



**FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI**

**KATEDRA
KYBERNETIKY**

Bakalářská práce

**Analýza systémů pro udržení vozidla
v jízdním pruhu**

Filip Jašek



FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Analýza systémů pro udržení vozidla v jízdním pruhu

Filip Jašek

Vedoucí práce

Ing. Petr Neduchal, Ph.D.

© Filip Jašek, 2023.

Všechna práva vyhrazena. Žádná část tohoto dokumentu nesmí být reprodukována ani rozšiřována jakoukoli formou, elektronicky či mechanicky, fotokopírováním, nahráváním nebo jiným způsobem, nebo uložena v systému pro ukládání a vyhledávání informací bez písemného souhlasu držitelů autorských práv.

Citace v seznamu literatury:

JASEK, Filip. *Analýza systémů pro udržení vozidla v jízdním pruhu*. Plzeň, 2023. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Petr Neduchal, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Filip JAŠEK**
Osobní číslo: **A19B0293P**
Studijní program: **B0714A150005 Kybernetika a řídicí technika**
Specializace: **Automatické řízení a robotika**
Téma práce: **Analýza systémů pro udržení vozidla v jízdním pruhu**
Zadávací katedra: **Katedra kybernetiky**

Zásady pro vypracování

1. Proveďte rešerši v oblasti udržení vozidla v jízdním pruhu.
2. Zaměřte se jak na algoritmické řešení, tak i na moderní přístupy založené na hlubokém učení.
3. Proveďte rešerši dostupných datasetů a simulátorů.
4. Proveďte jednotlivé systémy, respektive přístupy k řešení úlohy.
5. Implementujte jedno algoritmické řešení a jedno založené na hlubokém učení. Implementace porovnejte.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

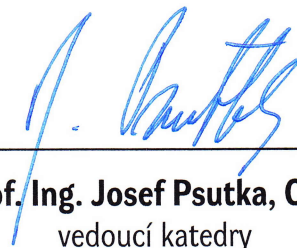
1. McCall, J. C., & Trivedi, M. M. (2006). Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. IEEE transactions on intelligent transportation systems, 7(1), 20-37.
2. Tang, J., Li, S., & Liu, P. (2021). A review of lane detection methods based on deep learning. Pattern Recognition, 111, 107623.
3. Zhang, Y., Lu, Z., Zhang, X., Xue, J. H., & Liao, Q. (2021). Deep learning in lane marking detection: a survey. IEEE Transactions on Intelligent Transportation Systems.

Vedoucí bakalářské práce: **Ing. Petr Neduchal**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2022**
Termín odevzdání bakalářské práce: **22. května 2023**



Doc. Ing. Miloš Železný, Ph.D.
děkan



Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 14. srpna 2023

.....

Filip Jašek

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Cílem bakalářské práce je analyzovat systémy, které lze použít pro udržení vozidla v jízdním pruhu na základě vizuálního senzoru - kamery. Analýza se skládá z obecného seznámení s tématem, představení jednotlivých přístupů, jejich slabin a relevantních simulátorů a datasetů. Tři hlavní metody jsou teoreticky více rozebrány a dvě z nich jsou implementovány a otestovány v praktické části na simulačním příkladu diferenciálního dvoukolového robota. Jedna je založena na algoritmických metodách zpracování obrazu a druhá na modernějším přístupu segmentace obrazu pomocí neuronové sítě, konkrétně U-Net. V rámci praktické demonstrace dvou zmíněných metod bylo provedeno i jejich srovnání s manuálně vytvořeným vzorem. Hlavním přínosem této práce je analýza přístupů k autonomnímu řízení na zjednodušené úloze udržení vozidla v jízdním pruhu a seznámení s překážkami v této oblasti.

Abstract

This bachelor's thesis aims to analyze systems that can be used for vehicle lane-keeping based on the visual sensor - a camera. The analysis consists of a general introduction to the topic, presentation of individual approaches, their weaknesses and relevant simulators and datasets. Three main methods are theoretically examined in more detail, and two of them are implemented and tested in the practical part of this thesis using a simulation example of a differential two-wheeled robot. One is based on algorithmic image processing methods, and the other one is on more modern approach of image segmentation using a neural network, specifically U-Net. As part of the practical demonstration of these two methods, a comparison was also conducted with a manually created template. The main contribution of this work is the analysis of approaches to autonomous driving in the simplified task of lane-keeping and familiarization with the challenges in this field.

Klíčová slova

autonomní řízení • simulátor • dataset • robot • zpracování obrazu • neuronová síť

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Petru Neduchalovi, Ph.D. za poskytnuté vybavení, věnovaný čas, trpělivost a cenné rady, které mi pomohly práci dokončit. Dále bych chěl poděkovat za konzultace řízení vozítka Ing. Martinu Goubeji, Ph.D. a Doc. Ing. Ondřeji Strakovi, Ph.D. V neposlední řadě děkuji za podporu a péči také své rodině a přítelkyni.

Obsah

| | |
|---------------------------------------------------|-----------|
| 1 Úvod | 3 |
| 1.1 Stupně autonomie | 4 |
| 2 Teoretická část - analýza přístupů | 7 |
| 2.1 Algoritmické řešení | 8 |
| 2.1.1 Stanovení modelu silnice | 8 |
| 2.1.2 Extrakce značení cesty | 11 |
| 2.1.3 Závěrečné zpracování | 13 |
| 2.1.4 Ovládání pohybu vozítka | 14 |
| 2.1.5 Shrnutí | 15 |
| 2.2 Řešení pomocí zpětnovazebního učení | 15 |
| 2.2.1 Value based | 17 |
| 2.2.2 Policy based | 17 |
| 2.2.3 Metoda actor-critic | 18 |
| 2.2.4 Shrnutí | 18 |
| 2.3 Řešení pomocí hlubokého učení | 19 |
| 2.3.1 Klasifikace obrazu | 20 |
| 2.3.2 Detekce objektů | 20 |
| 2.3.3 Sémantická segmentace obrazu | 21 |
| 2.3.4 Shrnutí | 21 |
| 2.4 Porovnání přístupů | 22 |
| 3 Simulátory a datasety | 25 |
| 3.1 Simulátory | 25 |
| 3.1.1 Gym-duckietown | 25 |
| 3.1.2 AirSim | 26 |
| 3.1.3 Carla | 27 |
| 3.1.4 Shrnutí | 27 |
| 3.2 Datasety | 28 |
| 3.2.1 Shrnutí | 30 |

| | |
|-------------------------------------------------|-----------|
| 4 Praktická část - řešení | 31 |
| 4.1 Použité prostředky | 31 |
| 4.2 Předzpracování | 32 |
| 4.2.1 Radiální zkreslení | 32 |
| 4.2.2 Tangenciální zkreslení | 33 |
| 4.2.3 Korekce zkreslení | 33 |
| 4.3 Algoritmické řešení | 34 |
| 4.3.1 Extrakce značení cesty | 35 |
| 4.3.2 Regulace pohybu | 40 |
| 4.4 Deep learning řešení - segmentace | 41 |
| 4.4.1 Tvorba datasetu | 41 |
| 4.4.2 Trénování modelu | 41 |
| 4.4.3 Predikce | 42 |
| 4.5 Zhodnocení | 43 |
| 5 Závěr | 47 |
| Bibliografie | 49 |
| Seznam obrázků | 57 |
| Seznam tabulek | 59 |

Autonomní řízení je zajímavou avšak náročnou vědní disciplínou. Mnoho firem a institucí vyvíjí mimořádnou snahu automatizovat dopravu a eliminovat tím tak negativní dopady lidských chyb a nepředvídatelných reakcí. Přes veškeré dostupné technologie, naráží systémy na překážky v podobě nehomogenních podmínek popsaných v článku [1] nebo nepředvídatelného chování ostatních účastníků provozu [2, 3]. Ačkoliv jsou v moderních vozidlech nejrůznější asistenční systémy již téměř standardem, jen malá část z nich je vybavena systémem schopným autonomního řízení. Za zmínku stojí americká automobilka Tesla¹, která jako jedna z mála nabízí za příplatek i možnost uživatelského přístupu k autonomnímu systému autopilot a využívat ho na veřejných komunikacích. Bohužel však stále není možné ho využívat bez pozornosti řidiče ani vyloučit možné kolize. Ty však nemusí být způsobeny samotným systémem, ale třeba i jiným vozidlem, cyklistou, nebo dokonce převzetím řízení člověkem.

Nejen s technologickými problémy je nutno bojovat, aby se prosadilo větší rozšíření těchto vozidel. Všeobecná skepse vůči autonomii v mobilitě vyplývající z analýzy sociální sítě Reddit [4] nepřispívá k pozitivnímu pohledu na tuto technologii. Původcem nedůvěry nebo obav z ní je způsobeno zejména mylným povědomím a nedostatečnou informovaností o její bezpečnosti [3].

Typickou predikcí budoucnosti mobility jsou právě autonomní prostředky, které budou schopny dopravovat lidi, kteří se mezitím budou věnovat například práci, odpočinku nebo jiným činnostem bez stresu z denního dojíždění. Dnes sice tato budoucnost stále není skutečností, ale technologickými pokroky se dostáváme stále blíže. Výrazný posun zajistilo nejen nedávné rozšíření neuronových sítí ale také již existující matematické metody a algoritmy, které se používají často i v kombinaci s neuronovými sítěmi.

V práci bude nejprve popsána samotná autonomie. Další kapitola se bude věnovat teoretickému principu jednotlivých přístupů k úloze detekce jízdního pruhu. V neposlední řadě budou představeny relevantní simulátory a datasey. Na závěr

¹<https://www.tesla.com>

budou porovnána dvě prakticky předvedená řešení na vzorovém datasetu.

1.1 Stupně autonomie

Protože pro autonomní systémy existuje celá řada definic, které popisují jejich různě sofistikované varianty, byla již v roce 2014 stanovena standardní definice 6ti stupňů autonomie organizací SAE² často odkazovaná jako "SAE Levels of Driving Automation™" od úrovně 0 (bez automatizace) do úrovně 5 (plně autonomní). Od té doby došlo k několika revizím. Poslední proběhla v roce 2021 a její plné znění lze získat na oficiálních stránkách společnosti [5]. Oficiální 41 stránkový dokument obsahuje detailní definice rolí řidiče a systému pro jednotlivé stupně. K obecnému porozumění tohoto rozdělení postačí zjednodušený graf na Obrázku 1.1.

SAE J3016™ LEVELS OF DRIVING AUTOMATION™
 Learn more here: [sae.org/standards/content/j3016_202104](https://www.sae.org/standards/content/j3016_202104)

Copyright © 2021 SAE International. The summary table may be freely copied and distributed AS-IS provided that SAE International is acknowledged as the source of the content.

| | SAE LEVEL 0™ | SAE LEVEL 1™ | SAE LEVEL 2™ | SAE LEVEL 3™ | SAE LEVEL 4™ | SAE LEVEL 5™ |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| What does the human in the driver's seat have to do? | You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering | | | You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat" | | |
| | You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety | | | When the feature requests, you must drive | These automated driving features will not require you to take over driving | |
| <small>Copyright © 2021 SAE International.</small> | | | | | | |
| | These are driver support features | | | These are automated driving features | | |
| What do these features do? | These features are limited to providing warnings and momentary assistance | These features provide steering OR brake/acceleration support to the driver | These features provide steering AND brake/acceleration support to the driver | These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met | | This feature can drive the vehicle under all conditions |
| Example Features | <ul style="list-style-type: none"> • automatic emergency braking • blind spot warning • lane departure warning | <ul style="list-style-type: none"> • lane centering OR • adaptive cruise control | <ul style="list-style-type: none"> • lane centering AND • adaptive cruise control at the same time | <ul style="list-style-type: none"> • traffic jam chauffeur | <ul style="list-style-type: none"> • local driverless taxi • pedals/steering wheel may or may not be installed | <ul style="list-style-type: none"> • same as level 4, but feature can drive everywhere in all conditions |

Obrázek 1.1: Graf zobrazující úrovně automatizace od 0 (bez automatizace) do 5 (plně autonomní). Převzat ze stránek společnosti SAE viz. [6].

Na základě zmíněné stupnice se podle pravidel pro jednotlivé stupně udělí systému odpovídající hodnocení. Tesla autopilot, považovaný za jeden z nejlepších, dosáhl dle článku [7] v tomto žebříčku pouze na stupeň 2 přidělený institucí "National Highway Traffic Safety Administration". Z toho je patrné, že je vývoj stále na

²Society of Automobile Engineers

počátku. Díky zvyšující se implementaci asistenčních systémů do běžně dostupných aut lze ale očekávat, že se počet vozidel se stupněm autonomie větší než 0 bude zvyšovat.

Jízdní asistenti v dnešní době běžně pomáhají varovat řidiče při překročení jízdních pruhů, přílišnému přiblížení, hlídají mrtvý úhel a mnoho dalšího. Většinu těchto asistenčních systémů lze zařadit do prvního stupně autonomie. Jde tedy o systémy, které dokážou řidiče pouze varovat nebo zasáhnout v omezeném rozsahu do řízení a to pouze pod jeho dozorem.

Zatímco výše zmíněné systémy slouží spíše pro podporu řidiče a zvýšení bezpečnosti, v systémech s autonomií stupně 3 až 5 se z člověka stává pasažér, kdy automobil zastává veškeré povinnosti řidiče. Tato práce se zabývá problematikou udržení vozidla v jízdním pruhu, jež je součástí systému ve všech stupních kromě 0. Z toho je patrné, že se jedná o jeden ze stavebních kamenů celého autonomního řízení vozidla.

Teoretická část - analýza přístupů

2

V následujícím textu bude nejdříve podrobně rozebrán problém asistenta pro udržení vozidla v pruhu a popsány nejpoužívanější přístupy. Jedná se o řešení algoritmické, řešení založené na zpětnovazebním učení¹ a nakonec řešení pomocí hlubokého učení² na bázi neuronových sítí.

I přes to, že se zdá jízda v pruhu jako problém s jednoduchým řešením, není to tak úplně pravda. Pro účely představení jednotlivých technologií bude problém zjednodušen předpokladem ideálních podmínek a zanedbáním variability okolního prostředí.

Složitost problému bude ilustrována na následujícím příkladu. Při vjezdu do zatáčky, řidič podle jejího tvaru zvolí vhodnou reakci (zpomalení, zatočení apod.) a zatáčku bezpečně projede. V popsané situaci se ale děje daleko více než se na první pohled zdá. Vše stojí na pozorování právě probíhající situace okolo automobilu a jeho samotného chování. Řidič musí mít prvně znalost o autě samotném, jeho velikosti, hmotnosti, výkonu, citlivosti řízení, brzd. Dále pomocí svého zraku pozoruje silnici a polohu, ve které se nachází a na získané informace reaguje zatočením volantu, sešlápnutím pedálu plynu nebo brzdy a pozoruje, jak se změnila zmíněná parametry. Tento cyklus člověk při řízení zpočátku zaznamená, ale po jisté době ho přestává vnímat vědomě.

Správnou otázkou zde ale je, jak pozná, kudy vede cesta a jak vlastně vypadá. Pro člověka je to přirozená věc, nad kterou se ani nepozastaví, ale stroji působí značný problém to rozpoznat. Prvně zmíněné reakce člověka lze totiž implementovat pomocí již známých regulačních smyček a modelů, ale spolehlivé nalezení samotné cesty není již tak přímočarý problém.

¹v literatuře známé jako reinforcement learning

²deep learning

2.1 Algoritmické řešení

Již dříve se lidé zajímali o nejrůznější metody autonomního řízení vozidel. Například nemocnice Motol v Praze nebo firma Amazon ve svých skladech dokonce implementovaly funkční systém autonomních vozítek na bázi sledování předem vymezené trasy zabudované v podlaze, kterou sledují pomocí odpovídajících senzorů. Tyto metody však vyžadují připravené prostředí a na úpravu již existující sítě pozemních komunikací by muselo být vynaloženo značné množství nákladů, aby navíc odolalo vlivům počasí, kterého jsou systémy ve skladech a uzavřených budovách ušetřeny.

Nynější komunikace jsou vytvořeny s ohledem na to, že se po nich budou řídit pohybovat na základě vizuální zpětné vazby. Výhradní použití ultrazvukových, radarových nebo lidarových senzorů se tedy pro řešení tohoto problému nehodí ačkoliv pro některé úlohy mohou být zásadní. Na základě těchto skutečností se jako nejpraktičtější jeví použití kamer.

Algoritmický přístup spočívá ve zpracování jednotlivých kamerových snímků za pomoci zavedených algoritmů do formy vhodné pro regulátor, který vytváří akční zásahy pro ovládání vozidla.

Obvyklý postup implementace popsany v [1] se skládá z kroků popsanych v samostatných kapitolách 2.1.1 až 2.1.3.

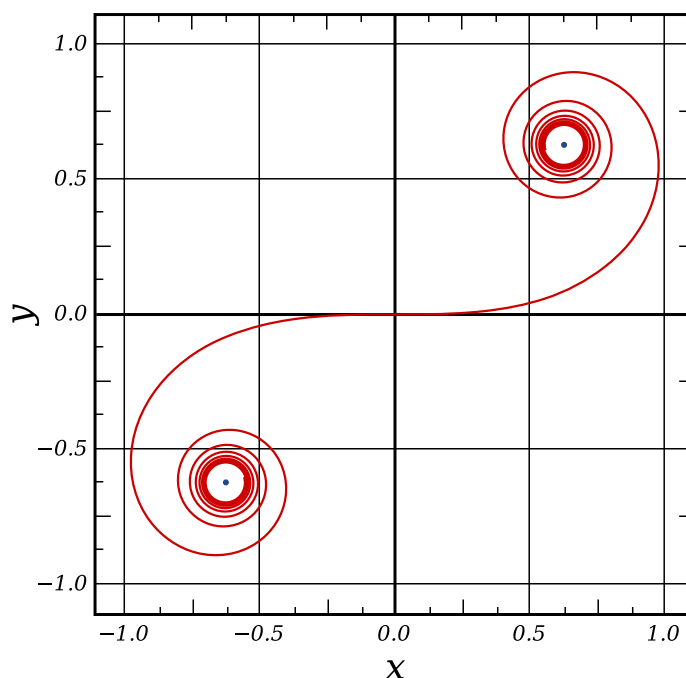
2.1.1 Stanovení modelu silnice

Pro eliminaci chyb a šumu v obrazu je vhodné předem určit model silnice, po které se bude vozidlo pohybovat. Vhodným modelem je pro většinu moderních silnic klotoid³ používaný při stavbě rychlostních cest a dálnic, popsán rovnicí 2.1.

Při optimalizaci detekce pro dálnici lze uvažovat, že na zpracovávaném úseku budou hranice cesty tvořit rovnoběžné přímky. Stejný model však nebude dobře aproximovat trasy s prudšími zatáčkami a bude potřeba použít jiný. Při výběru je nutné zohlednit i maximální rychlost, podle které je nutné přizpůsobit nejen viditelnost, ale právě i model. Pokud se totiž vozidlo bude pohybovat velmi malou rychlostí, lze použít jednodušší lineární model, protože na krátkém úseku lze potřebnou část cesty aproximovat opět přímkou i když se jedná o zatáčku.

Dalším parametrem je zmíněná viditelnost, která může být ovlivněna reakční dobou systému za kterou by měl vozidlo zastavit nebo vyvinout zásah eliminující nenadálé krizové situace. V tomto případě je nutné v modelu uvažovat i fyzikální vlastnosti samotného vozidla. Tato práce se, ale bude zabývat primárně problematikou detekce pruhů z kamery, pro kterou není takto pokročilý model potřeba.

³křivka nazývaná také jako Eulerova



Obrázek 2.1: Eulerova křivka.

Klotoida (viz. Obrázek 2.1) používaná pro návrh silnic, dálnic nebo horských drah pomáhá svým tvarem eliminovat rázové působení přetížení v zatáčkách a rozprostřít tak působící síly po celé délce dráhy.

Následující odvození aproximace klotoidy pro zjednodušení výpočtů bylo inspirováno článkem [8]. Rovnici 2.1 používanou pro popis této nekonečné křivky

$$R_i L_i = A^2, \quad (2.1)$$

kde R_i je radius křivky v daném bodě i , L_i její délka od počátku a A je parametr určující poměrnou velikost křivky.

Diferenciál celkové délky křivky je pak díky znalostem z obrázku 2.2 vyjádřen jako

$$dL = R d\tau, \quad (2.2)$$

dosazením za R vznikne

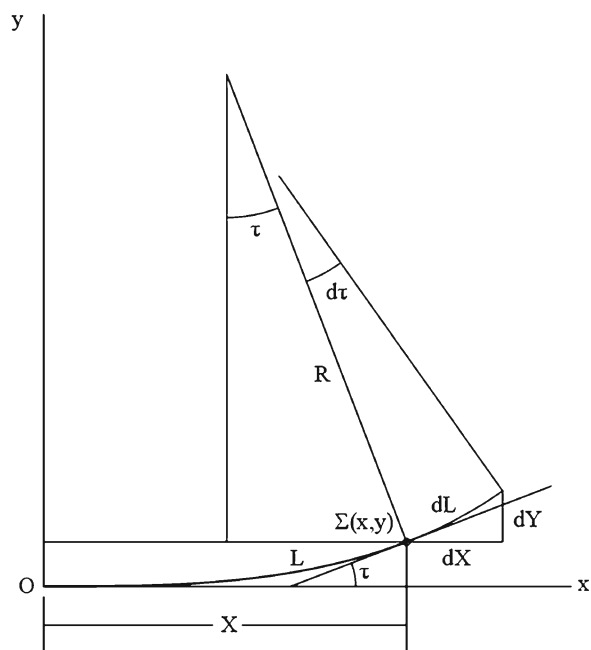
$$L dL = A^2 d\tau. \quad (2.3)$$

Integrací je docíleno

$$\frac{L^2}{2} = A^2 \tau + C, \quad (2.4)$$

kde C je integrační konstanta a vyjádřením úhlu tečny v obloukové míře τ vůči ose x jako

$$\tau = \frac{L^2}{2A^2}. \quad (2.5)$$



Obrázek 2.2: Část Eulerovy křivky.

S jeho použitím budou difference souřadnic rovny

$$dX = dL \cos \frac{L^2}{2A^2} \quad (2.6)$$

$$dY = dL \sin \frac{L^2}{2A^2}, \quad (2.7)$$

integrací za pomoci tzv. Fresnelových integrálů

$$x = \int_0^L \cos\left(\frac{l^2}{2A^2}\right) dl \quad (2.8)$$

$$y = \int_0^L \sin\left(\frac{l^2}{2A^2}\right) dl \quad (2.9)$$

a rozvojem v Taylorovu řadu

$$x = l - \frac{l^5}{40(A)^4} + \frac{l^9}{3456(A)^8} - \dots \quad (2.10)$$

$$y = \frac{l^3}{6(A)^2} - \frac{l^7}{336(A)^6} + \frac{l^{11}}{42240(A)^{10}} - \dots \quad (2.11)$$

Zachováním pouze prvních členů vznikne soustava rovnic

$$x = l \quad (2.12)$$

$$y = \frac{l^3}{6(A)^2} \quad (2.13)$$

jejíž řešením je kubická rovnice

$$y = \frac{x^3}{6(A)^2} \quad (2.14)$$

která říká, že klotoidu lze aproximovat polynomem třetího stupně

$$y = kx^3, \quad (2.15)$$

což značně zjednoduší budoucí výpočty z původní numerické integrace.

2.1.2 Extrakce značení cesty

Vzhledem k různorodému dopravnímu značení v jednotlivých státech, ale i jeho stavu v oblastech samotných států je obtížné navrhnout algoritmus spolehlivě detekující potřebné hranice cesty. Kromě samotného prostředí mohou mít na kvalitu značení vliv i parametry jako jsou stíny, déšť nebo samotná kvalita či opotřebení povrchu viz. Obrázek 2.3. To mohou být důvody proč například jednoduché metody na bázi hranové detekce selžou.

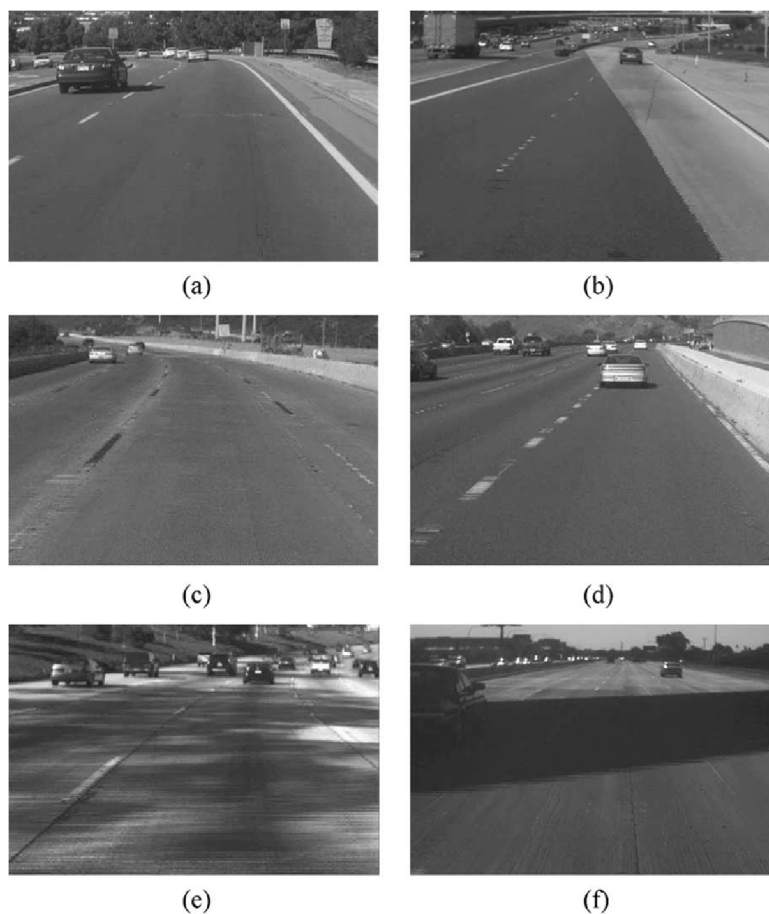
I přes použití robustnějších metod se naráží na komplikace zejména při zastínění části obrazu nejednotlivým stínem [1]. Podobně jako u volby modelu cesty může vzniknout problém při nasazování jednotlivých algoritmů v nejrůznějších prostředích. Proto je vhodné optimalizovat detekci čar na omezenou oblast a získat tak kvalitnější výsledky v definovaném prostředí.

Díky vysokému kontrastu silničního značení jsou obvykle pro jejich nalezení používány hranové detektory. Pro praktickou ukázkou byla zvolena metoda směrových tzv. "steerable" filtrů [9], které detekují hrany v požadovaném směru. Jejich aplikace probíhá konvolucí obrázku s jádrem viz. Obrázek 2.4 blíže popsán v článku [10]. Abychom ho získali, je potřeba derivovat kruhově symetrické jádro s Gaussovým rozložením viz. Obrázek 2.5. To se používá velmi často k vyhlazování a rozostřování obrázků pro odstranění šumu. Vypočte se pomocí následující rovnice

$$g(x, y) = a \cdot \exp \left(- \left(\frac{(x - x_0)^2}{2\sigma_X^2} + \frac{(y - y_0)^2}{2\sigma_Y^2} \right) \right), \quad (2.16)$$

kde x_0 a y_0 jsou střední hodnoty, σ_X a σ_Y jsou rozptyly a a je amplituda.

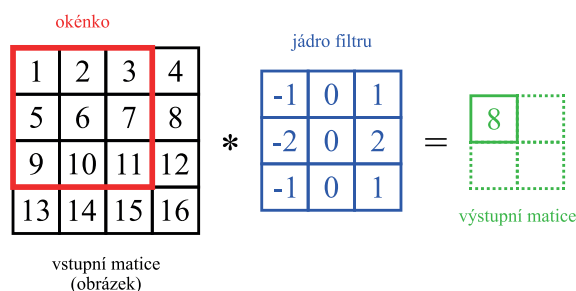
Derivací daného jádra v požadovaném směru, vznikne orientované jádro neboli "steerable filter". V závislosti na směru derivace se mění jeho působení a rozptylem



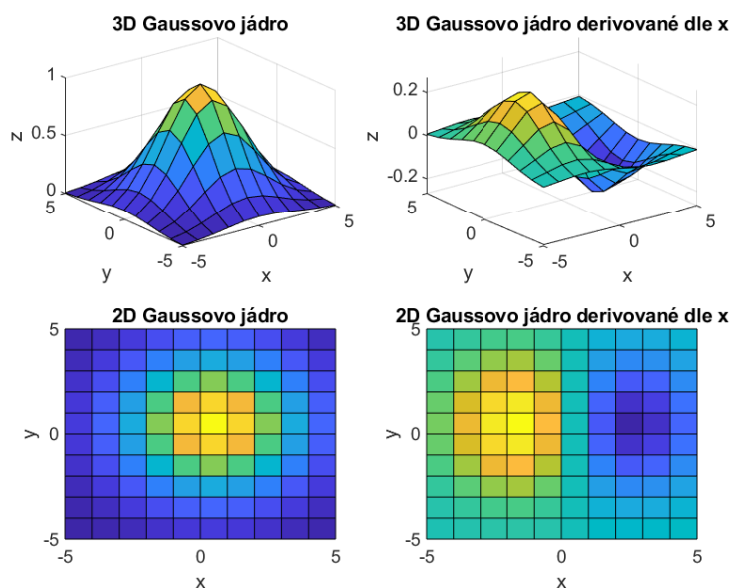
Obrázek 2.3: Fotografie různých variant stavu značení nebo podmínek. Cesta (a) zobrazující optimální plnou a přerušovanou čáru, (b) nejednotlivý povrch vozovky, (c) poškozené horizontální značení, (d) špatná kvalita čar a přítomnost stínů (e) a (f). Zdroj [1].

σ spolu s amplitudou a je ovlivňována jeho intenzita. V literatuře se lze také setkat s jinými jádry detekujícími hrany v určitém směru např. Sobel filtrem [11], který je velikosti 3×3 a jednu jeho formu pro osu x lze vidět na Obrázku 2.4.

Aplikace zmíněného filtru detekuje v obrázku silnice místa s největším gradientem v daných směrech. Bohužel zde nastává problém nežádoucího šumu zmíněného na počátku této sekce.



Obrázek 2.4: Vizualizace základní konvoluce na 2D matici reprezentující obrázek.

Obrázek 2.5: Vizualizace Gaussova kruhově symetrického jádra (nalevo) a jeho derivace podle x (napravo) ve 3D i 2D perspektivě.

2.1.3 Závěrečné zpracování

Závěrečné zpracování neboli postprocessing⁴ je nezbytnou součástí algoritického řešení, neboť právě zde se detekované čáry zpracovávají a transformují do dále použitelných dat. Jedním z rozšířených metod je aplikace Houghovy transformace [12], která dokáže najít čáru i přes její horší viditelnost. Nicméně používají se i jiné metody, které dokáží plnit roli mostu mezi extrakcí značení samotným a jeho sledováním.

V této práci je použit přístup hledání bodů čáry pomocí klouzavých okének. Ten spočívá ve stanovení počátku detekované čáry, zpravidla za použití histogramu a dále postupnému skládání okének na sebe s korekcí podle směru detekovaného

⁴V zahraniční literatuře je tak nazývána závěrečná úprava finálního výstupu.

shluku pixelů reprezentující čáru. Takto nalezené shluky uvnitř okének jsou položeny kubickou rovnicí za pomoci minimalizace kritéria nejmenších čtverců [13]. Zprůměrováním získaných rovnic pro jednotlivé křivky vznikne rovnice popisující ideální trasu uprostřed detekovaných čar.

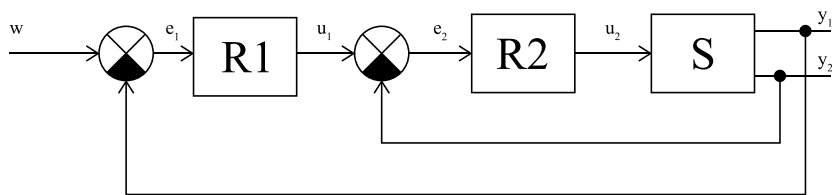
Protože jsou parametry polynomu známe, z vlastností křivky lze určit polohu počátku při dolní hranici obrazu a za pomoci derivace získat i její tečnu. Ta udává informaci o relativním natočení vozítka vzhledem k nalezené ideální trase. Tyto hodnoty budou sloužit jako vstup do regulátoru.

2.1.4 Ovládání pohybu vozítka

Pro samotný pohyb je potřeba hodnoty z minulé sekce transformovat na akční zásah akceptovatelný softwarem vozítka. V případě modelového příkladu jsou k dispozici dva akční zásahy v rozsahu $< -1, 1 >$, jeden pro každý motor odpovídajícího kola.

Použití výstupu detekce přímo k řízení vozidla není vhodné pro jeho šum nebo chybné detekce. Lepších výsledků je možné dosáhnout použitím regulátoru ke sledování reference v podobě polohy ideální trasy. K dispozici jsou dva vstupy a to natočení vozidla od tečny ideální trasy a jeho vzdálenost od počátku křivky při dolním okraji obrazu. To jsou dva parametry, které jsou důležité pro regulaci a protože se navzájem ovlivňují, jsou zavedeny jako vstup do regulátoru.

Zde narážíme na problém, kdy jeden regulátor může zpracovávat pouze jeden vstup. Volíme tedy kaskádní zapojení dvou regulátorů viz. Obrázek 2.6. Vnitřní regulátor zde bude mít jako vstup úhel natočení křivky a vnější regulátor vzdálenost od středu obrazu. Docílíme tak priority regulace natočení vozidla, která má zásadní vliv na budoucí trajektorii.



Obrázek 2.6: Blokové schéma navrženého kaskádního regulátoru s dvěma vstupy, kde R1 reprezentuje vnější a R2 vnitřní smyčku.

Popsaná regulace je velmi základní a tato práce se nebude zabývat složitějšími způsoby regulace jakožto i plánováním trasy, protože se jedná o náročnější a komplexnější problém nad rámec této bakalářské práce. V principu ale díky znalosti o pohybu vozidla v prostoru a dodatečnými informacemi o rychlosti nebo zrychlení lze škálovat zde pojednávanou problematiku k přesnější a jistější regulaci na požadovanou trajektorii.

2.1.5 Shrnutí

Ačkoliv je algoritmické řešení v základu jednoduché a pochopitelné, obsahuje řadu omezení, které musí být kompenzovány implementací složitějších funkcí. Jak bylo zmíněno v textu výše, hlavní překážkou algoritmického řešení jsou reálné podmínky, které mohou být variabilní i v rámci relativně krátkého úseku například v podobě stínů nebo mokré vozovky. Jeho velkou výhodou je ale to, že pro svůj chod ve většině případů nepotřebuje specifický hardware v podobě například grafické karty a je tak vhodný i pro méně výkonné stroje. Navíc při zajištění správných podmínek bude fungovat spolehlivě.

2.2 Řešení pomocí zpětnovazebního učení

Reinforcement learning (RL⁵) známý také jako zpětnovazební učení je specifický přístup strojového učení řešící daný problém bez přímého zásahu člověka na základě maximalizace zisku odměn přidělovaných za co nejlepší splnění specifický podmínek.



Obrázek 2.7: Schéma základní struktury algoritmu zpětnovazebního učení.

Základní struktura se skládá z agenta a prostředí viz. Obrázek 2.7. Agent je algoritmus, který volí jednotlivé kroky a interaguje s prostředím, které reprezentuje model daného problému a vrací aktuální stav (reakci) spolu s odměnou za danou akci. Tato odměna je dána hodnotící tzv. "reward" funkcí, specifikovanou podle řešeného problému. Například při použití prostředí ze hry, kde je potřeba sbírat dané předměty pro vítězství může být analogií k odměnám skóre daného hráče (agenta) a "reward" funkce zde bude počítat dosažené předměty a jejich individuální hodnotu. Pokud by tedy agent chtěl maximalizovat odměnu za danou hru může se stát, že bude sbírat pouze nejhodnotnější předměty, aby co nejrychleji nabyl nejvyšší odměny.

Jak je patrné z předešlého příkladu, hlavním parametrem ovlivňující chování agenta je "reward" funkce. Ta musí být správně specifikována a implementována,

⁵Zkratka pro zpětnovazební učení z anglického Reinforcement Learning.

čímž se vytyčí priority, které se má agent naučit. Pokud například nezahrneme čas v podobě záporné odměny (penalizace), nelze očekávat že prioritou pro agenta bude nasbírat odměny rychle.

Další důležitou volbou je struktura samotného agenta. Existují dvě možnosti. Buď implementací algoritmu nebo pomocí neuronové sítě. Algoritmus si bude pamatovat akce nebo jejich posloupnosti s nejvyšší hodnotou aby je byl schopen později aplikovat sám mimo fázi učení. Neuronová síť bude na základě odměn nastavovat své vnitřní parametry a výsledkem učení bude natrénovaný model schopný hrát danou hru nebo řešit problém. V dalších odstavcích se bude text věnovat primárně přístupu pomocí neuronových sítí neboli NN⁶.

Při standardním použití NN se používá buď učení s učitelem a nebo bez učitele, ale dle [14, p. 2] nespadá RL ani do jedné třídy jelikož učení zde stojí na přímé interakci s prostředím a zároveň se nesnaží najít strukturu v poskytnutých datech ale maximalizovat odměnu jako výsledek uskutečněných akcí.

Při použití jednoho RL agenta se při formalizaci dle [15, 14] považuje za standard Markovův rozhodovací proces (MDP). Ten se skládá ze sady stavů S , akcí A , přechodové (transakční) funkce T a funkce odměny R , které tvoří uspořádanou čtveřici (S, A, T, R) . Následně pokud ve stavu $s \in S$ bude zvolena akce $a \in A$ vznikne nový stav $s' \in S$ s pravděpodobností přechodu $T(s, a, s') \in (0, 1)$ a odměnou $r \in R(s, a)$. Stochastická policy (strategie) $\pi : S \rightarrow \mathcal{D}$ transformuje prostor stavů na podmíněnou pravděpodobnost jednotlivých akcí za podmínky stavu s , tedy $\pi(a|s)$. Cílem je tedy nalézt optimální policy (strategii) π^* která bude maximalizovat pro všechny stavy $s \in S$ sum of discounted rewards:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{H-1} \gamma^k r_{k+1} | s_0 = s \right\}, \quad (2.17)$$

kde $\gamma \in [0, 1]$ je discount factor rozhodující o váze budoucích odměn, $r_k = R(s_k, a_k)$ je odměna v čase k , H je počet časových kroků a $\mathbb{E}[\cdot]$ značí očekávanou hodnotu náhodné proměnné.

S výše zmíněnou strategií souvisí důležitý koncept action-value funkce tzv. Q-funkce:

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{H-1} \gamma^k r_{k+1} | s_0 = s, a_0 = a \right\} \quad (2.18)$$

Pro použití RL v oblasti autonomního řízení existuje více základních struktur a přístupů k řešení specifických problémů jak je popsáno ve článku [15]. Základní možnosti návrhu struktury lze rozdělit na dva přístupy, tzv. Value based a Policy based.

⁶Zkratka pro neuronové sítě z anglického Neural Networks

2.2.1 Value based

Q-learning je jeden z nejpoužívanějších RL algoritmů, který se učí za pomoci odhadu vlivu dvojice stavu s a akce a (viz. rovnice 2.18). Tato metoda má výhodu, že nepotřebuje model daného prostředí, při trénování totiž dochází k hledání optimálních hodnot Q přímo z interakcí s prostředím. Policy⁷ má svou optimální action-value Q -funkci 2.18, která je definována jako

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a), \quad (2.19)$$

Při trénování jde tedy o objevení optimální policy za pomoci nalezení optimální Q -funkce. To je umožněno díky Bellmanovu principu optimality, který vyjadřuje fakt, že budoucí hodnota stavu podle optimální policy se musí rovnat očekávanému stavu při aplikaci nejlepší akce ze všech možných

$$q^*(s, a) = \mathbb{E}[R(s_{k+1}, a_{k+1}) + \gamma \cdot \max_{a \in A(s)} q_{\pi}^*(s_{k+1}, a_{k+1}) | s_k = s, a_k = a] \quad (2.20)$$

$$= \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} Q^*(s', a') \right], \quad (2.21)$$

díky které lze stanovit vhodnou akci v daném stavu s , která povede na největší možnou hodnotu v budoucím stavu s' přes všechny možné akce a' a při dostatečném počtu opakování konverguje k optimální Q -funkci. Podrobněji tento problém popisuje [14, p. 63].

2.2.2 Policy based

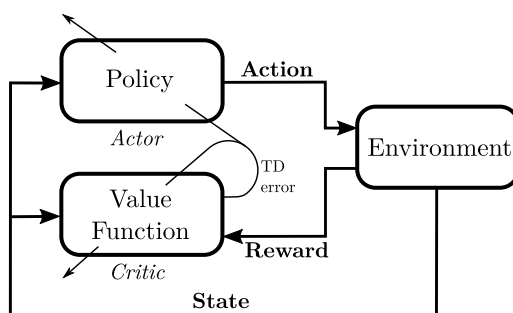
Na rozdíl od předešlé metody, která se soustředí na nalezení optimální policy za pomoci maximalizace kumulativní odměny (Q -funkce), tato ji hledá přímo a kumulativní odměna je až sekundárním parametrem, pokud je vůbec vypočítána. Typicky je policy reprezentována jako neuronová síť a k nalezení jejích parametrů slouží metoda poklesu gradientu (gradient descent) v prostoru stavů a akcí pro maximalizaci očekávané odměny.

Umožňuje navíc fungovat v prostředí se spojitými akcemi, což není ve value based možné pokud předpokládáme, že všechny akce ve stavech nejsou navzorkovány s nekonečně malou periodou. U nich se běžně určuje pravděpodobnost diskretních akcí a na jejich základě se pak vykonávají.

⁷strategie pro pohyb v daném prostředí dávající pravděpodobnost dané akce podle výše odměny

2.2.3 Metoda actor-critic

Jedná se o jednu ze spousty metod, která byla vybrána pro svou schopnost zpracovávat spojitě vstupy a reagovat spojitými akčními zásahy, což je v úloze udržení vozidla v jízdním pruhu výhodné. Taktéž ji lze využít na hraní her, které zmíněné vstupy vyžadují.



Obrázek 2.8: Schéma RL metody actor-critic. Zdroj [16]

Jde v podstatě o kombinaci předchozích RL přístupů. Skládá se ze dvou propojených bloků, "actor" (policy-based) a "critic" (value-based) viz Obrázek 2.8, kde "actor" zastupuje část agenta, který predikuje a provádí akce v prostředí a "critic" predikuje odměnu za daný krok v daném stavu. Oba bloky fungují paralelně a jsou na sobě vzájemně závislé. "Actor" bere jako vstup stav prostředí a jako výstup volí akci. "Critic" má složitější strukturu. Jako vstup bere aktuální stav prostředí a akci vybranou "actorem". Výstupem je predikovaná hodnota odměny.

Při trénování se postupuje postupně z fáze objevování ("exploration") do využívání (exploitation) přičemž se zmenšuje náhodný šum přidávaný k akcím. Po daném počtu epoch se upraví struktura "actora" aplikací parametrů "critica", který se snaží najít optimální Q-funkci minimalizující kumulativní ztrátovou funkci odměn, zatímco "actor" se snaží volit správnou akci pomocí poklesu gradientu.

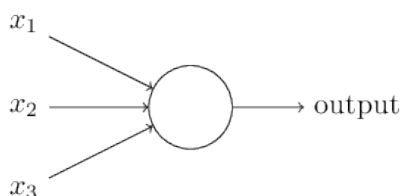
2.2.4 Shrnutí

Přes značné výhody RL přístupu, který nepotřebuje žádný model ani trénovací dataset se pojí několik negativ. Při trénování výše zmíněných modelů je potřeba stanovení správné funkce odměny, která má ve většině případů zásadní vliv na funkčnost a je potřeba dopředu myslet na striktní omezení a požadavky na agenta. Dále může být problematické samotné prostředí, ve kterém se bude pohybovat. V případě chyb nebo porušení pravidel je potřeba agenta resetovat do startovní pozice, což může být ve fyzickém prostředí problém. Posledním hlavním problémem je délka trénování v případě operace agenta v prostředí s reálným časem. Některé problémy však

mohou být vyřešeny buď použitím simulátorů a nebo specifických knihoven pro vývoj RL algoritmů jako je například OpenAI Gym [17].

2.3 Řešení pomocí hlubokého učení

S nedávným nárůstem výpočetního výkonu a optimalizace specifického hardwaru se stalo hluboké učení dostupnější. Vznikem frameworků jako je PyTorch, Keras a TensorFlow pro práci se strukturami neuronových sítí v populárních programovacích jazycích se zpřístupnila a zjednodušila práce s nimi. Nebylo již nutné programovat vlastní funkce a metody a více lidí se tak mohlo zapojit do vývoje složitějších architektur a soustředit se na stavbu samotných neuronových sítí. Dnes již existuje spousta návodů i mimo výzkumné práce, ať už specifických pro zmíněné frameworky a nebo těch sdílených komunitou.



Obrázek 2.9: Základní jednotka neuronové sítě - perceptron. Zdroj [18]

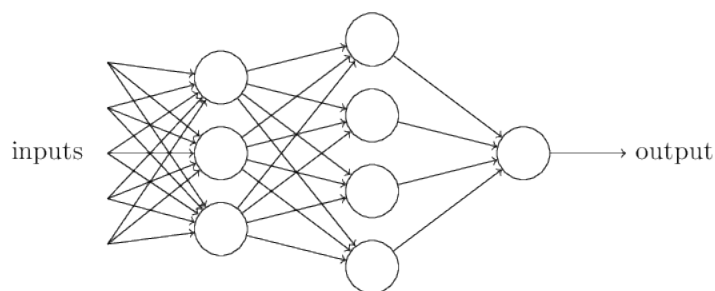
Samotné neuronové jsou matematické modely inspirované biologickým nervovým systémem, kde je nejmenší stavební jednotkou jeden neuron nazývaný také jako perceptron viz Obrázek 2.9. Ten je matematicky reprezentován jako funkce

$$y = f(\mathbf{W} \cdot \mathbf{X} + b),$$

kde \mathbf{W} je vektor vah, \mathbf{X} vektor vstupů, b je práh a f aktivační funkce transformující vstupní hodnotu součtu na danou výstupní funkci.

Z paralelně zapojených perceptronů se skládají jednotlivé vrstvy, které se řetězí dále do podoby požadované neuronové sítě např. na Obrázku 2.10. Učení pak probíhá nastavováním zmíněných vnitřních parametrů za pomoci gradientu vypočteném v dimenzi rovné počtu vstupů. Podrobnější informace o fungování lze získat z [19, 18].

Dnes již existuje spousta architektur optimalizovaných pro řešení specifických problémů a tudíž není nutné samostatně vyvíjet vlastní a postačí upravit již stávající. Pro oblast detekce pruhů z obrazu je vhodná kategorie tzv. konvolučních neuronových sítí. Ty lze dále rozdělit na problém klasifikace, detekce a segmentace popsanych v následujících sekcích.



Obrázek 2.10: Jednoduchá ukázka architektury vytvořená spojováním perceptronů. Zdroj [18]

2.3.1 Klasifikace obrazu

Jedná se o problematiku s nejjednodušší architekturou, která jako vstup bere matici reprezentující obrázek a jako výstup dává zpravidla pravděpodobnost předem definovaných kategorií. Jednou z nich je LeNet-5 [20] s pouhými 60000 parametry představená již v roce 1998. Její struktura se stala de facto standardní šablonou pro jiné mnohem větší sítě. I když se jedná o nejjednodušší typy sítí, spadají sem také velké sítě jako AlexNet [21] s 60M parametry nebo VGG-16 [22] se 138M parametry, které mají více vrstev a pracují s větším rozlišením vstupu, což může být v některých případech výhodné, ale značně náročné na HW zdroje. Od toho se odlišují například sítě Inception-v3 [23] na bázi technologie Network in network [24] nebo ResNet-50 [25] s použitím tzv. "skip connections" kombinující výstup předchozí vrstvy a výstupu současné. Jak je zřejmé z výše uvedeného vývoje sítí, zvyšováním počtu parametrů došlo k saturaci přesnosti sítě a tak musely být aplikovány jiné metody pro její zlepšení.

2.3.2 Detekce objektů

Detekční sítě jsou komplikovanější struktury a od klasifikace se liší tím, že neklasifikují vstupní obraz jako celek ale hledají v něm požadované objekty. Výstupem tedy bude region vymezený v obraze obdélníkem, ohraničující nalezený objekt a třída. V případě více objektů bude výstupem odpovídající množství zmíněného výstupu.

Detekci lze rozdělit na dvoustupňovou a jednostupňovou. Dvoustupňová se skládá z prvotní detekce samotných regionů, ve kterých jich je navrženo několik stovek a jejich následné seskupení do nejlepších z nich. Ty jsou poté transformovány na fixní velikost a pomocí konvoluční neuronové sítě klasifikovány do tříd. Typickým představitelem je R-CNN [26].

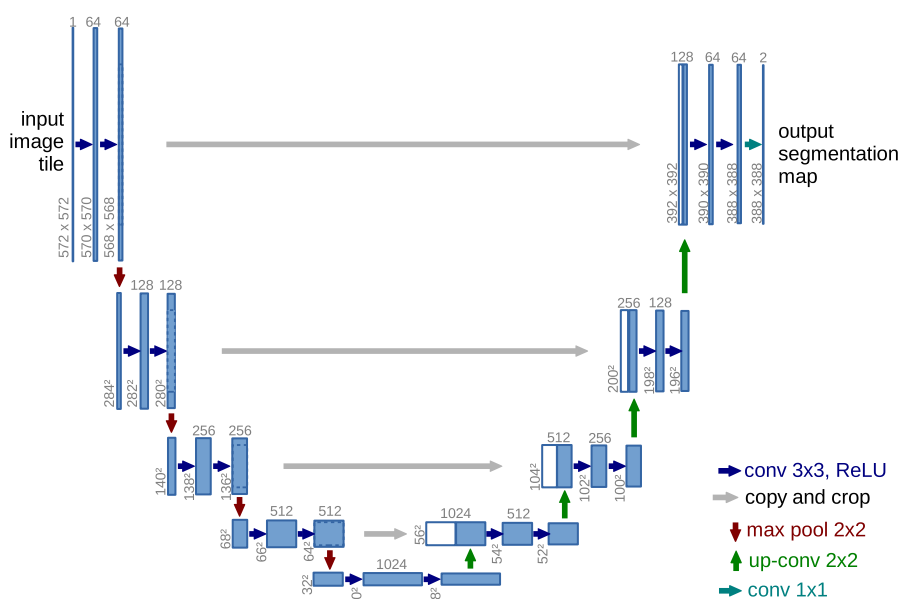
Jednostupňová detekce kombinuje výše zmíněné kroky do jediné konvoluční neuronové sítě (např. YOLO [27]), která predikuje regiony včetně pravděpodobnosti

přítomnosti objektu a jejich tříd. To umožňuje rychlejší detekci než u předchozího typu.

2.3.3 Sémantická segmentace obrazu

Představena byla již metoda, která klasifikuje celý vstupní obraz viz. 2.3.1 a nebo v něm hledá individuální objekty viz. 2.3.2. Tato se však zabývá klasifikací obrazu na mnohem nižší úrovni, konkrétně úrovni jednotlivých pixelů.

Obecný princip lze popsat tak, že vstupem neuronové sítě je opět obrázek, který je propagován skrze tzv. encoder-decoder strukturu. V části encoder se z obrázku extrahují vlastnosti, které se použijí v decoderu, který zpětně vytvoří matici stejné velikosti jako je vstup jen v podobě masky. Ta obsahuje informace o třídách jednotlivých pixelů a při aplikaci jejího překrytí se vstupním obrázkem lze zjistit, jak která část obrázku byla klasifikována. Jako příklad architektury lze uvést například U-Net viz. Obrázek 2.11 původně navržen pro využití v biomedicíně popsán v článku [28], ale nadále využit i několika dalšími strukturami viz. článek [29].



Obrázek 2.11: Struktura neuronové sítě U-Net definována v [28]. Zdroj [28]

2.3.4 Shrnutí

Obecně lze výše zmíněné metody shrnout do dvou kategorií, dvou stupňovou a jedno stupňovou. Dvou krokovou lze rozumět specifický přístup, kdy vybraná neuronová síť vykoná pouze část celkové detekce, a to zpravidla extrakci vlastností.

Následný krok zpracuje výstup neuronové sítě a provede výsledné úpravy jako je např. proložení polynomem.

Jedno kroková kategorie se týká metod a architektur, které v jednom kroku sjednotí výše zmíněné činnosti do jedné neuronové sítě.

Přístupy zmíněné v předchozích sekcích se dají použít k detekci pruhů různými způsoby. Klasifikaci lze využít k detekci polohy čar při velmi jednoduchém úkolu, neboť je vstupem celý obraz a výstupem pouze pravděpodobnosti jednotlivých tříd. Lze tedy klasifikovat i směr čar za předpokladu velkého množství výstupních tříd a také minimálním rušením v podobě jiných objektů. Klasifikace se již dnes pro své vysoko-úrovňové zpracování na detekci čar nepoužívá a dává se přednost pokročilejším nástrojům.

Detekce objektů je v autonomním řízení hojně používána, avšak spíše než pro detekci čar, tak pro rozpoznávání vertikálního dopravního značení ostatních účastníků provozu nebo objektů v blízkém okolí silnice (popelnice, stromy apod.). Při detekci čar by pro dostatečnou přesnost bylo nutné detekovat buď jednotlivé pruhy například přerušovaného pruhu a z pozice rekonstruovat jeho skutečný průběh. Pro tento účel se dává dnes přednost poslední možnosti.

Sémantická segmentace obrazu je v detekci čar suverénně nejlepší volbou z výše zmíněných. Se svojí přesností detekce na úrovni jednotlivých pixelů rozdělí celý vstupní obraz na jednotlivé regiony se svou kategorií. Ty lze pak separátně zpracovat a soustředit se v případě čar, pouze na odpovídající kategorii. V jistém úhlu pohledu dokáže nahradit zmíněnou detekci objektů s detailnější detekcí, ale nutno podotknout, že na rozdíl od ní není výstupem přímo souřadnice objektu s vymezeným prostorem, kde se nachází v podobě tzv. "bounding boxu"⁸, ale shluk klasifikovaných pixelů. Ten může vyžadovat další zpracování jako je vyhlazování nebo shlukování pro výsledné určení polohy objektu.

Poslední zmíněné metody pro svou přesnost a úspěšnost vyžadují velmi přesná trénovací data, která jsou často ručně anotována lidmi a proto velmi finančně a časově náročná. Naštěstí v poslední době díky většímu povědomí a zájmu firem rozvíjet se v této oblasti vzniká spousta datasetů pro úlohy autonomního řízení z reálného prostředí nebo syntetických za pomoci simulací, což je více rozebráno v kapitole 3.

2.4 Porovnání přístupů

Na úlohu udržení vozidla v jízdním pruhu je možné aplikovat libovolný ze zmíněných přístupů. Algoritmický nabídne jednoduchost a srozumitelnost použitím zavedených algoritmů avšak za cenu vyšší citlivosti na změny prostředí. Zpětno-

⁸Obdélník přesně ohraničující detekovaný objekt s minimem okolí.

vazební učení je náročnější na správný návrh agenta, hodnotící funkce a v případě reálného prostředí i na čas, ale za to dokáže ve výsledku i sám vozítkem pohybovat bez potřeby regulátoru v případě takového návrhu. Hluboké učení je dnes již velmi pokročilé a s nárůstem výpočetního výkonu i dostupnější. Podstatné pro něj je vytvořit kvalitní dataset a použít správnou architekturu. Bohužel ne veškeré chování je popsáno a může tak dojít k chybám, které díky komplexnosti dnešních neuronových sítí nelze jednoduše vysvětlit.

Simulátory a datasety

3

Nasazení nebo dokonce vývoj systémů pro autonomní řízení může být velmi problematický a nebezpečný. Mimo jiné je potřeba funkční prototyp vozidla s veškerými senzory a případně i volný prostor na testování, což se dokáže rychle projevit na potřebných financích.

Naštěstí se většina zmíněných nákladů a problémů dá omezit použitím simulačního softwaru, který dokáže poskytnout jak prostředí, tak senzory a navíc i referenční hodnoty přesného pohybu, kolizí s okolím apod. Výdaje na prvotní testování se tak omezí na minimum. Pro plynulé simulace je však potřeba dostatečně výkonný HW s dedikovanou grafickou kartou, protože dochází při jejím běhu k paralelním výpočtům i dalších pohledů sensorů viz 3.2, což dokáže PC značně zatížit.

3.1 Simulátory

Následující výběr simulátorů byl sestaven tak, aby byly snadné a rychlé ke spuštění, nevyžadovaly přihlašování pro přístup, fungovaly lokálně a nejlépe jak na platformě Linux, tak i Windows.

3.1.1 Gym-duckietown

Prvním simulátorem je gym-duckietown, který vznikl na bázi projektu Duckietown¹ blíže popsán v [30]. Ten začal jako předmět na MIT v roce 2016 a rozšířil se po celém světě s iniciativou zpřístupnit vzdělání v oblasti AI a robotiky všem v podobě zdokumentované vzdělávací platformy. Její součástí je právě i zmíněný simulátor, jehož prostředí je generováno na základě standardů popsanych v dokumentaci viz. [31].

Samotný simulátor je dostupný z [32] s více způsoby instalace. Jelikož je napsaný v jazyce Python, měl by zaručit lepší kompatibilitu napříč platformami, avšak při testování na Windows 10 ho nebylo možné jednoduše zprovoznit. Až při použití docker obrazu se ho podařilo spustit s menšími problémy.

¹<https://www.duckietown.org/>

3. Simulátory a datasety

Po stažení odpovídajícího docker obrazu bylo nutné upravit spouštěcí příkaz a nainstalovat aplikaci XLaunch² pro přesměrování grafického prostředí z docker kontejneru do Windows. Nutno podotknout, že při použití platformy Linux by nemělo být potřeba podstupovat zmíněné kroky, což ale nebylo otestováno.

Simulátor lze dále ovládat za pomoci python skriptů. Více informací k možnostem simulátoru lze získat ze zmíněného repository, z dokumentace viz. [31] nebo z článku o soutěži AI-DO [33].



Obrázek 3.1: Ukázka prostředí simulátoru gym-duckietown.

3.1.2 AirSim

Pokročilejším simulátorem je dnes již archivovaný AirSim vyvíjený jako opensource společností Microsoft od roku 2017 do 2022 blíže popsán v článku [34]. Jeho zdrojový kód i kompilované binární soubory jsou stále k dispozici na stránkách [35].



Obrázek 3.2: Ukázka prostředí simulátoru AirSim včetně možných pohledů senzorů.

²<http://www.straightrunning.com/XmingNotes/>

Narozdíl od předchozího je postaven na Unreal Engine (C++), ale nabízí i experimentální Unity (C#) vydání, což umožňuje větší prostor ve výkonu díky rychlejším programovacím jazykům. Nabídne tak pokročilejší prostředí s vyšším detailem, ovládání počasí a v úvodu zmíněné různé pohledy viz. Obrázek 3.2 a senzory.

Ovládání vozidla je provozováno přes Python nebo C++ API³ popsané v dokumentaci [36] a rovněž je možné ovládat více agentů ve stejném světě.

3.1.3 Carla

Další pokročilý open-source simulátor Carla na bázi Unreal Engine (C++) nabízí podobně jako AirSim realistickou simulaci a ovládání pomocí Python a C++ API. Navíc je zde také možné mít větší kontrolu nad množstvím generovaných NPC aut a osob. Nabízí také vlastní autopilot, který je možné kdykoliv zapnout a případně porovnávat s vlastním řešením.



Obrázek 3.3: Ukázka prostředí simulátoru Carla včetně možných pohledů senzorů.

Při vytvoření nového agenta se otevře nové okno s jeho vizualizací viz. Obrázek 3.3 a je tak možné ovládat manuálně jednotlivé agenty a vidět jejich separátní pohledy. S těmito možnostmi však přichází daleko vyšší požadavky na HW.

Narozdíl od AirSim poskytuje realističtější fyziku a samotná NPC⁴ reagují lépe na agenta např. že zastaví a při srážce se chovají tak, jak by se očekávalo ve skutečnosti. Prostedí je vymodelováno od nuly a jeho silnice jsou navrženy podle standardu OpenDRIVE[®] verze 1.4 dostupného z [37].

3.1.4 Shrnutí

Ačkoliv se gym-duckietown nejeví jako pokročilý simulátor, je postaven na základech existující platformy duckietown se spoustou návodů a soutěží, ze kterých se lze mnohé naučit. Samotnou mapu lze také individuálně navrhnout.

³Zkratka rozhraní pro programování aplikací z anglického Application Programming Interface.

⁴Zkratka nehrácké postavy z anglického Non-Playable Character.

AirSim nabízí pokročilejší grafiku, již připravené další pohledy a senzory jako je LIDAR, infračervená kamera nebo senzor vzdálenosti. Díky tomu je však náročnější na HW, zejména pak GPU⁵. Bohužel však při kolizích vozidel se stává, že NPC nezastaví a odmrští agenta nereálnou silou nebo ho tlačí před sebou, dokud ho nezastaví např. červený semafor.

Carla se dle nalezených rozdílů podle dokumentací od AirSimu příliš neliší. Zásadní rozdíl je však v konfigurovatelnosti prostředí a naprogramované fyzice NPC, kde poskytne vyšší kvalitu, ale za cenu výpočetní náročnosti.

Mimo zmíněné simulátory existuje spousta dalších a pro specifické úlohy lze použít i vyvinuté hry, které neposkytují sice takovou kontrolu nebo zpětnou vazbu, ale například pro RL mohou být dobrou alternativou.

Při testování simulátorů vyplynulo, že možností je hodně avšak při daných podmínkách v úvodu kapitoly se jeví jako vhodnější platforma Linux pro její větší kompatibilitu. Všechny zmíněné simulátory a většina dalších nabízí k ovládání Python API, ačkoliv samotné simulátory jsou často naprogramovány v jiném jazyce. Tento fakt ukazuje nejen na popularitu jazyka v této oblasti, ale také na přístupnost simulátoru pro větší skupinu lidí bez nutnosti znalosti komplikovanějších jazyků.

3.2 Datasety

Samotná existence simulátorů velmi rozšiřuje možnosti pro tvorbu tzv. syntetických datasetů, které je možné generovat již s požadovanými podmínkami a konkrétními anotacemi. V následujícím výběru datasetů však jsou jen ty, které byly vytvořeny a ručně anotovány v reálném prostředí, které obsahuje nedokonalosti na rozdíl od simulace. Ony nedokonalosti pak mohou být výhodou v případě reálného nasazením, protože pro model jsou již známé.

Duckietown. V rámci dříve zmíněné platformy Duckietown [30] lze získat oficiální object detection dataset v dokumentaci k AI-DO viz. [38] v rámci jejich kompletní dokumentace [31]. Výhodou může být, že zde není uvedena žádná licence. Obsahuje 1956 obrázků a 3 anotované kategorie objektů.

KITTI. Velmi dobrý zdroj datasetů pro autonomní řízení vzniklý jako projekt Karlsruhe Institute of Technology a Toyota Technological Institute at Chicago, který nabídne i samotné výstupy ze senzorů hloubky, nebo stereovize včetně potřebných parametrů. To vše velmi dobře anotováno, popsáno ve článcích jako je [39] a přímo na jejich stránkách [40] lze porovnat výsledky a přesnost s jinými metodami.

⁵Zkratka grafického procesoru z anglického Graphics Processing Unit.

Valeo Woodscape. Volně dostupný dataset vytvořený firmou Valeo pro poskytnutí dat komunitě pro vývoj řešení nad vstupním obrazem v podobě tzv. rybího oka. Obsahuje na 10k obrázků a 40 anotovaných tříd. Detaily sběru dat jsou popsány v článku [41] a ke stažení z [42].

A2D2. Dataset vytvořený firmou Audi zveřejněný pro podporu startupů a výzkumů zabývajících se autonomním řízením automobilu. K dispozici zde [43] blíže popsáno v článku [44]

Appoloscope. Dataset popsáný v [45], vytvořený čínskou firmou Baidu Research v rámci projektu Apollo [46]. Na jejich webu [47] jsou dostupné ukázky jednotlivých typů včetně nástrojů a návodů pro manipulaci, ale plnou verzi datasetu lze získat jen po zaslání emailové žádosti.

Argoverse. Dataset vzniklý ve spolupráci organizace Argo AI a studentů a fakult univerzit Carnegie Mellon University a Georgia Institute of Technology obsahující dvě verze viz články [48, 49]. Obě se ale zaměřují primárně na predikci pohybu objektů, obsahují velmi detailní data z LiDARU, obrazu a také mapy prostředí, kde byla data sbírána. Ke stažení je z webových stránek projektu [50].

Cityscapes. Dataset vzniklý dle článku [51] ve spolupráci firmy Daimler, univerzit "TU Darmstadt", "TU Dresden" a institutu "MPI Informatics". Jeho motivací bylo zpřístupnit dataset, který zachycuje pomocí stereovize komplexitu reálného světa dle autorů lépe než v tu dobu dostupné alternativy. Ke stažení jsou z jejich webových stránek [52] ukázky a po registraci i plná verze.

Comma2k19. Dataset vytvořený společností Comma [53], která se zabývá vývojem open-source autopilotem použitelným ve více než 250 modelech aut. Ve článku [54] autoři zmiňují, že se při sběru dat orientovali hlavně na kvalitu a využití již stávajících senzorů za pomoci napojení na komunikaci CAN. Ty byly dále obohaceny o kameru, akcelerometr a mimo jiné i GNSS daty zpracovanými knihovnou Laika [55] pro větší přesnost. Stažení je dostupné přes torrent zde [56, 57].

BDD100K. Dataset vzniklý ve spolupráci společnosti Element, Inc. a univerzit UC Berkley, Cornell University a UC San Diego s cílem vytvořit adekvátně velký dataset pro moderní autonomní systémy. Dle článku [58] obsahuje 100 tisíc anotovaných videí včetně lokalizačních dat. Ke stažení je po registraci na stránkách [59].

PixSet Dataset. Dataset vytvořen firmou LeddarTech. Mimo obrazová data obsahuje i surová data z LiDAR senzorů. Ke stažení je po registraci ze stránek [60] a

popsaný ve článku [61].

NuScenes. Dataset vytvořený firmou Motional. Obsahuje data jak pro úlohu segmentace a detekce objektů, tak pro plánování trasy. Na stránkách [62] je po registraci možné data stáhnout a následně i vlastní přístup otestovat a porovnat s dalšími řešeními.

Oxford RobotCar Dataset. Dataset vytvořený Oxfordskou univerzitou se záměrem poskytnout data stejného prostředí s měnícími se podmínkami a důrazem na lokalizaci, stereovizi a mapování jak je popsáno v článku [63] a po registraci ke stažení na stránkách [64].

PandaSet. Dataset vytvořený ve spolupráci společností Scale a Hesai orientovaný převážně na mapování prostředí za pomoci stereovize a LiDARů. Spolu s bližšími informacemi je ke stažení po registraci na jejich webových stránkách [65].

Udacity Car Dataset. Dataset vytvořený vzdělávací společností Udacity v rámci programu autonomního automobilu viz. [66] obsahující data vhodná pro trénování či testování detekce objektů. Ke stažení je z oficiálního GitHub repository [67] bez registrace.

Waymo Open Dataset. Dataset vytvořený ve spolupráci Waymo a Google s cílem zprostředkovat veřejnosti data pro další vývoj a výzkum v oblasti autonomního řízení a strojového vidění. Obsahuje jak data ze senzorů (kamery, LiDAR, etc.), tak i dataset s pohybem objektů a odpovídající 3D mapy. Po registraci je dostupný ze stránek [68].

3.2.1 Shrnutí

Jak je vidět výše, existuje spousta anotovaných datasetů, z nichž některé nabízí i data ze senzorů nebo možnost porovnání výsledků. Většinu těchto kvalitních datasetů není možné použít pro komerční vývoj bez zakoupení licence zpravidla po komunikaci s vlastníkem. Existuje spousta dalších zdrojů nebo portálů jako je například Kaggle [69], Roboflow [70] a další. U nich je však potřeba si dávat pozor na kvalitu a správnost anotací i samotného obsahu. Narozdíl od nich, u datasetů vytvořených pro soutěže, vývoj na univerzitě nebo v automobilce, lze předpokládat vyšší kvalitu.

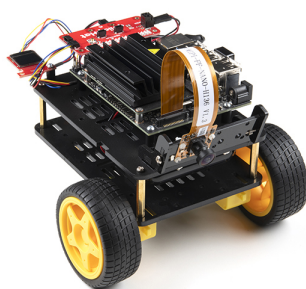
Praktická část - řešení

4

V následujícím textu bude popsán postup při implementaci dvou metod popsaných v teoretické části. Konkrétně se jedná přístup algoritmický a pomocí hlubokého učení. Veškeré použité kódy jsou k dispozici na GitHub repository¹ [76].

4.1 Použité prostředky

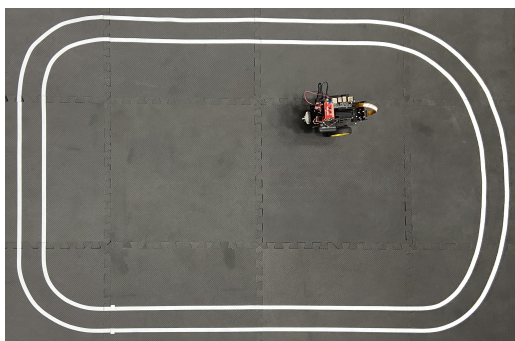
Jako základní HW, který byl uvažován pro demonstrační modelové příklady je dvoukolevý diferenciální robot ovládaný jednodeskovým počítačem Jetson Nano B01 dodaný jako set od firmy Sparkfun viz Obrázek 4.1. Samotné testování však probíhalo na herním laptopu s operačním systémem Windows 10, aby se omezila problematika s trénováním neuronové sítě a nedostatkem paměti pro knihovnu TensorFlow.



Obrázek 4.1: Použitý kit robota s jednodeskovým počítačem Jetson Nano B01. Zdroj [71]

Na vytvořené trénovací trase (viz Obrázek 4.2) s různě prudkými zatáčkami byla natočena videa průjezdu, která pak byla dále zpracována na zmíněném laptopu podle postupu popsaném v následujícím textu.

¹https://github.com/Fiiila/BPARR_2023



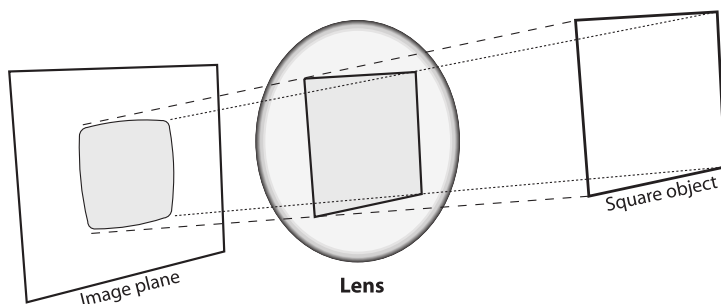
Obrázek 4.2: Vytvořená testovací trasa.

4.2 Předzpracování

Před samotnou detekcí pruhu je nutné nejprve upravit surová vstupní data v podobě snímků z kamery připevněné na robotovi. Vlivem použití reálných optických prvků a nepřesností při výrobě dochází ke zkreslení zachyceného obrazu. V této práci budou uvažována dvě nejčastější, radiální a tangenciální.

4.2.1 Radiální zkreslení

Radiální zkreslení vzniká vlivem prostupu obrazu skrze čočku. Její kruhový tvar pak obraz transformuje odlišně podle vzdálenosti místa prostupu světelného paprsku od středu čočky. Zkreslení podléhají body obrazu více vzdálené od jeho středu viz. Obrázek 4.3.



Obrázek 4.3: Zobrazení radiálního zkreslení při prostupu světelných paprsků čočkou. Zdroj [72, p. 376]

Tuto závislost lze také popsat pomocí prvních několika stupňů Taylorova rozvoje v okolí $r = 0$ jako

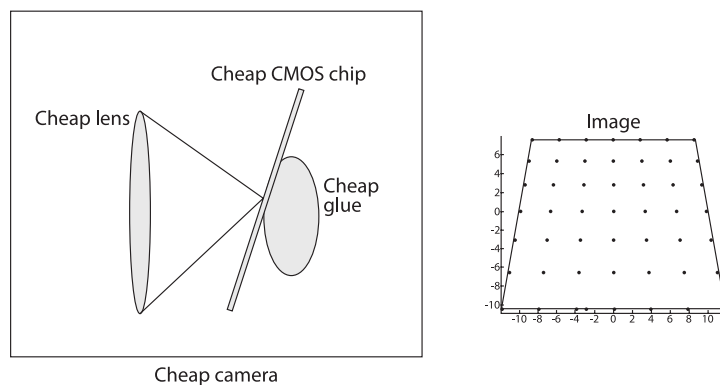
$$x_{corrected} = x \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (4.1)$$

$$y_{corrected} = y \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \quad (4.2)$$

kde (x, y) jsou původní zkreslené body a $(x_{corrected}, y_{corrected})$ jsou jejich nové polohy po korekci. Koeficienty k_1, k_2, k_3 jsou proměnné a popisují dané zkreslení.

4.2.2 Tangenciální zkreslení

Druhým nejčastějším typem je tangenciální zkreslení. Je způsobeno nepřesností výroby samotné kamery, kdy objektiv není uložen přesně paralelně se snímacím senzorem viz. Obrázek 4.4 a některé části obrazu se tak mohou jevit posunuté a roztažené. Minimální popis tohoto jevu je rovnicemi



Obrázek 4.4: Zobrazení tangenciálního zkreslení při nepřesném uložení objektivu vůči senzoru v levné kameře nalevo. Zdroj [72, p. 377]

$$x_{corrected} = x + [2p_1y + p_2(r^2 + 2x^2)] \quad (4.3)$$

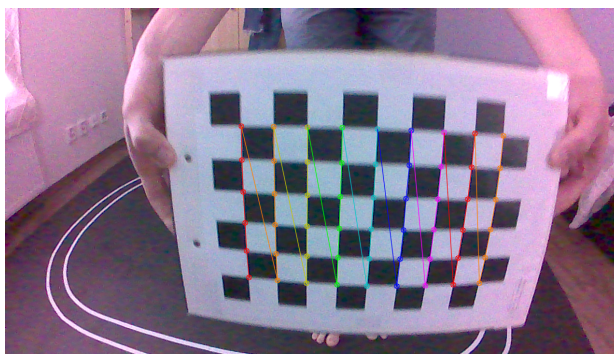
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2x] \quad (4.4)$$

opět popisující transformaci bodů zkresleného obrazu (x, y) na body po korekci $(x_{corrected}, y_{corrected})$. Proměnné p_1 a p_2 jsou parametry daného zkreslení.

4.2.3 Korekce zkreslení

Pro požadovanou korekci je potřeba zjistit neznámé parametry obou výše uvedených zkreslení. Pro jejich stanovení je potřeba provést měření v podobě snímání obrazu kamerou a získat tak body (x, y) reprezentující pozice pixelů ve zkresleném obrazu. Aby bylo možné získat body po korekci $(x_{corrected}, y_{corrected})$, je nutné do snímaného prostoru přidat kalibrační objekt v podobě šachovnice se známým rozměrem viz. Obrázek 4.5.

Díky knihovně OpenCV (v kódu jako cv2) lze pak snadno iterovat přes uložené snímky a nalézt jednotlivé vnitřní rohy šachovnice pomocí funkce `findChessboardCorners()`, která jako vstup použije černobílý obrázek. Nalezené body jsou dále zpřesněny funkcí `cornerSubPix()` a lze je poté vykreslit jako na Obrázku 4.5.



Obrázek 4.5: Ukázka jednoho snímku z kalibračního datasetu s šachovnicí s rozměry 9x6.

Nyní lze pomocí funkce `cv2.calibrateCamera()` a spárováním skutečných a naměřených bodů spolu s rozměry snímku získat vnitřní parametry kamery, zkreslení a vektory rotace a translace. Poté je použita metoda výpočtu PnP² pro odhad výchozí pozice kamery a optimalizačním algoritmem Levenberg-Marquardt jsou minimalizací odchylky zpřesněny skutečné a odhadnuté polohy bodů dle získaných parametrů.

Pro samotnou korekci je potřeba ještě zpřesnit matici vnitřních parametrů s ohledem na škálovací parametr `alpha` funkcí `cv2.getOptimalNewCameraMatrix()`. Proměnná `alpha` ovlivňuje počet nepotřebných pixelů, kde při `alpha=0` je jich minimum a při `alpha=1` jsou použity všechny navíc s černými místy viz. Obrázek 4.6.

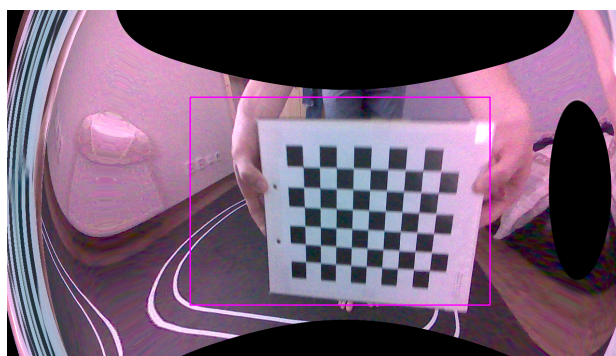
Následně nabízí OpenCV dvě možnosti korekce. Funkci `cv2.undistort()`, která vrátí přímo výsledek nebo `cv2.initUndistortRectifyMap()`, která vrátí dvě tzv. mapy popisující korekci a transformaci v ose x a y . Ty se následně aplikují pomocí `cv2.remap()`. V této práci je využita druhá možnost, neboť je při sekvenčním zpracování více obrázků rychlejší.

V případě volby `alpha > 0` zbývá jen oříznout obrázek na doporučený region s minimem nepotřebných pixelů vyznačený v Obrázku 4.6 růžovým obdélníkem. Zbylé pixely vně tohoto regionu jsou silně deformovány následkem korekce a v této práci nebudou využity ačkoliv to možné je. Předzpracování vstupu je tímto hotové a obrázek lze považovat za vstup do testovaných přístupů.

4.3 Algoritmické řešení

V tomto přístupu jde o praktickou demonstraci použití a limitů běžných algoritmických metod na úloze detekce pruhu.

²Perspective-n-Point (PnP) je výpočet řešící rotaci a translaci z 3D do 2D prostoru.



Obrázek 4.6: Ukázka snímku na Obrázku 4.5 po korekci s vyznačeným regionem s minimem nežádoucích pixelů.

4.3.1 Extrakce značení cesty



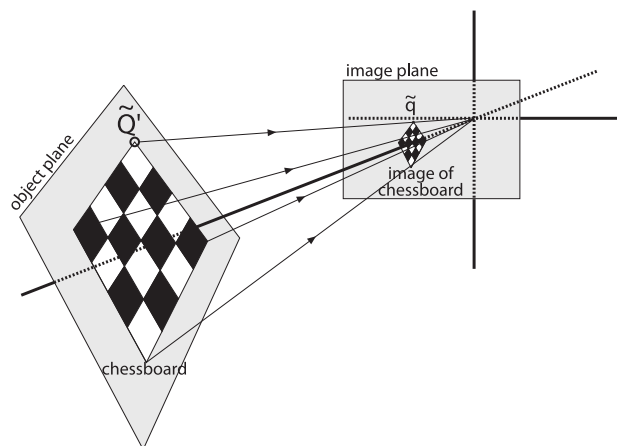
Obrázek 4.7: Vstupní snímek pro detekci.

Ze vstupního obrázku (Obrázek 4.7) je patrné, že potřebná část pruhu se nachází v dolní části obrazu. Pro lepší funkčnost se použije ořez horního obzoru, který neobsahuje relevantní informace pro detekci čar a následné řízení.

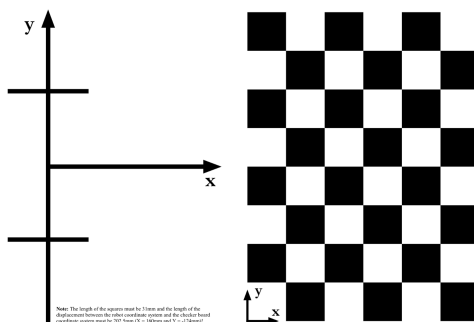
Oproti běžným pozemním komunikacím obsahuje testovací trasa i celkem prudké zatáčky a vzhledem k její menší celkové velikosti byl vybrán výřez nejbližší plochy za pomoci kalibrační šachovnice viz. Obrázek 4.9. S její pomocí se provede tzv. homografická transformace, při které dochází k transformaci z jedné roviny obrazu do jiné. K jejímu provedení je potřeba získat transformační matici. Pokud uvažujeme případ na obrázku 4.8 pak platí

$$\tilde{\mathbf{Q}} = [X \ Y \ Z \ 1] \quad (4.5)$$

$$\tilde{\mathbf{q}} = [x \ y \ 1]. \quad (4.6)$$



Obrázek 4.8: Schéma homografické transformace mezi dvěma rovinami. Zdroj [72, p. 385]



Obrázek 4.9: Použitá šachovnice pro homografickou transformaci. Zdroj [31]

V reálném případě se zjednodušuje reprezentace zmíněného \tilde{Q} vypuštěním rotace a translace v ose Z na

$$\tilde{Q}' = [X \ Y \ 1] \quad (4.7)$$

a homografickou transformační matici lze pak definovat jako

$$\mathbf{H} = s\mathbf{M} [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}], \quad (4.8)$$

kde s je faktor měřítka, \mathbf{r}_1 a \mathbf{r}_2 jsou rotační a \mathbf{t} translační vektory. \mathbf{M} je matice již známých vnitřních parametrů kamery definovaná jako

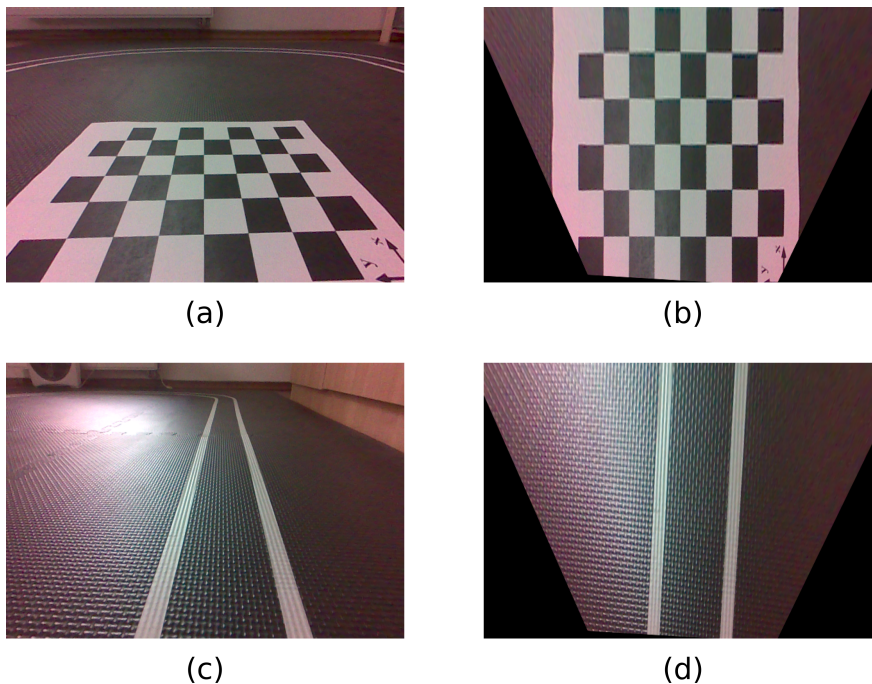
$$\mathbf{M} = \begin{bmatrix} \frac{f}{p_x} & 0 & c_x \\ 0 & \frac{f}{p_y} & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (4.9)$$

kde f je ohnisková vzdálenost, p_x a p_y jsou velikosti pixelu snímáče v milimetrech v odpovídajícím směru osy a c_x a c_y jsou souřadnice středu roviny pro korekci vychýlení snímáče z optické osy.

Výsledná rovnice transformace pak bude

$$\tilde{\mathbf{q}} = \mathbf{H}\tilde{\mathbf{Q}}' \quad (4.10)$$

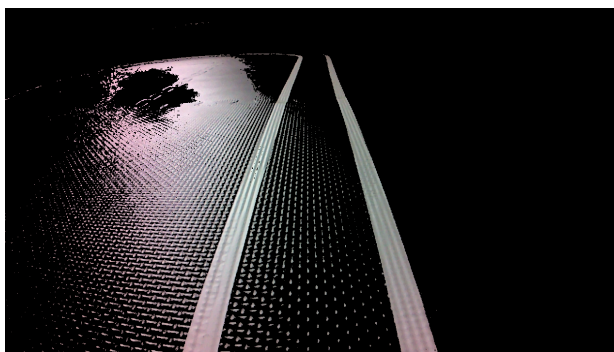
a v případě transformace opačné se použije inverzní matice \mathbf{H}^{-1} . Její aplikace poskytne v případě šachovnice na Obrázku 4.9 výsledek viz. Obrázek 4.10 a při zpracování vstupního obrázku s pruhy 4.7 dostaneme pohled z tzv. ptačí perspektivy (Obrázek 4.10 (d)), což je výhodné pro určení směru průběhu čar či jejich linearitu.



Obrázek 4.10: Aplikace homografické transformace na vstupní obraz (a), (c), kterou vznikly odpovídající (b) a (d).

V této fázi lze použít prahování podle barev v případě čar specifické barvy, čímž lze v ideálním případě rovnou extrahovat samotné pruhy. To je však v tomto příkladu nepoužitelné díky bílé barvě čar a četným odleskům podložky, které se jeví taktéž bíle viz Obrázek 4.11.

Před dalším krokem byl Obrázek 4.10(d) zašuměn konvolucí s Gaussovým jádrem o rozměrech 10×10 , aby vyhladil šum a nerovnosti podložky. Pro extrakci samotných čar byla použita technika hranové detekce konvolucí orientovaných jader vytvořených pomocí směrové derivace podle x Gaussova kruhově symetrického jádra o velikosti 10×10 a variancí $VAR[x] = 8$ a $VAR[y] = 1$ viz. sekce 2.1.2. Tyto



Obrázek 4.11: Ukázka neúspěšného pokusu prahování podle barvy.

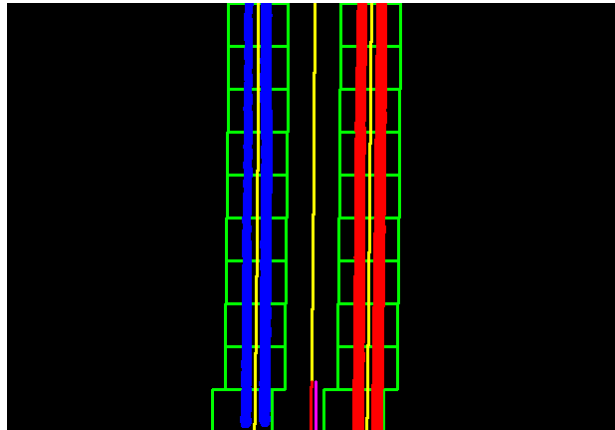
hodnoty byly určeny experimentálně na základě snahy eliminovat vroubky ve snímané leské podložce. Po binárním prahování metodou OTSU popsanou v článku [73] lze vidět že volba parametrů filtru úspěšně eliminovala odlesky podložky jak je vidět na Obrázku 4.12. Ten bohužel obsahuje i detekované hrany mezi černým pozadím a obrázkem. Tento jev lze odstranit tvorbou masky, která je statická a vázána na homografickou transformační matici.



Obrázek 4.12: Detekované hrany pomocí orientovaných filtrů v Obrázku 4.10(d).

Ze získaného binárního obrázku jsou dále extrahovány jednotlivé body náležící příslušné čáře. To je prováděno shlukováním a jak bylo zmíněno v teoretické části, pro tento krok je použita aplikace tzv. klouzavých okének (převzato a upraveno z [74]). Ty pracují na principu vykreslování obdélníků o daném rozměru na sebe s tím, že následující okénko změní svoji polohu v horizontální ose podle průměrné hodnoty horizontálních souřadnic bodů nalezených v něm, pokud by k posunu nedošlo. Tento postup se opakuje, dokud se nedosáhne maximálního počtu obdélníků nebo nejsou nalezeny žádné pixely hran.

Tento algoritmus ale vyžaduje stanovit startovací polohu. Omezením na dolní část binárního obrázku a tvorbou histogramu součtem přes sloupce získáme jasně viditelné vrcholky, definující počátek hledaných čar.



Obrázek 4.13: Zpracovaný binární obrázek metodou tzv. klouzavých okének včetně již rozdělených pixelů proložených kubickými polynomy, růžově vyznačeným středem obrazu a červenou tečnou středové trajektorie.

Pro každou čáru proběhne jeden tento algoritmus, který obdélníky vymezí prostor náležící dané čáře a pixely uvnitř přidá do odpovídajícího pole viz Obrázek 4.13. Tato pole bodů jsou pak použita na proložení bodů kubickým polynomem vybraným jako aproximací Klotoidy viz. rovnice 2.1.2 v teoretická částí práce. Na tuto operaci byla využita funkce `np.polyfit()` v rámci knihovny Numpy (`np`), která používá metodu minimalizace kvadratické chyby viz článek [13] a jejím výstupem jsou jednotlivé koeficienty následující obecné rovnice

$$y = p_1x^3 + p_2x^2 + p_3x + p_4. \quad (4.11)$$

Každá čára má samostatnou rovnici, která může být ovlivněna šumem. Proto byla stanovena ještě jedna čára, která je průměrem koeficientů pravé a levé rovnice. Ta zastává požadovanou ideální trajektorii, po které by se mělo vozítko dále ubírat. Z její rovnice lze také zjistit derivaci v nulovém bodě $x = 0$

$$\dot{y} = 3p_1^2x + 2p_2x + p_3 + 0 \quad (4.12)$$

$$\dot{y} = p_3 \quad (4.13)$$

a lineární rovnici tečny v bodě $x = 0$ jako

$$y_{tečna} = kx + q \quad (4.14)$$

, kde směrnice k je derivací rovnice viz. 4.13

$$k = \dot{y} = p_3 \quad (4.15)$$

a q parametr je výsledkem řešení následující rovnice dosazením $x = 0$ a $y = p_4$

$$y_{tečna} = p_3x + q \quad (4.16)$$

$$q = p_4, \quad (4.17)$$

což dá po dosazení do 4.14 výslednou rovnici tečny

$$y_{tečna} = p_3x + p_4. \quad (4.18)$$

Díky ní lze nyní určit odchylku ideální trajektorie od středu obrazu a její sklon. Tyto informace jsou potřebným vstupem pro regulátor, který bude způsobovat akční zásahy vozítka. Použité vozítko bohužel nemá enkodéry a nejsou tak k dispozici informace o poloze nebo rychlosti. Data z obrazu jsou však díky předsunuté kameře snímána kousek před vozítkem a jedná se tedy o ideální budoucí pozici, které se musí vozítko snažit přiblížit.

4.3.2 Regulace pohybu

Jelikož vozítko nemá k dispozici senzor k měření polohy ani rychlosti, jsou možnosti řízení omezené. Pohyb se ovládá na základě informace velikosti signálu na vstupu jednotlivých motorů. Jejich rozsah je v intervalu $\langle -1, 1 \rangle$, kde kladné hodnoty způsobují pohyb vpřed a záporné zpět. Takové ovládání není lineární nejen kvůli momentové charakteristice motorů, ale také valivému odporu mezi koly a podložkou.

Ze zpracování jsou k dispozici informace o natočení vozítka proti ideální budoucí trase a poloze oproti středu trasy. To jsou dva parametry, které je nutné současně řídit. Protože však na sebe mají přímý vliv, spojíme dva regulátory do tzv. kaskádního regulátoru zobrazeného na Obrázku 2.6 v teoretické části, kde vnější regulátor bere jako vstup diferenci polohy vozítka od trajektorie. Diference je vypočítána jako rozdíl pozice čáry při spodním okraji obrazu a jeho středu, která je dále normalizována šířkou obrazu. Rozdíl akčního zásahu vnějšího regulátoru a odchylky směru jízdy (náklonu) je pak dále použit jako vstup vnitřního. Jeho výstup je pak přímo akčním zásahem, který působí na jednotlivá kola vozítka. Požadovaná reference je nastavena na hodnotu 0, kterou docílíme centrování vozítka na požadovanou budoucí trajektorii.

Použité regulátory byly oba typu PI , kde k nastavování parametrů byla použita metoda třetin na rovném úseku testovací trajektorie.

4.4 Deep learning řešení - segmentace

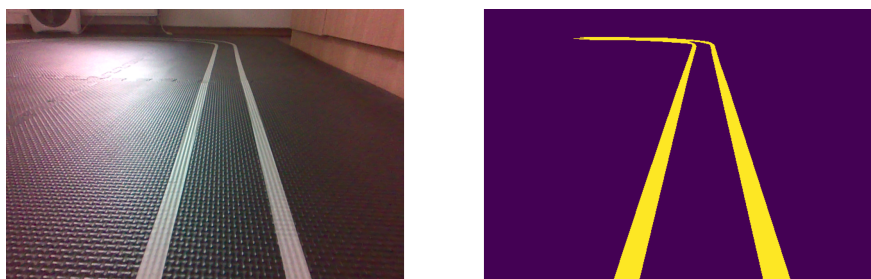
Pro přístup pomocí hlubokého řešení bylo vybráno dvoustupňové řešení zmíněné v teoretické části 2.3 s pomocí segmentace. Následující popis implementace tak v podstatě nahrazuje část extrakce čar v algoritickém řešení, zatímco výstupní zpracování zůstane totožné a proto zde již zmíněno nebude.

4.4.1 Tvorba datasetu

Dataset je klíčovou součástí jakéhokoliv řešení na bázi hlubokého učení, neboť má přímý vliv na kvalitu používaného modelu. Je důležité vytvořit ho přesně podle toho jaký výstup je očekáván.

Pro účely praktické realizace byl vytvořen z nahrávky průjezdu vozítka testovací trasou. Jelikož víme, že vstupem bude snímek již po korekci, použije se i v datasetu. Z 553 snímků bylo vybráno celkem 53.

Anotace byla provedena v programu CVAT³ ručním kreslením uzavřených polygonů reprezentující samotné čáry viz. Obrázek 4.14. Pro export byl vybrán formát COCO [75], jeden ze standardně používaných formátů.



Obrázek 4.14: Originální obrázek v datasetu (nalevo) spolu s jeho anotací (napravo).

4.4.2 Trénování modelu

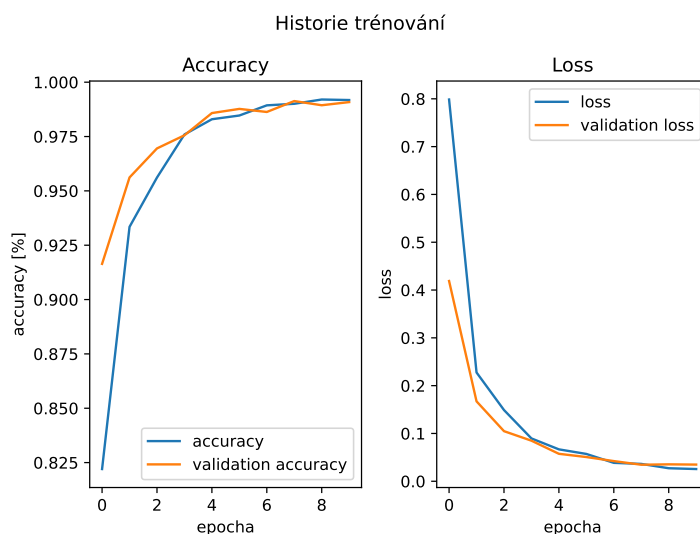
Jak bylo již zmíněno v teoretické části 2.3, existuje spousta architektur neuronových sítí. Po praktickou ukázkou byla vybrána síť U-Net, která dle článku [28] ke trénování nepotřebuje velký dataset, dosahuje dobrých výsledků a její struktura je použita v několika dalších architekturách jak je uvedeno v článku [29].

Neuronová síť byla sestavena od základu dle článku [28] v jazyce Python s pomocí frameworku TensorFlow a Keras. Jedinou změnou oproti článku je vstupní velikost sítě, která byla větší než obrázek z kamery vozítka a byla tak nastavena na (256, 256, 3) i s ohledem na konfiguraci trénovacího zařízení.

³Zkratka nástroje z anglického Computer Vision Annotation Toolkit.

Trénování bylo provedeno na herním laptopu s šestijádrovým procesorem *Intel Core i7-8750H* a grafickou kartou *Nvidia GeForce GTX 1050 Mobile (4GB)*. Z důvodu relativně malé paměti grafické karty bylo nutné omezit velikost trénovací dávky tak, aby trénování mohlo proběhnout. Jeho použité parametry jsou ve zdrojovém kódu na [76].

Celkové trénování na 53 snímcích skončilo po zhruba 2 minutách s validační přesností 99.23% viz. Obrázek 4.15, s použitím optimalizéru Adam, mírou učení ("learning rate") rovnou $1e^{-4}$ a ztrátovou funkcí ("Loss function") "binary crossentropy".

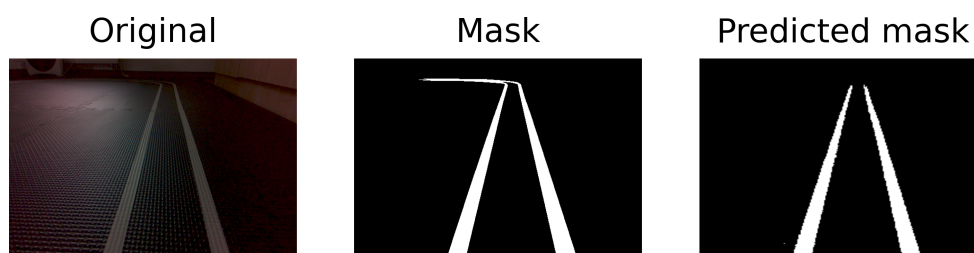


Obrázek 4.15: Průběh trénování sítě U-Net.

4.4.3 Predikce

Při predikci byl jako vstupní obrázek brán přímo snímek z kamery po korekci a až výstupní maska byla transformována do ptačí perspektivy pro algoritmus klouzavých okének.

Před vstupem do neuronové sítě bylo potřeba upravit matici tak, aby odpovídala jejímu vstupu. Byla tedy zmenšena na velikost (255, 255, 3) a rozšířena o jednu dimenzi na (1, 255, 255, 3). Predikovaný výstup 4.16 byl pomocí binárního prahování, pro odstranění málo pravděpodobných regionů, přeměněn na masku a zpět zvětšen na původní rozměr snímků z kamery (621, 430, 3) a upraven na pohled z ptačí perspektivy. Takto upravená maska již byla postoupena do metody klouzavých okének zmíněné v Algoritmické praktické části výše.

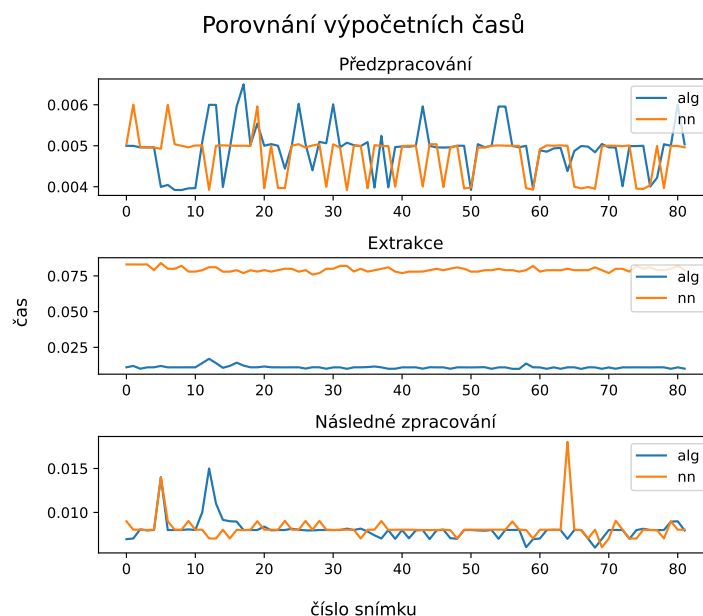


Obrázek 4.16: Srovnání vstupu (vlevo), ručně vytvořené masky (uprostřed) a predikované masky neuronovou sítí U-Net.

4.5 Zhodnocení

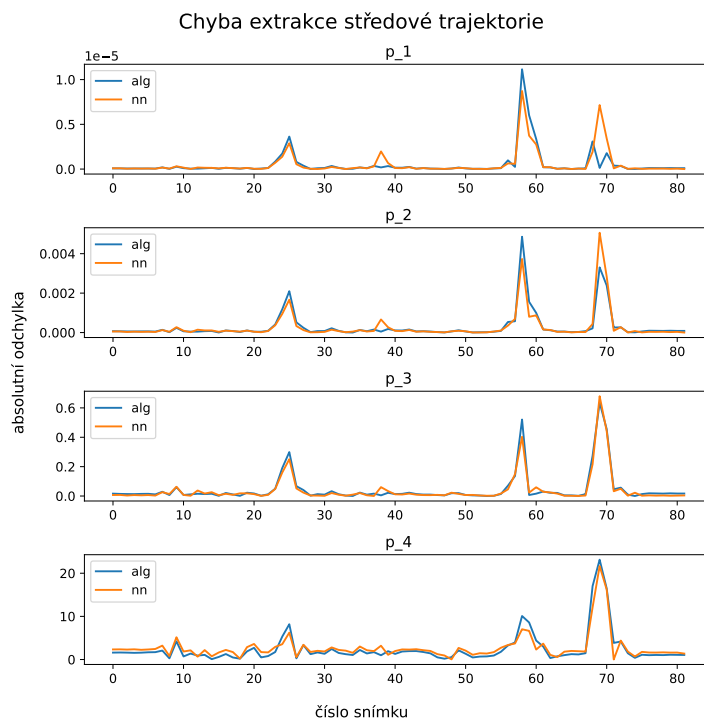
Pro porovnání implementovaných přístupů byl vytvořen nový dataset se snímky odlišnými od trénovacích dat, aby se zamezilo zvýhodnění neuronové sítě, která již tato data díky trénování zná. Tento validační dataset byl opět anotován programem CVAT a obsahuje celkem 82 snímků. Ty byly předloženy oběma metodám jako vstupní data.

Z běhu byla změřena doba trvání jednotlivých kroků viz. Obrázek 4.17, ze kterého je patrné, že krok předzpracování i následného zpracování trval dle očekávání stejně dlouho u obou metod, jelikož se jednalo o totožné postupy. V případě přístupu pomocí neuronové sítě však do měření nebyl započítán čas pro inicializaci modelu zhruba 4s.



Obrázek 4.17: Grafy s naměřenými časy trvání jednotlivých částí detekce čar.

Při extrakci je patrný již větší rozdíl, kdy algoritmická metoda čáry extrahuje v průměru za 0.0110 s, což je sedminou průměrného času docíleného neuronovou sítí 0.0794 s. Tento rozdíl je způsoben primárně kopírováním výsledku sítě z paměti grafické karty do operační. Tento jediný krok trvá přibližně 0.05s a tedy majoritu zmíněného času.



Obrázek 4.18: Grafy se změřenými absolutními odchylkami parametrů středové trajektorie od reference.

Tabulka 4.1: Porovnání přesnosti detekce koeficientů středové trajektorie.

| koeficienty stř. trajektorie | algoritmický přístup | přístup pomocí hlubokého učení |
|------------------------------|----------------------|--------------------------------|
| p_1 | 5.0640 | 5.2819 |
| p_2 | 0.0002 | 0.0002 |
| p_3 | 0.0457 | 0.0413 |
| p_4 | 2.4491 | 2.7286 |

Dalšími důležitými daty, které byly za běhu sbírány byly koeficienty hledaného polynomu středové trajektorie vozítka. Jako validační byly použity anotace vytvořené manuálně a s jejich pomocí byly vykresleny průběhy absolutních chyb jednotlivých metod viz Obrázek 4.18. Z něj lze odvodit, že větší odchylky od reference se

odehrály ve čtyřech rozích trajektorie, kde byly různě ostré zatáčky. V průměru se pak jedná o odchylky v Tabulce 4.1.

Z porovnání posledních dvou chyb parametrů použitých ke stanovení tečny v nulovém bodě pro regulaci pohybu vozítka nelze určit jednoznačně lepší přístup. Oba dokázaly stanovit trajektorii s obdobnou přesností. Algoritmický dosáhl o zhruba 10% vyšší přesnosti u koeficientu p_4 a naopak horší u p_3 . Z rychlosti výpočtu je ale patrné, že algoritmický přístup zvládl definovanou úlohu rychleji a v testovaných podmínkách je lepší volbou. Navíc k tomu nebyl potřeba žádný dataset ani grafická karta. Hluboké učení se tak na tento úkol jeví dle výsledků jako příliš komplikované, bez využití jeho plného potenciálu.

Cílem této bakalářské práce byla analýza systémů pro udržení vozidla v jízdním pruhu a praktická demonstrace dvou vybraných. V první teoretické části byly popsány tři přístupy k řešení úlohy a to algoritmický, pomocí zpětnovazebního a hlubokého učení. Zde bylo předestřeno jakým způsobem jednotlivé systémy fungují a co je k jejich funkci potřeba. Zmíněna byla také omezení v podobě variability prostředí a světla, která mají při použití kamer zásadní vliv.

V Druhé části byly představeny simulátory použitelné pro substituci fyzického prostředí pro vývoj a testování přístupů prezentovaných v první části a také veřejně dostupné anotované datasety vhodné pro úlohy týkající se autonomního řízení.

V poslední části byla popsána implementace algoritmického přístupu na reálných datech spolu s hlubokým učení. Algoritmický pracoval na bázi hranové detekce pomocí směrových filtrů a přístup hlubokého učení byl navržen jako dvou-
stupňový s použitou segmentační sítí s architekturou U-Net.

Oba přístupy byly porovnány na stejných validačních datech, což vyústilo v překvapivě nerozhodné výsledky, kdy oba přístupy došly s podobnou chybou ke stejnému výsledku. Vhodnější metoda pak byla stanovena podle času extrakce čar, kde získala lepší výsledky Algoritmická metoda, konkrétně 7× lepší s průměrným časem 0.0110s. Hlavním problémem pomalé predikce NN bylo kopírování výsledků z paměti GPU do paměti operační. V diskuzích se však objevuje, že se jedná o specifické chování Python knihovny TensorFlow. Jako možné řešení se jeví buď optimalizovat model a transformovat ho například do jiného formátu nebo použít jiný framework.

Další zlepšení výsledků by mohlo být dosaženo použitím sofistikovanější metody shlukování bodů čar, použitím více kamer nebo méně lesklého povrchu dráhy.

S pomocí analýzy a výsledků obsažené v této práci lze vybrat vhodný přístup pro detekci jízdního pruhu v jiném systému. Zároveň je možné tuto práci použít jako základ vlastního autonomního systému s pokročilejším řízením a dalšími funkcemi, jako například plánování trasy.

Bibliografie

1. MCCALL, J.C.; TRIVEDI, M.M. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*. 2006, roč. 7, č. 1, s. 20–37. ISSN 1558-0016. Dostupné z DOI: 10.1109/TITS.2006.869595.
2. KUTELA, Boniphace; DAS, Subasish; DADASHOVA, Bahar. Mining patterns of autonomous vehicle crashes involving vulnerable road users to understand the associated factors. *Accident Analysis & Prevention*. 2022, roč. 165, s. 106473. ISSN 0001-4575. Dostupné z DOI: <https://doi.org/10.1016/j.aap.2021.106473>.
3. LIU, Qian; WANG, Xuesong; WU, Xiangbin; GLASER, Yi; HE, Linjia. Crash comparison of autonomous and conventional vehicles using pre-crash scenario typology. *Accident Analysis & Prevention*. 2021, roč. 159, s. 106281. ISSN 0001-4575. Dostupné z DOI: <https://doi.org/10.1016/j.aap.2021.106281>.
4. CHEN, Kaiping; TOMBLIN, David. Using Data from Reddit, Public Deliberation, and Surveys to Measure Public Opinion about Autonomous Vehicles. *Public Opinion Quarterly*. 2021, roč. 85, č. S1, s. 289–322. ISSN 0033-362X. Dostupné z DOI: 10.1093/poq/nfab021.
5. COMMITTEE, On-Road Automated Driving (ORAD). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2021. Dostupné z DOI: https://doi.org/10.4271/J3016_202104.
6. SAE. 2021. Dostupné také z: https://www.sae.org/standards/content/j3016_202104.
7. NI, Jianjun et al. A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods. *Applied Sciences*. 2020, roč. 10. Dostupné z DOI: 10.3390/app10082749.
8. ELIOU, Nikolaos; KALIABETSOS, Georgios. A new, simple and accurate transition curve type, for use in road and railway alignment design. *European Transport Research Review*. 2013, roč. 6, č. 2, s. 171–179. Dostupné z DOI: 10.1007/s12544-013-0119-8.

9. FREEMAN, William T; ADELSON, Edward H et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*. 1991, roč. 13, č. 9, s. 891–906.
10. KIM, Sung; CASPER, Riley. Applications of convolution in image processing with MATLAB. *University of Washington*. 2013, s. 1–20.
11. SOBEL, Irwin. An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*. 2014.
12. MUKHOPADHYAY, Priyanka; CHAUDHURI, Bidyut B. A survey of Hough Transform. *Pattern Recognition*. 2015, roč. 48, č. 3, s. 993–1010. ISSN 0031-3203. Dostupné z DOI: <https://doi.org/10.1016/j.patcog.2014.08.027>.
13. SCIUTTO, S.J. Polyfit — A package for polynomial fitting. *Computer Physics Communications*. 1989, roč. 52, č. 3, s. 427–442. ISSN 0010-4655. Dostupné z DOI: [https://doi.org/10.1016/0010-4655\(89\)90117-3](https://doi.org/10.1016/0010-4655(89)90117-3).
14. SUTTON, Richard S.; BARTO, Andrew. *Reinforcement learning: An introduction*. The MIT Press, 2020.
15. KIRAN, B Ravi et al. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*. 2022, roč. 23, č. 6, s. 4909–4926. ISSN 1558-0016. Dostupné z DOI: 10.1109/TITS.2021.3054625.
16. RUBÍ, Bartomeu; MORCEGO, Bernardo; PÉREZ, Ramon. A Deep Reinforcement Learning Approach for Path Following on a Quadrotor. In: 2020. Dostupné z DOI: 10.23919/ECC51009.2020.9143591.
17. *Knihovna pro vývoj RL algoritmů OpenAI Gym*. [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/openai/gym>.
18. *Neural Networks and Deep Learning*. Determination Press, 2015. Dostupné také z: Michael%20A.%20Nielsen.
19. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
20. LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, roč. 86, č. 11, s. 2278–2324. ISSN 1558-2256. Dostupné z DOI: 10.1109/5.726791.
21. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F.; BURGESS, C.J.; BOTTOU, L.; WEINBERGER, K.Q. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, sv. 25. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

22. SIMONYAN, Karen; ZISSERMAN, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1409.1556>.
23. SZEGEDY, Christian; VANHOUCKE, Vincent; IOFFE, Sergey; SHLENS, Jonathan; WOJNA, Zbigniew. *Rethinking the Inception Architecture for Computer Vision*. 2015. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1512.00567>.
24. LIN, Min; CHEN, Qiang; YAN, Shuicheng. *Network In Network*. 2014. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1312.4400>.
25. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1512.03385>.
26. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, s. 580–587. ISSN 1063-6919. Dostupné z DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81).
27. REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 779–788. ISSN 1063-6919. Dostupné z DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
28. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1505.04597>.
29. HAO, Shijie; ZHOU, Yuan; GUO, Yanrong. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*. 2020, roč. 406, s. 302–321. ISSN 0925-2312. Dostupné z DOI: <https://doi.org/10.1016/j.neucom.2019.11.118>.
30. TANI, Jacopo et al. Duckietown: An Innovative Way to Teach Autonomy. In: ALIMISIS, Dimitris; MORO, Michele; MENEGATTI, Emanuele (ed.). *Educational Robotics in the Makers Era*. Cham: Springer International Publishing, 2017, s. 104–121. ISBN 978-3-319-55553-9. Dostupné z DOI: https://doi.org/10.1007/978-3-319-55553-9_8.
31. *The Duckietown library* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://docs-old.duckietown.org/daffy/>.
32. CHEVALIER-BOISVERT, Maxime; GOLEMO, Florian; CAO, Yanjun; MEHTA, Bhairav; PAULL, Liam. *Duckietown Environments for OpenAI Gym* [<https://github.com/duckietown/gym-duckietown>]. GitHub, 2018.

33. ZILLY, Julian et al. The AI Driving Olympics at NeurIPS 2018. *arXiv preprint arXiv:1903.02503*. 2019.
34. SHAH, Shital; DEY, Debadeepta; LOVETT, Chris; KAPOOR, Ashish. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: *Field and Service Robotics*. 2017. Dostupné z eprint: arXiv:1705.05065.
35. *GitHub repository simulátoru AirSim* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/microsoft/AirSim>.
36. *Dokumentace simulátoru AirSim* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://microsoft.github.io/AirSim/>.
37. *Popis standardu OpenDrive[®]* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.asam.net/standards/detail/opendrive/>.
38. [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: https://docs-old.duckietown.org/daffy/AID0/out/object_detection_dataset.html.
39. FRITSCH, Jannik; KÜHNEL, Tobias; GEIGER, Andreas. A new performance measure and evaluation benchmark for road detection algorithms. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 2013, s. 1693–1700. ISSN 2153-0017. Dostupné z DOI: 10.1109/ITSC.2013.6728473.
40. *KITTI dataseť* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.cvlibs.net/datasets/kitti/>.
41. YOGAMANI, Senthil et al. WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving. *arXiv preprint arXiv:1905.01489*. 2019.
42. *Valeo Woodscape dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://woodscape.valeo.com/woodscape/>.
43. *Audi A2D2 dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.a2d2.audi/a2d2/en.html>.
44. GEYER, Jakob et al. A2D2: Audi Autonomous Driving Dataset. 2020. Dostupné z arXiv: 2004.06320 [cs.CV].
45. HUANG, Xinyu et al. The ApolloScape Open Dataset for Autonomous Driving and Its Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020, roč. 42, č. 10, s. 2702–2719. ISSN 1939-3539. Dostupné z DOI: 10.1109/TPAMI.2019.2926463.
46. *Apollo projekt* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.apollo.auto/>.
47. *ApolloScape dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://apolloscape.auto/index.html>.

48. CHANG, Ming-Fang et al. Argoverse: 3D Tracking and Forecasting With Rich Maps. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, s. 8740–8749. ISSN 2575-7075. Dostupné z DOI: 10.1109/CVPR.2019.00895.
49. WILSON, Benjamin et al. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In: VANSCHOREN, J.; YEUNG, S. (ed.). *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Curran, 2021, sv. 1. Dostupné také z: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf.
50. *Argoverse dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.argoverse.org/index.html>.
51. CORDTS, Marius et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1604.01685>.
52. *Cityscapes Dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.cityscapes-dataset.com/>.
53. *Webové stránky společnosti Comma* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://comma.ai/>.
54. SCHAFER, Harald; SANTANA, Eder; HADEN, Andrew; BIASINI, Riccardo. *A Commute in Data: The comma2k19 Dataset*. 2018. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1812.05752>.
55. *Knihovna pro zpracování GNSS* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/commaai/laika>.
56. *Comma2k16 dataset GitHub* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/commaai/comma2k19>.
57. *Comma2k16 dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <http://academictorrents.com/details/65a2fbc964078aff62076ff4e103f18b951c5ddb>.
58. YU, Fisher et al. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, s. 2633–2642. Dostupné z DOI: 10.1109/CVPR42600.2020.00271.
59. *BDD100K dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://bdd-data.berkeley.edu/>.
60. *PixSet dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://leddartech.com/solutions/leddar-pixset-dataset/>.

61. DÉZIEL, Jean-Luc et al. *PixSet : An Opportunity for 3D Computer Vision to Go Beyond Point Clouds With a Full-Waveform LiDAR Dataset*. 2021. Dostupné z DOI: <https://doi.org/10.48550/arXiv.2102.12010>.
62. *NuScenes dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.nuscenes.org/>.
63. MADDERN, Will; PASCOE, Geoff; LINEGAR, Chris; NEWMAN, Paul. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*. 2017, roč. 36, č. 1, s. 3–15. Dostupné z DOI: 10.1177/0278364916679498.
64. *Oxford RobotCar dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://robotcar-dataset.robots.ox.ac.uk/>.
65. *PandaSet dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://pandaset.org/>.
66. *Udacity Self-Driving Car Project* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/udacity/self-driving-car/tree/master>.
67. *Udacity Car dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/udacity/self-driving-car/tree/master/annotations>.
68. *Waymo Open dataset* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://waymo.com/open/>.
69. *Kaggle* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.kaggle.com/>.
70. *Roboflow* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://public.roboflow.com/>.
71. *Kit robota* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://www.sparkfun.com/products/18486>.
72. BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 978-0-596-51613-0. 2008.
73. OTSU, Nobuyuki. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*. 1979, roč. 9, č. 1, s. 62–66. Dostupné z DOI: 10.1109/TSMC.1979.4310076.
74. *Implementace metody klouzavého okénka*. [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://github.com/muddassir235/Advanced-Lane-and-Vehicle-Detection>.
75. *Dokumentace formátu COCO pro anotace datasetů*. [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: <https://cocodataset.org/#format-data>.

76. *Repository s uloženými zdrojovými kódy této bakalářské práce.* [online]. [B.r.]. [cit. 2023-07-26]. Dostupné z: https://github.com/Fiila/BPARR_2023.

Seznam obrázků

| | | |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Graf zobrazující úroveň automatizace od 0 (bez automatizace) do 5 (plně autonomní). Převzat ze stránek společnosti SAE viz. [6]. | 4 |
| 2.1 | Eulerova křivka. | 9 |
| 2.2 | Část Eulerovy křivky. | 10 |
| 2.3 | Fotografie různých variant stavu značení nebo podmínek. Cesta (a) zobrazující optimální plnou a přerušovanou čáru, (b) nejednotlivý povrch vozovky, (c) poškozené horizontální značení, (d) špatná kvalita čar a přítomnost stínů (e) a (f). Zdroj [1]. | 12 |
| 2.4 | Vizualizace základní konvoluce na 2D matici reprezentující obrázek. . | 13 |
| 2.5 | Vizualizace Gaussova kruhově symetrického jádra (nalevo) a jeho derivace podle x (napravo) ve 3D i 2D perspektivě. | 13 |
| 2.6 | Blokové schéma navrženého kaskádního regulátoru s dvěma vstupy, kde R1 reprezentuje vnější a R2 vnitřní smyčku. | 14 |
| 2.7 | Schéma základní struktury algoritmu zpětnovazebního učení. | 15 |
| 2.8 | Schéma RL metody actor-critic. Zdroj [16]. | 18 |
| 2.9 | Základní jednotka neuronové sítě - perceptron. Zdroj [18]. | 19 |
| 2.10 | Jednoduchá ukázka architektury vytvořená spojováním perceptronů. Zdroj [18]. | 20 |
| 2.11 | Struktura neuronové sítě U-Net definována v [28]. Zdroj [28]. | 21 |
| 3.1 | Ukázka prostředí simulátoru gym-duckietown. | 26 |
| 3.2 | Ukázka prostředí simulátoru AirSim včetně možných pohledů senzorů. . | 26 |
| 3.3 | Ukázka prostředí simulátoru Carla včetně možných pohledů senzorů. . | 27 |
| 4.1 | Použitý kit robota s jednodeskovým počítačem Jetson Nano B01. Zdroj [71]. | 31 |
| 4.2 | Vytvořená testovací trasa. | 32 |
| 4.3 | Zobrazení radiálního zkreslení při prostupu světelných paprsků čočkou. Zdroj [72, p. 376]. | 32 |

| | | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.4 | Zobrazení tangenciálního zkreslení při nepřesném uložení objektivu vůči senzoru v levné kameře nalevo. Zdroj [72, p. 377] | 33 |
| 4.5 | Ukázka jednoho snímku z kalibračního datasetu ss šachovnicí s rozměry 9x6. | 34 |
| 4.6 | Ukázka snímku na Obrázku 4.5 po korekci s vyznačeným regionem s minimem nežádoucích pixelů. | 35 |
| 4.7 | Vstupní snímek pro detekci. | 35 |
| 4.8 | Schéma homografické transformace mezi dvěma rovinami. Zdroj [72, p. 385] | 36 |
| 4.9 | Použitá šachovnice pro homografickou transformaci. Zdroj [31] | 36 |
| 4.10 | Aplikace homografické transformace na vstupní obraz (a), (c), kterou vznikly odpovídající (b) a (d). | 37 |
| 4.11 | Ukázka neúspěšného pokusu prahování podle barvy. | 38 |
| 4.12 | Detekované hrany pomocí orientovaných filtrů v Obrázku 4.10(d). | 38 |
| 4.13 | Zpracovaný binární obrázek metodou tzv. klouzavých okének včetně již rozdělených pixelů proložených kubickými polynomy, růžově vyznačeným středem obrazu a červenou tečnou středové trajektorie. | 39 |
| 4.14 | Originální obrázek v datasetu (nalevo) spolu s jeho anotací (napravo). | 41 |
| 4.15 | Průběh trénování sítě U-Net. | 42 |
| 4.16 | Srovnání vstupu (vlevo), ručně vytvořené masky (uprostřed) a predikované masky neuronovou sítí U-Net. | 43 |
| 4.17 | Grafy s naměřenými časy trvání jednotlivých částí detekce čar. | 43 |
| 4.18 | Grafy se změřenými absolutními odchylkami parametrů středové trajektorie od reference. | 44 |

Seznam tabulek

| | | |
|-----|-----------------------------------------------------------------------|----|
| 4.1 | Porovnání přesnosti detekce koeficientů středové trajektorie. | 44 |
|-----|-----------------------------------------------------------------------|----|

