VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMOBILNÍHO A DOPRAVNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMOTIVE ENGINEERING

AUTONOMNÍ ŘÍZENÍ VOZIDLA POMOCÍ ZPRACOVÁNÍ OBRAZU

AUTONOMOUS CONTROL OF THE VEHICLE THROUGH IMAGE PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Leoš Fronc

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Pavel Kučera, Ph.D.

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automobilního a dopravního inženýrství
Student:	**Bc. Leoš Fronc**
Studijní program:	Automobilní a dopravní inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**doc. Ing. Pavel Kučera, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Autonomní řízení vozidla pomocí zpracování obrazu

Stručná charakteristika problematiky úkolu:

Naprogramování řídicího algoritmu pro zpracování obrazu z kamer pro autonomní řízení vozidla využitím vhodného softwaru např. jazyků C, C++, atd. Dále naprogramování komunikace mezi hardwarem např. typu Jetson Nano ve spojení se stereo kamerou ZED. Následné zpracovávání ověřit na použitém zařízení.

Cíle diplomové práce:

Vypracovat rešerši ohledně zpracovávání obrazu v reálném čase.
Vypracovat naprogramování algoritmu pro zpracování obrazu a detekci jízdních pruhů.
Naprogramovat komunikaci mezi hardwarem a kamerou.
Otestovat kompletní systém z hlediska funkčnosti a rozpoznávání jízdních pruhů.

Seznam doporučené literatury:

HANSEN, John H. L. Digital signal processing for in-vehicle systems and safety. 1. London: Springer, 2012. ISBN 978-1441996060.

SOLOMON, Chris a Toby BRECKON. Fundamentals of digital image processing: a practical approach with examples in Matlab. 1. Hoboken, NJ: Wiley-Blackwell, 2011. ISBN 978-0470844731.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

prof. Ing. Josef Štětina, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá tématem autonomních vozidel a především detekce jízdních pruhů. V práci jsou popsány a porovnány dva hlavní přístupy pro detekci jízdních pruhů – pomocí tradičních metod počítačového vidění a pomocí konvolučních neuronových sítí. Cílem práce bylo vytvořit systém, který by byl schopný rozpoznávat jízdní pruhy v reálném čase. Navržený systém byl sestaven z počítače Jetson Nano, ze stereo kamery ZED a z naprogramovaného algoritmu. Celkem byly vytvořeny dva algoritmy, které využívají zcela odlišných přístupů. Závěrem byl celý systém otestován z hlediska funkčnosti a schopnosti rozpoznávání jízdních pruhů.

KLÍČOVÁ SLOVA

Autonomní řízení vozidla, detekce jízdních pruhů, zpracování obrazu, počítačové vidění, konvoluční neuronové sítě, Jetson Nano

ABSTRACT

This diploma thesis deals with the topic of autonomous vehicles and especially lane detection. The paper describes and compares two main approaches to the lane detection - using traditional methods of computer vision and convolutional neural networks. The aim of the work was to create a system that would be able to recognize road lanes in a real time. The proposed system consisted of a Jetson Nano computer, a ZED stereo camera and a programmed algorithm. In total, two algorithms have been developed that use completely different approaches. Finally, the whole system was tested in terms of functionality and lane recognition.

KEYWORDS

Autonomous vehicle control, lane detection, image processing, computer vision, convolutional neural networks, Jetson Nano

BIBLIOGRAFICKÁ CITACE

FRONC, Leoš. Autonomní řízení vozidla pomocí zpracování obrazu [online]. Brno, 2022 [cit. 2022-05-19]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/136950. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automobilního a dopravního inženýrství. Vedoucí práce Pavel Kučera.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Pavla Kučery Ph.D. a s použitím informačních zdrojů uvedených v seznamu.

V Brně dne 20. května 2022 ……..……..

Leoš Fronc

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu diplomové práce doc. Ing. Pavlu Kučerovi Ph.D. za praktické rady a připomínky při psaní práce. Také bych rád poděkoval rodičům a přátelům za podporu během celého studia.

OBSAH

ÚVOD

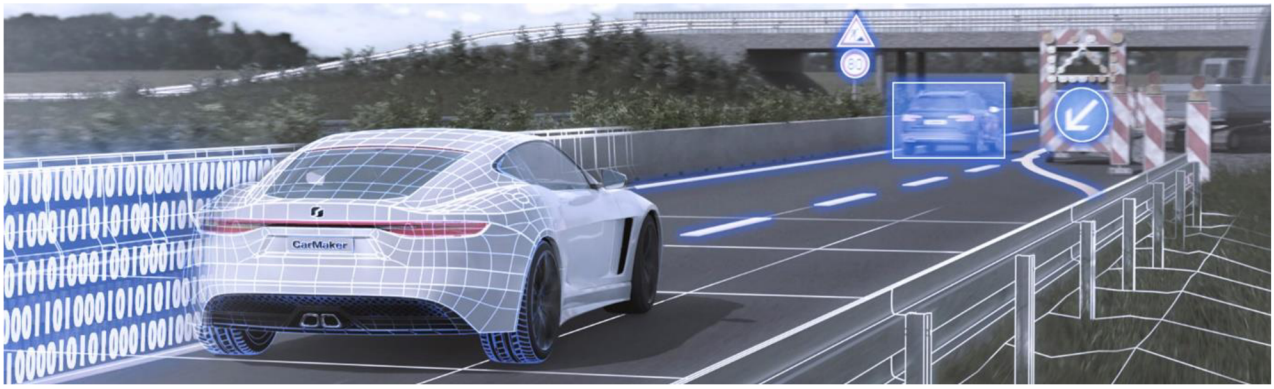
Vývoj autonomních systémů pro řízení silničních vozidel je snahou většiny automobilových výrobců. V současnosti již existuje řada komerčních systémů pro částečnou autonomizaci řízení. S vývojem těchto systémů souvisí i značné finanční náklady, proto bývá většina podstatných informací utajována. Stále se zvyšující výpočetní výkon umožňuje širší aplikaci autonomních systémů. Jejich základním požadavkem je běžet v reálném čase, proto jsou citlivé na rychlost zpracování.

Autonomní řízení je velmi komplexní problém, který bývá rozdělen na mnoho podoblastí. Alespoň částečná automatizace řízení má potenciál zvýšit komfort řidiče a především zvýšit bezpečnost dopravy. Plně autonomní vozidla mají potenciál zcela změnit způsob dopravy, tak jak je běžný v dnešní době. Postupný vývoj výpočetní techniky i programového vybavení přináší nové přístupy v oblasti autonomního řízení. Důležitou součástí autonomních systémů je získávání informací o okolí vozidla vhodnými senzory nebo v poslední době stále více rozšířenou komunikací automobilu s ostatními vozidly nebo infrastrukturou.

Cílem diplomové práce je vypracovat rešerši ohledně zpracování obrazu v reálném čase. V rámci teoretické části diplomové práce byl kladen důraz především na oblasti výzkumu související s detekcí jízdních pruhů. Na základě znalostí z rešerše byl vytvořen systém pro detekci jízdních pruhů. Navržený systém se skládá ze stereo kamery ZED ve spojení s vestavěným počítačem Jetson Nano, na kterém běží navržený algoritmus. Dále byly vytvořeny dva algoritmy, které uplatňují zcela odlišné přístupy. První algoritmus využívá tradiční metody počítačového vidění. Druhý algoritmus využívá konvolučních neuronových sítí a následného zpracování. Závěrem byl celý systém ověřen z hlediska funkčnosti v reálném provozu a jednotlivé algoritmy byly srovnány z hlediska charakteristických vlastností.

1 PŘEHLED AUTONOMNÍCH SYSTÉMŮ

V posledních letech bylo nejvíce nehod zaviněno chybou řidiče. Plně autonomní vozidlo je schopno vykonávat běžné jízdní úkony bez zásahu řidiče. Tím by bylo možné lidské chyby odstranit nebo alespoň minimalizovat následky špatných reakcí a rozhodnutí řidiče [1]. Také je patrný trend zvyšování komfortu řidiče a cestujících, kdy se asistenční systémy přesouvají z prémiových výbav do nižších až základních specifikací vozidla. Dnešní doba též vybízí k rozšiřování carsharingu a vytváření „sdílených automobilů“ v rámci snižování uhlíkové stopy a zlepšení ekologického dopadu automobilů. Tyto faktory a mnoho dalších souvisejících hlavně s rychlým vývojem výpočetní techniky, počítačového vidění a komunikačních sítí přispívají k rozvoji autonomních automobilů.



Obr. 1 Představa autonomního řízení [2]

Problematika autonomních vozidel je již dlouhou dobu známa a je předmětem vývoje nejenom předních světových automobilek, ale také technologických a dopravních firem, jako je Google či Uber. Ty se snaží získat technologický předstih, a proto bývá většina informací souvisejících s vývojem těchto systémů utajována.

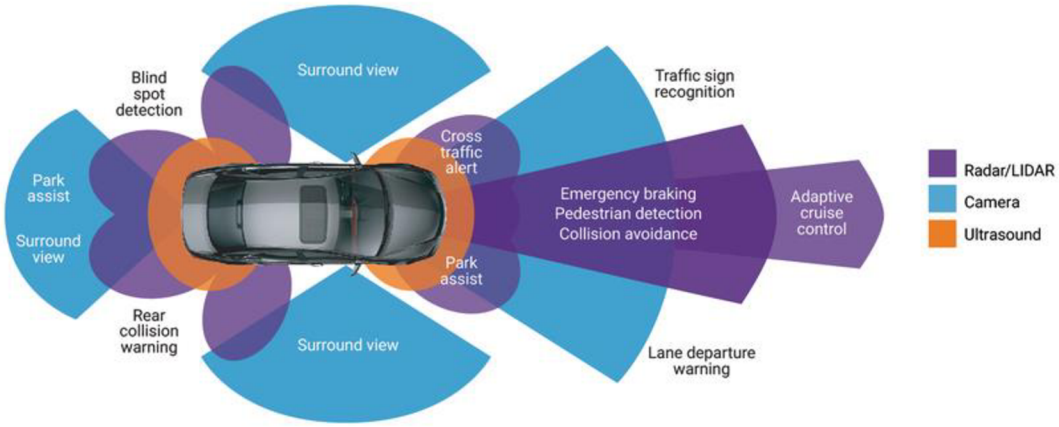
1.1 DĚLENÍ AUTOMATIZACE

Podle asociace automobilového průmyslu SAE International [3] v roce 2014 je definováno 5 úrovní automatizace, které se liší mírou asistence.

0. Bez žádné automatizace. Systém pouze varuje a upozorňuje řidiče, ale neovládá vůz.
1. Asistence řidiče („hands on“). Automaticky mohou probíhat složitější funkce (adaptivní tempomat nebo udržování jízdních pruhů), řidič musí být schopen kdykoli řídit.
2. Částečná automatizace („hands off“). Automat řídí, zrychluje i brzdí, (adaptivní tempomat a udržování jízdních pruhů), řidič musí být schopen kdykoli řídit.
3. Podmíněná automatizace („eyes off“). V definovaném prostředí se řidič nemusí věnovat řízení (jízda v dopravní zácpě, jízda na dálnici), řidič musí být připraven převzít řízení v časovém limitu, který stanoví výrobce.
4. Vysoká automatizace („mind off“). S výjimkou vysoce nebezpečného prostředí (nepříznivé počasí) řídí automat a řidič nezasahuje.
5. Plná automatizace („řízení volitelné“). Automat řídí do libovolného legálního cíle, řidič jen zadá cíl.

1.2 DETEKOVÁNÍ OKOLÍ AUTONOMNÍMI VOZIDLY

Základním principem fungování autonomních vozidel je detekování jejich okolí [4], především detekování jízdních pruhů, okolních vozidel a dalších účastníků silničního provozu, různých překážek na vozovce a v její blízkosti. Následné inteligentní zpracování je ovlivněno kvalitou a přesností snímaných dat. Pro detekci okolí vozidla se využívá několik přístupů, které lze rozdělit z hlediska použitých senzorů. Na *obr. 2* jsou znázorněny oblasti, které pokrývají jednotlivé senzory. Ke každé oblasti je zobrazen asistenční systém, který data z dané oblasti zpracovává. V následujících podkapitolách jsou popsány jednotlivé senzory, které autonomní vozidla využívají.



Obr. 2 Rozložení senzorů u automobilu [5]

KAMERA

Kamera je běžně dostupným snímačem viditelného spektra světla. Řadí se mezi pasivní snímače, protože pouze snímá světlo z okolí a žádné nevysílá. Můžeme je rozdělit na mono (standardní) kamery a stereo kamery, které mají přesně definovanou vzdálenost mezi objektivy a lze tak vypočítat vzdálenost snímaných objektů. Hlavní funkční výhodou kamer je, že na získaném obraze je možné rozeznávat barvy a struktury. Tato schopnost umožňuje detekovat a identifikovat dopravní značky, semafory, značení jízdních pruhů aj. Další výhoda je jejich relativně nízká pořizovací cena, dobrá dostupnost a malé rozměry. Nevýhodou je náchylnost na vlivy počasí a zhoršenou funkci v noci.

Obr. 3 Kamera od společnosti StereoLabs [6]

LIDAR

Lidar (Light Detection and Ranging) je zařízení používané k měření vzdálenosti pomocí laseru. Lidar je aktivním senzorem, který vysílá laserové paprsky a snímá jejich odrazy. Na základě doby letu jednotlivých paprsků (TOF time of flight) je vyhodnocená vzdálenost objektů od Lidaru. Lidar měří vzdálenosti bodově, frekvence měření jednotlivých bodů je až 150 kHz. Výsledkem je bodová 3D reprezentace celého okolí Lidaru. Dosah senzoru může být více než 250 metrů. Za normálních podmínek nabízí lepší přesnost než Radar. Senzor je citlivý na částice ve vzduchu (prach, déšť, sníh), a může tak docházet ke špatnému určení vzdálenosti objektu. Nevýhodou je vysoká cena a menší dostupnost na trhu. Na *obr. 4* je vidět bodová reprezentace okolí Lidaru.

Obr. 4 Výstupem Lidar senzoru je mračno bodů [7]

RADAR

Radar (Radio Detection and Ranging) je podobně jako Lidar aktivním snímačem. Funguje na principu vysílání impulsů rádiových vln a snímání jejich odrazu. Na základě doby letu jednotlivých impulsů je následně vypočítána vzdálenost objektu. Při použití stálé vlny, namísto impulsu, může být využito Dopplerova efektu k určení relativní rychlosti objektu od senzoru. Radary se liší svojí frekvencí nejčastěji 24 GHz, 77 GHz, 79 GHz. Obecně radary s vyšší frekvencí jsou přesnější a schopné rozeznávat více objektů. Dosah radarů se pohybuje okolo 50 až 150 metrů. Oproti Lidaru jsou radary výrazně levnější, dostupnější a jsou také odolné na změny počasí a prostředí. Nejčastěji bývají používány pro detekování objektů před senzorem a určování jejich rychlosti.

Obr. 5 Radary s různým dosahem [8]

ULTRAZVUKOVÝ SNÍMAČ

Ultrazvukový snímač je aktivní senzor, který funguje na principu vysílání ultrazvukového vlnění a následném měření doby letu těchto vln. Používá se k velmi přesnému detekování objektů na krátké vzdálenosti. V automobilovém průmyslu jsou ultrazvukové snímače používány jako parkovací senzory již řadu let. Hlavní výhodou je nízká cena a malé rozměry. Nevýhodou těchto snímačů je však fakt, že mohou být silně ovlivněny poruchami zvukových vln. Vzhledem k tomu, že rychlost zvuku se liší podle média, kterým prostupuje, dochází tedy k ovlivnění přesnosti při změně prostředí. Především se jedná o změnu teploty a vlhkosti. Kvůli tomu většina senzorů upravuje svoje parametry v závislosti na teplotě. Díky této korekci jsou poměrně odolné vůči změnám počasí.

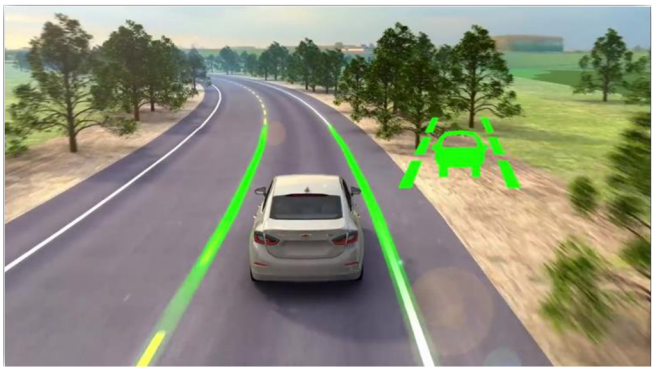
Obr. 6 Ultrazvukové snímače od firmy Bosch [8]

GPS – GLOBÁLNÍ LOKALIZAČNÍ SYSTÉM

GPS (Global Positioning System) umožňuje určit přesnou polohu snímače. Jedná se o pasivní snímač, který přijímá informace o poloze a času jednotlivých satelitů. Z této informace je možné vypočítat vzdálenost přijímače od satelitu. Pokud snímač má k dispozici informace od více (aspoň 3) satelitů, je možné vypočítat přesnou polohu snímače. Přesnost GPS klesá, jestliže je dráha mezi satelitem a snímačem narušena. Nejčastěji je signál narušen stromy, terénem nebo v městském provozu výškovými budovami, které kromě blokování způsobují i odraz signálu od oken či skleněných stěn. Navíc je zcela nemožný provoz v tunelech nebo vnitřních prostorech.

1.3 DETEKTOR JÍZDNÍCH PRUHŮ

Autonomní řízení vozidla je velmi komplexní úkol, proto je rozdělen na více dílčích částí. Základním předpokladem pro provoz samořídících automobilů je rozvoj v automatické detekci jízdních pruhů. Pokročilé systémy pro detekci jízdních pruhů jsou schopny detekovat jízdní pruhy na základě zpracování obrazu a vypočítat relativní polohu vozidla vůči detekovanému jízdního pruhu. V případě, že vozidlo vybočí z jízdního pruhu, je takovýto systém schopný varovat řidiče zvukovým, vizuálním, popřípadě haptickým signálem, nebo dokonce zasáhnout do řízení a upravit trajektorii jízdy. Jelikož většina systémů pro detekci jízdních pruhů využívá kamer, tak výsledná kvalita a samotná funkčnost detektoru závisí na mnoha faktorech. Především na dobrém stavu vozovky a viditelnosti jízdních pruhů samotných a dále na vlivech počasí (sníh, stíny, déšť, atd.).



Obr. 7 Ilustrace detekování jízdních pruhů [9]

1.3.1 ROZDĚLENÍ Z HLEDISKA FUNKCE

Následující část bude věnována základnímu rozdělení systémů pro detekci jízdních pruhů. Pro detekci pomocí zpracování obrazu existuje několik přístupů [10]. Tyto přístupy se liší přesností detekce, rychlostí a komplexností výpočtu. V současnosti se uvádí tři skupiny přístupů. Nejedná se o tři oddělené skupiny, ale jednotlivé algoritmy mohou využívat i více přístupů.

PŘÍSTUP ZALOŽENÝ NA DETEKCI VÝZNAMNÝCH BODŮ

Přístup založený na detekci významných bodů je nejzákladnějším přístupem detekce jízdních pruhů. Tento přístup využívá lokální charakteristiky obrazu, jako je gradient jasu, barvy a textury obrazu. Lokální přístup je obecně schopný detekovat jízdní pruhy neobvyklých tvarů, avšak je citlivý na špatné značení jízdního pruhu, na změnu osvětlení vozovky a především na stíny a odlesky.

PŘÍSTUP ZALOŽENÝ NA MODELU

Přístup založený na modelu se snaží adaptovat globální geometrický model jízdních pruhů na lokální charakteristiky obrazu. Obecně platí, že přístup založený na modelu je odolnější vůči lokálním změnám osvětlení nebo špatnému vodorovnému značení. Takovýto přístup bývá většinou vytvořen pro množinu scén s podobnými rysy, a proto takto vytvořené metody bývají méně přizpůsobivé podmínkám, pro které nebyly konstruovány. Navíc výpočet nejlepších parametrů modelu je časově náročný.

PŘÍSTUP ZALOŽENÝ NA KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

Tento přístup založený na konvolučních neuronových sítí (CNN) se vyvinul jako poslední z výše jmenovaných a v současnosti se stává stále více populární a postupně vytlačuje metody tradičního počítačového vidění.

Existující metody detekce jízdních pruhů pomocí CNN je možné dále rozdělit na jednofázové a dvoufázové. Dvoufázové metody se skládají z detekce základních rysů (příznaků) obrazu a následného zpracování (postprocessing) nalezených rysů. Pro nalezení příznaků se právě využívá konvolučních sítí založených na hlubokém učení. Teprve následným zpracováním jsou určeny jízdní pruhy. Jednokrokové metody mohou získat výsledky detekce přímo ze vstupního obrazu a není potřeba postprocesingu.

2 ZPRACOVÁNÍ OBRAZU V REÁLNÉM ČASE

Systémy reálného času (RT – real time) jsou takové systémy, jejichž korektnost závisí nejen na správném výsledku, ale také na časovém intervalu, za který měl být výsledek vypočítán. Jelikož prostředí, a tím i vstupní parametry RT systému, se neustále mění, zaručuje podmínka pravidelných časových intervalů mezi výpočty to, že výsledek bude vždy dostupný ze správných vstupních dat a ve správný okamžik. Většina RT systémů vykonává výpočty v nekonečném cyklu s explicitně určenou dobou odezvy. Tím se RT systém liší od „rychlého" systému, u kterého délka běhu hlavního cyklu je sice krátká, ale není předvídatelná. Podle možnosti nedodržení časového intervalu (deadline) lze RT systémy rozdělit na:

Soft RT systém – u tohoto systému je možné nedodržení časového limitu. Tento stav je však nežádoucí z výkonnostních důvodů a užitečnost výsledku se po termínu jeho dodání snižuje. Soft RT systém se např. využívá u videoher, či jiných multimediálních aplikací.

Hard RT systém – u takového systému při nedodržení časového intervalu dojde k totálnímu zhroucení systému. Využití hard RT systému je např. u detekce kritických podmínek, řízení jaderné elektrárny atd., kde musí být dodržena kritická bezpečnost.

Firm RT systém – je systém, u kterého je možné nedodržení několika málo časových intervalů, aniž by se systém zhroutil. Užitečnost výsledku je však v takovém případě nulová.

2.1 ZÁKLADY POČÍTAČOVÉHO VIDĚNÍ

Ze zadání diplomové práce vyplývá, že k detekci okolí vozidla bude použito zpracování obrazu. K získávání informací z obrazových dat slouží obor počítačového vidění (CV – computer vision). Počítačové vidění je rozsáhlý obor, který zahrnuje oblasti získávání obrazu a jeho následné úpravy za účelem získávání informací [11]. Během posledních několika let se počítačové vidění rozrostlo z oblasti výzkumu do praxe, kde poskytuje nárůst produktivity i zlepšení životní úrovně. Výkonnější a zároveň levnější výpočetní technika umožňuje nasazení počítačového vidění u stále více aplikací.

Počítačové vidění zahrnuje velké množství nástrojů a metod. Jejich použití závisí na požadovaném výstupu. Jednotlivé metody zpracování obrazu se liší svojí komplexností od primitivních filtrů až po metody, které získávají komplexní informace o snímané scéně. Značná část algoritmů pro detekci jízdních pruhů stále používá tradiční přístup na základě metod počítačového vidění. Nyní budou představeny vybrané metody, které těmito algoritmy bývají využívány.

2.1.1 DETEKOVÁNÍ HRAN

Hrana v obraze je jasně matematicky definovatelné místo a je tedy možná její opakovatelná detekce. Nejčastěji detektory hran fungují na principu porovnávání hodnot jasu jednotlivých pixelů v blízkém okolí vyšetřovaného bodu. V současnosti existuje velké množství detektorů hran. Tyto detektory se liší ve způsobu detekce, a proto vykazují různé vlastnosti. Kromě kvality určení pozice hrany se u detektorů hodnotí především jejich odolnosti vůči obrazovým změnám a výpočetní náročnosti, která nepřímo ovlivňuje rychlost výpočtu. V rámci detekování jízdních pruhů u většiny algoritmů převažuje detektor hran Canny (Canny edge detector).

DETEKTOR HRAN CANNY

Canny je komplexní detektor hran v obraze [12]. Canny se vyznačuje nízkou chybovostí, dobrou lokalizací (vzdálenost mezi detekovanými a skutečnými okrajovými pixely je minimální) a minimální odezvou (každá hrana má pouze jednu odezvu). Jeho funkce se skládá ze čtyř kroků v daném pořadí:

1. Vyhlazení obrazu
2. Určení gradientu
3. Nalezení lokálních maxim
4. Odstranění nevýznamných hran

Prvním krokem Cannyho detektoru je vyhlazení obrazu pomocí vhodného filtru. Nejčastěji bývá použito Gaussova filtru. Rovnice (1) představuje konvoluční jádro Gaussova filtru o velikosti 3×3. Díky vyhlazení je z obrazu odstraněn šum, který do značné míry může ovlivnit samotnou detekci.

$$B = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \tag{1}$$

Druhým krokem je určení gradientu jasu pro každý pixel. Nejčastěji bývá použit Sobelův operátor, který je schopen pomocí konvoluce zjistit hodnoty gradientu jasu v blízkém okolí zkoumaného bodu. Detektor Canny prověřuje gradienty jasu celkem ve 4 směrech (0°, 45°, 90°, 135°). Absolutní hodnota gradientu je následně dána součtem druhých mocnin gradientů v jednotlivých směrech. Rovnice (2) představují konvoluční jádra Sobelova operátoru v přímém směru osy x a ve směru pootočeném o 45°.

$$G_x = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix};\ G_{xy} = \begin{bmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{bmatrix} \tag{2}$$

Třetím krokem je nalezení lokálních maxim gradientů. Dojde tedy k potlačení hodnot, které maximem nejsou. Algoritmus porovnává velikost gradientu aktuálního pixelu s velikostí gradientu sousedícího pixelu v kladném a záporném směru. Pokud je velikost gradientu aktuálního pixelu největší ve srovnání s okolními pixely v daném směru, hodnota zůstane zachována. V opačném případě bude hodnota potlačena. Touto úpravou dojde k jednoznačnému určení pixelů, které odpovídají hraně.

Posledním krokem je odstranění nevýznamných hran. K tomu jsou použity dvě prahové hodnoty (horní a dolní). Pokud je gradient pixelu vyšší než horní práh, pixel je akceptován jako hrana. Je-li hodnota gradientu pixelu pod spodní prahovou hodnotou, je hrana zamítnuta. Jestliže je gradient pixelů mezi dvěma prahovými hodnotami, bude akceptován pouze v případě, že je připojen k pixelu, který jako hrana již byl akceptován.

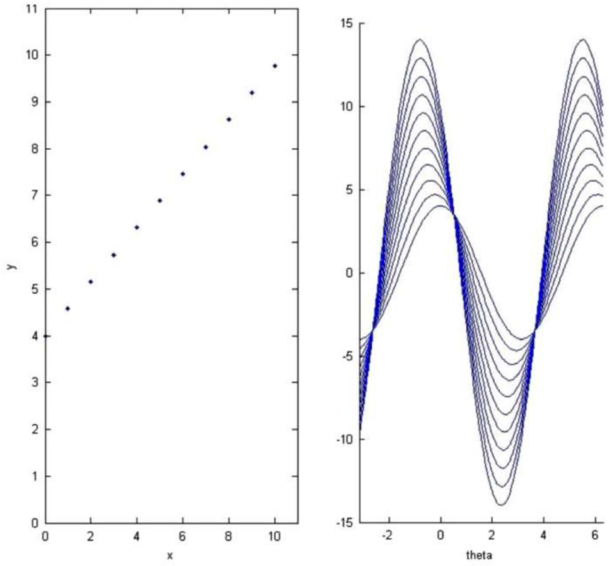
2.1.2 HOUGHOVA TRANSFORMACE

Houghova transformace je další velmi rozšířená metoda počítačového vidění používaná při detekci jízdních pruhů. Tato metoda je určená pro hledání geometrických tvarů, které mohou být definovány v parametrické formě [13]. Pro detekci jízdních pruhů se nejčastěji používá Houghova transformace, která hledá parametrické vyjádření přímky odpovídající vodorovnému značení jízdního pruhu. Algoritmus lineární Houghovy transformace odhaduje dva parametry,

které definují přímku. Obecně lze přímku v souřadnicovém systému (x, y) parametrizovat několika způsoby. V praxi se pro popis přímky (p) používá rovnice:

$$r = x\cos(\theta) + y\sin(\theta) \tag{3}$$

Kde r je kolmá vzdálenost od přímky k počátku souřadnic, théta je úhel mezi normálou přímky a osou x. V tomto případě je jeden bod v souřadnicovém systému (x, y) reprezentován křivkou odpovídající sinusoidě souřadnicovém systému (r, θ) viz obr. 8. Tato křivka odpovídá všem možným řešením rovnice (3) po dosazení daného bodu. Pokud se takto převedou všechny body ležící na přímce (p) v souřadnicovém systému (r, θ), vzniknou křivky, které se protínají v jednom bodě. Souřadnice tohoto bodu jsou hledanými parametry přímky (p).



Obr. 8 Převod jednotlivých bodů ze souřadnicového systému (x, y) na sinusoidy v souřadnicovém systému (r, θ) pomocí houghovy transformace [13]

2.2 UMĚLÉ NEURONOVÉ SÍTĚ

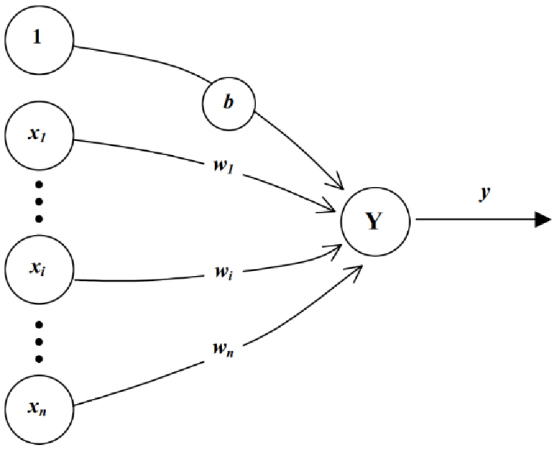
Vedle oboru počítačového vidění existuje disciplína hlubokého učení neuronových sítí (deep learning), jedná se o podobor strojového učení (machine learning). Tento výpočetní model se inspiruje v chování některých přírodních struktur. Jejich archtektura umožňuje uplatnění v široké škále aplikací.

2.2.1 STRUKTURA NEURONOVÉ SÍTĚ

Základní struktura neuronové sítě (neural network) je neuron [14] viz *obr. 9*. Neuron může mít libovolný počet vstupů, které jsou násobeny příslušnými váhami. Výpočet vnitřního potenciálu neuronu znázorňuje rovnice (4).

$$y = b + \sum_{i=1}^{n} w_i x_i \tag{4}$$

Kde w_i je váha neuronu, x_i je vstupní signál, b je práh (bias) a y představuje vnitřní potenciál neuronu, tato hodnota je dále upravena aktivační funkcí.



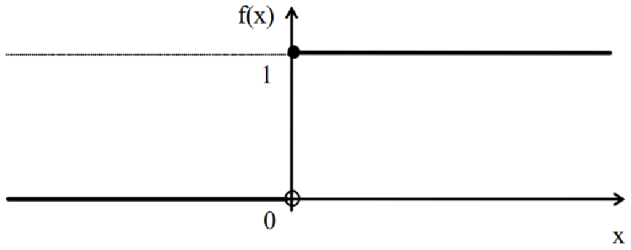
Obr. 9 Neuron se vstupními hodnotami, váhami a prahem [15]

Výstupní hodnota neuronu je dále transformována aktivační funkcí, ty mohou mít různé tvary a vlastnosti. Příklady nejčastěji používaných aktivačních funkcí jsou následující:

SKOKOVÁ FUNKCE

Tato aktivační funkce je nespojitá a binární. Pro hodnotu vstupu menšího než je určená mez (v daném případě 0) vrací nulu; a pro hodnoty větší než je daná mez, vrací jedna.

$$f(x) = \begin{cases} 1 & pokud \quad x \geq 0 \\ 0 & pokud \quad x < 0 \end{cases} \tag{5}$$

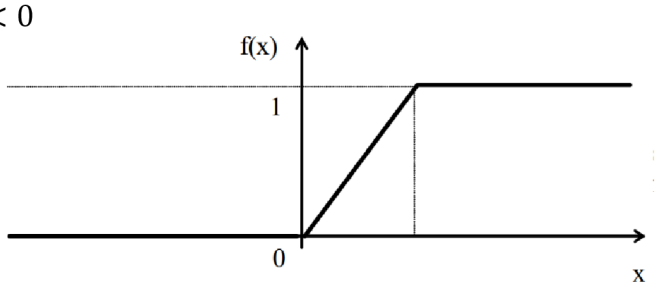


Obr. 10 Skoková aktivační funkce [15]

SATUROVANÁ LINEÁRNÍ FUNKCE.

Výstupem této funkce je v oblasti od nuly do jedné její vstupní hodnota. Pro vstupní hodnoty menší než nula je výstupní hodnota rovna nule. V případě, že je vstupní hodnota větší než jedna, je výstupem jedna.

$$f(x) = \begin{cases} 1 & x \geq 1 \\ x & 0 \leq x \leq 1 \\ 0 & x < 0 \end{cases} \tag{6}$$

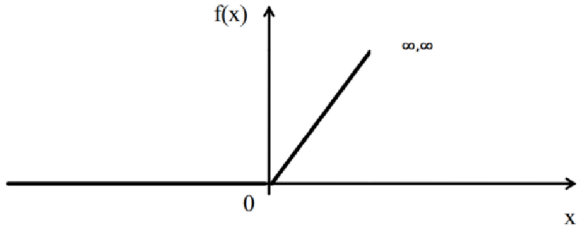


Obr. 11 Saturovaná lineární aktivační funkce [15]

REKTIFIKOVANÁ LINEÁRNÍ FUNKCE (RELU - RECTIFIED LINEAR UNIT).

Stejně jako v případě saturované lineární funkce je pro vstupní hodnoty menší než nula výstupem funkce nula. Pro hodnoty vstupu větší než nula je výstupem funkce její vstupní hodnota. Tato funkce je označovaná jako univerzální aktivační funkce a je popsána vztahem (7).

$$f(x) = \begin{cases} x & pokud \quad x \geq 0 \\ 0 & pokud \quad x < 0 \end{cases} \tag{7}$$

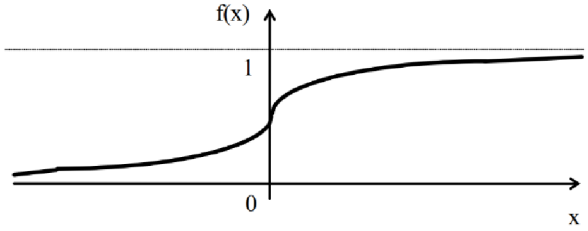


Obr. 12 Rektifikovaná lineární aktivační funkce [15]

STANDARDNÍ (LOGISTICKÁ) FUNKCE.

Tato funkce má tvar rovnice (8). Pro vstupní hodnoty blížící se k minus nekonečnu vrací hodnoty blížící se k nule a pro hodnoty blížící se k plus nekonečnu vrací hodnoty blížící se k jedné. Zároveň výstupní hodnota pro nulu je jedna polovina. Celá funkce je hladká a spojitá včetně její derivace.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

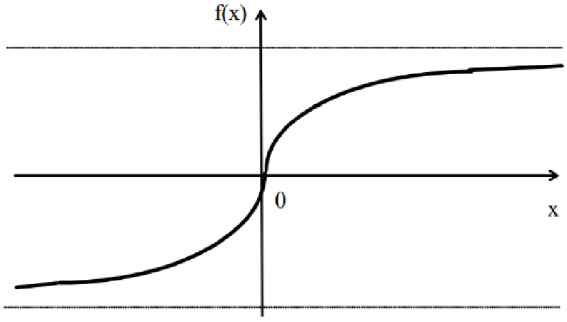


Obr. 13 Logická aktivační funkce [15]

HYPERBOLICKÝ TANGENS

Tato funkce má tvar rovnice (9). Její výstupní hodnoty se blíží k minus jedné v minus nekonečnu a k jedné v plus nekonečnu. Pro nulu je výstupní hodnota nula.

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{9}$$



Obr. 14 Aktivační funkce hyperbolický tangens [15]

Jednotlivé neurony jsou vzájemně propojeny tak, že výstup jednoho neuronu je vstupem dalšího neuronu. Každá neuronová síť má vstupní a výstupní vrstvu neuronů a mezi těmito vrstvami může být libovolný počet skrytých (pracovních) vrstev. Obecně se předpokládá, že navržená architektura je pevná a s časem se nemění. Po definování struktury neuronové sítě následuje její učení.

2.2.2 Učení neuronové sítě

Učením (trénováním) se mění váhy jednotlivých neuronů a adaptují se tak, aby výsledek co nejpřesněji odpovídal realitě [15], [16]. Existují dva hlavní způsoby učení neuronových sítí, a to učení s učitelem a učení bez učitele.

Učení s učitelem probíhá pomocí vstupních hodnot, které mají předem určené (anotované) výstupní hodnoty. Výstupní hodnoty jsou ve většině případů anotovány ručně, popřípadě s částečnou automatizací. Během trénování vstupní hodnoty postupně prochází sítí a výsledek se porovnává s předem anotovanými hodnotami. Následně je provedena korekce vah tak, aby došlo ke snížení chyby mezi vypočítanou výstupní hodnotou a předem určenou hodnotou. K úpravě vah pro snižování chyby se využívá optimalizačních algoritmů, které nehledají ideální řešení náhodně, ale na základě sestupu gradientů. Algoritmus zkoumá chybu v blízkém okolí jednotlivých vah a následně hledá místo s největším gradientem snižování chyby. Váhy jsou následně upraveny podle nalezeného gradientu. Vhodnou metodou pro efektivní stochastickou optimalizaci gradientu je Adam (Adaptive Moment Estimation, Adaptivní odhad momentu). Tato metoda vypočítává a upravuje rychlosti učení na základě změny gradientu. Metoda je jednoduchá na implementaci, vyžaduje málo paměti a rychle konverguje. Díky těmto vlastnostem je Adam vhodný pro širokou škálu oblastí strojového učení.

Učení bez učitele se liší tím, že tréninková sada neobsahuje žádné předem určené výstupní hodnoty. Ve většině případů cílem učení bez učitele bývá zjednodušení vstupních dat nebo jejich komprese. Je však nutno poznamenat, že většina úloh zaměřených na rozpoznávání vzorů v obraze zpravidla využívá učení s učitelem.

Vedle těchto dvou hlavních způsobů učení existuje kombinace učení s učitelem a bez učitele (semi-supervised learning), kde pouze část vstupních dat má určené výstupní hodnoty; a učení posilováním, kde se váhy upravují na základě odměn a trestů pomocí zpětné vazby. Tento způsob učení byl například využitý pro program AlphaGo, který řeší deskovou hru Go.

Přeučení

Hrozbou pro sítě učené s učitelem je tzv. přeučení (overfitting) [17]. Při nízkém počtu neuronů v síti platí, že schopnost vytvářet dobré výsledky je nízká, takto navržená síť v extrémních případech nedokáže nalézt hledané vzory. Nabízí se tedy možnost zvýšit počet skrytých vrstev a neuronů. Tato úvaha bohužel neplatí vždy. Důvody jsou hned dva, prvním z nich je, že nemáme neomezený výpočetní výkon a čas na trénování těchto obrovských neuronových sítí. Druhým důvodem je přeučení sítě, které nastává při dlouhém učení. Přeučení znamená, že síť upraví své parametry tak, aby co nejlépe odpovídaly trénovaným datům. Taková síť pravděpodobně nebude mít problém vytvořit správný výsledek na trénovaných datech, ale nebude schopná generalizace, tedy vytvořit správný výsledek na datech, na kterých nebyla učena. Proto data bývají rozdělena na dvě množiny. Trénovací množina slouží k učení sítě a bývá největší; testovací a validační množina slouží pro ověření, zda nedošlo k přeučení sítě a posouzení kvality modelu.

2.3 KONVOLUČNÍ NEURONOVÉ SÍTĚ

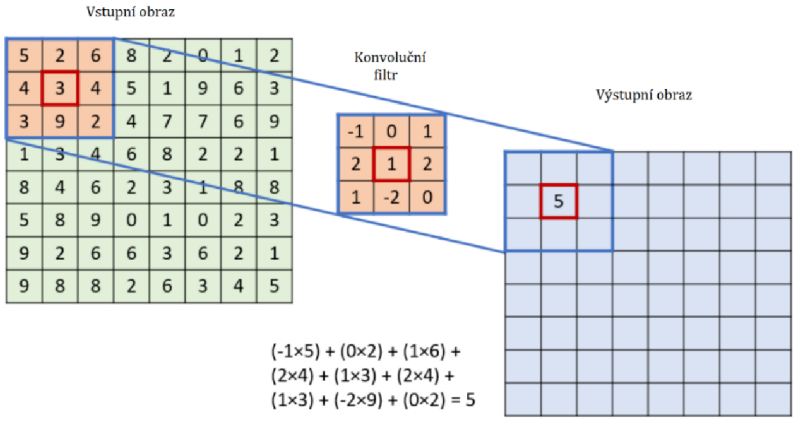
Konvoluční neuronová síť je vlastně speciální případ umělé neuronové sítě využívající konvoluce. Díky tomu se redukuje množství parametrů pro výpočet takovéto sítě a je možné efektivně zpracovávat vstupy velkých rozměrů např. obrazová data [18]. Díky tomu jsou CNN stále častěji nasazovány pro účely zpracování obrazu včetně detekce jízdních pruhů.

2.3.1 VRSTVY CNN

Konvoluční sítě se skládají z několika různých typů vrstev. Jednotlivé typy vrstev použité v praktické části budou nyní popsány [17], [18], [19].

KONVOLUČNÍ VRSTVA

Konvoluční vrstva je hlavní součástí konvoluční neuronové sítě. Cílem konvoluční vrstvy je naučit se rozpoznávání vzorů ze vstupních dat. Jednotlivé filtry (konvoluční jádra) o dané šířce a výšce několika málo pixelů (typicky 3x3 nebo 5x5) se posouvají přes vstupní obraz a počítají skalární součiny jednotlivých pixelů vstupního obrazu s hodnotami filtru, viz *obr. 15*. Každá konvoluční vrstva se skládá z několika konvolučních filtrů. Takto vytvořené 2D příznakové mapy (feature maps) mají stejné rozměry jako původní obrázek, ale obsahují již aplikované filtry.

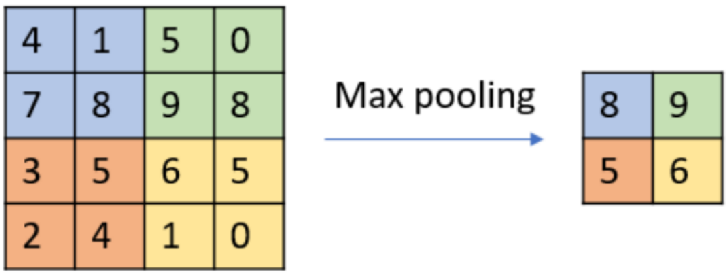


Obr. 15 Příklad konvoluce vstupního obrazu s konvolučním filtrem [20]

První konvoluční vrstvy většinou extrahují rohy, hrany, kříže a podobné jednoduché tvary. Hlubší konvoluční vrstvy následně extrahují komplexnější tvary.

POOLING VRSTVA

Pooling vrstva má za cíl postupně snižovat rozměry obrazu, a tím redukovat počet parametrů i výpočetní náročnost modelu. Tato vrstva má na vstupu jednotlivé příznakové mapy z předešlé konvoluce. Následně jsou aplikovány pooling filtry, které z celé filtrované oblasti vrací pouze 1 hodnotu. Nejčastěji se používá velikost filtru 2x2 s krokem 2 tedy bez překrytí. Tímto způsobem se redukuje 75 % vstupních dat. Existuje více typů sdružovacích operátorů, nejčastěji se jedná o max-pooling (z filtrované oblasti vybere nejvyšší hodnotu) a average-pooling (z filtrované oblasti vybere průměrnou hodnotu). *Obr. 16* zobrazuje příklad vrstvy max pooling. Jelikož během sdružování dochází ke ztrátě dat, nedoporučuje se větší velikost filtru než 3x3.



Obr. 16 Příklad redukce vstupního obrazu max pooling vrstvy [20]

DEKONVOLUČNÍ VRSTVA

Dekonvoluční vrstva nebo také transponovaná konvoluční vrstva je vlastně inverzní operace ke konvoluci. Oproti konvoluci, kde několik vstupních pixelů je pomocí konvolučního filtru spojeno do jediného výstupního pixelu, je u dekonvoluce jeden pixel vstupního obrazu násobený dekonvolučním filtrem a rozdělen do několika pixelů na výstupu. Stejně jako u konvoluce je použito více dekonvolučních filtrů.

UPSAMPLING VRSTVA

UpSampling je téměř inverzní operací pro pooling. Na rozdíl od pooling vrstvy, kde několik vstupů je redukováno na jeden výstup, je u UpSampling vrstvy jeden vstup rozšířen na několik výstupů. Při této operaci defacto nevzniká žádná nová informace. Jedná se pouze o rozšíření dimenze výstupního obrazu.

DROPOUT VRSTVA

Dropout slouží jako prevence proti přeučení. Tato vrstva je aktivována pouze při učení neuronové sítě. Její funkcí je s danou pravděpodobností vybrat některé neurony a ty během jedné iterace učení ignorovat. Jediným parametrem této vrstvy je tedy pravděpodobnost, s kterou bude neuron ignorován. Tato vrstva má za důsledek to, že síť nastavuje parametry tak, aby výsledek byl invariantní vůči změnám na vstupu.

2.3.2 KONVOLUČNÍ NEURONOVÉ SÍTĚ PRO DETEKCI JÍZDNÍCH PRUHŮ

V závislosti na principu detekce jízdního pruhu může být tato úloha definována jako klasifikace obrazu, detekce objektů nebo sémantická segmentace [21]. Konvoluční neuronové sítě jsou využívané k široké škále úloh počítačového vidění. Vzhledem k úzké vazbě mezi detekcí jízdních pruhů a základním zpracováním obrazu je vhodné představit základní funkce konvolučních neuronových sítí. Současně tato kapitola bude věnována základnímu srovnání konkrétních architektur konvolučních neuronových sítí.

KLASIFIKACE OBRAZU

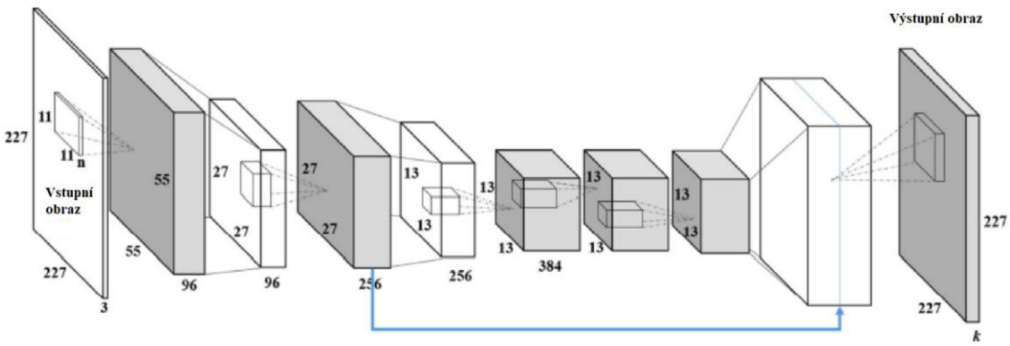
Klasifikace obrazu má obecně za cíl rozlišit, jaký objekt je na vstupním obrázku. Zvětšením velikosti sítě je možné rozšířit možnosti řešení, na druhou stranu roste výpočetní náročnost a pravděpodobnost přeučení. Pro detekci jízdních pruhů pomocí klasifikace obrazu je například využita síť DeepLane. Tato síť vytváří množinu pixelů s pravděpodobností, že se na nich vyskytuje jízdní pruh. Vstupem této neuronové sítě je obraz o rozlišení 240 x 360 pixelů. Výstupem neuronové sítě je vektor, který obsahuje až 316 různých souřadnic jízdních pruhů. DeepLane obecně dosahuje dobrých výsledků.

DETEKCE OBJEKTŮ

Hlavním cílem detekce objektů v obraze je nalezení daného objektu a určení jeho polohy. Postup určování jízdních pruhů pomocí detekce objektů bývá nejčastěji ve třech krocích. Prvním krokem je preprocesing obrázku, který upraví snímek tak, aby odpovídal vstupu pro neuronovou síť. Druhým krokem je vlastní rozpoznání objektů, které představují vodorovné značení, pomocí CNN. Posledním krokem je následné zpracování, které z množiny nalezených objektů vytvoří výsledné jízdní pruhy. Příkladem sítě s uvedeným přístupem je STLNet (Spatial-Temporal Lane detection Network) uvedená v [22]. Metoda STLNet je schopná přesně detekovat hranice jízdních pruhů a klasifikovat jednotlivé typy čar v různých podmínkách a prostředích.

SÉMANTICKÁ SEGMENTACE

Hlavním cílem sémantické segmentace obrazu je rozdělení snímku do několika oblastí s požadovanými vlastnostmi, rysy nebo celými objekty. Pro segmentaci obrazu jsou široce využívané plně konvoluční neuronové sítě viz *obr. 17*. Segmentace je tedy velmi vhodná pro určení tvaru hledaných oblastí. Současný trend algoritmů pro rozpoznávání jízdních pruhů cílí spíše na získání přesné klasifikace polohy s přesností na pixel než na specifikaci tvaru. Navzdory této potenciální nevýhodě dosáhly algoritmy založené na segmentaci výborných výsledků.



Obr. 17 Příklad plně propojené konvoluční neuronové sítě [21]

2.4 PŘEHLED SOUVISEJÍCÍCH ALGORITMŮ

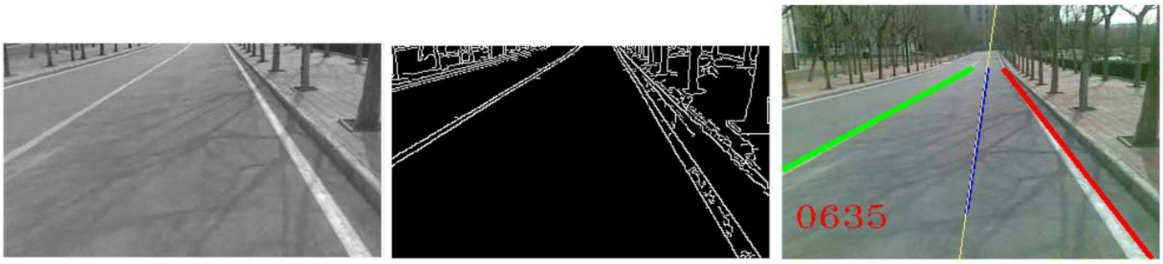
V této kapitole budou detailněji popsány konkrétní algoritmy používané pro detekci jízdních pruhů. Jednotlivých implementací existuje velké množství a stále jsou vytvářené nové, lepší metody.

Jelikož většina algoritmů má podobné rysy, byly vybrány pouze čtyři reprezentativní algoritmy. Ty vždy využívají stěžejní myšlenku, která charakterizuje celou skupinu. Vybrané algoritmy jsou seřazeny od nejstaršího po nejnovější, tomu také odpovídá jejich komplexnost, která se neustále zvyšuje. První dvě metody jsou založené pouze na tradičních metodách počítačového vidění. Poslední dvě pro detekci jízdních pruhů využívají umělých neuronových sítí.

METODA 1

Metoda The Implementation of Lane detective Based on OpenCV navržená [23] využívá klasické koncepce hledání hran pomocí detektoru hran Canny a následném detekováním přímky pomocí Houghovy transformace s využitím knihovny OpenCV. Celý algoritmus funguje následovně. Po nahrání videa do paměti je obraz oříznut o neměnnou a manuálně zvolenou

oblast zájmu, která sníží velikost obrazu o 25%, následující kroky jsou prováděny pouze na oříznutém obraze. Druhým krokem je převedení obrazu do stupňů šedi. Na takto zredukovaném obraze jsou nalezeny hrany pomocí detektoru hran Canny. Posledním krokem je Houghova transformace, která vytvoří přímky z detekovaných hran, které odpovídají vodorovnému značení jízdních pruhů. Dodatečně je také přidána přímka značící střed jízdního pruhu. Výstupy jednotlivých fází lze vidět na *obr. 18*. Tato navržená metoda běžela na procesoru Pentium 4, 3.0GHz s rychlostí detekce 15 Hz a přesností detekce 90%. Jedná se o základní metodu pro detekci jízdních pruhů, která dosahuje nejlepších výsledků na dlouhých rovných úsecích.

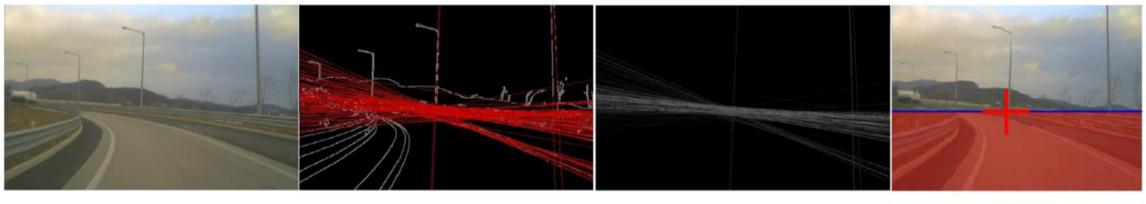


Obr. 18 a) oříznutý obrázek převedený do stupňů šedi; b) detekce hran pomocí detektoru Canny c) originální snímek s detekovanými jízdními pruhy [23]

METODA 2

Metoda Real-time illumination invariant lane detection for lane departure warning system představená [24] pracuje ve třech fázích, jež jsou: hledání bodu horizontu (Vanishing point), fáze prahování obrazu za účelem detekce značení jízdního pruhu a fáze seskupování a vykreslování jízdních pruhů.

V první fázi se určuje bod horizontu. Pro efektivní snížení výpočetní náročnosti je vytvořena oblast zájmu (Region of Intrest). Obecně lze předpokládat, že se vozovka nachází pouze ve spodní části obrazu, proto je na každém vyšetřovaném snímku nalezen bod horizontu. Hledaná oblast zájmu je následně určena jako veškerá oblast pod nalezeným bodem horizontu. Bod horizontu na snímku je určen ve čtyřech krocích. Prvním krokem je detekování hran v celém obraze pomocí detektoru Canny. Následně se pomocí Houghovy transformace vytvoří přímky odpovídající přímým částem hran. Poté jsou nalezeny všechny průsečíky těchto přímek. Jakmile jsou nalezeny všechny průsečíky přímek, tak proběhne volba bodu horizontu. Ten je určen jako těžiště všech nalezených průsečíků v obraze. Poloha bodu horizontu následně určuje výšku oblasti zájmu. Celý postup hledání bodu horizontu lze vidět na obr. 19.



Obr. 19 Fáze detekce horizontu: a) původní snímek; b) nalezené hrany pomocí detektoru Canny a aplikovaná Houghova transformace; c) poloha průsečíků; d) výsledný bod horizontu [24]

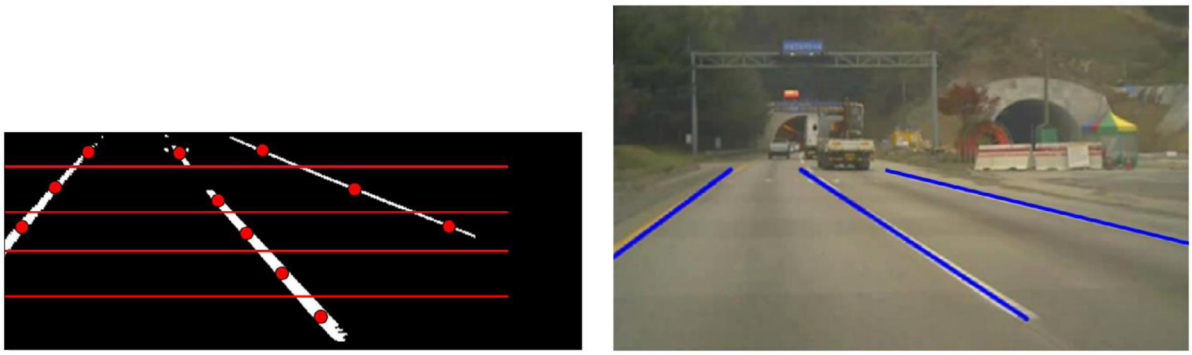
Druhou fází je detekce značení vozovky. Výstupem této fáze je binární obraz, kde 1 znamená oblast značení jízdního pruhu a 0 jsou ostatní oblasti. Aby bylo možné vytvořit binární obraz, je snímek převeden z RGB do YCbCr spektra (barevného modelu). Podle autora metody je v daném barevném modelu jízdní pruh výraznější. Zároveň je řešení rozděleno pro bílé a žluté vodorovné značení. Pro bílé značení je využita Y složka (jas obrazu) a je vytvořen histogram odpovídající zastoupení hodnot jasu v obraze viz rovnice (10). Kde C_y odpovídá kumulativnímu histogramu, a parametr (k) představuje konkrétní hodnotu jasu. Následně jsou jako značení jízdního pruhu vybrány ty pixely, které splňují hraniční podmínku podle rovnice (11). Kde B(x, y) odpovídá binárnímu obrazu o daných souřadnicích a parametr T_y představuje hraniční podmínku intenzity, se kterou bude pixel označen jako jízdní pruh.

$$C_y(k) = \sum_{k=0}^{255} Hist_y(k) \tag{10}$$

$$B(x,y) = \begin{cases} 1, & když\ C_y\big(I(x,y)\big) > T_y \\ 0, & else \end{cases} \tag{11}$$

Obdobný proces je použit pro žluté značení, ale je využita Cb složka (modrá chrominační komponenta) z YCbCr spektra. Poté dojde ke sloučení bílého a žlutého binárního obrazu pomocí operace OR.

Posledním krokem je rozdělení obrazu do 5 sekcí, kde každá sekce reprezentuje ekvidistantu od vozidla. V jednotlivých sekcích jsou označeny shluky pixelů, které odpovídají značení vozovky a jsou nalezeny jejich středy. Posledním krokem je výpočet lineární regrese pomocí metody nejmenších čtverců. Rozdělení binárního obrazu do 5 sekcí a výslednou detekci lze vidět na *obr. 20*.

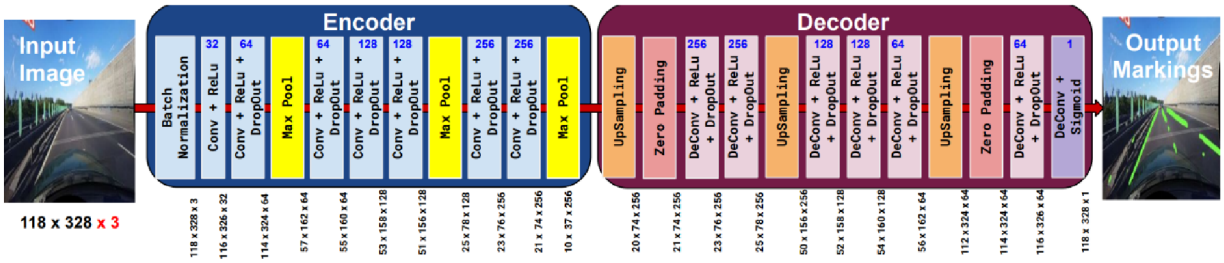


a) b)

Obr. 20 a) binární obraz rozdělený do 5 sekcí s nalezenými shluky a vyznačenými středy; b) výsledná detekce jízdního pruhu [24]

METODA 3

Další vybraná metoda Traffic Lane Detection using FCN [25] využívá k řešení problému detekce jízdních pruhů sémantickou segmentaci a plně konvoluční neuronovou síť (FCN – Fully Convolutional Network). Model neuronové sítě má architekturu enkodér-dekodér viz *obr. 21*. Tato architektura je typická pro většinu segmentačních CNN.



Obr. 21 Architektura plně konvoluční sítě [25]

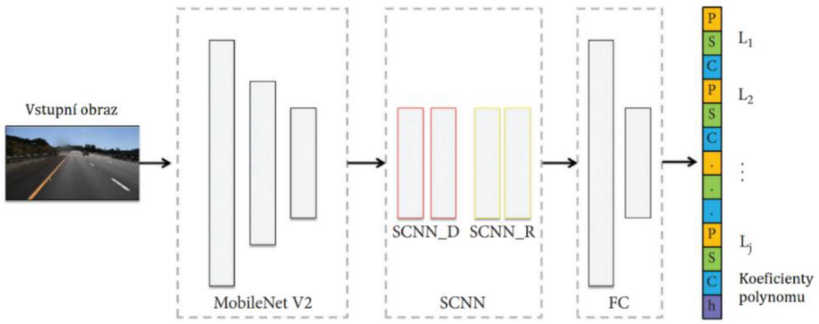
První částí modelu je 7 konvolučních vrstev společně s vrstvami dropout a maxpooling. Cílem enkodéru je zmenšit velikost snímku, a přitom zachovat klíčové vlastnosti obrazu. Druhou částí je naopak dekodér, který se skládá ze šesti dekonvolučních vrstev společně s vrstvami dropout a upsampling. Dekodér zvětší obrázek zpět na jeho původní velikost a zároveň z komprimovaných klíčových vlastností obrazu vytvoří detekované jízdní pruhy ve stupních šedi.

Pro trénování navržené neuronové sítě byl využit dataset CULane. Jelikož snímky z datasetu neodpovídaly velikosti vstupních dat do neuronové sítě, musely být předem upraveny. Model neuronové sítě byl vytvořen pomocí knihovny Keras s vestavěnými funkcemi.

Algoritmus dosahuje dobrých výsledků na rovných úsecích s dobře viditelnými jízdními pruhy. Na druhou stranu za některých podmínek nejsou výsledky stále příliš uspokojivé. Autor metody uvádí, že model nevykazuje příliš dobré výsledky, pokud jde o detekci tří pruhů a prostřední pruh je přerušovaný. Algoritmus využívá velkého datasetu jízdních pruhů pořízeného za různých podmínek, proto by měl být model robustní na změnu scény.

METODA 4

Metoda A Network That Balances Accuracy and Efficiency for Lane Detection [26] se snaží zlepšit přesnost a zároveň redukovat výpočetní náročnost algoritmu pomocí neuronových sítí. Metoda je založena na MobileNet v2, SCNN (Spital Convolutional Neural Network) a plně konvoluční neuronové síti. Strukturu sítě lze vidět na *obr. 22*.



Obr. 22 Struktura neuronové sítě [26]

V první fázi je použita MobileNet v2 síť pro rychlé zredukování obrazu a snížení výpočetní náročnosti. MobileNet v2 využívá oddělitelné konvoluce (separable convolution) místo standardní pro zrychlení a zjednodušení výpočtů jednotlivých parametrů modelu. Oddělitelná konvoluce rozděluje konvoluční okna na menší např. jedno okno 3x3 na dvě 3x1 viz *obr. 23*. Následně je použita vrstva SCNN pro efektivní zpracování informací v obraze. Poslední v pořadí jsou dvě plně propojené konvoluční vrstvy. Výstupní vrstva těchto sítí obsahuje

koeficienty polynomů, které odpovídají značení jízdních pruhů a spolehlivosti určení jízdního pruhu. Výsledek lze vyjádřit následovně:

$$f(\mathrm{I};\ \theta) = \left(\{P_j, s_j, c_j\}_{j=1}^{M_{max}}, h\right) \tag{12}$$

Kde Pj jsou souřadnice polynomu, který odpovídá značení jízdního pruhu; s_j je rozdíl mezi začátkem polynomu a nejnižším pixelem obrazu (když s_j je nulové jízdní pruh začíná se spodní hranou obrazu); cj odpovídá spolehlivosti určení polynomu; j představuje index jízdního pruhu; h je výška horizontu měřená od spodní hrany obrazu, tato hodnota je pro všechny jízdní pruhy v daný okamžik stejná.

Obr. 23 Příklad oddělitelné konvoluce [27]

Pro trénování této sítě bylo využito TuSimple datasetu s již klasifikovanými jízdními pruhy. Databáze obsahuje 72000 snímků o rozlišení 1280×720 pixelů, které zachycují převážně dálniční prostředí. Základním principem učení CNN je snižování tzv. loss funkce viz rovnice (13).

$$loss = W_p loss_p + W_s loss_s + W_c loss_c + W_h loss_h \tag{13}$$

Kde $loss_p$, $loss_s$, $loss_c$, $loss_h$ představují dílčí loss funkce a jsou vypočítány jako střední kvadratická odchylka mezi vypočítanou a skutečnou hodnotou. W_p, W_s, W_c, W_h jsou koeficienty vah pro jednotlivé loss funkce, ty jsou upravovány manuálně (pro W_s, W_c, W_h je váha nastavena na 1 a pro W_p je váha nastavena na 800).

Výsledná vytrénovaná síť pracovala s přesností vyšší než 92 % a s rychlostí detekce 30 snímků za sekundu na grafické kartě Nvidia P4000 s 1792 CUDA jádry a 8 GB paměti. Na obr. 24 je vidět výsledek neuronové sítě se třemi nalezenými jízdními pruhy, vyznačenými začátky jízdních pruhů a s vyznačenou výškou horizontu.

Obr. 24 Výsledek neuronové sítě [26]

3 NÁVRH SYSTÉMU

V rámci diplomové práce bylo zapotřebí navrhnout systém, který by byl schopný rozpoznávat jízdní pruhy v reálném čase a následně tento systém otestovat. Pro získávání obrazových dat byla použita stereo kamera ZED od společnosti StereoLabs. Jako výpočetní jednotka byl použit vestavěný mikro-počítač Jetson Nano 4 GB od společnosti Nvidia, na kterém poběží navržený algoritmus pro detekci jízdních pruhů. Nalezené jízdní pruhy budou následně zobrazeny na externím notebooku. Při zabudování systému do automobilu mohou být výsledná data zobrazována na některém z již v autě vestavěných displejů (head-up display, palubní display). Systém má být během svého provozu schopný rozpoznávat jízdní pruhy.

Všechny vytvořené skripty v rámci této práce byly vytvořeny v programovacím jazyce Python verze 3.6. Kromě samotného programovacího jazyka je k realizaci algoritmů potřeba řada knihoven popsaných v kapitole 3.2.

3.1 HARDWAROVÉ VYBAVENÍ

3.1.1 JETSON NANO DEVELOPER KIT

Jetson Nano je kompaktní jednodeskový počítač vyvinutý společností Nvidia především pro aplikace strojového učení, zpracování obrazu či mluvené řeči. Počítač disponuje čtyřjádrovým 64 - bitovým ARM A57 @ 1.43 GHz procesorem a 4 GB operační pamětí. Hlavní výpočetní jednotkou je integrovaná grafická karta s architekturou Maxwell a 128 CUDA jádry. Jednotka také obsahuje několik vstupů (HDMI port, 4x USB 3.0, Micro USB a další). Jako úložiště dat v počítači slouží microSD karta, ze které se následně i celý počítač spouští. Tato karta musí být k počítači dokoupena.

Společnost Nvidia poskytuje obraz systému JetPack SDK (Software Development Kit), který lze pomocí programu balena Etcher nahrát na microSD kartu. JetPack SDK obsahuje Jetson Linux Driver Package (L4T) s operačním systémem Linux a několika předinstalovanými knihovnami pro strojové učení a zpracování obrazu.

Obr. 25 Jetson Nano

3.1.2 STEREO KAMERA ZED

ZED je stereoskopická kamera od společnosti StereoLabs, která se specializuje na vývoj stereoskopického vidění. Kamera je vhodná pro použití v robotizovaných systémech a obecně u všech autonomních systémů. Kamera snímá obraz dvěma 1/3“ CMOS senzory s vysokou fotocitlivostí a rozlišením až 4416 ×1242. Senzory jsou od sebe vzdáleny 120 mm, díky tomu je možné triangulovat body v obraze a tím získat hloubkový obraz. Pro správnou funkci kamery je potřeba její kalibrace. Společnost StereoLabs poskytuje SDK, který nabízí řadu aplikací a funkcí pro práci s kamerou.

Obr. 26 Stereo kamera ZED

3.2 SOFTWAROVÉ VYBAVENÍ

PYTHON 3.9

Python je vysokoúrovňový programovací jazyk. Je vyvíjen jako open source software pro většinu běžných platforem (Windows, Mac OS a ve většině distribucí systému Linux je dokonce základní součástí instalace). Hlavním důvodem pro volbu jazyka python byla jeho jednoduchá syntaxe a dostupnost tutoriálů pro začátečníky. Pro python existuje velké množství knihoven, které usnadňují programování. Python také umožňuje volání knihoven jazyka C a C++, čímž dochází ke zrychlení běhu kódu při zachování jeho jednoduchosti.

OPENCV

OpenCV (Open Source Computer Vision Library) je otevřená knihovna zaměřená na počítačové vidění a strojové učení. Knihovna obsahuje přes 2500 optimalizovaných algoritmů jak z oblasti počítačového vidění, tak z oblasti strojového učení. Knihovna je napsána v jazyce C++, ale podporuje i programovací jazyky jako Python, Java, MATLAB a většinu operačních systémů. Kromě kódu obsahuje velké množství dokumentace a tutoriálů.

CUDA

CUDA (Compute Unified Device Architecture) je celý ekosystém, která umožňuje vytváření vysoce výkonných aplikací s GPU akcelerací. Technologie CUDA byla vyvinuta společností Nvidia pro jejich grafické karty. Hardwarová architektura CUDA čipu se skládá z mnoha jednoduchých skalárních procesorů. Důvodem tohoto uspořádání je možná paralelizace algoritmu, kdy jednotlivé výpočty mohou být rozděleny a vykonávány na více procesorech současně. Díky tomu dochází ke zrychlení celkového výpočetního času.

TENSORFLOW

TensorFlow je otevřená knihovna se zaměřením na strojové učení v mnoha odvětvích. Byla vyvinuta společností Google Brain. Umožňuje rychlé vytváření a trénování modelů umělých neuronových sítí. Knihovna podporuje většinu operačních systémů, stejně tak programovací jazyky jako C++, Java a Python.

KERAS

Keras je otevřená knihovna pro programovací jazyk Python, která poskytuje rozhraní pro vytváření umělých neuronových sítí. Tato knihovna obsahuje implementace většiny běžně používaných stavebních bloků neuronových sítí, jako jsou jednotlivé vrstvy, aktivační funkce, optimalizátory a řadu dalších nástrojů potřebných pro tvorbu neuronové sítě. Práce s knihovnou je velmi intuitivní a přehledná. Zároveň je knihovna vysoce optimalizovaná, takže dosahuje dobrých výkonů. Knihovna je velmi oblíbená, používají ji i světoznámé organizace a společnosti například NASA, YouTube a Waymo.

NUMPY

NumPy je jedna ze základních knihoven pro programovací jazyk Python. Jedná se o knihovnu, která umožnuje tvorbu vícerozměrných polí, matic a vektorů. Dále také umožňuje základní operace s těmito strukturami jako jsou např. základní algebra, statistické operace, diskrétní Fourierova transformace a další. NumPy usnadňuje vytváření kódu tím, že minimalizuje potřebu smyček. Rovněž zvyšuje rychlost výpočtu díky funkcím, které jsou předkompilované v jazyce C. Existuje řada dalších knihoven, které jsou závislé právě na polích NumPy a rozšiřují její použitelnost.

CUPY

CuPy podobně jako NumPy je knihovna pro programovací jazyk Python umožňující tvorbu a práci s vícerozměrnými poli. CuPy uplatňuje CUDA Toolkit pro plné využití GPU architektury ke zrychlení výpočtu. Toto rozhraní je vysoce kompatibilní s NumPy.

3.3 OŽIVENÍ POČÍTAČE JETSON NANO

Pro oživení Jetson Nano je výrobcem poskytovaný obraz systému, který je možný pomocí programu balena Etcher nahrát na micro-SD kartu, ze které je následně systém spouštěn. Společností Nvidia je doporučovaná minimální velikost micro-SD karty 32 GB o rychlostní třídě UHS-1. V navrženém systému byla použita micro-SD karta od firmy Kingston s kapacitou 128 GB a rychlostní třídou UHS-1. Nahraný obraz systému obsahuje L4T (Linux for Tegra) a řadu knihoven. L4T je operační systém založený na Linuxu pro řadu procesorů Tegra od společnosti Nvidia. Mezi hlavní knihovny patří CUDA Toolkit, cuDNN a TensorRT. Pro řešení diplomové práce byla použita nejaktuálnější verze obrazu JetPack 4.6.1.

Po vložení micro-SD karty do slotu je možné Jetson Nano spustit, buď s připojenou obrazovkou nebo v tzv. headless módu bez připojené obrazovky pomocí externího počítače. Pro první spuštění systému byla využita možnost s připojenou obrazovkou, byl vytvořen uživatelský účet, nastaven systémový jazyk, časové pásmo a klávesnice. Po přihlášení se již zobrazí plocha systému viz obr. 27.

Obr. 27 Plocha operačního systému Linux

Po spuštění systému bylo zapotřebí instalovat potřebné knihovny. Pro tento úkol byl použit správce balíčků pip, který je součástí instalace pythonu. Pomocí příkazu: `pip install <jmeno_knihovny>` byly nainstalovány následující knihovny podle tab. 1.

Tab. 1 Přehled jednotlivých verzí nainstalovaného softwaru

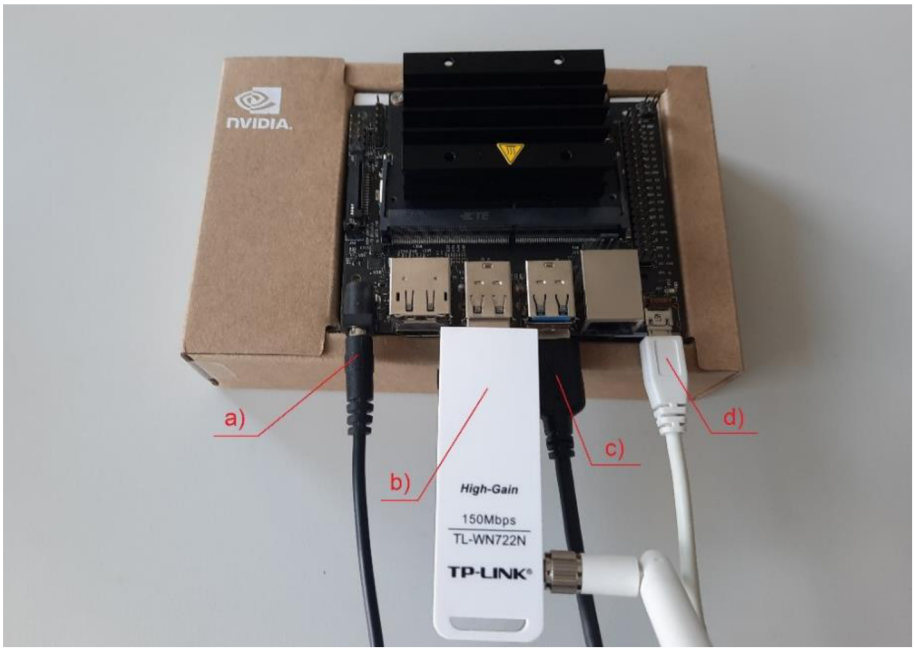
Python	1.6.5
OpenCV	4.1.1
Numpy	1.19.4
Cupy	9.6.0
Tensorflow	2.5.0
Keras	2.8.0
CUDA	10.2

Python je citlivý na kompatibilitu některých verzí, především u knihoven využívajících CUDA grafické akcelerace. Po instalaci veškerých knihoven byl Jetson Nano připraven ke spuštění vytvořených algoritmů. Celková doba instalace trvala okolo 5 hodin.

3.4 ZABUDOVÁNÍ HARDWARU VE VOZIDLE

Kamera byla umístěná na čelním skle v ose vozidla tak, aby kapota nezasahovala do výhledu kamery. V pozdější fázi testování bylo zjištěno, že toto umístění kamery není pro snímání vozovky úplně vhodné. Obraz byl ovlivněn velkým rozdílem jasu mezi oblohou a vozovkou. V testovací fázi byla tedy zvolena poloha, která sice částečně snímala kapotu automobilu, ale snímaný obraz nebyl přesycený slunečním jasem. Kamera disponuje 1,5 metrů dlouhým USB 3.0 kabelem, kterým je připojena k mikro-počítači Jetson Nano a zároveň je tímto kabelem napájena. Kamera pro svůj provoz požaduje napětí 5 V a proud 380 mA.

Jetson Nano je umístěný v boxu tak, aby bylo zamezeno nežádoucímu pohybu a současně bylo umožněno dostatečné chlazení. Mikro-počítač je připojený napájecím kabelem ke stejnosměrnému adaptéru do autozásuvky. Jako adaptér byl použit Car Charger Mini Style od firmy Vention. Ten pracuje se vstupním napětím 12 V a dodává požadované výstupní napětí 5 V s maximálním proudem 4 A. Kromě zdroje napětí a kamery je k mikropočítači také připojen Wifi adaptér TP-LINK TL-WN722N. Jedná se o adaptér určený pro zapojení prostřednictvím USB 2.0 rozhraní. Pracuje s pásmem 2.4 GHz a podporuje přenosové rychlosti v závislosti na síle signálu až do 150 Mbit/s. Jetson Nano umožňuje vzdálené připojení přes SSH (Secure Shell) protokol pomocí micro-USB portu.



Obr. 28 Připojení periferií k Jetson Nano a) zdroj napětí; b) Wifi adaptér; c) stereo kamera ZED; d) micro USB připojení k notebooku

Pro ověření a zobrazení výsledků slouží notebook, který je k mikro-počítači připojen bezdrátově pomocí VNC (Virtual Network Computing). VNC je grafický program, který umožňuje připojení ke vzdálené ploše. Notebook také slouží k ovládání mikropočítače a jsou z něj volány jednotlivé příkazy pro spuštění algoritmu pro detekci jízdních pruhů.

4 NÁVRH ALGORITMŮ

V této kapitole je rozebrán návrh dvou algoritmů, které byly v rámci diplomové práce vytvořeny. První vytvořený algoritmus pracuje pouze s využitím tradičních metod počítačového vidění. Druhý vytvořený algoritmus pracuje s využitím konvolučních neuronových sítí a následného zpracování za účelem získání jízdních pruhů. V kapitole 5 jsou tyto dva odlišné přístupy srovnány. Navrhnuté algoritmy jsou součástí elektronických příloh.

4.1 REAL TIME KOMUNIKACE MEZI KAMEROU A HARDWAREM

Pro zajištění komunikace mezi počítačem Jetson Nano a kamerou ZED bylo využito knihovny OpenCV. Funkce `cv2.VideoCapture(IDkamery / cesta souboru)` zajistí komunikaci hardwaru s kamerou. Pomocí této funkce také lze načíst vhodný video soubor. Tato možnost se především hodí pro testování algoritmu z důvodu opakovatelnosti měření.

Pro simulaci RT systému byla vytvořena smyčka while, která načítá snímky vždy v požadovaný čas pomocí funkce `cv2.cap.read()`. Tato funkce umožňuje načtení snímku z kamery popř. z videosouboru do požadované proměnné. Zajištění pravidelného běhu smyčky while je provedeno podmínkou if, která spouští načítání snímku vždy za určitý časový interval. Podmínka if nezastavuje celý algoritmus, ale je možné využít neobsazenou výpočetní kapacitu procesoru pro jiné účely.

4.2 DETEKCE POMOCÍ METOD POČÍTAČOVÉHO VIDĚNÍ

První algoritmus pro hledání jízdních pruhů je založený pouze na metodách počítačového vidění. Snímek zachycený kamerou je nejprve upraven pro další zpracování, dále je prahován a je na něm provedena transformace do ptačí perspektivy (z pohledu řidiče je obraz transformován do pohledu kolmého k vozovce). V takto upraveném obraze je nalezen začátek jízdního pruhu a následně další body, které odpovídají bodům pravého a levého jízdního pruhu. Dalším krokem je vytvoření polynomu druhého stupně, který je proložen nalezenými body. Finální fází je vykreslení výsledků.

4.2.1 BĚH ALGORITMU

V první fázi algoritmu jsou načteny potřebné knihovny a funkce, které budou použity v další fázi. Pro nahrání knihovny slouží import: `import <jméno modulu / funkce>.` Dále se nadefinují potřebné proměnné, naváže se komunikace mezi Jetson Nano a kamerou a následuje hlavní smyčka while. Uvnitř smyčky je načten snímek z kamery a poté jsou volány následující funkce v daném pořadí: „crop_image“, „perspective_transform“, „line_fit“ a „final_viz“. Poslední fází je vykreslení jízdních pruhů. Na *obr. 29* je zobrazen načtený snímek.

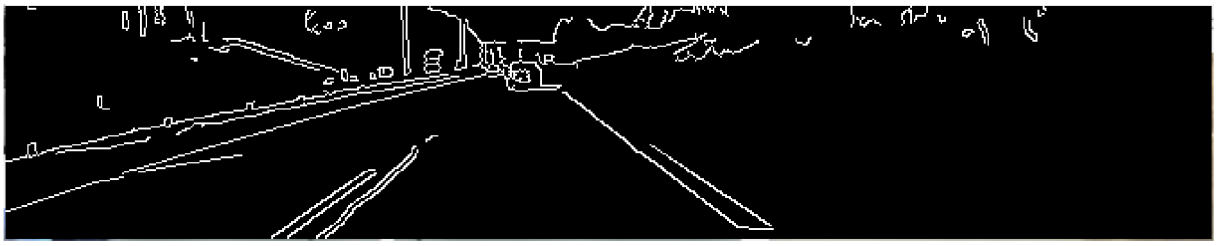
Obr. 29 Snímek načtený kamerou

CROP_IMAGE

První volaná funkce je crop_image. Protože se načtený snímek ze stereo kamery ZED skládá ze dvou obrazů, je v potaz brán pouze obraz z levého snímače. Dále je snímek oříznut o oblasti, na kterých se nemůže vyskytovat vozovka (část snímku s kapotou a obloha). Dalším krokem je převedení snímku z barevného modelu RGB do HSV pomocí funkce: `cv2.cvtColor(image, cv2.COLOR_RGB2HSV)`. Ze tří složek HSV modelu je vybrána složka V – Value, která odpovídá hodnotě jasu. HSV modelu bylo použito, protože bílé vodorovné značení je více výrazné než při použití převodu RGB do stupňů šedi. Na *obr. 30* je zobrazena jasová složka HSV spektra, zároveň je snímek oříznutý o nepotřebné oblasti.

Obr. 30 Jasová složka HSV barevného modelu a oříznutí originálního snímku

Na takto upraveném snímku je aplikován Gausův filtr a následně jsou na něm nalezeny hrany pomocí hranového detektoru Canny pomocí těchto funkcí: `cv2.GaussianBlur; cv2.Canny`. Parametry detektoru Canny musejí být nastaveny podle konkrétní situace tak, aby bylo nalezeno veškeré vodorovné značení, a přitom pokud možno co nejméně falešných detekcí, zejména pak defektů vozovky a stínů. Výstupem popsaných úprav je oříznutý binární obraz. *Obr. 31* znázorňuje výsledný binární obraz vytvořený detektorem hran Canny.



Obr. 31 Snímek po aplikování hranového detektoru Canny

PERSPECTIVE_TRANSFORM

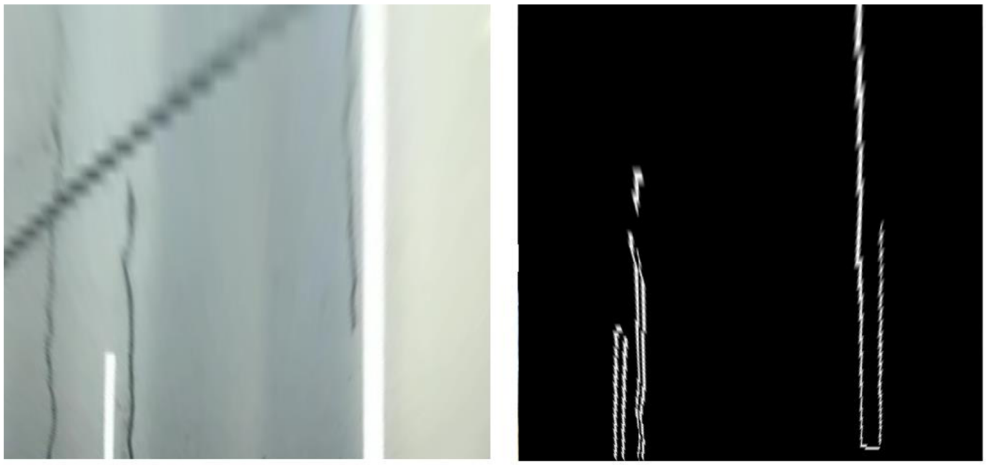
Funkce `perspective_transform` vytváří transformaci perspektivy a převádí obraz z čelního pohledu (pohledu řidiče) na pohled z ptačí perspektivy. To je provedeno pomocí dvou funkcí.

První funkce: `cv2.getPerspectiveTr`ansform; vytvoří transformační matici (m) ze čtyř párů odpovídajících si bodů (src – souřadnice vrcholů čtyřúhelníku ve zdrojovém obrázku; dst – souřadnice odpovídajících vrcholů čtyřúhelníku v cílovém obrazu).

Druhá funkce: `warpPerspective` vytvoří transformaci perspektivy původního obrázku (img) pomocí zadané transformační matice (m) a interpolační metody (flags). V navrženém řešení byla použita bilineární interpolace. Parametr size odpovídá rozměru výstupního obrázku.

V tomto obraze při správném nastavení odpovídají souřadnice jednotlivých pixelů souřadnicím jízdních pruhů. Nevýhodou je, že čtyřúhelníky podle kterých je počítána transformační matice,

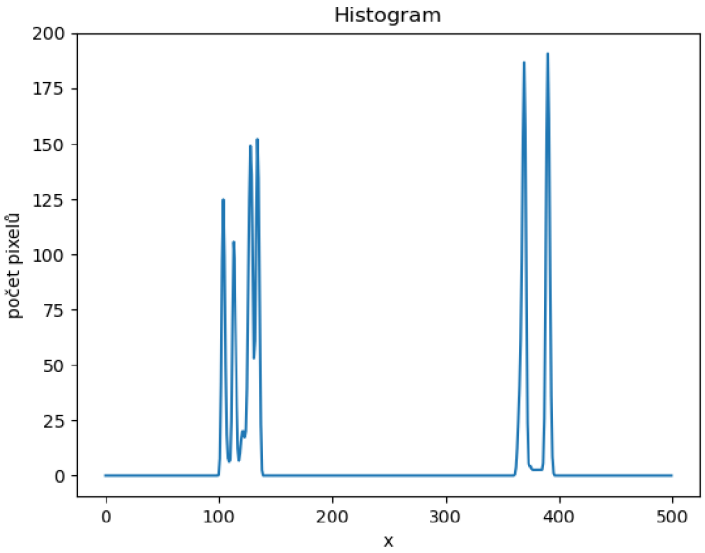
se zadávají ručně a při každé změně polohy kamery se toto nastavení musí měnit. Na *obr. 32* lze vidět dvojice snímků převedených do ptačí perspektivy. První snímek vyobrazuje originální snímek a druhý představuje binární obraz.



Obr. 32 Snímek transformovaný do ptačí perspektivy a) originální snímek b) binární obraz

LINE_FIT

Funkce line_fit lze rozdělit do tří fází. Vstupem do funkce je binární transformovaný obraz. V první fázi je vytvořen histogram tak, že jsou sečteny všechny nenulové pixely v jednotlivých sloupcích. Tento histogram vlastně představuje pravděpodobnost výskytu jízdního pruhu. Následně je histogram rozdělen na levou a pravou polovinu, v každé polovině je vybrán bod s nejvyšším součtem. Horizontální souřadnice tohoto bodu odpovídá souřadnici počátku jízdního pruhu. *Obr. 33* představuje histogram četnosti výskytu pixelu odpovídající jízdnímu pruhu (nenulový pixel). Vodorovná osa histogramu představuje vodorovnou osu vyšetřovaného snímku a svislá osa představuje zmíněnou četnost výskytu nenulového pixelu.



Obr. 33 Vytvořený histogram četnosti nenulových pixelů

V druhé fázi je vytvořeno celkem 8 oken, ve kterých jsou nalezeny nenulové pixely. Okna se vytváří vždy v páru pro levý a pravý jízdní pruh. Jakmile jsou nalezeny dva body představující počátek jízdního pruhu, je nad každým tímto bodem vytvořeno okno o velikosti 101 x 125 pixelů. V tomto okně jsou vybrány nenulové pixely a uloženy do seznamu, který představuje souřadnice jízdního pruhů. Pokud bylo nalezeno více než 20 nenulových pixelů je výchozí pozice pro tvorbu nového okna posunuta do středu horizontální souřadnice nenulových pixelů. Proces tvorby oken a zapisování souřadnic nenulových bodů do seznamu je takto proveden 8krát. Z takto nalezených souřadnic pro levý a pravý jízdní pruh se vypočítá polynomiální regrese druhého řádu.

FINAL_VIZ

Poslední funkce final_viz slouží k vykreslení polynomu a převedení obrazu zpět do čelního pohledu automobilu. Obr. 34 zobrazuje finální vykreslení jízdního pruhu.

Obr. 34 Finální vykreslení jízdního pruhu

4.3 DETEKCE POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

Jelikož algoritmus pomocí standartních metod počítačového vidění nedosahoval dostatečných výsledků (bylo zapotřebí neustále měnit parametry jednotlivých funkcí v závislosti na umístění kamery a změně osvětlení vozovky), byl vytvořen i druhý algoritmus pro detekci jízdních pruhů založený na CNN. Druhým důvodem pro vytvoření algoritmu založeného na CNN je možnost plného využití výpočetního potenciálu jednodeskového počítače Jetson Nano.

Algoritmus byl navržen tak, aby neuronová síť vytvářela segmentaci jízdního pruhu před vozidlem. V následném zpracování je zvýrazněno vodorovné značení jízdního pruhu.

4.3.1 BĚH ALGORITMU

Obdobně jako u předchozího algoritmu jsou v jeho první fázi načteny potřebné knihovny a jejich části. Dále se definují základní proměnné, nahraje se model neuronové sítě a následuje hlavní smyčka while. Uvnitř smyčky je načten obraz z kamery a poté je volána funkce „road_lines“, která vytvoří segmentaci jízdního pruhu. Dále je vytvořen polynom, který odpovídá vodorovnému značení jízdního pruhu pomocí funkce „polyfit_lane“, a v poslední fázi je zobrazen výsledný obraz.

ROAD_LINES

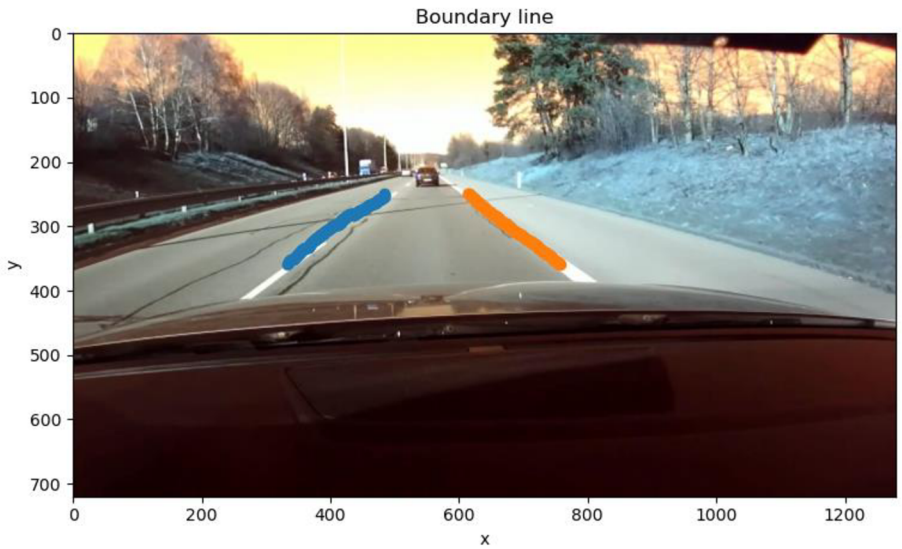
Funkce road_lines má pouze jeden vstupní parametr, kterým je snímek zachycený kamerou. Tento snímek je upraven tak, aby odpovídal vstupu do neuronové sítě. Je upravena jeho velikost a je převeden na Numpy pole. Následně je takto upravený snímek zpracován neuronovou sítí (jejíž struktura bude popsána v kapitole 4.3.2) a je vytvořena predikce jízdních pruhů. Výsledná predikce je vlastně jednobarevný obraz, kde intenzita barvy značí pravděpodobnost výskytu jízdního pruhu viz *obr. 35*.

Obr. 35 Nalezená predikce jízdního pruhu pomocí CNN

Nalezená predikce je dále upravována. Je vytvořen průměr z posledních 4 predikcí, tím se stane výstup více stabilní. Dále je upravena dimenze predikce tak, aby odpovídala jednomu barevnému signálu, v navrženém algoritmu je na výstupu použita zelená barevná komponenta. Výstupem z této funkce je původní snímek s detekovaným jízdním pruhem.

POLYFIT_LANE

Funkce polyfit_lane je schopna ze segmentovaného obrazu vytvořit polynom, který odpovídá vodorovnému značení. Vstupním argumentem do funkce je snímek predikce CNN. Na snímku jsou nalezeny kontury pomocí funkce findContours. Následně je ze seznamu všech nalezených kontur vybrána ta s největší plochou a je uložen její index. Největší obrys je dále upraven tak, že z celého obrysu jsou brány v úvahu pouze body, které odpovídají vodorovnému značení. Tyto nalezené body jsou rozděleny na levý a pravý jízdní pruh a jsou uloženy jejich souřadnice. *Obr. 36* zobrazuje pixely, které představují vodorovné značení.



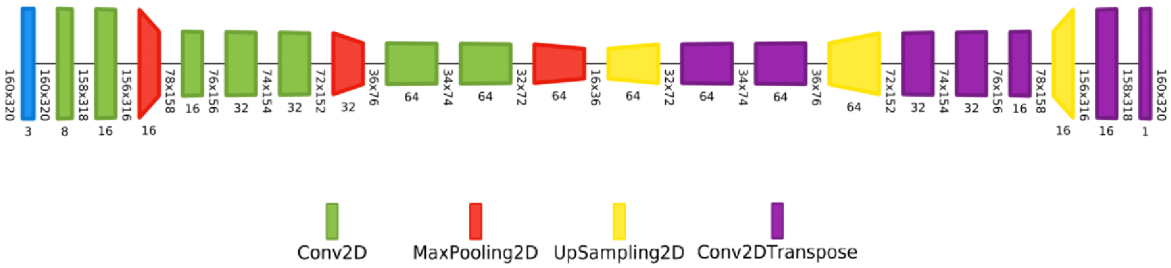
Obr. 36 Body odpovídající vodorovnému značení

Ze souřadnic nalezených bodů je vypočítána polynomiální regrese druhého řádu pro levý a pravý jízdní pruh, následně je vše vykresleno do původního snímku. Výsledek detekce lze vidět na *obr. 37*.

Obr. 37 Detekovaný jízdní pruh s vykresleným vodorovným značením

4.3.2 STRUKTURA SÍTĚ

Navržená neuronová síť se skládá z vrstev popsaných v kapitole 2.3.1. V průběhu testování bylo vytvořeno několik variant neuronové sítě pro ověření rychlosti a přesnosti detekce. Strukturu sítě lze vidět na *obr. 38*. Její tvar je zrcadlově převrácený a připomíná písmeno „U“, kde se velikost vstupu postupně komprimuje a následně expanduje zpět na původní velikost. Tato struktura je běžně používaná u segmentačních neuronových sítí jako je U-net, SegNet nebo DeconvNet.



Obr. 38 Navržená konvoluční neuronová síť [28]

Jednotlivé verze sítě se liší především svojí velikostí. Výchozí verze obsahuje nejvíce vrstev s nejvíce parametry. Dále byla vytvořena verze small se stejným počtem vrstev jako u výchozí verze, ale s méně parametry. Poslední verzí je verze tiny, která obsahuje nejméně vrstev i parametrů. Počty vrstev a jednotlivých parametrů sítě v závislosti na verzi lze vidět v tab. 2.

Tab. 2 Počet parametrů jednotlivých verzí CNN

Název varianty	Počet konv + dekonv vrstev	Počet pooling a unpooling vrstev	Počet parametrů
CNN – default	14	6	181 693
CNN – small	14	6	46 037
CNN – tiny	10	4	14 821

Vstupní vrstvou je samotný obraz o velkosti 320 x 160 x 3 (výška x šířka x počet barevných kanálů). Menší velikost obrazu již způsobovala nepřesnosti v detekci. Jednalo se tedy o kompromis mezi rychlostí a kvalitou detekce. Počty konvolučních filtrů v jednotlivých vrstvách pro všechny tři verze lze vidět v tab. 3. Počty dekonvolučních filtrů jsou stejné jako počty konvolučních filtrů. Po tučně zvýrazněných vrstvách následuje pooling vrstva. Všechny neurony obsahují ReLU aktivační funkci a konvoluční vrstvy označené hvězdičkou obsahují dropout vrstvu s parametrem pravděpodobnosti deaktivace neuronu 20%.

Tab. 3 Počet konvolučních filtrů v jednotlivých vrstvách a verzích CNN

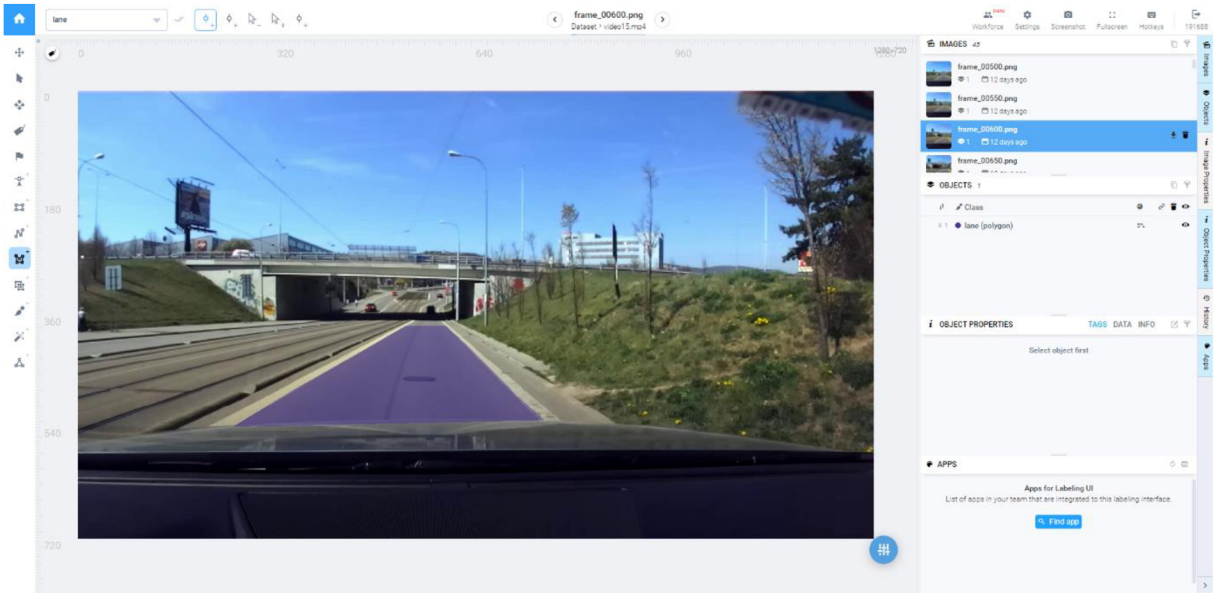
Název varianty	Konv 1	**Konv 2**	Konv 3	Konv 4	**Konv 5**	Konv 6	**Konv 7**
CNN – default	8	**16**	16*	32*	**32***	64*	**64***
CNN – small	8	**8**	8*	16*	**16***	32*	**32***
CNN – tiny	8	**8**	8*	16*	**16***	-	-

4.3.3 TVORBA DATASETU

Pro učení neuronové sítě byl vytvořen vlastní dataset. První fází vytváření datasetu byl sběr obrazových dat. K tomu byla použita kamera ZED, tedy stejná kamera jako byla použita pro následnou detekci. Snímky byly ukládány na připojený laptop ve video formátu mp4. Videozáznamy byly pořizovány za dne a za běžného provozu na komunikacích v blízkém okolí Fakulty strojního inženýrství. Z palubního počítače vozidla bylo zjištěno, že délka zaznamenaných silnic byla okolo 20 kilometrů. Z této délky byly vybrány pouze úseky s relativně dobře vyznačenými pruhy. Videa vhodná pro anotaci pokrývala komunikaci o celkové délce cca 15 kilometrů. Na *obr. 39* jsou vybrané snímky z datasetu.

Obr. 39 Vybrané snímky z datasetu

Na takto získaných videozáznamech bylo zapotřebí vyznačit jízdní pruhy pro samotné učení s učitelem. Pro účely označování byla využita platforma Supervise.ly. Jedná se o online službu, která umožňuje efektivní označování dat a vytváření datasetů. Po založení účtu je možné vytvořit projekt a nahrát do projektové složky vytvořená videa. Program je schopný videozáznam rozdělit na samostatné snímky. Následně jsou tyto snímky upraveny a setříděny tak, aby byla možná efektivní anotace (označování jízdních pruhů). Anotování probíhalo ručně pomocí nástrojů k tomu určených. Po anotování je možné data stáhnout ve formátu json.



Obr. 40 Prostředí Supervise.ly při anotování obrázků

Navržená neuronová síť se trénuje pomocí dvojice snímků, jednoho originálního a druhého anotovaného, na kterém jsou vyznačeny pouze jízdní pruhy. Pro převod dat z formátu json do png byla vytvořena funkce „json2png“. Tato funkce převádí textové informace o velikosti snímku a poloze souřadnic polygonu ve formátu json na jednobarevný obrázek s vykresleným polygonem ve formátu png. Takto vytvořený snímek se následně ukládá do určené složky. Práce s touto funkcí je efektivní, protože je schopna převádět celé složky s datasety najednou. Dvojici originálního a anotovaného snímku lze vidět na *obr. 41*.

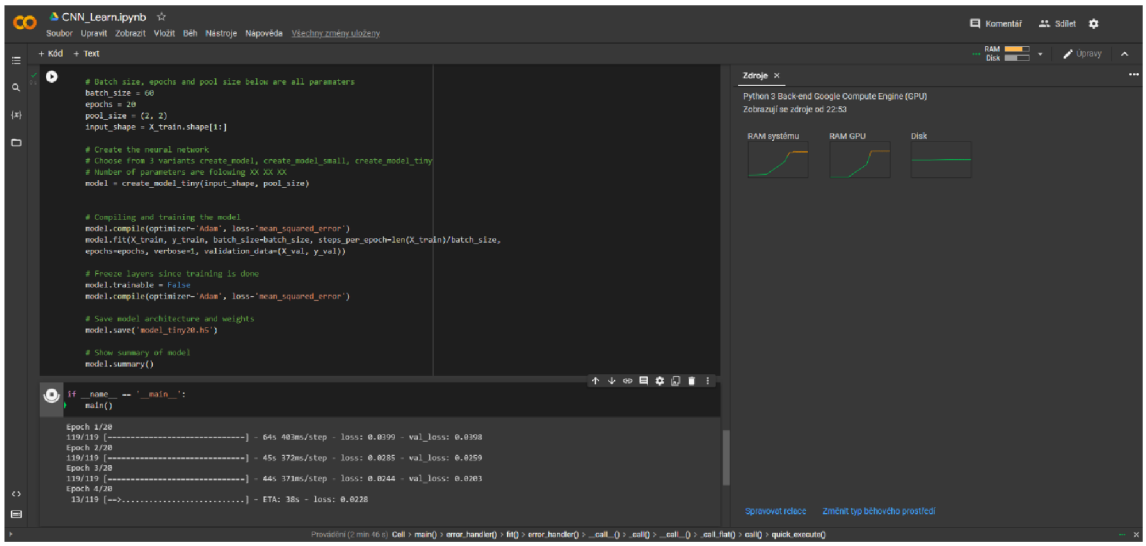
a) b)

Obr. 41 Dvojice a) originálního snímku a b) anotovaného snímku

Dále pro jednodušší manipulaci a nahrávání anotovaných a originálních snímků do neuronové sítě pro účely učení jsou data sterilizována a desterilizována pomocí knihovny picke. Funkce dump převede jakýkoliv objekt v Pythonu na jeden binární soubor a uloží jej. V navrženém algoritmu je tedy převáděna matice o velikosti (počet snímků x šířka snímku x výška snímku) na jeden soubor. Takto vytvořený soubor může být následně otevřen a rozbalen na potřebném místě pomocí funkce load. V tomto případě se data rozbalují v místě nahrávání dat určených pro učení neuronové sítě.

4.3.4 Učení neuronové sítě

Aby navržené varianty neuronové sítě dosahovaly optimálních výsledků, musely být vhodně zvoleny parametry pro trénování. Především se jednalo o počet trénovacích cyklů, velikost dávky a způsob výpočtu chyby. Pomocí knihovny Keras byl model nastaven pro trénování a následně trénován. Příkazem compile byl importován optimalizér Adam, nastavený na výchozí hodnoty. Dále byla importována chybová funkce „mean_squared_error“, která vypočítává chybu modelu pomocí střední kvadratické odchylky. Následuje příkaz fit, který již trénuje samotný model. Argumenty této metody jsou: vstupní data ve tvaru NumPy matice (originální snímky); výstupní data také ve tvaru NumPy matice (anotované snímky); velikost dávky pro jednu iteraci (batch_size) nastavená na hodnotu 20 snímků; počet tréninkových cyklů; verbose slouží k nastavení zobrazení postupu trénování; posledním argumentem je validační dataset, který byl náhodně vybrán z trénovacího datasetu. Celkový počet tréninkových cyklů byl nastaven na hodnotu 20. Při tomto nastavení model dosahoval nejlepších výsledků a nejevil známky přeučení. Model byl trénován v prostředí Google Colaboratory (Colab). Tento produkt od společnosti Google Research umožňuje psát a spouštět libovolný kód pythonu prostřednictvím prohlížeče a je zvlášť vhodný pro strojové učení. Colab je hostovaná služba, která k použití nevyžaduje žádné nastavení a přitom poskytuje bezplatný přístup k výpočetním zdrojům včetně GPU.



Obr. 42 Prostředí Google colab při učení neuronové sítě

Vytvořený model s váhami a konfigurací je uložen ve formátu .h5, což je souborový formát pro ukládání strukturovaných dat. Keras ukládá modely v tomto formátu, protože může snadno uložit váhy a konfiguraci modelu do jediného souboru. Díky tomu je umožněna jednoduchá manipulace s modelem.

5 EXPERIMENTÁLNÍ OVĚŘENÍ VÝSLEDKŮ

Důležitou částí diplomové práce bylo experimentální ověření kompletního systému kamera – Jetson Nano – software z hlediska rozpoznávání jízdních pruhů a schopnosti pracovat v reálném čase. Algoritmus založený na počítačovém vidění i algoritmus založený na CNN se všemi třemi variantami byl otestován v reálném příměstském provozu.

5.1 VÝPOČETNÍ ČAS

Pro testování doby běhu byl algoritmus nejprve spuštěn a teprve po 20 iteracích bylo spuštěno měření. To bylo provedeno z důvodu, že při prvních iteracích dochází k velkým časovým výchylkám. V tab. 4 je možné vidět průměrný a maximální výpočetní čas i přepočet na snímky za sekundu (FPS).

Tab. 4 Srovnání výpočetních času

Algoritmus	Průměrný čas [ms]	Maximální čas [ms]	FPS
CV	59,34	93,6	16.85
CNN – default	111,2	410,1	8.99
CNN – small	108,9	345,7	9.18
CNN – tiny	90,9	356,8	11.00

VÝPOČETNÍ ČAS ALGORITMU ZALOŽENÉHO NA CV

Z časového srovnání vyplývá, že nejkratších výpočetních časů dosahuje algoritmus založený na metodách počítačového vidění. To je způsobeno tím, že ve většině případů bylo použito již optimalizovaných funkcí knihoven Numpy, OpenCV a především knihovny Cupy, která umožňuje paralelizovat některé výpočty s využitím grafické karty. Zároveň v první fázi zpracování obrazu je snímek oříznut o oblasti, na kterých se nevyskytuje vozovka. Díky tomu došlo ke zmenšení obrazu téměř o třetinu. V dalších fázích algoritmus pracuje především s jednobarevným nebo binárním obrazem. Tímto postupem byl redukován objem zpracovávaných dat nejméně o třetinu. Při pohledu na výpočetní čas jednotlivých částí zpomaluje algoritmus nejvíce funkce line_fit, která trvá více než polovinu běhu algoritmu. Přehled výpočetních časů jednotlivých fází algoritmu lze vidět v tab. 5.

Tab. 5 Srovnání výpočetních časů jednotlivých částí algoritmu založeného na CV

Funkce	Výpočetní čas [ms]
Celkový čas	59,3
Crop_image	6,9
Perspective_transform	6,6
Line_fit	34,6
Final_viz	10,9

VÝPOČETNÍ ČAS ALGORITMU ZALOŽENÉHO NA CNN

Algoritmus založený na CNN je ve srovnání dvakrát pomalejší. Při bližším prozkoumání doby běhu jednotlivých částí algoritmu je patrné, že nejrychleji proběhne část preprocessingu. Je to logické, protože se jedná pouze o jednoduché upravení snímku do podoby vstupu neuronové sítě.

Druhým v pořadí byl výpočetní čas postprocessingu. Z celkového výpočetního času se jednalo o více než třetinu. To je způsobeno především vytvářením kontur ze segmentovaného snímku a výpočtem koeficientů polynomu, které odpovídají vodorovnému značení.

Nejdelší dobu trval výpočet predikce modelu. Zrychlení tohoto času by teoreticky bylo možné převodem modelu do více optimalizované formy, např. pomocí TensorRT.

Tab. 6 Srovnání výpočetních časů jednotlivých částí algoritmu založeného na CNN

Verze	Preprocessing [ms]	CNN – model [ms]	Postprocessing [ms]
Default	2,1	66,3	41,8
Small	2,1	65,8	40,1
Tiny	2,1	49,3	38,7

Délka výpočtu modelu se liší v závislosti na verzi modelu viz tab. 6. Tento rozdíl však není nijak zvlášť významný. Časový rozdíl mezi výchozí a nejmenší verzí činí pouze 170 milisekund, to představuje 25% z celkové doby běhu modelu. Přitom rozdíl v počtu parametrů činí 166872 parametrů, což představuje pokles o více než 91% parametrů. V rámci testování byla vytvořena i varianta modelu, která zpracovává snímek s poloviční rozměry. Toto opatření částečně zrychlilo chod neuronové sítě, ale model již nedosahoval dobrých výsledků.

Metoda založená na konvolučních neuronových sítích jako celek nedosahuje příliš dobrých výpočetních časů, značnou část výpočtu zabírá převod segmentovaného jízdního pruhu na polynom. Možnou optimalizací tohoto algoritmu by bylo zvolení jiné segmentační metody, která by přímo detekovala vodorovné značení jízdního pruhu, a tím by se redukoval čas potřebný pro následné zpracování. Druhou možností by bylo vytvoření neuronové sítě, která by přímo určovala koeficienty polynomu (podobně jako Metoda 4 z teoretické části diplomové práce).

Pro zlepšení rychlosti běhu obou algoritmů by kromě optimalizace kódu bylo možné použít výkonnějšího počítače. Společnost Nvidia nabízí celou řadu produktů Jetson, které jsou speciálně navržené pro běh neuronových sítí při zachování nízké elektrické spotřeby.

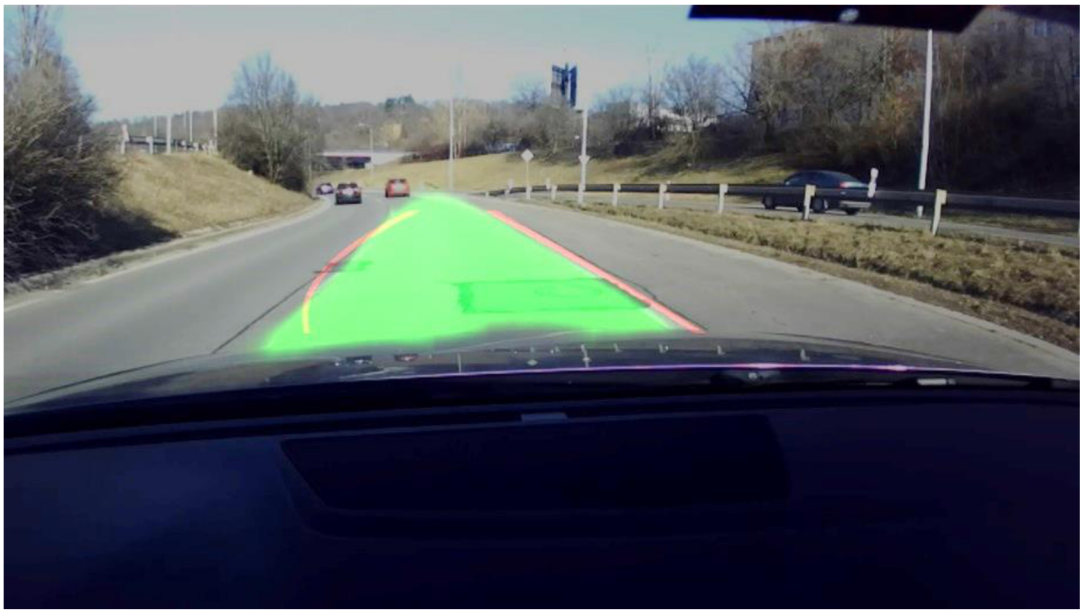
5.2 SPOLEHLIVOST DETEKCE

Pro testování spolehlivosti detekce byl vytvořen nový dataset, který obsahoval více než 600 snímků, které nebyly použity pro trénování neuronové sítě. Jednotlivé snímky byly zpracovány daným algoritmem a následně bylo posouzeno, zda se jedná o správnou detekci. Z takto získaných dat byla vypočítána procentuální úspěšnost detekování jízdního pruhu, která je zapsána v tab. 7.

Tab. 7 Přesnost detekce jízdního pruhu

Algoritmus	Správně detekované jízdní pruhy [%]
CV	71,8
CNN – default	78,3
CNN – small	54,1
CNN – tiny	13,5

Z vyhodnocených dat je patrné, že nejlepší detekce jízdního pruhu dosahuje algoritmus založený na CNN ve svojí základní verzi s nejvíce parametry. Spolehlivost detekce konvoluční neuronové sítě s nižším počtem parametrů rapidně klesá, a to až do zcela nepřípustné míry. Hlavním problémem tohoto algoritmu bylo především nepřesné vytvoření polynomu. Kdy predikovaná segmentace jízdního pruhu byla sice správná, ale následné zpracování již nedosahovalo takové přesnosti viz obr. 43.



Obr. 43 Špatné vytvoření polynomu při správné segmentaci jízdního pruhu

Velmi často dochází k chybné segmentaci neuronové sítě, když je značná část vozovky pokryta stínem viz obr. 44. Podobný efekt nastával při průjezdu vozidla v sousedním jízdním pruhu, nebo když se jiné vozidlo jedoucí ve stejném jízdním pruhu přiblížilo vozidlu z výhledu. Zmíněnému ovlivnění segmentace by šlo pravděpodobně předejít rozšířením datasetu o další snímky nebo rozšířením neuronové sítě.

Obr. 44 Citlivost CNN na velké oblasti pokryté stínem

Verze CNN s menším počtem parametrů a vrstev již nedosahovaly srovnatelných výsledků a jejich schopnost segmentovat jízdní pruh se rapidně snižovala. Navíc na výsledné segmentaci již byly patrné známky přeučení viz obr. 45.

Obr. 45 Známky přeučení CNN

Algoritmus založený na počítačovém vidění dosahoval v porovnání se základní verzí nižší spolehlivosti. Přesnost určení vodorovného značení je snáze narušena lokálními nedokonalostmi vozovky, jako jsou například praskliny, skvrny nebo stíny. Další problém nastává, pokud je zakřivení jízdního pruhu tak velké, že se dostane mimo oblast, která je převáděna do ptačí perspektivy. V takovém případě není algoritmus schopný daný pruh detekovat. Přesná detekce je závislá především na správném určení binárního obrazu. Při většině situací je toto určení dostatečné pouze pomocí detektoru hran Canny. Větší stabilitu

by bylo možné zajistit komplexnějším způsobem vytváření binárního obrazu např. pomocí adaptivního filtru nebo kombinací různých detektorů hran.

Ani jeden algoritmus není konstruovaný na to, aby detekoval více jízdních pruhů zároveň. Kvůli tomu dochází k chybným detekcím při přejíždění z jednoho pruhu do druhého. Z některých snímků je patrné, že polynomická regrese 2. stupně není pro přesné určení jízdního pruhu dostatečná. S vyššími stupni polynomu však roste výpočetní náročnost modelu.

5.3 FUNKČNOST NAVRŽENÉHO SYSTÉMU

Po nainstalování, zapojení počítače Jetson Nano a nastavení kamery ZED je možné spustit algoritmy pro detekci jízdních pruhů pomocí externího notebooku. Pro zobrazení výsledků detekce lze sdílet obrazovku Jetson Nano pomocí VNC.

Jeden z problémů navrženého systému bylo polohování kamery. To bylo vyřešeno pomocí držáku na mobilní telefon umístěného v prostoru pod zrcátkem. Z praktického hlediska se jednalo o nevyhovující umístění, které částečně omezovalo výhled řidiče. Druhým problémem bylo samotné nasměrování kamery, kdy při jejím nastavení docházelo k odleskům od čelního skla. Zároveň byl obraz ovlivňován odrazy z kapoty a jasem oblohy. Všechny zmíněné problémy jsou v praxi řešeny zabudováním kamery přímo do držáku zpětného zrcátka.

Při běhu neuronových sítí na Jetson Nano dochází ke značnému navýšení příkonu zařízení. Jelikož nebyl použit doporučený zdroj, je možné, že při některých operacích může dojít k překročení limitů zařízení a celý operační systém Jetson Nano by spadl. Během testování k pádu nedošlo, ale je potřeba tuto skutečnost brát v potaz. V případě vzniklých problémů existují minimálně dvě řešení: přepnutí Jetson Nano do 5Wattového módu, které omezí výkon zařízení, nebo pořízení doporučeného zdroje.

Z celkového pohledu je navržený systém funkční a schopný rozpoznávat jízdní pruhy před vozidlem. Optimalizací algoritmu, volbou výkonnějšího počítače a zabudováním kamery do automobilu by bylo možné dostáhnout lepší spolehlivosti, plynulosti a použitelnosti systému v praxi.

ZÁVĚR

V diplomové práci byly popsány základní principy autonomních systémů s důrazem na detekci jízdních pruhů. Byly zde objasněny metody tradičního počítačového vidění, které využívají matematických operací k získávání informací z obrazu. Dále byly popsány konvoluční neuronové sítě, které jsou stále více nasazovány v souvislosti získávání informací z obrazu.

V praktické části byl vytvořen systém pro detekci jízdních pruhů v reálném prostředí. Celý systém se skládá ze stereo kamery ZED a navrženého algoritmu, který běží na počítači Jetson Nano. V rámci diplomové práce byly vytvořeny dva algoritmy. První algoritmus využívá tradičních metod počítačového vidění. Tento algoritmus v první fázi co nejrychleji zjednodušuje obraz, ve kterém následně nalezne pixely, které odpovídají jízdním pruhům. Poslední fází je vykreslení polynomu z nalezených bodů. Druhý algoritmus je založený na segmentaci obrazu pomocí konvolučních neuronových sítí. Běh tohoto algoritmu lze rozdělit do několika fází. V první fázi je obraz upraven tak, aby odpovídal vstupům CNN, následuje samotná neuronová síť a v poslední fázi je ze segmentovaného jízdního pruhu vytvořen polynom, který odpovídá vodorovnému značení.

Oba tyto algoritmy byly srovnány z hlediska jejich charakteristických vlastností. Bylo zjištěno, že první navržený algoritmus dosahoval lepších výpočetních časů, ale kvůli jeho stavbě je citlivý na změnu obrazu. Při změnách polohy kamery či změně jasu bylo potřeba pro správnou detekci jízdních pruhů změnit některé parametry algoritmu. To z něj tedy činí prakticky ne velmi dobře aplikovatelný algoritmus. Druhý navržený algoritmus dosahoval téměř dvojnásobně dlouhého výpočetního času, což je v případě autonomního řízení zcela nežádoucí, kdy vozidlo při dálničních rychlostech ujede desítky metrů za sekundu. Tento problém vznikl nevhodně navrženou kombinací CNN a post-processingu. Lepší optimalizací algoritmu nebo navržením jiné CNN by bylo možné tento problém vyřešit. Nespornou výhodou algoritmu využívající CNN je, že výsledky detekce jsou odolnější vůči lokálním obrazovým změnám. Další výhodou je možnost upravení velikosti sítě, a tím ovlivňovat rychlost i kvalitu detekce. Rozšířením trénovacího datasetu by bylo možné zlepšit přesnost a spolehlivost detekce jízdních pruhů za různých dopravních situací (změna počasí, stíny, nerovnosti na vozovce atd.). Pro testování algoritmů a především trénování CNN byl vytvořen dataset, který zachycuje komunikace v okolí Fakulty strojního inženýrství v Brně.

Rozšiřování technologií autonomního řízení do praxe má pozitivní dopad na bezpečnost silničního provozu a také na samotný vývoj těchto systémů. Díky tomu se stávají levnější a dostupnější pro běžné uživatele. Je však třeba brát v potaz, že samořiditelná vozidla nejsou zatím schopna pracovat bez lidského dohledu. V budoucnu mohou zcela změnit dopravu, jak ji známe dnes. Stále se však jedná se o komplexní problém, který přesahuje do několika oblastí výzkumu.

POUŽITÉ INFORMAČNÍ ZDROJE

[1] NAROTE, Sandipann P., Pradnya N. BHUJBAL, Abbhilasha S. NAROTE a Dhiraj M. DHANE. A review of recent advances in lane detection and departure warning system. *Pattern Recognition* [online]. 2018, **73**, 216–234. ISSN 00313203. Dostupné z: doi:10.1016/j.patcog.2017.08.014

[2] IPG - AUTOMOTIVE. *Simulation-based ADAS function development and testing* [online]. [cit. 2022-04-16]. Dostupné z: https://ipg-automotive.com/en/applications/adas/

[3] SAE J3016. *J3016 - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* [online]. duben 2021 [cit. 2022-04-15]. Dostupné z: doi:https://doi.org/10.4271/J3016_201401

[4] CAMPBELL, Sean, Niall O'MAHONY, Lenka KRPALCOVA, Daniel RIORDAN, Joseph WALSH, Aidan MURPHY a Conor RYAN. Sensor Technology in Autonomous Vehicles : A review. *29th Irish Signals and Systems Conference, ISSC 2018* [online]. 2018. Dostupné z: doi:10.1109/ISSC.2018.8585340

[5] SYNOPSYS. *What is ADAS (Advanced Driver Assistance Systems)? – Overview of ADAS Applications | Synopsys* [online]. 2022 [cit. 2022-04-15]. Dostupné z: https://www.synopsys.com/automotive/what-is-adas.html

[6] STEREOLABS. *Stereolabs - Capture the World in 3D* [online]. [cit. 2022-04-15]. Dostupné z: https://www.stereolabs.com/

[7] VELODYNE LIDAR, Inc. *Guide to LiDAR Wavelengths - Velodyne LiDAR* [online]. [cit. 2022-04-15]. Dostupné z: https://velodynelidar.com/blog/guide-to-lidar-wavelengths/

[8] BOSCH, GmbH Robert. Bosch Mobility Solutions iBooster. *Bosch Mobility Solutions* [online]. [cit. 2022-04-15]. Dostupné z: https://www.bosch-mobility-solutions.com/en

[9] CHEVROLET. *Guide to Vehicle Safety Features* [online]. [cit. 2022-04-16]. Dostupné z: https://www.chevroletarabia.com/ye-en/why-chevrolet/safety

[10] WAYKOLE, Swapnil, Nirajan SHIWAKOTI a Peter STASINOPOULOS. Review on lane detection and tracking algorithms of advanced driver assistance system. *Sustainability* [online]. 2021, **13**(20). ISSN 20711050. Dostupné z: doi:10.3390/su132011417

[11] BEBIS, George, Dwight EGBERT a Mubarak SHAH. Review of computer vision education. *IEEE Transactions on Education* [online]. 2003, **46**(1), 2–21. ISSN 00189359. Dostupné z: doi:10.1109/TE.2002.808280

[12] DING, Lijun a Ardeshir GOSHTASBY. Computer Science and Engineering Department Wright State University. *Pattern Recognition*. 2000, (34), 721–725.

[13] MUKHOPADHYAY, Priyanka a Bidyut B. CHAUDHURI. A survey of Hough Transform. *Pattern Recognition* [online]. 2015, **48**(3), 993–1010. ISSN 00313203.

Dostupné z: doi:10.1016/j.patcog.2014.08.027

[14] VOLNÁ, Eva. NEURONOVÉ SÍTĚ 1. 2008, Ostravská univerzita v Ostravě.

[15] LI, Zewen, Fan LIU, Wenjie YANG, Shouheng PENG a Jun ZHOU. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2021, 1–21. ISSN 2162-237X. Dostupné z: doi:10.1109/tnnls.2021.3084827

[16] LIU, Weibo, Zidong WANG, Xiaohui LIU, Nianyin ZENG, Yurong LIU a Fuad E. ALSAADI. A survey of deep neural network architectures and their applications. *Neurocomputing* [online]. 2017, **234**, 11–26. ISSN 18728286. Dostupné z: doi:10.1016/j.neucom.2016.12.038

[17] O'SHEA, Keiron a Ryan NASH. An Introduction to Convolutional Neural Networks [online]. 2015. Dostupné z: http://arxiv.org/abs/1511.08458

[18] ALBAWI, Saad, Tareq Abed MOHAMMED a Saad AL-ZAWI. Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017* [online]. 2018, 1–6. Dostupné z: doi:10.1109/ICEngTechnol.2017.8308186

[19] GUO, Tianmei, Jiwen DONG, Henjian LI a Yunxing GAO. Simple convolutional neural network on image classification. *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017* [online]. 2017, 721–724. Dostupné z: doi:10.1109/ICBDA.2017.8078730

[20] PODAREANU, Damian, Valeriu CODREANU, Sandra AIGNER a Caspar Van Leeuwen EDITOR. Best Practice Guide - Deep Learning. *ResearchGate* [online]. 2019, (February), 1–51. Dostupné z: doi:10.13140/RG.2.2.31564.05769

[21] TANG, Jigang, Songbin LI a Peng LIU. A review of lane detection methods based on deep learning. *Pattern Recognition* [online]. 2021, **111**. ISSN 00313203. Dostupné z: doi:10.1016/j.patcog.2020.107623

[22] HUANG, Yuhao, Shitao CHEN, Yu CHEN, Zhiqiang JIAN a Nanning ZHENG. Spatial-temproal based lane detection using deep learning. *IFIP Advances in Information and Communication Technology* [online]. 2018, **519**, 143–154. ISSN 18684238. Dostupné z: doi:10.1007/978-3-319-92007-8_13

[23] WU, Ye, Yuetian SHAN, Yunhe XU, Shu WANG a Yuchen ZHUANG. The implementation of lane detective based on OpenCV. In: *Proceedings - 2010 2nd WRI Global Congress on Intelligent Systems, GCIS 2010* [online]. 2010, s. 278–281. ISBN 9780769543048. Dostupné z: doi:10.1109/GCIS.2010.120

[24] SON, Jongin, Hunjae YOO, Sanghoon KIM a Kwanghoon SOHN. Real-time illumination invariant lane detection for lane departure warning system. *Expert Systems with Applications* [online]. 2015, **42**(4), 1816–1824. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2014.10.024

[25] ZHANG, Shengchang, Ahmed EI KOUBIA a Khaled Abdul Karim MOHAMMED.

Traffic Lane Detection using FCN [online]. 2020. Dostupné z: http://arxiv.org/abs/2004.08977

[26] ZHANG, Ce, Yu HAN, Dan WANG, Wei QIAO a Yier LIN. A Network That Balances Accuracy and Efficiency for Lane Detection. *Mobile Information Systems* [online]. 2021, **2021**, 1–5. ISSN 1875-905X. Dostupné z: doi:10.1155/2021/1099434

[27] WANG, Chi-feng. *A Basic Introduction to Separable Convolutions* [online]. [cit. 2022-04-20]. Dostupné z: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

[28] BÄUERLE, Alex, Christian VAN ONZENOODT a Timo ROPINSKI. Net2Vis – A Visual Grammar for Automatically Generating Publication-Tailored CNN Architecture Visualizations. *IEEE Transactions on Visualization and Computer Graphics* [online]. 2021, **27**(6), 2980–2991. Dostupné z: doi:10.1109/TVCG.2021.3057483

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

Zkratky

2D	Označení dvourozměrného prostoru
3D	Označení trojrozměrného prostoru
Adam	Adaptive moment estimation
CNN	Convolutional neural network
CUDA	Compute unified device architecture
CV	Computer vision
FCN	Fully convolutional network
FPS	Frames per second
L4T	Linux for Tegra
OpenCV	Open Source Computer Vision Library
ReLU	Rectified linear unit
RT	Real time
SCNN	Spital Convolutional Neural Network
SDK	Software Development Kit
STLNet	Spatial-Temporal Lane detection Network
VNC	Virtual Network Computing

SEZNAM PŘÍLOH

- Algoritmus pro detekci jízdních pruhů založený na metodách počítačového vidění
- Algoritmus pro detekci jízdních pruhů založený na konvoluční neuronové síti
- Sada skriptů pro přípravu datasetu