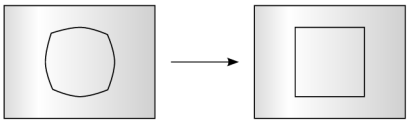
3.1.3 Transformace snímků

Pro přesnější klasifikaci jsou ze snímků odstraněny nežádoucí deformace. Další uzel odstraňuje radiální distorzi a perspektivní zobrazení, aby byla všechna parkovací místa v obraze podobně velká. Ukázky snímků bez transformací a s odstraněnými deformacemi lze vidět

na obrázku 3.5. Odstranění zmíněných deformací kromě přesnější klasifikace zaručuje také snížení toku dat, odstraňuje totiž ze snímků nepotřebné okolí parkoviště.

Radiální distorze

Radiální distorze neboli radiální zkreslení je jedním z nejvýznamnějších zkreslení obrazu [21]. Při radiální distorzi není příčné zvětšení po celém poli obrazu stejné, čímž je porušena geometrická podobnost předmětu a jeho obrazu. V obraze se projevuje zakřivenými liniemi, které by měly být přímé. Toto zkreslení je způsobeno zakřivením čočky kamery, často je ale úmyslné, neboť je kamera schopna zachytit rozsáhlejší oblast. Objektiv s čočkou, jejíž záběr má velmi široký úhel (diagonální úhel okolo 180°), se označuje jako rybí oko.



Obrázek 3.3: Čtverec na snímku s radiální distorzí je opraven na čtverec s rovnými liniemi.

Jelikož je obraz po krajích zdeformován více než uprostřed, je vhodné pro následující trénování deformaci odstranit (viz obrázek 3.3). Radiální distorzi lze ze snímku odstranit aplikací následujících rovnic [21]:

$$\hat{x} = x_c + L(r)(x - x_c) \qquad \hat{y} = y_c + L(r)(y - y_c) \tag{3.1}$$

(x, y) jsou původní souřadnice, $(\hat{x}, \hat{y})$ jsou nové souřadnice s odstraněnou distorzí a (x_c, y_c) určuje střed radiálního zkreslení s poloměrem r. Funkce $L(r)$ je definována pro pozitivní hodnoty r a platí $L(0) = 1$. Funkce dostatečně aproximující odstranění distorze může být dána Taylorovým polynomem:

$$\hat{x} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad \hat{y} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.2}$$

Pro odstranění distorze je třeba znát koeficienty $\{k_1, k_2, k_3, x_c, y_c\}$. Možný způsob jejich zjištění je použitím kalibračního videa a výpočtem z tohoto videa, např. funkcí `calibrateCamera` z knihovny OpenCV. Společně s těmito koeficienty je třeba znát i kalibrační matici kamery. Knihovna OpenCV umožňuje odstranit i tangenciální zkreslení. Na snímku toto zkreslení však není tolik výrazné, a proto mu zde není věnována větší pozornost.

Perspektiva

Perspektiva je optický jev, při němž se vzdálenější objekty zdánlivě jeví menší než bližší objekty [21]. V obraze se projevuje zkracováním linií vzdálenějších od kamery, sbíháním rovnoběžek směrem k jednomu bodu (úběžník) a zmenšováním vzdáleností mezi objekty vzdálenějšími od kamery.

Pro následné zpracování snímků je vhodné, aby byla všechna parkovací místa na snímku podobně velká, čehož je možné dosáhnout právě odstraněním perspektivního zkreslení (obrázek 3.4). K odstranění je třeba transformační matice, kterou lze snadno vypočítat. Pro výpočet je nutné zadat alespoň tři body v původním obraze a jim odpovídající tři body v novém obraze, které jsou svým rozložením shodné s tvarem reálnému objektu. V knihovně

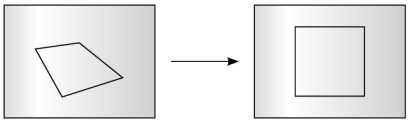
openCV provádí tuto transformaci funkce `findHomography`, která hledá perspektivní transformaci H mezi zdrojovou a cílovou plochou:

$$s_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.3}$$

Samotný výpočet se snaží minimalizovat chybu zpětné projekce:

$$\sum_i \left(\left(x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \right) \tag{3.4}$$

Jelikož je obraz mimo parkoviště pro detekci obsazenosti nepodstatný, je ho možné společně s odstraněním perspektivního zkreslení odstranit. Označí-li uživatel celé parkoviště čtyřmi body v rozích parkoviště, lze jako odpovídající body v novém obraze použít právě rohy výsledného snímku. V závislosti na rozměrech parkoviště a snímku může být obraz deformován (roztažen) v ose x nebo y. To lze vyřešit správným nastavením rozměrů snímku. Vzhledem k tomu, že jsou ale roztažena všechna parkovací místa, může být i takto deformovaný obraz již ponechán (za předpokladu, že jsou takto deformované snímky použity i pro trénování klasifikátoru).



Obrázek 3.4: Čtverec na snímku s perspektivním zkreslením je opraven na čtverec s rovnoběžnými stranami.

3.1.4 Lokální klasifikátory

Lokální klasifikátory jsou jednou z nejvýznamnějších částí celé aplikace. Návrh systému zahrnuje větší množství jednodušších lokálních klasifikátorů, jejichž dílčí výsledky jsou dále zpracovány a slučovány. Lokální klasifikátor je proto rozdělen na dvě části. První část je společná pro všechny lokální klasifikátory a zahrnuje rutinní práce jako je přijetí snímků či odeslání výsledků. V této části je také na začátku běhu aplikace načten konfigurační soubor, který popisuje umístění parkovacích míst, jejich tvar a rozdělení do skupin.

Výstupem každého lokálního klasifikátoru je sada měkkých rozhodnutí o obsazenosti příslušných parkovacích míst. Pro odesílání výsledků lze zvolit jeden ze dvou režimů. Pokud jsou všechna místa zpracovávána současně, odesílá se jedna zpráva slučující obsazenost všech parkovacích míst. Pokud je pro zpracování parkovacího místa třeba delší doby, může klasifikátor odesílat postupně více zpráv, jež obsahují vždy jedno parkovací místo. Oba přístupy je také možné kombinovat.

Samotná klasifikace obsazenosti parkovacích míst probíhá v druhé části lokálního klasifikátoru. Tato část je již pro každý klasifikátor rozdílná a právě na ní závisí úspěšnost celé klasifikace. Jako lokální klasifikátor je možné použít prakticky jakoukoliv funkci rozhodující o obsazenosti parkovacích míst, jejíž úspěšnost je lepší než 50 %. Příklady úspěšných klasifikátorů jsou hranový klasifikátor, klasifikátor založený na modelu pozadí a detekci pohybujících se vozidel či klasifikátor založený na porovnávání histogramů. Během vývoje bylo implementováno několik klasifikátorů, které testovaly návrh systému. Příkladem může

Obrázek 3.5: Původní snímek (vlevo nahoře), snímek po odstranění radiální distorze (vpravo nahoře) a snímek po odstranění perspektivy (dole)

být hranový klasifikátor určující obsazenost parkovacího místa pouze na základě počtu výrazných hran na tomto místě. Po přidání výrazně úspěšnějších klasifikátorů byly tyto jednodušší verze nepotřebné, byly proto ze systému odstraněny. Aktuální verze popisované aplikace používá dva klasifikátory, hlubokou neuronovou síť a klasifikátor s modelem pozadí. Jejich funkčnost bude podrobněji popsána dále.

Návrh systému umožňuje klasifikaci všech parkovacích míst najednou nebo klasifikaci po jednom místě. V případě použití druhé varianty (např. kvůli déle trvající době výpočtu) je pro výběr následujícího parkovacího místa pro klasifikaci využita plánovací fronta se třemi prioritami. Nejnižší prioritu získávají parkovací místa automaticky, pokud není přidán požadavek s vyšší prioritou. Parkovací místa s nejnižší prioritou testuje klasifikátor postupně a nepřetržitě. Pokud globální klasifikátor odešle lokálnímu klasifikátoru požadavek na zkontrolování obsazenosti určitého parkovacího místa, přidá se do fronty toto místo s nejvyšší prioritou. Místa s nejvyšší prioritou jsou klasifikována vždy přednostně, neboť jsou pro globální klasifikátor nejvýznamnější. K požadavku dochází např. v případě, kdy jiný klasifikátor detekoval změnu obsazenosti a je třeba aktuální stav posoudit. Po klasifikaci místa s nejvyšší prioritou je za uživatelem zvolený čas naplánovaná klasifikace tohoto místa se střední prioritou, aby bylo rozhodnutí zkontrolováno.

3.1.5 Globální klasifikátor

Výsledky lokálních klasifikátorů jsou slučovány v globálním klasifikátoru, který je společný pro celý systém. Globální klasifikátor má dvě hlavní funkce. Za prvé slučuje měkká rozhodnutí lokálních klasifikátorů v rámci jedné skupiny, tedy určuje finální obsazenost parkovacích míst z pohledu jedné konkrétní kamery. V případě jednokamerového systému se jedná o hlavní funkci globálního klasifikátoru.

V případě vícekamerového systému je druhou funkcí slučování rozhodnutí o obsazenosti jednoho parkovacího místa snímaného více kamerami. Přiřazení dvou a více označení ke konkrétnímu parkovacímu místu je realizováno prostřednictvím globálního konfiguračního souboru, který mapuje označení v rámci každé scény na globální označení parkovacího místa. Toto mapování zadává uživatel před spuštěním aplikace obdobně jako označování jednotlivých parkovacích míst.

Tyto dvě funkce jsou v systému realizovány současně, neboť pohled z další kamery lze chápat jen jako další sadu klasifikátorů. Zaznamenávání dílčích výsledků v globálním klasifikátoru a jejich zpracovávání se realizuje následovně.

Zásobník dílčích výsledků

Základní jednotkou, která realizuje uchovávání výsledků, jejich zpracování a vyhodnocení je objekt se zásobníkem nazvaný `Rate Stack`, viz obrázek 3.6. Každému parkovacímu místu v globálním prostoru je přiřazen právě jeden zásobník. Pokud se parkovací místo nachází na více pohledech více kamer a v každém pohledu je označeno jiným identifikátorem, prostřednictvím globálního mapování se tomuto místu přiřadí pouze jeden globální identifikátor, a proto i jeden zásobník.

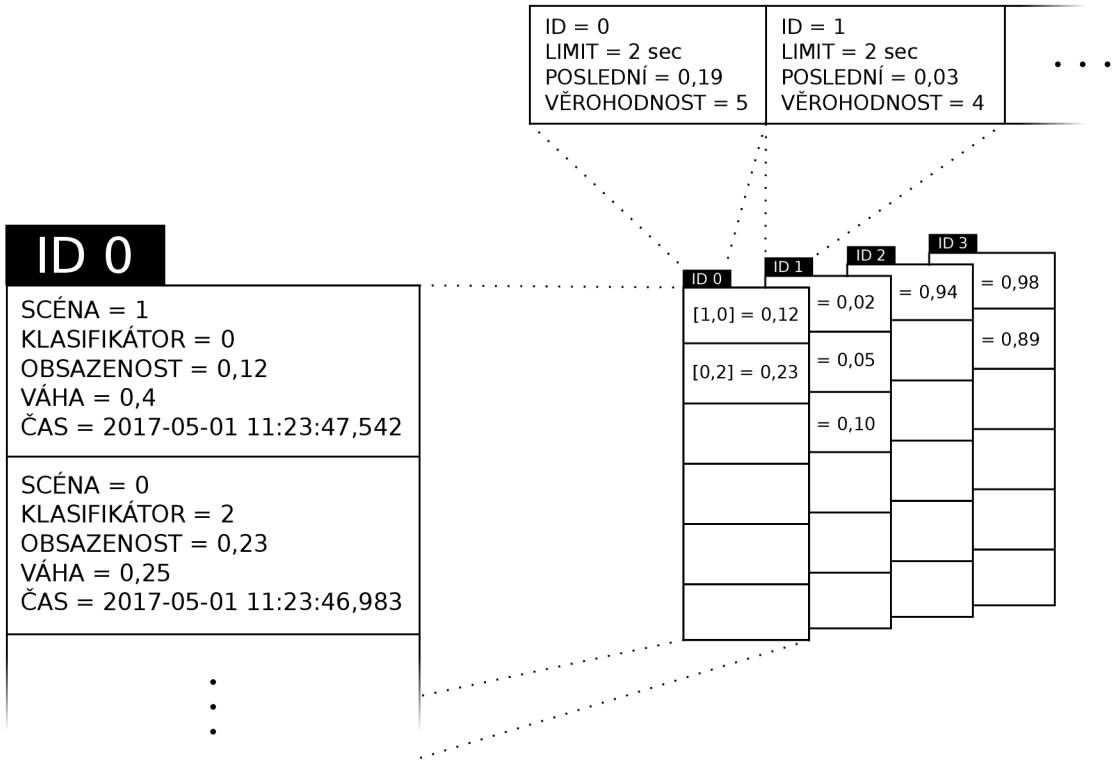
Obdrží-li globální klasifikátor jeden či více výsledků od lokálních klasifikátorů, uloží tyto výsledky do příslušného zásobníku se správným identifikátorem. Kromě samotné hodnoty je uložen i identifikátor scény a klasifikátoru, váha klasifikátoru a časové razítko. V pravidelných intervalech jsou pak tyto hodnoty využity pro výpočet konečného rozhodnutí o obsazenosti. Způsob výpočtu závisí na zvoleném typu objektu `Rate Stack`. Jsou navrženy tři typy objektů, které lze pro tento výpočet využít.

Vážený průměr: Nejjednodušším způsobem sloučení výsledků je vážený průměr. Na základě předchozí úspěšnosti samostatných lokálních klasifikátorů je každému z nich přiřazena váha, kterou se násobí výsledek vypočítaný tímto klasifikátorem.

SVM: Druhou možností je Support Vector Machine, jejímž základem je lineární klasifikátor. Tato metoda má výhodu v tom, že je schopna začlenit do rozhodování i další informace, které ovlivňují úspěšnost lokálních klasifikátorů. Těmito informacemi mohou být aktuální denní doba, míra osvětlení parkoviště a další.

Neuronová síť: Třetí možností je jednoduchá neuronová síť. Obdobně jako SVM umožňuje zahrnout do rozhodování přídavné informace, navíc se ale nejedná o lineární klasifikátor. Její použití má význam v případě, kdy je k dispozici velké množství klasifikátorů a přídavných informací.

Objekt se zásobníkem dílčích výsledků obsahuje kromě zásobníku další potřebné informace, jako jsou časový limit, poslední hodnota obsazenosti a věrohodnost. Časový limit je použit vždy, když se získávají platné hodnoty ze zásobníku. Časové razítko každého záznamu je porovnáno s tímto limitem a pokud je starší než limit, je tento záznam ze zásobníku odstraněn. Poslední hodnota se uchovává pro případ, že by nastal problém s vyhodnocováním současné obsazenosti (např. výpadek kamery). V takovém případě je nejschůdnější řešení



Obrázek 3.6: Zásobníky s obsazeností lokálních klasifikátorů a dodatečnými informacemi

odesílat poslední dostupnou informaci o obsazenosti. Věrohodnost pak říká, do jaké míry je výsledek věrohodný v rámci rychlých změn. Využívá se pro odstranění tzv. "přeblikávání" obsazenosti, způsobené hodnotami blízkými rozhodovacímu prahu.

3.1.6 Prezentace výsledků

Jelikož je pro člověka numerická podoba identifikátorů a obsazenosti míst těžko čitelná, mohou být získané výsledky několika způsoby znázorněny a následně zveřejněny. K okamžité kontrole správnosti systému jsou odesílány přes protokol HTTP a zobrazovány na webové stránce jako barevné označení získaných výsledků na snímcích z kamer. Pro pozdější analýzu jsou ukládány do databáze jako změna obsazenosti konkrétního místa v aktuálním čase. Z uložených dat je možné získávat různé statistiky a vykreslovat odpovídající grafy. Mezi takové grafy patří historie obsazenosti, průměrná obsazenost během dne, průměrné trvání obsazenosti jednoho místa a další.

V reálném nasazení se nabízejí dvě základní technologie zobrazování výsledků řidiči. Komplexnější z nich zahrnuje zjednodušený model parkoviště v mobilní aplikaci či na webové stránce, na kterou se může řidič kdykoliv podívat. Takovéto schéma obsahuje více informací pro řidiče, který si může lépe vybrat preferované parkovací míst. Aplikace může zahrnovat znázornění barev automobilů pro lepší orientaci, dobu trvání obsazenosti či předpokládaný čas uvolnění parkovacího místa. Méně komplexním avšak pohodlnějším řešením jsou světelné cedule rozmístěné po parkovišti, které řidiče přímo navádějí k volným parkovacím místům. V případě cedulí řidič nemusí vlastnit mobilní telefon ani jiné zařízení a především vidí směr volných míst okamžitě bez nutnosti zapínání aplikace.

3.2 Klasifikace obsazenosti místa

Jak již bylo řečeno, lokálním neboli dílčím klasifikátorem se může stát prakticky jakákoliv funkce, která ohodnotí obsazenost parkovacích míst v reálném čase s úspěšností lepší než 50 %. Z důvodu omezených zdrojů a přehlednosti je však žádoucí co nejlepší úspěšnost klasifikace. Následuje výčet navrhovaných klasifikátorů vždy s krátkým popisem.

Hranový klasifikátor

Jednoduchý klasifikátor založený na předpokladu, že se na neobsazeném parkovacím místě vyskytuje řádově méně výrazných hran než na obsazeném místě s automobilem. Úspěšnost této metody je závislá na povrchu parkoviště a na vzhledu jednotlivých automobilů. Jelikož jsou detekované hrany znázorněny konstantní šířkou čáry, velmi tomuto klasifikátoru napomáhá odstranění perspektivy. Na snímcích s perspektivou jsou totiž vzdálenější místa výrazně menší než místa bližší, poměr označených pixelů coby hran se proto také výrazně liší. Odstraněním perspektivy se tyto rozdíly ztrácejí. Výhodou hranového klasifikátoru je velmi malá náročnost a schopnost klasifikace z jediného snímku. Není proto třeba počáteční inicializace a klasifikátor je možné využít i pro inicializaci dalších klasifikátorů nebo pro usnadnění anotací uživatelem. Hranový klasifikátor není výpočetně náročný, avšak úspěšnost se výrazně zhorší v případě okluzí či stínů.

Klasifikátor s modelem pozadí

Složitější klasifikátor, než je klasifikátor hranový, využívá ke klasifikaci model pozadí. Porovnáním aktuálního snímku s modelem pozadí je možné detekovat pohybující se automobily, které přijíždějí či odjíždějí na sledované parkovací místo. Metoda vyžaduje počáteční inicializaci za předpokladu, že jsou některá parkovací místa obsazená. V takovém případě není možné vytvořit si model pozadí na začátku, princip je třeba obrátit. Nejprve se vytvoří model pro přítomný automobil, při první významné změně se místo klasifikuje jako volné a poté je možné vytvořit model pozadí. Modelu pozadí je v takovém případě třeba předat informaci, které části snímku se mají aktualizovat (pozadí) a které mají zůstat neaktualizovány (obsazená místa).

Metoda má hlavní rizika v aktualizaci modelu pozadí. Na rozdíl od sledování jedoucích vozidel na vozovce není možné aktualizovat model pozadí po celou dobu. Po dobu obsazenosti parkovacího místa se pozadí může výrazně změnit (změna světelných podmínek, sníh na vozovce, stín atp.). Po opuštění místa automobilem se model pozadí od aktuálního snímku liší natolik, že je místo stále považováno za obsazené. Právě z tohoto důvodu není možné využít pouze jednoduché metody a je třeba obrátit se k metodám sofistikovanějším.

Porovnání s referenčním bodem

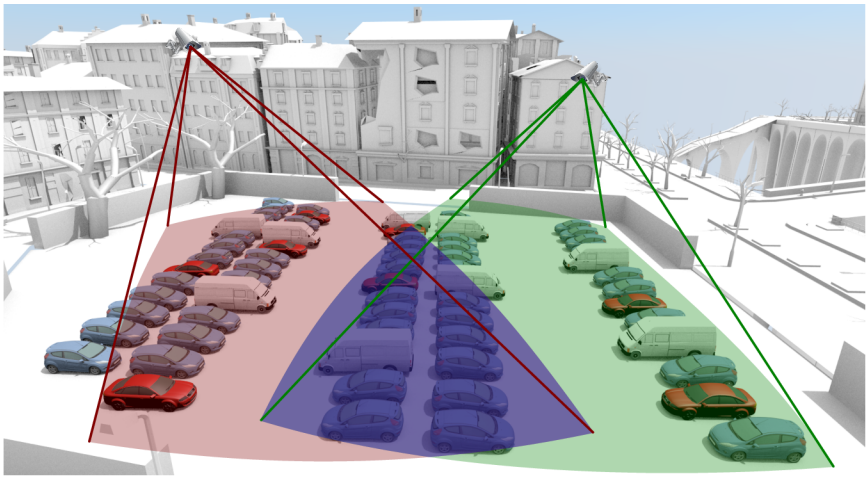
Podpůrnou metodou při změně okolních podmínek je porovnání s referenčním bodem. Uživatel ke každému sledovanému parkovacímu místu přiřadí oblast, která se bude měnit pravděpodobně stejným způsobem jako pozadí daného místa. Podmínkou je, aby byla tato oblast po většinu času neobsazená, většinou se tedy bude jednat o vozovku před parkovacím místem. Detekovaný rozdíl je možné uplatnit při porovnání s modelem pozadí nebo v jiných metodách. Samostatně je však tato metoda nepoužitelná, neboť se v případě průjezdu automobilu referenčním bodem zásadně změní porovnání tohoto bodu s parkovacím místem a dojde k chybnému určení obsazenosti.

Detekce vozidel

Sofistikovanější metodou je přímá detekce vozidla. Tu je možné provést různými metodami, které se předem naučí, jak automobil na parkovacím místě vypadá. K detekci mohou být využity metody AdaBoost, Haarův klasifikátor a další. Vzhledem k úspěšnosti a povaze zadání této práce je použita hluboká neuronová síť. Úskalí těchto metod v rámci detekce na parkovišti je skutečnost, že kamera snímá parkovací místa pod různými úhly. Proto je obtížnější vytvořit jeden model pro libovolné detekované vozidlo. Nabízejí se dvě možnosti řešení. První možností je vytvoření většího počtu modelů automobilů z různých úhlů a parkovacímu místu přiřadit model, který místu nejvíce odpovídá. V ideálním případě bude mít každé parkovací místo vlastní model. Tato varianta je velmi náročná na trénovací data i na běh aplikace, neboť je třeba uchovávat modely pro všechna místa. Metoda bude selhávat v případech, kdy vozidlo nestojí přesně na parkovacím místě, ale zasahuje do něj jen svou částí. Druhou možností je trénovat pouze jednu síť, vytvořit ale větší počet tříd, mezi nimiž se bude síť na základě vstupních snímků rozhodovat. Mezi takové třídy může patřit prázdné parkovací místo, vozidlo z boku, vozidlo zepředu a další. Po klasifikaci jsou třídy sloučeny zpět do dvou základních (volno nebo obsazeno).

3.3 Sloučení výsledků klasifikátorů

Na obrázku 3.7 je znázorněna možnost překryvu pohledů dvou kamer umístěných na budově přilehlé k parkovišti. Barevně jsou zde označeny tři oblasti. Červená oblast je snímána levou kamerou a obsazenost bude rozhodnuta na základě klasifikátorů ze skupiny příslušící této kameře. Obdobně budou parkovací místa nacházející se v zelené oblasti označena za pomoci klasifikátorů z druhé skupiny. Nejzajímavější je modrá oblast uprostřed, v níž se pohledy obou kamer překrývají. Nejjednodušší možností by bylo použití klasifikátorů ze skupiny s kamerou, z které jsou parkovací místa vidět lépe. V takovém případě by se však ztratily informace z druhé kamery, které by při rozhodování mohly pomoci. Proto budou parkovací místa, která se nacházejí v této oblasti, zpracována klasifikátory z první i druhé skupiny a využity tak snímky z obou kamer. Navíc je možné využít tuto informaci pro zpřesnění detekcí i ostatních míst, neboť je možné porovnat vzhled stejné oblasti z různých kamer.



Obrázek 3.7: Ukázka překryvu pohledů z dvou kamer. Ideální pozice kamery by byla na budově uprostřed, jelikož je ale zničená, není na ni možné kameru umístit.

3.4 Shrnutí

Aplikace byla vytvořena v prostředí Robotického operačního systému a rozdělena na jednotlivé uzly, každý uzel s vlastní funkcionalitou. Nejprve jsou snímky zpracovány z kamery, následně jsou všechny vadné snímky vyfiltrovány. Před samotnou klasifikací se ještě odstraní dvě deformace, radiální distorze a perspektiva. Nejdůležitějšími uzly jsou lokální klasifikátory, jejichž výstupem jsou měkká rozhodnutí o obsazenosti samotných míst. Dva hlavními lokální klasifikátory se zakládají na hluboké neuronové síti a modelu pozadí. Doplňují je jednodušší klasifikátory, např. hranový klasifikátor. Konečné rozhodnutí o obsazenosti provádí globální klasifikátor, jenž slučuje výsledky dílčích klasifikátorů z více kamer a dalších přídavných informací. Výsledná obsazenost je prezentována uživateli v různých podobách.

Kapitola 4

Implementace aplikace

Jak již bylo řečeno v předchozím textu, celý systém je navržen a implementován v Robotickém operačním systému (ROS). ROS podporuje programovací jazyky C++ i Python, v rámci této práce byly využity oba. Naprostá většina uzlů je programována v jazyce C++, Python je použit pro uzly nesouvisející s klasifikací, tzn. zpracování a odesílání dat do databáze a odesílání dat do webového rozhraní. Ačkoliv je Robotický operační systém experimentálně provozován i na dalších platformách, doporučeny a plně podporovány jsou dvě, Ubuntu a Debian. Pro vývoj byl zvolen nejprve Linux Ubuntu 14.04, který je podporován verzí ROSu Indigo Igloo. Později byl vývoj přesunut na Linux Ubuntu 16.04 a ROS Kinetic Kame.

Tato kapitola se nejprve věnuje použitým softwarovým nástrojům, z nichž nejvíce popisuje Robotický operační systém. Následuje popis implementace kritických částí systému. Na závěr kapitoly je přiblíženo označování parkovacích míst.

4.1 Použité softwarové nástroje

V dnešní době vzniká stále více volně dostupných knihoven a nástrojů, které vývojářům usnadňují práci. Ve většině případů není žádoucí vytvářet vlastní implementace již vytvořených a především dobře otestovaných algoritmů a funkcí. V této práci je dle zadání využit Robotický operační systém [30] (zkráceně ROS), do jehož prostředí je aplikace zasazena. Pro zpracování obrazu a mnohé funkce, týkající se počítačového vidění, je možné použít rozšířenou knihovnu OpenCV (kde CV neboli Computer Vision znamená počítačové vidění). Další podpůrné knihovny jsou sdruženy ve frameworku Qt, který pro tuto práci není nezbytný, avšak usnadňuje práci a sjednocuje některé přístupy. Navíc tento framework umožňuje tvorbu grafických aplikací, čímž usnadňuje vytváření aplikací s uživatelským rozhraním. Jednou z těchto aplikací je například grafický editor pro označování parkovacích míst a jejich anotaci.

Naprosto zásadní pro tuto práci je skutečnost, že ROS integruje knihovny Qt i OpenCV a obsahuje balíček `cv_bridge`, který propojuje funkce ROSu a OpenCV. Bez tohoto propojení by nebylo možné posílat obrázky a zpracovávat je pomocí funkcí z knihovny OpenCV. Nevýhodou tohoto řešení se stává závislost na konkrétních verzích jednotlivých nástrojů a obtížnější přechod na novější verze. Příkladem nechť je uveden import frameworku Qt, který se značně liší pro verze Qt 4.8 a Qt 5.0. Rozdíly jsou především v konfiguračním souboru pro překlad `CMakeList.txt`, o kterém bude více informací dále. Obdobně komplikované je využití nejnovější verze knihovny OpenCV. ROS implicitně obsahuje verzi

OpenCV 2.4. Pokud chce programátor použít verzi OpenCV 3.0 či vyšší, je třeba přenastavit importování starší verze z ROSu a nastavit jako implicitní novou verzi včetně všech cest k této knihovně.

4.1.1 ROS

Robot Operating System, zkráceně ROS, se stal především v oblasti mobilní robotiky velice populárním a používaným frameworkem. Používají ho tisíce lidí na celém světě. V roce 2013 bylo oficiálních uživatelů přes 1500, přes 3300 lidí pracujících na online dokumentaci a 5700 lidí poskytujících online podporu [1]. Systém je šířen pod licencí BSD, je tedy distribuován jako otevřený, volně šiřitelný software a je zdarma [38]. Plně podporován je ROS pro operační systém Linux Ubuntu, experimentální provoz je však na mnoha dalších platformách. Velkou výhodou ROSu je podpora mnoha sensorů, motorů i celých robotů. Aktuálně podporuje asi 45 komerčních robotů [41].

Koncept

ROS je navržen tak, aby byl co nejvíce modulární a flexibilní, uživatelům tedy umožňuje použít jen některé jeho části, některé části si pak uživatel může vytvořit sám. Výsledný program se skládá z jednotlivých procesů neboli uzlů. Uzly spolu vzájemně komunikují prostřednictvím zpráv. Díky rozdělení do uzlů je možné používat jeden kód pro více aplikací, případně spouštět uzly na různých počítačích. Základní filozofie ROSu lze vyjádřit pěti body:

- Peer to Peer
- Distribuovaný do uzlů
- Nezávislý na programovacím jazyku
- Snadno použitelný
- Zdarma a open source

Systém je rozdělen na dvě základní vrstvy: jádro, které je spravováno kalifornskou společností Willow Garage, a robotický software, který je dílem ROS komunity [24].

Uzly

Základními prvky systému vytvořenému v ROSu jsou uzly. Uzly si lze představit jako samostatné programy či skripty, které jsou spouštěny a ukončovány odděleně. Při spouštění může uživatel uzlu zadat libovolný počet parametrů, které uzel může či nemusí zpracovat. Uzly spolu mohou komunikovat prostřednictvím témat a zpráv, jinak jsou ale odděleny. Díky tomu mohou být spouštěny nezávisle na sobě, případně i na různých zařízeních. Pokud jeden uzel selže, ostatní uzly mohou fungovat i nadále, než je chybný uzel restartován. Rozdělení do uzlů umožňuje vytvářet různé kombinace z již vytvořených uzlů a tím docílit odlišných variant požadované aplikace bez nutnosti změny kódu a kompilace. Další výhodu lze spatřit v implicitním rozdělení pro jednotlivá jádra procesoru bez nutnosti rozdělování do vláken uživatelem. Tento fakt je obzvláště při náročných výpočtech zpracování obrazu velmi užitečný a usnadňuje programátorovi práci.

Publisher a Subscriber

Jak již bylo řečeno, uzly spolu mohou komunikovat prostřednictvím témat a zpráv. Pokud uzel vysílá zprávy, je označován jako publisher. Uzel přijímající zprávy se nazývá subscriber. Jeden uzel může zároveň vysílat i přijímat více zpráv.

Témata a zprávy

Uzly komunikují po tématech za pomoci zpráv. Téma neboli topic si lze představit jako kanál, do kterého publisher odešle zprávu. Tuto zprávu pak může jakýkoliv subscriber přijmout a případně zpracovat. Toto téma musí být definováno v každém uzlu, který chce pomocí tohoto tématu odesílat či přijímat zprávy. Pro lepší přehlednost je možné přidělit tématu jmenný prostor, pomocí kterého se zamezí nechtěným propojením ve velkých projektech. Zprávy, kterými uzly po tématech komunikují, mohou být různého typu. Nejčastějšími typy jsou `bool`, `string`, `int8`, `int32`, `float32` a další. Zprávy je možné strukturovat, ROS také nabízí již vytvořené struktury, které výrazně usnadňují práci. Příkladem může být struktura pro čas `time` nebo obrázek `sensor_msgs::ImagePtr`.

Catkin

Aby vývojáři ROSu nemuseli vytvářet vlastní systém pro překlad open-source programů, psaných pro robotické prostředí, využili jako základ již existující CMake. Catkin, používaný v ROSu, je pouze nástavba nad CMakem v podobě mnoha maker. Úlohou těchto maker je, aby se překládané balíčky ze zdrojových kódů v systému chovaly jako regulérně nainstalované binární balíčky. Jelikož je k překladu použit samotný CMake, je dosáhnuto multiplatformnosti bez většího úsilí. Aby mohl být balíček pomocí Catkinu přeložen, musí obsahovat konfigurační soubor pro CMake `CMakeLists.txt` a manifest `package.xml`, ve kterém jsou informace o balíčku, autorovi, licence a další.

Roslaunch

V případě větších projektu je často třeba spouštět několik uzlů najednou, ve velkých projektech může počet uzlů růst až do desítek. Pro tyto případy je v ROSu přítomen program *Roslaunch*, který umožňuje spouštět více uzlů v jeden okamžik. Definici uzlů a jejich parametry je třeba zadat do konfiguračního launch souboru, který se jako parametr zadá programu *Roslaunch* při spuštění. Uživatel si tak může předpřipravit řadu launch souborů s odlišnými uzly a parametry. Pro spuštění požadované varianty programu stačí vybrat příslušný launch soubor. Jelikož by byly i tyto soubory v rámci velkých projektů nepřehledné, *Roslaunch* umožňuje hierarchickou strukturu launch souborů. V rámci jednoho hlavního souboru je možné načíst více dílčích konfiguračních souborů, ze kterých jsou spuštěny požadované uzly, případně načteny další konfigurační soubory.

4.1.2 OpenCV

Se stále se rozvíjející oblastí počítačového vidění vznikla potřeba sjednotit mnohé algoritmy, které se v počítačovém vidění často využívají. Z toho důvodu vznikla pod záštitou firmy Intel v roce 1999 knihovna sdružující základní funkce a aplikace pro počítačové vidění. Motivací pro vytvoření této knihovny byl pro firmu Intel předpoklad, že se s dostupnějšími aplikacemi zvýší poptávka po výkonnějších procesorech. Dnes se tato knihovna využívá

v nejrůznějších výzkumných i komerčních institucích. Knihovna OpenCV je vydána pod BSD licencí a je zdarma pro komerční i akademické využití [23].

4.1.3 Qt

Nejen pro vývoj desktopových aplikací existuje již více než dvacet let volně dostupný framework Qt. Qt je multiplatformní, podporuje systémy Linux, OS X, Windows, Android, iOS a další [5]. Framework je napsán v programovacím jazyce C++ a slouží jako rozšíření pro tento jazyk. Před běžným překladem se používá metaobjektový překladač (Meta-Object Compiler), který analyzuje hlavičkové soubory a převádí kód do standardního C++, jenž může být přeložen běžnými překladači jako GCC, MinGW nebo MSVC. Společnost Trolltech, předchůdce Qt, byla založena v roce 1990. Od té doby projekt prodělal řadu změn, v současnosti se společnost nazývá Qt Company a jejím vlastníkem je finská společnost Digia Plc. Qt je dostupné pod řadou licencí, mezi nimi se nacházejí placené licence pro komerční využití i licence zdarma jako GPL či LGPL.

4.2 Klasifikace obsazenosti místa

Následuje stručný popis, jak a s použitím jakých dostupných prostředků byly implementovány nejvýznamnější části aplikace. Každý klasifikátor přijímá na vstupu zprávu obsahující hlavičku a snímek, výstupní zpráva má následující formát:

```
std_msgs/Header header        # hlavicka zpravy
int32 scene_id                # identifikator sceny
int32 detector_id             # identifikator detektoru
place_occupancy[] places      # seznam vsech parkovacich mist
      - int32 id              # identifikator mista
      - float32 rate          # mira obsazeni mista
```

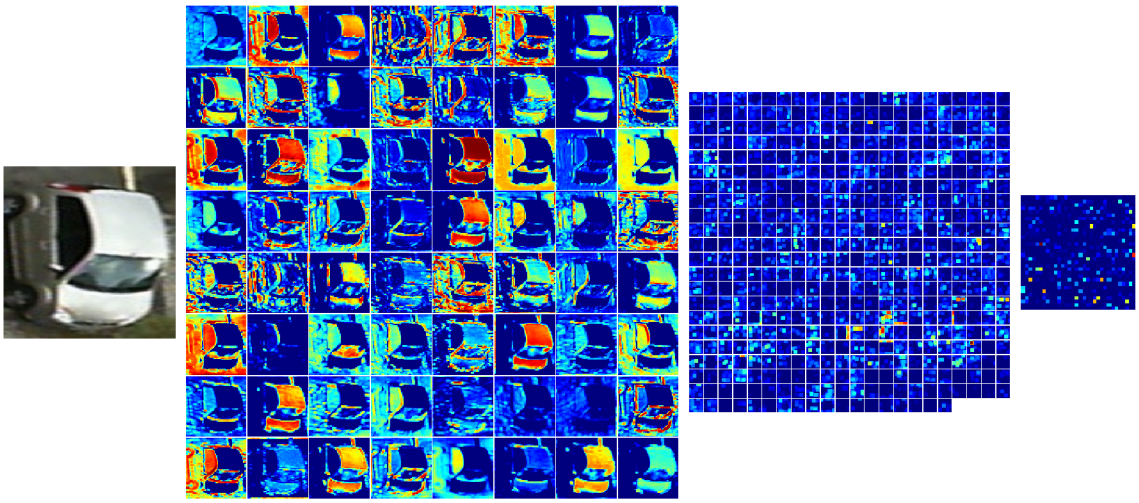
Header je hlavička obsahující identifikátor zprávy a časové razítko. Datový typ *int32* odpovídá běžně využívanému typu *integer*, obdobně *float32* je 32 bitový *float*. Typ *place_occupancy[]* je pole sdružující všechna parkovací místa, která byla právě klasifikována, a obsahuje pro každé místo dvojici lokálního identifikátoru tohoto místa a jeho míra obsazenosti vypočtená tímto klasifikátorem.

4.2.1 Klasifikace pomocí hluboké neuronové sítě

Pro práci s neuronovou sítí byl zvolen framework BVLC Caffe, který umožňuje rozhraní pro C++ i Python, podporuje externí definici vrstev neuronové sítě a je velmi optimalizovaný a tedy i rychlý. Pro rozhraní s tímto frameworkem byl zvolen jazyk C++. Pro trénování i počáteční testování úspěšnosti hluboké neuronové sítě byl použit nástroj *DIGITS* od *NVIDIA*, příklad možného výstupu tohoto nástroje lze vidět na obrázku 4.1. Z hlubokých neuronových sítí byla pro nejlepší výsledky na testech zvolena architektura *GoogLeNet* [37], která byla předtrénovaná na automobily a dotrénovaná na vlastní sadě. Vrstvy sítě již nebylo třeba upravovat. Na obrázku 4.1 je znázorněn vstup sítě a výstupy tří vybraných vrstev. Výstupy vrstev na začátku sítě jsou pro člověka stále interpretovatelné, čím je vrstva hlouběji zanořená, tím je její výstup méně zřejmý. V systému jsou použity dvě výstupní třídy, jedna třída pro volné parkovací místo, druhá pro obsazené místo libovolným vozidlem. Jelikož tříd není víc, jsou hodnoty v těchto dvou třídách komplementární. Výstupem sítě je pravděpodobnost $< 0, 1 >$, s jakou je na testovacím vzorku obsazené parkovací

místo. Pokud je tedy výstupem hodnota 0, síť si je jista, že je místo volné. Hodnotou 1 síť vyjadřuje, že je místo jistě obsazené. Jakákoliv hodnota uvnitř intervalu připouští s určitou pravděpodobností obě varianty, čehož je využito v případě více detektorů či více pohledů kamer na testované místo. Rozhodovací práh je možné upravit, výchozí a ověřená hodnota je nastavena na 0,5.

Framework Caffe umožňuje výpočet na grafické kartě i na procesoru, obě varianty byly použity a zahrnuty do systému. Pokud se klasifikace počítá na grafické kartě, jsou odeslány všechny parkovací místa ke klasifikaci naráz, v případě procesoru jsou kvůli výrazně vyšší (padesátinásobné) době výpočtu odesílány parkovací místa po jednom.



Obrázek 4.1: Vstup a tři vybrané výstupy jednotlivých vrstev neuronové sítě

4.2.2 Klasifikace pomocí modelu pozadí

Při vývoji uzlu s modelem pozadí bylo experimentováno s knihovnou BGSLibrary [34]. Jelikož je systém třeba při dalším nasazování instalovat na více zařízeních a s integrací této knihovny do systému byly problémy, byla místo ní využita funkce z již používané knihovny OpenCV. Model pozadí je realizován směsí Gaussových rozložení prostřednictvím modulu `bgfg_gassmix2`, který pro potřeby aplikace dostatečně splňuje úspěšnost.

Funkcionalita byla upravena tak, aby umožňovala dva režimy. První režim je aplikován na všechna volná parkovací místa (a při spuštění systému pro místa, u nichž zatím není známo pozadí) a zajišťuje rychlou aktualizaci modelu pozadí. Tím je zajištěna robustnost vůči změnám světelných podmínek, neboť je model aktualizován rychleji, než se mění okolní podmínky. Rychlost aktualizace byla nastavena tak, aby se model aktualizoval co nejrychleji, ale přitom byl schopný detekovat rychlou změnu přijíždějícího vozidla. Druhý režim se použije pro obsazená parkovací místa. V tomto případě není model pozadí aktualizován vůbec, aby bylo stále možné porovnávat stojící vozidlo s vozovkou a v případě odjezdu vozidla tuto shodu detekovat. Pro výběr režimu je do funkce s modelem pozadí vložena maska označující volná a obsazená parkovací místa.

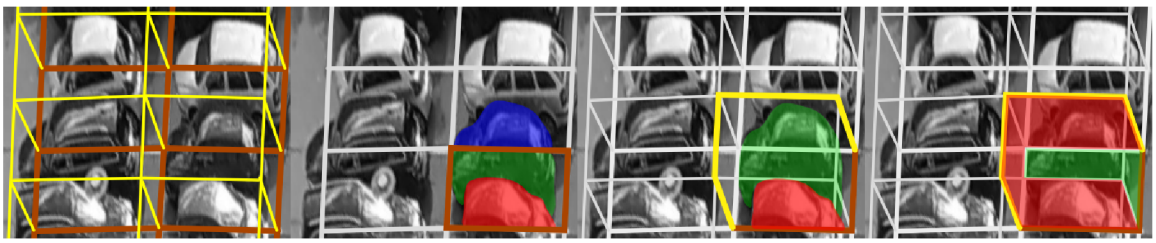
4.3 Označení parkovacích míst

Jelikož je systém vyvíjen pro statické kamery a parkoviště, na nichž se poloha parkovacích míst neliší, je možné označit parkovací místa před začátkem klasifikace. Tím se z detekce automobilů v obraze stává jednodušší problém klasifikace obsazenosti konkrétního místa.

Protože kamera není umístěna kolmo nad parkovacím místem, používá se označení místa simulující 3D. Nejprve uživatel ohraničí celé parkoviště a zadá rozměry v reálném světě, z čehož lze spočítat umístění kamery. Poté ohraničí každé parkovací místo čtyřmi body a výškou a systém spočítá 3D model parkovacího místa na snímku (obrázek 4.2). Hnědé čáry označují podstavy parkovacích míst označené uživatelem. Žluté čáry jsou na základě informací o výšce a pozici kamery automaticky dopočítány a představují 3D model parkovacího místa.

Obrázek 4.2: Označení parkovacích míst

Výhody takovéhoto způsobu označení jsou představeny na obrázku 4.3. Zde lze vidět výřez ze snímku 4.2 se čtyřmi parkovacími místy (4.3a). Obrázky 4.3b a 4.3c srovnávají problémy při označení pouze ve 2D a v simulovaném 3D. Pochopitelně by mohl uživatel označit místo ve 2D větším obdélníkem pokrývajícím celý automobil. Takový přístup by však byl komplikovanější, pokud by uživatel označoval snímek prázdného parkoviště. Další výhoda představeného označování je v uchování informace o podstavě místa na vozovce pro pozdější vizualizaci. Především je pak možné extrahovat konfliktní oblasti, ve kterých se parkovací místa překrývají a může zde snadno dojít k mylné interpretaci pohybu (obrázek 4.3d).

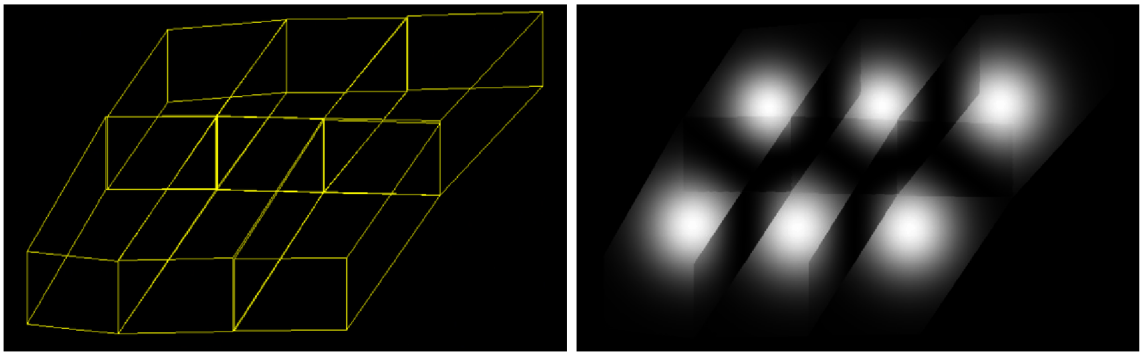


(a) (b) (c) (d)

Obrázek 4.3: Výřezy označení parkovacích míst a ukázka rozdílů 2D a 3D.
(a) označení ve 3D
(b) 2D, zelená - zaznamenaná část, modrá - nezaznamenaná část, červená - okluze
(c) 3D, zelená - zaznamenaná část automobilu, červená - okluze
(d) odstranění překrývajících se ploch, zelená - nepřekrývající se, červená - překrývající se

V hluboké neuronové síti se označení používá pro vyříznutí trénovacích vzorků (obrázek 4.4) a následně pro samotnou klasifikaci, při které jsou vstupy právě tyto výřezy. Pro klasifikaci modelem pozadí je navíc na vyříznuté parkovací místo aplikována maska, která odstíní oblasti překryvů sousedních parkovacích míst (obrázek 4.5). Maska je spočítána pomocí Gaussova rozložení, což dobře aproximuje pravděpodobnost, s jakou bude na daném pixelu přítomný automobil. Výslednou masku lze získat vzájemným odečtením těchto samostatných masek, čímž se výrazně eliminují překrývající se oblasti.

Obrázek 4.4: Ukázky výřezů míst (8 obsazených a 8 volných)



Obrázek 4.5: Kompozice šesti masek parkovacích míst

4.4 Použitý hardware

Aplikace byla vyvíjena na notebooku Lenovo K50i s procesorem *Intel Core i7-4702MQ CPU @ 2.20GHz*, dále popisované testy na procesoru byly spouštěny na tomto stroji. Stolní počítač, na němž je spuštěno nepřetržité testování, obsahuje procesor *Intel Core i5-6400 CPU @ 2.70GHz*. K tomuto počítači jsou připojeny dvě 5MP IP kamery *Camera Bullet*

ip66, z nichž jsou získávány aktuální snímky. Neuronová síť byla trénována a testována na grafické kartě *GeForce GTX 1080*. Aplikace byla navíc testována na vestavěném zařízení od NVIDIA s grafickou kartou Maxwell a podporou CUDA.

4.5 Shrnutí

Kapitola představila použitý Robotický operační systém a jeho základní koncept a stručně popsala další použité softwarové nástroje. V druhé části kapitoly byly uvedeny podrobnosti týkající se implementace dvou hlavních klasifikátorů a byl uveden způsob označování parkovacích míst a jeho výhody. Závěrem byly zmíněny hardwarové prostředky použité k vývoji i testování.

Kapitola 5

Testování

Celý systém, jeho kompletní funkčnost a provázanost samostatných uzlů, byly testovány na datových sadách z testovacího parkoviště v Dánsku, které je popsáno dále. Pro jednoduchou interpretaci získaných dat byly implementovány i základní vizualizační prvky, jako je například graf historie obsazenosti celého parkoviště (viz obrázek 5.3). Ukázku označení výsledků klasifikace na snímcích z kamery lze vidět na obrázku 5.4. Samotnou klasifikaci obsazenosti jednotlivých parkovacích míst bylo možné testovat kromě na záznamech z parkoviště v Dánsku i na dostupné datové sadě *PKLot*. Tato datová sada obsahuje pouze snímky parkoviště pořízené s delšími časovými intervaly, nikoliv navazující videozáznam. Nebylo na ní tedy možné testovat celkový provoz systému ani klasifikátor s modelem pozadí detekující pohyb. Jiné dostupné datové sady snímků parkoviště nalezeny nebyly.

Tato kapitola se věnuje především testování úspěšnosti systému, která je velmi úzce spojena s úspěšností samotné hluboké neuronové sítě. Kapitola obsahuje několik vzájemně odlišených testů. Závěrem jsou představeny nejčastější chyby a jejich navrhované řešení.

Parkoviště v Dánsku

Z reálného parkoviště v Dánsku byly získány záznamy o celkové délce 32 hodin. Toto parkoviště bylo snímáno dvěma kamerami a obsahovalo 193 parkovacích míst, viz obrázek 5.1. 16 parkovacích míst bylo k dispozici na obou pohledech z připojených kamer. Aplikace nepřetržitě (cca dva měsíce) vyhodnocuje obsazenost zmíněného testovacího parkoviště, aby byly odhaleny i nepředvídatelné změny. Pomocí nepřetržitého provozu byly odhaleny a následně opraveny některé zřídka se vyskytující chyby, např. občasné poškození snímků z kamery nebo problémy s připojením k datovému serveru.

Datová sada PKLot

Kromě popsaného testovacího parkoviště v Dánsku byl systém testován na dostupné datové sadě *PKLot* [14]. Tato databáze obsahuje 12417 snímků zachycených ze dvou různých parkovišť, z nichž jedno parkoviště snímaly dvě kamery. Databáze je hierarchicky rozdělena do tří složek podle kamery, každá z nich se dále dělí podle počasí na snímcích (slunečno, oblačno, déšť). Ukázky pohledů kamer za různého počasí lze vidět na obrázku 5.2. Po rozdělení snímků na jednotlivá parkovací místa databáze obsahuje přibližně 695900 vzorků. Databázi je možné volně používat s podmínkou jejího citování.

Obrázek 5.1: Ukázka pohledů dvou kamer na testovací parkoviště v Dánsku.

5.1 Vliv deformací na úspěšnost

První test zkoumal vliv odstranění deformací z obrazu na úspěšnost klasifikace. Pro vzory bez odstraněných deformací byla získána úspěšnost klasifikace na dvou datových sadách z trénovacího parkoviště — jež jsou popsány v dalším testu — 95,23 % a 94,18 %, pro vzory s odstraněnými deformacemi úspěšnost 99,20 % a 99,01 %. Další testy se proto věnovaly výhradně vzorům s odstraněnými deformacemi.

5.2 Porovnání existujících architektur

Byly vybrány dvě předtrénované architektury, jejichž úspěšnost byla nejlepší, a dále byly upraveny pro různou velikost vstupů. Těmito architekturami byla konvoluční neuronová síť *GoogLeNet* s velikostmi vstupů 32×32, 64×64 a 224×224 pixelů a reziduální neuronová síť *ResNet* s velikostí vstupů 256×256 pixelů. (Vyřazeny byly např. sítě *GoogLeNet256* či *VGG16-32*.) *GoogLeNet* obsahuje 22 vrstev s parametry (27 včetně slučovacích vrstev), *ResNet-50* se skládá z 50 vrstev. Druhý test zjišťoval úspěšnost vybraných sítí natrénovaných a testovaných na vzorcích s nejlepším rozlišením, kterého je v systému možné dosáhnout. Byla vytvořena datová sada pro dotrénování, obsahující 7376 trénovacích vzorů (3791 obsazených míst a 3585 volných míst), a dvě odlišné datové sady pro testování, první s 1881 vzory (1181 obsazených míst a 700 volných míst) a druhá s 1802 vzory (1020 obsazených míst a 782 volných míst). Velikosti vzorků se pohybovaly v rozmezí od 120×80 do 130×130. Data byla pořízena ze dvou kamer (5mpx, rozlišení 2592×1920) testovacího parkoviště v různé časové doby, za měnících se světelných podmínek i počasí ve 25 různých dnech. Vždy byla vybrána trénovací epocha s nejlepším výsledkem na testovací sadě. Tabulka 5.1 shrnuje úspěšnost klasifikace a rychlost klasifikace jednoho parkovacího místa na CPU (*Intel Core i7-4702MQ CPU @ 2.20GHz*) a na GPU (*GeForce GTX 1080*). Na základě výsledků byla vybrána a použita v aplikaci nejúspěšnější síť *GoogLeNet* 224×224, v případě potřeby vyšší rychlosti by bylo vhodnější použít *GoogLeNet* s menší velikostí vzorů.

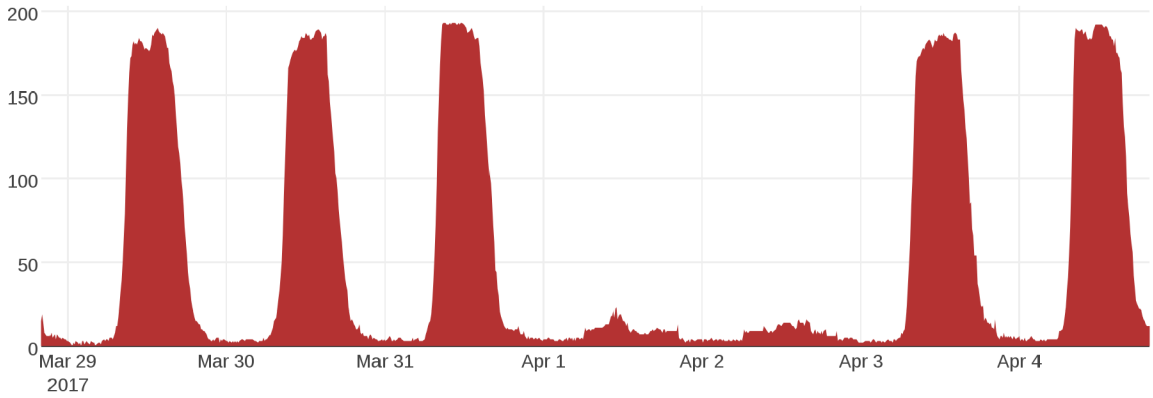
Obrázek 5.2: Snímky z datové sady PKLot. Řádky odlišují různé kamery, sloupce odlišují počasí (oblačno, slunečno, déšť)

název sítě	velikost vzoru	rychlost CPU	rychlost GPU	úspěšnost sada 1	úspěšnost sada 2
GoogLeNet	32×32	1,1s	3,7ms	98,83 %	99,01 %
GoogLeNet	64×64	1,2s	5,3ms	99,10 %	98,90 %
GoogLeNet	224×224	1,4s	11,7ms	99,64 %	98,96 %
ResNet	256×256	1,7s	13,3ms	97,18 %	92,75 %

Tabulka 5.1: Srovnání hlubokých neuronových sítí

5.3 Vliv zmenšení vstupů na úspěšnost

Třetí test zjišťoval vliv zmenšení a opětovného zvětšení velikosti trénovacích i testovacích vzorů na úspěšnost dvou nejlepších sítí z předchozího testu. Snahou bylo zjistit minimální velikost vzorů, pro kterou má systém nejvyšší úspěšnost. Na základě získaných výsledků je možné odhadnout počet parkovacích míst, které je schopna snímat jedna kamera. Čím menší vzorek je síť stále schopna klasifikovat, tím více míst může být obsaženo na jednom snímku. Z grafu 5.5 si lze všimnout, že je možné zmenšit vzory až na velikost 32×32 pixelů a stále bude mít síť *GoogLeNet* 224×224 úspěšnost vyšší než 99,5 %. Při dalším zmenšení je již úspěšnější síť *GoogLeNet* 64×64. Zajímavostí je úspěšnost téměř 96 % pro vzory o velikosti 4×4 pixely. Tak vysoké úspěšnosti je možné dosáhnout především díky tomu, že je trénovací i testovací sada pořízena pouze z jednoho parkoviště a je tedy pravděpodobné, že síť není dostatečně obecná.



Obrázek 5.3: Graf historie obsazenosti testovacího parkoviště (od středy do úterý)

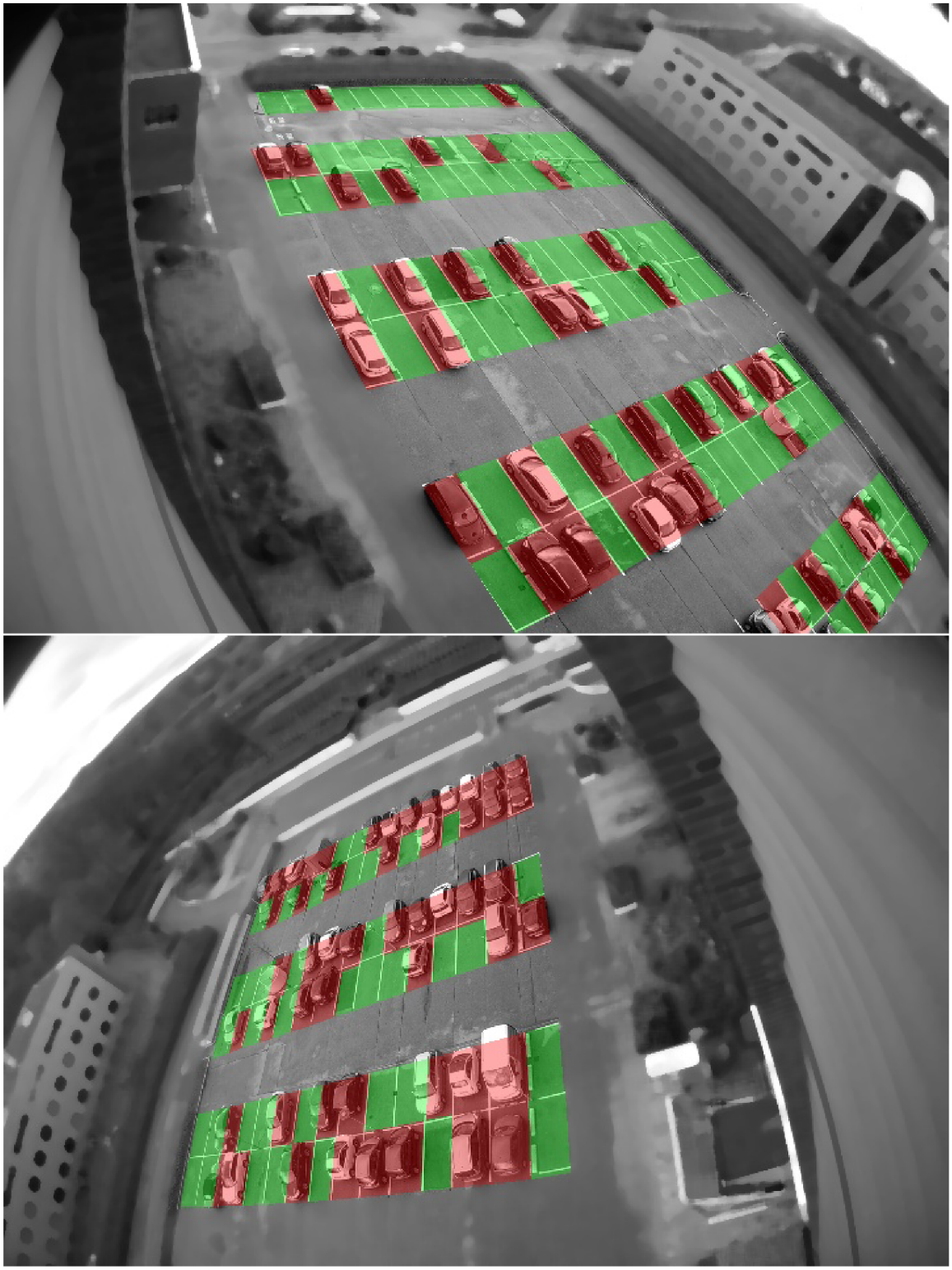
5.4 Úspěšnost během různých podmínek

Čtvrtý test zjišťoval vliv různých okolních podmínek na úspěšnost klasifikace použitou sítí *GoogLeNet* 64 × 64. Test byl rozdělen na dvě části, každá testovala rozdílné podmínky. První část testovala úspěšnost na testovacím parkovišti v Dánsku, neboť bylo možné získat záznamy s velmi se lišícími podmínkami. Testovanými podmínkami bylo oblačno, slunečno, mlha, déšť, slunečno po dešti, šero a tma. Vzhledem k poloze parkoviště se nepodařilo získat záznamy se sněhem. Datová sada ale bohužel není dostatečně obsáhlá, proto byla druhá část testu provedena na datové sadě *PKLot*. *PKLot* rozděluje vzorky na tři různé podmínky, které byly testovány samostatně (oblačno, slunečno, déšť). Byla natrénována další neuronová síť *GoogLeNet*, jejímž vstupem pro trénování byla datová sada obsahující 54360 vzorků (35571 volných míst a 18789 obsazených). Výsledky obou částí testu shrnují tabulky 5.2 a 5.3.

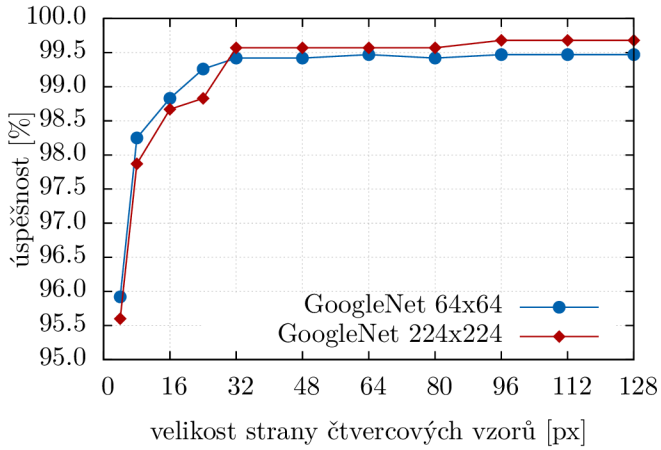
Ovlivňující podmínky	počet vzorků dat	úspěšnost
Oblačno	2123	99,74 %
Slunečno	1447	98,06 %
Mlha	290	99,12 %
Déšť	579	99,09 %
Slunečno po dešti	386	97,63 %
Šero	675	98,24 %
Tma	833	93,34 %
Celkem	**6333**	**98,01 %**

Tabulka 5.2: Úspěšnost při různých podmínkách (parkoviště v Dánsku)

Z těchto tabulek je zřejmé, že nejlepší úspěšnosti dosahuje hluboká neuronová síť během oblačnosti, při níž se na parkovišti nevyskytují žádné stíny ani jiné rušivé elementy. Stále velmi dobré úspěšnosti lze dosáhnout i při dešti a mlze, kdy je sice lehce zhoršena viditelnost, avšak ne natolik, aby to výrazně ovlivnilo klasifikaci. Během slunečného dne se na parkovišti vyskytují ostré stíny, které mají na svědomí chybné klasifikace. Po dešti se navíc kromě stínů vyskytují na místech, kde dříve stály automobily, mokré části vozovky tvarem připomínající automobily. Úspěšnost při takto stížených podmínkách se však stále jeví jako dostatečná.



Obrázek 5.4: Pohledy z obou kamer na testovací parkoviště v Dánsku s označením obsazenosti získané z klasifikátorů



Obrázek 5.5: Závislost úspěšnosti na velikosti vzorů

Velmi špatné klasifikace je dosaženo za velké tmy. Nutno však podotknout, že je testovací parkoviště částečně osvětleno umělým osvětlením, proto i tato klasifikace přesahuje 90 %.

Druhá část testu kromě ovlivňujících podmínek porovnávala také úspěšnost dvou sítí natrénovaných na odlišných datech. První síť byla natrénována na datech z parkoviště v Dánsku, druhá síť na datech z datové sady *PKLot*. Sítě pak byly testovány odlišnými daty z datové sady *PKLot*. Z tabulek 5.2 a 5.3 vyplývá, že je nutné natrénovat vlastní síť pro každou skupinu parkovišť zvlášť. Úspěšnost na datových sadách ze stejného, nebo odlišného parkoviště než na jaké sadě byla síť natrénována se může lišit i o více než 15 %. Vzhledem k povaze zadání ale zatím není nutné vytvářet jeden společný model pro všechna parkoviště.

Ovlivňující podmínky	Kamera*	počet vzorků dat	úspěšnost síť Dánsko	úspěšnost síť PKLot
Oblačno	PUC	10000	84,08 %	99,96 %
	UFPR04	10000	90,87 %	99,94 %
	UFPR05	10000	81,99 %	99,74 %
Slunečno	PUC	10000	87,12 %	99,85 %
	UFPR04	10000	85,12 %	97,96 %
	UFPR05	10000	81,51 %	99,47 %
Déšť	PUC	10000	86,93 %	99,97 %
	UFPR04	10000	86,29 %	99,96 %
	UFPR05	10000	79,16 %	99,93 %
Celkem	—	**90000**	**84,79 %**	**99,64 %**

* zkratky PUC, UFPR04 a UFPR05 jsou převzaty z databáze. UFPR04 a UFPR05 snímají stejné parkoviště z různých úhlů.

Tabulka 5.3: Úspěšnost při různých podmínkách (datová sada *PKLot*)

5.5 Časté chyby a jejich řešení

Systém pochopitelně není bezchybný, na základě testování byly vybrány tři nejčastější případy chybné klasifikace. Poté byly navrženy způsoby jak tyto chybné klasifikace vyřešit. Příklady problémových situací lze vidět na obrázku 5.6.

Obrázek 5.6: Časté chyby: okluze, nedostatečné světelné podmínky, ostré stíny

Okluze: Problém s překrývajícími se automobily závisí především na výšce nad zemí, ve které je umístěna kamera. Minimální výška se liší pro různé typy vozidel a vzdálenost kamery od parkovacích míst. Obecně lze říci, že je třeba umístit kameru v minimální výšce 12 metrů nad zemí. Požadovaná výška závisí i na natočení parkovacích míst, neboť pohled z boku vyžaduje vyšší výšku umístění kamery než pohled zepředu nebo zezadu. Pokud není možné umístit kameru dostatečně vysoko, doporučuje se přidat další kameru, která bude snímat parkovací místo z jiného úhlu. Tím se výrazně zredukuje počet chyb způsobených nedostatečně viditelným automobilem.

Nedostatečné světelné podmínky: Aplikace funguje dobře za zhoršených viditelnostních podmínek, např. za šera. V úplné tmě je klasifikace však prakticky nemožná a nenabízí se žádné softwarové řešení. Snadnější avšak méně účinným řešením je infračervené přisvícení kamery. Pokud bude použito, je potřeba přidat snímky s tímto přisvětlením do trénovací sady pro neuronovou síť, v ideálním případě potom natrénovat speciální síť na noční klasifikaci. Jelikož má infračervené přisvětlení kamery většinou dosah nejvýše 50 metrů, není toto řešení příliš úspěšné. Úspěšnějším avšak nákladnějším řešením je osvětlení celého parkoviště lampami a zajištění dobré viditelnosti pro všechny použité kamery.

Ostré stíny: Třetí častou chybou bývá klasifikace prázdných parkovacích míst, na nichž jsou ostré stíny připomínající automobil. Toto je již problém softwaru a je ho možné řešit. Nejjednodušším způsobem je přidání dostatečného množství vzorků se stíny do trénovací datové sady. Datová sada je průběžně doplňována a chyby se daří eliminovat. Pokud by chyby způsobené stíny stále přetrvávaly, lze použít pokročilejších technik pro odstranění stínů z obrazu před samotnou klasifikací.

5.6 Shrnutí

Kapitola představila testovací parkoviště a testovací datovou sadu *PKLot*. Popsala čtyři provedené testy, které měřili především úspěšnost klasifikace pomocí hluboké neuronové sítě, která hraje klíčovou v roli v úspěšnosti celého systému. Při testování se projevily tři nejčastější chyby. Tyto chyby byly na závěr kapitoly popsány a bylo navrženo jejich možné řešení.

Kapitola 6

Závěr

Práce představila existující principy detekce vozidel v obraze ze statické kamery, podrobně se věnovala modelům pozadí a hlubokým neuronovým sítím a představila existující řešení tohoto tématu. Popsala systém pro klasifikaci obsazenosti parkovacích míst z více-kamerového systému a představila dvě základní metody, které jsou použity: hlubokou neuronovou síť a klasifikaci pomocí modelu pozadí. Klasifikátor s konvoluční hlubokou neuronovou sítí využívá již existující sít *GoogLeNet*, která byla z dostupných sítí vybrána pro nejlepší úspěšnost. Klasifikátor s modelem pozadí neprovádí konečné rozhodnutí o obsazenosti, ale označuje ta parkovací místa, která je třeba oklasifikovat nejdříve. Základní kostrou aplikace se stal koncept Robotického operačního systému, díky němuž je aplikace rozdělena do samostatných uzlů vzájemně komunikujících prostřednictvím zpráv. Tento koncept umožňuje variabilitu aplikace a snadné přidání nového klasifikátoru či jiného uzlu. Aplikace je navržena s ohledem na více-kamerový systém, jehož pohledy z různých kamer se mohou překrývat. Při návrhu a vývoji aplikace byl kladen důraz především na rychlý výpočet pro běh v reálném čase a na robustnost vůči klimatickým podmínkám, jako je změna osvětlení, změna počasí a další. Systém je navržen a implementován tak, aby byl schopný aktualizovat změny obsazenosti každou vteřinu, a to při spuštění na grafické kartě i procesoru.

Výsledná aplikace byla otestována na testovacím parkovišti a dostupné databázi *PKLot*. Na datové sadě z testovacího parkoviště dosahuje velmi dobré úspěšnosti 99.6 %, funkčnost celého systému je testována v reálném provozu. Testováním byla potvrzena robustnost systému vůči většině měnících se světelných a viditelnostních podmínek. Byly detekovány nejčastější chyby, při nichž systém selhává a bylo navrženo jejich možné řešení.

Integrace řešení detekce obsazenosti parkovacích míst do vyššího celku usnadní řidičům vyhledání volného parkovacího místa, omezí nežádoucí provoz po parkovišti a v neposlední řadě sníží produkci CO_2. Zřizovatelům parkoviště pak může nabídnout řadu informací o době obsazenosti, vytíženosti konkrétních míst a další. Instalací popisovaného systému se předpokládá, že se parkoviště stane lukrativnějším a přiláká více zákazníků. Navíc zjednoduší vyhledání těžko dostupných volných parkovacích míst, čímž zvýší využitelnost parkoviště.

Práce byla prezentována na studentské konferenci *Excel@FIT 2017* a získala čtyři ocenění. Soutěžní článek i poster je přiložen jako příloha C, respektive D. Poté byla anglická verze článku přijata i na konferenci *SCCG 2017*. Anglickou verzi článku pro *SCCG 2017* lze nalézt jako přílohu E. Kromě konferencí se práce objevila také v řadě médií.

6.1 Následující vývoj

V pokračujícím vývoji bude snaha o zvýšení úspěšnosti zvětšováním trénovací sady a zobecnění natrénovaných modelů přidáním dat z dalších parkovišť. Na základě externích požadavků bude přidána další funkčnost aplikace, např. předpokládaný čas uvolnění parkovacího místa, pokročilé statistiky o využití parkoviště a další. Systém bude dále optimalizován pro fungování na vestavěném systému s omezenými výpočetními zdroji. Pro reálné nasazení bude potřeba přidat rozhraní pro vizualizační tabule, které mají být umístěny na parkovišti. Zmiňovaným rozhraním může být i mobilní telefon, pro nějž by bylo vhodné vytvořit uživatelsky přívětivou webovou stránku nebo celou mobilní aplikaci. Mezi navrhovanými rozšířeními se nevyskytuje přidání dalšího klasifikátoru, neboť se hluboká konvoluční neuronová síť ukázala jako jedna z nejúspěšnějších. Pro další rozvoj je doporučeno zaměřit se na další zvýšení úspěšnosti této sítě, případně nahrazením architektury za novou, vznikne-li v následujících letech úspěšnější než zde použitá *GoogLeNet*.

Výhledově by bylo zajímavé zbavit systém závislosti na označování parkovacích míst. Systém by detekoval všechna vozidla na snímku a automaticky počítal volný prostor mezi nimi. Možným řešením by bylo označení oblasti snímku, např. kraj vozovky, na níž by měl systém vozidla detekovat. Využití lze nalézt nejen pro kontrolu přítomnosti automobilů v zakázané oblasti. Kritickým bodem tohoto přístupu bude doba výpočtu, neboť je detekce výpočetně o mnoho náročnější než zde popsaná klasifikace. Dalším rozšířením může být kromě klasifikace obsazenosti také určení typu přítomného vozidla (osobní automobil, nákladní automobil, autobus, motorka aj.). Pokud budou odstraněny definice parkovacích míst, lze aplikaci upravit pro podporu pohyblivé kamery. Kamera může být např. motorická, jež odstraní nutnost více kamer natočených pod různými úhly. Jinou variantou může být dron, létající nad parkovištěm. V případě dronu by bylo třeba pokročilejší stabilizace obrazu a využití neuronové sítě, která by byla schopna detekovat automobil z různé výšky.

Jelikož mě téma velmi zaujalo a mám možnost na vývoji v rámci firmy dále pracovat, chtěl bych alespoň část z navržených rozšíření v blízké době implementovat.

Literatura

[1] *Is ROS For Me?* [Online; navštíveno 25.10.2016].
URL http://www.ros.org/is-ros-for-me/

[2] Al-Kharusi, H.; Al-Bahadly, I.: *Intelligent Parking Space Detection System Based on Image Processing. World Journal of Engineering and Technology*, 2014, doi:10.4236/wjet.2014.22006.

[3] Amato, G.; Carrara, F.; Falchi, F.; aj.: Car parking occupancy detection using smart camera networks and Deep Learning. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, June 2016, s. 1212–1217, doi:10.1109/ISCC.2016.7543901.

[4] Barnich, O.; Droogenbroeck, M. V.: *ViBe: A Universal Background Subtraction Algorithm for Video Sequences. IEEE Transactions on Image Processing*, 2010, ISSN 1941-0042.

[5] Blanchette, J.; Summerfield, M.: *C++ GUI Programming with Qt 4*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006, ISBN 0131872494.

[6] Canziani, A.; Paszke, A.; Culurciello, E.: An Analysis of Deep Neural Network Models for Practical Applications. *CoRR*, ročník abs/1605.07678, 2016.
URL http://arxiv.org/abs/1605.07678

[7] Cao, J.; Wang, J.: Global asymptotic and robust stability of recurrent neural networks with time delays. *IEEE Transactions on Circuits and Systems I: Regular Papers*, ročník 52, č. 2, 2005: s. 417–426.

[8] Chauvin, Y.; Rumelhart, D. E.: *Backpropagation: Theory, Architectures, and Applications*. Psychology Press, 1995, ISBN 978-0805812596.

[9] Chen, X.; Xiang, S.; Liu, C. L.; aj.: *Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks. IEEE Geoscience and remote sensing letters*, 2014, ISSN 1558-0571.

[10] Cucchiara, R.; Grana, C.; Piccardi, M.; aj.: *Detecting moving objects, ghosts, and shadows in video streams. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003, ISSN 0162-8828.

[11] Culibrk, D.; Marques, O.; Socek, D.; aj.: *Neural Network Approach to Background Modeling for Video Object Segmentation. IEEE Transactions on Neural Networks*, 2007, ISSN 1941-0093.

[12] Culurciello, E.: Neural Network Architectures. [Online; navštíveno 29.4.2017].
URL https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

[13] Cybenko, G.: Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, ročník 2, č. 4, 1989: s. 303–314.

[14] De Almeida, P. R.; Oliveira, L. S.; Britto, A. S.; aj.: PKLot–A robust dataset for parking lot classification. *Expert Systems with Applications*, ročník 42, č. 11, 2015: s. 4937–4949.

[15] Dokur, O.; Katkoori, S.; Elmehraz, N.: Embedded system design of a real-time parking guidance system. In *2016 Annual IEEE Systems Conference (SysCon)*, April 2016, s. 1–8, doi:10.1109/SYSCON.2016.7490653.

[16] Elgammal, A.; Hanvood, D.; Davis, L.: *Non-parametric model for background subtraction. 6th European Conference on Computer Vision Dublin, Ireland, June 26–July 1, 2000 Proceedings, Part II*, 2003, ISSN 0302-9743.

[17] Fusek, R.; Mozdřeň, K.; Šurkala, M.; aj.: *AdaBoost for Parking Lot Occupation Detection. International Conference on Systems, Man and Cybernetics*, 2013.

[18] Fusek, R.; Sojka, E.; Šurkala, K. M. M.: *Energy based Descriptors and their Application for Car Detection. Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, 2014.

[19] Gomez-Ol, R. G.: Deep Learning frameworks: a review before finishing 2016. [Online; navštíveno 1.5.2017].
URL https://medium.com/@ricardo.guerrero/deep-learning-frameworks-a-review-before-finishing-2016-5b3ab4010b06

[20] Gurney, K.: *An Introduction to Neural Networks*. Taylor & Francis, 2003, ISBN 9780203451519.
URL https://books.google.cz/books?id=sn6oBHq8qQQC

[21] Hartley, R.; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Cambridge books online, Cambridge University Press, 2003, ISBN 9780521540513.
URL https://books.google.cz/books?id=si3R3Pfa98QC

[22] Jermsurawong, J.; Ahsan, M. U.; Haidar, A.; aj.: Car Parking Vacancy Detection and Its Application in 24-Hour Statistical Analysis. In *2012 10th International Conference on Frontiers of Information Technology*, Dec 2012, s. 84–90, doi:10.1109/FIT.2012.24.

[23] Kaehler, A.; Bradski, G.: *Learning OpenCV 3*. O'Reilly Media, 2016, ISBN 978-1491937990.

[24] Koubaa, A.: *Robot Operating System (ROS): The Complete Reference*. číslo v. 1 in Studies in Computational Intelligence, Springer International Publishing, 2016, ISBN 9783319260549.
URL https://books.google.cz/books?id=wY2RCwAAQBAJ

[25] Larochelle, H.; Bengio, Y.; Louradour, J.; aj.: *Exploring Strategies for Training Deep Neural Networks. The Journal of Machine Learning Research*, 2009.

[26] Maria, G.; Baccaglini, E.; Brevi, D.; aj.: *A drone-based image processing system for car detection in a smart transport infrastructure. Proceedings of the 18th Mediterranean Electrotechnical Conference*, 2016.

[27] Naba, A.; Pratama, B. M.; Nadhir, A.; aj.: *Haar-like feature based real-time neuro car detection system. International Seminar on Sensors, Instrumentation, Measurement and Metrology (ISSIMM)*, 2016.

[28] Piccardi, M.: *Background subtraction techniques: a review*. International Conference on Systems, Man and Cybernetics*, 2004.

[29] Prasad, A. S. G.; Sharath, U.; Amith, B.; aj.: Fiber Bragg Grating sensor instrumentation for parking space occupancy management. In *2012 International Conference on Optical Engineering (ICOE)*, July 2012, s. 1–4, doi:10.1109/ICOE.2012.6409571.

[30] Quigley, M.; Conley, K.; Gerkey, B. P.; aj.: ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[31] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi:10.1007/s11263-015-0816-y.

[32] Seki, M.; Wada, T.; Fujiwara, H.; aj.: *Background subtraction based on cooccurrence of image variations. Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, ISSN 1063-6919.

[33] Sifuentes, E.; Casas, O.; Pallas-Areny, R.: Wireless Magnetic Sensor Node for Vehicle Detection With Optical Wake-Up. *IEEE Sensors Journal*, ročník 11, č. 8, Aug 2011: s. 1669–1676, ISSN 1530-437X, doi:10.1109/JSEN.2010.2103937.

[34] Sobral, A.: BGSLibrary: An OpenCV C++ Background Subtraction Library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013. URL https://github.com/andrewssobral/bgslibrary

[35] Stauffer, C.; Grimson, W.: *Adaptive background mixture models for real-time tracking. Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, 1999, ISSN 1063-6919.

[36] Stránský, V.: *Analýza obsazenosti parkoviště*. Diplomová práce, FIT VUT v Brně, Brno, 2014.

[37] Szegedy, C.; Liu, W.; Jia, Y.; aj.: Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, s. 1–9.

[38] Thomas, D.: *ROS / Introduction*. [Online; navštíveno 27.10.2016]. URL http://wiki.ros.org/ROS/Introduction

[39] Wren, C.; Azarhayejani, A.; Darrell, T.; aj.: *Pfinder: real-time tracking of the human body. Pattern Analysis and Machine Intelligence*, 1997.

[40] Wu, K.; Zhang, H.; Xu, T.; aj.: *Overview of Video-Based Vehicle Detection Technologies. The 6th International Conference on Computer Science & Education*, 2011.

[41] Yer, S.: *Robots Using ROS*. [Online; navštíveno 30.10.2016].
URL http://wiki.ros.org/Robots

[42] Yusnita, R.; Norbaya, F.; Basharuddin, N.: Intelligent Parking Space Detection System Based onImage Processing. *International Journal of Innovation, Management and Technology*, ročník 3, č. 3, 2012: str. 232.

[43] Zhang, Z.; Tao, M.; Yuan, H.: A Parking Occupancy Detection Algorithm Based on AMR Sensor. *IEEE Sensors Journal*, ročník 15, č. 2, Feb 2015: s. 1261–1269, ISSN 1530-437X, doi:10.1109/JSEN.2014.2362122.

[44] Zhou, H.; Li, Z.: An Intelligent Parking Management System Based on RS485 and RFID. In *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct 2016, s. 355–359, doi:10.1109/CyberC.2016.74.

[45] Šolić, P.; Marasović, I.; Stefanizzi, M. L.; aj.: RFID-based efficient method for parking slot car detection. In *2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sept 2015, s. 108–112, doi:10.1109/SOFTCOM.2015.7314120.

Příloha A

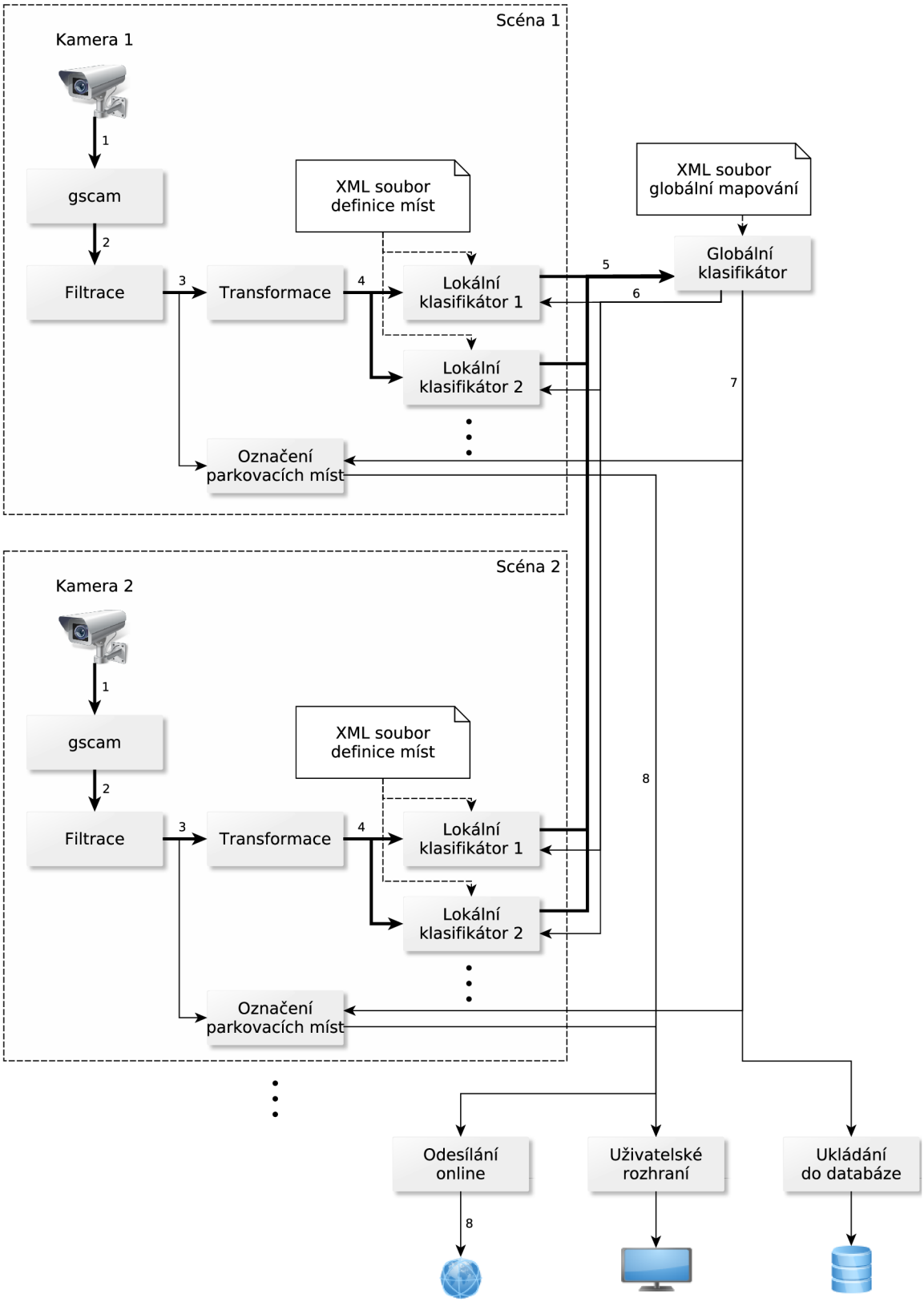
Obsah CD

Přiložené CD má následující strukturu:

/text/DP-xstran16.pdf	Text diplomové práce v pdf
/text/src/	Zdrojové soubory písemné práce v LaTeX
/readme.txt	Návod instalace a spuštění aplikace
/program/intellpark_src/	Složka se zdrojovými kódy aplikace
/program/testovaci_data/	Složka s ukázkovými daty pro test
/result/oznacene-dansko.mp4	Ukázka zruchleného videa po detekci
/excel/Parkoviste-clanek.pdf	Článek publikovaný na konferenci *Excel@FIT 2017* v českém jazyce
/excel/Parkoviste-poster.pdf	Poster prezentovaný na konferenci *Excel@FIT 2017*
/sccg/Parking-article.pdf	Článek pro konferenci *SCCG 2017* v anglickém jazyce

Příloha B

Podrobný model systému



Obrázek B.1: Podrobný model systému. Tétama (1: rtsp, 2: nezpracované snímky, 3: vybrané nejlepší snímky v určité frekvenci, 4: snímky bez deformací, 5: měkká rozhodnutí o obsazenosti, 6: dotazy pro zkontrolování určitého místa, 7: finální rozhodnutí o obsazenosti, 8: snímek s označením, 9: http)

Příloha C

Článek pro konferenci *Excel@FIT 2017*

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

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

Vizuální systém pro detekci obsazenosti parkoviště pomocí hlubokých neuronových sítí

Václav Stránský*

Abstrakt

Jelikož neustále roste počet automobilů a snižuje se šance na zaparkování, začala ve městech vznikat inteligentní parkoviště. Tato práce se zabývá návrhem a implementací robustního systému pro analýzu obsazenosti parkoviště z kamerových záznamů. Systém analyzuje jednotlivá parkovací místa ze záznamů z více-kamerového systému s možností překryvu mezi kamerami. Aplikace je navržena a implementována v Robotickém operačním systému (ROS) a její jádro se skládá ze dvou oddělených klasifikátorů. Úspěšnější, avšak pomalejší, je klasifikace pomocí hluboké neuronové sítě. Rychlou interakci řeší méně přesný klasifikátor pohybu s modelem pozadí. Systém je schopen fungovat v reálném čase, a to na grafické kartě i na procesoru. Úspěšnost systému na testovací datové sadě z reálného provozu jednoho parkoviště přesahuje 93%. Ke konceptu chytrých měst neodmyslitelně patří efektivní parkovací řešení založené na znalosti obsazenosti jednotlivých parkovacích míst. Tato práce popisuje právě takový systém, který umožní snadnou orientaci na parkovišti, a to jak pro správu, tak pro řidiče.

Klíčová slova: detekce obsazenosti parkoviště — detekce automobilů — hluboké neuronové sítě — model pozadí

Přiložené materiály: Demonstrační video

*xstran16@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

S neustále rostoucím počtem automobilů vznikají nepříjemné potíže s parkováním. Ačkoliv se na parkovišti nacházejí stále volná parkovací místa, pro řidiče bývá komplikované tato místa na rozlehlém parkovišti nalézt. Řešením mohou být aplikace, které označují obsazenost jednotlivých parkovacích míst a prostřednictvím informativních tabulí či jiné vizualizační metody informují řidiče o pozici volných míst. Vývojem právě takové aplikace se zabývá tato práce.

Popisovaná aplikace je vyvíjena od základů v programovacích jazycích C++ a Python s použitím existujících knihoven OpenCV [1] a QT4 [2]. Hlavní důraz je kladen na samotnou klasifikaci obsazenosti parkovacích míst a fungování systému jako celku. Rozšiřujícmi funkcemi jsou záznam detekovaných změn, tvorba grafů ze získaných dat a doplňující informace pro uživatele. Požadavkem je, aby výsledná aplikace fungovala neustále a v reálném čase s použitím levného hardwarového vybavení. Klasifikace musí fungovat za každého počasí a s měnícími se světelnými podmínkami. Řešení by mělo být obecné, avšak neuronovou síť je možné dotrénovat na konkrétní parkoviště.

V současné době jsou využívané systémy tvořeny

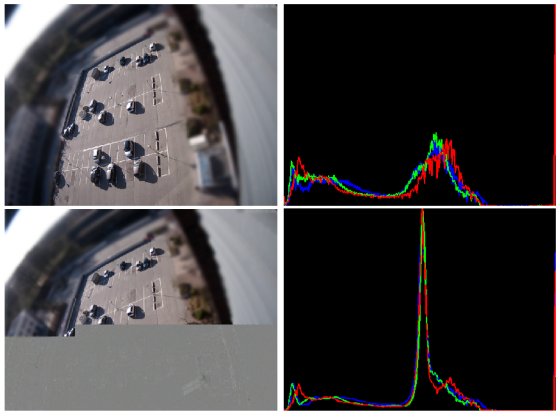
senzory, které jsou založené na mikrovlnách [3], magnetickém poli [4][5] nebo ultrazvuku [6][7]. Jelikož každé parkovací místo vyžaduje vlastní fyzický senzor, instalace systému i následná údržba je poměrně náročná. Pokud by chtěl provozovatel změnit rozložení parkovacích míst, je třeba přesunout i příslušné senzory. V posledních letech proto začaly vznikat aplikace detekující obsazenost parkovacích míst za použití počítačového vidění. Oproti senzorovým systémům je výrazně jednodušší instalace, změna rozložení míst i údržba. Kamerový systém navíc umožňuje živý náhled na scénu a s ním spojené rozšířené využití (rozpoznání typu vozidla, barvy aj.). Úspěšností se ale zatím se senzorovými systémy nemohou měřit. Monitorování kamerami přináší i řadu nevýhod, jako jsou legislativní problémy a nutnost informovat uživatele parkoviště. Obtížnější je pak detekce za nepříznivého počasí nebo v noci při nedostatečném umělém osvětlení. Pokud cizí objekt zastíní výhled kamery nebo se kamera odkloní (např. vlivem větru), může začít docházet k chybným detekcím. Srovnání existujících metod shrnuje tabulka 1.

Tabulka 1. Srovnání existujících řešení

Metoda	Úspěšnost	Náročnost instalace	Náročnost údržby
Mikrovlny	95%	vysoká	vysoká
Magnetické pole	>99%	vysoká	střední
Ultrazvuk	98-99%	vysoká	střední
Počítačové vidění	85-98%	střední	nízká

Existující metody využívající počítačové vidění používají SVM [8], segmentaci obrazu [9], histogram gradientů [10] či kaskádový detektor [11]. Porovnání úspěšnosti jmenovaných metod je komplikované, neboť každý autor přístupu využil jinou testovací sadu. Autory udávaná úspěšnost těchto metod se pohybuje mezi 85% a 95%, datové sady ale většinou neobsahují okluze či zhoršenou viditelnost. Lepší úspěšnosti dosahují metody využívající hluboké neuronové sítě [12][13], jejichž úspěšnost přesahuje 95%. Hlavním úskalím je rychlost klasifikace, kterou autoři uvádějí na 15 sekund na celé parkoviště.

Systém popisovaný v této práci je navržen a implementován v Robotickém operačním systému [14], díky němuž je rozdělen do několika oddělených uzlů. Hlavními uzly jsou zpracování snímků z kamery, filtrace snímků, transformace snímků, lokální klasifikátory, globální klasifikátor a prezentace výsledků. Snímky jsou filtrovány na základě porovnání histogramů správných a chybných snímků. Pro lepší detekci jsou ze snímku odstraněny nežádoucí transformace, distorze[1] a perspektiva[2]. Předpřipravený snímek je použit pro klasifikaci obsazenosti parkovacích míst pomocí dvou klasifikátorů. Hlavní důraz je kladen na klasifikaci pomocí strojového učení, ke kterému je využita hluboká konvoluční neuronová síť. Detekci příjezdu a odjezdu doplňuje klasifikátor založený na modelu pozadí. Výsledky obou klasifikátorů jsou sloučeny a zveřejněny. Pokud systém obsahuje více kamer, jsou snímky z každé kamery zpracovávány nezávisle, výsledky dílčích klasifikátorů se sloučí všechny najednou.



Obrázek 1. Správný a chybný snímek s odpovídajícími histogramy

Představované řešení je zajímavé svou vysokou úspěšností, jednodušší instalací a schopností klasifikovat obsazenost stovek parkovacích míst v reálném čase. Systém není bezchybný, v některých případech dochází k chybné klasifikaci. Takových případů je však méně než 7% a s postupným vývojem aplikace a zvětšováním trénovací sady jich stále ubývá. Metoda nejčastěji selhává za zhoršené viditelnosti (v noci, při orosení kamery), při změnách počasí (déšť, sníh, ostré stíny) a v případech extrémní okluze, kdy je automobil téměř nebo úplně zakryt větším objektem (nákladní vůz).

2. Návrh architektury v ROSu

Aplikace je vytvořena v prostředí Robotického operačního systému (zkráceně ROS). Zásadní výhodou ROSu v rámci této aplikace je možnost rozdělit program do samostatných uzlů (procesů), které spolu vzájemně komunikují pomocí zpráv (komunikace je založena na TCP/IP). To umožňuje restartování pouze jednoho uzlu v případě jeho selhání, jednoduchou

[1] Při distorzi (zkreslení obrazu) není příčné zvětšení po celém poli obrazu stejné, čímž je porušena geometrická podobnost předmětu a jeho obrazu.

[2] Při perspektivním zobrazení se zobrazované předměty zdánlivě zmenšují a sbíhají.

Obrázek 2. Původní snímek, snímek po odstranění distorze a snímek po odstranění perspektivy

tvorbu různých variant aplikace na základě požadavků, transparentnost, ladění pomocí odchytávání zpráv a další. Na schématu 3 lze vidět stručný návrh struktury celé aplikace. Ohraničenou skupinu uzlů lze přidat vícekrát, podle počtu kamer. Výsledky z těchto skupin jsou zpracovány dohromady a prezentovány.

2.1 Uzly systému

Zpracování snímků z kamery Snímky z kamery zpracovává již vytvořený uzel v ROSu `Gscam`. Přijímá video z IP kamery vysílané pomocí RTSP, kontroluje funkčnost kamery a při jejím výpadku obnoví odesílání snímků. Zpracované snímky vysílá pomocí již vytvořené zprávy, včetně přídavných informací o obrázku a stavu kamery.

Filtrace snímků Další uzel porovnává histogram příchozího snímku s průměrným histogramem předchozích snímků (obrázek 1). Pokud se histogramy liší, snímek není odeslán a vybírá se jiný snímek. Pokud jsou histogramy podobné, snímek se odešle a průměrný histogram je aktualizován. Takto jsou odstraněny vadné snímky z kamery a přitom zachovány snímky s postupnou změnou (např. změna osvětlení). Uzel odesílá snímky s nastavenou frekvencí, přebytečné snímky zahazuje.

Transformace snímků Pro přesnější klasifikaci jsou ze snímků odstraněny nežádoucí deformace. Další uzel odstraňuje distorzi a perspektivní zobrazení, aby byla všechna parkovací místa v obraze podobně velká. Ukázky snímků bez transformací a s odstraněnými deformacemi lze vidět na obrázku 2.

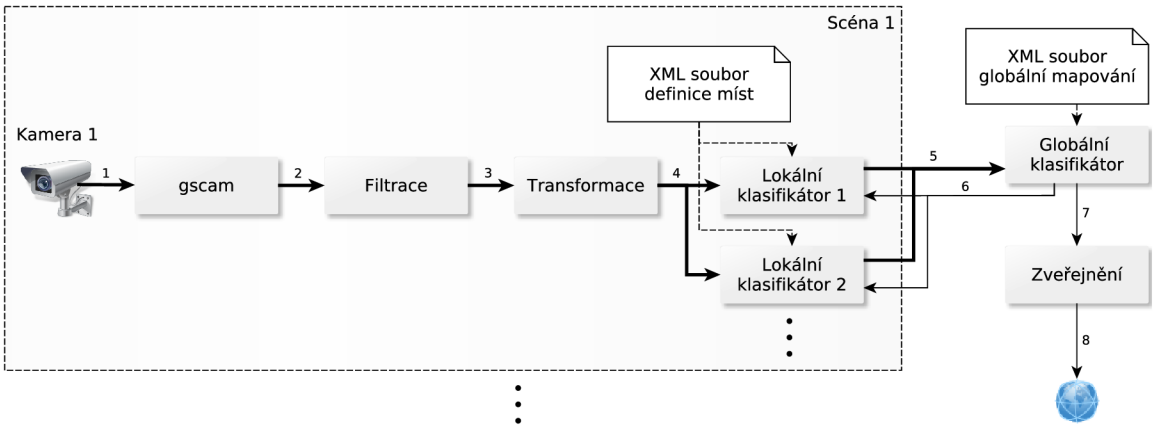
Lokální klasifikátory Lokální klasifikátor je jednou z nejvýznamnějších částí celé aplikace. Návrh aplikace umožňuje větší množství jednodušších lokálních klasifikátorů, jejichž dílčí výsledky jsou dále zpracovávány a slučovány. Uzel při inicializaci načte konfigurační soubor, který popisuje umístění parkovacích míst, jejich tvar a rozdělení do skupin. Výstupem každého lokálního klasifikátoru je sada měkkých rozhodnutí o obsazenosti příslušných parkovacích míst. Jako lokální klasifikátor je možné použít prakticky jakoukoliv funkci rozhodující o obsazenosti parkovacích míst, jejíž úspěšnost je lepší než 50%. Příklady úspěšných lineárních klasifikátorů jsou hranový klasifikátor, klasifikátor založený na modelu pozadí a detekci pohybujících se vozidel či klasifikátor založený na porovnávání histogramů. Aktuální verze popisované aplikace obsahuje dva klasifikátory, které jsou popsány v následující kapitole.

Globální klasifikátor Výsledky lokálních klasifikátorů jsou slučovány v globálním klasifikátoru, který je společný pro celou aplikaci. Lze zvolit vážený průměr (výchozí), neuronovou síť nebo SVM. Globální klasifikátor má dvě hlavní funkce. Za prvé slučuje měkká rozhodnutí lokálních klasifikátorů v rámci jedné skupiny, tedy určuje finální obsazenost parkovacích míst z jedné konkrétní kamery. Druhou funkcí je pak slučování rozhodnutí o obsazenosti jednoho parkovacího místa snímaného více kamerami. Přiřazení dvou a více označení ke konrétnímu parkovacímu místu je realizováno prostřednictvím konfiguračního souboru, který mapuje označení v rámci každé scény na globální označení parkovacího místa.

Prezentace výsledků Získané výsledky jsou několika způsoby zveřejňovány. K okamžité kontrole správnosti systému mohou být odesílány přes protokol HTTP a zobrazovány na webové stránce. Pro pozdější analýzu jsou ukládány do databáze jako změna obsazenosti konkrétního místa v aktuálním čase.

3. Klasifikace obsazenosti místa

Pro klasifikaci jsou použity a zde popsány dva odlišné klasifikátory, klasifikace pomocí hluboké neuronové sítě a pomocí modelu pozadí.



Obrázek 3. Stručné schéma návrhu systému znázorňuje nejvýznamnější ROS uzly, které jsou propojeny pomocí témat (1: rtsp, 2: raw snímky, 3: vybrané nejlepší snímky v určité frekvenci, 4: snímky bez deformací, 5: měkká rozhodnutí o obsazenosti, 6: dotazy pro zkontrolování určitého místa, 7: finální rozhodnutí o obsazenosti, 8: http)

3.1 Klasifikace pomocí hluboké neuronové sítě

Pro práci s neuronovou sítí byl vybrán framework Caffe [15], který umožňuje provádět výpočty na procesoru nebo na grafické kartě. Pro návrh celé struktury neuronové sítě využívá Caffe serializačního textového formátu Protobuffer od Google. Caffe má k dispozici více variant rozhraní (příkazová řádka, Python, MATLAB, C++), což umožnilo snadné napojení na vyvíjený systém. Vzhledem k úspěšnosti existujících a volně dostupných sítí by bylo kontraproduktivní navrhovat vlastní síť, na základě testování (5) tedy byla vybrána síť *GoogleNet* [16], kterou již nebylo třeba dále upravovat. V popisovaném systému jsou použity dvě výstupní třídy, volné nebo obsazené místo. Výstupem sítě je pravděpodobnost, s jakou je na testovacím vzorku obsazené místo. Síť byla trénována na třicet epoch, použita byla nejlepší z nich, patnáctá epocha. Ověřovací přesnost (validation accuracy) dosahovala od této epochy hodnoty 99,7642%.

Návrh systému umožňuje klasifikaci všech parkovacích míst (při použití GPU) nebo klasifikaci po jednom místě (při použití CPU). Klasifikátor obsahuje plánovací frontu se třemi prioritami. Nejnižší prioritu získávají parkovací místa, na nichž nebyla v nejbližší době detekována žádná změna, ale je vhodné ohodnocení zkontrolovat. Nejvyšší prioritu získají místa, na nichž model pohybu detekoval změnu a je třeba neprodleně zjistit obsazenost. Pokud se obsazenost změní, je za uživatelem zvolený čas naplánovaná klasifikace se střední prioritou, aby bylo rozhodnutí zkontrolováno.

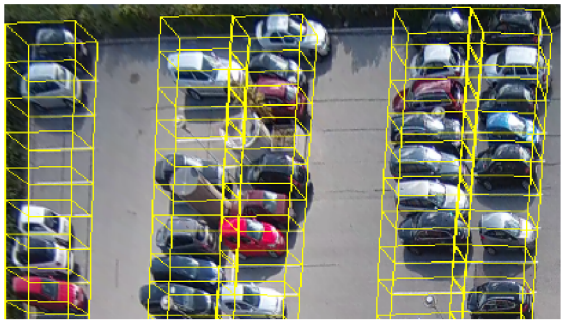
3.2 Klasifikace pomocí modelu pozadí

Druhý klasifikátor využívá ke klasifikaci model pozadí. Porovnáním aktuálního snímku s modelem pozadí je možné detekovat pohybující se automobily, které přijíždějí či odjíždějí na sledované parkovací místo. Metoda vyžaduje počáteční inicializaci za předpokladu, že jsou některá parkovací místa obsazená. V takovém případě není možné vytvořit model pozadí na začátku, princip je třeba obrátit. Nejprve se vytvoří model pro přítomný automobil, při první významné změně se místo klasifikuje jako volné a poté je možné vytvořit model pozadí. Modelu pozadí je v takovém případě třeba předat informaci, které části snímku se mají aktualizovat (pozadí) a které mají zůstat neaktualizovány (obsazená místa).

Metoda má hlavní rizika v aktualizaci modelu pozadí. Na rozdíl od sledování jedoucích vozidel na vozovce není možné aktualizovat model pozadí po celou dobu. Po dobu obsazenosti parkovacího místa se pozadí může výrazně změnit (změna světelných podmínek, sníh na vozovce, stín atp.). Po opuštění místa automobilem se model pozadí od aktuálního snímku liší natolik, že je místo stále považováno za obsazené.

3.3 Sloučení výsledků klasifikátorů

Aplikace je navržena tak, aby byla schopná bez obtíží fungovat i na průměrném hardwaru bez grafické karty. Na průměrném procesoru (viz kapitola 5) trvá klasifikace jednoho parkovacího místa neuronovou sítí jednu až dvě vteřiny, klasifikace celého parkoviště s dvěmi sty místy tedy trvá řádově minuty. Naproti tomu zpracování snímku druhým popisovaným klasifikátorem splňuje podmínku vyhodnocení snímku do jedné vteřiny. Reálně systém funguje tak, že jsou



Obrázek 4. Označení parkovacích míst

Obrázek 5. Ukázky výřezů míst (vlevo a uprostřed obsazená, vpravo volné)

výsledky z modelu pozadí odesílány ke globálnímu rozhodnutí a v případě detekce změny se globální klasifikátor dotáže neuronové sítě na správnost. Pokud model pozadí nedetekuje žádnou změnu, pro kontrolu klasifikuje neuronová síť postupně všechna místa. Konečné rozhodnutí je odesláno zpět do klasifikátorů a model pozadí se aktualizuje.

4. Označení parkovacích míst

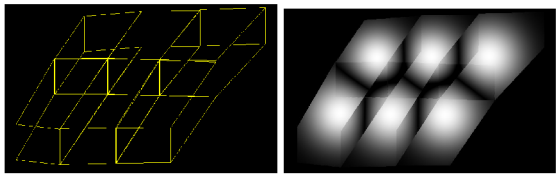
Jelikož je systém vyvíjen pro statické kamery a parkoviště, na nichž se poloha parkovacích míst neliší, je možné označit parkovací místa před začátkem klasifikace. Tím se z detekce automobilů v obraze stává jednodušší problém klasifikace obsazenosti konkrétního místa.

Protože kamera není umístěna kolmo nad parkovacím místem, používá se označení místa ve 3D. Nejprve uživatel ohraničí celé parkoviště a zadá rozměry v reálném světě, z čehož lze spočítat umístění kamery. Poté ohraničí každé parkovací místo čtyřmi body a výškou a systém spočítá 3D model parkovacího místa na snímku. Ukázku označení několika míst lze vidět na obrázku 4.

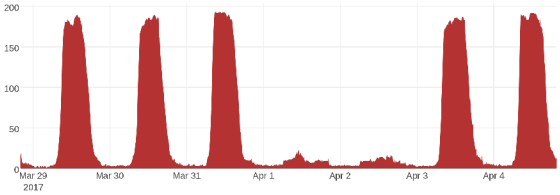
V hluboké neuronové síti se označení používá pro vyříznutí trénovacích vzorků (obrázek 5) a následně pro samotnou klasifikaci, při které jsou vstupy právě tyto výřezy. Pro klasifikaci modelem pozadí je navíc na vyříznuté parkovací místo aplikována maska, která odstíní oblasti překryvů sousedních parkovacích míst (obrázek 6).

5. Testování

Systém byl testován na datových sadách o celkové délce 32 hodin z reálného parkoviště, které bylo



Obrázek 6. Kompozice šesti masek parkovacích míst

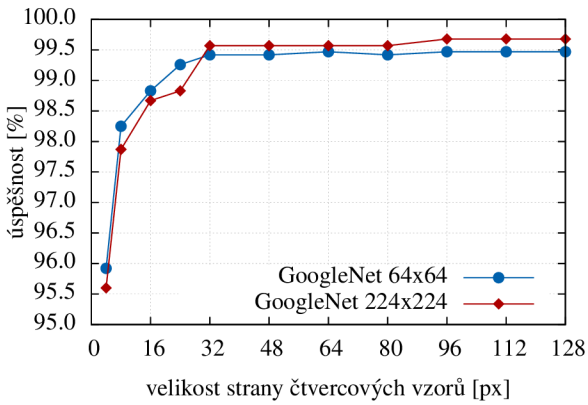


Obrázek 7. Graf historie obsazenosti testovacího parkoviště (od středy do úterý)

snímáno dvěma kamerami a obsahovalo 193 parkovacích míst. Pro jednoduchou interpretaci získaných dat byly implementovány i základní vizualizační prvky, jako je například graf historie obsazenosti celého parkoviště (viz obrázek 7).

Úspěšnost celého systému závisí především na úspěšnosti hluboké neuronové sítě, proto se další testy zabývaly právě neuronovými sítěmi. První test zkoumal vliv odstranění deformací z obrazu na úspěšnost klasifikace. Pro vzory bez odstraněných deformací byla získána úspěšnost klasifikace na dvou datových sadách 95,23% a 94,18%, pro vzory s odstraněnými deformacemi úspěšnost 99,20% a 99,01%. Další testy se proto věnovaly výhradně vzorům s odstraněnými deformacemi.

Byly vybrány dvě předtrénované architektury, jejichž úspěšnost byla nejlepší, a dále byly upraveny pro různou velikost vstupů. Těmito architekturami byla konvoluční neuronová síť *GoogleNet* s velikostmi vstupů 32x32, 64x64 a 224x224 pixelů a reziduální neuronová síť *Resnet* s velikostí vstupů 256x256 pixelů. (Vyřazeny byly např. sítě *GoogneNet256* či *VGG16-32*.) *GoogleNet* obsahuje 22 vrstev s parametry (27 včetně slučovacích vrstev), *ResNet* se skládá z 50 vrstev. Druhý test zjišťoval úspěšnost vybraných sítí natrénovaných a testovaných na vzorcích s nejlepším rozlišením, kterého je v systému možné dosáhnout. Byla vytvořena datová sada pro dotrénování, obsahující 7376 trénovacích vzorů (3791 obsazených míst a 3585 volných míst), a dvě odlišné datové sady pro testování, první s 1881 vzory (1181 obsazených míst a 700 volných míst) a druhá s 1802 vzory (1020 obsazených míst a 782 volných míst). Velikosti vzorků se pohybovaly v rozmezí od 120x80 do 130x130. Data byla pořízena ze dvou kamer (5mpx, rozlišení 2592x1920) testovacího parkoviště v různé časové doby, za měnících se světelných podmínek



Obrázek 8. Závislost úspěšnosti na velikosti vzorů

i počasí ve 25 různých dnech. Vždy byla vybrána trénovací epocha s nejlepším výsledkem na testovací sadě. Tabulka 2 shrnuje úspěšnost klasifikace a rychlost klasifikace jednoho parkovacího místa na CPU (*Intel Core i7-4702MQ CPU @ 2.20GHz*) a na GPU (*GeForce GTX 1080*). Na základě výsledků byla vybrána a použita v aplikaci nejúspěšnější síť *GoogleNet 224x224*, v případě potřeby vyšší rychlosti by bylo vhodnější použít *GoogleNet* s menší velikostí vzorů.

Tabulka 2. Srovnání hlubokých neuronových sítí

název sítě	velikost vzoru	rychlost CPU	rychlost GPU	úspěšnost sada 1	úspěšnost sada 2
GoogleNet	32x32	1,1s	3,7ms	98,83%	99,01%
GoogleNet	64x64	1,2s	5,3ms	99,10%	98,90%
GoogleNet	224x224	1,4s	11,7ms	99,64%	98,96%
Resnet	256x256	1,7s	13,3ms	97,18%	92,75%

Třetí test zjišťoval vliv zmenšení a opětovného zvětšení velikosti trénovacích i testovacích vzorů na úspěšnost dvou nejlepších sítí z předchozího testu. Snahou bylo zjistit minimální velikost vzorů, pro kterou má systém nejvyšší úspěšnost. Na základě získaných výsledků je možné co nejvíce omezit datový tok od kamer k centrální jednotce, v níž mohou být vzory opět zvětšeny na potřebnou velikost vstupu sítě. Z grafu 8 si lze všimnout, že je možné zmenšit vzory až na velikost 32x32 pixelů a stále bude mít síť *GoogleNet 224x224* úspěšnost vyšší než 99,5%. Při dalším zmenšení je již úspěšnější síť *GoogleNet 64x64*. Zajímavostí je úspěšnost téměř 96% pro vzory o velikosti 4x4 pixely. Tak vysoké úspěšnosti je možné dosáhnout především díky tomu, že je trénovací i testovací sada pořízena pouze z jednoho parkoviště a je tedy pravděpodobné, že síť není dostatečně obecná.

6. Závěr

Práce popsala systém pro klasifikaci obsazenosti parkovacích míst a představila dvě základní metody, které jsou použity: hlubokou neuronovou síť a model pozadí. Výsledná aplikace dosahuje na testovací sadě velmi dobré úspěšnosti 99.6% a je testována v reálném provozu. Integrace řešení detekce obsazenosti parkovacích míst do vyššího celku usnadní řidičům vyhledání volného parkovacího místa, omezí nežádoucí provoz po parkovišti a v neposlední řadě sníží produkci CO_2. Zřizovatelům parkoviště pak může nabídnout řadu informací o době obsazenosti, vytíženosti konkrétních míst a další.

V pokračujícím vývoji bude snaha o zvýšení úspěšnosti zvětšováním trénovací sady a zobecnění natrénovaných modelů přidáním dat z dalších parkovišť. Na základě požadavků cílového zákazníka bude přidána další funkčnost aplikace, např. předpokládaný čas uvolnění parkovacího místa, pokročilé statistiky o využití parkoviště a další. V současné době je systém upravován pro fungování na vestavěném systému s omezenými výpočetními zdroji.

Poděkování

Tato práce byla vytvořena jako část magisterské diplomové práce na základě firemního zadání pod vedením Ing. Jaroslava Rozmana, PhD.[3] ze strany školy a Ing. Davida Hermana[4] ze strany firmy.

Literatura

[1] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2nd edition, 2013.

[2] Jasmin Blanchette and Mark Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[3] O. Dokur, S. Katkoori, and N. Elmehraz. Embedded system design of a real-time parking guidance system. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–8, April 2016.

[4] Z. Zhang, M. Tao, and H. Yuan. A parking occupancy detection algorithm based on amr sensor. *IEEE Sensors Journal*, 15(2):1261–1269, Feb 2015.

[5] P. Šolić, I. Marasović, M. L. Stefanizzi, L. Patrono, and L. Mainetti. Rfid-based efficient

[3]Fakulta informačních technologií, Vysoké učení technické v Brně

[4]RCE Systems, s.r.o., Brno, Česká republika

method for parking slot car detection. In *2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 108–112, Sept 2015.

[6] H. Zhou and Z. Li. An intelligent parking management system based on RS485 and RFID. In *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 355–359, Oct 2016.

[7] E. Sifuentes, O. Casas, and R. Pallas-Areny. Wireless magnetic sensor node for vehicle detection with optical wake-up. *IEEE Sensors Journal*, 11(8):1669–1676, Aug 2011.

[8] A. S. G. Prasad, U. Sharath, B. Amith, B. R. Supritha, S. Asokan, and G. M. Hegde. Fiber bragg grating sensor instrumentation for parking space occupancy management. In *2012 International Conference on Optical Engineering (ICOE)*, pages 1–4, July 2012.

[9] R Yusnita, Fariza Norbaya, and Norazwinawati Basharuddin. Intelligent parking space detection system based onimage processing. *International Journal of Innovation, Management and Technology*, 3(3):232, 2012.

[10] Radovan Fusek, Eduard Sojka, Karel Mozdřeň, and Milan Šurkala. Energy based descriptors and their application for car detection. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 1, pages 492–499. IEEE, 2014.

[11] Gabriele Maria, Enrico Baccaglini, D Brevi, M Gavelli, and Riccardo Scopigno. A drone-based image processing system for car detection in a smart transport infrastructure. In *Electrotechnical Conference (MELECON), 2016 18th Mediterranean*, pages 1–5. IEEE, 2016.

[12] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo. Car parking occupancy detection using smart camera networks and deep learning. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1212–1217, June 2016.

[13] J. Jermsurawong, M. U. Ahsan, A. Haidar, H. Dong, and N. Mavridis. Car parking vacancy detection and its application in 24-hour statistical analysis. In *2012 10th International Conference on Frontiers of Information Technology*, pages 84–90, Dec 2012.

[14] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Příloha D

Poster pro konferenci *Excel@FIT 2017*

Excel@FIT 2017

40

VIZUÁLNÍ SYSTÉM PRO DETEKCI OBSAZENOSTI PARKOVIŠTĚ POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ

Václav Stránský

- Obsazenost parkoviště v reálném čase
- Měření doby stání a historie obsazenosti
- Implementace v Robotickém operačním systému (ROS)
- Kvalitní pomalejší klasifikace hlubokou neuronovou sítí
- Rychlá interakce pomocí klasifikátoru s modelem pozadí
- Úspěšnost přes 93%
- Testováno na záznamech z reálného parkoviště

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ParkingDetection
Technology for smart cities

RCE SYSTEMS

Příloha E

Článek pro konferenci *SCCG 2017*

A Visual System for Detecting the Occupancy of a Car Park by Deep Neural Networks

Vaclav Stransky[a,b,1], Jaroslav Rozman[a,1], David Herman[b,1], David Hlavon[b,1], Adam Babinec[a,b,1]

[a] *Faculty of Information Technology, Brno University of Technology, Czech Republic*
[b] *RCE Systems, s.r.o., Brno, Czech Republic*

Abstract

The concept of smart cities is inherently connected with efficient parking solutions based on the knowledge of individual parking space occupancy. The subject of this paper is the design and implementation of a robust system for analysing parking space occupancy from a multi-camera system with the possibility of visual overlap between cameras. The system is designed and implemented in Robotic operating system (ROS) and its core consists of two separate classifiers. The more successful, however, a slower option is detection by a deep neural network. A quick integration is provided by a less accurate classifier of movement with a background model. The system is capable of working in real time on a graphic card as well as on a processor. The success rate of the system on a testing data set from real operation exceeds 93%. Smart parking concepts inherently include efficient parking solutions based on the knowledge of the occupancy of individual parking spaces. This paper describes just such a system that allows easy orientation in the parking lot, both for management and for the driver.

Keywords: occupancy of a car park, vehicle detection, deep neural networks, background model

1. Introduction

With a continuously growing number of cars, there are inconvenient difficulties with parking them. Although there are still vacant parking spaces in a car park, it is often complicated for the drivers to find these spaces in a large car park. A solution can be seen in applications that mark the occupancy of individual parking spaces and inform the drivers about their location on information boards or through another visualization method. The development of such an application is the subject of this paper.

The described application is developed from square one in C++ and Python programming languages using the existing libraries OpenCV [1] and QT4 [2]. The main focus is on the classification of parking space occupancy itself and on the functioning of the system as a whole. Additional functions include recording of detected changes, creating graphs from obtained data and additional information for the users. The requirement is for the final application to work non-stop and in real time using inexpensive hardware equipment. The classification must work in all weathers and in changing light conditions. The solution should be general but the neural network can be trained for a specific car park.

The used systems currently consist of sensors based on microwaves [3], on magnetic field [4][5] or on ultrasound [6][7]. Since each parking space requires its own physical sensor, the installation of the system as well as the subsequent maintenance is rather demanding. If the operator wished to change the layout of parking spaces, it would also be necessary to move the respective sensors. Therefore, applications detecting the occupancy of parking spaces using computer vision have been coming into existence lately. Compared to sensor systems, the installation is much easier, especially the layout of the spots as well as maintenance. Moreover, a camera system enables a live overview of the scene and associated extended usage (identifying the type of vehicle, colour etc.) However, their success rate is incomparable to sensor systems. Monitoring with cameras also brings a number of disadvantages, such as legislation problems and the necessity to inform the car park users. Detection gets more difficult in bad weather or at night with insufficient artificial lighting. If a foreign object gets into the view of the camera or if the camera tilts (e.g. due to wind), false detections can occur. A comparison of the existing methods is summarized in table 1.

Table 1: Comparison of existing solutions

Method	Success rate	Demands of installation	Demands of maintenance
Microwaves	95%	high	high
Magnetic field	>99%	high	medium
Ultrasound	98-99%	high	medium
Computer vision	85-98%	medium	low

The existing methods using computer vision use SVM [8], image segmentation [9], gradient histogram [10] or cascade de-

Email addresses: xstran16@stud.fit.vutbr.cz, vaclav.stransky@rcesystems.cz (Vaclav Stransky), rozmanj@fit.vutbr.cz (Jaroslav Rozman), david.herman@rcesystems.cz (David Herman), david.hlavon@rcesystems.cz (David Hlavon), ibabinec@fit.vutbr.cz, adam.babinec@rcesystems.cz (Adam Babinec)

tector [11]. Comparing the success rate of the listed methods is complicated since each author used a different testing set. The success rate of these methods given by the authors ranges from 85% to 95%, however, data sets most often do not include occlusions or low vision. Higher success rate is achieved by methods using deep neural networks [12][13], where the success rate exceeds 95%. The main shortcoming is the speed of classification, which the authors give as 15 seconds for the entire car park.

The system described in this paper is designed and implemented in a Robot operating system [14], thanks to which it is divided into several separate nodes. The main nodes are: processing images from camera, filtering the images, transforming the images, local classifiers, a global classifier and presentation of results. Images are filtered based on a comparison of histograms of correct and faulty images. To achieve a better detection, undesirable transformations, distortions[1] and perspective[2] are removed from the image. An image prepared in this way is used for parking space occupancy classification using two classifiers. The main focus is on classification using machine learning, which uses deep convolutional neural network. Detection of arrival and departure is supplemented by a classifier based on a background model. The results of both classifiers are merged and published. If the system consists of more cameras, the images from each camera are processed independently and the results of partial classifiers are merged all at once.

The introduced solution is interesting thanks to its high success rate, easier installation and the ability to classify the occupancy of hundreds of parking spaces in real time. The system is not flawless, in some cases, there is faulty classification. However, this occurs in less than 7% of cases and with the gradual development of the application and the increase of the training set, there is less and less of them. The method most often fails in low visibility (at night, when the camera is wet with dew), due to changes of weather (rainfall, snow, sharp shadows) and in cases of extreme occlusion, when a car is partially or completely blocked by a larger object (lorry).

2. Design of architecture in ROS

The application is created in the environment of Robot operating system (ROS). The key advantage of ROS within this application is the possibility to divide the program into individual nodes (processes) that communicate with each other by messages (communication is based on TCP/IP). This allows reset of only one node in case of its failure, easy creation of various versions of the application based on requirements, transparency, fine-tuning by means of intercepting messages and suchlike. See the scheme 2 for a brief overview of the structure of the entire application. A defined group of nodes can be added more times, based on the number of cameras. The results of these groups are processed together and presented.

[1] When there is a distortion of the image, the transverse magnification is not the same across the entire image field, which disrupts the geometrical similarity of the object and its image.

[2] In a perspective image, the displayed objects seem to grow smaller and come together.

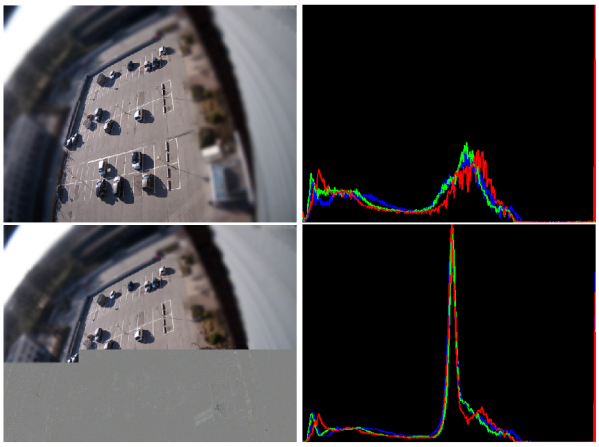


Figure 1: A correct and a faulty image with respective histograms

2.1. *Nodes of the system*

Processing images from the camera Images from camera are process by `Gscam` node already created in ROS. It receives video from IP camera transmitted by RTSP, checks the functionality of the camera and in case of its failure, renews the transmission of images. It transmits the processed images by means of an already created message, including additional information about the image and the status of the camera.

Filtering of images Another node compares the histogram of an incoming image with an average histogram of previous images (figure 1). If the histograms differ, the image is not sent and another image is selected. If the histograms are similar, the image is sent and the average histogram is updated. This way helps to remove faulty images from the camera and at the same time, to keep the images with a gradual change (e.g. the change of lighting). The node sends the images with a set frequency and discards redundant images.

Transformation of images For a more accurate classification, undesirable deformations are removed from the images. The next node removes distortion and perspective image so that all parking spaces in the image are of similar size. Samples of images without transformations and with removed deformations can be seen in figure 3.

Local classifiers A local classifier is one of the most important parts of the entire application. The design of the application enables a larger number of simpler local classifiers whose partial results are further processed and merged. During initialization, the node loads a configuration file that describes the layout of parking spaces, their shape and division into groups. The output of each local classifier is a set of soft decisions about the occupancy of the respective parking spaces. As a local classifier, practically any function deciding about the occupancy of parking spaces with a success rate of more than 50% can be used. Examples of successful linear classifiers include edge classifier, classifier based on a background model

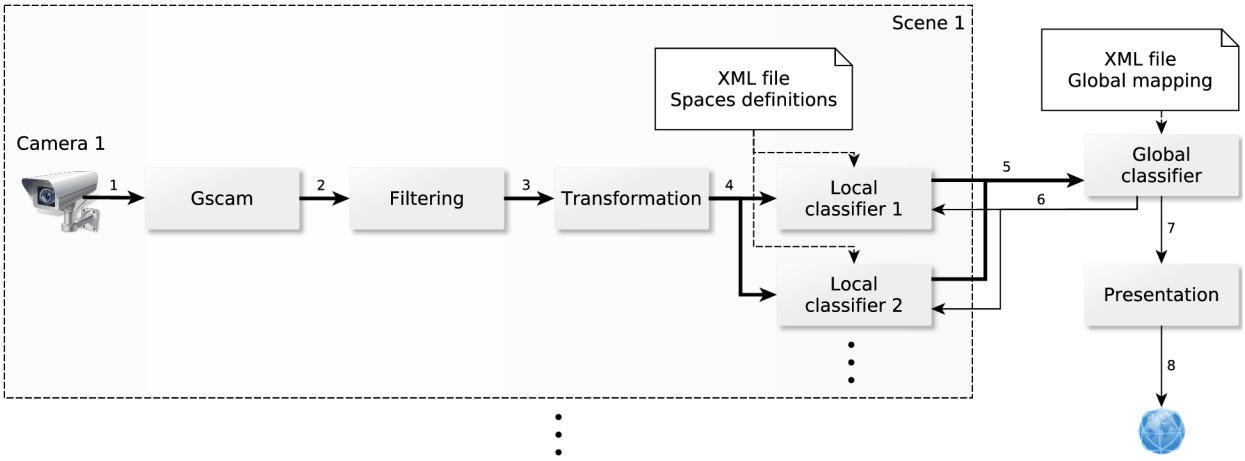


Figure 2: A brief scheme of the system design depicts the most important ROS nodes (1: rtsp, 2: raw images, 3: selected best images at a certain frequency, 4: images with removed deformations, 5: soft decisions about occupancy, 6: queries to check a specific location, 7: final decision about occupancy, 8: http)

Figure 3: Original image, image after removal of distortion and image after removal of perspective

and detection of moving vehicles or a classifier based on comparison of histograms. The current version of the described application contains two classifiers that are described in the following section.

Global classifier The results from local classifiers are merged in a global classifier that is common for the entire application. The selections include weighted average (original), neural network or SVM. The global classifier has two main functions. Firstly, it merges the soft decisions from local classifiers within one group, which means that it determines the final occupancy of parking spaces from one specific camera. Another function is the merging of decisions about occupancy of one parking space scanned by two cameras. The assignment of two or more markings to a specific parking space is done by a configuration file that maps the markings within each scene onto the global marking of a parking space.

Presentation of results The obtained results are published in several ways. For an immediate check of the correctness of the system, they can be sent via HTTP protocol and displayed on a web page. For a later analysis, they are saved into the database as a change in the occupancy of a specific space in current time.

3. Classification of parking space occupancy

For classification, two different classifiers are used and described here: classification using a deep neural network and classification using a background model.

3.1. Classification using a deep neural network

For work with neural network, we selected framework Caffe [15] that enables to carry out calculations on a processor or on a graphic card. For the design of the entire structure of neural network, Caffe uses Protobuffer serialization text format from Google. In Caffe, there are several interface varieties (command line, Python, MATLAB, C++), which allowed a simple connection to the developed system. Due to the success rate of existing and freely available networks, it would be counter-productive to design our own network; therefore, based on testing (section 5), the *GoogleNet* network was selected [16], which needs no further adjustments. In the described system, two output classes are used: a vacant or occupied space. The output of the network is the probability that the testing sample contains an occupied space. The network was trained for thirty epochs, out of which the best one was used: the fifteenth epoch. Validation accuracy starting from this epoch reached the value of 99.76%.

The design of the system enables classification of all parking spaces (using GPU) or classification by one space (using CPU). The classifier contains a planning queue with three priorities. The lowest priority is assigned to parking spaces where no change has been detected recently but it is suitable to check the evaluation. The highest priority is assigned to spaces where the movement model detected a change and it is necessary to find out the occupancy immediately. If the occupancy changes, a classification with medium priority is planned after a time defined by the user so that this decision can be checked.

3.2. Classification using a background model

The second classifier uses a background model for classification. When comparing a current image with the background model, it is possible to detect moving vehicles that are arriving at or leaving the monitored parking space. This method requires initialization assuming that some parking spaces are occupied. In this case, it is impossible to create the background model at the beginning and the principle must be reversed. Firstly, a model for the situation when a car is present is created; after the first major change, the space is classified as vacant and then it is possible to create a background model. In this case, the background model needs to be provided with information which parts of the image are to be updated (background) and which are to remain without updates (occupied spaces).

The main risks of this method lie in updating the background model. As opposed to monitoring travelling vehicles on a road, it is impossible to keep updating the background model for the entire time. During the period when the parking space is occupied, the background may change significantly (change of light conditions, snow on the road, shadow etc.). After the vehicle has left the space, the background model differs from the current image to such an extent that the space is still considered as occupied.

3.3. Merging the results of the classifiers

The application is designed so that it can run without problems on average hardware without a graphic card. On an average processor (see section 5), the classification of one parking space by a neural network takes one to two seconds, the classification of the entire car park with two hundred spaces thus takes several minutes. As opposed to that, processing the image with the second described classifier meets the condition of evaluating the image within one second. In reality, the system works in this way: the results from the background model are sent for a global decision, and in case a change has been detected, the global classifier asks the neural network about correctness. If the model does not detect any change, the neural network classifies all spaces one by one to check. The final decision is sent back to the classifiers and the background model is updated.

4. Marking of parking spaces

Since the system is developed for static cameras and car parks where the position of parking spaces does not change, it is possible to mark parking spaces before the start of the classification. Thus, detecting vehicles in an image becomes a simpler problem of classifying the occupancy of a specific space.

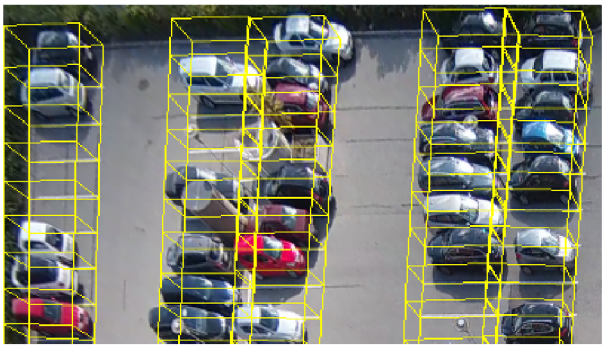


Figure 4: Marking of parking spaces

Figure 5: Examples of cut-out spaces (occupied spaces on the left and in the middle, vacant space on the right)

Since the camera is not placed vertically above the parking space, space marking in 3D is used. Firstly, the user marks the entire car park and enters the real dimensions, from which the placement of the camera can be calculated. Then, the user marks each parking space by four points and by height and the system calculates a 3D model of the parking space in the image. For an example of marking of several spaces see figure 4.

In deep neural network, the marking is used to cut out training samples (figure 5) and subsequently for the classification itself where the inputs are these segments. Moreover, for classification by a background model, a mask is applied on the cut-out parking space that eliminates the overlapping areas of adjacent parking spaces (figure 6).

5. Testing

The system was tested on data sets of total length of 32 hours from a real car park that was monitored by two cameras and consisted of 193 parking spaces. For an easy interpretation of the obtained data, also basic visualization elements were implemented, such as a graph of the occupancy history of the entire car park (see figure 7).

The success rate of the entire system depends predominantly on the success rate of the deep neural network, therefore, further tests dealt with neural networks. The first test examined the influence of removing deformation from the image on the classification success rate. For samples without removed deformations we obtained a classification success rate of 95.23% and 94.18% on two data sets, for samples with removed deformations, the success rate was 99.20% and 99.01%. For this

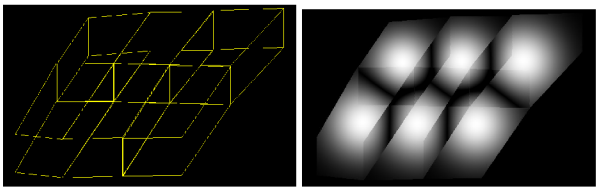


Figure 6: Composition of six masks of parking spaces

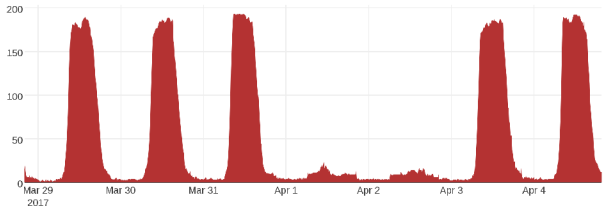


Figure 7: Graph of the occupancy history of the testing car park (from Wednesday to Tuesday)

reason, further tests dealt only with samples with removed deformations.

Two pre-trained architectures were selected whose success rate was the highest, and they were further adjusted for various input sizes. These architectures were a convolutional neural network *GoogleNet* with input sizes of 32x32, 64x64 and 224x224 pixels, and a residual neural network *Resnet* with input sizes of 256x256 pixels. (Networks such as *GoogneNet256* or *VGG16-32* were eliminated.) *GoogleNet* contains 22 layers with parameters (27 including merging layers), *ResNet* consists of 50 layers. The second test examined the success rate of the selected networks pre-trained and tested on samples with the best resolution that can be achieved in the system. We created a data set for final training that contained 7376 training samples (3791 occupied spaces and 3585 vacant spaces) and two different data sets for testing; the first one with 1881 samples (1181 occupied spaces and 700 vacant spaces) and the other one with 1802 samples (1020 occupied spaces and 782 vacant spaces). The sizes of samples ranged from 120x80 to 130x130. Data were obtained from two cameras (5mpx with 2592x1920 resolution) from the testing car park at various points in time, in changing light conditions and weather conditions on 25 different days. The training epoch that was selected was always the one with the best result on the testing set. Chart 2 summarizes the classification success rate and the speed of classification of one parking space on CPU (*Intel Core i7-4702MQ CPU @ 2.20GHz*) and on GPU (*GeForce GTX 1080*). Based on the results, the most successful network was selected and used in the application, namely *GoogleNet 224x224*; if a higher speed is needed, it would be more suitable to use *GoogleNet* with a smaller sample size.

The third test examined the influence of reducing and repeated enlargement of the size of both training and testing samples on the success rate of the two best networks from the previous test. The goal was to find out the minimum size of samples for which the system has the highest success rate. Based on the

Table 2: Comparison of deep neural networks

Name of network	Size of sample	Speed of CPU	Speed of GPU	Success rate of set 1	Success rate of set 2
GoogleNet	32x32	1.1s	3.7ms	98.83%	99.01%
GoogleNet	64x64	1.2s	5.3ms	99.10%	98.90%
GoogleNet	224x224	1.4s	11.7ms	99.64%	98.96%
Resnet	256x256	1.7s	13.3ms	97.18%	92.75%

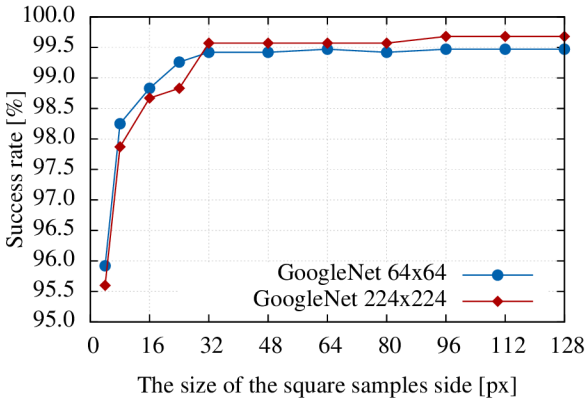


Figure 8: Dependence of success rate on samples size

obtained results, it is possible to reduce data flow from cameras to the central unit as much as possible; then the samples can be enlarged again in the central unit to the required input size of the network. From graph 8, it can be seen that it is possible to reduce the samples as much as to the size of 32x32 pixels, and the *GoogleNet 224x224* network will still have a success rate of more than 99.5%. In case of further reduction, the *GoogleNet 64x64* network becomes more successful. An interesting fact is a success rate of almost 96% for samples of the size of 4x4 pixels. Such a high success rate can be achieved primarily thanks to the fact that both the training and the testing sets were obtained from only one car park and it is therefore likely that the network is not general enough.

6. Conclusion

This paper described a system for classification of parking space occupancy and introduced two basic methods that are used: a deep neural network and a background model. The resulting application achieves a very good success rate of 99.6% on the testing set and is being tested in real operation. The integration of the solution of parking space occupancy detection into a higher whole will make it easier for the drivers to find a vacant parking space, eliminate undesirable traffic in the car park and last but not least, will also reduce the production of CO_2. It can also provide the car park managers with a lot of information about occupancy times, utilization of specific spaces and more.

In the continued development, the goal will be to increase the success rate by increasing the training set and generalizing the trained models by adding data from other car parks. Based on the requirements of the target customer, other functionalities of the application will be added, such as the expected time when the parking space will become vacant, advanced statistics about car park utilization etc. The system is currently being adjusted to work on an integrated system with limited computer sources.

7. Acknowledgements

This paper summarizes a part of a masters diploma thesis that is being created based on a company assignment under the supervision of Ing. Jaroslav Rozman, PhD.[3] on the part of FIT BUT and Ing. David Herman[4] on the part of the company RCE systems s.r.o.

References

[1] Bradski, G., Kaehler, A.. Learning OpenCV: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, Inc.; 2nd ed.; 2013. ISBN 1449314651, 9781449314651.

[2] Blanchette, J., Summerfield, M.. C++ GUI Programming with Qt 4. Upper Saddle River, NJ, USA: Prentice Hall PTR; 2006. ISBN 0131872494.

[3] Dokur, O., Katkoori, S., Elmehraz, N.. Embedded system design of a real-time parking guidance system. In: 2016 Annual IEEE Systems Conference (SysCon). 2016, p. 1–8. doi:\bibinfo{doi}{10.1109/SYSCON.2016.7490653}.

[4] Zhang, Z., Tao, M., Yuan, H.. A parking occupancy detection algorithm based on amr sensor. IEEE Sensors Journal 2015;15(2):1261–1269. doi:\bibinfo{doi}{10.1109/JSEN.2014.2362122}.

[5] Åoli, P., Marasovi, I., Stefanizzi, M.L., Patrono, L., Mainetti, L.. Rfid-based efficient method for parking slot car detection. In: 2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM). 2015, p. 108–112. doi:\bibinfo{doi}{10.1109/SOFTCOM.2015.7314120}.

[6] Zhou, H., Li, Z.. An intelligent parking management system based on RS485 and RFID. In: 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). 2016, p. 355–359. doi:\bibinfo{doi}{10.1109/CyberC.2016.74}.

[7] Sifuentes, E., Casas, O., Pallas-Areny, R.. Wireless magnetic sensor node for vehicle detection with optical wake-up. IEEE Sensors Journal 2011;11(8):1669–1676. doi:\bibinfo{doi}{10.1109/JSEN.2010.2103937}.

[8] Prasad, A.S.G., Sharath, U., Amith, B., Supritha, B.R., Asokan, S., Hegde, G.M.. Fiber bragg grating sensor instrumentation for parking space occupancy management. In: 2012 International Conference on Optical Engineering (ICOE). 2012, p. 1–4. doi:\bibinfo{doi}{10.1109/ICOE.2012.6409571}.

[9] Yusnita, R., Norbaya, F., Basharuddin, N.. Intelligent parking space detection system based onimage processing. International Journal of Innovation, Management and Technology 2012;3(3):232.

[10] Fusek, R., Sojka, E., Mozdřeň, K., Šurkala, M.. Energy based descriptors and their application for car detection. In: Computer Vision Theory and Applications (VISAPP), 2014 International Conference on; vol. 1. IEEE; 2014, p. 492–499.

[11] Maria, G., Baccaglini, E., Brevi, D., Gavelli, M., Scopigno, R.. A drone-based image processing system for car detection in a smart transport infrastructure. In: Electrotechnical Conference (MELECON), 2016 18th Mediterranean. IEEE; 2016, p. 1–5.

[12] Amato, G., Carrara, F., Falchi, F., Gennaro, C., Vairo, C.. Car parking occupancy detection using smart camera networks and deep learning. In: 2016 IEEE Symposium on Computers and Communication (ISCC). 2016, p. 1212–1217. doi:\bibinfo{doi}{10.1109/ISCC.2016.7543901}.

[13] Jermsurawong, J., Ahsan, M.U., Haidar, A., Dong, H., Mavridis, N.. Car parking vacancy detection and its application in 24-hour statistical analysis. In: 2012 10th International Conference on Frontiers of Information Technology. 2012, p. 84–90. doi:\bibinfo{doi}{10.1109/FIT.2012.24}.

[14] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., et al. Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software. 2009,.

[15] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22Nd ACM International Conference on Multimedia. MM '14; New York, NY, USA: ACM. ISBN 978-1-4503-3063-3; 2014, p. 675–678. doi:\bibinfo{doi}{10.1145/2647868.2654889}. URL `http://doi.acm.org/10.1145/2647868.2654889`.

[16] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, p. 1–9.

[3]Faculty of Information Technology, Brno University of Technology
[4]RCE Systems, s.r.o., Brno, Czech Republic