ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra radioelektroniky

Distribuovaný kamerový systém pro správu parkovacích míst

DIPLOMOVÁ PRÁCE

Vedoucí práce: Ing. Stanislav Vítek, Ph.D.

Autor: Bc. Petr Melničuk, KME, Multimediální technika

Praha 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Melničuk** Jméno: **Petr** Osobní číslo: **406094**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Multimediální technika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Distribuovaný kamerový systém pro správu parkovacích míst

Název diplomové práce anglicky:

The Distributed Camera System for the Management of Parking Slots

Pokyny pro vypracování:

Navrhněte a implementujte distribuovaný kamerový systém, který bude sloužit ke správě parkovacích míst. Při vypracování se řiďte následujícími pokyny:
1) Proveďte studii dostupných řešení a vhodných technologií. Diskutujte vliv jednotlivých komponent na efektivitu a přesnost uvažovaného řešení.
2) Na základě studie navrhněte kamerový systém. Systém se bude skládat z kamerových modulů, které budou schopny vyhodnocovat obsazenost parkovacích míst.
3) Implementujte vhodný způsob kalibrace, který umožní překrývání zorných polí kamer.
4) Navrhněte vhodný způsob komunikace mezi moduly a předávání informací o obsazenosti.

Seznam doporučené literatury:

[1] AGHAJAN, Hamid; CAVALLARO, Andrea (ed.). Multi-camera networks: principles and applications. Academic press, 2009.
[2] SONG, Dezhen. Sharing a vision: systems and algorithms for collaboratively-teleoperated robotic cameras. Springer, 2009.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Stanislav Vítek Ph.D., 13137

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **10.02.2017** Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce: **30.08.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Dne 25. května 2017 v Praze

.....

Bc. Petr Melničuk

Poděkování

Děkuji panu Ing. Stanislavovi Vítkovi, Ph.D. za odborné vedení během celého mého studia na vysoké škole, vstřícnost a rady při vypracovávání této diplomové práce.

Obsah

Abstrakt

Tato diplomová práce se zabývá distribuovaným kamerovým systémem pro správu parkovacích míst. Cílem práce je návrh a implementace všech částí systému počínaje inteligentními kamerovými moduly a konče algoritmy, které jsou schopny vyhodnocovat obsazenost parkovacích míst na základě informací z více kamer. Práce představuje malé kamerové moduly sestavené z Raspberry Pi Zero a výpočetně nenáročný mechanismus extrakce obsazenosti parkovacích míst z obrazu na bázi deskriptoru HOG a klasifikátoru SVM. Navržené řešení umožňuje rozpoznávat obsazenost parkovacích míst rychlostí 10 míst za sekundu s přesností vyšší než 90 % v různorodých podmínkách. Spolehlivost algoritmů je ověřena na třech rozdílných testovacích sadách, které dohromady obsahují přes 700 000 snímků parkovacích míst.

Klíčová slova
správa parkovacích míst, počítačové vidění, distribuované kamerové systémy, inteligentní dopravní systémy, parkování, doprava v klidu, Raspberry Pi Zero, knihovna OpenCV, HOG, SVM, Python

Abstract

This master thesis deals with distributed camera system for the management of parking slots. The goal of the thesis is to design and to implement every part of the system from intelligent camera modules to algorithms, which can determine occupancy of the parking slots based on the information from multiple cameras. The thesis introduces small camera modules based on Raspberry Pi Zero and computationally efficient mechanism for the occupancy detection based on the HOG descriptor and SVM classifier. The proposed solution can deliver occupancy information at the rate of 10 parking slots per second with more than 90 % accuracy in a very wide range of conditions. Reliability of the implemented algorithm is evaluated with three different test sets which altogether contain over a 700 000 parking slot images.

Key Words
management of parking slots, computer vision, distributed camera networks, intelligent traffic systems, parking, Raspberry Pi Zero, OpenCV library, HOG, SVM, Python

Seznam používaných zkratek

ARM	advanced RISC machine
AUC	area under curve
CCD	charge-coupled device
CMOS	complementary metal-oxide-semiconductor
CPU	central processing unit
CSI	camera serial interface
DSI	display serial interface
FFC	flexible flat cable
FNR	false negative rate
FOV	field of view
FPR	false positive rate
GPIO	general purpose input/output
GPU	graphics processing unit
HD	high definition
HDMI	high definition multimedia interface
HOG	histogram of oriented gradients
HW	hardware
I2C	inter-integrated circuit
ID	identification
IoT	Internet of Things
LBP	local binary patterns
LBQ	local phase quantization
Li-Ion	lithium-ion
MIPI	mobile industry processor interface
PC	personal computer
RAM	random access memory
RGB	red, green, blue
RISC	reduced instruction set computing
ROC	receiver operating characteristic
RPi	Raspberry Pi
SD	standard definition
SIFT	scale-invariant feature transform
SoC	system on chip
SPI	serial peripheral interface
SSH	secure shell
SVM	support vector machine
TNR	true negative rate
TPR	true positive rate
UART	universal asynchronous receiver/transmitter
USB	universal serial bus
ZIF	zero insertion force

1. Úvod

Zásluhou snižující se ceny kamerových senzorů a výpočetních systémů je v dnešní době možné realizovat relativně levné rozsáhlé kamerové sítě. Síť mnoha kamer je schopná monitorovat velkou rozlohu a poskytovat užitečné informace v celé řadě aplikací. Zvětšování počtu kamer vybízí k přechodu od tradičnějšího, centralizovaného způsobu zpracování dat, k distribuovanému, kde každý kamerový uzel disponuje určitým výpočetním výkonem a podílí se na zpracování dat. [1]

Tato diplomová práce se zabývá distribuovaným kamerovým systémem pro správu parkovacích míst, ve kterém inteligentní kamery transformují obrazová data na zprávy obsahující informace o obsazenosti parkoviště. Zprávy jsou datově nenáročné a pro jejich přenos postačuje i kanál s velmi malou kapacitou. Informace mohou být zasílány řidičům, prostřednictvím Internetu a zároveň se ukládat pro účely vytváření statistik. Téma práce je multidisciplinární a týká se zejména následujících oborů:

- inteligentní dopravní systémy;
- počítačové vidění a zpracování signálů;
- kabelové a bezdrátové kamerové sítě;
- vestavěné počítačové systémy;
- multimediální technika.

Motivací k zabývání se touto problematikou je častá situace při parkování ve městech, kdy je obtížné najít volné parkovací místo. Tento problém není způsoben pouze nedostatkem legálních ploch pro stání vozidel, ale často také špatnou regulací parkovacích cen. [2]

V oblastech, kde je parkování zadarmo anebo je velmi levné, může docházet k zaplnění všech dostupných parkovacích míst. Řidiči, kteří zde chtějí zaparkovat, obvykle frustrovaně nějaký čas krouží oblastí s nadějí, že dojde k uvolnění alespoň jednoho místa. Kroužením však řidiči zhušťují provoz na komunikacích a zhoršují kvalitu ovzduší nadbytečnými emisemi. Naopak příliš vysoká cena parkování způsobí, že hodně parkovacích míst zůstane volných a plocha vyhrazená pro parkování je tak neefektivně využita. [2]

Vzorovou ukázkou toho, jak může systém s detekcí obsazenosti pomoci regulaci cen, je projekt SFpark spuštěný v roce 2011 v San Francisco [3]. Tamní systém dokáže monitorovat 7 000 parkovacích míst v ulicích a 12 250 míst v parkovacích garážích.

Okamžitý stav parkovacích míst je nahráván na server a je k dispozici řidičům prostřednictvím mobilní aplikace. Jednou za měsíc je vyhodnoceno, jak se reálná obsazenost lišila od optimální (~ 85 % míst obsazeno) a dojde k drobným úpravám cen. Pokud byla obsazenost v oblasti příliš vysoká, tak se parkovací ceny zvýší, v opačném případě se sníží. Výhody plynoucí z nasazení takového systému pro správu parkovacích míst ve velkém měřítku, jsou např.:

- snadné a rychlé hledání parkovacího místa;
- snížení kroužení, tedy spotřeby paliva a emisí;
- získání cenných dat pro dopravní plánování;
- možnost nastavení spravedlivého systému cen;
- vyšší výnosy měst, při snížení průměrných parkovacích cen.

Cílem této práce je zejména návrh a implementace distribuovaného kamerového systému, který je schopen poskytovat údaje o obsazenosti monitorovaného parkoviště. Pro realizaci systému jsou využity stacionární inteligentní kamerové moduly založené na jednodeskovém počítači Raspberry Pi Zero, které sami dokážou transformovat obrazová data na informace o obsazenosti. Pro zapsání algoritmů je využit programovací jazyk Python především spolu s knihovnami OpenCV[1], NumPy[2] a Picamera[3]. Algoritmy využívají kombinaci deskriptoru HOG a klasifikátoru SVM. Snahou bylo připravit systém, který není specializován na žádné konkrétní parkoviště a dokáže úspěšně detekovat obsazenost v širokém spektru podmínek. Úspěšnost detekce je ověřena na snímcích aut v ulicích, časosběrných sekvencích fakultního parkoviště a také na databázi PKlot [4].

Členění dokumentu je následující. Druhá kapitola se zabývá základním rozdělením systémů pro správu parkovacích míst a dostupnými technologiemi pro rozpoznávání obsazenosti. Ke konci této kapitoly jsou definovány kamerové systémy a jsou uvedeny výhody, které oproti ostatním systémům přinášejí. Třetí kapitola pojednává o kamerových systémech z praktických hledisek a navrhuje vhodné konfigurace systémů. Ve čtvrté kapitole jsou uvedeny metody zpracování a klasifikace dat. Součástí této kapitoly je především teoretický rozbor souvislostí spjatých s tématikou. Praktičtější část práce začíná od páté kapitoly popisem návrhu a realizace kamerových modulů na bázi Raspberry Pi Zero. V šesté kapitole jsou popsány vytvořené algoritmy a zvolené postupy pro dosažení funkčního systému. Sedmá kapitola se zabývá výsledky implementovaných algoritmů a hodnotí jejich úspěšnost. Osmá kapitola obsahuje shrnutí celé této práce.

[1] *OpenCV*: http://opencv.org/

[2] *NumPy*: http://www.numpy.org/

[3] *PiCamera*: https://picamera.readthedocs.io/en/release-1.13/

2. Systémy pro správu parkovacích míst

Základní funkcí systémů pro správu parkovacích míst je detekce obsazenosti, tedy detekce přítomnosti vozidla na určitém místě nebo v určité oblasti. Podle typu lokace, kde vozidlo stojí, lze parkování rozdělit na tzv. ***on-street*** *(*podél dopravního toku) a ***off-street*** (mimo komunikace, tedy v prostorech speciálně vybudovaných pro účel odstavování vozidel např.: parkovací garáže, parkoviště u obchodních center).

U ***off-street*** parkování bývá obvyklé důsledné označení jednotlivých stání, a lze tedy dobře očekávat, kde a jak budou vozidla umístěna. Rozměry stání a přilehlých manévrovacích prostor jsou v ČR stanoveny normou[4]. Při stanovení šířek a délek parkovacích míst se vychází především z rozměrů vozidel a způsobu jejich řazení (kolmé, podélné, šikmé). [5]

V případě ***on-street*** parkování nemusí vyznačení jednotlivých míst existovat a umístění vozidel je nutné odhadovat. Tento způsob parkování s sebou často nese nevýhody pro území, ve kterém je realizován (snížení bezpečnosti provozu, omezení pro pěší, ztráta velké plochy, problémy se správou komunikace) [5], ale například v Praze jde o častý typ parkování. Z pohledu systémů pro správu parkovacích míst představuje *on-street* parkování větší problém než *off-street* parkování, neboť veškeré podmínky, které by mohly souviset s nasazením systémů jsou hůře definovatelné. Vozidla mohou stát na různých podložích, pod různým úhlem a náklonem. V bezprostřední blízkosti stání může probíhat silniční, tramvajová, či vlaková doprava. V okolí se také může vyskytovat vegetace a místa od sebe mohou být nerovnoměrně oddělena (chodníky, sloupy, nerovnosti atd.).

Tato práce je primárně zaměřena na *off-street* parkování, ale principy používané v implementaci by šlo s vhodným rozšířením aplikovat i na parkování typu *on-street*.

Přítomnost vozidel je možné zjišťovat celou řadou detektorů pracujících na různých fyzikálních principech (aktivní infračervené, pasivní infračervené, ultrazvukové, indukční, magnetometrické, piezoelektrické, pneumatické, váhové, mikrovlnné, akustické, obrazové atd.). V dopravních měřeních se detektory v základu dělí na intrusivní a neintrusivní, podle toho, zda zasahují, nebo nezasahují do vozovky. Pro zjišťování obsazenosti parkovacích míst se v dnešní době využívá obou typů. [6][7]

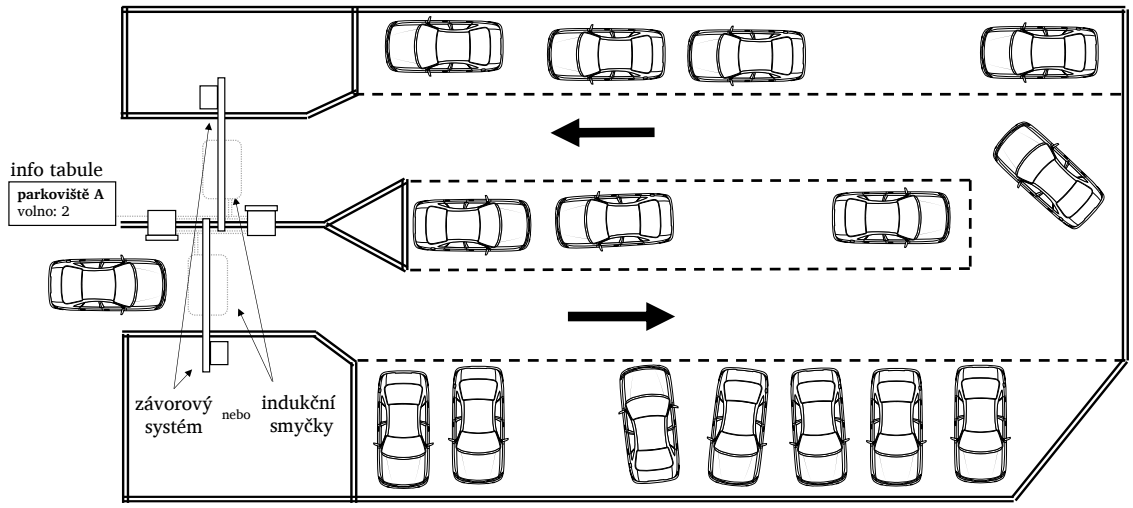
[4] ČSN 73 6056. *Odstavné a parkovací plochy silničních vozidel.* Úřad pro technickou normalizaci, 2011.

Podle [8] lze systémy pro správu parkovacích míst rozdělit celkem do čtyř kategorií na: čítačové, s drátovými senzory, s bezdrátovými senzory a na kamerové s počítačovým viděním.

2.1 Čítačové systémy

Čítačové systémy jsou nejjednodušší variantou a fungují na principu počítání aut při vjezdu na parkoviště a při výjezdu z parkoviště. Typicky používanými senzory jsou například indukční smyčky ve vozovce nebo mechanická závora.

Přesnost tohoto řešení je velmi vysoká, ale nevýhodou je, že může fungovat pouze v uzavřené oblasti. Zároveň nedokáže poskytovat informace o obsazenosti konkrétních míst, a tedy navigovat řidiče k neobsazeným místům. Systém dokáže pouze určovat počet aut na celém parkovišti. [8]



Obr. 1: Schéma čítačového systému

2.2 Systémy s drátovými senzory

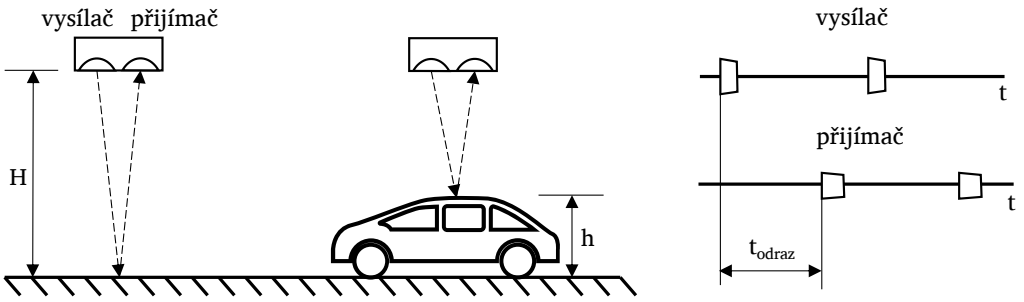
Systémy s drátovými senzory využívají obvykle konfiguraci jednoho detektoru na jedno parkovací místo. Tím překonávají kvalitou čítačové systémy, protože dokáží určovat obsazenost konkrétních míst. V této kategorii se například používají pasivní, či aktivní infračervené detektory, mikrovlnné radary, nebo ultrazvuk. [8][9]

Pro detekci se pravděpodobně nejvíce využívá ultrazvuk (Obr. 2). Vysílač umístěný ve výšce H produkuje krátké pulzy zvuku v periodických intervalech. Vyslaný zvuk se šíří prostředím rychlostí c_{zvuk}, odráží se od překážky s výškou h a vrací se do přijímače. Doba trvání mezi vyslaným a přijatým zvukem t_{odraz} odpovídá vzdálenosti překážky od které se zvuk odrazil. Výšku překážky lze snadno vypočítat podle vztahů uvedených u schématu.

V případě obsazeného místa se bude detekovaná výška blížit výšce automobilu a v případě volného místa se bude blížit nule. [6]

$$h \cong H - \frac{(c_{zvuk} \cdot t_{odraz})}{2} \text{ [m]}$$
$$c_{zvuk} \cong (331{,}8 + 0{,}61 \cdot \vartheta) \text{[m/s]}$$

H ... výška umístění detektoru [m]
ϑ ... teplota vzduchu [C°]



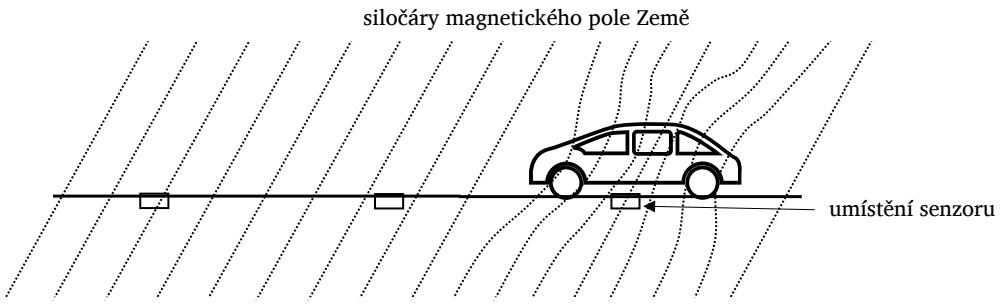
Obr. 2: Schéma ultrazvukového systému

S ultrazvukovými senzory se lze nejčastěji setkat ve vnitřních garážích. Garáže jednak poskytují dobré možnosti uchycení detektorů přímo nad parkovací místa a zároveň se v nich málo mění klimatické podmínky, a tedy i rychlost zvuku. Nevýhodami řešení s drátovými senzory jsou především vyšší pořizovací náklady (mnoho senzorů, kabeláž, instalace) a doba potřebná pro zprovoznění. Přesnost závisí na parametrech prostředí, použitých technologiích a způsobu zpracování signálů. [6][8]

2.3 Systémy s bezdrátovými senzory

Systémy s bezdrátovými senzory jsou napájeny pomocí baterií a komunikují obvykle prostřednictvím radiových vln. U bezdrátových senzorů se stejně jako u drátových využívá konfigurace jednoho senzoru na jedno parkovací místo.

Typicky používanými zařízeními jsou senzory pracující s detekcí vlastností magnetického pole. Pro maximalizaci výdrže na baterii se obvykle jedná o pasivní magnetorezistivní senzory, které nevyzařují žádnou energii, ale pouze detekují změny v magnetickém poli Země. Optimální umístění takového senzoru je do středu parkovacího místa. Feromagnetické části vozidla deformují pole a zvyšují kolem něj hustotu siločar, detekovatelnou jako nárůst absolutní hodnoty magnetické indukce (Obr. 3). [10]

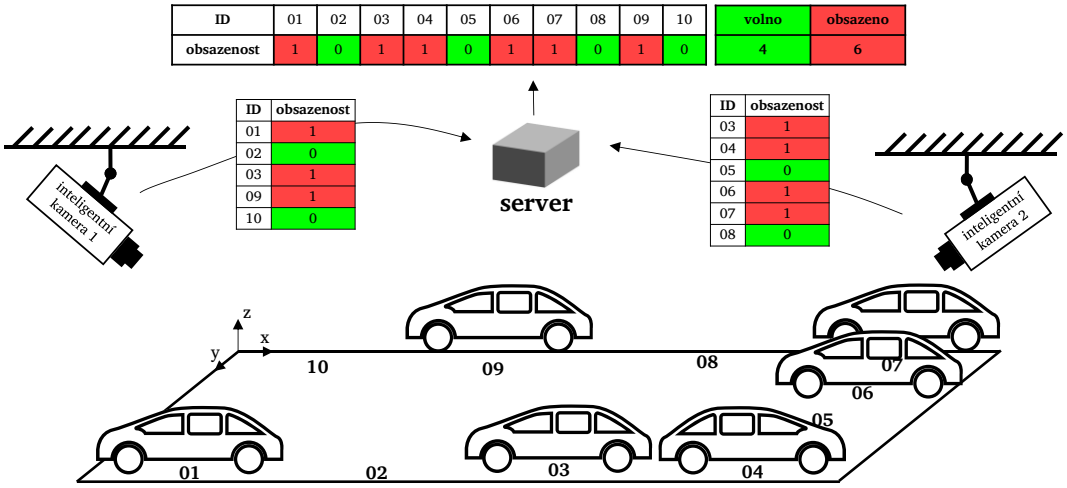


Obr. 3: Schéma systému s magnetorezistivním senzorem

Senzory se zapouštějí přímo do vozovky nebo se k ní přichycují, jedná se tedy o detektory intrusivní [7]. Odolnost senzoru je vysoká a výhodou je i to, že svým vzhledem neláká vandaly. Instalace je možná na libovolné místo, ať už do vnitřních, či venkovních prostor. Nevýhodami bezdrátových systému s detekcí magnetického pole jsou opět poněkud vyšší náklady spojené s pořizovací cenou samotných senzorů, instalací a dále pak také nutnost výměny baterie přibližně po 3 až 5 letech provozu. Další nevýhodou může být snížení přesnosti v případě instalace do prostředí s rušivými kovovými elementy v okolí senzoru.

2.4 Kamerové systémy s počítačovým viděním

Kamerové systémy s počítačovým viděním spadají do skupiny neintrusivních detektorů a poskytují moderní přístup k detekci obsazenosti parkovacích míst. Možné schéma takového systému je znázorněné níže na Obr. 4.



Obr. 4: Schéma distribuovaného kamerového systému pro správu parkovacích míst

Systémy pro svou funkci využívají počítačového vidění. Počítačové vidění je rychle rostoucí odvětví z oblasti výpočetní techniky a snímaní digitálního obrazu. Jde o souhrnný název pro získávání, zpracovávání, analyzování a rozpoznávání obrazových dat. Obecně se dá říci, že počítačové vidění je transformace dat digitálního obrazu

na formu dat, která odpovídá požadované aplikaci [13]. Návrhem kamerových systémů se zabývá kapitola 3 a kapitola 5 se pak zabývá návrhem kamerového systému vytvořeného v rámci této práce. V porovnání s již zmiňovanými senzory nabízejí kamerové systémy určité podstatné výhody, byť existují i určité nevýhody: [8]

Výhody:

- jeden senzor (kamera) na mnoho míst;
- rychlá instalace a údržba systému;
- o řád nižší cena [12];
- snadná rozšiřitelnost;
- potenciál využití kamer i k dalším účelům (detekce incidentů, bezpečnostní dohled).

Nevýhody:

- náchylnost k vandalismu;
- energeticky náročnější senzory [11];
- možnost nepřesností v případě špatných klimatických podmínek.

Výčet dalších výhod, především z ekonomického pohledu, lze najít na produktových stránkách vybraných komerčně dostupných řešení pro správu parkovacích míst pomocí kamerových systémů (Park Assist[5], Streetline[6] a Parquery[7]). Výrobci obvykle pochopitelně neuvádějí technické detaily jejich systémů, ale rozvádějí dobře možnosti jejich použití.

[5] Park Assist: https://www.parkassist.com/

[6] Streetline: https://www.streetline.com/

[7] Parquery: http://parquery.com/

3. Aspekty návrhu kamerových systémů

Návrh spolehlivého kamerového systému s detekcí obsazenosti musí brát v úvahu mnoho faktorů. Je potřeba umístit vybrané snímací zařízení do prostoru, rozhodnout o způsobu jeho připojení a promyslet, co vše může při dané konfiguraci ovlivňovat výsledný vzhled snímku parkovacího místa.

3.1 Připojení a napájení

Pro schopnost předávání dat musí být kamery nutně připojeny do komunikační sítě. V základním rozdělení mohou být kamery **drátové** a **bezdrátové**.

Drátové připojení umožňuje vyšší a stabilnější přenosové rychlosti při nižší spotřebě energie. Tento typ připojení je oprávněný především u systémů, kde se obraz z kamer ukládá i zpracovává centrálně. Centralizovaný způsob zpracování dat však není optimální z hlediska modularity systému. Velký počet kamerových uzlů zvyšuje celkové nároky na síť a na výpočetní výkon centrálního serveru [12]. Decentralizace je možné dosáhnout distribuováním výpočetních úkonů jednotlivým kamerám. Inteligentní kamery dokáží transformovat obraz na úspornější datovou reprezentaci (obsazenost parkoviště) a tím snížit nároky na HW centrálního prvku a komunikační sítě.

V případě distribuovaného systému z ilustrace (Obr. 4) stačí pro popis obsazenosti jednoho parkovacího místa pouze 1 bit (obsazeno/volno). Přenos takové informace lze vhodně realizovat pomocí **bezdrátových sítí** pro IoT [11]. Například platforma Sigfox[8] umožňuje přenášet 96 bitové zprávy 140krát za den, což při pravidelném zasílání zpráv přibližně odpovídá aktualizačnímu intervalu 10 minut. Tento interval lze pro lepší zachycení situace ve špičce adaptivně měnit a zasílat zprávy v nepravidelných intervalech, podle významu změn ve stavu obsazenosti.

U kompletně **bezdrátových kamer** je kromě samotné komunikace potřebné řešit i jejich napájení. Kamery obvykle spotřebují větší množství energie než jednodušší senzory a návrh je proto komplikovanější. Základním trikem pro snížení spotřeby energie je využití spacích režimů s periodickým buzením. Aktualizaci stavu obsazenosti parkoviště není nutné provádět s vysokou vzorkovací frekvencí, neboť se předpokládají pomalejší změny. Při spacím režimu lze minimalizovat spotřebu na hodnoty v řádu jednotek mW a tím výrazně prodloužit výdrž baterie. Dalším trikem pro optimalizaci výdrže mohou být solární panely, se kterými lze dosáhnout úplné energetické soběstačnosti kamery. [11]

[8] *Sigfox* [online], [cit. 2017-05-07]. Dostupné z: http://www.sigfox.com/en/sigfox-iot-technology-overview

3.2 Parametry snímacích zařízení

Dalším hlediskem návrhu systémů jsou parametry snímacích zařízení. Získávají se digitální obrazová data, která si s sebou nesou určité charakteristiky související zejména se snímacím zařízením a snímanou scénou. Konfigurace snímacího zařízení a jeho odezva na snímanou scénu mají podstatný vliv na kvalitu obrazových dat.

V současnosti jsou k dispozici relativně levná zařízení se CCD nebo CMOS sensory s rozlišením několika milionů bodů. Rozlišení je často udáváno na prvních místech ve specifikacích, ačkoliv se obvykle nejedná o nejdůležitější parametr. Pro rigoróznější popsání zařízení je potřeba zvažovat další parametry, jako například:

- parametry objektivu;
- rozlišovací schopnost;
- šumové charakteristiky snímače;
- citlivost a přenosová funkce snímače;
- kompresní standard.

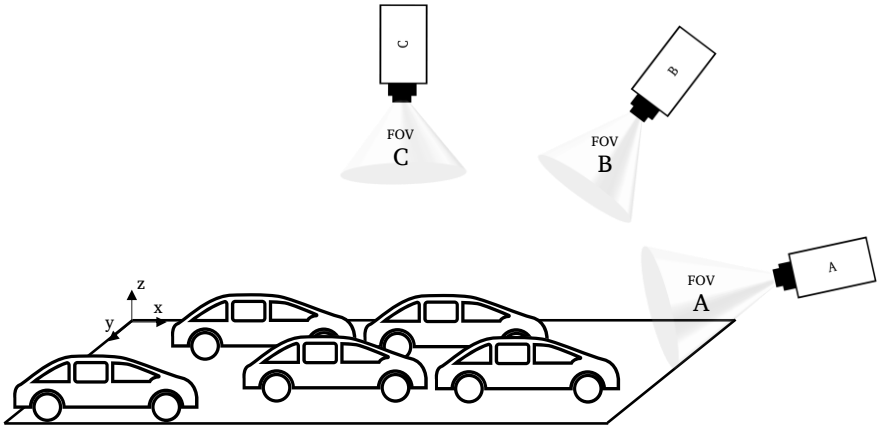
Co se týče rozlišení, tak má smysl se ptát, zda je jeho zvyšování vždy užitečné. Je potřeba mít na paměti, že zejména u menších snímačů nepřináší zvyšování rozlišení podstatné zvýšení detailů snímku. Mezní rozlišovací schopnost jakéhokoliv optického systému je totiž omezena difrakcí. Rozostření difrakcí se projeví tím více, čím menší bude vzdálenost sousedních pixelů na snímači a čím větší bude clonové číslo objektivu. Rozlišovací schopnost většiny systémů je však omezena ještě před vlivem samotné difrakce, hlavně nedokonalostmi optiky.

3.3 Geometrické uspořádání

Podstatnou součástí návrhu kamerových systémů je uvážení vzájemné pozice kamer a snímaných objektů. Je žádoucí najít co nejefektivnější řešení maximalizující pokrývanou plochu s minimálním počtem kamer. Návrh však musí brát v potaz další důsledky spjaté s umístěním kamery do určité pozice [1]. Ilustrační snímek (Obr. 5) naznačuje celkem tři vybraná umístění kamery vůči rovině parkoviště. To, zda kamera zabere všechna parkovací místa, záleží kromě jejího umístění také na jejím zorném poli (FOV). Velké FOV je spojené s širokoúhlými objektivy, které umožňují zabrat větší plochu, což však může být za cenu nechtěných geometrických zkreslení.

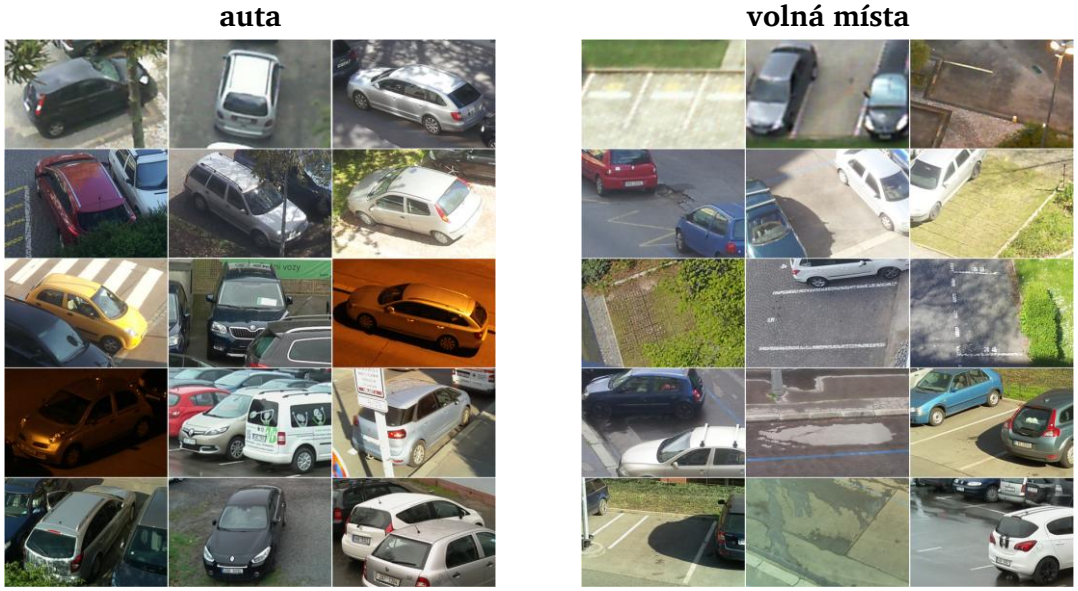
Kamera A snadno zabere všechna parkovací místa, ale za cenu silných překryvů, které zhoršují rozlišitelnost obsazenosti. **Kamera C** dokáže zachytit parkovací místa bez

překryvu, ale všechny se do jejího záběru vejdou pouze při umístnění kamery vysoko anebo při použití objektivu s velkým FOV. V záběrech z **kamery B** se mohou vyskytovat překryvy, které však nebudou tak výrazné jako v případě kamery A. Všechna parkovací místa lze zabrat i s méně širokoúhlým objektivem z nižší výšky než v případě kamery C. Varianta B se jeví jako nejvýhodnější, neboť jde o kompromis mezi velikostí záběru a závažností překryvů. Kamery mohou být umístěny na sloupy veřejného osvětlení, případně na okolní budovy.



Obr. 5: Vybrané varianty umístění kamery vůči rovině parkoviště

I při vymezení výšky kamery a jejího náklonu vůči parkovací ploše existuje stále mnoho stupňů volnosti, jak mohou parkovací místa z pohledu kamery vypadat (Obr. 6). Pro návrh obecně pracujícího systému je potřeba mít na paměti, že finální vzhled výřezu parkovacího místa je nadále ovlivňován mnoha dalšími faktory (typy aut, natočením aut, počasím, vlivy světla, podložím parkovacího místa, překryvy atd.).



Obr. 6: Variace ve vzhledu výřezu parkovacích míst

4. Metody zpracování a klasifikace dat

Tato kapitola se zabývá teoretickým rozborem problematiky z pohledu zpracování a klasifikace dat, získávaných z kamerových systémů.

4.1 Binární klasifikace

Problém automatického rozpoznávání obsazenosti parkovacích míst lze, nehledě na typ používaného systému a princip detekce, zařadit do problematiky binární klasifikace. Binární klasifikace se týká problematiky mapování prvků z množiny X do dvou předem daných tříd {auta-obsazeno, volná místa} na základě klasifikačního pravidla. Podklady pro tuto kapitolu pochází především z [14] se vztažením informací k tématice práce.

$$X = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}\ je\ vzorek\ dat\ naměřený\ specifickým\ typem\ senzoru\}$$

Klasifikačním pravidlem je libovolně komplexní mechanismus transformace vzorku dat z X jehož finálním výsledkem je predikce třídy vzorku. Pro objektivní posouzení kvality výsledku klasifikace se je potřeba zamyslet, jakými prostředky je možné popsat schopnost klasifikátoru od sebe správně rozlišovat auta (pozitivní vzorky p) od volných míst (negativní vzorky n). U binární klasifikace vzorku dat mohou nastat celkem čtyři případy:

- skutečně pozitivní (***TP*** – *true positive*) (auto rozpoznáno) ✓
- skutečně negativní (***TN*** – *true negative*) (volné místo rozpoznáno) ✓
- falešně pozitivní (***FP*** – *false positive*) (volné místo nerozpoznáno) x
- falešně negativní (***FN*** – *false negative*) (auto nerozpoznáno) x

Při problému detekce automobilů znamená ***TP***, že na zkoumaném místě auto stojí a klasifikátor tuto situaci správně vyhodnotil. ***TN*** je druhou variantou správného výsledku a jde o případ, kdy se volné místo správně vyhodnotí jako volné. ***FP*** značí, že volné místo bylo označeno jako obsazené a ***FN*** naopak, že obsazené místo bylo označené jako volné.

4.1.1 Matice záměn

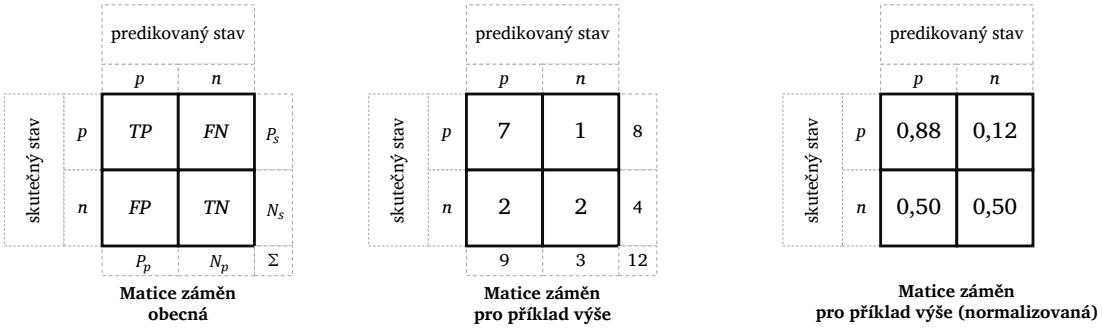
Příklad: Blíže nespecifikovaný systém byl nasazen na problém rozpoznávání obsazenosti celkem 12 parkovacích míst (p – obsazeno, n – volno). Bylo provedeno automatické vyhodnocení obsazenosti (predikce), které bylo srovnáno s reálnou obsazeností.

parkovací místo	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	$\sum p$	$\sum n$
skutečný stav	*n*	*p*	*p*	*p*	*n*	*n*	*p*	*p*	*p*	*p*	*n*	*p*	8	4
predikovaný stav	*n*	*p*	*p*	*n*	*n*	*p*	*p*	*p*	*p*	*p*	*p*	*p*	9	3
správnost predikce	*TN*	*TP*	*TP*	*FN*	*TN*	*FP*	*TP*	*TP*	*TP*	*TP*	*FP*	*TP*		

Tab. 1: Zadání ilustračního příkladu

Systém klasifikoval 9 parkovacích míst jako obsazených ($P_p = 9$) a 3 jako volné ($N_p = 3$). Ve skutečnosti bylo obsazeno 8 parkovacích míst ($P_s = 8$) a volné byly 4 ($N_s = 4$). Při znalosti skutečných stavů lze zkonstruovat tzv. matici záměn (*confusion matrix*).

Matice záměn je základním nástrojem, který slouží jako výchozí bod pro výpočet výkonnostních metrik klasifikačních systémů. Matice srovnává skutečný stav s predikovaným a uvádí ho ve vhodné formě (Obr. 7). V případě binární klasifikace se jedná o matici velikosti 2×2 obsahující distribuci správnosti predikce. Hlavní diagonála popisuje správnou predikci (*TP*, *TN*), vedlejší chybnou (*FP*, *FN*).



Obr. 7: Matice záměn

Srozumitelnější formou matice se může zdát její normalizovaná varianta (napravo), popisující míru zastoupení (***R*** – *rate*) správnosti predikce v příslušných kategoriích (P_s, N_s) pro používanou testovací sadu:

- míra skutečně pozitivních (***TPR***) $TPR = TP / P_s$
- míra skutečně negativních (***TNR***) $TNR = TN / N_s$
- míra falešně pozitivních (***FPR***) $FPR = FP / N_s$
- míra falešně negativních (***FNR***) $FNR = FN / P_s$

Tyto údaje jsou však zbaveny informace o počtech P_s a N_s, což znemožňuje výpočet některých užitečných metrik. Jednou z metrik, u které je potřeba vycházet z originální (nenormalizované) matice je **přesnost (*ACC* – *accuracy)***.

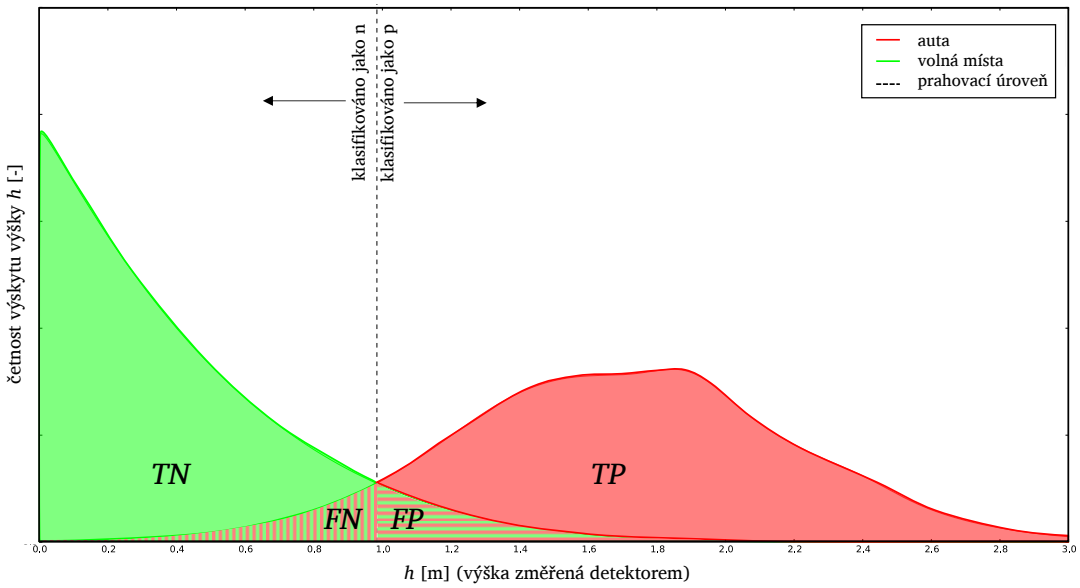
$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P_s + N_s} = \frac{TP + TN}{P_n + N_n} = \frac{7 + 2}{12} = 75\ \%\ \text{(pro příklad výše)}$$

ACC udává relativní zastoupení všech správných výsledků v celé testovací sadě. Jedná se o jednočíselný parametr, a proto se dobře udává do výsledků, může však být zavádějící, protože se jeho hodnota mění v závislosti na poměru počtu pozitivních a negativních dat.

Doposud zmíněná měřítka úspěšnosti klasifikace jsou konkrétním výsledkem pro jedno nastavení klasifikátoru. Většina klasifikačních mechanismů umožňuje pro finální rozhodnutí o třídě testovaného vzorku nastavit nějakou rozhodovací úroveň neboli práh.

4.1.2 ROC analýza

Příklad: Systém s ultrazvukovým detektorem (Obr. 2) monitoruje obsazenost parkovacích míst. Pro jedno měření je získána hodnota odpovídající změřené výšce h a zároveň skutečný stav obsazenosti. Naměřením dostatečného množství hodnot se získá distribuce výsledků. V důsledku různé výšky vozidel a vzniku chyb při měření se distribuce jednotlivých tříd mohou překrývat. Velikost a způsob překryvu předurčuje k jakým chybám bude při klasifikaci docházet. Určením pevného prahu (např. $h = 1$ m) se nastaví pracovní bod, kterému odpovídá konkrétní matice záměn (Obr. 8).



Obr. 8: Simulace distribuce hodnot naměřených detektorem

Posunem pracovního bodu se mění matice záměn, tedy rozložení typu chyb. Při nízkém prahu je vysoká pravděpodobnost, že bude auto detekováno (*TPR*), ale je zároveň vysoká šance, že budou jako auta klasifikovány i vzorky, které autům neodpovídají (*FPR*).

Na druhou stranu při vysokém prahu je klasifikátor konzervativnější a jako auto označí pouze vzorek, který mu odpovídá s velkou jistotou. Níže v tabulce (Tab. 2) jsou naznačeny změny přesnosti a matice záměn pro několik diskrétních prahů.

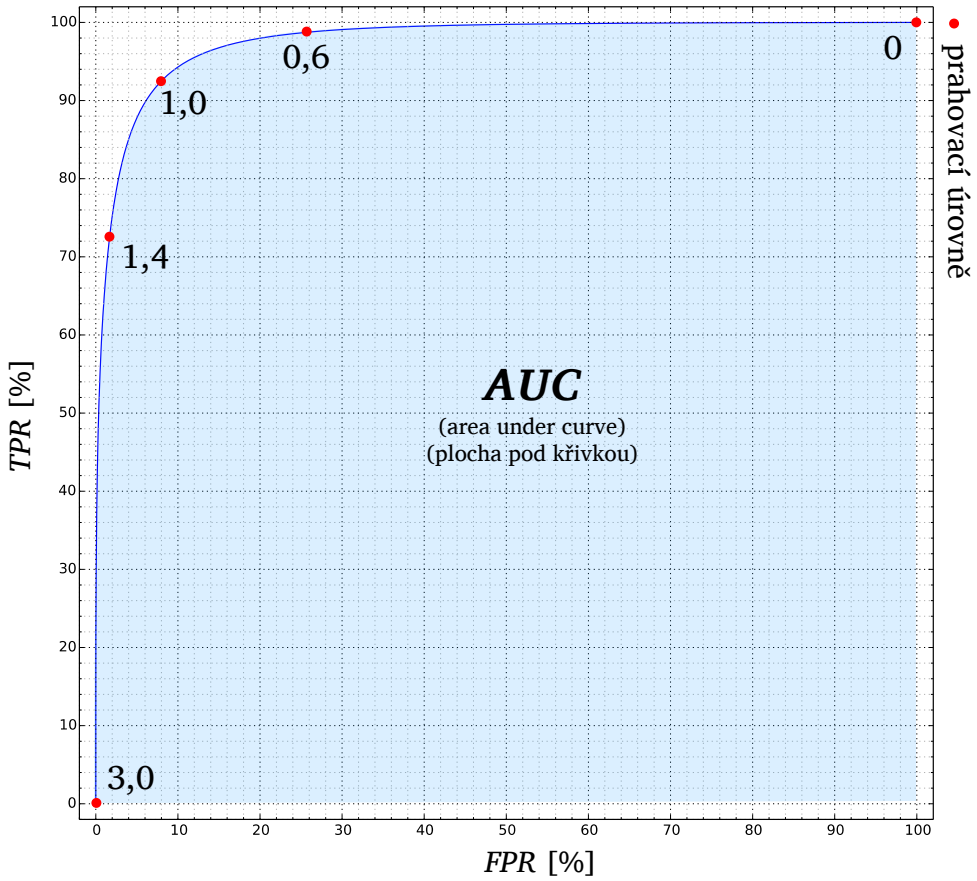
práh [m]	**0**	**0,6**	**1,0**	**1,4**	**3,0**
TPR [%]	100	98,7	92,4	72,6	0
FPR [%]	100	25,7	7,9	1,7	0
FNR [%]	0	1,3	7,6	27,4	100
TNR [%]	0	74,3	95,1	98,3	100
ACC [%]	50	86,5	92,3	85,4	50

Tab. 2: Přesnost a matice záměn v závislosti na prahu

Nejvyšší přesnosti dosahuje klasifikátor v místě překryvu distribucí ($h \cong 1$ m). Maximální přesnost však nemusí být nejdůležitějším kritériem pro výběr prahu.

Při předávání informace o obsazenosti řidičům je žádoucí minimalizovat počet situací, kdy je parkovací místo obsazené a systém ho indikuje jako volné (*FN*). Posun prahu k nižším hodnotám tento pokles *FN* zajistí, ale zároveň se tím zvýší *FP*, tedy počet situací, kdy je parkovací místo volné a systém ho označuje jako obsazené. Vysoká hodnota *FP* je také nežádoucí, ale pro řidiče jde o méně nepříjemnou situaci. I při nižší přesnosti se tedy systém může jevit jako spolehlivější.

Pro posouzení volby prahu, lze sestavit tzv. **ROC křivku**, která znázorňuje vztah mezi výnosy (*TPR*) a náklady (*FPR*), pro všechny možné hodnoty prahu. Křivka slouží jako účinný nástroj pro analýzu úspěšnosti algoritmu a dokáže poskytovat stále stejné údaje nezávisle na poměru počtu pozitivních a negativních vzorků dat v testovací sadě. Graf na Obr. 9 znázorňuje křivku ROC pro probíraný ilustrační příklad.



Obr. 9: ROC křivka pro ilustrační příklad

Čím více křivka tíhne k bodu (0, 1), tím větší je plocha pod křivkou (***AUC*** – a*rea under curve*) a dá se předpokládat úspěšnější klasifikace (hodně skutečně pozitivních a málo falešně pozitivních). Plochu lze spočítat pomocí Riemannovy sumy.

$$\boldsymbol{AUC} = \sum \mathrm{ROC}\,(FP) \cdot \Delta FP, \qquad \in <0;1>$$

Metrika *AUC* se opět hlavně hodí jako jednočíselná hodnota uváděná do výsledků. Pro hlubší pochopení úspěšnosti klasifikace je však vhodnější sledovat celou křivku.

Hodnocení klasifikace podle *AUC*:

- $AUC = 1{,}0$ … perfektní ($FP = FN = 0$);
- $AUC > 0{,}9$ … dobrá;
- $AUC > 0{,}5$ … lepší, než náhodná;
- $AUC = 0{,}5$ … náhodná;
- $AUC < 0{,}5$ … lepší, než náhodná (prohození tříd).

4.2 Zpracování obrazu

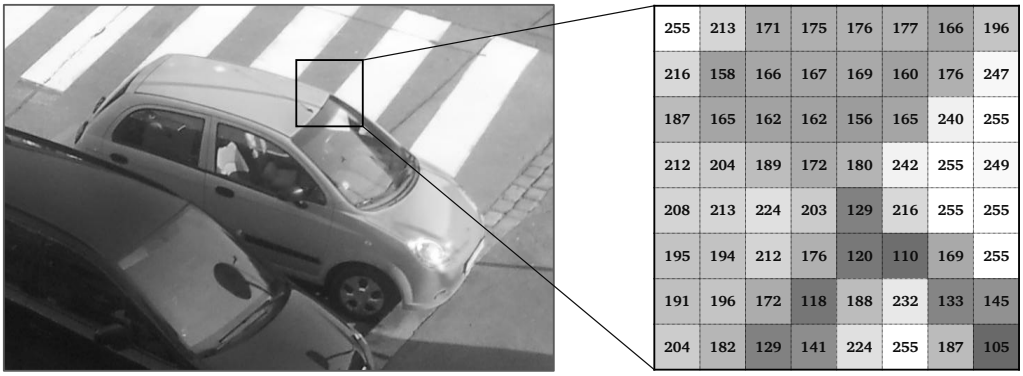
Tato kapitola se zabývá reprezentací a interpretací obrazu, a poskytuje také stručný přehled o vybraných přístupech rozpoznávání obsazenosti, dle odborných zdrojů.

4.2.1 Reprezentace a interpretace

Obsazenost parkovacích míst na jednotlivých snímcích (Obr. 6) je celkem snadno rozlišitelná při pohledu lidským zrakem. Člověk disponuje dobrými kognitivními schopnostmi a je schopen učinit velmi rychlé rozhodnutí o tom, zda je parkovací místo obsazené, či nikoliv. Jednou z nejpozoruhodnějších funkcí lidského zraku totiž je, že umožňuje lidem díky vizuálním zkušenostem detekovat a klasifikovat objekty, které nikdy dříve neviděli. Lidé se této schopnosti učí již od nejútlejšího věku. S dospíváním organismu a nárůstem vizuálních zkušeností se přirozeným učením v mozku staví velice silná a stabilní základna zrakových vjemů, která umožňuje výše zmíněnou schopnost. [15][16]

Počítačový systém se od lidského zraku značně liší a pracuje na základě jasně definovaného algoritmu, který má na vstupu digitální obrazová data. V případě barevného obrazu se obvykle jedná o trojrozměrnou matici obsahující čísla, která udávají intenzitu jednotlivých barevných složek příslušného bodu (pixelu) v obraze. Standardními barevnými složkami (kanály) jsou červená (R), zelená (G) a modrá (B), kterými lze z principu aditivního mísení barev vytvořit širokou škálu barevných odstínů. Šedotónový

obraz je dvourozměrná matice čísel, jež udávají intenzitu jednotlivých bodů (Obr. 10). Intenzita bodů bývá typicky vyjadřována 8 bitovým číslem (256 možných úrovní). [16]



Obr. 10: Reprezentace šedotónového digitálního obrazu (dvourozměrná matice)

Obvyklým postupem u velké části systémů je předzpracování vstupních obrazových dat do méně objemnější reprezentace, které se říká příznakový vektor. Příznakový vektor je předáván klasifikátoru, který činí rozhodnutí o obsazenosti. Jako příznakový vektor může posloužit výřez snímku normalizovaný na určitou velikost [12][20] nebo jiná struktura získaná různými deskriptory, které z dat extrahují vybrané vlastnosti [17][18][19][4].

4.2.2 Vybrané přístupy k interpretaci

(Hasegawa, Nohsoh, Ozawa, 1994) [17]

V jednom z historicky nejstarších systémů volí autoři pro detekci obsazenosti jako extrahovanou vlastnost pohyb. Princip jejich systému spočívá v detekci pohybu na bázi prostého výpočtu rozdílových snímků. Rozdílové snímky jsou dále upravovány morfologickými operacemi a následně prahovány. V sekvenci takto upravených rozdílových snímků jsou nalezeny vazby pro zjištění trajektorie pohybujících se objektů. Velikost pohybujícího se objektu a charakter jeho pohybu předurčuje, zda se jedná o auto, či nikoliv. Systém je schopen poskytovat pouze údaje o počtu automobilů na celém parkovišti a při inicializaci potřebuje informovat o počátečním množství aut.

Představená detekce pohybu není úsporná z hlediska výpočetních nároků, neboť je nutné zpracovávat velké množství redundantních dat. Systém vyžaduje neustálé kontinuální získávání snímků s vysokou frekvencí, což nedává prostor pro efektivní snížení spotřeby energie (viz kapitola 3.1). Systémy využívající časových charakteristik se moc nevyskytují, i když jejich přesnost může být relativně vysoká, jak ukazují novější práce [22] a [23]. Všechny další přístupy, které zde budou uvedeny, neuvažují časovou vazbu mezi snímky a každý snímek řeší individuálně.

(Hsu, Tanaka, Sugie, Ueda, 2002) [21]
V další relativně starší publikaci si autoři, jako rozlišující vlastnost obsazenosti, zvolili hrany. Systém pracuje s šedotónovými snímky, které jsou zprvu předzpracovány logaritmickou převodní charakteristikou. V dalším kroku jsou nalezeny hrany pomocí Laplaceova operátoru a prahování. Rozhodnutí o obsazenosti je určeno z poměrů celkového množství hran v definovaných oblastech. Autoři udávají přesnost přes 95 % pro jednu testovací sekvenci parkoviště s 53 parkovacími místy.

Zvolené řešení je méně výpočetně náročné než [17] a zároveň poskytuje informace o obsazenosti jednotlivých míst. Při generalizaci podmínek testování se ale dá očekávat razantní snížení úspěšnosti vzhledem k nízké robustnosti klasifikačního mechanizmu.

(Delibaltov, Wu, Loce, Bernal, 2013) [19]
V tomto článku autoři modelují každé parkovací místo jako určitý objem v 3D prostoru, čímž se snaží eliminovat problémy s výraznými překryvy aut, které jsou způsobeny jejich snímáním pod malým vertikálním úhlem (kamera A na Obr. 5). Parkovací místa jsou poloautomaticky vyznačena a následně modelována hustotou pravděpodobnosti toho, že určitý pixel je součástí parkovacího místa. Pro detekci je použit rotačně invariantní LBP deskriptor se SVM klasifikátorem s nelineárním jádrem. LBP-SVM klasifikátor pracuje po malých blocích, které mohou být klasifikovány jako jedna ze šesti tříd: auta, země, stromy, nebe, budovy a zbytek. Finální výsledek o obsazenosti parkovacího místa je získán váženou sumou výsledků blokové klasifikace, kde váhy jsou dány modely parkovacích míst.

(De Almeida, Oliveira, Britto, Silva, 2015) [4]
Článek se z velké části zabývá testováním robustnosti klasifikace. Kromě vytvoření mohutné databáze PKLot, ověřují autoři použitelnost deskriptorů textury založených opět na LBP a dále na LPQ s využitím klasifikátoru SVM. Z evaluace klasifikace autorům vyplynula přesnost 99 %, když byl algoritmus testován na stejném pohledu parkoviště jako byl trénován. V případě, že byl pro trénování zvolen jiný pohled parkoviště dosáhli přesnosti okolo 89 %.

Je nutné poznamenat, že databáze PKLot obsahuje téměř 700 000 výřezů parkovacích míst pořízených v různých světelných a klimatických podmínkách. Hlubší pohled na tuto databázi je uveden v kapitole 7.3, kde je využita pro evaluaci implementovaného systému.

Další vybrané deskriptory

Pro problémy spjaté s detekcí automobilů v obraze se lze dále setkat se systémy pracujícími s celou řadou dalších deskriptorů. Níže je uveden výpis několika dalších možných deskriptorů s odkazy na příslušnou literaturu:

- vlnková transformace [24];
- SIFT deskriptor [18];
- HOG deskriptor [25][26];
- Hue histogram deskriptor [27].

(Amato, Carrara, Falchi, Gennaro, Meghini, 2017) [12]

Tento systém je v současnosti pravděpodobně nejmodernější a nejspolehlivější, co se generalizace týče. Jádro systému je založeno na hlubokých konvolučních neuronových sítích, které přímo vyhodnocují výřezy parkovacích míst. Při předzpracování snímků není využit žádný deskriptor a příprava spočívá pouze v provedení čtvercového výřezu oblasti parkovacího místa a změně rozlišení na 244 × 244 pixelů.

Základem pro úspěch tohoto systému je natrénování neuronové sítě na známých datech. Toto trénování je velmi výpočetně náročné a může zabrat dlouhou dobu, ale samotná klasifikace již výpočetně náročná není. Pro evaluaci svého systému využili autoři jednak již zmíněnou databázi PKLot a také své databáze[9]. Jejich databáze sice obsahují méně snímků, ale představují náročnější situace s mnoha překryvy a s více pohledy kamer.

Navržený systém je schopný správně klasifikovat obsazenost parkovacích míst s přesností vyšší než 90 % ve velmi širokém spektru podmínek. Za svou vynikající schopnost generalizace vděčí systém především poznatkům z teorie oblasti *deep learning*. V době psaní této práce bylo možné sledovat[10] funkci systému v reálném čase prostřednictvím webového prohlížeče. Problematikou využití konvolučních neuronových sítí pro rozpoznávání obsazenosti se dále například zabývá [20].

[9] *CNR park-datasets* [online], [cit. 2017-05-10]. Dostupné z: http://claudiotest.isti.cnr.it/park-datasets/

[10] *Raspberry Smart Cameras Live View: Visual monitoring of parking lot occupancy status Research area of CNR – Pisa* [online], [cit. 2017-05-09]. Dostupné z: http://claudiotest.isti.cnr.it/telecamere.html

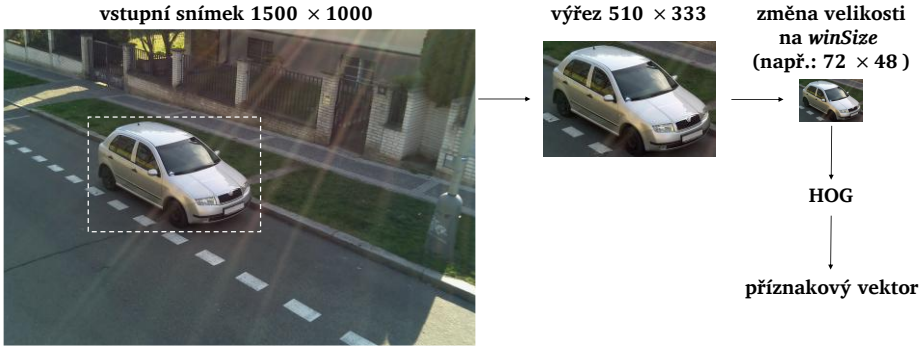
4.3 Nástroje použité v implementaci

Velmi důležitými bloky v implementaci jsou deskriptor HOG a klasifikátor SVM. Využitím těchto bloků v implementovaných algoritmech se zabývá kapitola 6, zde jsou tyto nástroje probrány jako samostatné funkční bloky.

4.3.1 Deskriptor HOG

HOG – Histogram orientovaných gradientů je deskriptor používaný pro extrakci příznaků neboli význačných vzorů z obrazových dat. Tento deskriptor zaznamenal velký úspěch především na poli detekce lidí [40] a od té doby začal být používán pro detekci celé řady dalších objektů. HOG funguje na principu modelování lokálních vzorů textury pomocí výpočtu blokově normalizovaných distribucí směrů a sil hran v obraze [41]. Tato kapitola vychází především z [42] a [40].

Deskriptor na svém vstupu vyžaduje snímky konstantní velikosti (*winSize*) a obvykle je tedy potřeba provést výřez, popřípadě změnu velikosti snímku (Obr. 11).



Obr. 11: Předzpracování pro HOG deskriptor

Pro získání **HOG příznakového vektoru** je prvním krokem výpočet gradientů snímku ve směru x a ve směru y. Gradienty g_x a g_y, lze získat filtrací vstupního snímku f pomocí diskrétní konvoluce s jádry h_x a h_y, podle vztahů uvedených níže.

výpočet gradientů ve směru x

$$g_x(x,\, y) = \sum_{s=-1}^{1} f(x+s,\, y)\, \cdot h_x(s)$$

$$h_x = (-1, 0, 1)$$

výpočet gradientů ve směru y

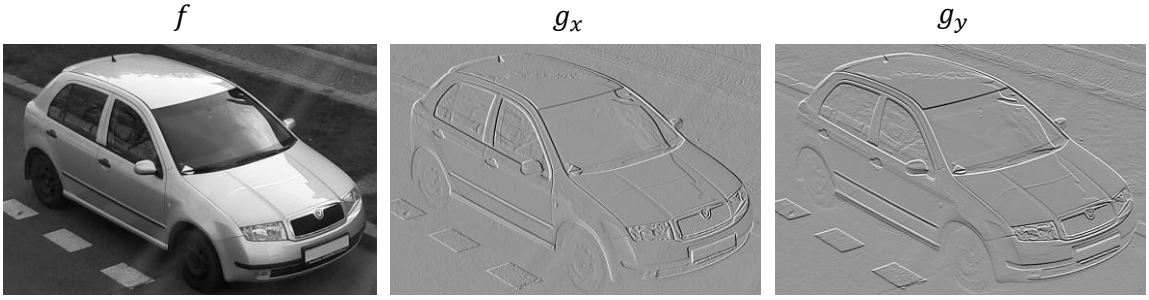
$$g_y(x,\, y) = \sum_{t=-1}^{1} f(x,\, y+t)\, \cdot h_y(t)$$

$$h_y = (-1, 0, 1)^T$$

$$x \in \{2,\, 3,\, 4\, \ldots\, M-1\},\; y \in \{2,\, 3,\, 4\, \ldots\, N-1\} \quad \text{(1 pixel od krajů)}$$

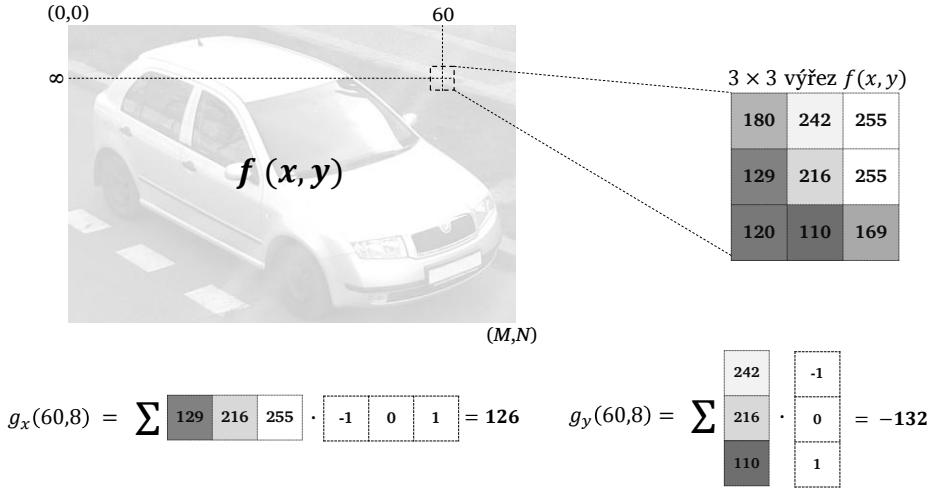
$f(x,\, y)$	vstupní snímek velikosti $M \times N$ (např.: 72×48)
$g_x(x,\, y), g_y(x,\, y)$	snímky gradientů ve směru x, y
$h_x(s), h_y(t)$	konvoluční jádro pro získání diference ve směru x, y

Gradienty ve směru x popisují horizontální změny průběhu intenzity a gradienty ve směru y popisují vertikální změny průběhu intenzity (Obr. 12). Gradient je kladný, když intenzita s prostorovou souřadnicí narůstá (světlé oblasti v ilustračních snímcích níže), nulový při konstantním průběhu (šedá) a záporný, v případě, že intenzita klesá (tmavé oblasti). Za pomoci gradientů je možné zachytit hrany v obraze. Při výpočtu gradientů snímku s kanály RGB se gradienty počítají pro každý kanál zvlášť.



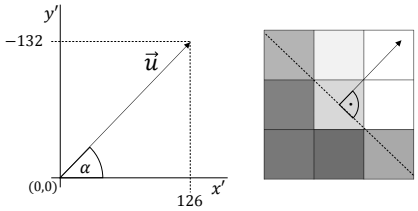
Obr. 12: Vstupní snímek a snímky gradientů

Pro lepší ilustraci je zde ještě uvedena ukázka (Obr. 13) výpočtu gradientu pro jeden vybraný pixel v obraze ($x = 60, y = 8$).



Obr. 13: Výpočet gradientu pro jeden vybraný pixel v obraze

Vypočtené hodnoty gradientu jsou souřadnicemi vektoru $\vec{u}$, který popisuje směr nárůstu intenzity (Obr. 14). Vektor $\vec{u}$ má svou velikost $|\vec{u}|$ a směr α, které jsou dány základními pravidly geometrie. Normála k vektoru $\vec{u}$ koresponduje s hranou v obraze.

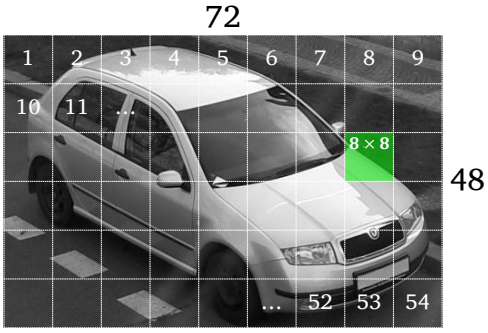


Obr. 14: Znázornění gradientu pro jeden vybraný pixel[11]

[11] Je konvencí, že osa *y* bývá v souvislosti se souřadnicemi snímků převrácená

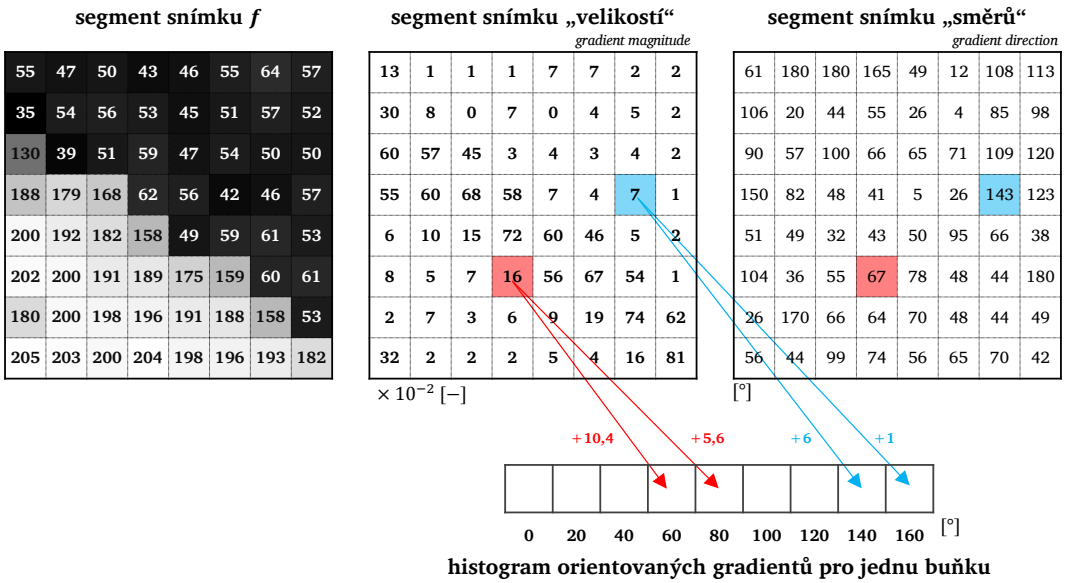
Ze snímků g_x a g_y lze efektivně vypočítat polární souřadnice (velikost, směr) vektoru $\vec{u}$ pro každý bod v obraze a získat tím snímek „velikostí“ a snímek „směrů“. Snímek „velikostí“ popisuje síly hran a snímek „směrů“ jejich úhly. V případě uvažování orientace vektorů mohou být úhly v rozsahu -180° až 180°. V HOG se však orientace často neuvažuje a úhly se potom pohybují pouze v rozsahu 0° až 180°.

Elementárním prvkem HOG je buňka (*cell*), jejíž velikost je potřeba na začátku definovat (typicky 8×8). Buňky se získají segmentací ze snímku „velikostí“ a snímku „směrů“. Pro názornější ilustraci je segmentace demonstrována na vstupním snímku (Obr. 15).



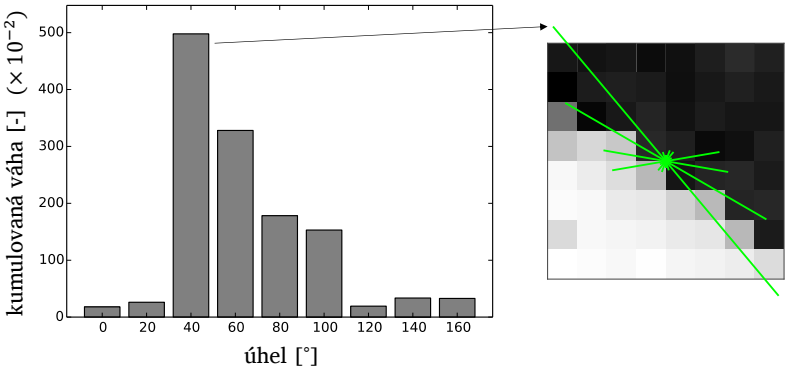
Obr. 15: Buňky pro HOG deskriptor

Pro okno o rozlišení 72×48 a velikosti buňky 8×8 je získáno celkem 54 segmentů „velikostí“ a 54 segmentů „směrů“. Z každé dvojice těchto segmentů je získán histogram orientovaných gradientů s předem stanoveným počtem sloupců (typicky 9). V případě 9 sloupců a neuvažování orientace vektorů, odpovídají sloupce uhlům 0° až 180° po 20°. Hodnota 180° je rovna hodnotě 0°, kvůli charakteru polárních souřadnic. Histogram je získán kumulací vah (snímek „velikostí“) do příslušných sloupců. Získávání histogramu je dále demonstrováno na jedné vyznačené buňce z Obr. 15.



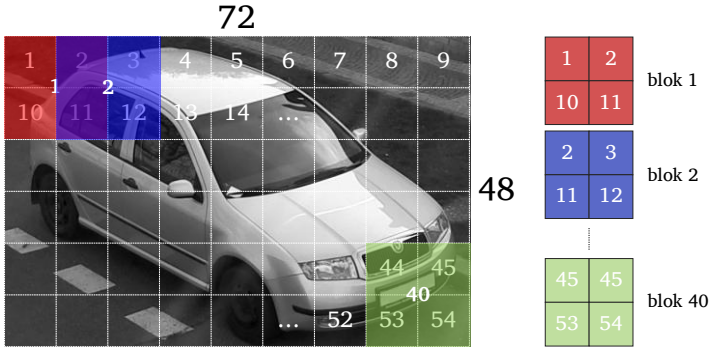
Obr. 16: Získání histogramu pro jednu buňku

Po kumulaci všech vah v buňce (Obr. 16) je získán 9 prvkový vektor, popisující distribuci vlastností gradientů v jedné buňce. Pro uváděný příklad mají největší kumulovanou váhu úhly 40° a 60° (normála 130° a 150°), což dobře koresponduje s jasně viditelnou hranou v obraze. Na Obr. 17 je vynesen histogram pro zkoumanou buňku a také jeho alternativní vizualizace pomocí různě dlouhých natočených čar. Úhel čar vychází z hodnot na ose x upravených o 90° a délka čar odpovídá hodnotám na ose y.



Obr. 17: Vizualizace histogramu z jedné buňky

Výpočtem histogramu pro každou buňku se získá celkem 486 hodnot (54 buněk × 9 sloupců histogramu), které již lze použít jako příznakový vektor. Je však výhodnější provést ještě tzv. blokovou normalizaci, která umožní, aby byly příznaky invariatní vůči různých světelným změnám. Princip blokové normalizace spočívá v konkatenaci výsledků z několika sousedících buněk (typicky 4 buňky) do vektoru a následné úpravě jeho složek pro dosažení jednotkové délky. Normalizace je obvykle realizovana s překryvem 50 %, jak naznačuje Obr. 18.



Obr. 18: Bloková struktura HOG (typicky 1 blok složen ze 4 buněk)

Konečná dimenze příznakového vektoru (počet prvků) pro uváděný příklad je 1440 (40 bloků, kde každý blok obsahuje 36 normalizovaných hodnot histogramu). Celý výsledný vektor lze vizualizovat různými zajímavými způsoby[12], což však není obsahem této práce. Finální příznakový vektor HOG představuje úspornější reprezentaci vstupního snímku, která zachovává podstatné informace o tvarech a hranách v obraze. Další zpracování tohoto příznakového vektoru obvykle spočívá v jeho předání klasifikátoru.

[12] HOGhles: Visualizing Object detection Features: http://web.mit.edu/vondrick/ihog/

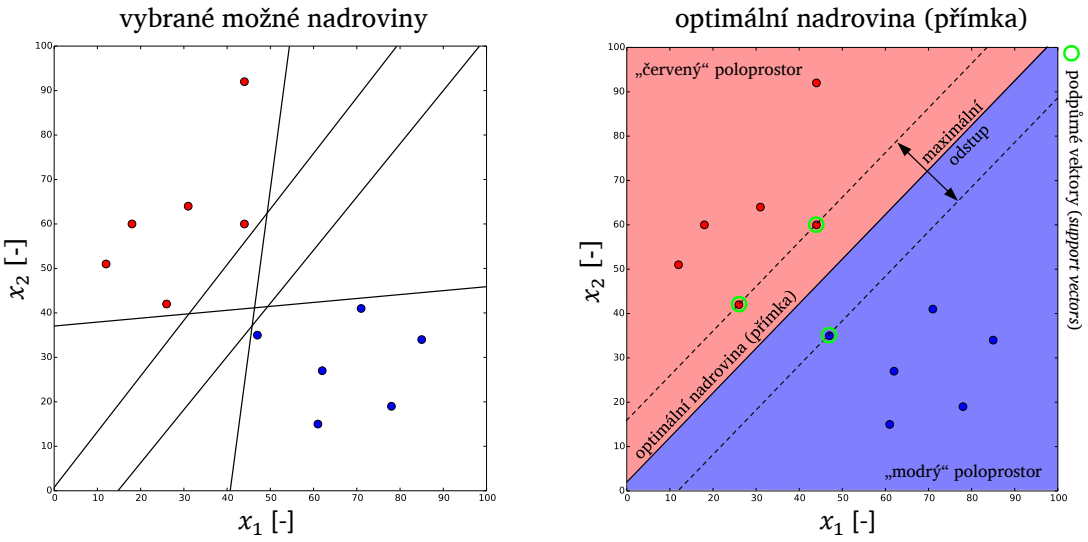
4.3.2 Klasifikátor SVM

SVM – Support vector machine je matematickým modelem z oblasti tzv. strojového učení s učitelem. V této práci je model využíván pro klasifikaci dat, která jsou reprezentována příznakovými vektory získanými pomocí HOG deskriptoru. Na poli detekce obsazenosti parkovacích míst je SVM nejpoužívanějším klasifikátorem [4]. Základní princip SVM spočívá v rozdělení prostoru příznakových vektorů dimenze n na dva poloprostory, a to na základě označených trénovacích dat. Jako rozhodovací hranice je použita nadrovina, což je v podstatě libovolná lineární funkce v n-rozměrném prostoru. Klasifikace neznámého vzorku dat je výpočetně nenáročná a jde o rozhodnutí určující, kterému ze dvou poloprostorů vektor náleží. Tato kapitola vychází především z [43] a [44].

Hlavním úkolem SVM je nalézt optimální nadrovinu, která má co největší odstup (*margin*) od vzorků trénovacích dat. Koncept SVM lze dobře demonstrovat ve 2D, kde nadrovině odpovídají přímky. Nechť jsou trénovacími daty příznakové vektory $\vec{x}$ dimenze dva. K dispozici je celkem 12 vektorů dvou různých tříd (červená a modrá). Po vynesení vektorů do grafu lze celkem intuitivně předpokládat jakými různými přímkami lze od sebe data oddělit. Na grafu vlevo na Obr. 19 jsou znázorněny vybrané možné nadroviny (přímky), které od sebe jednotlivé třídy trénovacích dat perfektně oddělují. Žádná z přímek však není optimální, neboť není dosaženo kritéria maximálního odstupu. Graf vpravo znázorňuje přímku, která kritérium splňuje a rozděluje prostor optimálně. Vektorům na čárkovaně vyznačené hranici se říká podpůrné (*support vectors*).

x_1	47	61	71	78	85	62	18	26	31	44	44	12
x_2	35	15	41	19	34	27	60	42	64	60	92	51
třída	●	●	●	●	●	●	●	●	●	●	●	●

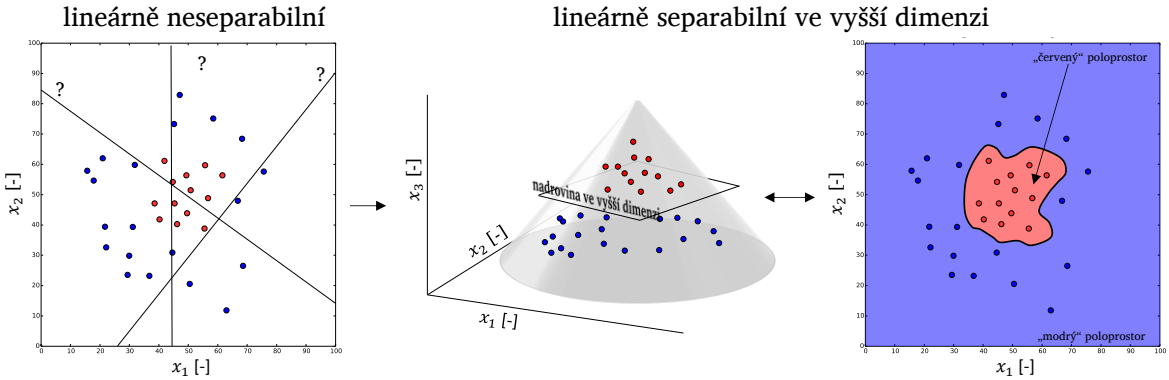
Tab. 3: Zadání demonstračního příkladu (trénovací data)



Obr. 19: Určení nadroviny, demonstrační příklad

V tomto demonstračním případě bylo možné od sebe prvky jednotlivých tříd oddělit přímkami (lineárně separabilní data). To však nemusí být vždy možné, kvůli rozložení dat v prostoru (lineárně neseparabilní data).

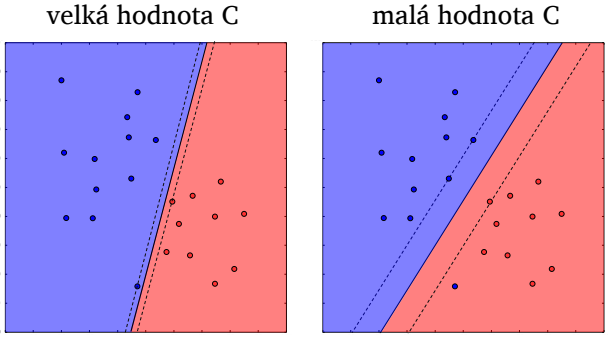
Tento problém obchází SVM mapováním příznakových vektorů do vyšší dimenze. Zvýšení dimenze dokáže pomoci vytvořit z původně lineárně neseparabilních dat data lineárně separabilní ve vyšší dimenzi. Data ve vyšší dimenzi jde opět oddělit nadrovinou. Myšlenku zvyšování dimenze příznakových vektorů znázorňuje snímek níže (Obr. 20).



Obr. 20: SVM lineární separace ve vyšší dimenzi

Požadavkem pro nalezení optimální nadroviny ve vyšší dimenzi je opět maximalizace odstupu nadroviny od trénovacích dat. Výpočty potřebné k nalezení správné rozhodovací hranice jsou realizovány za pomoci tzv. jádrových funkcí, které mapují data do vyšší dimenze, a přitom umožňují výpočty provádět v původním prostoru.

Jedním z důležitých parametrů SVM je hodnota C. Touto hodnotou lze nastavit, jak velký je požadavek na to, aby se klasifikátor nedopouštěl chyb na množině trénovacích dat. Velká hodnota C znamená, že se množina trénovacích dat rozdělí, s nulovým, či minimálním počtem chyb. To ale nemusí vždy být optimální řešení, neboť se v trénovacích datech mohou vyskytovat vektory, které danou třídu popisují špatně. Malá hodnota C umožní najít nadrovinu s větším odstupem, za cenu chyb na trénovací množině (Obr. 21).



Obr. 21: Vliv parametru C na hledání nadroviny

5. Návrh kamerového systému

Po uvážení aspektů návrhu systému, bylo potřeba zvolit vhodné technologie pro realizaci kamerových modulů, které budou schopny obraz snímat, zpracovávat a zpracované informace odesílat. Tato kapitola se zabývá sestavením kamerových modulů a rozborem použitých komponent. Jako ústřední element modulů byly testovány různé verze počítače Raspberry Pi (B+, 3, Zero a Zero W).

Výhody použití Raspberry Pi jako výpočetního elementu inteligentní kamery, oproti využití hardwaru se specifickým účelem, spočívají především v rychlosti návrhu aplikací prostřednictvím vyšších programovacích jazyků v Linuxovém systému. Dále má počítač relativně příznivou velikost, cenu, napájecí nároky a dobré možnosti připojení. V odborných zdrojích se lze s využitím Raspberry Pi pro podobné účely setkat například v [11][12][28].

5.1. Raspberry Pi

Raspberry Pi (dále jen RPi) je souhrnný název pro populární počítače z rodiny malých jednodeskových počítačů vyvíjených ve Velké Británii neziskovou organizací Raspberry Pi Foundation. První generace počítače se poprvé dostala na trh v roce 2012. Za vznikem celého projektu tehdy stála především myšlenka podpory vzdělávání dětí v oblasti informatiky v Anglii [29].

RPi 3 RPi Zero

Obr. 22: Vybrané verze Raspberry Pi počítače[13]

V souvislosti s touto prací je především zajímavé, že počítače RPi nabízejí poměrně dobrý multimediální výkon díky vhodnému zužitkování třídy procesorů, které jsou obvykle užívány ve světě set-top boxů, video přehrávačů, či chytrých mobilních telefonů. Zmíněná

[13] Zdroj obrázku: *Pimoroni* [online], [cit. 2017-05-10]. Dostupné z: https://shop.pimoroni.com/

zařízení mají schopnosti relativně úsporně provádět náročné operace s obrazem, jako například velmi rychlé kódování videí ve vysokém rozlišení. [29]

Součástí každého RPi je SoC multimediální procesor, který v jednom malém pouzdře integruje velkou část systémových komponent, včetně CPU, GPU a paměti. Výsledný procesor založený na architektuře ARM nabízí solidní výkon při malé spotřebě energie především při zpracování multimédií prostřednictvím GPU VideoCoreIV [29][30][31].

5.1.1 Specifikace

Existuje několik variant RPi, mezi nimiž jsou rozdíly zejména ve výpočetním výkonu, v rozvržení jednotlivých částí na PCB, v počtu různých rozhraní a v dalších specifikách. Základní srovnání hlavních verzí RPi nastiňuje Tab. 4. Při návrhu nových verzí se vývojáři snaží, aby byly RPi zpětně kompatibilní i třeba co se týče rozměrů. [30][31].

Verze RPi	**RPi A+**	**RPi B+**	**RPi 2**	**RPi 3**	**RPi Zero**
Generace	**1**		**2**	**3**	**-**
Snímek					
Vydáno	11/2014	11/2014	10/2015	02/2016	03/2016
~ Cena	£20	£27	£32	£32	£4
SoC	Broadcom BCM2835		Broadcom BCM2836	Broadcom BCM2837	Broadcom BCM2835
CPU	700 MHz 32-bit single-core ARM1176JZF-S		900 MHz 64-bit quad-core ARM Cortex-A53	1,2 GHz 64-bit quad-core ARM Cortex-A53	1 GHz 32 bit single-core ARM1176JZF-S
GPU	Broadcom VideoCore IV OpenGL ES 2,0, OpenVG 1080p30 H. 264 vysoký profil kódování/dekódování				
	250 MHz			400 MHz OpenVG 1080p60	250 MHz
SDRAM	256 MiB	512 MiB	1024 MiB		512 MiB
Napájení IDLE/LOAD [mA]	5 V (DC), MicroUSB typ B nebo GPIO				
	110/170	250/310	260/420	310/580	100/250
Rozměr [mm]	65×56×12	85×56×17	85×56×17	86×57×17	65×30×5
Váha [g]	23	40	40	45	9

Tab. 4: Základní srovnání různých verzí Raspberry Pi [30][31][32][33]

Společnými vlastnostmi jednotlivých verzí (mimo své poslání šířit informatiku) jsou například hardwarová rozhraní jako USB, HDMI, MicroSD slot, výstupní analogové audio/video rozhraní, CSI pro připojení speciálních kamer, DSI pro připojení displeje a GPIO piny, přes které lze realizovat komunikaci I2C, SPI, či UART. Popřípadě se dají piny využít k jiné libovolné interakci s vnějším světem nastavováním či čtením jejich logických úrovní [30][31][32]. Z pohledu této práce je klíčové především rozhraní CSI pro připojení kamer, které je stručně probráno v 5.1.3.

Další společnou vlastností pro všechny verze RPi je možnost výběru operačního systému. Doporučeným operačním systémem pro RPi a systémem, který je používán na zařízeních v této práci je speciální Linuxová distribuce Raspbian[14], která vyšla z distribuce Debian optimalizací pro hardware RPi. Kromě systémů založených na Linuxu je možné využít například Windows 10 IoT Core, nebo třeba RISC OS [30].

5.1.2 Alternativy

Počítače z rodiny RPi nejsou na dnešním trhu jedinou variantou zařízení podobného typu. Existuje mnoho dalších podobných výrobků[15], konkrétně jde například o:

- Banana Pi;
- Orange Pi;
- BeagleBone;
- Nano PC;
- Cubieboard;
- Dragonboard;
- Udoo;
- FireFly;
- PandaBoard;
- LattePanda;
- Intel Edison;
- ASUS Tinker Board

Některé z počítačů, stejně jako RPi, nabízejí standardizované CSI rozhraní pro kamery. Žádný z výrobků však nemá pravděpodobně tak širokou uživatelskou základnu a dostupnou podporu jako RPi.

5.1.3 Snímací zařízení

Rozhraní

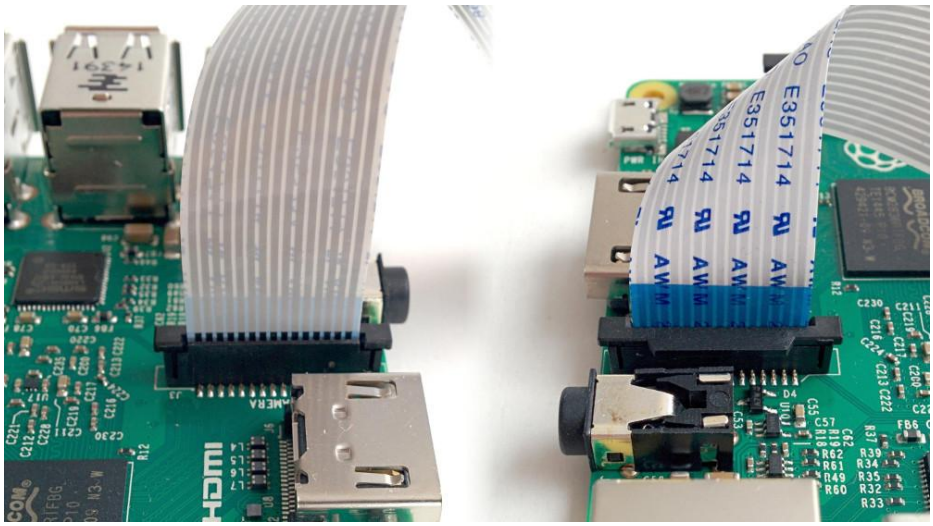
Při letmém pohledu na RPi by se jako snímací zařízení mohla nabízet například USB 2.0 webkamera. Jde o častý a přímočarý postup využití RPi v oblastech snímání obrazu, nejedná se však vždy o postup nejvhodnější. Praktická propustnost USB 2.0 je na RPi relativně nízká (pro nekódovaná obrazová data) a u USB webkamer lze tedy očekávat

[14] Konkrétně Raspbian Jessie Lite (datum uvedení: 10.4.2017)

[15] *Comparison of single-board computers* [online], [cit. 2017-02-11]. Dostupné z: https://en.wikipedia.org/wiki/Comparison_of_single-board_computers

data nějakým způsobem redukovaná. Daleko vhodnějším snímacím zařízením pro RPi jsou speciální kamery s rozhraním CSI. Podobně jako SoC, tak i CSI je ze světa mobilních technologií.

Na své desce nabízí RPi sériové rozhraní CSI-2, které vychází ze standardu vytvořeného organizací MIPI. CSI je specifikací, která definuje přímé spojení mezi kamerou a procesorem. Konkrétně na RPi je rozhraní fyzicky realizováno 15 pinovým ZIF konektorem (Obr. 23), do kterého se kamery připojují pomocí plochého kabelu FFC [31][34][35].



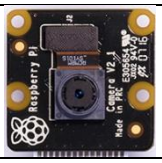
Obr. 23: ZIF konektor k připojení kamery pomocí CSI-2 rozhraní FFC kabelem[16]

MIPI CSI-2 je dnes velmi oblíbeným rozhraním pro kamery v chytrých mobilních telefonech díky pečlivému designu, nízkým výkonovým nárokům a propustnosti [36]. Díky tomuto rozhraní je možné, s minimální spotřebou, přímo do grafické karty RPi přenášet nekomprimovaný videosignál ve vysokém rozlišení i s vysokým snímkovým kmitočtem.

Raspberry Pi kamery

K RPi je možné prostřednictvím rozhraní CSI-2 připojit originální kamerové moduly, které se objevily na trhu přibližně rok po vydání samotného RPi. Moduly užívají CMOS snímače, které lze najít v některých chytrých mobilních telefonech. Zatím vyšly celkem dvě generace kamer s tím, že v každé generaci existují dvě varianty kamery (s a bez IR filtru před snímačem) [30][34]. Základní srovnání parametrů jednotlivých verzí nabízí Tab. 5.

[16] Zdroj obrázku: *Raspberry Pi Learning Resources* [online], [cit. 2017-05-10]. Dostupné z: https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/

Verze kamery	v1	v1 NoIR	v2	v2 NoIR
Snímek				
Vydáno	05/2013	10/2013	04/2016	04/2016
~ Cena	£20	£20	£28	£28
Snímač	Omnivision CMOS – OV5647		Sony CMOS – IMX219PQ	
Formát snímače	3,67 x 2,74 mm (~ ¼“), Crop factor ≅ 10x			
Rozlišení	2592 x 1944 (5 Mpx)		3280 x 2464 (8 Mpx)	
Velikost pixelu	1,4 x 1,4 μm		1,12 x 1,12 μm	
Objektiv	Závit M6x0.35 f = 3,6 mm, N = f/2,9		Závit M6x0.35 f = 3 mm, N = f/2	
FOV [˚]	54 x 41		62 x 49	
Závěrka	Rolling shutter t max = 6 s		Rolling shutter t max = 10 s	
IR-CUT filtr	ANO	NE	ANO	NE
Rozměry [mm]	~ 25 x 24 x 10			
Váha [g]	3			

Tab. 5: Základní srovnání oficiálních RPi kamer [30][34][37]

Výhodou využití těchto kamer je to, že uživatel má možnost snadno nastavit mnoho parametrů snímání. Samozřejmostí jsou automatické režimy měření expozice a scénické režimy typu „sport“, „noc“, „pláž“ atp., které slouží pro rychlé nastavení bez nutnosti hlubšího zkoumání problematiky snímání. Moduly však umožnují i celou řadu manuálních nastavení, jako například: nastavení expoziční doby, citlivosti, vyvážení bílé, režimu měření či parametru kompenzace expozice [30].

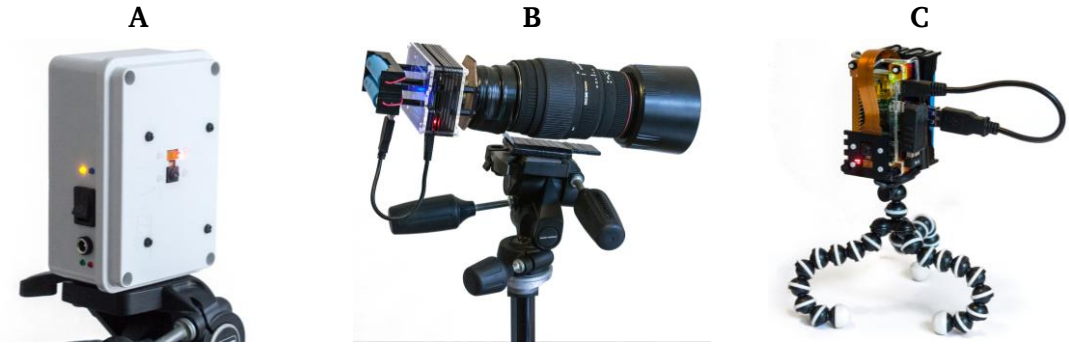
Ke kameře lze přistupovat buď přes tzv. *video port*, nebo *still port*, podle toho, zda je žádoucí snímat statické snímky, nebo video. Pro snímání statických snímků je obvykle využita celá plocha snímače a v případě požadavků na jiné, než plné rozlišení je snímek podvzorkován nebo je vybrána pouze jeho část [38]. Pro snímání videa existuje několik diskrétních módů, kterými se však nemá v této práci smysl hlouběji zabývat[17], protože implementované algoritmy pracují pouze se snímky získávanými přes *still port*.

[17] Další informace k módům a portům snímání uvádí tvůrce knihovny Picamera zde: *Camera Hardware* [online], [cit. 2017-05-10]. Dostupné z: http://picamera.readthedocs.io/en/release-1.12/fov.html

Podobně jako u mobilních telefonů, tak i u RPi kamer je objektiv pravděpodobně nejslabším článkem celého systému. Přestože má snímač rozlišení 5 Mpx (v1) / 8 Mpx (v2), dá se očekávat, že reálné rozlišení bude z důvodu slabého objektivu v kombinaci s malými pixely na snímači zhruba čtvrtinové (viz kapitola 3.2) a [39].

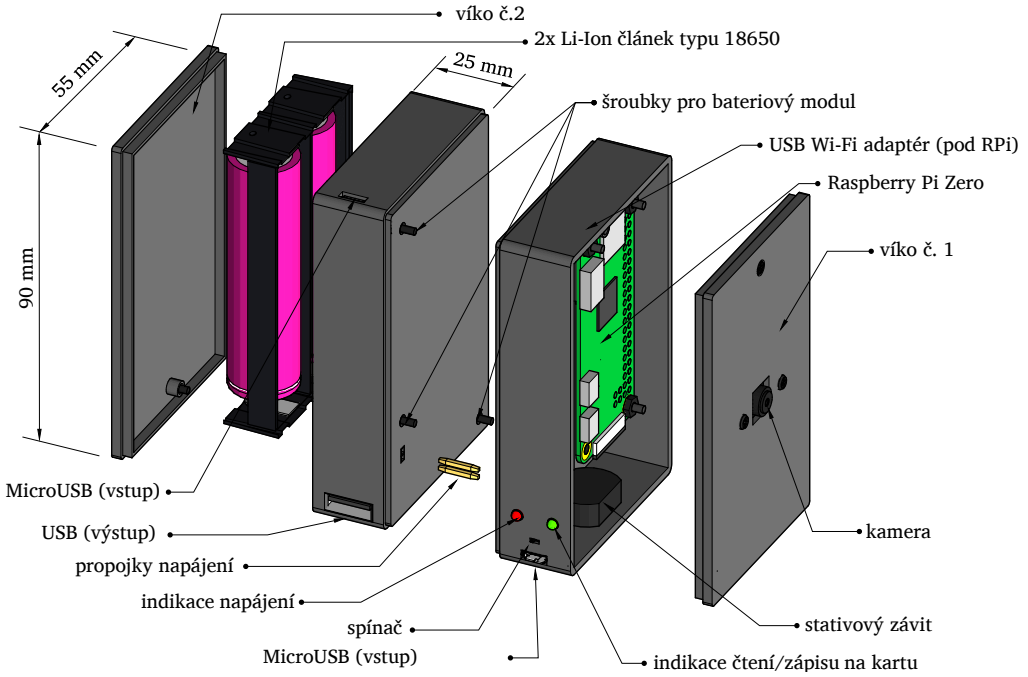
5.2 Sestavení kamerových modulů

V souvislosti s touto prací bylo sestaveno několik různých bezdrátových prototypů kamerových modulů na bázi RPi. Pro usnadnění návrhu finálního systému bylo zvoleno bateriové napájení, které umožnilo volný pohyb v prostoru. Moduly první generace sloužily pouze pro prověření možností využití RPi v diskutovaných systémech (Obr. 24).



Obr. 24: První generace kamerových modulů

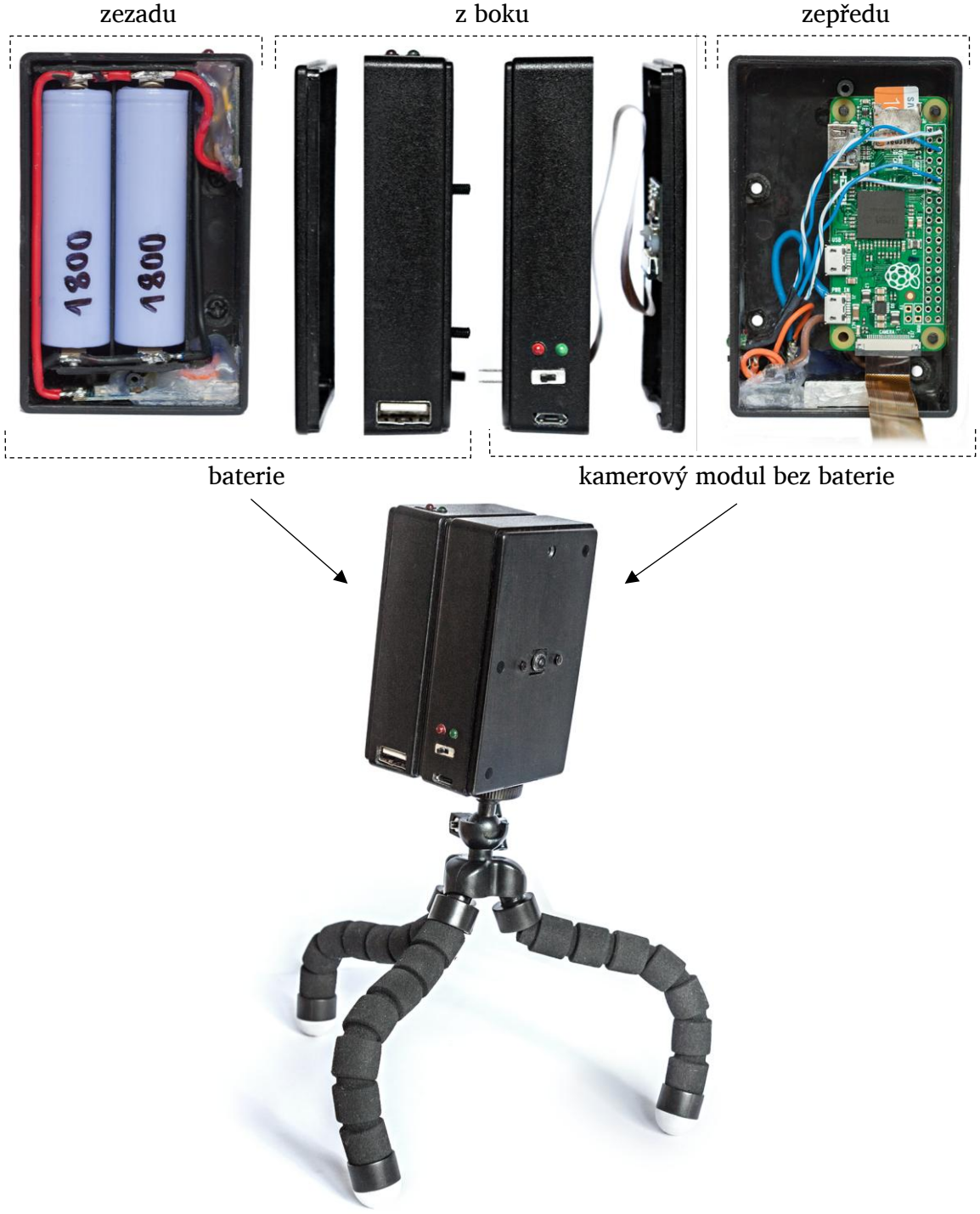
Prototypy **A** a **B** využívají RPi plné velikosti (testováno s RPi B+ a RPi 3) a prototyp **C** pracuje s RPi Zero. Po srovnání parametrů (Tab. 4) a testech rychlosti několika vybraných algoritmů zpracování obrazu bylo pro realizaci finálních modulů zvoleno RPi Zero. Návrh kamerového modulu druhé generace je znázorněn na Obr. 25.



Obr. 25: Návrh druhé generace kamerového modulu

5.1.2 Finální funkční prototyp

Celkem jsem sestavil tři stejné funkční prototypy kamerového modulu, založené na RPi Zero (Obr. 26). Prototypy především kombinují samotné RPi, kamerový modul v1, USB Wi-Fi adaptér 802.11 b/g/n připájený zezadu k desce, standardní stativový závit s průměrem 1/4", dva Li-Ion články 3,7 V typu 18650 a DC-DC step-up měnič 5 V/2 A. Baterii jednoho ze sestavených modulů lze nabíjet přímo, u ostatních modulů je potřeba články vyjmout a nabít je externě.



Obr. 26: Druhá generace kamerového modulu

Při výběru komponent bylo jednou z hlavních motivací udržet co nejnižší pořizovací cenu za celý systém. Jeden kompletní kamerový modul, včetně malého stativu (Obr. 26) vyšel přibližně na 800 Kč, je však nutné uvést, že v ceně nejsou zahrnuty Li-Ion články, neboť ty byly získány zdarma, recyklací starých notebookových akumulátorů.

Spotřeba kamerového modulu v zátěži se pohybuje okolo 1,5 W a dva články s celkovou kapacitou přibližně 13 Wh ($2 \cdot 3{,}7\ \mathrm{V} \cdot 1\,800\ \mathrm{mAh}$) tedy dovolují zhruba 9 hodin provozu. V rámci práce nebyly realizovány žádné kroky k prodloužení této doby, neboť primárním zájmem byla realizace algoritmů pro rozpoznávání obsazenosti parkovacích míst. Jednou z možností pro prodloužení výdrže baterií je aplikace spánkových režimů (viz kapitola 3.1), což by však vyžadovalo další obvody, které by sloužily pro spínání napájení (RPi v základu spacími režimy nedisponuje) [11].

5.2.1 Připojení a získávání obrazových dat

Bezdrátová komunikace s modulem je realizována prostřednictvím Wi-Fi standardů 802.11 b/g/n, levným USB přijímačem/vysílačem. Pro konfiguraci a ovládání je převážně využíván protokol SSH. Moduly se dokáží připojit k *hotspotu* chytrého telefonu a být jím pak ovládány v rámci lokální sítě například prostřednictvím aplikace JuiceSSH[18].

Volba USB Wi-Fi adaptéru není optimální z hlediska spotřeby. V [11] byla změřena spotřeba RPi s a bez USB Wi-Fi adaptéru a naměřený rozdíl činil až 0,8 W. Jako úspornější náhradu za Wi-Fi navrhují autoři téže publikace technologii Zigbee[19], jejíž napájecí nároky jsou mnohem nižší, ale zároveň je se potřeba spokojit s výrazně nižšími přenosovými rychlostmi. Nižší přenosové rychlosti by nebyly problém pro systém nasazený v reálném provozu, který přes síť zasílá opravdu pouze informace o obsazenosti. Při vývoji algoritmů bylo však potřeba čas od času přenášet i obrazová data, což ospravedlňuje volbu zmíněných Wi-Fi standardů.

Obrazová data z kamer připojených pomocí CSI, lze získávat různými prostředky. Základním způsobem je například systémový balík *raspicam*[20], sloužící pro ovládání kamery z příkazové řádky. Pro získávání obrazových dat a jejich streamování je velmi užitečným nástrojem rozhraní *RPi-Cam-Web-Interface*[21], určené pro komunikaci s moduly prostřednictvím sítě libovolným internetovým prohlížečem.

[18] JuiceSSH aplikace: https://juicessh.com/

[19] Zigbee aliance: http://www.zigbee.org/

[20] Raspicam dokumentace: https://www.raspberrypi.org/documentation/raspbian/applications/camera.md

[21] RPi-Cam-Web-Interface: https://github.com/silvanmelchior/RPi_Cam_Web_Interface

Pro systémy, kterými se tato práce zabývá, není bezpodmínečně nutné obrazová data ukládat, či streamovat. Primárním požadavkem je co nejefektivnější transformace obrazových dat na obsazenost parkoviště. Pro tento účel dobře poslouží Python knihovna *Picamera*[22], která nabízí snímání dat v různých formátech do datových struktur, které jsou k dispozici při běhu programu. Tím dává knihovna prostor k okamžitému zpracování dat bez nutnosti jejich zapisování na paměťovou kartu. Mezi použitelné struktury patří například:

- io.BytesIO (pole bajtů, ke kterému je možné se chovat jako k souboru);
- socket (Socket – Python modul k realizaci síťového rozhraní);
- PIL.Image[23] (knihovna pro manipulaci s obrazem);
- numpy.array (knihovna pro práci s *n*-dimenzionálními objekty (primárně));
- Vlastní objekt splňující určité požadavky.

V této práci je v algoritmech používáno numpy.array, protože se jedná o datovou strukturu, kterou používá OpenCV pro schraňování obrazových dat. V ukázkovém kódu níže, jsou snímky reprezentovány jako matice o velikosti H (výška) × W (šířka) × CH (počet kanálů). Kanály jsou za sebou seřazené ve formátu bgr a jednotlivé prvky matice, které odpovídají intenzitám pixelů, jsou datového typu uint8 (dekadicky rozsah celých čísel $<0;255>$).

Zapsání obrazových dat do numpy.array (pouze Python 3. x)

```
#!/usr/bin/python3
#==============================INICIALIZACE====================================
# Import potrebnych modulu
from picamera import PiCamera()
import numpy as np
import time

camera = Picamera()                     # Inicializace kamery
camera.resolution = (1600, 1200)        # Nastaveni rozliseni
img = np.empty((1200,1600,3), np.uint8) # Definice prazdneho pole pro snimky
time.sleep(1)                           # Kratka pauza pro zahrati kamery
#==============================SNIMANI=========================================
camera.capture(img, format = 'bgr')     # Ziskani snimku ve formatu bgr
print np.mean(img[:,:,0])               # Prumerna hodnota kanalu B <0;255>
camera.close()                          # Zavreni objektu kamery
```

Přes objekt třídy PiCamera je možné nastavovat parametry snímání a několik efektů, které se počítají rychle pomocí GPU. Některé efekty lze použít k předzpracování snímků a šetřit tak CPU, které je potřebné k dalším úkonům.

[22] *PiCamera*: https://picamera.readthedocs.io/en/release-1.13/

[23] *Python Imaging Library (PIL)*: http://www.pythonware.com/products/pil/

vybrané parametry objektu třídy PiCamera

brightness	exposure_mode	color_effects	sensor_mode
constrast	meter mode	image_effects	rotation
saturation	shutter_speed	awb_gains	vflip
sharpness	framerate	awb_mode	hflip
drc_strength	resolution	analog_gain	crop
add_overlay	iso	digital_gain	zoom

Jak již bylo zmíněno, obrazová data lze snímat v různých formátech. Výběru formátu je třeba přizpůsobit datovou strukturu. V případě formátu bgr a datové struktury numpy.array (uváděno výše v příkladech) stačí zajistit, aby mělo pole příslušnou velikost.

formáty snímaní dat s metodou capture

jpeg	png	gif	bmp	bayer
rgb	bgr	rgba	bgra	yuv

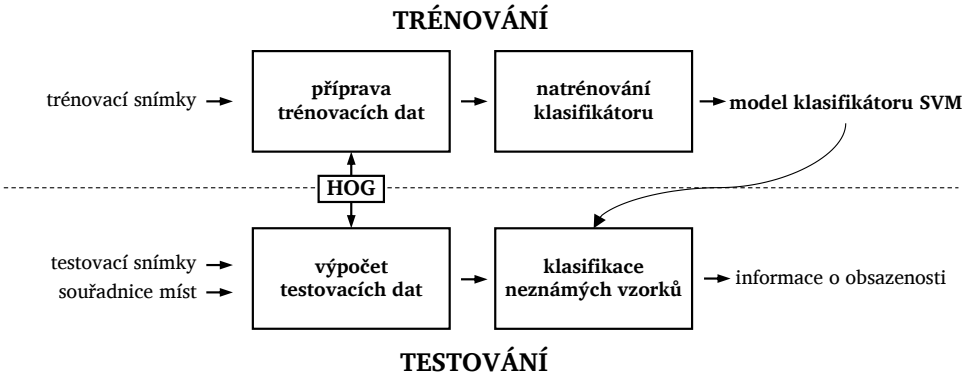
Zajímavostí je možnost získávat surová data ze snímače ještě před procesem demosaicing a před jinými zásahy GPU na RPi (formát bayer). Získávají se 10 bitové hodnoty odpovídající intenzitám pixelů pod Bayerovou maskou. Pro účely práce je využíván nekomprimovaný formát bgr a formát jpeg s parametrem kvality nastaveným na maximální hodnotu 100.

6. Implementace

Obsahem této kapitoly je především popis vytvořených algoritmů a postupů, zvolených pro dosažení funkčního systému. Implementace je provedena výhradně v programovacím jazyku Python. Pro získávání obrazu je použita knihovna Picamera a rozhraní RPi-Cam-Web-Interface. Obraz je zpracováván za pomoci knihoven OpenCV a NumPy. Cílový HW byl popsán v předchozí kapitole.

Mojí hlavní snahou bylo vytvořit výpočetně nenáročné algoritmy, které budou bez problémů fungovat na RPi Zero. Implementace se skládá z několika dílčích skriptů, které demonstrují různé části systému. Popis skriptů je součástí přílohy práce, spolu s video ukázkami pro vybrané skripty. Většinu demonstračního kódu jsem vyvíjel na PC, ale hlavní bloky rozpoznávání jsem realizoval funkcemi, které jsou kompatibilní se systémem na RPi.

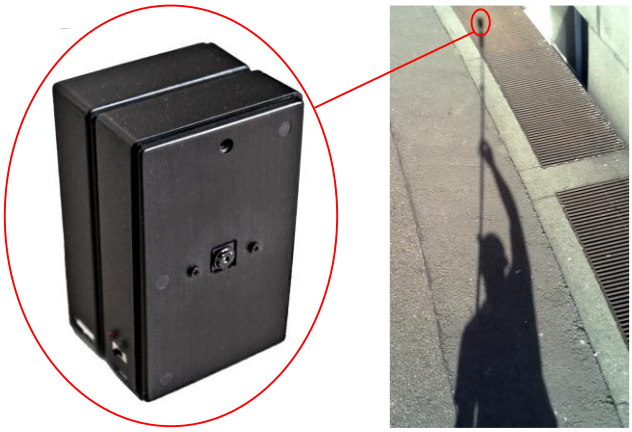
Celou implementaci lze v základu rozdělit na trénovací a testovací část (Obr. 27). V trénovací části jsou pořízené snímky parkovacích míst zpracovány a je získán model klasifikátoru SVM (kapitola 4.3.2). Model je předán do fáze testování, kde se na jeho základě, a na základě výřezů parkovacích míst zpracovaných pomocí HOG, určuje obsazenost.



Obr. 27: Základní schéma implementace

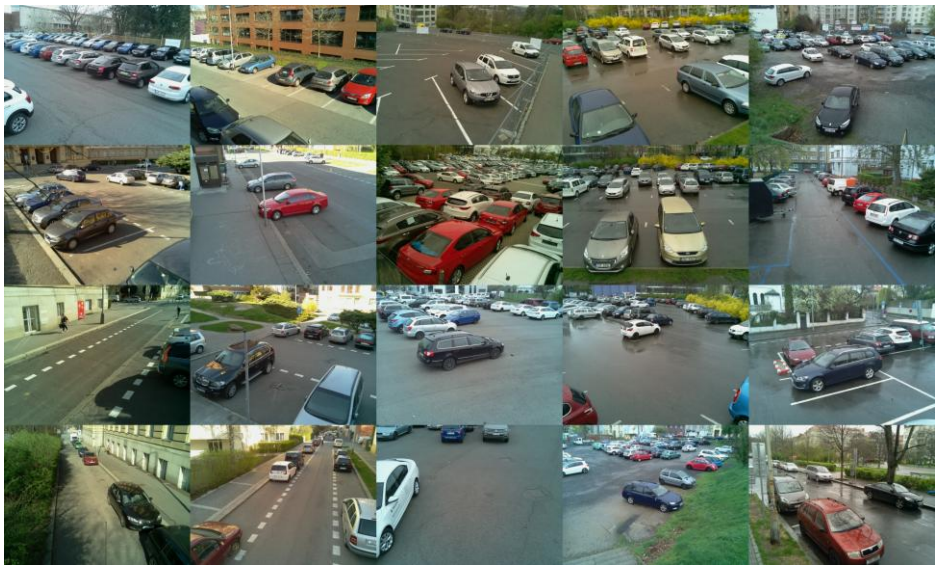
6.1 Trénování

Prvním krokem vývoje systému bylo pořízení tzv. trénovacích snímků. Trénovací snímky jsou obrazová data obsahující obsazená a volná parkovací místa. Pro získání těchto snímků jsem převážně použil vytvořené kamerové moduly umístěné na teleskopické tyči, která umožnila pořizovat snímky z výšky okolo 4 m od země (Obr. 28).



Obr. 28: Získávání trénovacích snímků

S kamerovými moduly a teleskopickou tyčí jsem za různého počasí pořídil okolo 1 000 snímků parkovacích ploch a okolí (Obr. 29). Snímky byly pořízeny ve formátu jpeg s maximální kvalitou.



Obr. 29: Ukázka snímků z kamerového modulu na teleskopické tyči

Mimo obrazových dat pořízených s touto konfigurací, jsem pro trénovací data použil i několik snímků vyfocených mobilním telefonem a digitální zrcadlovkou, především z oken budov.

Dalším krokem byla separace výřezů parkovacích míst z těchto snímků. Pro tento účel jsem vytvořil jednoduché grafické rozhraní, které umožňuje provádět výřezy velmi rychle a s dobrou přesností. Obdélníkový výřez je proveden pouze dvěma kliknutími myší, definováním rohů výřezu za pomoci vertikální a horizontální rysky. Stejný mechanismus je využíván i při definování parkoviště a také při testování vybrané oblasti zájmu. Videa demonstrující vyřezávání oblastí zájmu jsou součástí přílohy.

Z přibližně 1 000 vstupních snímků jsem vyseparoval 1 120 pozitivních vzorků (výřezy aut) a 1 378 negativních vzorků (výřezy volných míst a okolí). Auta jsem se snažil vyřezávat tak, aby vyplnila celý výřez snímku. Výřezy jsem ukládal do nekomprimovaného formátu png. Níže je znázorněn náhodný výběr několika pozitivních a negativních vzorků (Obr. 30).

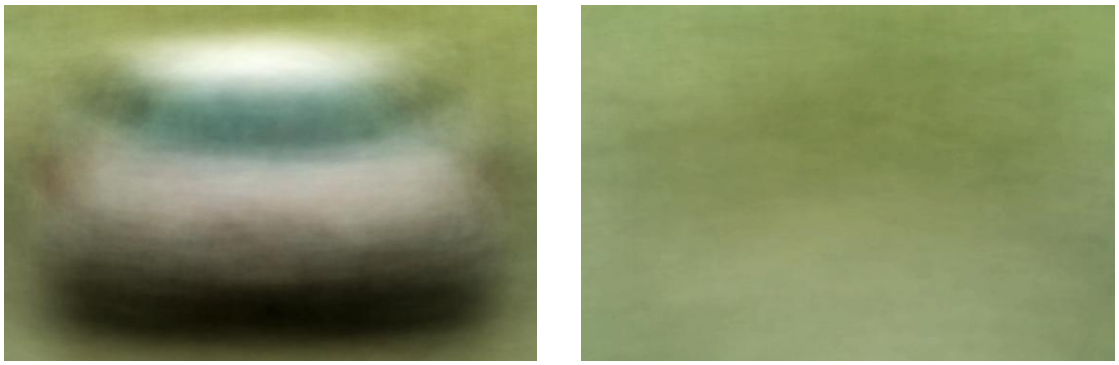
pozitivní vzorky **negativní vzorky**

Obr. 30: Vybrané pozitivní a negativní vzorky

Po vyříznutí vzorků z originálních vstupních snímků nedochází k žádným dalším úpravám, a proto mají vzorky různé rozlišení a poměry stran. Pro extrakci příznakových vektorů pomocí HOG deskriptoru jsou však potřeba snímky s konstantní velikostí (kapitola 4.3.1). Musel jsem se tedy rozhodnout, na jakou velikost vzorky přeformátuji. Provedl jsem analýzu distribuce poměrů stran, na jejímž základě jsem se rozhodl pro poměr stran 3:2.

Přeformátování vzorků na stejnou velikost umožňuje provádět zajímavé experimenty, jako například vypočtení průměrného snímku zvlášť pro pozitivní a negativní vzorky. Kontrast takového snímku je velmi nízký, a proto byl v ilustraci uměle zvýšen (Obr. 31).

průměrný pozitivní vzorek **průměrný negativní vzorek**



Obr. 31: Průměrný pozitivní a negativní vzorek

Vzorky konstantní velikosti je už možné předat deskriptoru HOG a vypočítat z nich příznakové vektory. Tento přístup jsem vyzkoušel, ale musel jsem od něj ustoupit, neboť výsledný klasifikátor trpěl příliš mnoha falešnými pozitivy (viz kapitola 4.1.1). Tato chyba byla zřejmě způsobená tím, že standardní HOG nedokáže příznakovým vektorem rozumně popsat rozdíly různě natočených aut.

Pro snížení počtu falešných pozitivů jsem přistoupil k řešení, ve kterém se uvažuje vzájemná pozice kamery a aut. Z pohledu trénování systému to znamenalo, že jsem musel rozdělit pozitivní vzorky do několika kategorií podle orientace aut v obraze. Orientaci jsem popsal jednočíselnou hodnotou, která odpovídá úhlu, jež v obrazu svírá spojnice mezi koly vozu s horizontální linií (úhel φ na Obr. 32). Spojnici jsem pro každý vzorek definoval manuálně, algoritmem velmi podobným algoritmu použitému pro vyřezávání.

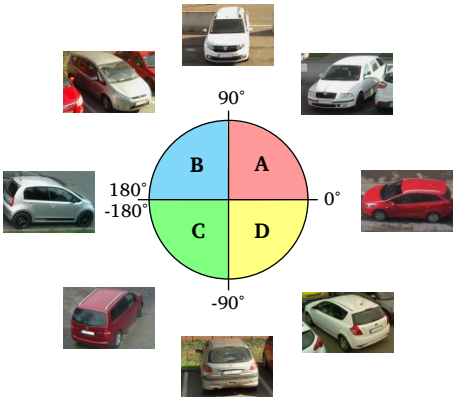
Obr. 32: Aproximace orientace auta v obraze, definování spojnice mezi koly

Tento model je velmi jednoduchý a opravdovou orientaci vozu v obraze pouze aproximuje. Při snímání aut z podobné výšky však umožní docela dobře rozlišit pozitivní vzorky podle jejich orientace (Obr. 33 a Obr. 34).

Obr. 33: Snímky aut podle odpovídajícího úhlu φ

Úhel φ může nabývat hodnot od -180° až po 180°, ve skutečnosti však nemá smysl všechny orientace pro trénovací data uvažovat. Řidiči mohou do parkovacího místa obvykle zaparkovat dvěma způsoby (např.: -90° a 90° nebo -135° a 45°) a tyto úhly se tedy hodí sloučit, čímž se rozsah zmenší na 0° až 180°. I poté je zbytečné pracovat se všemi úhly, neboť auta jsou v podstatě symetrická a horizontální převrácení dovoluje další sloučení úhlů (např.: 135° odpovídá 45°). Ve skutečnosti stačí tedy rozdělit všechny pozitivní

vzorky do kategorií v rozsahu 0° až 90° a přiřadit k nim snímky jiných úhlů podle zmíněných pravidel. Výše popisované vlastnosti ještě ilustruje snímek níže (Obr. 34).



vztah kvadrantu vůči kvadrantu A	
B	horizontální symetrie
C	opačná orientace v parkovacím místě
D	opačná orientace v parkovacím místě vůči horizontální symetrii

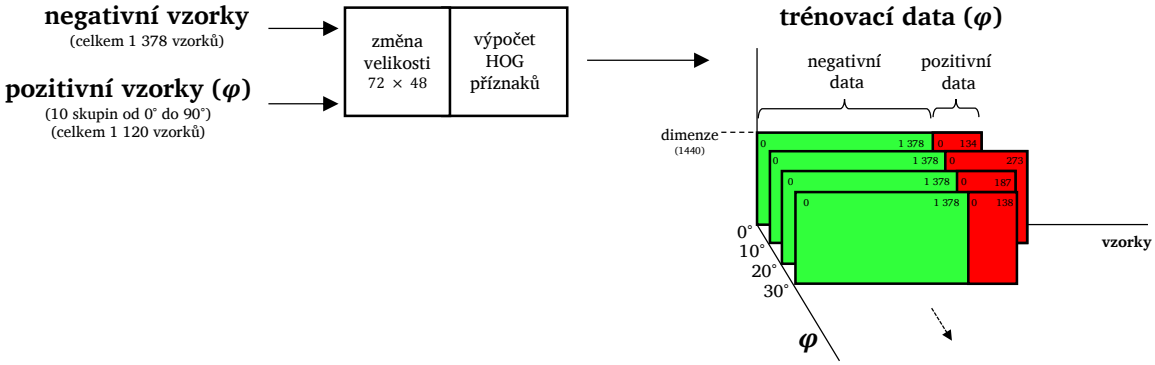
Obr. 34: Znázornění orientací auta a vlastností kvadrantů

Rozdělil jsem všechny pozitivní vzorky (celkem 1120) do 10 skupin po 10° bez překryvu. Počty vzorků v jednotlivých skupinách jsou poněkud nevyvážené (Tab. 6), což je zapříčiněno tím, že jsem nevědomky vyfotil nejvíce aut z boku.

název skupiny	0	10	20	30	40	50	60	70	80	90
počet snímků	134	273	187	138	107	96	66	44	46	29

Tab. 6: Počet vzorků v jednotlivých skupinách pozitivních vzorků

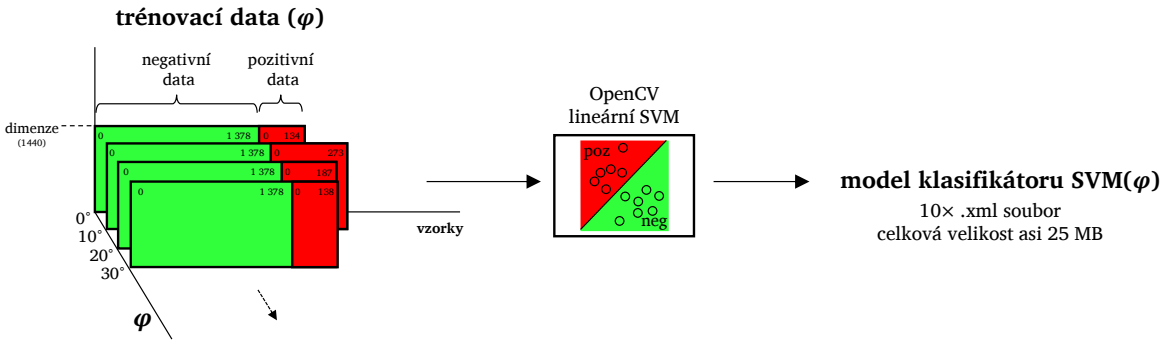
Ze skupin pozitivních vzorků závislých na úhlu a negativních vzorků jsem vytvořil trénovací data pomocí deskriptoru HOG (4.3.1). Všechny vstupní vzorky jsou nejdříve zmenšeny na rozlišení 72 × 48 a následně je pro každý vypočten jeden příznakový vektor dimenze 1440. Výsledky jsou vhodným způsobem uspořádány do matic (Obr. 35).



Obr. 35: Schéma výpočtu trénovacích dat

Výsledné matice jsem uložil pomocí knihovny NumPy do nekomprimovaného formátu npz spolu se značkami tříd vzorků a předal je k dalšímu zpracování.

Posledním krokem v trénovací části implementace bylo natrénování modelu klasifikátoru SVM (kapitola 4.3.2), které spočívalo v předání trénovacích dat do OpenCV funkce, jež na základě zadaných parametrů vypočítala podpůrné vektory a nalezla optimální nadrovinu. Pro minimalizaci výpočetní náročnosti při testování jsem zvolil lineární jádro, při kterém se nezvyšuje dimenze příznakových vektorů. Díky poměrně velké dimenzi a malému počtu vzorků se ukázalo, že jsou data trénovací množiny lineárně separabilní. Výsledný model klasifikátoru SVM se skládá z deseti xml souborů o celkové velikosti okolo 25 MB (Obr. 36). Model je výstupem celé trénovací fáze a spolu s parametry HOG deskriptoru jde o jediná data, která jsou předávána do testovací fáze.



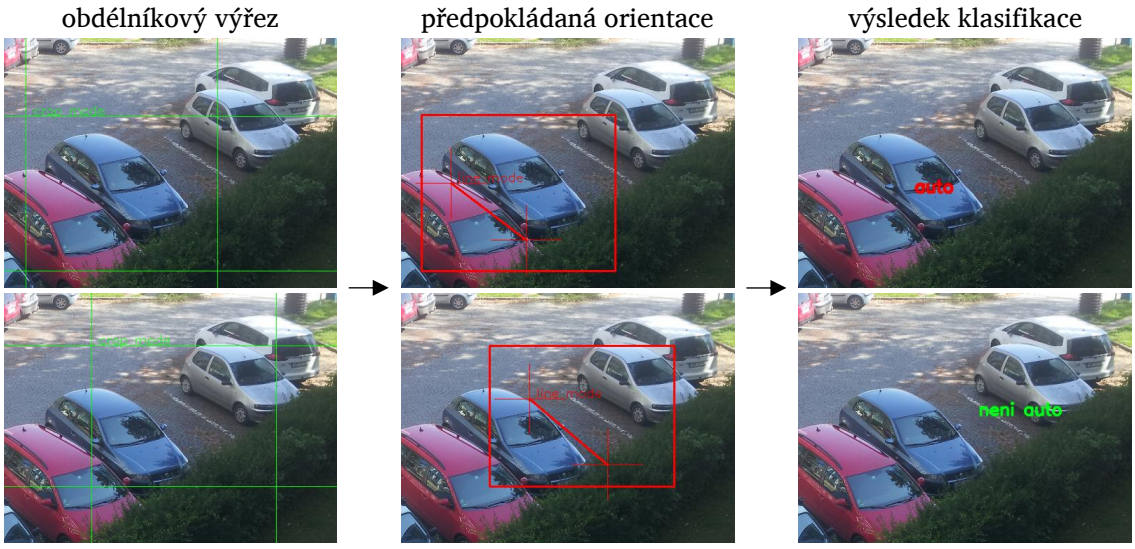
Obr. 36: Příprava modelu klasifikátoru

6.2 Testování

V testovací fázi implementace se zabývám automatickým určením obsazenosti parkovacích míst ve vstupním snímku na základě souřadnic, získaného modelu klasifikátoru z trénovací fáze a parametrů HOG deskriptoru. Pro intuitivnější demonstraci funkce algoritmů jsem rozdělil testování celkem do čtyř kategorií:

1. testování jednoho výřezu zadaného prostřednictvím grafického rozhraní;
2. testování více parkovacích míst z pohledu jedné kamery;
3. testování více parkovacích míst z pohledu více kamer;
4. testování databáze PKLot.

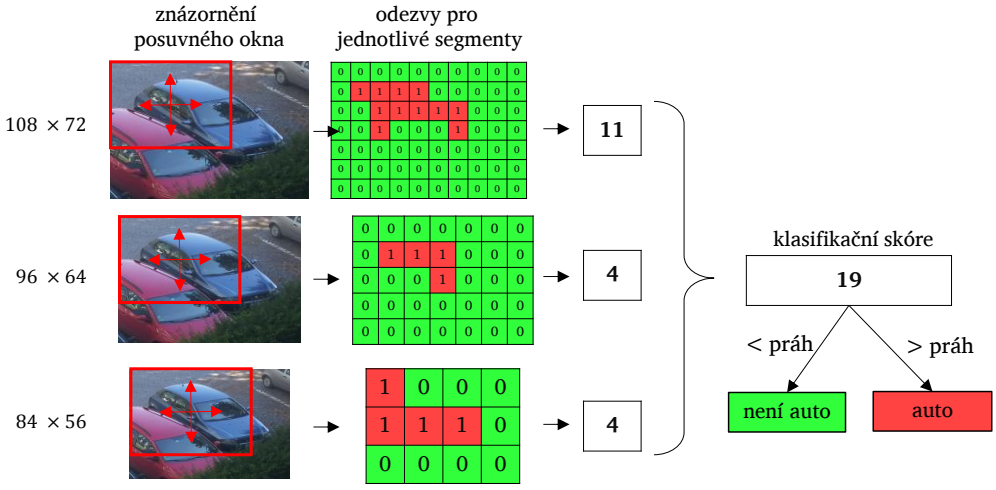
Do první kategorie spadá algoritmus, který slouží k **okamžité klasifikaci manuálně zadaného výřezu**. Užitečnost tohoto algoritmu spočívá v získání velmi rychlého náhledu na úspešnost klasifikace v různorodých podmínkách. Grafické rozhraní umožňuje procházet snímky ve vybraném adresáři a specifikovat v nich oblasti, které mají být rozpoznány. Zprvu je definován obdélníkový výřez a následně i předpokládaná orientace vozu, po čemž následuje okamžitá klasifikace (Obr. 37).



Obr. 37: Testování jednoho výřezu zadaného prostřednictvím grafického rozhraní

Nyní krátce popíši **jádro klasifikace jednoho výřezu obrazu**, které je totožné pro všechny kategorie testování. Na rozdíl od trénování není u testování možné specifikovat výřezy těsně okolo vozu (Obr. 30), neboť jeho přesná pozice a velikost není známá. Místo výřezů aut se provádějí výřezy oblastí parkovacích míst, které jsou větší než samotná auta.

Klasifikace je potom realizována pomocí posuvného okna v různě velikých kopiích výřezu oblasti parkovacího místa (Obr. 38). Z každé kopie se pomocí posuvného okna 72×48 získá několik segmentů. Pro každý jednotlivý segment je vypočten příznakový vektor HOG, stejným mechanismem jako u trénování. Výsledné příznakové vektory jsou analyzovány klasifikačním modelem SVM a je určena predikce třídy. V případě, že model klasifikuje daný příznakový vektor jako auto, je predikce rovna jedné. V opačném případě je hodnota predikce nulová. Sečtením výsledků pro všechny segmenty se získá klasifikační skóre, které se pro stanovení finálního výsledku klasifikace prahuje.



Obr. 38: Klasifikace jednoho výřezu parkovacího místa

Celý algoritmus zjišťování obsazenosti jednoho parkovacího místa je relativně výpočetně nenáročný. Na PC[24] jsem naměřil dobu zpracování jednoho parkovacího místa okolo 10 ms a na RPi Zero okolo 100 ms. Analýza více parkovacích míst najednou algoritmus ani nezpomaluje, ani nezrychluje. RPi Zero je tedy schopné analyzovat obsazenost rychlostí 10 parkovacích míst za sekundu.

Analýzou více parkovacích míst najednou se zabývám ve zbývajících kategoriích testování. Ve druhé kategorii jsou naráz automaticky analyzovány parkovací místa v záběru jedné kamery. Tomuto typu testování však musí předcházet definování jednotlivých parkovacích míst. Každé parkovací místo je definováno pomocí grafického rozhraní šesti hodnotami (1 identifikátor místa, 4 hodnoty souřadnic pro provedení výřezu a 1 úhel předpokládané orientace vozu v parkovacím místě (Tab. 7 a Obr. 39)).

ID	souřadnice výřezu				φ
00	706	587	959	815	-127
01	180	587	499	756	-11
02	286	529	567	673	-5
03	349	480	610	635	-3
04	828	503	1098	691	-135
…	…	…	…	…	…
13	1185	227	1293	349	-137

Tab. 7: Definice parkovacích míst

Obr. 39: Ilustrace pro tabulku vlevo

Testování všech míst probíhá v cyklu tak, že každé parkovací místo je vyhodnoceno zvlášť podle již popsaného mechanismu. Výsledkem klasifikace je vektor obsazenosti vycházející z prahování klasifikačního skóre pro každé místo. Tabulka níže (Tab. 8) znázorňuje vektor klasifikačního skóre a vektor obsazenosti pro snímek na Obr. 39 s prahem nastaveným na hodnotu 15.

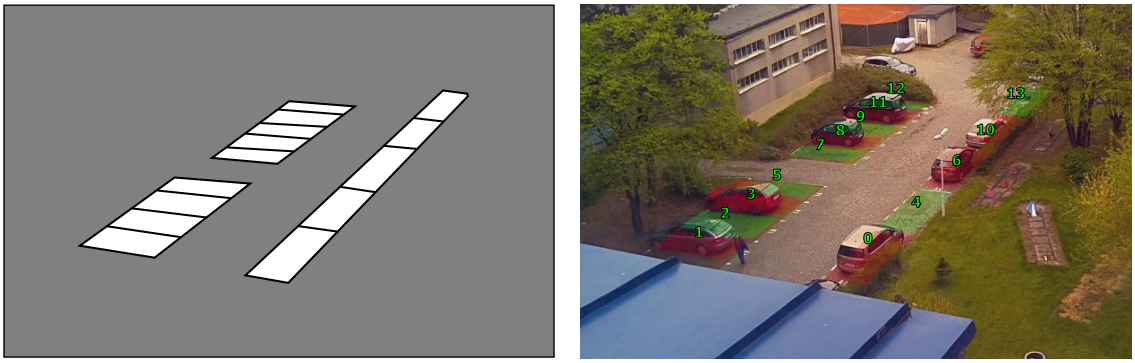
ID	00	01	02	03	04	05	06	07	08	09	10	11	12	13
skóre	50	77	1	89	0	0	34	1	21	2	74	62	0	0
obsazenost	1	1	0	1	0	0	1	0	1	0	1	1	0	0

Tab. 8: Výsledek klasifikace záběru na více parkovacích míst

Výsledky klasifikace jsem vizualizoval přímo do obrazu ploškami, jejichž souřadnice jsou automaticky získány z masek, které jsem vytvářel v grafickém editoru (Obr. 40).

[24] PC: CPU AMD jádro 3,2 GHz, RAM 8 GB. Algoritmy využívají pouze 1 jádro procesoru

maska pro vizualizaci obsazenosti **vizualizace obsazenosti**

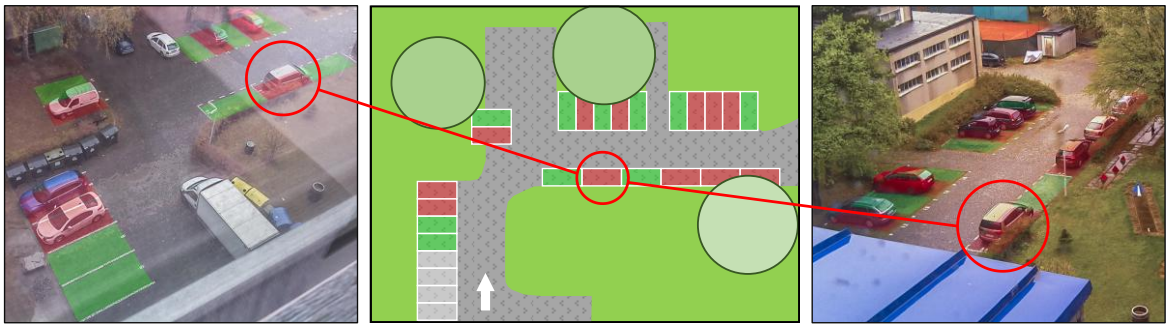


Obr. 40: Vizualizace obsazenosti

V předposlední kategorii testování se zabývám **slučováním informací o obsazenosti z více kamer**. Slučování má na starost server, který od kamer přijímá vektory obsazenosti (Obr. 4), případně vektory klasifikačního skóre. Vhodný způsob předávání dat jsem nastínil v kapitole 3.1 a v kapitole 5.2.1 jsem informoval o způsobu předávání dat vytvořenými kamerovými moduly.

Zorná pole kamer se mohou i nemusí překrývat. Pro případ, že více kamer zabírá jedno parkovací místo, jsem nastavil kritérium výběru záběru (kalibrace). Kritérium může odpovídat manuálnímu nastavení (místům jsou jasně přiřazené kamery) nebo může vycházet z výsledků klasifikačního skóre. V druhém případě je jedno parkovací místo analyzováno všemi kamerami, které ho zabírají a server následně vyhodnotí věrohodnost výsledků podle získaných hodnot klasifikačního skóre. Jako rozhodující záběr je vybrán ten, pro který vyjde klasifikační skóre nejvzdálenější od prahu.

Výsledkem testování celého parkoviště z pohledu více kamer je vektor obsazenosti, jež je možné opět vizualizovat. Pro tento účel jsem zvolil jednoduchou grafiku, která zobrazuje situaci parkoviště shora (Obr. 41). V případě, že by měl být systém nasazen ve velkém měřítku, by bylo vhodnější vizualizovat situaci například vrstvou umístěnou na mapové podklady, podobně jako je to realizováno třeba v systému SFpark [3].



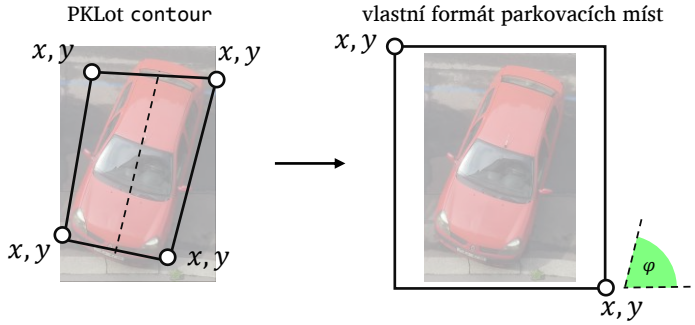
Obr. 41: Slučování informací o obsazenosti z více kamer s překryvem

Poslední kategorií testování je **testování databáze PKLot** [4]. Databáze PKlot mi posloužila jako evaluační sada dat pro zjištění úspěšnosti klasifikace pomocí implementovaných algoritmů, natrénovaných na vlastních datech. Výsledky testování se zabývá kapitola 7. Kromě samotných snímků parkoviště obsahuje databáze soubory xml, jejichž součástí jsou údaje o jednotlivých parkovacích místech v obraze. Jedno parkovací místo je v databázi definováno způsobem uvedeným níže.

formát parkovacího místa v databázi PKlot [4]

```
<space id="14" occupied="1">
	<rotatedRect>
		<center x="300" y="207"/>
		<size w="55" h="32"/>
		<angle d="-74"/>
	</rotatedRect>
	<contour>
		<point x="278" y="230"/>
		<point x="290" y="186"/>
		<point x="324" y="185"/>
		<point x="308" y="230"/>
	<contour>
</space>
```

Formát definování jednoho parkovacího místa, použitý v této práci, se od tohoto formátu mírně liší. Zásadní rozdíl spočívá především v tom, že autoři definovali parkovací místa čtyřmi kliknutími (contour) velmi těsně okolo předpokládané pozice vozu a ze získaných souřadnic nechali automaticky spočítat parametry natočeného obdélníku (rotatedRect). Parametry natočeného obdélníku však nebyly pro separaci parkovacích vyhovující. Hlavní problém spočíval v tom, že parametr natočení (angle) často dobře neodpovídal skutečné orientaci parkovacího místa. Tento problém byl vyřešen využitím původních souřadnic (contour), ze kterých jsem sestrojil přímku, jejíž úhel jsem považoval za orientaci místa. Pro separaci výřezu jsem také využil původních souřadnic, s tím, že jsem nepatrně rozšířil ohraničující oblast a transformoval jí na nenatočený obdélník (Obr. 42).



Obr. 42: Ilustrace transformace souřadnic pro využití databáze PKLot

Po změně formátu souřadnic probíhá klasifikace stejně jako při testování více parkovacích míst z pohledu jedné kamery. Ukládají se výsledky klasifikačního skóre, ze kterých lze srovnáním se skutečnou obsazeností určit úspěšnost klasifikace.

7. Spolehlivost vytvořeného systému

Spolehlivost systému lze prověřit analýzou úspěšnosti klasifikace při různých podmínkách testování. Jak bylo uvedeno ve 3. kapitole, existuje mnoho faktorů, které mohou ovlivnit výsledný vzhled výřezu parkovacího místa. Systém, připravený pro konkrétní parkoviště a konkrétní podmínky, může dosahovat velmi vysokých úspěšností klasifikace v daných podmínkách, ale jiných v podmínkách může výrazně selhávat.

Při implementaci systému jsem se bez ohledu na podmínky testování snažil dosáhnout dobré úspěšnosti. Úspěšnost klasifikace jsem ověřil celkem na třech různých sadách testovacích snímků, které obsahují vyvážený počet obsazených a volných míst (Tab. 9). Pro zhodnocení jsem použil základní metriky binární klasifikace (kapitola 4.1).

název sady	**počet různých záběrů [-]**	**počet výřezů parkovacích míst [-]**		
		volno	**obsazeno**	**celkem**
ParkingSlots	142	326	448	774
FELLot	4	38 661	32 426	71 084
PKLot	3	348 125	318 951	667 076

Tab. 9: Základní parametry použitých sad testovacích snímků

Tato kapitola se dále zabývá popisem charakteristik použitých testovacích sad a výsledky úspěšnosti klasifikace. Na základě získaných výsledků provedených testů lze konstatovat, že implementovaný systém je velice robustní vůči změnám podmínek testování a dosahuje **přesnosti nad 90 %**.

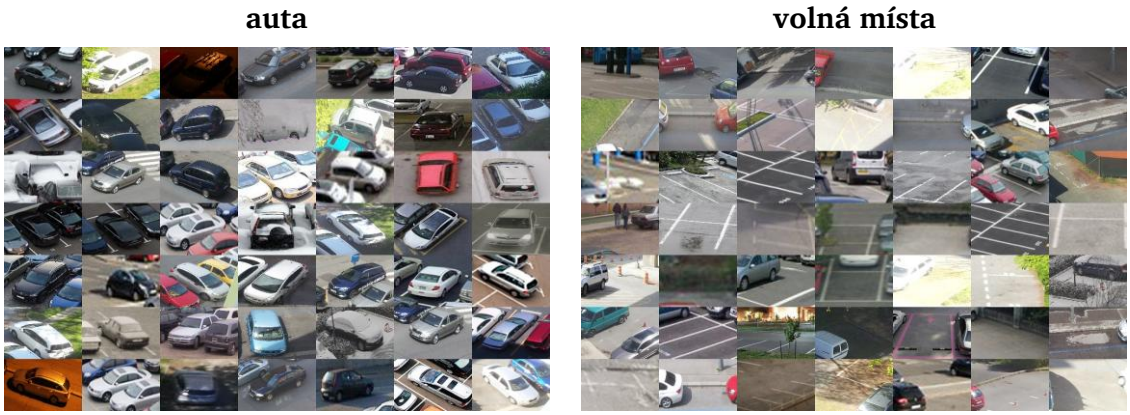
Algoritmus dokáže dobře rozpoznávat i výřezy, na které nebyl vůbec trénován (zasněžená auta, auta se střešními nosiči, jiné typy aut, auta v noci apod.), což svědčí o vhodném výběru kombinace deskriptoru a klasifikátoru. Co se týče chybných výsledků klasifikace, zdá se, že systém trpí více falešnými negativy než pozitivy (z ROC analýzy PKLot 7.3). To může být způsobeno tím, že trénovací data obsahují výrazně více negativních, než pozitivních vzorků (kapitola 6.1 a Tab. 6).

Výřezy pro chybné výsledky klasifikace mají na pohled málo společných znaků, a je tedy těžké specifikovat konkrétní nedokonalosti systému z pohledu jeho úspešnosti. O falešných pozitivech se dá říci, že obvykle obsahují vetší množství ostrých hran vzdáleně připomínajících tvar vozu (čáry na silnici, stopy od vody apod.). U falešných negativů se mi nepodařilo prostým pohledem najít žádné výrazné znaky, které by mohly způsobovat chybnou klasifikaci.

7.1 PKSlots

Sada PKSlots se skládá celkem ze 774 výřezů parkovacích míst. Zhruba polovinu výřezů jsem vyseparoval ze snímků, které jsem vyfotil mobilním telefonem nebo digitální zrcadlovkou. Druhou polovinu výřezů jsem získal ze snímků stažených z Internetu.

Všechny výřezy byly provedeny celkem ze 142 velmi rozdílných snímků. Snímky byly rozličného rozlišení a byly získány rozdílnými zařízeními za různých podmínek, z různých úhlů i vzdáleností. Některá parkovací místa jsou dokonce zasněžená. Ukázka vybraných výřezů z této testovací sady je na Obr. 43.



Obr. 43: Náhodně vybrané výřezy ze sady PKSlots

Výřezy jsem provedl pomocí již popisovaného grafického rozhraní a nechal je klasifikovat vytvořeným algoritmem. Klasifikované výřezy byly poté seřazené podle skóre, což usnadnilo označování skutečných tříd výřezů. Po označení skutečných tříd jsem provedl ROC analýzu (4.1.2) a zjistil jsem optimální nastavení prahu pro maximalizaci přesnosti. Pro optimální práh ($t = 8$) vyšla následující matice záměn.

obecná matice záměn

TP	*FN*
FP	*TN*

četnosti [-]

423	25
28	298

relativní četnosti [%]

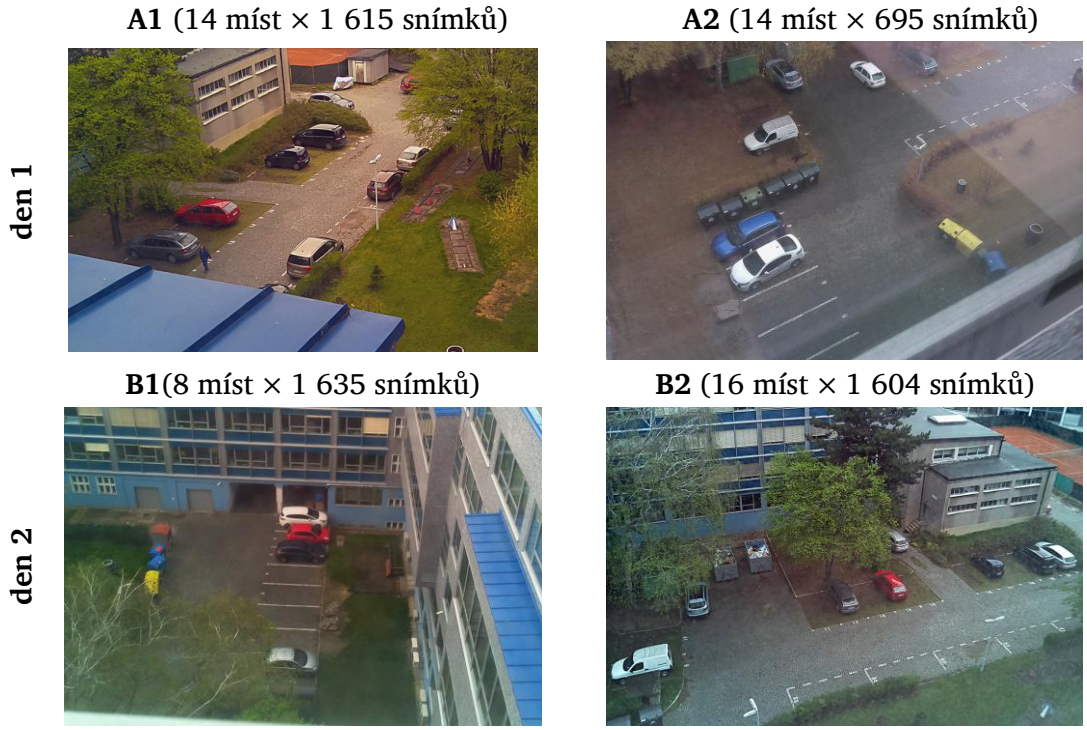
94,4	**5,6**
8,6	**91,4**

Tab. 10: Matice záměn PKSlots, $t = 8$

Ze 774 výřezů jich bylo chybně klasifikováno pouze 53 (25 aut a 28 volných míst). Z tohoto údaje lze snadno vypočítat přesnost $\boldsymbol{ACC} = \frac{(774-53)}{774} = \mathbf{93,2}$ %. Vzhledem k velké variabilitě snímků jde o velmi uspokojivý výsledek a dobré je i to, že hodnota *FN* je nižší než *FP*.

7.2 FELLot

Testovací sada FELLot je sada snímků malé oblasti fakultního parkoviště, kterou jsem pořídil v průběhu dvou dní sestavenými kamerovými moduly. Snímky jsem získával s 15s intervalem pomocí knihovny Picamera (5.2.1) přibližně[25] od 09:00 do 16:00. Kamery byly umístěné za skly oken budovy. Celkem jsem nafotil 5 549 snímků (ze 4 záběrů), ze kterých jsem extrahoval 71 084 výřezů parkovacích míst (Obr. 44).



Obr. 44: Jednotlivé záběry kamer pro testovací sadu FELLot

Tuto datovou sadu jsem testoval jako první, a to bohužel trochu nešikovným způsobem. Srovnání reálné a klasifikované obsazenosti jsem totiž provedl jednorázově manuálně pro jedno nastavení prahu. V důsledku této chyby jsem nemohl provést ROC analýzu jako u ostatních testovacích sad a úspěšnost tedy uvádím pouze pro úroveň prahu $t = 15$, která však zřejmě není optimální vzhledem k přesnosti. Výsledky této sady jsou uvedeny pro všechny záběry dohromady (Tab. 11).

obecná matice záměn

TP	*FN*
FP	*TN*

četnosti [-]

27 639	4 784
773	37 888

relativní četnosti [%]

85,2	**14,8**
2,0	**98,0**

Tab. 11: Matice záměn FELLot, $t = 15$

Přesnost $\boldsymbol{ACC} = \frac{(27\,639+37\,888)}{71\,084} = \mathbf{92,2}\ \%$.

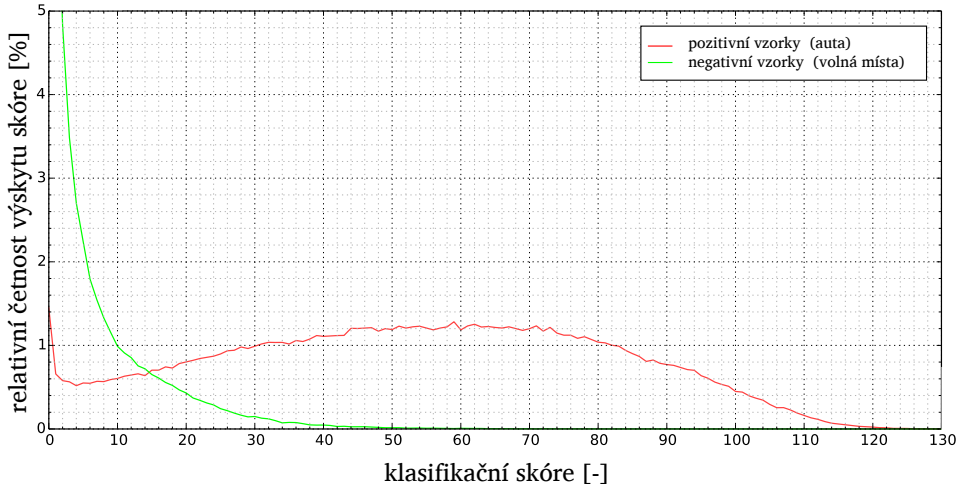
[25] Vyjma záběru A2, který bežel pouze do 12:00

7.3 PKlot

Poslední a největší použitou datovou testovací sadou je databáze PKlot [4] původem z brazilských univerzit v Paraná. Databáze obsahuje 695 899 výřezů parkovacích míst, ale pro účel této práce jsem jich využil o něco méně (667 076) z důvodu problémů s některými xml soubory (k některým místům neexistovala informace o obsazenosti). Autoři nasnímali celkem tři různá parkoviště (PUCPR, UFPR05, UFPR04) v průběhu několika dní pomocí HD webkamery a získaná data rozdělili do třech skupin podle počasí (Obr. 45). Pro jednotlivá místa v obraze zapsali jejich souřadnice a obsazenost. Způsob testování této databáze uvádím na konci kapitoly 6.2.

Obr. 45: Vybrané snímky z databáze PKLot, převzato z [4]

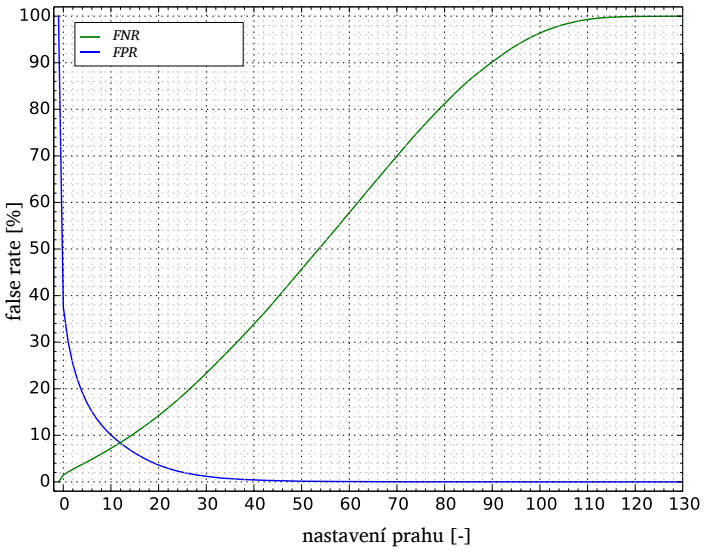
Vypočítal jsem klasifikační skóre pro každý výřez a výsledky roztřídil podle skutečné obsazenosti. Následně jsem vynesl distribuci klasifikačního skóre (Obr. 46).



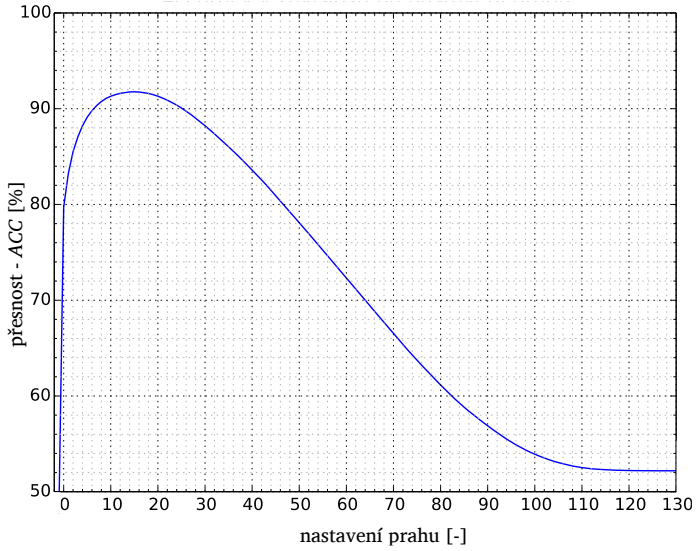
Obr. 46: Distribuce klasifikačního skóre všech výřezů z databáze PKLot

Krok na ose *x* odpovídá hodnotě 1, neboť klasifikační skóre může nabývat pouze celých čísel od 0 do *N*, kde *N* odpovídá celkovému počtu segmentů získaných posuvným oknem (kapitola 6.2 a Obr. 38). Pro většinu negativních vzorků vyšlo klasifikační skóre nulové (62 % všech negativních vzorků). Četnost negativních vzorků s vyšším klasifikačním skóre ubývala exponenciálně. Distribuce klasifikačního skóre pozitivních vzorků lehce připomíná normální rozdělení se střední hodnotou okolo 60 a se směrodatnou odchylkou zhruba 30. Hlavní rozdíl oproti normálnímu rozdělení nastává v 0, což je zapříčiněno tím, že velkému počtu pozitivních vzorků bylo špatně přiděleno nulové klasifikační skóre.

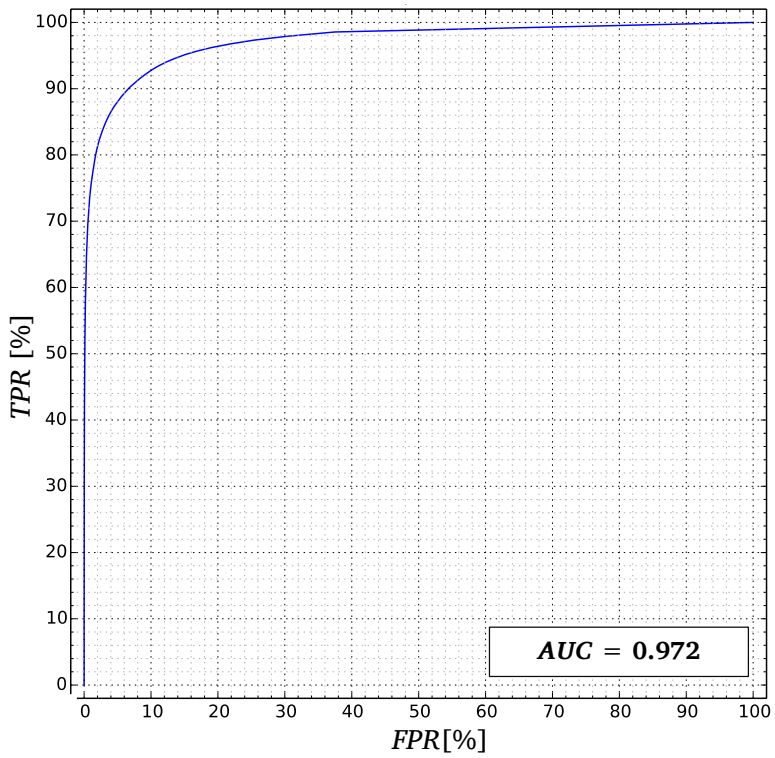
Z nenormalizované varianty distribuce klasifikačního skóre lze prahováním získat řadu charakteristik (kapitola 4.1). Výsledky pro jednotlivé kategorie snímků (počasí, parkoviště) je možné dohledat v datové příloze. Zde opět uvádím pouze celkové výsledky, tedy výsledky pro všech 667 076 klasifikovaných výřezů (Obr. 47, Obr. 48, Obr. 49).



Obr. 47: Závislost míry chyb klasifikace na nastavení prahu klasifikačního skóre, PKLot



Obr. 48: Závislost přesnosti (*ACC*) na nastavení prahu, PKLot



Obr. 49: ROC křivka PKLot

Graf na Obr. 47 udává závislost míry chybné klasifikace na nastavení prahu s tím, že je zde rozlišeno, zda bylo rozpoznáváno volné, nebo obsazené místo. V probíraných systémech však nelze na *FPR* a *FNR* pohlížet zcela samostatně. Další graf (Obr. 48) vychází z celkového počtu chyb klasifikace při různě nastaveném prahu. Menší počet chyb znamená přesnější klasifikaci. **Maximální přesnost *ACC*** vychází **91,8 %**, a to pro práh $t = 15$**.** Matice záměn pro toto nastavení prahu je uvedena v Tab. 12.

obecná matice záměn		četnosti [-]		relativní četnosti [%]	
TP	*FN*	285 542	33 409	**89,5**	**10,5**
FP	*TN*	21 562	326 563	**6,2**	**93,8**

Tab. 12: Matice záměn PKLot, $t = 15$

Maximální přesnost však nemusí být nejdůležitějším kritériem při nastavování prahu klasifikačního skóre. Snahou může být snížit *FN* i za cenu snížení přesnosti, neboť jde o situaci, která je v případě předávání obsazenosti řidičům velmi nežádoucí (kapitola 4.1.2). Je možné například zvolit práh $t = 8$, jemuž odpovídá matice záměn v Tab. 13.

obecná matice záměn		četnosti [-]		relativní četnosti [%]	
TP	*FN*	299 812	19 139	**94,0**	**6,0**
FP	*TN*	42 566	305 559	**12,2**	**87,7**

Tab. 13: Matice záměn PKLot, $t = 8$

Při tomto prahu se sice sníží přesnost ***ACC*** na hodnotu **90,7 %,** ale minimalizuje se počet situací, kdy systém označuje obsazené parkovací místo jako volné (***FNR*** **= 6 %**). Další snižování prahu umožňuje tuto situaci minimalizovat ještě více, ale je to již za cenu prudkého nárůstu *FP* a poklesu *ACC*.

Poslední graf (Obr. 49) shrnuje charakter úspěšnosti jedinou křivkou, nezávisle na počtu pozitivních a negativních vzorků v testovací sadě. Plocha pod křivkou ***AUC*** vychází **0,972**, což lze považovat za velmi dobrý výsledek.

Klasifikaci všech snímků z databáze jsem vizualizoval formou 6minutového časosběrného videa. Jeden ze snímků s vizualizací je zde znázorněn níže na Obr. 50.

Obr. 50: Vizualizace klasifikace databáze PKLot

8. Závěr

V této diplomové práci se zabývám distribuovaným kamerovým systémem pro správu parkovacích míst, ve kterém inteligentní kamery transformují obrazová data na informace o obsazenosti parkoviště. Cílem práce bylo navrhnout a implementovat všechny části systému počínaje kamerovými moduly a konče algoritmy, jež jsou schopny vyhodnocovat obsazenost parkovacích míst na základě informací z více kamer.

Provedl jsem analýzu dostupných řešení detekce obsazenosti parkovacích míst a uvedl jsem jaké výhody kamerové systémy přinášejí, oproti jiným konvenčnějším řešením jako například ultrazvukovým detektorům. Na základě studie distribuovaných kamerových systémů jsem sestavil tři malé levné bateriové kamerové moduly, založené na počítači Raspberry Pi Zero, které jsem následně využil pro trénování i testování algoritmů. Algoritmy jsem napsal v programovacím jazyku Python zejména spolu s knihovnami OpenCV, NumPy a Picamera.

Pro účely trénování jsem pomocí kamerových modulů získal okolo 2 500 různorodých výřezů parkovacích míst, které jsem rozdělil podle obsazenosti. Výřezy obsahující auta jsem dále rozdělil do několika kategorií podle orientace aut v obraze. Z roztříděných výřezů jsem následně pomocí HOG deskriptoru vypočítal příznakové vektory, z nichž jsem získal model úhlově závislého klasifikátoru SVM. Model je předáván do fáze testování, kde se na jeho základě určuje obsazenost. Klasifikace je realizována s využitím posuvného okna v různě velkých kopiích vstupních výřezů parkovacích míst. Výsledky obsazenosti z jednotlivých modulů mohou být předávány serveru prostřednictvím Wi-Fi standardů. Server slučuje informace z více kamer a výsledek může sdílet s potencionálními zájemci.

V implementaci jsem se zaměřil především na zpracování obrazu v jednotlivých kamerových modulech, neboť jde o vstupní předpoklad úspěšného systému při slučování informací z více kamer. Slučování informací z více kamer jsem realizoval pouze jednoduchými mechanismy, které v případě překryvu kamer upřednostňují jeden záběr před jinými na základě zadaného kritéria.

Implementovaný algoritmus dokáže na Raspberry Pi Zero rozpoznávat obsazenost rychlostí 10 parkovacích míst za sekundu s přesností obvykle vyšší než 90 %, a to ve velmi širokém spektru podmínek. Úspěšnost algoritmu jsem ověřil na třech různorodých testovacích sekvencích obsahujících celkem přes 700 000 výřezů parkovacích míst.

Implementovaný systém by bylo možné rozšířit celou řadou vylepšení. Z hardwarového pohledu se jedná především o snížení spotřeby kamerových modulů pomocí spacích režimů, které by spolu se solárním panelem mohly zajistit energetickou soběstačnost modulů. Pro implementované algoritmy se jako rozšíření nabízí uplatnění systému na obecnější typ parkování *on-street*, kde by bylo potřeba nejen klasifikovat zadané výřezy, ale parkovací místa vyhledávat. Dalšími možnými kroky pro vylepšení algoritmů by mohla být jejich optimalizace a například využití grafické karty Raspberry Pi pro předzpracování snímků.

Tato práce je příspěvkem k řadě moderních systému pro správu parkovacích míst a nabízí nový přístup k detekci obsazenosti pomocí nenáročných algoritmů zpracování obrazu. Praktické využití implementovaného systému si s drobnými modifikacemi dokáži představit především v oblasti zefektivnění městského parkování.

Seznam obrázků a tabulek

Obrázky, u nichž není uveden zdroj, jsou dílem autora práce. Mohou se však na nich vyskytovat malé části, které byly získány z Internetu vyhledáváním s filtrem: „Povoleno opětovné použití s úpravami“. Zdroje také nejsou uváděny u výstupů algoritmů.

Seznam použité literatury a zdrojů

[1] AGHAJAN, Hamid; CAVALLARO, Andrea (ed.). Multi-camera networks: principles and applications. Academic press, 2009.

[2] SHOUP, Donald C. Cruising for parking. *Transport Policy*, 2006, 13.6: 479-486.

[3] *SFpark* [online], San Francisco [cit. 2017-05-01]. Dostupné z: http://sfpark.org/

[4] DE ALMEIDA, Paulo RL, et al. PKLot – A robust dataset for parking lot classification. *Expert Systems with Applications*, 2015, 42.11: 4937-4949.

[5] KOCOUREK, Josef. 2016. *Provoz a projektování místních komunikací (PPMK):* Doprava v klidu. [cit. 2017-04-29].

[6] PŘIBYL, Ondřej. 2015. *Měření a zpracování dat (MDS):* Neintrusivní dopravní detektory [Přednáška]. [cit. 2015-04-10].

[7] PŘIBYL, Ondřej. 2015. *Měření a zpracování dat (MDS):* Detektory zasahující do vozovky, úvod do detekce [Přednáška]. [cit. 2015-04-10].

[8] BONG, D. B. L.; TING, K. C.; LAI, K. C. Integrated approach in the design of car park occupancy information system (COINS). *IAENG International Journal of Computer Science*, 2008, 35.1: 7 - 14.

[9] BUNTIĆ, Mario; IVANJKO, Edouard; GOLD, Hrvoje. ITS supported parking lot management. In: International Conference on Traffic and Transport Engineering-Belgrade. Belgrade, Serbia. 2012.

[10] WOLFF, Joerg, et al. Parking monitor system based on magnetic field sensor. In: *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 2006. p. 1275-1279.

[11] ABAS, Kevin; PORTO, Caio; OBRACZKA, Katia. Wireless smart camera networks for the surveillance of public spaces. *Computer*, 2014, 47.5: 37-44.

[12] AMATO, Giuseppe, et al. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 2017, 72: 327-334.

[13] BRADSKI, Gary a Adrian KAEHLER. 2008. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol: O'Reilly, xvii, 555 s. ISBN 978-0-596-51613-0.

[14] FAWCETT, Tom. An introduction to ROC analysis. *Pattern recognition letters*, 2006, 27.8: 861-874.

[15] BEHNKE, Sven. 2003. *Hierarchical neural networks for image interpretation*. New York: Springer, xii, 224 p. ISBN 35-404-0722-7.

[16] MELNIČUK, Petr. 2015. *Automatická detekce a identifikace registračních značek vozidel*. Praha. Bakalářská práce.

[17] HASEGAWA, Tameharii; NOHSOH, K.; OZAWA, S. Counting cars by tracking moving objects in the outdoor parking lot. In: *Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994*. IEEE, 1994. p. 63-68.

[18] BHASKAR, Harish; WERGHI, Naoufel; AL-MANSOORI, Saeed. Rectangular Empty Parking Space Detection using SIFT based Classification. In: *VISAPP*. 2011. p. 214-220.

[19] DELIBALTOV, Diana, Wencheng WU, Robert P. LOCE a Edgar A. BERNAL. Parking lot occupancy determination from lamp-post camera images. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, [online]. IEEE, 2013, s. 2387-2392

[20] CAZAMIAS, Jordan; MAREK, Martina. Parking Space Classification using Convolutional Neural Networks.

[21] HSU, Chihping; Tanaka, T.; Sugie, N.; Ueda K. et al. Locating vehicles in a parking lot by image processing. In: *IAPR workshop on machine vision applications*. 2002. p. 475-478.

[22] DEL POSTIGO, Carlos Gálvez; TORRES, Juan; MENÉNDEZ, José Manuel. Vacant parking area estimation through background subtraction and transience map analysis. *IET Intelligent Transport Systems*, 2015, 9.9: 835-841.

[23] TSCHENTSCHER, Marc; NEUHAUSEN, Marcel. Video-based parking space detection. In: *Proceedings of the Forum Bauinformatik*. 2012. p. 159-166.

[24] KŁOSOWSKI, M.; WÓJCIKOWSKI, M.; CZYŻEWSKI, A. Vision-based parking lot occupancy evaluation system using 2D separable discrete wavelet transform. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 2015, 63.3: 569-573.

[25] BULAN, Orhan, et al. Video-based real-time on-street parking occupancy detection system. Journal of Electronic Imaging, 2013, 22.4: 041109-041109.

[26] RYBSKI, Paul E., et al. Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010. p. 921-928.

[27] BAROFFIO, Luca, et al. A visual sensor network for parking lot occupancy detection in smart cities. In: Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, 2015. p. 745-750.

[28] MOLANO, José Ignacio Rodríguez; BETANCOURT, Diana; GÓMEZ, Germán. Internet of things: A prototype architecture using a Raspberry Pi. In: *International Conference on Knowledge Management in Organizations*. Springer International Publishing, 2015. p. 618-631.

[29] UPTON, Eben. a Gareth. HALFACREE. *Raspberry Pi user guide*. Chichester, England: John Wiley, c2012. ISBN 111846446X.

[30] *Raspberry Pi web* [online]. [cit. 2017-02-11]. Dostupné z: https://www.raspberrypi.org/

[31] *RPi hardware* [online]. [cit. 2017-02-11]. Dostupné z: http://elinux.org/RPi_Hardware

[32] *Raspberry Pi* [online]. Dostupné z: https://en.wikipedia.org/wiki/Raspberry_Pi

[33] *The MagPi magazine: Raspberry Pi 3 benchmarks* [online]. [cit. 2017-02-11]. Dostupné z: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

[34] *Raspberry Pi camera modules* [online]. [cit. 2017-02-11]. Dostupné z: http://elinux.org/Rpi_Camera_Module

[35] *Raspberry Pi CSI Camera Interface* [online]. [cit. 2017-02-11]. Dostupné z: https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI_Camera_Interface.html

[36] Camera interface specification*: technology brief* [online]. [cit. 2017-02-11]. Dostupné z: http://mipi.org/specifications/camera-interface

[37] *RPi-Cam-Web-Interface* [online]. [cit. 2017-02-12]. Dostupné z: http://elinux.org/RPi-Cam-Web-Interface

[38] *Picamera documentation* [online]. [cit. 2017-02-11]. Dostupné z: http://picamera.readthedocs.io/en/release-1.12/

[39] *Raspberry Pi Single-Board Computer & Raspberry Pi Camera Module* [online]. [cit. 2017-02-11]. Dostupné z: http://www.truetex.com/raspberrypi

[40] DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005. p. 886-893.

[41] WANG, Youlu; VELIPASALAR, Senem; GURSOY, Mustafa Cenk. Distributed wide-area multi-object tracking with non-overlapping camera views. *Multimedia tools and applications*, 2014, 73.1: 7-39.

[42] *Learn OpenCV: Histogram of Oriented Gradients* [online], [cit. 2017-05-13]. Dostupné z: https://www.learnopencv.com/histogram-of-oriented-gradients/

[43] *Support vector machine* [online], [cit. 2017-05-14]. Dostupné z: https://en.wikipedia.org/wiki/Support_vector_machine

[44] *Introduction to Support Vector Machines* [online], [cit. 2017-05-14]. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Informace k příloze

Součástí této práce je datová příloha obsahující především zdrojové kódy implementovaných algoritmů a video ukázky pro demonstraci jejich funkce. O adresářové struktuře a předávaných souborech dále podrobněji informuje soubor *ctete.pdf*.