VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PREDPOVEDANIE TRAJEKTÓRIE VOZIDIEL A CHODCOV PRE ASISTENČNÉ SYSTÉMY RIADENIA

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK MUDROŇ**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Mudroň Marek**
Program: Informační technologie
Název: **Předpovídání trajektorie vozidel a chodců pro asistenční systémy řízení**
Predicting Trajectories of Vehicles and Pedestrians for Driving Assistent Systems
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se se způsoby rozpoznávání typických účastníků dopravního provozu a s dostupnými implementacemi pro snadnou konfiguraci a trénování neuronových sítí.
2. Shromážděte datové sady pro průběžné testování systému.
3. Na základě získaných poznatků navrhněte a implementujte systém, který dokáže predikovat trajektorie vozidel a chodců i ve složitějších situacích, případně detekovat a hodnotit nebezpečí kolizních situací.
4. Vyhodnoťte výsledky systému na reprezentativním vzorku dat.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

V tejto práci sa zaoberám vytváraním reprezentácie dopravnej scény pomocou spracovania monokulárneho záznamu. Na základe vytvorenej reprezentácie sa snažím predpovedať trajektórie detekovaných vozidiel v krátkom časovom horizonte 2 sekúnd. Súčasné postupy využívajú viacero drahých senzorov na zhromažďovanie okamžitých informácií o prostredí. Predstavujem postup, ktorý pomocou viacerých modelov strojového učenia, dokáže z videa zistiť rovnaké údaje, ako spomínané senzory. Výsledkom je systém umožňujúci zníženie nákladov na senzory pre vytváranie reprezentácie prostredia a predikciu trajektórie vozidiel v scéne. Ďalším prínosom je porovnanie presnosti modelov, trénovaných na odlišne spracovaných dátach. Pri porovnaní poskytujem údaje o tom, ako veľmi sa približujú k presnosti najspoľahlivejších predikčných systémov.

Abstract

This bachelor thesis deals with representation of a traffic scene by processing monocular video sequence. I try to predict a trajectory of detected vehicles in a short time horizon, based on created representation. Current approaches use multiple expensive sensors to gather instant information of environment. In the thesis I introduce technique, which is able to extract data from an environment by image processing techniques without the need of expensive sensors. The result of this work is a system creating opportunity to reduce the sensor costs of a system for scene representation and trajectory prediction of vehicles in the scene. In addition, comparison of models trained on differently processed data is provided, as well as data about how my system approximates the most reliable prediction models.

Kľúčové slová

predikcia trajektórie, autonómne systémy, plánovanie trasy, detekcia objektov, spracovanie obrazu, rastrová reprezentácia, 3D detekcia, sledovanie objektov, lokalizácia, sémantická segmentácia, monokulárny záznam

Keywords

trajectory prediction, autonomous systems, path estimation, object detecion, image processing, raster representation, 3D detection, object tracking, localization, semantic segmentation, monocular video

Citácia

MUDROŇ, Marek. *Predpovedanie trajektórie vozidiel a chodcov pre asistenčné systémy riadenia.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Predpovedanie trajektórie vozidiel a chodcov pre asistenčné systémy riadenia

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. RNDr. Pavla Smrža Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Marek Mudroň
10. mája 2022

Poďakovanie

Rád by som poďakoval pánovi doc. RNDr. Pavlu Smržovi, Ph.D. za cenné rady pri návrhu systému, odporúčania pri problémoch, a za ochotu a pomoc pri tvorbe bakalárskej práce.

Obsah

Kapitola 1

Úvod

Čo robí ľudí schopnými vynájsť sa v nečakaných situáciách je schopnosť učiť sa z vlastnej skúsenosti, odvolávať sa na minulé udalosti a zovšeobecňovať tieto znalosti pre použitie v budúcnosti. V posledných rokoch bolo vynaložené mnoho úsilia na prenesenie týchto schopností do vývoja autonómnych vozidiel tak, aby boli schopné sa orientovať v komplexných scenároch s viacerými objektami. Plánovanie trasy pre bezpečnú navigáciu v takýchto prostrediach sa však nemôže spoliehať len na súčasnú polohu a pohyb okolitých objektov. Namiesto toho vyžaduje predpovedanie neznámych premenných. Jednou z nich je aj dynamický pohyb účastníkov scény.

Najmodernejšie systémy pre predikciu trajektórie účastníkov cestnej premávky, či už chodcov, vozidiel, alebo cyklistov, využívajú pre tvorbu reprezentácie prostredia širokú škálu senzorov, od lidaru, radarov, sonarov až po niekoľko kamier. Väčšina takýchto systémov má prístup ku grafovej reprezentácii scény či k HD vektorovým mapám, ktoré obsahujú podrobné geometrické informácie o objektoch, ako sú jazdné pruhy, prechody pre chodcov, značky a ďalšie. Samotné predikčné systémy pracujú s touto spracovanou reprezentáciou prostredia, no v mnohých prípadoch autori neuvádzajú, a nie je možné určiť, ako bol daný vstup spracovaný do formy vhodnej pre predikčný model.

Preto sa táto práca okrem samotnej predikcie pohybu vozidiel venuje aj problematike tvorby vhodnej reprezentácie okolitého prostredia. Jej cieľom je konať len za použitia monokulárneho záznamu, ktorý oproti použitiu drahých senzorov vyžaduje len jednu kameru. Týmto vytvára príležitosť pre zníženie nákladov na tvorbu systému s účelom predpovedania trajektórie. Na základe toho navrhujem model pozostávajúci z dvoch častí. Prvá časť je zodpovedná za tvorbu vhodnej reprezentácie prostredia a druhá časť spracúva túto reprezentáciu a vytvára predikciu.

Hneď v nasledujúcej kapitole je definovaná problematika extrakcie priestorových informácií z monokulárneho obrazu nevyhnutných pre predikciu trajektórie detekovaných vozidiel v scéne. V nej sú najskôr v krátkosti opísané vybrané typy neurónových sietí, ktoré sú pre moju prácu relevantné, keďže sú nevyhnutným prvkom pre spracovanie obrazu a časovo závislých postupností dát. Kapitola ďalej popisuje techniky používané pre tvorbu reprezentácie prostredia. Tie zahrňujú odhad vzdialeností prvkov v scéne 2.2.1 a ich následnú projekciu do 3D množiny bodov. V podkapitole 2.2.2 pokračuje opisom spôsobu, ktorým sa vytvoreným bodom priraďuje význam či sémantická trieda. Následne je uvedený spôsob projekcie množiny 3D bodov do 2D roviny. Táto kapitola sa v podkapitole 2.3 ďalej venuje zisťovaniu pozícií objektov v scéne. Obsahom je aj sledovanie týchto detekcií medzi susednými snímkami, z čoho získavam minulú trajektóriu zistených objektov. Taktiež uvádza nástroj, pomocou ktorého je možné zistiť relatívnu polohu kamery v prostredí len na

základe kamerového záznamu. V sekcii 2.4 popisujem spôsob, pomocou ktorého prebieha lokalizácia kamery v scéne bez GPS či IMU senzorov. Kapitola končí opisom predikčného modelu, ktorý na základe minulej trajektórie vozidla dokáže odhadnúť jeho budúcu pozíciu. Taktiež sú v sekcii 2.5 uvedené metriky, pomocou ktorých sa meria spoľahlivosť predikcií.

Kapitola 3 navrhuje postup, ktorým pristupujem k problému definovanému v predchádzajúcej kapitole. Predstavuje schému zreťazeného spracovania, kde je pomocou techník spracovania obrazu vytváraná vhodná interpretácia prvkov prostredia pomocou monokulárneho kamerového záznamu. Nakoniec opisuje spôsob, akým bola táto informácia spracovaná predikčným modelom.

Vo štvrtej kapitole sú uvedené experimenty a ich výsledky, vykonané na vytvorenom systéme predikcie z monokulárneho obrazu. Cieľom experimentov bolo zistiť, do akej miery sa dokáže presnosť predikcii vytvoreného systému priblížiť presnosti zložitejších a presnejších modelov. Ďalej obsahuje porovnanie s rovnakým predikčným modelom, ktorý však využíval viacero senzorov. V tejto kapitole sú taktiež uvedené zhodnotenia, nedostatky a návrhy na ich vylepšenie.

Práca je ukončená záverom, kde na základe získaných výsledkov zhodnocujem jej prínos pre moderné autonómne systémy.

Kapitola 2

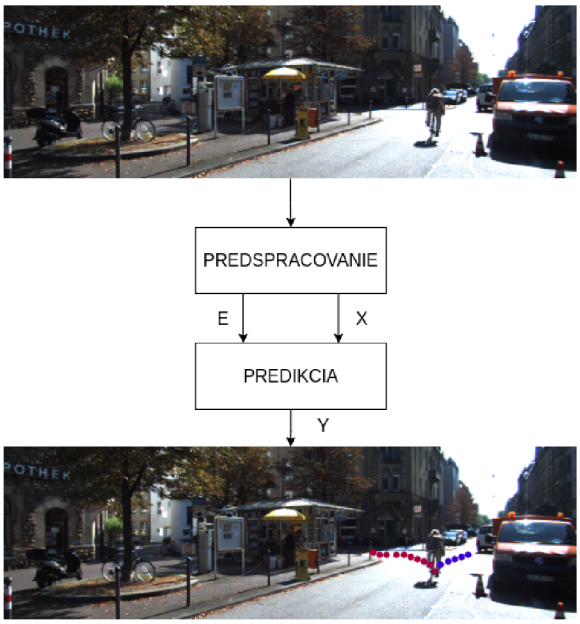
Rozbor problematiky

Cieľom tejto kapitoly je oboznámiť čitateľa s riešeným problémom a s nástrojmi, pomocou ktorých je možné spracovať monokulárny obraz a získať sémantickú 2D mapu reprezentujúcu okolie.

Práca sa predovšetkým venuje predikcii viacerých trajektórií, najmä v mestských oblastiach. Príkladmi, kde môžu takéto predpovede byť nevyhnutné, sú krízové body cestných komunikácií ako križovatky a kruhové objazdy, kde objekty volia z viacerých, rovnako pravdepodobných trás. Predpovedanie trajektórie je možné vyjadriť ako funkciu

$$y = f(x, e)$$

kde x je sledovaná trajektória, e je reprezentácia prostredia a y predstavuje predpovedanú trajektóriu. Avšak vstupné premenné nie sú priamo obsiahnuté v monokulárnom obraze, a preto je potrebné ich z neho extrahovať komplexným spracovaním. V tejto kapitole sa ďalej venujem postupom, pomocou ktorých sú objekty v scéne detekované, a pomocou ktorých sa sleduje ich chronologická závislosť medzi susednými snímkami. Nakoniec spomínam akým spôsobom prebieha predikcia trajektórie a opisujem použitý predikčný model.



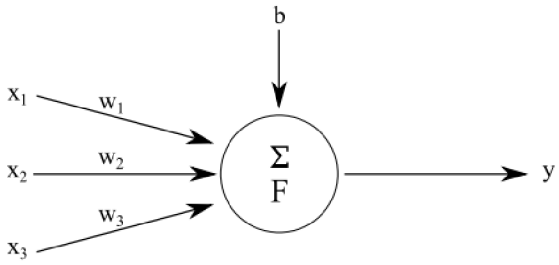
Obr. 2.1: Príklad situácie, kde je predikcia trajektórie užitočná, keďže môže ochrániť cyklistu, od náhleho zrýchlenia vozidla v dôsledku toho, že vodič predpokladá, že cyklista bude odbočovať po ružovej trase, no v skutočnosti pokračuje po modrej, prevzaté a upravené z datasetu KITTI [5]

2.1 Dopredné neurónové siete

V tejto časti sa venujem opisu fungovania vybraných druhov neurónových sietí, a čerpám najmä z publikácií [6][8][12].

Jedným z najviac používaných nástrojov pre spracovanie obrazu sú neurónové siete. Ide o výpočtový model využívaný v oblasti strojového učenia, zostavený na základe abstrakcie biologický neurónov v nervovom systéme. Neurónová sieť pozostáva zo základných výpočtových jednotiek neurónov, medzi ktorými sú vytvorené väzby. Vstupom každého z neurónov je n číselných hodnôt, ktoré tvoria vstupný vektor $\overrightarrow{x}$ pozostávajúci z týchto zložiek. Následne sa vykonáva vektorové násobenie vstupného vektora a vektora váh $\overrightarrow{w}$. K výsledku tejto operácie sa pričítava hodnota b taktiež zvaná *bias*. Výsledná hodnota je použitá ako vstup pre aktivačnú funkciu $f(x)$. Výsledok získaný z aktivačnej funkcie pre danú hodnotu je považovaná za výstup neurónu y [12].

$$y = f(b + \sum_{i=1}^{n} w_i x_i)$$

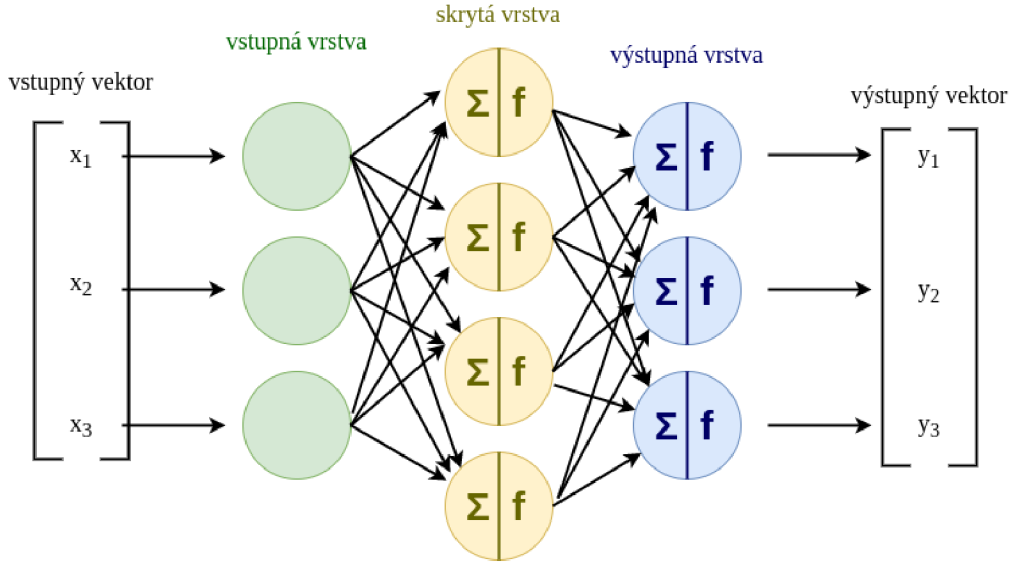


Obr. 2.2: Schéma neurónu, prevzaté a upravené z [12]

Príkladom aktivačnej funkcie je funkcia $ReLU$ (angl. Rectified Linear Unit) [1], ktorá mapuje vstupné hodnoty do intervalu $\langle 0, \infty \rangle$. Je využívaná predovšetkým z dôvodu minimalizácie problému miznúceho gradientu [12], kedy učenie vrstiev vzdialenejších od výstupnej vrstvy prebieha pomalšie z dôvodu použitia reťazového pravidla pri derivácii zloženej funkcie. Predpis funkcie ReLU je daný vzťahom

$$r(x) = \begin{cases} 0 & \text{pre x} < 0 \\ x & \text{pre x} \geq 0 \end{cases}$$

Dopredná neurónová sieť sa skladá z jednotlivých neurónov, ktoré sú organizované do vrstiev. Vstupná vrstva slúži na odovzdanie vstupného vektora $\overrightarrow{x}$ neurónovej sieti. Za ňou nasleduje niekoľko skrytých vrstiev. Jednotlivé výstupy neurónov, lokalizovaných vo výstupnej vrstve, dohromady tvoria výstupný vektor $\overrightarrow{y}$. Vstupom neurónov v nasledujúcich vrstvách sú výstupné hodnoty neurónov z predchádzajúcich vrstiev, a v prípade vstupnej vrstvy je to vektor $\overrightarrow{x}$.



Obr. 2.3: Schéma doprednej neurónovej siete, prevzaté a upravené z [6]

2.1.1 Trénovanie neurónových sietí

Aby neurónová sieť dokázala riešiť úlohy spojené s danou problematikou, je potrebné určiť parametre, pomocou ktorých vypočíta výstup na základe vstupu. Spomenuté parametre sa neurónová sieť musí naučiť, tomuto procesu sa hovorí *trénovanie*. Pred fázou trénovania je nutné pripraviť trénovaciu dátovú sadu tvorenú položkami, z ktorých každá pozostáva zo vstupného a výstupného vektora.

Na začiatku sa v neurónovej sieti náhodne inicializujú váhy vzhľadom na vybrané štatistické rozdelenie. Následne sa pre každý trénovací vstup dopredným prechodom vypočíta výstup. Ten sa potom porovná s výstupným vektorom príslušného vstupu. Toto porovnanie prebieha pomocou *stratovej funkcie* (angl. loss function, taktiež cost function) [6]. Čím je hodnota získaná z tejto funkcie nižšia, tým bližšie sa neurónová sieť blíži k správnemu výsledku.

So získanou hodnotou stratovej funkcie je možné optimalizovať súčasné hodnotý váh neurónovej siete. Touto optimalizáciou sa myslí úprava váh vedúca k zníženiu hodnoty stratovej funkcie a teda chyby, ktorej sa dopúšťa. Na ladenie hodnôt parametrov sa používa algoritmus *spätného šírenia chyby* (angl. backpropagation) [8], používajúci optimalizačný algoritmus *gradientného zostupu*(angl. gradient descent) [8]. Ten umožňuje iteratívne pozmeňovať hodnoty váh tak, aby sa hodnota stratovej funkcie približovala k lokálnemu minimu. Pre určenie tejto zmeny je využitý *gradient*, reprezentovaný vektorom určujúcim smer najstrmejšieho stúpania v danom bode oboru hodnôt funkcie. Pri posune váh v smere vektora opačného ku gradientu dochádza k zníženiu hodnoty stratovej funkcie, a tým k optimalizácii siete. Veľkosť posunu nie je konštantná, čím miernejší je pokles funkcie, tým menšia je veľkosť posunu [12].

Gradient pre váhu w_{t} je možné získať pomocou parciálnej derivácie stratovej funkcie E vzhľadom k váhe w_{t}. Pre zaistenie konvergencie je taktiež potrebné určiť hodnotu *miery učenia* ϵ (angl. learning rate). Gradient g_{t} pre váhu w_{t} je určený vzťahom

$$g_{\mathrm{t}} = -\epsilon \frac{\partial E}{\partial w_t}$$

Novú hodnotu váhy w_{t+1} je potom možné zistiť vzťahom

$$w_{\mathrm{t+1}} = w_{\mathrm{t}} - \epsilon \frac{\partial E}{\partial w_t}$$

Algoritmus *backpropagation* (angl. spätné šírenie chyby) [22] najprv použije vstupy z trénovacej dátovej sady na určenie výstupov neurónovej siete. Použitím reálneho a očakávaného výstupu je určená hodnota stratovej funkcie. Následne je od výstupnej vrstvy postupne až po prvú skrytú vrstvu určený gradient, a pre každú váhu určená jej nová hodnota podľa vyššie uvedeného vzťahu. Tento postup sa opakuje pre všetky trénovacie vstupné vektory [12].

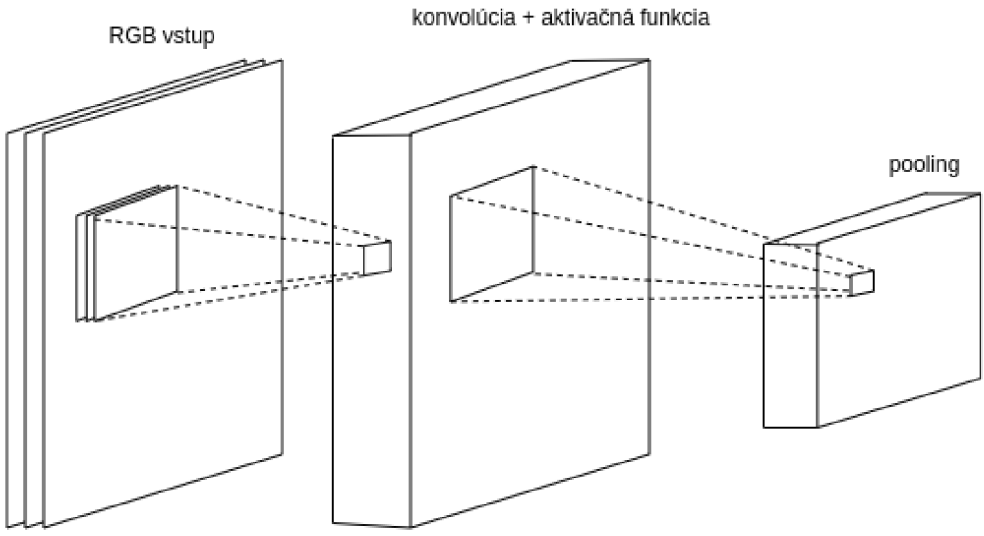
2.1.2 Konvolučné neurónové siete

Konvolučné neurónové siete [2] sú druhom neurónovej siete určenej pre spracovanie dát, ktoré majú mriežkovitú topológiu. Príklady zahŕňajú dáta z časových postupností alebo 2D mriežky pixelov. Ich názov indikuje to, že používajú matematickú operáciu *konvolúciu*. Konvolučná neurónová sieť je taká neurónová sieť, ktorá používa konvolúciu aspoň v jednej zo svojich vrstiev [8].

Skladá sa z dvoch druhov vrstiev, konvolučnej a združovacej. Vstupom konvolučnej vrstvy je vektor buď zo vstupnej vrstvy neurónovej siete alebo z predchádzajúcej vrstvy. Ďalším prvkom tejto vrstvy je minimálne jeden *konvolučný filter* (angl. kernel), ktorý je podľa požiadavok jednorozmerný, alebo v prípade spracovania obrazu, dvoj a viac rozmerný. Jeho rozmery sú menšie ako veľkosť vstupu.

Filter je aplikovaný na vstupný vektor pomocou vektorového násobenia. Výsledok z tejto operácie sa uloží do vektora obsahujúceho vlastnosti zo vstupu (angl. feature map) [8]. Následne sa filter posunie o predom definovanú hodnotu kroku (angl. stride), a proces sa znovu opakuje. Po prejdení celým vstupom vznikne spomenutá mapa vlastností. Na túto mapu sa následne aplikuje aktivačná funkcia.

V ďalšej fáze je aplikovaná operácia združovania (angl. pooling) [2]. Združovacia funkcia nahradí výstup konvolučnej vrstvy na určitej pozícii výstupom štatistickej funkcie, ktorej vstupom sú susedné hodnoty v tejto pozícii. Operácia združovania pomáha k tomu, aby bola reprezentácia čo najviac odolná voči malým posunom (transláciám) vo vstupe. Odolnosťou voči posunom rozumieme to, že ak vstup posunieme o malú hodnotu, tak sa hodnoty výstupu združovacej funkcie nemenia [8].



2.1.3 Rekurentné neurónové siete

Rekurentné neurónové siete [8] sú druhom neurónových sietí umožňujúcich spracovávať sekvenčné dáta. Táto sekvencia dát je v tvare $x^{(1)}, ..., x^{(t)}$.

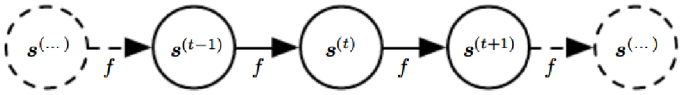
Príklad dynamického systému

$$s^{(t)} = f(s^{(t-1)})$$

kde $s^{(t)}$ predstavuje stav systému v časovom okamihu t. Rovnica je rekurentná, keďže stav v čase t je závislý od stavu v čase $t-1$. Pre konečný počet časových intervalov $t = \tau$ je možné výraz rozvinúť

$$s^{(\tau)} = f(s^{(\tau-1)})$$
$$s^{(\tau)} = f(f(s^{(\tau-2)}))$$

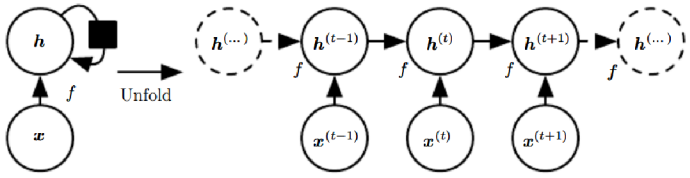
Opakovaným rozvíjaním výrazu je možné získať výraz, ktorý nebude obsahovať rekurenciu. Takýto výraz môže byť reprezentovaný acyklickým výpočtovým grafom



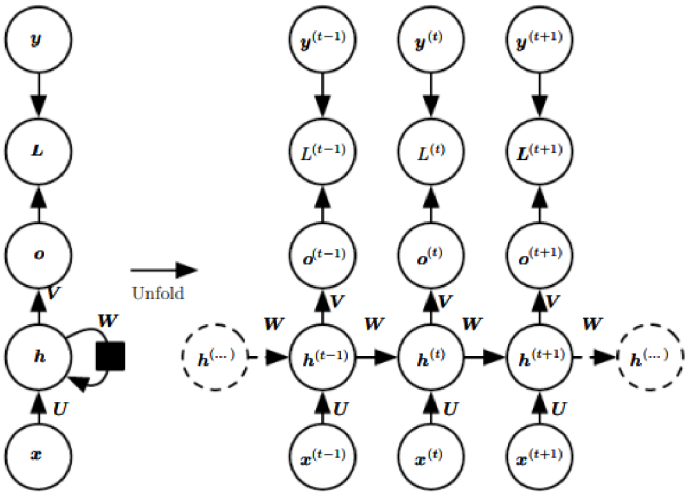
Dynamický systém, ktorého vstupom je okrem minulého stavu aj externá premenná

$$s^{(t)} = f(s^{(t-1)}, x^{(t)})$$

Každá funkcia zahrňujúca rekurenciu môže byť považovaná za rekurentnú neurónovú sieť.



Výpočtový graf definujúci výpočet stratovej funkcie rekurentnej neurónovej siete mapujúcej vstupnú sekvenciu x hodnôt na príslušnú sekvenciu o výstupných hodnôt. L je stratová funkcia, U, W a V sú váhy medzi prepojeniami jednotlivých uzlov neurónovej siete.



Obr. 2.4: Schéma rekurentnej neurónovej siete, prevzaté z [8]

Vyššie uvedená schéma definuje výraz

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

2.2 Reprezentácia prostredia

Samotnej predpovedi trajektórie predchádza spracovanie monokulárneho obrazu a jeho prevod do formy vhodnej pre predikčný model. V tejto sekcii opisujem techniky, nevyhnutné pre vytvorenie vhodnej reprezentácie prostredia.

2.2.1 Odhad vzdialenosti

Pri odhade vzdialenosti ide o určenie vzdialenosti každého pixelu vzhľadom ku kamere. Úlohou nástrojov, ktoré sú na toto určené, je výpočet hĺbkovej mapy D_t z RGB vstupu I_t. Tieto techniky slúžia ako lacná náhrada LIDAR senzoru. V práci zameniteľne používam termíny "odhad hĺbky" a "odhad vzdialenosti". Je viacero spôsobov, pomocou ktorých dané modely pristupujú k trojrozmernej informácii o priestore.

Inverzná relatívna vzdialenosť

Väčšina modelov neposkytuje absolútnu hĺbku ale inverznú hĺbku R, z ktorej je možné získať absolútnu vzdialenosť pomocou vzorca

$$D = \frac{a}{R} + b$$

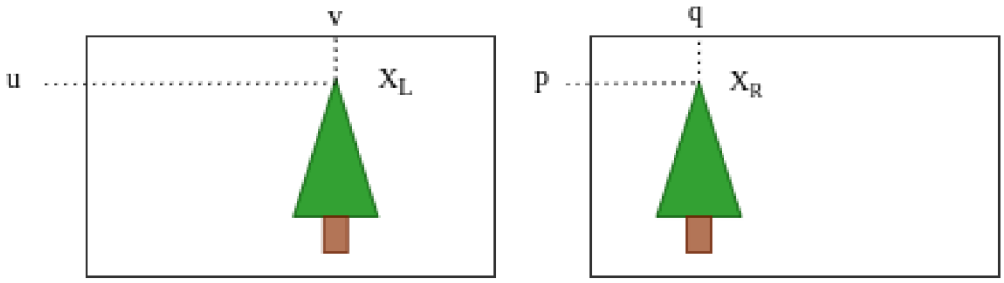
kde R je výstup modelu, predstavujúci inverznú relatívnu vzdialenosť, a je škálovací parameter a b je odchýlka. Parametre a a b sú pomocou lineárnej regresie, alebo zložitejších štatistických postupov, odhadované z aktuálnej fotografie, a preto sa môžu medzi sekvenčne idúcimi snímkami líšiť. Tento prístup je používaný modelom *MiDaS* [20].

Jeho výhodou je, že je ľahko rozšíriteľný pre kamery s rôznymi vnútornými parametrami, ktorými sú ohnisková vzdialenosť šošovky a optické centrum.

Nevýhodou je, že medzi výstupmi modelu pre jednotlivé snímky, nie je zachovaná konzistencia. Hodnota R teda môže v odlišných snímkach predstavovať rozličnú inverznú vzdialenosť. Stáva sa to najmä pri prechodoch z prostredí, kde sa mení rozsah vzdialeností. Napríklad z prostredia, kde vidno oblohu do prostredia úzkej uličky.

Odhad vzdialenosti zo stereo kamery

Ďalším spôsobom je odhad vzdialenosti zo stereo kamerového záznamu [10]. Tento záznam sa získava z dvoch kamier s rovnakými parametrami, ktoré sú orientované rovnakým smerom, no sú vzájomne horizontálne posunuté. Ak sa vzdialenosť medzi kamerami nemení, tak je možné pomocou nepomerov v obrazoch získať vzdialenosť. Tomuto nepomeru sa taktiež hovorí *disparita* a predstavuje vzdialenosť medzi pixelmi obsahujúcimi zodpovedajúci bod v pravom a ľavom obraze stereo kamier. Disparita je nepriamo úmerná vzdialenosti.



Proces určenia disparity samozrejme zahŕňa aj výber sledovaného bodu v ľavom obraze a hľadanie jemu príslušného bodu v pravom. Výber tohto bodu prebieha pomocou štatistických metód, a väčšinou ide o body, ktoré v obraze určitým spôsobom vyčnievajú, teda hrany či vrcholy objektov nachádzajúcich sa na snímkach. Priestorový bod X sa premietne do projekčnej roviny ľavej kamery v bode X_L so súradnicami (u, v). Nájdený bod má v pravom obraze súradnice (p, q).

Čím je pozorovaný bod ďalej od kamier, tým bude disparita pixelov, ktoré mu prislúchajú, menšia a so zvyšujúcou vzdialenosťou sa približujúca hodnote 0. Naopak pre blízky objekt sa hodnota disparity zvyšuje.

Disparita sa vypočíta ako veľkosť vektora medzi súradnicami týchto bodov [26]. Pomocou tohto koeficientu a parametrov kamier vieme určiť disparitu pre ostatné pixely v obraze. Priestorovú vzdialenosť jednotlivých prvkov získame pomocou vzorca

$$D = \frac{(vzdialenosť\ kamier) * (ohnisková\ vzdialenosť\ kamery)}{P}$$

kde P predstavuje vypočítanú disparitu.

Tento prístup však taktiež nepripadá do úvahy, keďže v našej práci sa snažíme pracovať len s monokulárnym záznamom.

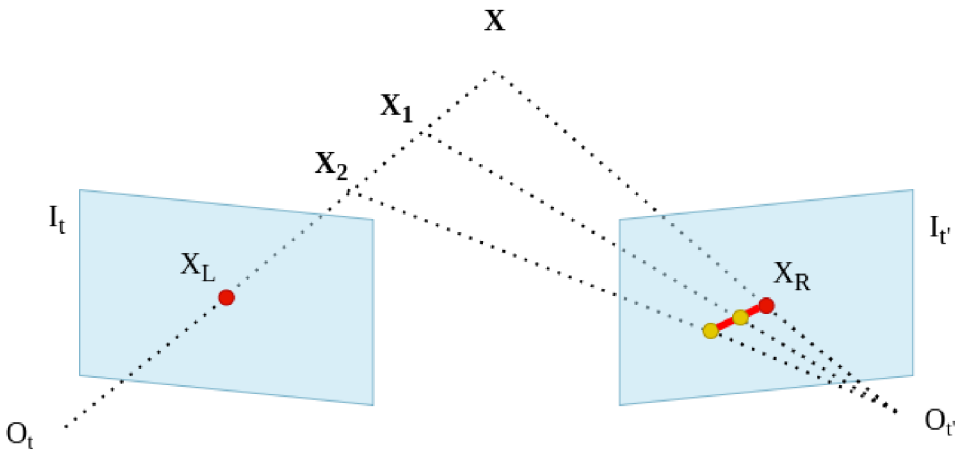
Self-supervised stereo tréning

Ďalším spôsobom, akým je možné získať hĺbkovú informáciu o scéne z monokulárneho obrazu, je pomocou inteligentného modelu. Pri jeho trénovaní sú dostupné stereo dáta. Na ich základe je možné vytrénovať hĺbkový model odhadovaním disparity medzi dvojicou kamier. Self-supervised model pre odhad hĺbky predstavuje trénovanie modelu na predpovedanie vzhľadu cieľového obrazu z pohľadu na iný obrázok. Týmto cieľovým obrazom je disparita [7]. Self-supervised learning je technika strojového učenia, kedy model získava kontrolné dáta zo samotných vstupných dát, často spracovaním ich vnútornej štruktúry. Pre naše účely je tento spôsob odhadu hĺbky vhodný.

Monodepth2

V tejto práci používam na spracovanie obrazu pre získanie hĺbky model monodepth2 [7].

Tento model formuluje problém získavania hĺbkovej informácie ako minimalizáciu chyby fotometrickej reprojekcie počas trénovania. Chyba fotometrickej reprojekcie L_p predstavuje vzdialenosť medzi vypočítanou projekciou 3D bodu do roviny a jeho skutočnou polohou v rovine. Slúži na kvantifikáciu toho, ako veľmi 3D bod X zodpovedá jeho skutočnej projekcii X_R.



Obr. 2.5: Body X_1 a X_2 predstavujú odhadnutú 3D polohu bodu X. Táto poloha je odhadovaná z ľavého obrazu I_t. Po reprojekcií tejto odhadnutej polohy do obrazu $I_{t'}$ sa chyba fotometrickej reprojekcie prejaví ako 2D vzdialenosť medzi pixelom do ktorého sa premietla jedna z odhadnutých polôh X_n a pixelom, predstavujúcim jeho skutočnú projekciu X_R.

Chyba fotometrickej reprojekcie L_p je definovaná ako

$$L_{\mathrm{p}} = \sum_{t'} pe(I_{\mathrm{t}}, I_{\mathrm{t'}\to\mathrm{t}})$$

$$I_{\mathrm{t'}\to\mathrm{t}} = I_{\mathrm{t'}}\langle proj(D_{\mathrm{t}}, T_{\mathrm{t}\to\mathrm{t'}}, K\rangle$$

kde $\mathrm{T_{t\to t'}}$ predstavuje relatívnu polohu zdrojového pohľadu $\mathrm{I_t}$ vzhľadom k polohe cieľového pohľadu $I_{t'}$. Tieto pohľady sú získane zo stereo kamier. D_t je predpovedaná hĺbka, ktorú sa model snaží vypočítať zo vstupného obrazu I_t. *pe()* reprezentuje chybu fotometrickej rekonštrukcie, čo je vzdialenosť v pixelovom priestore. *proj()* sú výsledné 2D súradnice premietnutej hĺbky D_t do $I_{t'}$, $\langle\rangle$ je vzorkovací operátor. Týmto vzorkovacím operátorom je reprezentovaný výber 3D bodov, premietnutých do cieľového obrazu $I_{t'}$.

$$pe(I_{\mathrm{a}}, I_{\mathrm{b}}) = \frac{\alpha}{2}(1 - SSIM(I_{\mathrm{a}}, I_{\mathrm{b}})) + (1-\alpha) \parallel I_{\mathrm{a}} - I_{\mathrm{b}} \parallel_1$$

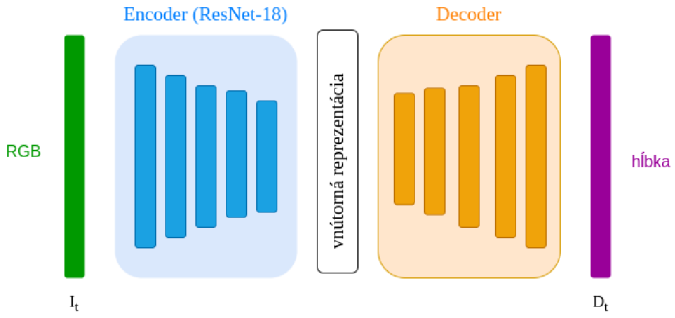
kde $\alpha = 0.85$ [7]. SSIM (structural similarity index measure) je index používaný ako metrika podobnosti dvoch obrázkov. Ide o koeficient v intervale od $\langle 0, 1\rangle$. Hodnota 1 indikuje, že dané obrazy sú *veľmi podobné alebo rovnaké*, zatiaľ čo 0 indikuje, že vstupné obrazy sú *veľmi odlišné* [28]. Táto sekcia čerpala najmä z publikácie [7].

Štruktúra modelu

Model monodepth2 [7] je tvorený neurónovou sieťou s plne konvolučnou architektúrou *encoder-decoder*.

Vstup do encoderu môže mať rozličné rozmery. Jeho výstupom je vnútorná reprezentácia vstupu o nemenných rozmeroch Decoder spracúva vytvorenú vnútornú reprezentáciu a vytvára výstup o požadovaných rozmeroch.

Ako encoder je použitá konvolučná neurónová sieť ResNet-18 [9].



Obr. 2.6: Schéma architektúry encoder-decoder, prevzaté a upravené z [7]

2.2.2 Panoptická segmentácia

Panoptická segmentácia predstavuje metódu spracovania obrazu, pomocou ktorej sú jednotlivým pixelom priradené sémantické triedy z konečnej množiny tried. Pozostáva z dvoch procesov. Proces pri ktorom sa určuje sémantická trieda objektom bez závislosti na ich počte a identifikácii sa nazýva *sémantická segmentácia*.

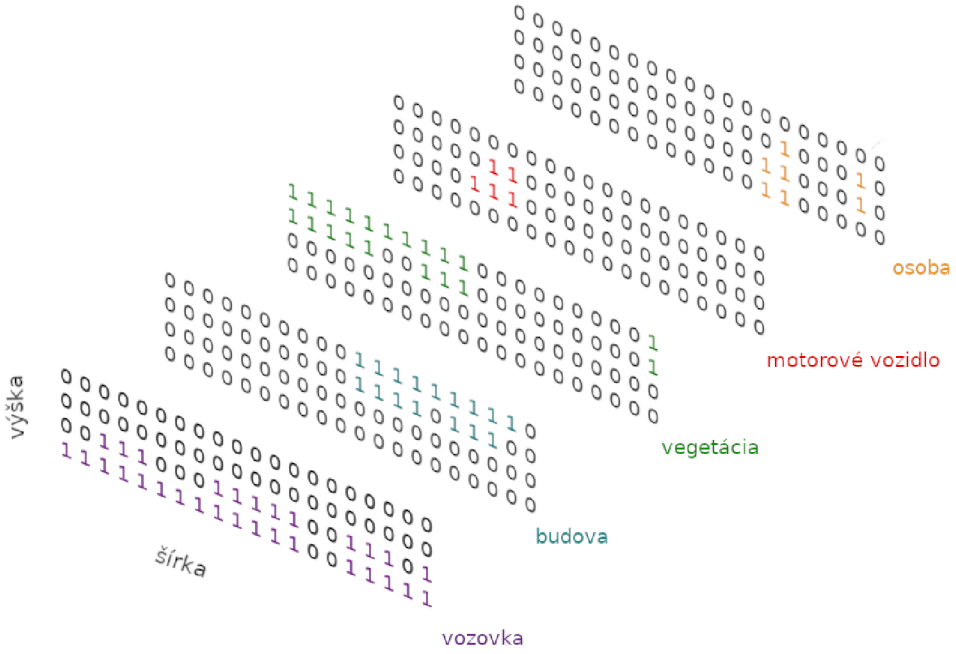
Pokiaľ sa v obraze nachádza viacero objektov zodpovedajúcich rovnakej sémantickej triede, tak je každému z nich navyše priradený jednoznačný identifikátor. Tento proces sa nazýva *inštančná segmentácia*. V mojom prípade pre určenie prvkov v scéne postačuje poznať sémantickú triedu, keďže identifikátor detekovaných objektov sledujem a priraďujem spôsobom opísaným v nasledujúcej sekcii.

Obr. 2.7: Inštančná segmentácia dokáže priradiť jednotlivým sémantickým objektom jedinečný identifikátor

Vstupom modelu sú RGB dáta s rozmermi $(h, w, 3)$. Výstupom je sémantický obraz o rozmeroch $(h, w, 1)$, kde každý pixel obsahuje identifikátor sémantickej triedy reprezentovaný celým číslom. Toto číslo je z množiny identifikátorov, ktoré daný model dokáže kategorizovať.

Podobne ako je to pri štandardných kategorických cieľových premenných, tak aj tu sa získava pomocou techniky *one-hot encoding* [6]. V podstate vytvára výstupový kanál pre každú z dostupných sémantických tried.

Výstup dokáže byť združený do segmentačnej mapy pomocou funkcie *argmax*, ktorá vracia index maximálnej hodnoty pozdĺž špecifikovaného rozmeru, v tomto prípade jednotlivých kanálov.



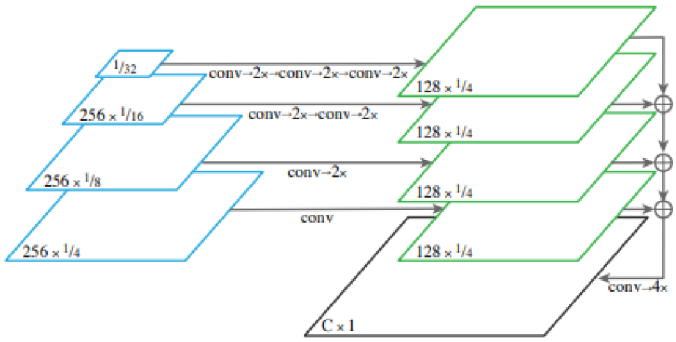
Obr. 2.8: Samostatný kanál pre každú triedu výstupu sémantickej segmentácie

Panoptic Feature Pyramid Networks

Hoci sa vyskytli pokusy o zjednotenie sémantickej a inštancie segmentácie, špecializácia, ktorá je v súčasnosti potrebná na dosiahnutie dostatočnej spoľahlivosti každej z nich, bola možno nevyhnutná vzhľadom na ich paralelný vývoj a samostatné spôsoby ohodnotenia [11].

Vzhľadom na rozdiely v architektúre týchto metód by sa dalo očakávať, že pri navrhovaní jednej siete pre obe úlohy je potrebná kompromisná presnosť buď inštančnej alebo sémantickej segmentácie, kedy jedna zlepšuje svoj výkon na úkor druhej. Panoptic Feature Pyramid Networks [11] predstavuje flexibilnú a efektívnu architektúru, ktorá dokáže zachovať presnosť pre obe úlohy pomocou jedinej siete, ktorá súčasne generuje výstupy založené na regiónoch (inštančná segmentácia) a výstupy pre sémantickú segmentáciu. V tejto práci sa zameriavam len na vetvu implementujúcu sémantickú segmentáciu.

Základom tohto modelu je FPN (Feature Pyramid Network) [15]. Táto metóda prijíma ako vstup jednorozmerný obrázok ľubovoľnej veľkosti, a výstupom sú proporcionálne veľké mapy vlastností (angl. feature maps). Model je plne konvolučný na viacerých úrovniach. Tento proces je nezávislý od použitých konvolučných architektúr.



Obr. 2.9: Vetva sémantickej segmentácie, prevzaté z [11]

Model Panoptic FPN [11] zlučuje informáciu získanú z FPN pyramídy do jednotného výstupu Začína sa od najnižšej úrovne FPN v mierke 1/32, vykonávajú sa tri štádiá prevzorkovania čím je nakoniec získaná mapa vlastností v mierke $\frac{1}{4}$. Každé štádium pozostáva zo skupinovej normalizácie (skupinová normalizácia rozdelí kanály do skupín a pre každú vypočíta priemer a rozptyl pre normalizáciu) [30], nasleduje rektifikácia pomocou aktivačnej funkcie ReLU a 2× bilineárneho prevzorkovania (funguje tak, že hodnota pridávaného pixelu sa vypočíta ako váhový priemer štyroch susedných pixelov). Opísané štádium sa opakuje aj pre FPN vrstvy v mierkach 1/16, 1/8, a 1/4 avšak progresívne sa pri každej vrstve znižuje počet štádií prevzorkovania. Výsledkom je množina máp obsahujúcich vlastnosti v rovnakej mierke 1/4, ktoré sú následne maticovo sčítané. Výsledok sa spracuje 1x1 konvolúciou, štvornásobným prevzorkovaním a aktivačnou funkciou *softmax* čím je na každý pixel určená sémantická trieda.

2.3 Detekcia objektov

Detekcia objektov je technika počítačového videnia, ktorá umožňuje identifikovať a lokalizovať objekty v obraze. Týmto spôsobom môže byť použitá na počítanie objektov v scéne, určenie trasy a presnej lokácie. Kokrétne, detekcia objektov vytvára v prípade 2D detekcie obdĺžnikové hranice, a v prípade 3D detekcie sú to hranice v tvare kvádru, ktorým sa hovorí *bounding boxy*.

Detekcia v 2D obraze síce vyznačí polohu objektov záujmu, no v tejto práci je potrebná informácia o ich polohe a orientácii v priestore, nakoľko sa neskôr snažím predpovedať ich trajektóriu vzhľadom k ich aktuálnej priestorovej polohe. Preto som nútený využiť pokročilejšie techniky, ktoré dokážu určiť aj tieto priestorové informácie.

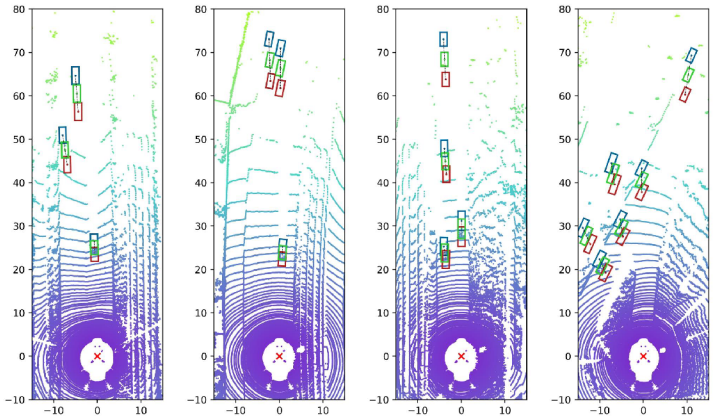
2.3.1 Detekcia v priestore

Monokulárnej 3D detekcii sa v posledných rokoch venuje veľa pozornosti [19][14][16]. V porovnaní s metódami založenými na lidare [32] alebo stereo kamerách [27], je v značnej nevýhode práve kvôli nedostatku hĺbkových podnetov. Z tohto dôvodu monokulárny 3D detektor nemôže dosiahnuť uspokojivého výkonu ani pri komplexných architektúrach modelov hĺbkového učenia [16].

Súčasné metódy preto najprv odhadujú veľkosť 3D alebo 2D bounding boxu a potom vyvodzujú hĺbku. Odhad 2D bounding boxu je presnejší najmä z dôvodu viac vyvinutej

2D detekcie. Takže autorov nástrojov určených pre 3D detekciu viac znepokojuje chyba odvodenia hĺbky, ktorá je spôsobená chybou odhadu výšky 3D bounding boxu [16].

Malá zmena v odhade veľkosti 3D bounding boxu spôsobí veľkú zmenu pri odhade jeho polohy.

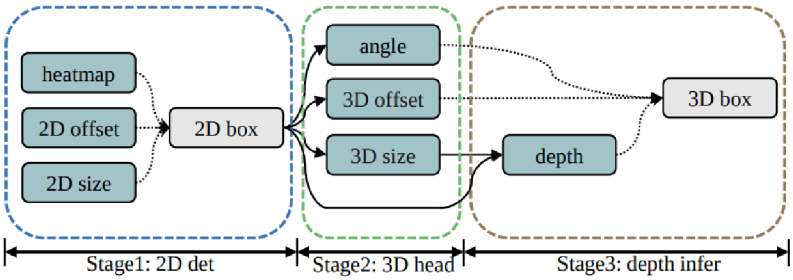


Obr. 2.10: Vizualizované príklady posunu vzdialenosti v dôsledku chyby pri odhade výšky 3D bounding boxu ±0.1m. Je zobrazená scéna z vtáčej perspektívy. Jednotky horizontálnej a vertikálnej osi sú v metroch a vertikálna os predstavuje hĺbku. Zelené obdĺžniky reprezentujú skutočnú polohu vozidla v scéne, modré a červené sú obdĺžniky posunuté v dôsledku +0.1m a -0.1m chyby pri odhade výšky bounding boxu. Obrázok je prevzatý z publikácie [16].

Geometry Uncertainty Projection Network - GUPNet

Táto sekcia čerpá najmä z vedeckej publikácie [16]. GUPNet je nástroj pre odhad polohy objektov záujmu v monokulárnom obraze pomocou estimácie priestorovej hĺbky.

Vstupom je RGB obraz, ktorý je najskôr spracovaný pomocou 2D detekčného prvku, čím sú získané 2D rámčeky. Potom sa vypočítajú niektoré základné 3D informácie o ohraničení (uhol rotácie, rozmery a 3D premietnutý stred pre každý rámček). Následne modul Geometry Uncertainty Projection (GUP) predpovedá distribúciu hĺbky.



Obr. 2.11: Hierarchia podúloh GUPNetu, prevzaté z [16]

Činnosť modelu GUPNet je rozčlenená na niekoľko podúloh. Prvou z nich je 2D detekcia, ktorá je postavená na modeli CenterNet [32], ktorý zahŕňa tri 2D detekčné vetvy na výpočet polohy, veľkosti a koeficientu istoty pre každý potenciálny 2D rámček. Vetva tepelnej mapy vypočítava tepelnú mapu s veľkosťou *(šírka, výška, C)* kde C je počet kategórií detekovaných objektov. Na základe toho, 2D odchýlková vetva vypočíta odchýlku na spresnenie hrubých umiestnení na presný stred rámčeka a vetva odhadu 2D veľkosti predpovedá veľkosť pre každý rámček, ktorý obsahuje vlastnosti na úrovni objektu a nezahŕňajú šum pozadia.

Ďalšou z podúloh GUPNetu je 3D detekcia. Nad vlastnosťami 2D bounding boxov je vytvorených niekoľko vetiev, aby bolo možné predpovedať niektoré základné informácie o 3D ohraničení. 3D odchýlková vetva má za cieľ odhadnúť 3D stredovú projekciu na 2D mapách vlastností. Vetva predikcie uhla predpovedá relatívny uhol natočenia α. A vetva odhadu rozmerov odhaduje parametre 3D rozmerov vrátane výšky, šírky a dĺžky.

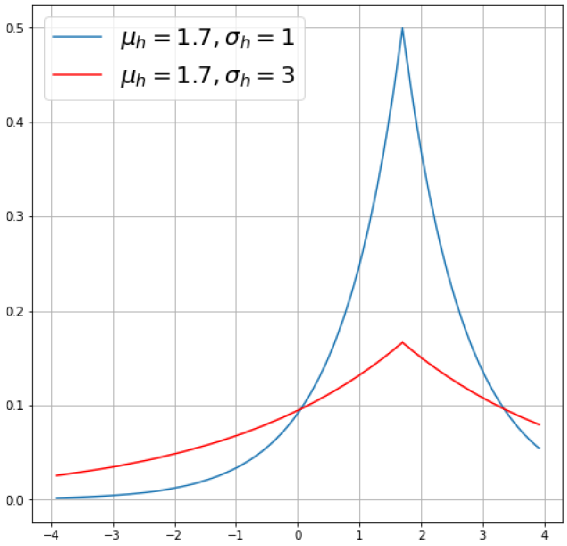
Základné 3D detekčné vetvy poskytujú väčšinu informácií o 3D bounding boxe, okrem hĺbky. Vzhľadom na obtiažnosť priameho odhadu hĺbky autori navrhujú nový model *projekcie neistoty geometrie* (angl. geometry uncertainty projection) [16] . Celkový modul buduje proces projekcie v rámci pravdepodobnosti, a nie v jednotlivých hodnotách, takže model môže vypočítať teoretickú neistotu pre odvodenú hĺbku, ktorá môže naznačovať spoľahlivosť jej odhadu a môže byť tiež užitočná pre hĺbkové učenie. Na dosiahnutie tohto cieľa autori najprv stanovujú predpoklad, že predpoveď 3D výšky pre každý objekt je Laplaceovo rozdelenie $La(\mu_h, \lambda_h)$ [1]. Smerodajná odchýlka sa vzťahuje na parameter $\lambda : \sigma = \sqrt{2}\lambda$

Distribučné parametre μ_h a σ_h sú predpovedané v štádiu odhadu 3D veľkosti. μ_h označuje cieľový výstup regresie a σ_h je neistota odhadu. V dôsledku toho možno stratovú funkciu odhadu 3D výšky definovať ako

$$L_{\text{h3d}} = \frac{\sqrt{2}}{\sigma_h}|\mu_h - h_{3d}^{gt}| + log(\sigma_h)$$

Optimalizácia stratovej funkcie L_{h3d} prebieha v dôsledku znižovania rozdielu medzi vypočítanou výškou μ_h a skutočnou výškou h_{3d}^{gt}. Vzorky s vysokým šumom alebo vzorky zložité na spracovanie disponujú vysokou hodnotou parametra σ_h, čím je indikovná nízka istota príslušného odhadu výšky.

[1] Funkcie hustoty pravdepodobnosti Laplacovho rozdelenia má zápis $La(\mu, \lambda) = \frac{1}{2\lambda}e^{(\frac{|x-\mu|}{\lambda})}$



Obr. 2.12: Porovnanie funkcií rozdelenia pravdepodobnosti riadiacich sa Laplacovým rozdelením. Modrá funkcia vyjadruje vypočítané parametre výšky vozidla $\mu_h = 1.7$ a neistoty $\sigma_h = 1$. červená funkcia na druhej strane vyjadruje situáciu, kedy parameter výšky vozidla je taktiež $\mu_h = 1.7$, no parameter neistoty $\sigma_h = 3$. Je vidieť, že pri modron grafe, je pravdepodobnosť správneho odhadu h_{3d} oveľa vyššia pri hodnote 1.7 ako je pri rovnakej hodnote s použitím vyššieho parametra neistoty v červenom grafe

.

Na základe zisteného rozdelenia h_{3d} možno aproximovať hĺbku, riadiacu sa taktiež Laplacovým rozdelením $La(\mu_p, \lambda_p)$.

Aby bola získaná lepšia predpovedaná hĺbka, na úpravu výsledkov počiatočnej projekcie sa pridá naučená odchýlka (angl. bias) $La(\mu_b, \lambda_b)$. (taktiež riadiaca sa Laplacovým rozdelením). V súlade s tým možno konečné rozdelenie hĺbky ovplyvnené odchýlkou zapísať ako

$$d = La(\mu_p, \sigma_p) + La(\mu_b, \sigma_b)$$

$$\mu_d = \mu_p + \mu_b$$

$$\sigma_d = \sqrt{(\sigma_p)^2 + (\sigma_b)^2}$$

Termínom *koeficient neistoty geometrie* autori odkazujú práve na koeficient σ_d. V ňom je zahrnutá neistota projekcie a taktiež neistota parametra odchýlky (bias). Takto sa aj malá neistota v odhade výšky 3D bounding boxu odrazí na hodnote σ_d. Na optimalizáciu konečného rozdelenia hĺbky je aplikovaná stratová funkcia

$$L_{\text{depth}} = \frac{\sqrt{2}}{\sigma_d}|\mu_d - d^{gt}| + log(\sigma_d)$$

kde d^{gt} je skutočná vzdialenosť bounding boxu. Autori aj pri odhade vzdialenosti predpokladajú Laplacovo rozdelenie.

Od spoľahlivého systému sa očakáva, že spätnou väzbou bude koeficient spoľahlivosti vysoký pre dobrý odhad a nízky pre zlý odhad. Keďže navrhnutá *neistota odhadu projekcie* má schopnosť indikovať neistotu hĺbky, tak je normalizovaná medzi 0-1 pomocou exponenciálnej funkcie

$$p_{\text{depth}} = e^{-\sigma_d}$$

Konečné skóre odhadu polohy 3D objektu môže byť určené ako

$$p_{3d} = p_{depth} \times p_{2d}$$

Modul GUP rieši hlavne efekt akumulácie chýb v štádiu dopredného prechodu neurónovou sieťou. Tento efekt však narušuje aj tréningový postup. Konkrétne na začiatku trénovania nie je predpoveď h_{2d} ani h_{3d} ani zďaleka presná, čo zavádza celkové trénovanie a narušuje spoľahlivosť. Aby sa efekt tohto problému minimalizoval, autori navrhujú hierarchické učenie úloh (angl. Hierarchical Task Learning) na kontrolu váh pre každú úlohu v každej epoche. Tento postup je motivovaný tým, že každá úloha by mala začať trénovanie až potom, čo boli vytrénované všetky predchádzajúce. Rozdelenie na podúlohy je zobrazené v schéme 2.11.

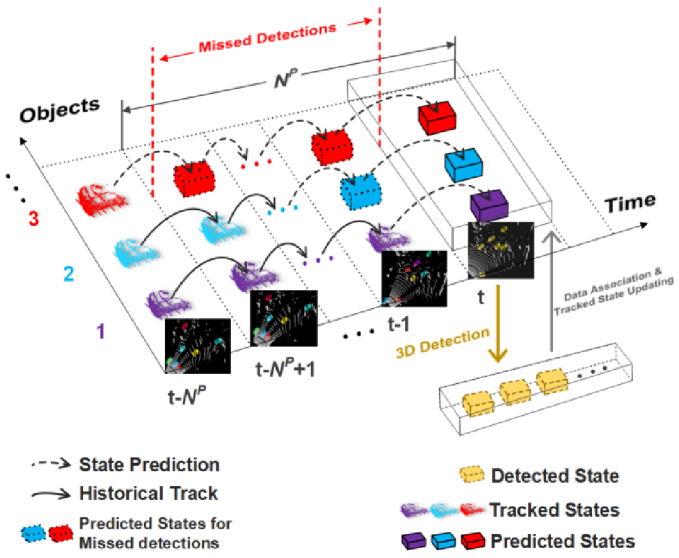
2.3.2 Sledovanie detekcií

Sledovanie detekcií je kritickým prvkom pre extrakciu dynamickej informácie z dopravnej scény. Jeho úlohou je asociácia súčasných detekcií s detekciami získanými v predchádzajúcich snímkach. Táto asociácia prebieha na základe priraďovania identifikátoru každej z detekcií. Problémom viacerých metód je, keď detektor v niektorom čase nezaznamená sledovaný objekt.

3D sledovanie viacerých objektov

Autori nástroja na 3D sledovanie viacerých detekcií v pointcloude [29] navrhujú spôsob založený na schéme asociácie údajov riadenej predikciou, ktorá dokáže efektívne využívať vlastnosti objektov v pointcloude a sledovať dočasne nedetekované objekty. Navrhovaný tracker sa riadi štvorstupňovým rámcom sledovania detekcii.

1. detekcia objektov pomocou hĺbkového učenia a 3D detektora objektov
2. odhad možných súčasných stavov sledovaných objektov pomocou navrhovaného prediktoru založeného na modeli pohybu s konštantným zrýchlením a modeli spoľahlivosti predpovede. Priraďuje každému predpovedanému stavu inú spoľahlivosť predpovede (najmä pre tie s dočasne vynechanými detekciami)
3. priradiť predpovedané stavy k detekovaným stavom na základe spoľahlivosti predpovede
4. aktualizácia spárovaných detekcií uloženie nespárovaných detekcií ako práve prebiehajúcich



Obr. 2.13: Ilustrácia formulovaného problému sledovania objektov, prevzaté z [29]

Proces sledovania viacerých objektov v 3D [29] autori formulujú ako asociáciu detekovaných stavov X_t a predpovedaným stavom $\hat{Z}_t$ odhadnutým z predchádzajúcich sledovaných stavov Z_{t-1}.

Tracker prijíma informácie o detekovaných objektoch, ktoré produkuje detektor založený na hlbokom učení. Výsledky detekcie pozostávajú z ich 3D bounding boxu, a koeficientu spoľahlivosti detekcie. Na odhad reálneho pohybu objektu sa všetky zistené 3D bounding boxy transformujú do globálneho súradnicového systému pomocou údajov GPS/IMU alebo lokalizácie pomocou SLAM. V diskrétnom čase t sú tieto zistené stavy označené

$$X_t = \{X_t^i\}_{i=1}^{N_t^X} \subset \mathbb{R}^{D^X \times 1}$$

kde N_t^X je počet detekcií v danom diskrétnom čase, a D^X je počet rozmerov týchto detekcií. $X_t^i = [x_t^i, y_t^i, z_t^i, w_t^i, h_t^i, l_t^i, \alpha_t^i,]^T$ je detekovaný stav i-teho objektu v diskrétnom čase t. x, y, z sú jeho globálne súradnice, w, h, l sú šírka, výška a dĺžka bounding boxu prislúchajúceho danému objektu, α je uhol rotácie objektu okoly osi y.

Tracker vyžaduje aj množinu predtým plne sledovaných stavov Z_{t-1} ako vstup, ktoré pozostávajú zo sledovaných stavov Z_{t-1}^* a predtým predpovedaných stavov dočasne vynechaných detekcií Z'_{t-1}. Vzťah medzi týmito množinami je $Z_{t-1} = Z_{t-1}^* \cup Z'_{t-1}$. Na vykonanie presnejšieho odhadu pohybu každý plne sledovaný stav zahŕňa aj vektor rýchlosti a zrýchlenia. Množina plne sledovaných stavov v diskrétnom čase $t-1$

$$Z_{t-1} = \{Z_{t-1}^j\}_{j=1}^{N_{t-1}^Z} \subset \mathbb{R}^{D^Z \times 1}$$

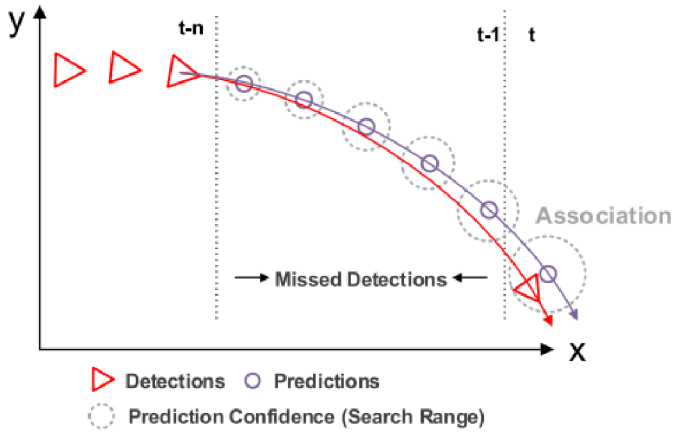
kde N_{t-1}^Z a D^Z je počet detekcií a počet rozmerov predtým plne sledovaných stavov. Plne sledovaný stav j-teho objektu je $Z_{t-1}^j = [x_{t-1}^j, y_{t-1}^j, z_{t-1}^j, v_{t-1}^j, a_{t-1}^j, w_{t-1}^j, h_{t-1}^j, l_{t-1}^j, \alpha_{t-1}^j,]^T$ kde v a a sú vektormi rýchlosti a zrýchlenia.

Na vykonanie asociácie detekcií autori odhadujú súbor možných súčasných stavov v čase t pomocou predchádzajúcich plne sledovaných stavov. Súbor predpovedaných stavov je teda označený ako

$$\hat{Z}_t = \{\hat{Z}_t^j\}_{j=1}^{N_{t-1}^Z} \subset \mathbb{R}^{D^Z \times 1}$$

kde $\hat{Z}_t^j$ predstavuje predpovedaný stav j-tej plne sledovanej detekcie v diskrétnom čase t.

Pre každý z plne sledovaných stavov Z_{t-1}^j je odhadnutý možný súčasný stav $\hat{Z}_t^j$ pomocou *konštantného zrýchlenia.*



Obr. 2.14: Ilustrácia asociácie údajov riadenej spoľahlivosťou predpovedí. Na základe kvality detekcie, tracker priraďuje spoľahlivosť predpovede pre každý predpokladaný stav, aby upravil rozsah vyhľadávania priradenia identity, prevzaté z [29]

Existujú dva prípady pre nezhodné predpovedané stavy Z_t^ϵ. Prvým prípadom je, že objekt prirodzene zmizne v zornom poli. Druhým prípadom je, že detektor objekt nedetekoval z dôvodu, že je dočasne zakryty inými objektmi alebo je príliš ďaleko od kamery. Na rozlíšenie týchto dvoch prípadov je nastavený prah predikcie N_p. Keď predpovedaný stav nemôže byť aktualizovaný detekovaným stavom na viac ako N_p snímok, považuje sa to za prirodzenú stratu detekcie. Takto predpovedaný stav bude vymazaný a nebude sa znova sledovať. V opačnom prípade sa predpovedané stavy zachovajú v dôsledku toho, že objekty môžu byť dočasne vynechané detektorom a môžu sa znova objaviť v budúcich snímkach. Predpovedané stavy zameškaných detekcií v diskrétnom čase t, označené ako Z_t', sa používajú na iteratívne vykonávanie predikcie stavu a asociácie údajov v diskrétnom čase $t+1$.

2.4 Vizuálna lokalizácia

Na lokalizáciu zariadení sa zväčša používa GPS senzor. Získavajúc dáta z niekoľkých satelitov je možné algoritmicky určiť polohu senzoru na presnosť niekoľko desiatok centimetrov. Rotácia okolo špecifickej osi sa zisťuje pomocou zotrvačnej meracej jednotky IMU (angl. inertial measurement unit). Táto jednotka taktiež určuje zmenu polohy pri pohybe, meria rýchlosť či zrýchlenie. Nakoľko sa v našej práci snažíme pracovať len s monokulárnym obrazom, sme nútení vyhnúť sa funkciám spomenutým senzorov, a namiesto toho sa spoľahnúť na monokulárny obraz a informácie, ktoré sú v ňom obsiahnuté.

2.4.1 Lokalizácia a mapovanie - SLAM

Táto sekcia čerpá najmä z vedeckej publikácie [18]. Techniky SLAM vytvárajú mapu neznámeho prostredia, a lokalizujú senzor v prostredí so silným zameraním na fungovanie

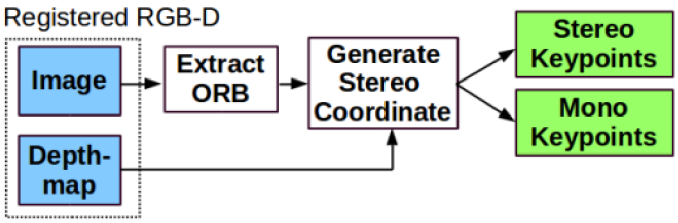
v reálnom čase. Spomedzi rôznych možností senzorov sú kamery lacné a poskytujú bohaté informácie o prostredí, ktoré umožňujú presné rozpoznanie miesta. Preto je v súčasnosti veľký záujem o riešenia vizuálnej lokalizácie SLAM, kde hlavným senzorom je kamera. Rozpoznanie miesta je kľúčovým modulom systému SLAM na uzavretie okruhov (zistenie, kedy sa senzor vráti do zmapovanej oblasti a opravu akumulovanej chyby pri prieskume) a na premiestnenie kamery po zlyhaní sledovania v dôsledku agresívneho pohybu, alebo pri reinicializácii systému.

Lokalizáciu je možné vykonať pomocou monokulárnej kamery, čo je najlacnejšie riešenie. Keďže však hĺbka nie je pozorovateľná len z jednej kamery, mierka mapy a odhadovaná trajektória nie sú známe. V takomto prípade lokalizácia nie je škálovaná ale len relatívna. Monokulárny SLAM trpí škálovaním stupnice a môže zlyhať pri vykonávaní rotácií. Použitím stereo alebo RGB-D kamery sú všetky tieto problémy vyriešené a umožňujú najspoľahlivejšie riešenia.

ORB-SLAM2

Systém pre vizuálnu lokalizáciu kamery, ORB-SLAM2, beží na troch hlavných paralelných vláknach

1. sledovanie polohy na lokalizáciu kamery pri každej snímke nájdením zhôd s mapou a minimalizovaním chyby reprojekcie len za použitia pohybu kamery
2. lokálne mapovanie pre správu správu mapy a jej optimalizáciu
3. uzavretie okruhu pre detekciu veľkých okruhov a korekciou akumulovanej odchýlky



Obr. 2.15: Postup predspracovania RGBD záznamu, ktorý používa ORB-SLAM2, prevzaté z [18]

ORB-SLAM2 predbežne spracúva vstup na extrahovanie záchytných prvkov na miestach kľúčových bodov, ako je znázornené v schéme 2.15. Vstupné snímky sa potom zahodia, a všetky systémové operácie sú založené na týchto záchytných prvkoch, takže systém je nezávislý od toho, či je záznam stereo alebo RGB-D. Zistené monokulárne a stereo kľúčové body sú ďalej klasifikované ako blízke alebo vzdialené. V prípade kamier RGB-D sú kľúčové body extrahované pomocou algoritmu ORB [21]. Každému kľúčovému bodu so súradnicami v obraze (u, v) je priradená hodnoty hĺbky d na rovnakej súradnici.

Kľúčový bod je klasifikovaný ako *blízky* alebo *vzdialený*. Blízke kľúčové body poskytujú informácie o mierke, posune a rotácii. Na druhej strane, vzdialené body poskytujú presné informácie o rotácii, ale menej presné informácie o mierke a posune.

2.5 Predikcia trajektórie

V posledných rokoch sa vynaložilo značné úsilie na predpovedanie trajektórie účastníkov dopravnej scény. Niekoľko z nich bolo zameraných na predpovedanie trajektórie chodcov, buď považovaných za jednotlivcov alebo davy, pričom využívajú aj sociálne správanie a interaktivitu medzi jednotlivcami. Sociálne správanie, ktoré je relevantné pre chodcov, je oveľa menej relevantné pre vozidlá. V tomto kontexte sa pozornosť presúva na pozorovanie minulej trajektórie pohybu jednotlivých vozidiel a pochopenie okolitého prostredia. Táto kapitola čerpá z publikácií [13] [17] [25].

2.5.1 Grafová reprezentácia

Dnešné pokročilé modely priamo kódujú štruktúrované HD mapy [3], predstavujúce čiary jazdných pruhov, ako uzly grafu. Tieto prístupy sú vysoko spoľahlivé, a to dosahujú s použitím menšieho počtu parametrov ako prístupy založené na rasterizácii [13] [17] [25].

Na druhej strane, je prístup, ktorý scénu reprezentuje ako grafy, nemožné použiť v reálnom čase z toho dôvodu, že veľa informácie o prostredí môže byť pre vozidlo v danom okamihu zakrytých inými, nerelevantnými objektami. O to ťažšie je to, keď používame len jeden monokulárny kamerový záznam. Príkladom takejto situácie je, keď vozidlo stoji pred semaforom a vo vedľajších pruhoch stoja vozidlá, cez ktoré nie je možné odhadnúť vzdialenosť krajnice, šírku vozovky, či iné sémantické prvky. Taktiež je daný prístup problematický, keď sa v diaľke nachádza križovatka a nevie odhadnúť smer pruhov. Z tohto dôvodu je grafová reprezentácia scény vhodná len v tom prípade, ak už je vytvorená, a systém ju nemusí vytvárať v reálnom čase, ale stačí, keď sa v nej vie orientovať.

2.5.2 Rastrová reprezentácia

Staršie prístupy kódujú HD mapy prostredia pomocou rastrovaného obrazu z vtáčej perspektívy a konvolučných vrstiev. Aj keď tento prístup využíva silu moderných architektúr konvolučných neurónových sietí, rasterizácia mapy môže byť výpočtovo neefektívna a vedie k čiastočnej strate informácií.

Jej výhodou však je, že môže byť získavaná počas behu v reálnom čase, a nepotrebuje veľký počet senzorov na získavanie informácie o okolí. Pre nás je tento prístup práve kvôli tomuto vhodný, keďže máme k dispozícií len monokulárny záznam z jednej kamery a preto mu v tejto sekcii venujeme značne väčšiu pozornosť.

2.5.3 MANTRA

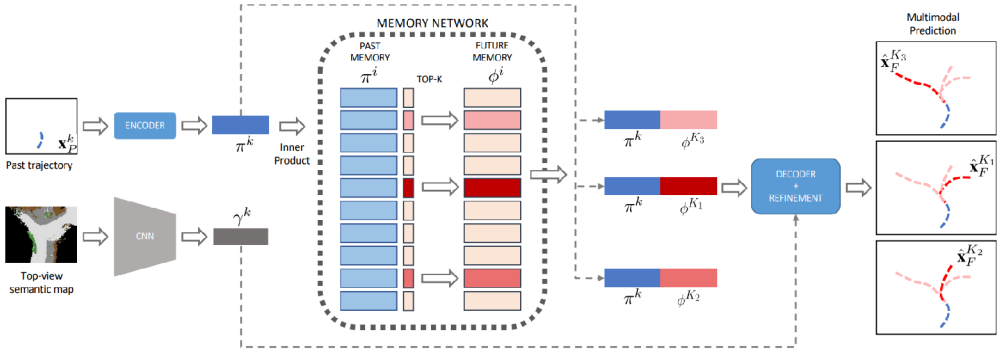
MANTRA (Memory Augmented Neural TRAjectory predictor) [17] implementuje neurónovú sieť MANN (Memory Augmented Neural Network) pre predikciu trajektórie vozidiel.

Na zapamätanie vzoriek sú minulé a budúce trajektórie oddelene uložené v pamäti v zakódovanej forme. Toto umožňuje použiť kódovanie pozorovanej trajektórie ako pamäťový kľúč na čítanie zakódovanej budúcnosti a ich spoločné dekódovanie na generovanie predpovede.

Skutočné súradnice sú získané dekódovaním budúcnosti čítanej z pamäte, na základe pozorovanej minulosti. Týmto spôsobom výstup nie je jednoduchou kópiou predtým videných príkladov, ale je namiesto toho novo vygenerovanou trajektóriou získanou zo skúseností systému, ako aj z doteraz pozorovanej inštancie. Čítaním viacerých budúcnosti z pamäte

možno získať rôzne zmysluplné predpovede. Architektúra tohto modelu využíva encoder-decoder rozšírený o asociatívnu pamäť.

Model sa môže po vytrénovaní postupne zlepšovať pri pozorovaní nových príkladov online. Táto vlastnosť je dôležitá pre priemyselné aplikácie v automobilovom priemysle a v súčasnosti chýba v iných prediktoroch.



Obr. 2.16: Architektúra modelu MANTRA, prevzaté z [17]

Autori formulujú predikciu trajektórie vozidla ako problém odhadu $P(\hat{x}_F|\hat{x}_P, c)$, kde $\hat{x}_F$ je predpovedaná budúca trajektória, $\hat{x}_P$ je pozorovaná trajektória a c je reprezentácia sémantického kontextu. Za trajektóriu vozidiel je považovaná postupnosť dvojrozmerných súradníc. Minulé trajektórie sú dané pozíciami pozorovanými až po referenčný bod identifikovaný ako súčasný. Podobne, budúce trajektórie sú postupnosťou pozícií, v ktorých sa ocitne v ďalších časových krokoch.

Predikcia trajektórie založená na pamäti

Daná je vzorová trajektória $x^i = [x_P^i, x_F^i]$, nech $\pi^i = \Pi(x_P^i)$ a $\phi^i = \Phi(x_F^i)$ sú dve kódovacie funkcie, ktoré mapujú 2D súradnice minulých a budúcich trajektórií do dvoch samostatných reprezentácií. Podobne nech $\Psi(\pi^i, \phi^i)$ je funkcia, ktorá dekóduje dvojicu minulých-budúcich kódovaní do súradníc budúcej trajektórie Je definovaná $M = \pi^i, \phi^i$ ako asociatívna pamäť obsahujúca $|M|$ kódovaní minulostí a budúcností. Keď je pozorovaná nová trajektória x_P^k, jej kódovanie π^k sa používa ako kľúč na získanie zmysluplných vzoriek z pamäte. Mechanizmus adresovania pamäte je implementovaný ako kosínusová podobnosť [31] medzi minulými kódovaniami, čo vytvára hodnotu podobnosti s_i na všetkých miestach pamäte

$$s_i = \frac{\pi^k \pi^i}{\|\pi^k\| \|\pi^i\|} \quad i = 0, ..., |M|$$

Kosínusová podobnosť je metrika, ktorá pomáha určiť, nakoľko podobné sú dátové objekty bez ohľadu na ich veľkosť. V kosínusovej podobnosti sa s dátovými objektmi v množine údajov zaobchádza ako s vektorom.

Podľa tejto podobnosti sú budúce kódovania ϕ^j kombinované s kódovaním pozorovanej minulosti π^k. Nové páry kódovaní sa transformujú do 2D súradníc pomocou dekódovacej funkcie Ψ

$$\hat{x}_F^j = \Psi(\pi^k, \phi^j) \qquad j = 1, ..., K$$

π^k je pevné, zatiaľ čo ϕ^j sa mení v závislosti na vzorke načítanej z pamäte. Budúce kódovania ϕ^j fungujú ako poradcovia, ktorí naznačujú možné výsledky na základe predchádzajúceho pozorovania. Táto stratégia umožňuje modelu pozerať sa dopredu do pravdepodobných budúcnosti, aby predpovedal tú správnu. Keďže viacero ϕ^j je možno použiť nezávisle, môžeme dekódovať viacero predpovedaných trajektórií a získať multimodálnu predpoveď v prípade neistoty (napr. na križovatka).

Funkcie kódovania a dekódovania Π, Φ, Ψ sa trénujú spoločne. Encodery sa učia mapovať minulé a budúce body do zmysluplnej reprezentácie a dekodér sa učí reprodukovať budúcnosť. To umožňuje generovať trajektórie, ktoré sa líšia od trajektórií v pamäti, a nie sú len jednoduchou kópiou už pozorovaných vzoriek.

Memory Augmented Neural Networks

Základnou charakteristikou modelov MANN [24] je použitie kontroléru s externou adresovateľnou pamäťou. Používa sa na ukladanie explicitných informácií a na prístup k selektívne relevantným položkám. Pamäťový kontrolér je trénovaný na optimalizáciu obsahu pamäte dynamicky spravovaných predpovedí. Na rozdiel od RNN sa prechody zo stavu do stavu získavajú operáciami čítania a zápisu a udržiava sa súbor nezávislých stavov. Dôležitým aspektom je, že v pamäťových sieťach nie je počet parametrov viazaný na veľkosť pamäte. To znamená, že zvýšenie počtu pamäťových slotov nezvýši počet parametrov.

Predikčný model MANTRA sa učí ukladať do pamäte len to, čo je nevyhnutne potrebné na vykonávanie presných predpovedí. Následne uložené údaje z pamäti využíva na vytvorenie viacerých výstupov z jedného vstupu, čo vedie k multimodálnej prediktívnej schopnosti celého systému.

Pamäťový kontrolér

Tradičné Memory Augmented Neural Networks [24] sú navrhnuté tak, aby sledovali kolekcie údajov, ktoré sa zvyčajne označujú ako epizódy. Modely sú vybavené funkčnou pamäťou na ukladanie relevantných informácií o epizóde, aby sa pre epizódu vytvoril zmysluplný výstup. Pamäť sa však vymazáva pre každú epizódu a trénovaný je kontrolér, ktorý rozhodne, čo sa má čítať alebo zapisovať.

Tento kontrolér je trénovaný tak, aby vysielal pravdepodobnosť zápisu $P(w)$ zakaždým, keď je vzorka pozorovaná. Účelom jeho trénovania je vytvorenie kompaktnej permanentnej pamäte. Trénovanie takéhoto kontroléru môže byť náročné, pretože $P(w)$ nezávisí len od dôležitosti pozorovanej vzorky, ale aj od aktuálneho stavu pamäte. Preto sa počas trénovania neriadi chybou predikcie ale namiesto toho posiela chybu rekonštrukcie e do kontroléru, ktorý rozhodne, či bola sieť dostatočne blízko k pravde. Na vynútenie tohto správania je definovaná stratová funkcia kontroléru L_c

$$L_c = e.(1 - P(w)) + (1 - e).P(w)$$

kde e nadobúda hodnoty v intervale $\langle 0, 1\rangle$. Keď je hodnota chyby nízka (teda keď e je blízke hodnote 0), tak

$$L_c \simeq P(w)$$

a teda je hodnota pravdepodobnosti zápisu minimalizovaná.

Naopak keď e je blízke hodnote 1, a teda hodnota chyby je vysoká, tak

$$L_c \simeq 1 - P(w)$$

čím kontolér zvýší pravdepodobnosť zápisu do pamäte.

To, čo sa kontrolér učí, je adaptívny prah chyby rekonštrukcie, ktorý umožňuje uložiť do pamäte len to, čo je užitočné pre predpoveď, čím sa obmedzuje redundancia. Ak model vykazuje veľkú chybu predikcie, kontrolér zapíše aktuálnu vzorku s kódovaním budúcej trajektórie do pamäte. Keď sa toto deje, znamená to, že v pamäti chýbajú vzorky na presnú rekonštrukciu budúcnosti. Zapísaním vzorky do pamäte teda model zlepší svoje predikčné schopnosti.

Adaptívna funkcia chybovosti s prahom v závislosti od časového kroku

$$e = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_i(\hat{x}_F, x_F)$$

kde $\mathbb{I}_i(\hat{(x)}_F, x_F)$ je indikačná funkcia rovná 1 ak i-ty bod predikcie $\hat{x}_F$ leží vo vnútri prahovej vzdialenosti od skutočnej pravdy a 0 ak leží mimo nej. Autori používajú inú hodnotu prahovej vzdialenosti pre každý predpovedaný časový okamih. Pre okamich 2s je táto hodnota 1m a lineárne sa znižuje k 0.

2.5.4 Metriky

Spoľahlivosť každého modelu sa meria pomocou štandardných techník, ktoré sa líšia medzi spôsobmi využitia modelu. Najčastejšie používanými metrikami pre kvantifikáciu spoľahlivosti predikčných modelov sú *Average Displacement Error* (*ADE*) a *Final Displacement Error* (*FDE*) [4].

FDE je metrika daná vzdialenosťou medzi konečným bodom trajektórie predpovedanej modelom a konečným bodom skutočnej trajektórie. Väčšinou sa jedná o vzdialenosť v rovine

$$FDE_\tau = |p_p^\tau, p_{gt}^\tau| \quad t \in (0, \tau)$$

kde p_p^τ je predpovedaná pozíciu predikcie v časovom okamihu τ, p_{gt}^τ je skutočná poloha predikcie v okamihu τ, t označuje počet časových úsekov od súčasného okamihu a τ predstavuje konečný časový okamih predikcie.

ADE udáva priemernú vzdialenosť medzi bodmi trajektórie odhadnutej modelom a bodmi skutočnej trajektórie v rovnakých časových okamihoch

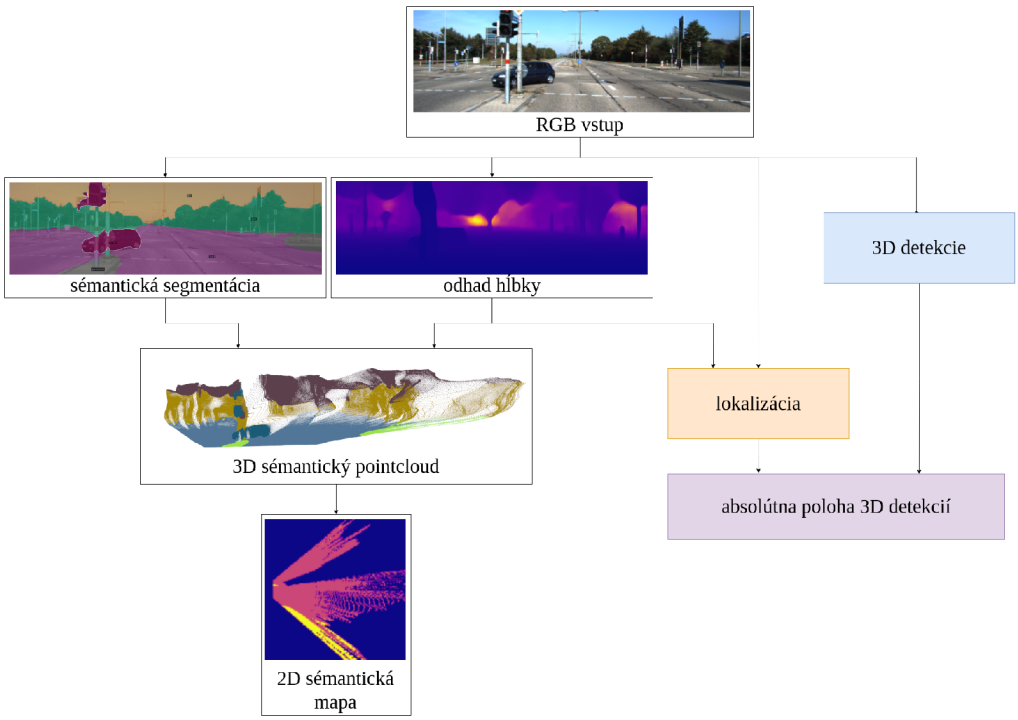
$$ADE_\tau = \frac{1}{\tau}\sum_{t=1}^{\tau} |p_p^t, p_{gt}^t|$$

kde τ predstavuje konečný časový okamih predikcie, t je daný časový okamih, v ktorom sa meria vzdialenosť medzi predpovedanou polohou p_p^t a skutočnou polohu p_{gt}^t.

Kapitola 3

Návrh a implementácia

V tejto kapitole sa venujem spôsobu, akým boli kombinované nástroje pre spracovanie monokulárneho obrazu. Opisujem ako bola vytváraná vhodná reprezentácia prostredia, ktorá následne slúži ako vstup pre predikčný model.



Obr. 3.1: Schéma zreťazeného spracovania monokulárneho kamerového záznamu

Celý systém spracovania obrazu aj predikčný model boli implementované v programovacom jazyku Python vo verzii 3.8. Okrem štandardne využívaných a verejne dostupných modulov pre spracovanie a vizualizáciu dát ako sú numpy[1] či matplotlib[2] som využil aj modul poskytujúci manipuláciu dát pre potreby strojového učenia PyTorch[3].

[1] https://numpy.org/
[2] https://matplotlib.org/
[3] https://pytorch.org/

Nakoľko nie sú k dispozícii dáta z ostatných senzorov, je nutné získavať informáciu o prostredí z jedného monokulárneho zázanmu. Na to používam vytrénované modely strojového učenia založené na neurónových sieťach. Spracovaním obrazu vytváram reprezentáciu prostredia vhodnú pre predikčný model.

V prvom rade ide o získanie horizontálnej 2D sémantickej mapy zobrazujúcej sémantické triedy jednotlivých prvkov scény. Ďalej zisťujem horizontálne pozície detekovaných účastníkov scény. Tie sú vstupom predikčného modelu, ktorý na základe minulej pozície objektov a sémantickej mapy v danom momente určuje ich potenciálne budúce trajektórie.

Operačný systém	Ubuntu 64-bit
RAM	16GB
Grafická karta	NVIDIA GTX1650
Pamäť grafickej karty	4096 MB

Tabuľka 3.1: Pri každom z použitých nástrojov je uvedený čas behu na referenčnom systéme

3.1 2D sémantická mapa

Surový RGB obraz získaný priamo z kamery nie je vhodným vstupom predikčného modelu z niekoľký dôvodov.

Prvým dôvodom je že informácia o prostredí a sledovaných objektoch nie je priamo dostupná a treba použiť techniky, ktoré ju vedia vydolovať.

Druhým dôvodom je, že monokulárny obraz sa líši medzi kamerami použitými na ich zaznamenávanie. Okrem rozmerov jednotlivých snímok sú tieto odlišnosti spôsobené aj vnútornými parametrami kamery, ktorými sú ohnisková vzdialenosť a optické centrum.

V dôsledku odlišností v zázname je preto obtiažne získať predikčný model, ktorý by bol invariantný voči zmene typu používaných kamier.

3.1.1 Odhad vzdialenosti

Nakoľko pracujem s priestorovými informáciami, či už je to poloha vozidiel alebo tvorba horizontálnej sémantickej mapy okolia, je nevyhnutné získať vzdialenosť jednotlivých prvkov. Hĺbková informácia sa dá získavať viacerými spôsobmi.

Odhad hĺbky zo stereo kamery využíva podobný mechanizmus ako náš mozog pri spracovaní obrazu prostredia. Na základe rozdielu v polohách jednotlivých kamier (očí) zisťuje disparitu medzi jednotlivými záznamami. Disparita je nepriamo úmerná vzdialenosti. Preto sa vzdialenosť jednotlivých bodov dá zistiť zo vzdialenosti jedného bodu. Ďalším spôsobom je použitie lidar senzoru, poskytujúceho vzdialenosti objektov v jeho 360-stupňovom okolí.

V tejto práci používam na odhad vzdialenosti z monokulárneho záznamu nástroj monodepth2[4] [7] poskytujúci viacero vytrénovaných modelov pre náš účel. Konkrétne sme zvolili model *mono+stereo_640x192*.

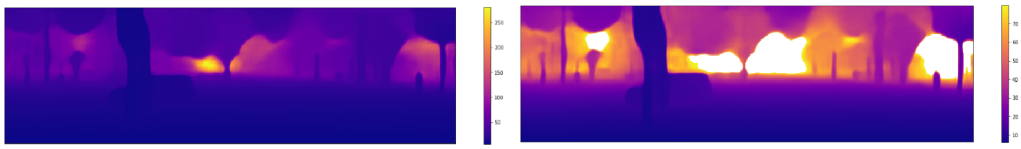
Zvolený model bol vytrénovaný na stereo zázname datasetu KITTI[5]. Výstupom modelu je disparita. Preto skutočnú hĺbku získavam pomocou vzorca

$$vzdialenosť = \frac{1}{disparita} * 5.4$$

[4] https://github.com/nianticlabs/monodepth2
[5] http://www.cvlibs.net/datasets/kitti/

Inverzná hodnota disparity je násobená škálovacím parametrom 5.4, nakoľko je vzdialenosť stereo kamier datasetu KITTI 54cm a model udáva hodnoty vzdialenosti v desiatkach cm. Týmto spôsobom som vypočítal hodnoty vzdialenosti pre jednotlivé pixely v metroch.

Nakoľko sú prvky scény vo vzdialenosti väčšej ako hodnota určitého prahu irelevantné pre potreby predikčného modelu, tak ich ignorujem. V získanej hĺbkovej mape po takomto filtrovaní vzniknú diery. Používam hodnotu prahu 80m.



Obr. 3.2: Hĺbková mapa pred a po filtrovaní hodnôt vzdialeností väčších ako 80m

3.1.2 Sémantická segmentácia

Po určení vzdialenosti jednotlivých pixelov prichádza na rad klasifikácia pixelov do sémantických tried. Na tento účel používam nástroj detectron2[6], poskytujúci výber viacerých modelov pre 2D detekciu objektov a sémantickú segmentáciu obrazu.

V tejto práci používam model *panoptic_fpn_R_101_3x*, poskytujúci 2D detekciu objektov, sémantickú segmentáciu a inštančnú segmentáciu. Pre potrebné účely však využívam len funkciu sémantickej segmentácie, nakoľko na detekciu vozidiel používam techniky opísané v kapitole 3.2.2.

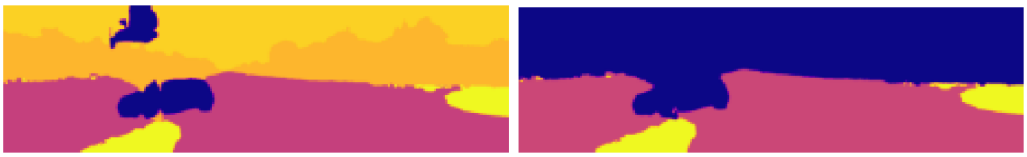
Podobne ako pri odhade vzdialenosti, tak aj pri sémantickej segmentácii výsledok vypočítaný modelom filtrujem. Týmto filtrom sa zbavujem informácie v obraze, ktorá je pre moje účely irelevantná. Triedy, ktoré ostali následne po filtrovaní združujem podľa nasledujúcej tabuľky

vozovka	1
chodník	2
tráva	3
štrk	3
hlina	3
stena	4
plot	4
pozadie	0

Tabuľka 3.2: Tabuľka obsahujúca mapovania relevantných tried prvkov scény, pomocou ktorých sú tieto triedy združované do spoločných kategórií

Všimnime si, že som vynechal viacero kategórií objektov, ktoré môžu poskytovať užitočnú informáciu o pohybe sledovaných vozidiel v scéne. Medzi týmito vynechanými triedami sú napríklad trieda *budov* a trieda *stromov*. Dôvodom, prečo sú dané triedy vynechané,je že pri tvorbe horizontálnej 2D sémantickej mapy 3.1.4 môžu byť dôležité sémantické triedy (napr. vozovka) prekryté menej dôležitými triedami, ktoré boli spomenuté.

[6] https://github.com/facebookresearch/detectron2



Obr. 3.3: Výstup sémantickej segmentáciu pred a po filtrovaní tried

Medzi vyfiltrované triedy je zaradená aj trieda vozidiel. Dôvodom je, že daná trieda neposkytuje informáciu o prostredí, ale o objekte a ako už bolo spomenuté, polohu objektov zisťujem postupom opísaným v kapitole 3.2.2 a príslušnosť pixelov danému objektu považujem za irelevantnú informáciu pre predikčný model.

3.1.3 Sémantický pointcloud

Hĺbkovú a sémantickú mapu je potrebné previesť do 3D množiny bodov reprezentujúcich detekované prvky scény. Tejto množine sa taktiež hovorí *pointcloud*.

Samotnému premietnutiu hĺbkovej mapy do 3D priestoru predchádza klasifikácia prvkov tejto hĺbkovej mapy. Ide o proces, pri ktorom sa kombinuje získaná hĺbková mapa a sémantická mapa. Keďže pri oboch z nich som vykonával filtrovanie, je nad nimi nutné vykonať prienik množín, čím sa zabráni vziku 3D bodov bez priradenej sémantickej triedy a vzniku pixelov sémantických tried, ktorým by nebola priradená priestorová hĺbka.

Obr. 3.4: Výsledok prieniku filtrovanej hĺbkovej a sémantickej mapy. Posledný obrázok zobrazuje, pozície pixelov, ktoré budú použité na vytváranie pointcloudu. Pixely na pozíciách červenej a oranžovej farby budú ignorované. Pixely fialovej farby sú prítomné v sémantickej aj hĺbkovej mape, a preto sú zachované.

Na vytváranie pointcloudu využívam knižnicu open3d[7] umožňujúcu spracovanie a vizualizáciu 3D dát. Na vytvorenie pointcloudu treba poznať parametre kamery, ktorou bola snímka vytvorená. Pre dataset KITTI sú parametre kamery uvedené v nasledujúcej tabuľke

ohnisková vzdialenosť	f_x	721.537
	f_y	721.537
optické centrum	c_x	609.559
	c_y	172.854

Tabuľka 3.3: Vnútorné parametre kamery datasetu KITTI [5]. Všetky údaje sú v pixeloch

Z parametrov kamery je neskôr vytvorená matica

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

[7] http://www.open3d.org/

Táto matica umožňuje premietnuť 3D priestorový bod $P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ do 2D bodu $p = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$ pomocou

$$p = K * P$$

$$\begin{pmatrix} u \\ v \\ vektoru1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Využívam však opačný proces a síce transformáciu hĺbkovej mapy z 2D do 3D priestoru. Toto vyžaduje vytvorenie inverznej matice k matici K.

$$K^{-1} \begin{pmatrix} 1/f_x & 0 & c_x/f_x \\ 0 & 1/f_y & c_y/f_y \\ 0 & 0 & 1 \end{pmatrix}$$

Bodu p v rovine však chýba súradnica z-ovej osi. Preto ako túto súradnicu využívam hodnotu získanú z hĺbkovej mapy na príslušnom pixeli d_p.

$$P = K^{-1} * p$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1/f_x & 0 & c_x/f_x \\ 0 & 1/f_y & c_y/f_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ d_p \end{pmatrix}$$

Nakoniec je každému 3D bodu priradená sémantická trieda podľa predtým vytvoreného prieniku hĺbkovej a sémantickej mapy.

3.1.4 Ortografická projekcia

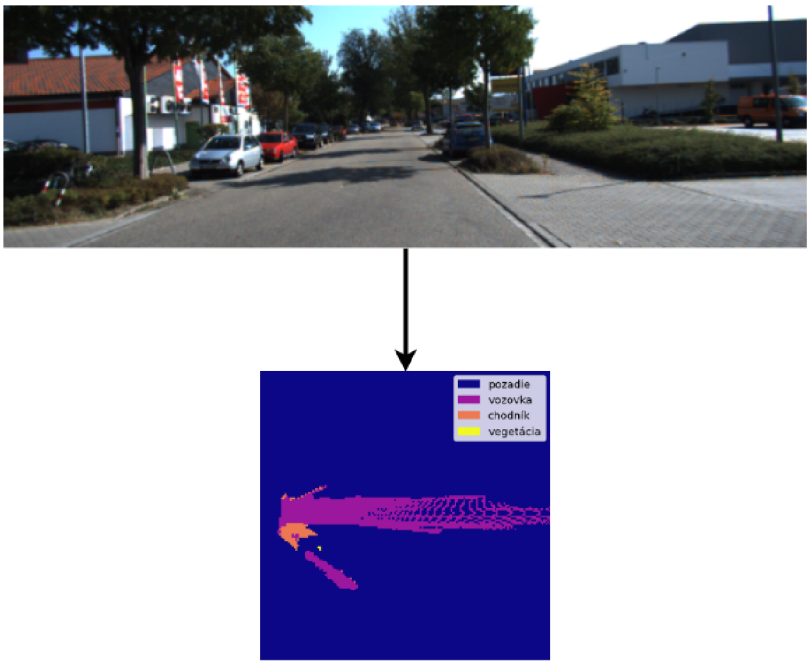
Množina sémanticky ohodnotených priestorových bodov je zložitá na spracovanie predikčným modelom a obsahuje nadmerné množstvo informácie. Preto je ďalším krokom implementácie zjednodušenie reprezentácie prostredia do 2D horizontálnej mapy.

Toto zjednodušenie dosahujem pomocou *ortografickej projekcie*. Ide o bežnú metódu zobrazenia 3D objektov pomocou roviny. V tomto prípade používam projekciu pomocou pohľadu zhora. Všetky body 3D priestoru sa premietnu na rovinu predstavujúcu horizontálnu plochu v zornom poli kamery.

Projekcia prebieha tak, že zanedbávam y-ovú súradnicu jednotlivých bodov. Môže sa však stať, že niektoré body môžu mať rovnaké súradnicové zložky x a z, teda sa líšia len v y-ovej zložke. V takomto prípade môže pri ortografickej projekcii nastať ich prekrytie a vtedy bod, ktorý sa premietol ako prvý, bude prekrytý bodom, premietnutým v poradí za ním. K tomuto pristupujeme tak, že body s najviac významnou sémantickou triedou premietam naposledy.

Najdôležitejšou sémantickou triedou je v mojom prípade vozovka. Preto najprv premietam body ostatných tried, bez ohľadu na ich sémantickú kategóriu. Body predstavujúce vozovku premietam ako posledné.

Nakoľko používam maximálnu vzdialenosť 80m, a jeden pixel 2D mapy predstavuje plochu o veľkosti $0.5m \times 0.5m$, tak dostávam mriežku o veľkosti $160px \times 160px$. Z pohľadu na obrázok je vidieť, že pozícia kamery je v strede ľavej strany obrázku.



3.2 Súradnice detekcií

3.2.1 Lokalizácia kamery

Na lokalizáciu kamery v prostredí používam techniku SLAM (Simultaneous localization and mapping). Konkrétne využívam nástroj ORB-SLAM2[8] [18], ktorý umožňuje lokalizáciu pomocou stereo kamier, RGBD kamier, či monokulárneho záznamu.

Nástroj však pri použití monokulárneho záznamu má problém so škálovaním a lokalizáciou kamery v prostredí s použitím absolútnych jednotiek (metre). Preto využívam možnosť nástroja pracovať s RGBD záznamom.

Snímka RGBD obsahuje okrem štandardných RGB zložiek obrázku aj hĺbkovú zložku. V obraze z kamery táto zložka chýba, preto ju pridávam z predtým odhadnutej hĺbkovej mapy opísanej v sekcii 3.1.1.

Keďže nástroj ORB-SLAM2 vyžaduje, aby bola hĺbková informácia uložená v 16-bitovom kladnom celom čísle, tak získané hodnoty hĺbky v desatinných číslach je potrebné previesť do tohto rozsahu. Stanovená maximálna hodnota hĺbky je 80m, a to znamená, že hodnoty z intervalu $\langle 0.0, 80.0\rangle$ mapujem do intervalu $\langle 0, 2^{16}\rangle$. Toto mapovanie prebieha jednoduchým násobením parametrom S

$$S = \left\lfloor \frac{2^{16}}{80} \right\rfloor = 819$$

Počas celého prechodu snímkami scény vracia ORB-SLAM2 maticu transformácie reprezentujúcu transformáciu kamery na aktuálnej snímke vzhľadom k jej orientácii polohe na prvej snímke. Táto matica je v tvare

[8] https://github.com/raulmur/ORB_SLAM2

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

kde zložky R_{**} udávajú rotáciu kamery vzhľadom k orientácii kamery na prvej snímke, a zložky t_* udávajú transláciu vzhľadom k pozícii na prvej snímke.

3.2.2 Detekcia vozidiel

Na detekciu polohy a orientácie okolitých vozidiel z monokulárneho záznamu je použitý nástroj GUPNet[9] [16]. Tento som pre moje účely zvolil z dôvodu jeho výsledkov na datasete KITTI, kde spomedzi monokulárnych modelov vykazoval lepšie výsledky ako väčšina ostatných.

Vstupom modelu GUPNet je okrem snímky aj matica reprezentujúca vnútorné parametre kamery, pomocou ktorej bola snímka vytvorená.

Výstupom nástroja je zoznam detekcií Q, z ktorých každá detekcia q_i je tvorená postupnosťou $[id_i, p_i, d_i, s_i, k_i, \alpha_i, r_y]$ kde

- id_i je identifikátor triedy detekovaného objektu
- p_i je relatívna pozícia stredu bounding boxu vzhľadom k pozícii kamery
- d_i obsahuje rozmery bounding boxu vo formáte (výška, šírka, dĺžka)
- s_i predstavuje skóre, ktoré udáva s ako istotou bol daný objekt detekovaný
- k_i obsahuje relatívne pozície bodov definujúcich kváder ohraničujúci detekovaný objekt
- uhol α_i určuje, kde sa detekcia nachádza vzhľadom k pohľadu kamery
- uhol r_y udáva rotácia bounding boxu vzhľadom k pohľadu kamery

3.2.3 Sledovanie detekcií

Pre získanie postupnosti detekcií príslušných ku konkrétnemu vozidlu je potrebné detekciám priradiť jednoznačný identifikátor, ktorý v poradí snímok reprezentuje konkrétny sledovaný objekt. Tento identifikátor je zisťovaný pomocou metódy sledovania 3D detekcií použitím nástroja 3D Multi-object Tracker [10] [29]. Teória opisujúca fungovanie nástroja a jeho prístup k problematike sú uvedené v sekcii 2.3.2.

Vstupom modelu je samotná detekcia, ktorú prevádzam do formátu postupnosti údajov [*výška*, *šírka*, *dĺžka*, *x*, *y*, *z*, r_y]. Ďalším vstupom je *skóre*, teda istota s akou bola detekcia získaná. Keďže sa jedná o metódu 3D sledovania objektov, tak je potrebné brať do úvahy aj dynamiku pohybu kamery, a preto model potrebuje aj túto informáciu. Získavam ju z pozičnej matice získanej pomocou monokulárnej lokalizácie opísanej v sekcii 3.2.1. Nakoniec je vstupom aj poradie snímky v rámci sekvencie snímok v zázname.

[9] https://github.com/SuperMHP/GUPNet
[10] https://github.com/hailanyi/3D-Multi-Object-Tracker

Model vo výstupe priradí každému bounding boxu identifikátor, čím som získal postupnosti detekcií pre konkrétne detekované vozidlá v takom v poradí, v akom sa vyskytujú na snímkach.

	640×192	1242×375
odhad vzdialenosti [ms]	24 - 30	-
sémantická segmentácia[ms]	350 - 380	720 - 780
tvorba 2D mapy[ms]	8 - 10	30 - 40
lokalizácia[ms]	20 - 30	35 - 45
3D detekcia[ms]	25 - 32	80 - 100
sledovanie detekcií[ms]	1 - 2	1 - 2
spolu	**500 - 600**	**900 - 1000**

Tabuľka 3.4: Tabuľka s časmi výpočtu jednotlivých častí zreťazeného spracovania monokulárneho obrazu v závislosti od veľkosti vstupného obrázku[12].

3.3 Predikcia

Ako predikčný prvok používam model MANTRA [17]. Bol vybraný z toho dôvodu, že jeho implementácia je verejne dostupná[13]. Ďalším dôvodom voľby tohto modelu je, že pracuje s vhodnou reprezentáciou prostredia, a teda sémantickou 2D mapou, ktorá je výsledkom spracovania obrazu.

Keďže 3D detektor poskytuje relatívnu pozíciu vozidiel vzhľadom k polohe kamery, treba minulé a budúce detekcie transformovať do takého pohľadu, akoby boli pozorované v danom momente.

Nakoľko počas každého časového okamihu získavame 2D mapu, zoznam detekcií a transformačnú maticu kamery, tak je možné zistiť, ako by sa zmenili súradnice polohy vozidla, detekovaného v čase $t - \tau$, keby boli pozorované z pozície kamery v čase t.

Detekcie v aktuálnom čase t označené d_t^{rel} je možné transformovať do absolútnych súradníc d_t^{abs} pomocou transformačnej matice M_t získanej pomocou lokalizácie

$$d_t^{abs} = M_t \times d_t^{rel}$$

Tieto detekcie v absolútnych súradniciach neskôr prevádzam do relatívnych súradníc vzhľadom k pohľadu kamery v aktuálnom čase t

$$d_t^{rel} = M_t^{-1} \times d_{t-\tau}^{abs}$$

kde τ predstavuje ľubovoľný počet časových okamihov do minulosti.

Všetky minulé detekcie transformujem nasledovane

$$d_t^{rel} = M_t^{-1} \times M_{t-\tau} \times d_{t-\tau}^{rel} \qquad \tau = \{\beta, ..., 0\}$$

kde β je maximálny počet krokov do minulosti, ktoré vnímame ako minulú trajektóriu.

Budúce polohy detekcií transformujem podobne

[12] pri projekcii hĺbkovej mapy do 3D pointcloudu je pri použití zmenšeného obrázku nutné použiť upravené parametre kamery. Táto úprava prebieha tak, že pri obrázkoch o rozmeroch (640, 192) sú všetky parametre kamery, uvedené v tabuľke3.3, vydelené 2

[13] https://github.com/Marchetz/MANTRA-CVPR20

$$d_t^{rel} = M_t^{-1} \times M_{t+\lambda} \times d_{t+\lambda}^{rel} \qquad \lambda = \{1, ..., \alpha\}$$

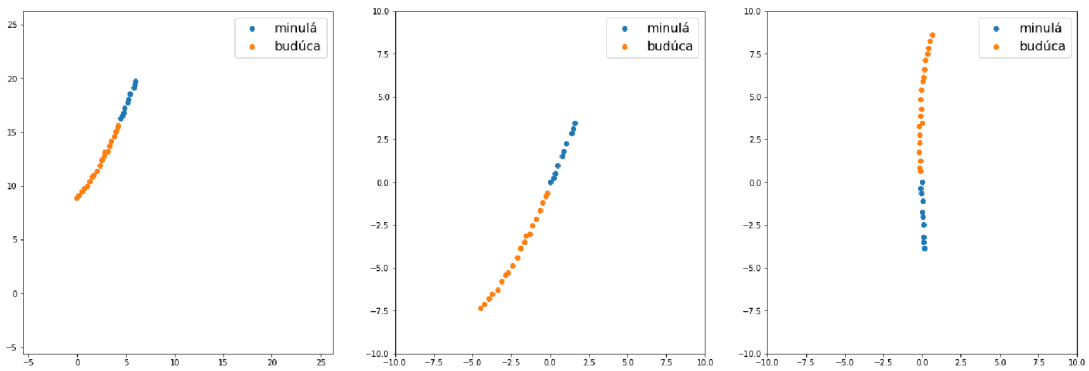
kde λ definuje konkrétny počet časových okamihov do budúcnosti, α je maximálny počet krokov do budúcnosti, ktoré vnímame ako budúcu trajektóriu.

Predikčný model MANTRA pozostáva z troch samostatných častí, ktoré sú trénované individuálne, a každá nasledujúca časť je trénovaná až po tom, čo boli vytrénované všekty časti modelu, ktoré jej predchádzajú

1. Auto-encoder
2. Memory controller
3. IRM

Pre dosiahnutie *translačnej invariancie*, teda toho, že model bude predpovedať trajektóriu nezávisle od umiestnenia sledovaných trajektórií, je potrebné ich normalizovať. Celú sekvenciu detekcií presúvam tak, aby sa detekcia v aktuálnom čase nachádzala v strede súradnicovej sústavy.

Rotačná invariancia je, na druhej strane, vlastnosť modelu, kedy model predpovedá trajektóriu nezávisle na jej orientácii. Je dosiahnutá tak, že trajektóriu rotujem v takom smere, aby budúca trajektória pokračovala v smery osi y.



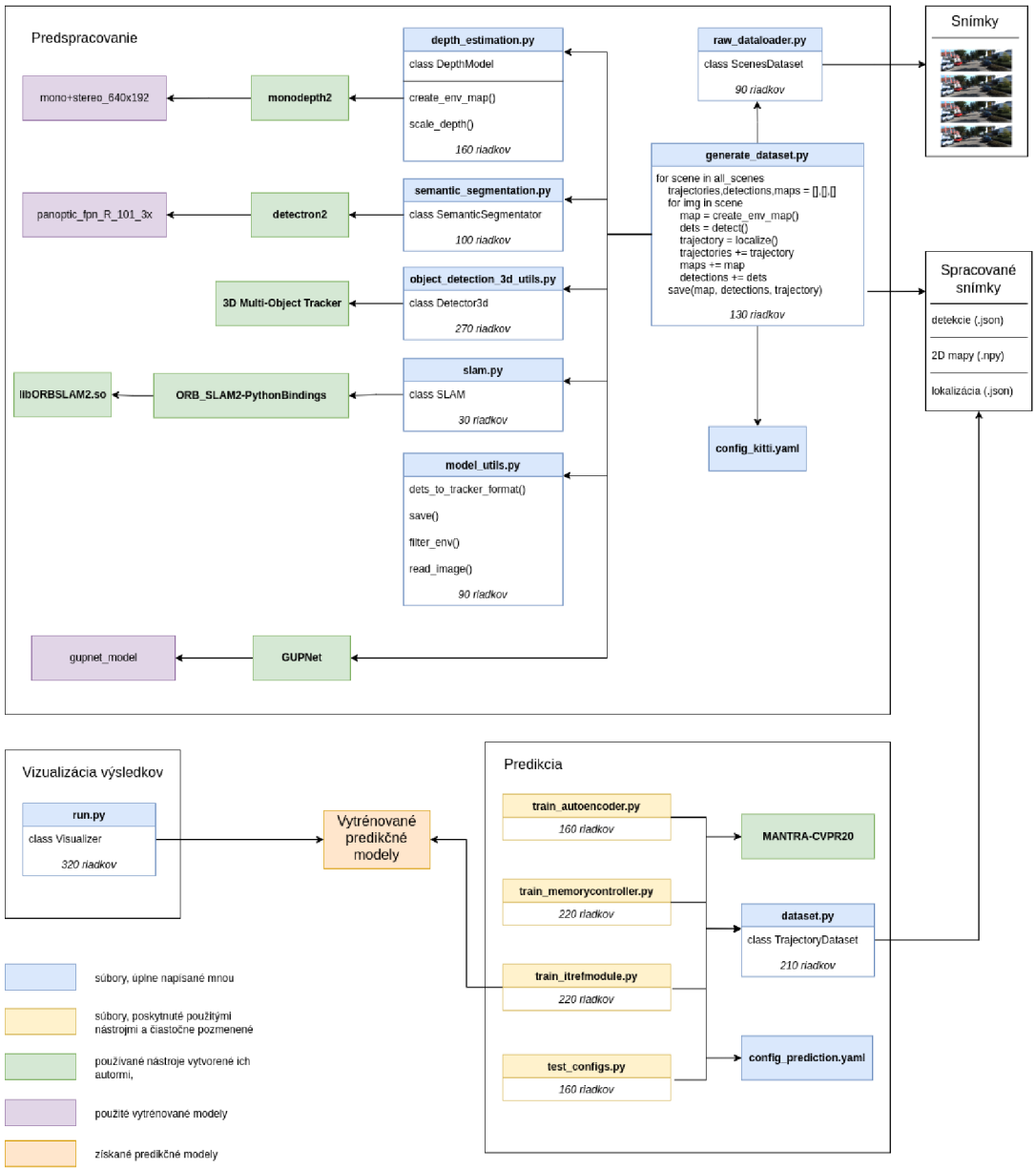
Obr. 3.5: Na prvom grafe je zobrazená trajektória tak, ako je extrahovaná z obrazu, na druhom obrázku je znázornená translačná invariancia, keďže súčasná detekcia sa nachádza v strede súradnicovej sústavy. Na poslednom obrázku je zobrazená rotačná invariancia, nakoľko budúca trajektória pokračuje v smere osi y.

Keďže sa v predspracovaných dátach nachádzajú aj sekvencie detekcií stojacich vozidiel, tak je potrebné ich vyfiltrovať z dôvodu ich nepriaznivého vplyvu na trénovanie modelu. Toto filtrovanie prebieha tak, že pre všetky trojsekundové sekvencie sa zisťuje či je vozidlo v pohybe pomocou vzorcov

$$\text{stojace} : Var[X_i - \mu_i] \leq s$$

$$\text{pohybujúce} : Var[X_i - \mu_i] > s$$

kde Var je funkcia pre zistenie rozptylu zo vstupu, X_i je vstupná sekvencia, μ_i je priemerná hodnota vstupnej sekvencie a s je hranica, ktorá rozhoduje o tom či vozidlo stojí alebo sa pohybuje. V našej práci je použitá hranica $s = 1.0$. Ako vstup do predikčného modelu sú následne použité len vozidlá, ktoré sa pohybujú.



Obr. 3.6: Diagram zobrazujúci štruktúru systému pre spracovanie obrazu, trénovanie a testovanie predikčných modelov, a konečnú vizualizáciu fungovania systému za behu

Kapitola 4

Experimenty

Pre experimenty využívam monokulárny záznam datasetu KITTI [5]. Pomocou modulu na predspracovanie dát, som spracoval jednotlivé snímky vybraných sekvencií. Tým som získal niekoľko 3s dlhých postupností detekcií s príslušnými 2D sémantickými mapami. Nakoľko je kamerový záznam dátovej sady vzorkovaný na frekvencii 10 Hz, tak spomenuté trojsekundové sekvencie tvoria 30 2D súradníc, ktoré následne v experimentoch využívam ako sledované a predpovedané trajektórie.

Dáta som predspracoval dvomi spôsobmi

1. S použitím pôvodného rozlíšenia vstupného obrazu 1242×375
2. S použitím rozlíšenia zmenšeného na približne polovicu, a teda 640×192

	trénovacie sekvencie	testovacie sekvecie
1242×375	5800	3019
640×192	3236	1338

Tabuľka 4.1: Tabuľka zobrazujúca počty extrahovaných trénovacích a testovacích sekvencií o dĺžke 3s, získaných pri spracovávaní záznamu s uvedeným rozlíšením

Po predspracovaní dát som vytrénoval niekoľko modelov. Pri trénovaní každého z nich, boli použité parametre z nasledujúcej tabuľky

epochy	600
learning rate	0.0001
optimalizátor	Adam

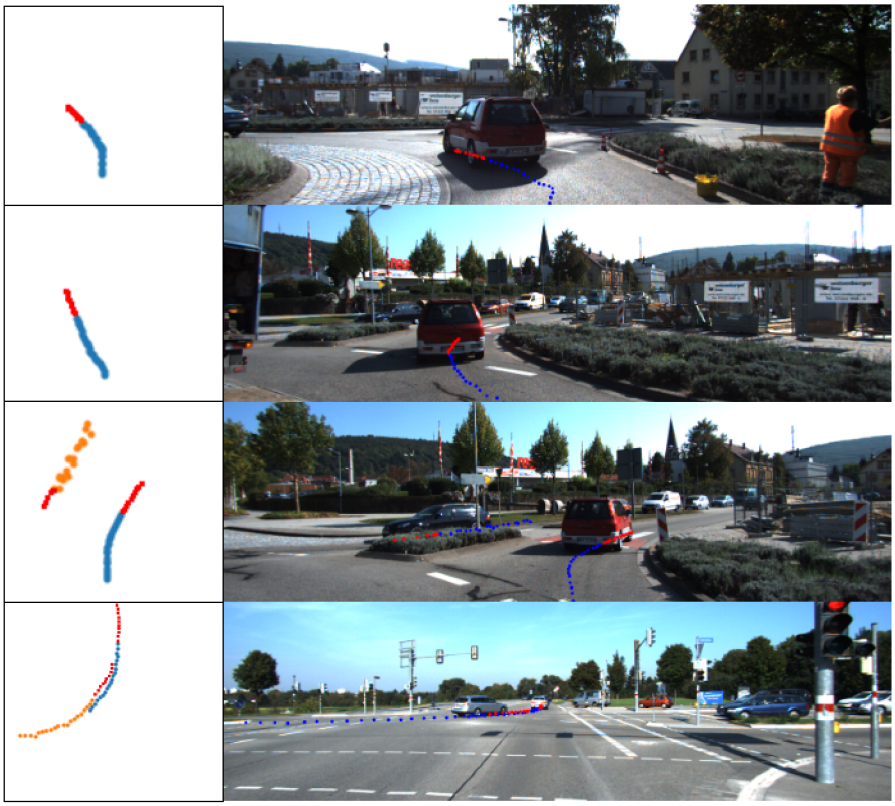
Tabuľka 4.2: Tabuľka zobrazujúca trénovacie parametre modelov, prevzaté z [17]

	dĺžka trajektórie		ADE		FDE	
	sledovaná	predpovedaná	1s	2s	1s	2s
translačná + rotačná invariancia	1s	2s	**3.228**	6.15	**5.548**	12.334
	2s	1s	3.662	-	6.74	-
translačná invariancia	1s	2s	3.949	7.645	7.07	15.023
	2s	1s	4.295	-	7.938	-
bez kontextu scény	1s	2s	3.229	**6.12**	5.507	**12.256**
	2s	1s	3.482	-	7.938	-

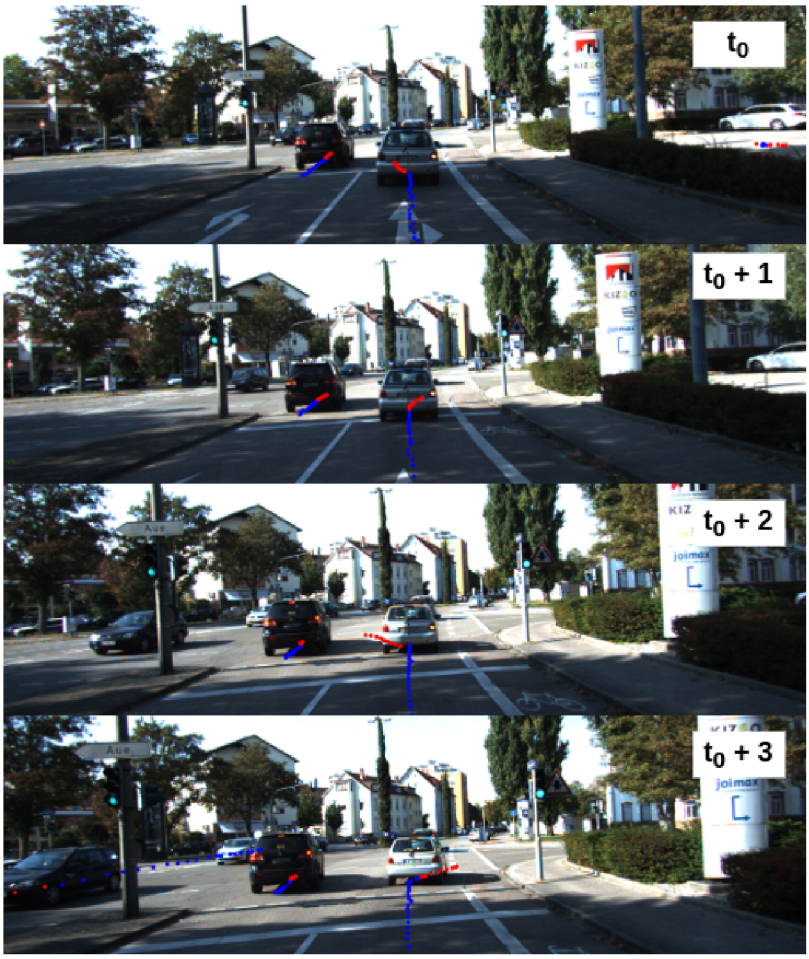
Tabuľka 4.3: Tabuľka zobrazujúca výsledky jednotlivých modelov pre dataset KITTI s pôvodnými rozmermi obrázkov (1242, 375, 3)

	dĺžka trajektórie		ADE		FDE	
	sledovaná	predpovedaná	1s	2s	1s	2s
translačná + rotačná invariancia	1s	2s	3.773	3.862	3.457	4.488
	2s	1s	3.524	-	5.079	-
translačná invariancia	1s	2s	3.045	4.075	3.943	6.404
	2s	1s	3.29	-	3.728	-
bez kontextu scény	1s	2s	**0.843**	**1.492**	**1.333**	**2.962**
	2s	1s	0.852	-	1.47	-

Tabuľka 4.4: Tabuľka zobrazujúca výsledky jednotlivých modelov pre dataset KITTI s upravenými rozmermi obrázkov (640, 192, 3)



Obr. 4.1: Vizualizácia validných predikcií



Obr. 4.2: Vizualizácia nekonzistentných výsledkov predikcie. Pre snímky idúce v poradí bezprostredne za sebou je vidieť nekonzistentné a meniace sa predikcie trajektórie vozidla odbočujúceho vpravo.

4.1 Zhodnotenie výsledkov

Jedným zo zistení získaných pri experimentoch je, že kontext scény nemá veľký vplyv na predpovedanie trajektórie. V niektorých prípadoch sa dokonca prejavil ako negatívny prvok zhoršujúci presnosť predikčného modelu.

Pri porovnaní výsledkov tabuliek je vidieť, že spracovanie zmenšeného obrazu viedlo, vo väčšine prípadov, k výrazne vyššej presnosti, čo ide proti intuícii.

Ďalším zistením je, že nie v každom prípade sa dĺžka sledovanej trajektórie (1s alebo 2s) prejavila ako prvok zdokonaľujúci generovanie budúcej trajektórie. Dá sa to sledovať v stĺpcoch s dĺžkou predikcie 1s.

Jednou z vlastností, ktoré sa ukázali ako vhodné pri trénovaní a následne testovaní, je rotácia vstupnej sekvencie pre dosiahnutie rotačnej invariancie. Oproti modelom, pri ktorých bola použitá len translačná invariancia, vykazujú modely trénované aj na rotovaných sekvenciách, vyššiu presnosť.

Ukázalo sa, že reprezentácia prostredia získaná z jedného kamerového záznamu sa prejavila ako nedostatočná, nakoľko autorom predikčného modelu MANTRA sa podarilo dosiahnuť konzistentných výsledkov na oveľa presnejšej úrovni, avšak s použitím viacerých senzorov a pripravenej mapy celého prostredia. Okrem toho sú schopní predpovedať trajektóriu počas dlhšieho časového horizontu.

Oproti komplexnejším modelom, ako je Trajectron++ [23] alebo PGP [4], ktoré sa dopúšťajú maximálnej chyby v desiatkach centimetrov, sa môj systém prejavil ako veľmi nespoľahlivý. Dopúšťa sa chyby predikcie, ktorá je pri behu systému v priemysle neakceptovateľná.

	ADE				FDE			
	1s	2s	3s	4s	1s	2s	3s	4s
translačná + rotačná invariancia	**0.17**	**0.36**	**0.61**	**0.94**	**0.30**	**0.75**	**1.43**	**2.48**
translačná invariancia	0.25	0.51	0.88	1.38	0.45	1.09	2.10	3.58
bez kontextu scény	0.18	0.39	0.67	1.04	0.33	0.85	1.59	2.65

Tabuľka 4.5: Výsledky modelu MANTRA, prevzaté z publikácie [17]

4.2 Nedostatky systému a možné výlepšenia

Jedným z nedostatkov celého systému je spôsob vyhodnocovania. Nakoľko metriky, ktoré sú v práci použité na určenie presnosti systému zachytávajú len chybu predikčného modelu, tak nie je možné z nich určiť chybu vzniknutú v dôsledku akumulácie pri spracovaní obrazu jednotlivými prvkami. Toto by sa dalo vyriešiť pomocou porovnania predikcií vytvorených systémom a predikcií určených priamo v datasete. Keďže dataset KITTI nedisponuje sekvenciami s anotovanými 3D polohami sledovaných objektov, je nutné v rámci testovania použiť inú dátovú sadu, prípadne vyhodnotiť, ako sa jednotlivé prvky reťazového spracovania obrazu správajú pri použití iných dát.

Ďalším nedostatkom je nemožnosť získať z monokulárneho záznamu sekvencie detekcií dlhšie ako 3s, keďže detekované vozidlá sa v zornom poli kamery vyskytujú krátku dobu. Iné prístupy [23] [3] tento problém riešia zväčšením zorného poľa pomocou viacerých senzorov, no keďže sa v našej práci zameriavame len na spracovanie monokulárneho obrazu, tak tento prístup nie je vhodný.

Systém taktiež vyžaduje zlepšenia v oblasti času výpočtu predspracovania vstupného obrazu, ako je uvedené v tabuľke 3.4. Veľký priestor na zdokonalenie je v procese tvorby 2D mapy, konkrétne pri sémantickej segmentácii.

Náročnosť celého systému na výpočtové zdroje je možné odľahčiť použitím relatívne lacného GPS/IMU pre lokalizáciu namiesto SLAM. Týmto sa poruší prvotný zámer, kedy sa mal použiť len monokulárny obraz, no použitím finančných zdrojov na aplikovanie GPS/IMU jednotky, sa ušetria financie pre výpočtovú jednotku, ktorá bude môcť byť menej výkonná, nakoľko nebude musieť lokalizovať kameru pomocou zložitých maticových operácií, ale dostane túto informáciu priamo zo senzora.

Kapitola 5

Záver

Zadaním bakalárskej práce bolo zoznámenie sa so spôsobmi rozpoznávania typických účastníkov cestnej premávky a s dostupnými implementáciami pre konfiguráciu a trénovanie neurónových sietí. Na základe získaných poznatkov som mal navrhnúť a implementovať systém, ktorý dokáže predikovať trajektórie detekovaných objektov aj v zložitejších dopravných situáciách. Tento cieľ som splnil prostredníctvom štúdia danej problematiky, implementácie systému a porovnania výsledkov so systémami založenými na zložitejšej reprezentácii prostredia.

Prvým krokom práce bolo preštudovanie technológií umožňujúcich spracovanie obrazu a následné dolovanie informácií, ďalej mechanizmov strojového učenia a neurónových sietí. V ďalšej fáze bol s využitím známych nástrojov navrhnutý postup reťazového spracovania monokulárneho obrazu. Podľa tohto postupu boli vydolované informácie kombinované pre dosiahnutie vhodnej reprezentácie prostredia a určenie polohy kamery a detekovaných objektov záujmu. Získaná množina informácií o scéne bola následne zhromaždená vo vhodnej forme a rozdelená na trénovaciu a testovaciu podmnožinu. Na základe trénovacích dát bolo vytrénovaných niekoľko modelov, ktoré boli neskôr porovnané. Tieto výsledky boli zverejnené a okomentované.

Predikčný systém, založený na spracovaní monokulárneho obrazu, dokáže predpovedať trajektórie vozidiel v časovom horizonte 2s na základe pozorovania trvajúceho 1s a naopak. Oproti komplexnejším systémom pracujúcim so zložitejšou reprezentáciou okolia však neposkytuje presnosť a ani čas spracovania na akceptovateľnej úrovni tak, aby bol nasadený v priemysle.

Vďaka tejto práci som získal nové poznatky v oblasti spracovania obrazu a princípmi strojového učenia. Práca mi umožnila získať skúsenosti s plánovaním a rozvrhnutím postupu na dlhodobom projekte, návrhom komplexnejších systémov, a štrukturovaním a tvorbou odborného textu.

Výslednú prácu by som rád obohatil o schopnosť spracovania monokulárneho obrazu získaného z kamier s rôznymi vnútornými parametrami, čím by sa rozšírila dátová sada. Taktiež by som celý systém zoptimalizoval, aby spracovanie obrazu trvalo menšiu dobu, keďže sémantická segmentácia sa prejavila ako prvok najviac ovplyvňujúci čas spracovania. Systém by som taktiež rozvinul o možnosť voľby predikčného modelu a tým o možnosť porovnania ich správania na základe rovnakej reprezentácie prostredia.

Literatúra

[1] Agarap, A. F. Deep learning using rectified linear units (relu). *ArXiv preprint arXiv:1803.08375*. 2018.

[2] Albawi, S., Mohammed, T. A. a Al Zawi, S. Understanding of a convolutional neural network. In: Ieee. *2017 international conference on engineering and technology (ICET)*. 2017, s. 1–6.

[3] Deo, N., Wolff, E. a Beijbom, O. Multimodal trajectory prediction conditioned on lane-graph traversals. In: PMLR. *Conference on Robot Learning*. 2022, s. 203–212.

[4] Deo, N., Wolff, E. a Beijbom, O. Multimodal trajectory prediction conditioned on lane-graph traversals. In: PMLR. *Conference on Robot Learning*. 2022, s. 203–212.

[5] Geiger, A., Lenz, P., Stiller, C. a Urtasun, R. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*. 2013.

[6] Géron, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Ö'Reilly Media, Inc.", 2019.

[7] Godard, C., Mac Aodha, O., Firman, M. a Brostow, G. J. Digging into Self-Supervised Monocular Depth Prediction. October 2019.

[8] Goodfellow, I., Bengio, Y. a Courville, A. *Deep learning*. MIT press, 2016.

[9] He, K., Zhang, X., Ren, S. a Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, s. 770–778.

[10] John Lambert. *Stereo and Disparity* [https://johnwlambert.github.io/stereo/]. 2022.

[11] Kirillov, A., Girshick, R., He, K. a Dollár, P. Panoptic feature pyramid networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, s. 6399–6408.

[12] Křištof, J. *Komunikační agent pro informace o Brně*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23414/.

[13] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H. et al. Desire: Distant future prediction in dynamic scenes with interacting agents. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, s. 336–345.

[14] Li, P., Zhao, H., Liu, P. a Cao, F. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In: Springer. *European Conference on Computer Vision*. 2020, s. 644–660.

[15] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. et al. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, s. 2117–2125.

[16] Lu, Y., Ma, X., Yang, L., Zhang, T., Liu, Y. et al. Geometry Uncertainty Projection Network for Monocular 3D Object Detection. *ArXiv preprint arXiv:2107.13774*. 2021.

[17] Marchetti, F., Becattini, F., Seidenari, L. a Del Bimbo, A. MANTRA: Memory Augmented Networks for Multiple Trajectory Prediction. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.

[18] Mur Artal, R. a Tardós, J. D. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*. 2017, zv. 33, č. 5, s. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[19] Park, D., Ambrus, R., Guizilini, V., Li, J. a Gaidon, A. Is Pseudo-Lidar needed for Monocular 3D Object detection? In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, s. 3142–3152.

[20] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K. a Koltun, V. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2020.

[21] Rublee, E., Rabaud, V., Konolige, K. a Bradski, G. ORB: An efficient alternative to SIFT or SURF. In: Ieee. *2011 International conference on computer vision*. 2011, s. 2564–2571.

[22] Rumelhart, D. E., Durbin, R., Golden, R. a Chauvin, Y. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*. Lawrence Erlbaum Hillsdale, NJ, USA. 1995, s. 1–34.

[23] Salzmann, T., Ivanovic, B., Chakravarty, P. a Pavone, M. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Springer. *European Conference on Computer Vision*. 2020, s. 683–700.

[24] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D. a Lillicrap, T. Meta-learning with memory-augmented neural networks. In: PMLR. *International conference on machine learning*. 2016, s. 1842–1850.

[25] Srikanth, S., Ansari, J. A., Ram, R. K., Sharma, S., Murthy, J. K. et al. Infer: Intermediate representations for future prediction. In: IEEE. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, s. 942–949.

[26] Teledyne FLIR LLC. *How is depth determined from a disparity image?* [https://www.flir.eu/support-center/iis/machine-vision/knowledge-base/how-is-depth-determined-from-a-disparity-image//]. 2022.

[27] Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M. et al. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, s. 8445–8453.

[28] Wang, Z., Bovik, A. C., Sheikh, H. R. a Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*. IEEE. 2004, zv. 13, č. 4, s. 600–612.

[29] Wu, H., Han, W., Wen, C., Li, X. a Wang, C. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*. IEEE. 2021.

[30] Wu, Y. a He, K. Group normalization. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, s. 3–19.

[31] Xia, P., Zhang, L. a Li, F. Learning similarity with cosine similarity ensemble. *Information Sciences*. Elsevier. 2015, zv. 307, s. 39–52.

[32] Zhou, X., Wang, D. a Krähenbühl, P. Objects as points. *ArXiv preprint arXiv:1904.07850*. 2019.

Príloha A

Obsah pamäťového média

Táto kapitola obsahuje obsah priloženého pamäťového média (SD karty). Nasledujúci obsah uvádza základný prehľad obsahu:

- **doc/technical_report**: zdrojové súbory technickej správy bakalárskej práce
- **doc/poster/poster.png**: plagát prezentujúci prácu vo formáte **png**
- **doc/poster/poster.pdf**: plagát prezentujúci prácu vo formáte **pdf**
- **src**: zložka so zdrojovými súbormi práce
- **src/README.md**: súbor README obsahujúci základný popis projektu, užívateľskú príručku, návod na inštaláciu a používanie
- **src/requirements.txt**: súbor obsahujúci zoznam závislostí
- **src/config**: zložka s konfiguračnými súbormi pre predspracovanie obrazu
- **src/sequences.txt**: súbor obsahujúci zoznam s názvami sekvencií datasetu KITTI, ktoré boli v práci použité

Príloha B

Plagát