



**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

**VYUŽITÍ SYNTETICKÝCH DAT PRO ZLEPŠENÍ DETEKCE  
CYKLISTŮ A CHODCŮ V AUTONOMNÍM ŘÍZENÍ**  
USING SYNTHETIC DATA FOR IMPROVING DETECTION OF CYCLISTS AND PEDESTRIANS IN  
AUTONOMOUS DRIVING

**BAKALÁŘSKÁ PRÁCE**  
BACHELOR'S THESIS

**AUTOR PRÁCE**  
AUTHOR

**VEDOUCÍ PRÁCE**  
SUPERVISOR

**ZUZANA KOPČILOVÁ**

**doc. RNDr. PAVEL SMRŽ, Ph.D.**

**BRNO 2023**

## Zadání bakalářské práce



145572

Ústav: Ústav počítačové grafiky a multimédií (UPGM)  
Studentka: **Kopčilová Zuzana**  
Program: Informační technologie  
Specializace: Informační technologie  
Název: **Využití syntetických dat pro zlepšení detekce cyklistů a chodců v autonomním řízení**  
Kategorie: Umělá inteligence  
Akademický rok: 2022/23

### Zadání:

1. Seznamte se se způsoby rozpoznávání objektů a s dostupnými implementacemi pro snadnou konfiguraci a trénování neuronových sítí.
2. Na základě získaných poznatků navrhněte a implementujte řešení pro tvorbu syntetické datové sady, využitelné pro trénování systémů autonomního řízení.
3. Vytvořte dostatečné množství vzorků pro zlepšení detekce cyklistů a chodců.
4. Vyhodnoťte vliv využití nové datové sady na výsledky detekce.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

### Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu 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, prof. Dr. Ing.

Datum zadání: 1.11.2022

Termín pro odevzdání: 10.5.2023

Datum schválení: 24.4.2023

## **Abstrakt**

Tato práce se zabývá tvorbou umělé datové sady pro autonomní řízení a možností jejího využití pro zlepšení přesnosti detekce zranitelných účastníků provozu. Existující práce v této oblasti buď nezveřejňují svůj postup tvorby datové sady, nebo nejsou vhodné pro účely 3D detekce objektů. V rámci této práce jsou představeny konkrétní kroky pro vytvoření umělé datové sady. Získané vzorky jsou následně validovány pomocí jejich vizualizace a při experimentech využity pro učení modelu detekce objektů VoxelNet.

## **Abstract**

This thesis deals with creating a synthetic dataset for autonomous driving and the possibility of using it to improve the results of vulnerable traffic participants' detection. Existing works in this area either do not disclose the dataset creation process or are unsuitable for 3D object detection. Specific steps to create a synthetic dataset are proposed in this work, and the obtained samples are validated by visualization. In the experiments, the samples are then used to train the object detection model VoxelNet.

## **Klíčová slova**

autonomní řízení, 3D detekce objektů, lidar, simulátor CARLA, datová sada KITTI, syntetická datová sada, VoxelNet, detekce chodců, detekce cyklistů

## **Keywords**

autonomous driving, 3D object detection, lidar, CARLA simulator, KITTI dataset, synthetic dataset, VoxelNet, pedestrian detection, cyclist detection

## **Citace**

KOPČILOVÁ, Zuzana. *Využití syntetických dat pro zlepšení detekce cyklistů a chodců v autonomním řízení*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

# **Využití syntetických dat pro zlepšení detekce cyklistů a chodců v autonomním řízení**

## **Prohlášení**

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....  
Zuzana Kopčilová  
8. května 2023

## **Poděkování**

Ráda bych poděkovala panu doc. RNDr. Pavlu Smržovi, Ph.D. za odborné vedení mé bakalářské práce, zejména za jeho trpělivost, a v neposlední řadě také za příležitost zabývat se takto zajímavým tématem.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Rozbor problematiky</b>	<b>4</b>
2.1	Systémy autonomního řízení . . . . .	4
2.2	Detekce objektů . . . . .	8
2.3	Simulátor CARLA . . . . .	17
<b>3</b>	<b>Návrh a implementace</b>	<b>19</b>
3.1	Návrh řešení . . . . .	19
3.2	Nastavení prostředí simulátoru . . . . .	20
3.3	Tvorba datové sady . . . . .	21
3.4	Výsledná datová sada . . . . .	27
<b>4</b>	<b>Experimenty</b>	<b>29</b>
4.1	Experimenty využívající model VoxelNet . . . . .	29
4.2	Validace vytvořené datové sady . . . . .	37
4.3	Zhodnocení výsledků . . . . .	40
<b>5</b>	<b>Závěr</b>	<b>41</b>
	<b>Literatura</b>	<b>42</b>
<b>A</b>	<b>Obsah přiloženého paměťového média</b>	<b>45</b>
<b>B</b>	<b>Plakát</b>	<b>46</b>

# Kapitola 1

## Úvod

Autonomní řízení lze nepopiratelně označit za téma velmi aktuální — v médiích se pravidelně objevuje nejen v souvislosti s nejnovějšími technologiemi, ale také jako součást problematiky elektromobility a ekologie osobní přepravy, a dokonce bývá diskutováno i jako jedno z možných řešení v rámci snahy o zvýšení bezpečnosti silničního provozu.

S touto pomyslnou lupou nad odvětvím autonomního řízení přichází i mnoho úhlů pohledu a názorů na něj, jeho přínos či dokonce jeho moralitu. Jistá opatrnost, která je nám lidem vlastní, je pravděpodobně na místě — přece jen bylo v posledních letech možno zaznamenat hned několik významných případů, kdy došlo k selhání některého z částečně autonomních systémů osazovaných do osobních vozidel, jejichž cílem je zajištění bezpečnosti posádky vozidla i jeho okolí.

Pro uvedení do kontextu této práce je zde tedy vhodné krátké zamýšlení nad tím, jaké mohou být dopady selhání samořízeného vozidla a pro koho by byly nejzávažnější. V provozu se kromě motorových vozidel všech tvarů a velikostí pohybuje i mnoho výrazně zranitelnějších účastníků, které můžeme souhrnně označit pojmy chodci a cyklisté. A právě na jejich bezpečí při interakci s autonomními vozidly se tato práce zaměřuje.

Čtenáři budou možná povědomé mediálně pokryté případy, kdy tehdejší držitel titulu nejbezpečnějšího vozidla světa nerozpoznal cyklistu ve své bezprostřední blízkosti, nebo ne-hoda autonomního vozidla společnosti Uber, jejíž obětí se stala chodkyně. V návaznosti na uvedený typ nehod byla v rámci této práce zvolena konkrétně problematika zaznamenání a rozpoznání zranitelných účastníků provozu, jelikož mnoho existujících systémů, které se zabývají monitorováním okolí vozidla a detekcí objektů v jeho okolí, vykazuje výrazně větší spolehlivost při rozpoznávání vozidla než při rozpoznávání chodce či cyklisty.

Za účelem odhalení příčin a následného vyrovnaní tohoto nepoměru byla prozkoumána datová sada, na níž byly některé existující systémy trénovány, a následně byla zvolena strategie jejího doplnění dalšími vzorky, v nichž se chodci a cyklisté s větší frekvencí vyskytují. K tomu byl využit simulátor autonomního řízení CARLA, který bude v následujících kapitolách podrobněji představen.

V kapitole rozboru problematiky je nejprve popsána standardní architektura systémů autonomního řízení a rozpad jejich funkcionality na jednotlivé podúlohy. Následuje rozbor úlohy detekce objektů, výsledků dosahovaných aktuálními modely, a v neposlední řadě také populární datové sady KITTI. Závěrem kapitoly je podrobně představen simulátor autonomního řízení CARLA.

Kapitola návrhu a implementace pak již představuje konkrétní řešení, které bylo v rámci této práce zvoleno ke zvýšení spolehlivosti detekce zranitelných účastníků provozu za využití

volně dostupného existujícího modelu pro detekci objektů, a popisuje klíčové části jeho technického provedení.

Experimenty provedené k ověření, zda bylo dosaženo žádoucích výsledků, jsou popsány v další kapitole. Na vyhodnocení jejich průběhu navazuje závěr práce, v němž je shrnuto vše, čeho bylo dosaženo a čeho by bylo v případném navazujícím výzkumu dosáhnout možno.

## Kapitola 2

# Rozbor problematiky

Cílem této kapitoly je seznámit čtenáře se systémy autonomního řízení, rozdělit jejich funkcionality na jednotlivé podúlohy a krátce shrnout metodami učení těchto systémů, jejich testování a vyhodnocování získaných výsledků.

V první podkapitole je představeno rozdělení systémů autonomního řízení na tři funkční bloky, které jsou následně podrobně popsány. Následuje podkapitola, zabývající se úlohou detekce objektů, která je primárním zaměřením této práce. Krátce představeny jsou výsledky dosahované existujícími modely, často užívaná datová sada KITTI a také konkrétní model VoxelNet, který byl zvolen pro experimenty provedené v rámci ověření validity řešení, jež tato práce prezentuje. Závěrečná podkapitola se zaměřuje na simulátor autonomního řízení CARLA a možnosti jeho využití.

### 2.1 Systémy autonomního řízení

Automatizace řízení vozidla byla společností SAE International rozdělena do šesti stupňů, kdy nultý představuje vozidlo bez automatizace, a pátý zcela samostatný systém, kterému je pouze zadávána cílová destinace. Stupeň autonomie systému se zároveň dle této stupnice odvíjí i od způsobu monitorování jeho okolí — v nižších stupních (0-2) odpovídá za monitorování okolí řidič vozidla, ve vyšších stupních se již jedná o zodpovědnost vozidla samotného. Dosud nasazované systémy podpory řízení dosahují nejvýše třetího stupně, který vyžaduje fyzickou přítomnost a alespoň částečnou pozornost řidiče. Do nižších stupňů této škály pak spadají různé podpůrné systémy řízení, například samostatné parkovací asistenty [20].

Z historického hlediska lze za prvního předchůdce moderních asistenčních systémů označit již systém ABS, zabraňující zablokování kola při brzdění, který byl v sériově vyráběném osobním vozidle poprvé použit před více než 40 lety [16]. Od té doby prošlo automobilové odvětví rapidním vývojem, a spolu s ním i nasazované bezpečnostní systémy, přičemž aktuální trendy již dokonce směřují k plně autonomnímu řízení.

I přesto, že jeho využití stojí v cestě prozatím nejen omezení, související s technickým provedením, ale také aktuální legislativa, projevili ve vývoji vlastního systému pro autonomní vozidla v posledních letech iniciativu i mnozí velikáni ze světa informačních technologií.

Specifika systémů jednotlivých výrobců se přirozeně mohou v mnohem lišit, jejich obecnou funkcionality však lze rozdělit do následujících tří modulů [2, 20]:

- vnímání kontextu situace — monitorování vnitřního stavu vozidla a stavu jeho okolí a jejich interpretace;

- rozhodování o chování vozidla na základě získaných poznatků;
- ovládání vozidla.

### 2.1.1 Vnímání kontextu situace

Jak již bylo předestřeno, vnímání dopravní situace autonomním vozidlem lze rozdělit na monitorování vnitřních stavů i okolní scény (tedy sběr dat) a interpretaci získaných dat. Obě fáze budou v této podkapitole blíže popsány.

#### Užívané senzory

Pro získání komplexní představy o okolí vozidla jsou využívány různé typy senzorů (součástek, které měří určitou fyzikální veličinu v dané oblasti a získaná data předávají k dalšímu zpracování).

Jedním z nejznámějších typů senzoru je kamera. Častým způsobem jejího využití ve vozidlech jsou zadní parkovací kamery, které kromě usnadnění parkovacích manévrů zároveň i výrazně zvyšují bezpečnost osob v okolí vozidla (z tohoto důvodu je například v USA její osazení povinné pro všechna vozidla prodávaná po květnu 2018). Výstupem moderních kamer je rozlišený vysoce kvalitní obraz, jehož reálnou kvalitu však mohou znatelně snižovat zhoršené viditelnostní podmínky, například déšť nebo mlha.

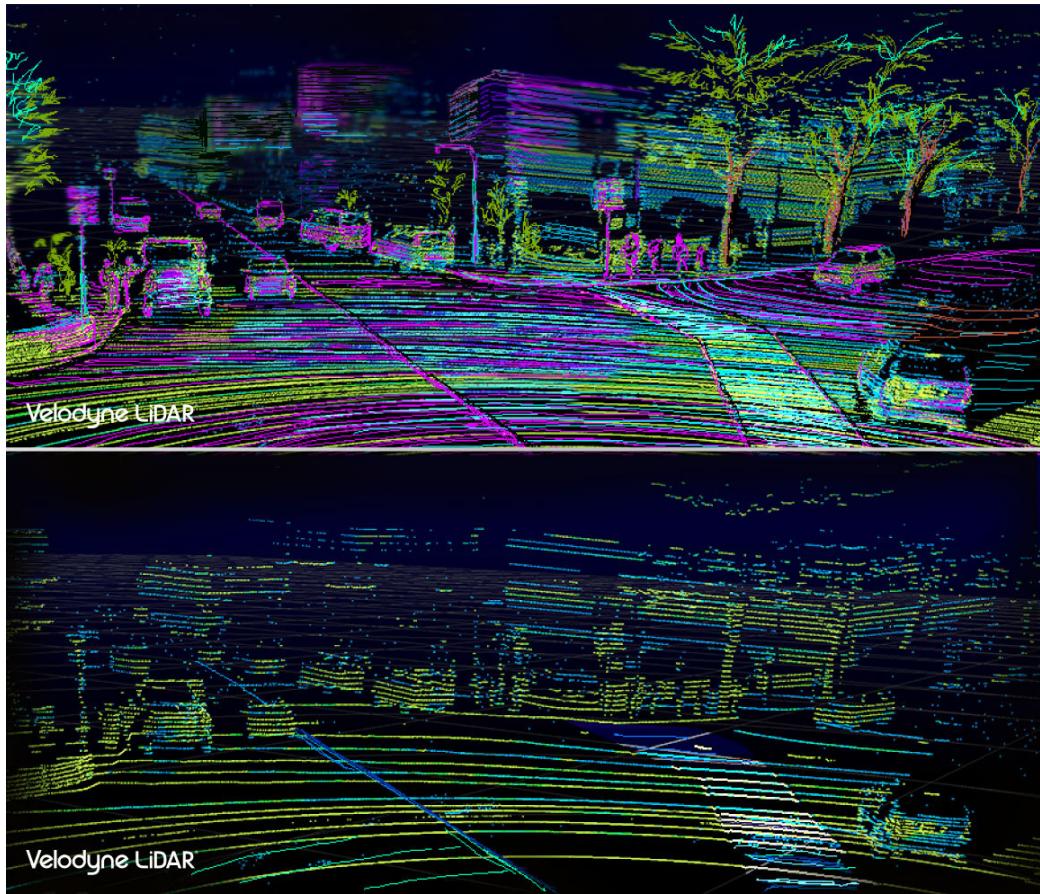
V pokročilých asistenčních systémech se kamery využívají nejen pro zlepšení výhledu, ale také k rozpoznávání dopravních značek, detekci ostatních účastníků provozu, sledování jízdy v pruhu a podobně. Kamerami pokrytý zorný úhel se pak může blížit i  $360^\circ$ . Pro účely autonomního řízení jsou využívány nejen běžné barevné kamery (RGB kamery), ale také kamery hloubkové (RGB-D kamery), které umožňují výpočet vzdálenosti objektů od vozidla, na němž je senzor osazen [4].

K detekci objektů ve vzdálenostech řádově až stovek metrů mohou sloužit také radary — zařízení určená k vyhledávání okolních objektů a určování vzdálenosti od nich za pomocí elektromagnetického vlnění. Vzhledem k jejich vhodnosti pro užití ve vyšších rychlostech patří mezi systémy, využívající radarové senzory, například hlídání slepého úhlu, asistent změny jízdního pruhu nebo adaptivní tempomat [27].

Velmi užitečné jsou také senzory typu lidar (Light Detection and Ranging). Funkčně se jedná o druh radaru, který využívá jeden či více laserových paprsků, které se odráží od okolních objektů zpět do zdrojového zařízení, přičemž je opět vypočítána vzdálenost objektu od zdroje paprsku. Výstupem lidaru je takzvané mračno bodů, které představuje prostorový model okolí vozidla (pro mračno bodů je běžně užíván také anglický pojem *point cloud*).

Jednotlivé lidary se mohou lišit například v počtu vysílaných paprsků (častými hodnotami jsou 32 nebo 64, případně i 128 paprsků), ale také v pokrytém úhlu — horizontálně mohou zabírat celých  $360^\circ$ , ale existují i lidary pokrývající poloviční úhel, 120 stupňů apod. Počet vysílaných paprsků ovlivňuje hustotu výsledného mračna bodů — na obrázku 2.1 je ilustrován rozdíl výstupu lidaru využívajícího 128 paprsků oproti polovičnímu počtu paprsků [25].

Historicky byly lidary využity například ke zkoumání povrchu vesmírných těles [22], avšak se snižující se cenou a velikostí senzoru je lze využívat už i v komerčních aplikacích. Například některé z produktů společnosti Apple Inc. využívají lidar ke skenování 3D objektů, přesnějšímu odhadu vzdálenosti, a s ním souvisejícímu zaostření fotoaparátu, a dalším účelům, včetně her s rozšířenou realitou.



Obrázek 2.1: Porovnání výstupu lidarů využívajících 128 (nahoře) a 64 (dole) paprsků.  
Zdroj: webové stránky společnosti Velodyne Lidar [23]

Mezi největší nevýhody lidaru patří v porovnání s jinými typy senzorů jeho vysoká cena. Důvodem je jeho komplexnější mechanické řešení a cena výrobních materiálů. Zároveň však není příliš vhodný pro využití v intenzivně zvlněném terénu. Lidar sám o sobě navíc nedokáže rozpoznávat barvy, což je potřebné například pro rozpoznávání dopravních situací, kde figurují semafory, proto je často využíván v kombinaci s kamerami.

Naopak velmi důležitou pozitivní vlastností lidaru je jeho schopnost spolehlivě rozeznávat objekty i za tmy (viz obrázek 2.2), což je i jedním z důvodů, proč se tato práce zaměřuje na detekci objektů za pomoci lidarových dat.

Kromě snímání okolních objektů je pro systémy autonomního řízení důležitá také lokalizace vozidla za účelem plánování jeho trasy, k čemuž je využíván senzor typu GPS (globální polohový systém) [20].



Obrázek 2.2: Porovnání výstupu kamery a lidaru za nočního osvětlení, převzato a upraveno z webových stránek společnosti Velodyne Lidar [23]

### Interpretace dat

V rámci fáze interpretace dat prezentuje SAE definice plně autonomního vozidla požadavky detekce, rozpoznání a klasifikace objektů a událostí [20] (v prostých termínech se jedná o nalezení objektu v okolí vozidla a jeho zařazení do některé ze sémantických tříd, určujících jeho význam v kontextu dopravní situace. Úlohou detekce objektů v okolí vozidla se podrobněji zabývá podkapitola 2.2).

Z pohledu implementace systému autonomního řízení je do fáze interpretace dat získaných ze senzorů možno zařadit také úlohy zpracování vstupního obrazu, sledování detekovaných objektů (efektivně se jedná o zkoumání změn jejich stavu v průběhu určité časové sekvence, získanými daty může být například trajektorie jejich pohybu), sémantické segmentace a odhadu vzdáleností [12].

V rámci zpracování sesbíraných dat je obvyklá také tzv. fúze senzorů, kdy jsou skombinována data z více různých vstupů. Důvodem pro ni může být šum vyskytující se v datech jednotlivých senzorů — je-li určitý jev nalezen v datech více senzorů, je už jen málo pravděpodobné, že by se jednalo o šum či odchylku.

Výstupem modulu pro vnímání stavu vozidla a jeho okolí je takzvaný model světa — data získaná ze senzorů se propojí s již uloženými daty (například znalostmi o daném prostředí ve formátu map) a společně utvoří kompletní reprezentaci prostředí, v němž se autonomní vozidlo nachází. Model světa je následně využíván pro rozhodování o dalších krocích chování vozidla, na něž se zaměřuje následující podkapitola [20].

#### 2.1.2 Rozhodování

Rozhodování o budoucím chování vozidla je nejkomplexnějším funkčním blokem systému autonomního řízení — systém zde předpovídá chování subjektů v okolí a na základě získaných predikcí v kombinaci s modelem světa navrhuje možné akce vozidla.

Významnou úlohou tohoto modulu je plánování trajektorie vozidla — na základě modelu světa jsou nalezeny oblasti, v nichž se nenachází žádné překážky, a následně je navržena sada možných trajektorií vozidla. Z těchto trajektorií je k provedení vybrána ta nejlepší.

Dále pod funkcionalitu rozhodování a plánování spadá také správa dostupné energie (optimální využití baterie, regenerativní brzdění ap.), diagnóza komponentů systému a zpracování chybových stavů.

Kromě plánovaných akcí jsou modulu pro ovládání vozidla předávány také reakce na kritické podněty – například přiblížuje-li se vozidlo velmi rychle k překážce či naopak nějaký objekt k vozidlu, je aktivován systém pro zamezení srážce a vyslány pokyny k nouzovému

brzdění či vyhnutí se překážce (tato funkctionalita se pod názvem AEB<sup>1</sup> vyskytuje i v aktuálně vyráběných vozidlech v podobě asistenčního systému).

Části systému, implementující tyto reaktivní funkce, naslouchají jen několika málo senzorům a jsou aktivní po celou dobu provozu (jsou-li povoleny) [2].

### 2.1.3 Ovládání vozidla

Mezi komponenty modulu pro ovládání vozidla patří ty, které přímo zodpovídají za manipulaci s platformou vozidla. Na nejvyšší úrovni lze ovládání vozidla rozdělit na dvě hlavní úlohy, a to stabilizaci platformy a provádění trajektorie.

Cílem stabilizace vozidla je permanentní zachování jeho ovladatelnosti. Tuto úlohu tedy plní podsystémy jako kontrola trakce, systémy proti zablokování kol, nebo pokročilejší elektronické stabilizační systémy [2].

Průjezd trajektorie naplánované modulem pro rozhodování o budoucím chování vozidla je prováděn za pomoci komponent pro ovládání jeho rychlosti (nastavována primárně akcemi akcelerace a brzdění) a směru (dán úhlem natočení volantu). Je-li pro korektní provedení nutná signalizace (například světelná, oznamující změnu směru jízdy), je také vyvolána tímto funkčním blokem [20].

## 2.2 Detekce objektů

Detekce objektů je jednou z disciplín spadajících pod oblast počítačového vidění a zpracování obrazu. Koncept lze stručně popsat následovně – úkolem systému je vyhledat v datovém vstupu objekty, které spadají do tříd s různým sémantickým významem. Příkladem takové třídy mohou být třeba chodci, nebo vozidla. V praxi se lze s detekcí objektů setkat nejen v odvětví autonomního řízení, ale také například v agrokultuře u monitorování kusů dobytku, různých samostatných domácích robotech, bezpečnostních kamerových systémech nebo při sběru statistických dat – třeba denního počtu zákazníků supermarketů, nebo počtu vozidel projíždějících určitými oblastmi.



Obrázek 2.3: Ukázka výstupu detekce objektů na vzorku datové sady KITTI [9], vygenerováno pomocí [29]

<sup>1</sup>AEB — Autonomous Emergency Breaking – Autonomní nouzové brzdění

Historicky stavěla detekce objektů na několika různých principech. Prvotní systémy pracovaly na bázi posuvných oken, která se postupně přesouvala po vstupním obrazu a hledala vyskytující se shodu. Některé systémy detekovaly objekty jako soubor jednotlivých prvků – vozidlo například jako spojení kol, karoserie a oken [28].

Tato práce se zaměřuje na přístup využívající neuronové sítě, který se do popředí odvětví dostal zhruba v letech 2012-2014, kdy probíhala takřka revoluce hlubokého učení. Principy tohoto přístupu jsou podrobněji popsány v následující podkapitole, na kterou navazují podkapitoly pojednávající o metodách vyhodnocování výsledků dosažených detekcí objektů, představení datové sady KITTI [9], užívané pro učení a testování modelů jednotlivých úloh autonomního řízení, a v neposlední řadě také konkrétní model VoxelNet [26].

### 2.2.1 Dělení metod detekce objektů

V kontextu přístupu využívajícího neuronové sítě existují dvě hlavní skupiny metod – dvoufázové a jednofázové. Dvoufázové metody objekty nejprve lokalizují a až v dalším kroku navrhují jejich klasifikaci do příslušné třídy. Metody jednofázové provádí obě tyto činnosti v téže fázi zpracování. Obecně lze označit dvoufázové metody za spolehlivější a jednofázové za rychlejší [24].

Dále lze metody dělit i na základě toho, zda scénu, v níž objekty rozpoznávají, reprezentují jako rovinu či trojrozměrný prostor. 2D detekce objektů se zabývá objekty v rovině, tedy takovými, které jsou reprezentovány pouze dvěma rozměry a pro účely detekce jsou approximovány jako obdélník, jehož strany se dotýkají krajních bodů objektu. 3D detekce oproti tomu bere v potaz i hloubku daného objektu, nepotýká se tedy do takové míry s problematikou perspektivy. Celý objekt je approximován jako těsný kvádr, jeho stěny se opět dotýkají krajních bodů tělesa. Pro approximační obdélníky a kvádry je běžně užíván společný odborný anglický název *bounding box*.

Metody prostorové detekce objektů lze rozdělit podle formátu dat, z nichž vycházejí. Třemi nejčastějšími skupinami jsou:

- metody využívající RGB obraz
- metody využívající lidarové mračno bodů
- metody kombinující RGB obraz i mračno bodů

Mezi největší výzvy prostorové detekce objektů za využití pouze kamerových vstupních dat patří rekonstrukce hloubky. Častým řešením je převod obrazu na syntetické mračno bodů vytvořené za pomoci transformačních matic dané kamery, jeho kvalita však neodpovídá kvalitě dat získaných přímo z lidarových senzorů. Naopak využití čistě lidarových dat přináší problém nízké hustoty mračen bodů (prázdný prostor obvykle tvorí více než 90 % získané reprezentace prostředí). Vhodným řešením těchto překážek je kombinace vstupů z obou senzorů (kamery i lidaru), při tomto přístupu je však náročné dosažení perfektního zarovnání obou typů dat [19].

### 2.2.2 Metody založené na lidarových datech

Metody detekce objektů z lidarového vstupu obecně zahrnují tři fáze – zpracování vstupních dat do vhodné reprezentace prostoru (nutné vzhledem k jejich nestrukturované podobě), extrakce příznaků a nakonec samotné detekce objektů (výstupem je orientovaný hraniční kvádr každého detekovaného objektu spolu s jeho klasifikací).

Možných reprezentací mračna bodů je hned několik – častá je reprezentace bodová, voxelová, sloupcová, grafová či projekční.

Bodová reprezentace ponechává mračno bodů v téměř původní podobě, pouze je omezena jeho velikost, čehož je dosaženo vzorkováním (může být zcela náhodné, nebo se může řídit například metodou FPS<sup>2</sup>).

Voxelová reprezentace dělí 3D prostor o rozměrech  $[L, W, H]$  do voxelů<sup>3</sup> o konstantní velikosti  $[u_L, u_W, u_H]$ . Voxely jsou obvykle popisovány jako nulové (nenachází se v nich žádné body) a nenulové.

Sloupcová reprezentace je velmi podobná voxelové, avšak prostor není segmentován ve směru osy  $z$ . Prostor o rozměrech  $[L, W, H]$  je tedy dělen na sloupce velikosti  $[u_L, u_W, H]$  (výška sloupce je shodná s výškou celého prostoru).

Grafová reprezentace modeluje každý z bodů mračna jako uzel grafu a jeho spojnice se všemi sousedními body do maximální vzdálenosti  $d$  jako grafové hrany.

Projekční reprezentace transformuje body z trojrozměrného prostoru do roviny, přičemž tato transformace je dána zvolenou perspektivou – obvykle se jedná o perspektivu ptačí (*BEV – Bird's Eye View*) nebo perspektivu čelního pohledu (*FV – Front View*) [25].

Příklady modelů detekce objektů využívajících jednotlivé způsoby reprezentace prostoru jsou uvedeny v tabulce 2.1.

Reprezentace prostoru	Modely
Bodová	<i>PointRCNN, StarNet</i>
Voxelová	<i>Vote3Deep, VoxelNet</i>
Sloupcová	<i>PointPillars</i>
Grafová	<i>PointGNN</i>
Projekční – BEV	<i>BirdNet, YOLO3D</i>
Projekční – FV	<i>FVNet</i>

Tabulka 2.1: Příklady modelů využívajících jednotlivé metody reprezentace prostoru.  
Zdroje: [12, 25]

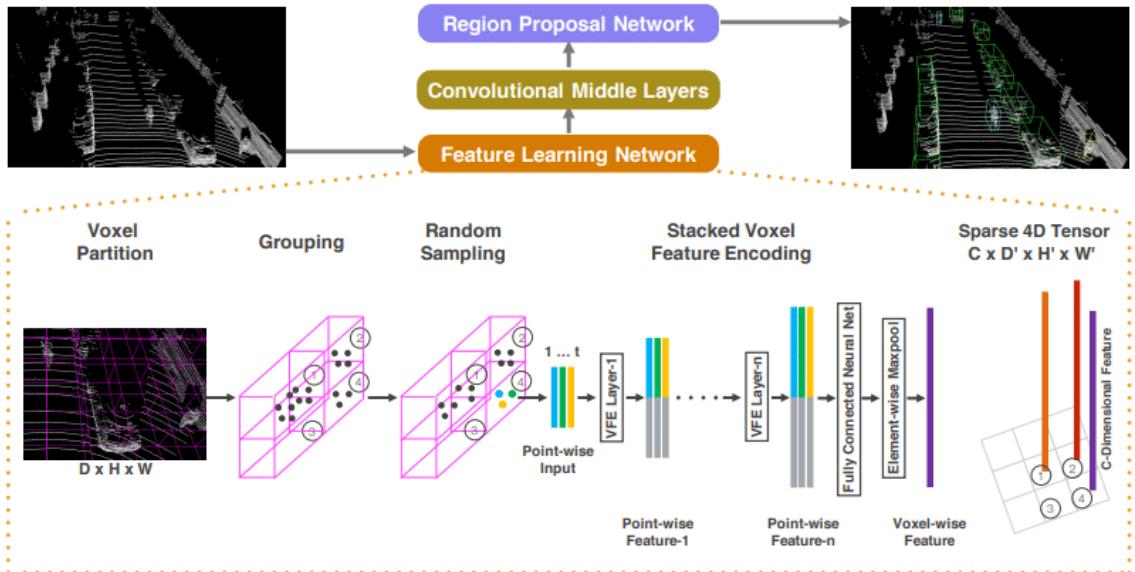
## VoxelNet

VoxelNet [26] je jednofázovou metodou detekce objektů z lidarového mračna bodů, která, jak již napovídá její název, využívá voxelovou reprezentaci prostoru. Body nacházející se uvnitř každého z voxelů jsou seskupeny a následně je z nich náhodně vybrán předem daný počet ke zpracování (obsahuje-li daný voxel nejméně tento počet bodů). Důvodem náhodného výběru namísto zpracování všech voxelů je velikost obvyklého mračna bodů, která se pohybuje kolem 100 tisíc bodů, přímé zpracování každého z nich tedy představuje velkou výpočetní zátěž. Zároveň je vzorkováním alespoň částečně vyrovnaná výrazná variabilita počtu bodů v jednotlivých voxelech.

---

<sup>2</sup>FPS – Furthest Point Sampling – vzorkování nejvzdálenějších bodů, metoda při každé iteraci prochází množinu všech nezpracovaných bodů a vybírá z nich ten, který se nachází v největší vzdálenosti od posledního zpracovaného bodu.

<sup>3</sup>Voxel je prakticky 3D obdobou pixelu, namísto plochého čtverce má však tvar krychle, či kvádru. Jeho účelem je rozdelení prostoru na menší popsatelné oblasti.



Obrázek 2.4: Architektura modelu VoxelNet, převzato ze zdroje [26]

Z neprázdných voxelů jsou pomocí takzvané VFE<sup>4</sup> vrstvy extrahovány příznaky, které jsou pro další fáze zpracování reprezentovány řídkým čtyřrozměrným tenzorem<sup>5</sup> [10].

Tento tensor je předán konvolučním vrstvám, které agregují skupiny voxelů, čímž snižují jejich celkový počet. Následně jsou agregované voxelы zploštěny do dvojrozměrné podoby z ptačí perspektivy – vzniká tzv. BEV mapa příznaků. Finální BEV mapa je nakonec předána poslednímu z funkčních bloků modelu, kterým je RPN síť (*Region Proposal Network*, konvoluční síť předpovídající oblasti, v nichž se nachází detekovaný objekt) [26].

Schéma architektury modelu je ukázáno na obrázku 2.4. Předností modelu VoxelNet je jeho výkonnost, naopak nevýhodou je relativně pomalé zpracování dat způsobené mnoha prováděnými konvolučními operacemi [10].

### 2.2.3 Metody posuzování výsledků

Aby bylo možné porovnávat výsledky dosažené různými detekčními algoritmy, je k jejich vyhodnocení nutné znát a využívat stejné standardní metriky. Definice metrik jsou převzaty ze zdrojů [14] a [17].

Před samotným popsáním těchto metrik je nutno zavést několik užívaných pojmu pro kategorizaci výsledku detekce:

- správná detekce (*TP – True Positive*) – byl detekován existující objekt
- falešně pozitivní (*FP – False Positive*) – byl detekován neexistující objekt
- falešně negativní (*FN – False Negative*) – existující objekt nebyl detekován

<sup>4</sup>VFE – *Voxel Feature Encoding*, volně přeloženo „zakódování voxelových příznaků“

<sup>5</sup>Tenzor představuje multidimenzionální pole, přičemž jeho řád odpovídá počtu indexů potřebných k adresaci konkrétní hodnoty, je tedy zobecněním vektorů (tenzory prvního řádu) i matic (tenzory druhého řádu). Využití tensorů je velmi běžné v oblastech strojového učení a počítačového vidění [18].

Správnost detekce bývá běžně vyhodnocována metodou IoU (Intersection-over-Union), tedy porovnáním hodnoty podílu plochy sjednocení a průniku detekovaného ohraničení a reálného ohraničení objektu se zvolenou prahovou hodnotou. Je-li pak poměr IoU větší než zadaná prahová hodnota, je detekce považována za správnou.

$$\text{IoU} = \frac{\text{sjednocení}}{\text{průnik}} = \frac{\text{area of overlap}}{\text{area of union}}$$

Obrázek 2.5: Metoda intersection-over-union

### Přesnost

Přesnost (*precision* –  $P$ ) odpovídá poměru případů, kdy byla detekce objektů správná. Jedná se tedy o poměr správných detekcí vzhledem k jejich celkovému počtu.

$$P = \frac{TP}{TP + FP} \quad (2.1)$$

### Úplnost

Úplnost (*recall* –  $R$ , také zvaná sensitivita) udává pravděpodobnost, že bude reálný ohraňující kvádr správně detekován.

$$R = \frac{TP}{TP + FN} \quad (2.2)$$

### Křivka přesnost-úplnost

Pro určení hodnot několika dalších metrik slouží křivka přesnost-úplnost (*precision-recall*,  $P-R$ ). Na ose  $x$  této křivky jsou hodnoty úplnosti, na ose  $y$  pak hodnoty přesnosti, přičemž rozmezí hodnot obou os je maximálně  $[0, 1]$ . V případě dosažení ideálních výsledků se křivka blíží bodu  $(1, 1)$ .

### Průměrná přesnost

Průměrná přesnost (*average precision* –  $AP$ ) je vypočítána z více, nejčastěji 11 hodnot přesnosti, odpovídajícím na P-R křivce rovnoměrně rozmištěným hodnotám úplnosti (hodnoty od 0 do 1, včetně, vždy ve vzdálenosti 0.1). Hodnoty přesnosti jsou interpolovány tak, že je zvolena nejvyšší z hodnot dosažených mezi bodem předcházejícím a aktuálním.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \dots, 1\}} P_i(r) \quad (2.3)$$

## Průměrná přesnost napříč třídami

Průměrná přesnost napříč všemi třídami (*mean average precision – mAP*) detekovanými v rámci experimentu je pro N tříd objektů definována následovně:

$$mAP = \frac{1}{N} \sum_{i=0}^N AP_i \quad (2.4)$$

### 2.2.4 Datové sady

Pro učení modelů určených k detekci objektů v lidarových datech jsou potřeba vzorky dat, zahrnující kromě mračna bodů také jeho anotaci – nezbytná je znalost pozice a velikosti ohraničujících kvádrů vyskytujících se objektů.

Mezi vhodné datové sady patří například sady KITTI [9], nuScenes [3] a Waymo Open Dataset [21], které jsou níže blíže představeny.

#### KITTI

KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [9] je tvůrce jedné z nejpopulárnějších datových sad, užívaných pro učení modelů pro autonomní řízení. Obsahuje totiž velké množství dat sesbíraných za jízdy osobním vozidlem po městských i mimoměstských cestách a dálnicích v okolí města Karlsruhe, pokrývá tedy mnoho různých dopravních situací.

Sada KITTI definuje několik kategorií úloh, k jejichž učení a testování ji lze využít. Sem patří například detekce objektů ve variantách 2D, 3D i BEV, sledování trasy objektů po jejich detekci nebo sledování horizontálního značení na vozovce (to je v praxi využíváno třeba v asistentečních systémech jízdy v pruhu).

Část sady určená pro 3D detekci objektů [9], která je využívána pro experimenty v rámci této práce, zahrnuje obrazová data z kamer, jim odpovídající lidarová mračna bodů, kalibrační matice senzorů (formát popsán v tabulce 2.2) a soubory s popisky vyskytujících se objektů (formát popsán v tabulce 2.3). Při tvorbě sady byly jejími autory ke snímání okolního prostředí využity dvě barevné kamery, dvě achromatické kamery, lidar (Velodyne HDL-64E) a senzor GPS (navigační systém) [8].

KITTI definuje tři úrovně obtížnosti detekce vyskytujících se objektů – jednoduchou („easy“), středně náročnou („moderate“) a náročnou („hard“). Konkrétní detekované objekty spadají do jednotlivých kategorií v závislosti na jejich vzdálenosti od vozidla osazeného senzory a na jejich viditelnosti. Jednoduše rozpoznatelné jsou objekty s nulovým překrytím, silně překryté objekty naopak spadají do náročné kategorie.

Metrikou užívanou pro vyhodnocování výsledků detekce je průměrná přesnost napříč třídami (*mAP*). Prahou hodnota IoU pro prohlášení detekce vozidla za úspěšnou je 0.7, pro cyklisty a chodce je tato hodnota 0.5. Pro výpočet průměrné přesnosti (*AP*) vychází KITTI ze 40 bodů úplnosti na PR křivce namísto standardních 11 bodů [25].

Potenciálním nedostatkem této datové sady je znatelný nepoměr mezi počtem výskytů vozidel oproti chodcům s cyklisty v jejích testovacích vzorcích, který je uveden v tabulce 2.4. Ještě markantnější je pak rozdíl mezi počtem vzorků náročné kategorie, obsahujících tyto typy objektů, přičemž silně překrytých výskytů cyklistů se v celé datové sadě nachází pouze 31.

Počet hodnot	Název	Význam
3x4	p0-p3	Projekční matice kamery (KITTI využívá čtyři kamery)
3x3	r0_rect	Rektifikační matice kamery
3x4	tr_velodyne_to_cam	Matice pro převod souřadnic lidaru do souřadnic kamery
3x4	tr_imu_to_velo	Matice převádějící data senzoru náklonu do souřadnic lidaru

Tabulka 2.2: KITTI formát kalibračních souborů

Počet hodnot	Název	Význam
1	type	Typ objektu – relevantními hodnoty jsou zde především <i>Car</i> , <i>Cyclist</i> a <i>Pedestrian</i>
1	truncated	Poměrná část objektu, která se nachází mimo hranice obrazu kamery
1	occluded	Míra překrytí daného objektu
1	alpha	Observační úhel objektu v rozmezí hodnot $[-\pi, \pi]$
4	bbox	Souřadnice krajních pixelů ohraničujícího obdélníka (dvojice souřadnic $x$ a $y$ pro jeho levý horní a pravý dolní roh)
3	dimensions	3D rozměry objektu v metrech (pořadí: výška, šířka, délka)
3	location	3D souřadnice pozice objektu v metrech (v souřadnicích kamery, pořadí: x, y, z)
1	rotation_y	Rotace objektu okolo osy $y$ v souřadnicích kamery (rozmezí hodnot $[-\pi, \pi]$ )

Tabulka 2.3: KITTI formát popisků

Kromě již výše zmíněných tříd (osobní vozidlo, chodec, cyklista), definuje KITTI v popiscích celkem pět dalších tříd objektů, včetně dodávek, nákladních vozidel a tramvají, což značně rozšiřuje možnosti využití, a tím i popularitu této datové sady.

### Práce s datovou sadou KITTI

V rámci procesu učení modelu detekce objektů na vzorcích KITTI jsou kamerová a/nebo lidarová data anotována pomocí údajů, uvedených v popiscích, a kalibračních matic.

Pro účely 2D detekce lze průmět ohraničujícího obdélníka objektu do obrazu kamery získat velmi jednoduše ze souřadnic jeho levého horního a pravého dolního rohu (hodnoty *bbox* pro daný řádek anotačního souboru).

Pro modely prostorové detekce je v závislosti na typu vstupních dat, s nimiž pracují, nutné znát průmět ohraničujícího kvádru detekovaného objektu do obrazu kamery, nebo do lidarového mračna bodů. Souřadnice vrcholů ohraničujícího kvádru v obrazu jsou určovány

Typ objektu	Výskyty v sadě	Výskyty se silným překrytím
Osobní auto	19 756	3 099
Chodec	2 207	257
Cyklista	734	31

Tabulka 2.4: Trénovací sada KITTI – Výskyt objektů, zdrojem je analýza vzorků sady vlastním skriptem (více v příloze A)

pomocí souřadnic jeho pozice (*location*) a jeho rotace a rozměrů (*rotation\_y*, *dimensions*). Pro mapování prostorových kamerových souřadnic do souřadnic pixelových pak slouží projekční matice kamery, jíž byl daný obraz získán. Do souřadnic lidarových lze pak ohraničující kvádr ze souřadnic kamerových transformovat za pomoci inverzní matice k matici převodu lidar-kamera (*tr\_velodyne\_to\_cam*), která je uvedena v kalibračních souborech.

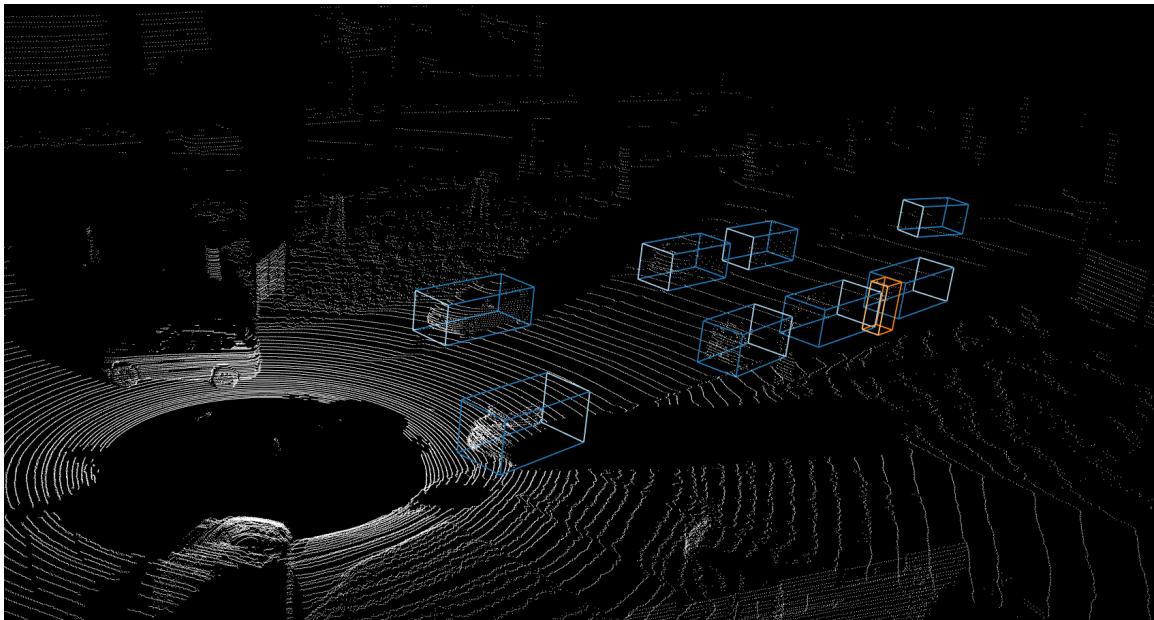
Ukázka průmětů ohraničujících obdélníků do obrazu kamery a ohraničujících kvádrů do obrazových a lidarových dat vzorků datové sady KITTI je uvedena na obrázcích 2.6, 2.7 a 2.8.



Obrázek 2.6: Průmět ohraničujících obdélníků do obrazu kamery, vygenerováno za pomocí [29]



Obrázek 2.7: Průmět ohraničujících kvádrů do obrazu kamery, vygenerováno za pomocí [29]



Obrázek 2.8: Průmět ohraňujících kvádrů do mračna bodů, vygenerováno za pomoci [29]

### NuScenes

Datová sada nuScenes [3], která byla zveřejněna v roce 2020, a je tedy výrazně novější než datová sada KITTI, zahrnuje lidarová mračna bodů ze senzoru využívajícího 32 laserových paprsků, radarová data z pěti radarů a kamerová data ze šesti RGB kamer. Zmiňované senzory jsou na vozidle sbírajícím vzorky sady rozmištěny tak, že je jimi pokryt zorný úhel  $360^\circ$ .

V rámci této sady je definováno celkem 23 různých tříd objektů, z nichž 10 je vyhodnocováno při prostorové detekci. NuScenes definuje vlastní skalární metriku NDS – *nuScenes Detection Score*, která je vypočítána na základě průměrné úspěšnosti detekce napříč třídami (*mAP*) a dalších metrik určovaných pro správné detekce (*TP*).

Mezi tyto metriky správných detekcí patří například odchylky predikované orientace, velikosti nebo rychlosti detekovaného objektu oproti jejich reálným hodnotám [25].

### Waymo Open Dataset

Waymo Open Dataset [21] je další z novějších datových sad pro učení modelů detekce objektů pro autonomní řízení. Sada zahrnuje lidarová data ze čtyř lidarů s krátkým dosahem a jednoho lidaru se středním dosahem a kamerová data z pěti RGB kamer s vysokým rozlišením. Ke vstupním datům náleží také kalibrace jednotlivých senzorů a transformace pro převody mezi nimi. Senzory opět pokrývají úhel celých  $360^\circ$  okolo vozidla, na němž jsou osazeny.

Sada WOD definuje pouze čtyři třídy objektů – vozidla, chodce, cyklisty a dopravní značky. Pro tyto třídy objektů, nachází-li se v blízkosti vozidla, jsou dostupné také popisky jejich ohraňujících kvádrů. Objekty jsou členěny do dvou úrovní obtížnosti jejich detekce („LEVEL1“, „LEVEL2“). Do vyšší úrovně obtížnosti je objekt zařazen, pokud sestává z méně než šesti lidarových bodů, nebo je mu tato úroveň manuálně přiřazena anotátorem.

Pro posouzení výsledků detekce využívá sada metriku plochy pod křivkou průměrné přesnosti (*AP*), namísto přístupu interpolace určitého počtu bodů na PR křivce tedy tuto křivku integruje [25].

## 2.3 Simulátor CARLA

CARLA [6] je open-source simulátor určený pro výzkum v odvětví autonomního řízení. Jeho název je zkratkou sousloví „car learning to act“, sloužit tedy může k vývoji, učení a validaci systémů. Tato podkapitola se věnuje bližšímu představení simulátoru a možnostem jeho využití. Vzhledem ke komplexnosti simulátoru jsou zde uvedena primárně fakta podstatná pro technické řešení této práce. Zdrojem technických údajů o simulátoru je jeho dokumentace [6].

Jednou z největších výzev pro autonomní řízení jsou městská prostředí za velké intenzity provozu, kdy je nutné, aby systém detekoval a následně sledoval chování mnohdy až stovek okolních účastníků provozu. Zároveň je velmi důležité, aby se systém správně zachoval ve scénářích, jako je vběhnutí dítěte do vozovky, či pád osoby na přechodu pro chodce. Testování vyvíjených systémů v těchto podmírkách v reálném světě je však z bezpečnostních důvodů prakticky nemožné.

Toto zjištění vedlo kromě vzniku platformy CARLA také k pokusům o využití různých počítačových her na bázi simulátoru řízení pro učení a testování vyvíjených systémů [15], ty však povětšinou neposkytovaly dostatečnou komplexitu městského provozu.

Z technického hlediska využívá CARLA herní engine<sup>6</sup> Unreal Engine 4, který poskytuje vysokou kvalitu obrazu a realistickou herní fyziku a umožňuje přidávání dalších rozšíření. Samotný simulátor je pak koncipován jako architektura server-klient. Server umožňuje připojení více klientů, provádí samotnou simulaci a vykresluje obraz aktuální scény.

CARLA zahrnuje klienské API<sup>7</sup> v jazyce Python, umožňující tvorbu klientských skriptů, které mohou serveru odesílat různé příkazy (například k ovládání simulovaných vozidel) a naopak z něj odebírat data o aktuálním stavu jeho prostředí.

Simulátor obsahuje hned několik různých map městského prostředí, které jsou tvořeny 3D modely nepohyblivých objektů (budovy, stromy a další vegetace, dopravní značky ap.). Většina map má i svou vrstvenou variantu, ve které lze jednotlivé vrstvy prostředí přidávat či odebírat (vrstvami zde rozumíme soubor objektů v mapě s tímtoé sémantickým významem – například soubor cest s chodníky nebo soubor všech budov), pro účely této práce však byly využity pouze mapy nevrstvené. Pohyblivé objekty (aktéři) jsou do prostředí mapy přidávány klientskými skripty.

Součástí platformy CARLA je knihovna šablon všech objektů, které lze v simulaci využívat – pohyblivých i nepohyblivých. Z pohyblivých objektů se ve verzi 0.9.13 jedná o předlohy několika desítek modelů vozidel, vycházejících z vozů vyskytujících se v reálném provozu (z osobních aut byl napodoben například Citroën C3, ale zahrnutý jsou i modely dodávek či hasičského vozidla), několik modelů motocyklů, jeden model jízdního kola a zhruba 50 různých modelů chodců. Z nepohyblivých objektů zahrnuje knihovna šablon například dopravní kužely a lavičky, ale také různé drobné objekty, které mohou používat chodci, a velký výběr senzorů pro snímání simulovaného světa. Tyto senzory lze připojit k některému z vozidel, a učinit je tak pohyblivými.

<sup>6</sup>Doslově přeloženo „herní motor“, avšak český překlad tohoto pojmu se prozatím příliš nepoužívá. Zjednodušeně řečeno se jedná o program, v němž je tvořen celek dané počítačové hry.

<sup>7</sup>Application Programming Interface – česky aplikační programové rozhraní, tedy rozhraní pro předávání informací mezi daným systémem a programy, které jej využívají

### 2.3.1 Využití simulátoru

Jedním z nejčastějších využití simulátoru CARLA je porovnávání kompletních systémů autonomního řízení. K simulátoru náleží žebříček těchto systémů, který porovnává výkonost autonomních agentů v realistických dopravních scénářích. Systémy, účastnící se této soutěže, mohou využívat většinu typů senzorů, které byly v rámci této práce dosud zmíněny – GPS, lidar, radary, kamery a další.

Kromě těchto kompletních systémů však lze simulátor využívat i pro modely zabývající se jednotlivými podúlohami autonomního řízení. Několik existujících prací se zabývá specificky využitím syntetických dat ze simulátoru pro učení modelů detekce objektů.

Řešení KittCarla [5] se zabývá tvorbou datové sady v KITTI formátu. Simulovaný lidarový senzor byl nastaven se stejnými parametry jako model lidaru Velodyne HDL-64E využitý pro tvorbu původní sady KITTI. Dále byly simulovány dvě RGB kamery s rozlišením 1394x1024 pixelů a zorným úhlem 72° (obrazová data KITTI mají rozměr 1242x375 pixelů a pokrývají rozný úhel 90°). Spolu s lidarovými a kamerovými daty byly vygenerovány také kalibrační soubory, avšak datová sada nezahrnuje soubory s popisky vyskytujících se objektů, a pro využití k učení modelů prostorové detekce je tedy nevhodná.

Další prací, která se zabývala využitím umělých dat ze simulátoru, je [7]. Za využití verze CARLA 0.9.4 bylo autory této práce vytvořeno 8100 vzorků, sestávajících z lidarových a kamerových dat, jejich kalibračních souborů i souborů s popisky detekovatelných objektů. Tyto vzorky byly v rámci provedených experimentů využity pro učení modelů VoxelNet, YOLO3D a PointPillars (jedná se o modely detekce objektů z lidarových dat, diskutované v podkapitole 2.2.2). Pro vyhodnocení výsledků byla zvolena metrika průměrné přesnosti napříč třídami (*mAP*), v modifikaci užívané datovou sadou KITTI.

Výsledky dosažené v rámci citované práce ukázaly, že kombinace reálných a syntetických datových vzorků do jedné datové sady není pro učení modelů vhodná – došlo totiž ke snížení přesnosti. Pozitivní výsledky ale přineslo učení modelu na pouze syntetických datech, následované jeho doladěním na vzorcích reálné datové sady. Podrobnosti nebo postup tvorby využité datové sady bohužel nebyly jejími autory zveřejněny.

CarFree [13] je řešením pro získání umělých datových vzorků určených primárně k detekci chodců z kamerového vstupu. Motivací jeho autorů byla možnost doplnění existujících sad vzorky z nebezpečných dopravních situací. Ačkoliv se řešení nezabývá detekcí z lidarových dat, je pro tuto práci zajímavé, jelikož se věnuje i automatické tvorbě souborů s popisky vyskytujících se objektů. Prezentovaným postupem jejich tvorby byly kroky: transformace souřadnic, převod souřadnic objektu na souřadnice hraničního obdélníku, filtrování překrytých objektů a nakonec oprava hraničního obdélníku tak, aby se jeho strany v každém případě dotýkaly krajních bodů objektu. Pro filtrování objektů podle jejich překrytí i opravy jejich hraničních obdélníků byla využita data získaná z obrazu sémantické segmentační kamery dostupné v simulátoru CARLA. Překrytí objektů bylo posuzováno pouze kontrolou, zda barva jejich středového pixelu odpovídá sémantickému významu objektu, nebylo tedy možné rozlišit úrovně překrytí jako v datové sadě KITTI.

Výsledky dosažené učením modelu 2D detekce objektů na kombinaci vzorků reálné sady a sady vytvořené řešením CarFree byly pozitivní v každém z prezentovaných případů. Nejvhodnějšími umělými vzorky byly takové, které obsahovaly čtyři nebo více detekovatelných objektů.

# Kapitola 3

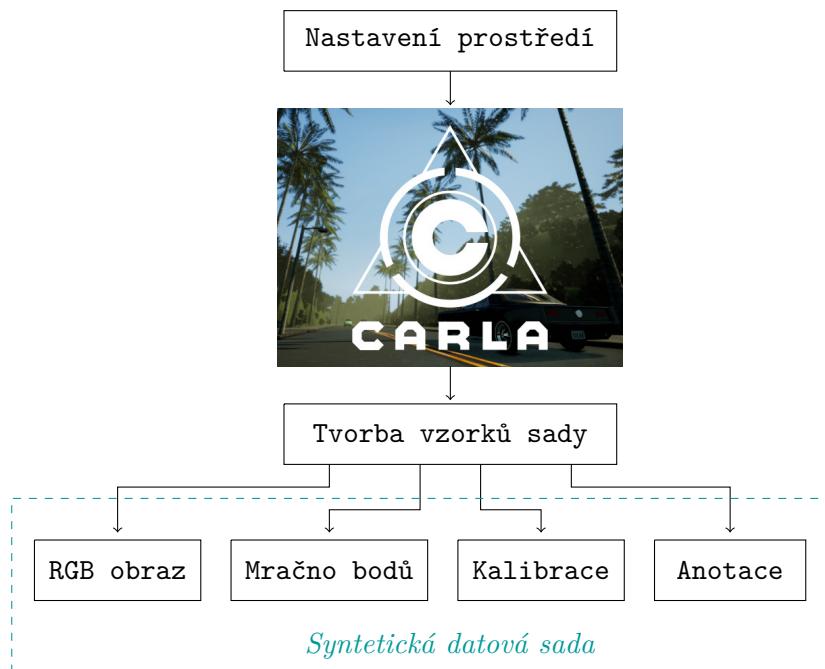
## Návrh a implementace

Práce se zaměřuje na možnost zlepšení výsledků rozpoznání cyklistů a chodců, dosahovaných existujícími modely, za pomoci rozšíření datové sady KITTI o další vzorky, získané ze simulátoru autonomního řízení CARLA. Cílem kapitoly je tedy přiblížit čtenáři nejprve abstraktní myšlenku, a následně i technické provedení řešení pro tvorbu takového rozšíření sady, jehož předností je primárně velmi nízká časová i finanční náročnost oproti tvorbě tradiční datové sady v KITTI formátu.

V poslední podkapitole je pak popsána výsledná datová sada, včetně prostředí, které bylo k její tvorbě využito.

### 3.1 Návrh řešení

Navrhované řešení pro tvorbu umělé datové sady lze rozdělit na dvě hlavní fáze – nastavení prostředí simulátoru a tvorbu vzorků sady. Schéma řešení je přiloženo na obrázku 3.1.



Obrázek 3.1: Schéma řešení

V rámci přípravy prostředí je zvolena některá z dostupných map simulátoru, v níž jsou následně vytvořeny instance požadovaných účastníků provozu. Fáze tvorby vzorků datové sady probíhá v cyklech, přičemž v každém z nich jsou načtena, zpracována a uložena data všech čtyř skupin souborů definovaných formátem KITTI pro prostorovou detekci objektů (RGB obraz, lidarové mračno bodů, kalibrační soubory, anotační soubory). Obě fáze tvorby datové sady jsou blíže představeny v následujících podkapitolách.

## 3.2 Nastavení prostředí simulátoru

Jelikož účelem nově vytvářené datové sady je alespoň částečné vyrovnání nepoměru mezi počtem výskytů cyklistů a chodců oproti vozidlům v datové sadě KITTI, je nutno v simulátoru nejprve vygenerovat provoz s požadovanou reprezentací těchto aktérů.

Pro co největší možnou diverzitu jednotlivých vzorků je žádoucí využít ke tvorbě datové sady několik různých map simulátoru. Je však nutno volit takovou mapu, která obsahuje pouze minimum neaktivních vozidel (v některých mapách jsou například zaparkována podél cesty), jelikož jejich přítomnost ve výhledu ego vozidla<sup>1</sup> nelze z dat simulátoru zjistit, a znemožňuje tedy tvorbu odpovídajících popisků snímků.

K provedení výše popsaných úkonů slouží skript `generator_actors.py`<sup>2</sup>, jehož struktura vychází z dokumentace simulátoru CARLA [6] a zveřejněných ukázkových skriptů.

Skriptu jsou předány následující vstupní parametry:

- volba mapy (v závislosti na verzi simulátoru)
- požadovaný počet vozidel
- požadovaný počet cyklistů
- požadovaný počet chodců

Vozidla a cyklisté využívají pro autopilotní mód tzv. traffic manager (v doslovém překladu správce provozu), a jelikož je jejich samostatná aktivita žádoucí, je zde vytvořena i instance této třídy a připojena k odpovídajícímu portu serveru. Traffic manager umožňuje také nastavení chování vozidel, která jsou pod jeho správou – například jejich minimální vzdálenost od vozidla před sebou (zde nastavena na 3 metry), nebo míru jejich dodržování rychlostních limitů určených dopravními značkami (ponechána výchozí).

Prvním krokem umístění aktéra do světa simulátoru je výběr jeho šablony z knihovny simulátoru. Zvolená šablona je následně přizpůsobena nastavením hodnot jejích konkrétních vlastností jako je barva a nebo právě zapnutí autopilotního chování.

Každá mapa pak má určitou množinu bodů, do nichž lze umístit nově vytvořenou instanci aktéra (tzv. spawn points). Pro každou instanci vozidla/cyklisty je náhodně vybrán jeden z těchto bodů. Jelikož je jejich počet omezený, může dojít ke kolizi, kdy se simulátor pokusí umístit dva aktéry do jednoho bodu mapy, což může mít za následek menší výsledný počet vygenerovaných aktérů, než byl požadován.

Po vytvoření všech vozidel a cyklistů jsou tyto objekty dávkově předány do správy traffic manageru a následně jsou jim nastavena zapnutá potkávací světla či, v případě cyklistů, přední a zadní světla.

<sup>1</sup>Běžně užívaný technický termín označující vozidlo, pro nějž jsou údaje o okolí získávány, synonymem může být „vozidlo z výhledu“.

<sup>2</sup>Všechny skripty zmiňované v této technické zprávě jsou obsahem přiloženého paměťového média, jejich kompletní seznam je uveden v Příloze A.

Dalším krokem je osazení světa simulátoru chodci, což vzhledem k požadavku na jejich samostatnou aktivitu opět probíhá ve více krocích. Ke každému chodci totiž náleží také instance jeho ovladače, který umožňuje nastavovat chodcům náhodnou cílovou lokalitu, do níž pak budou v průběhu simulace směřovat.

Po vytvoření všech požadovaných aktérů je žádoucí, aby opustili body, v nichž byli vytvořeni, a bylo tak zamezeno kolizím při spuštění dalšího skriptu. Aktuální skript tedy zavolá 100 časových kroků simulace (jeden krok odpovídá jedné polovině sekundy), během nichž se aktéři pohnou ze svých původních pozic. Následně již tento skript další časové kroky simulátoru nevyvolává, pouze na ně čeká. Zde je nastaven časový limit – v případě, že v tomto limitu není klientský skript serverem o časové změně informován, odstraní nejprve všechny instance aktérů z mapy a následně se ukončí.

Je-li žádoucí, aby se v průběhu tvorby datové sady v simulátoru dynamicky měnilo počasí, je k tomu vhodné využít standardní skript simulátoru s touto funkcionalitou, vzhledem k čemuž zde tato možnost není duplicitně implementována.

### 3.3 Tvorba datové sady

V této podkapitole je popsán postup tvorby vzorků datové sady v KITTI formátu – tedy souborů obsahujících obraz snímaný kamerou, lidarové mračno bodů, popisky objektů vyskytujících se ve výhledu a kalibrační matice senzorů.

Čtení a ukládání vzorků je úlohou skriptu `generator_kitti.py`, kterému je zadán požadovaný počet vytvořených vzorků a případně i to, od jaké počáteční hodnoty ukládané soubory číslovat (výchozím nastavením je zde snímek 0).

Po připojení se k serveru simulátoru a traffic manageru (jeho účel a inicializace byly popsány v předcházející podkapitole) následuje načtení šablon pro senzory – RGB kamery, hloubkovou kameru (potřebná pro určení míry překryvu jednotlivých objektů) a lidar. Senzorům jsou nastaveny požadované parametry, imitující senzory užité ke tvorbě datové sady KITTI.

V dalším kroku je vytvořena instance ego vozidla a instance všech senzorů, které jsou k ní pro účely simulace pevně připojeny.

Následně se spouští hlavní smyčka tvorby nové datové sady, jejíž počet iterací odpovídá požadovanému počtu vytvářených vzorků. Klientský skript volá časové kroky simulátoru, při každém z nich odebírá senzorová data a po 60 krocích volá postupně funkci pro načtení aktuálně viditelných objektů a poté funkce pro uložení všech čtyř typů souborů. Není-li v dané iteraci ve výhledu žádný detekovatelný objekt, je provedeno dalších 60 časových kroků.

#### 3.3.1 Seznam viditelných objektů

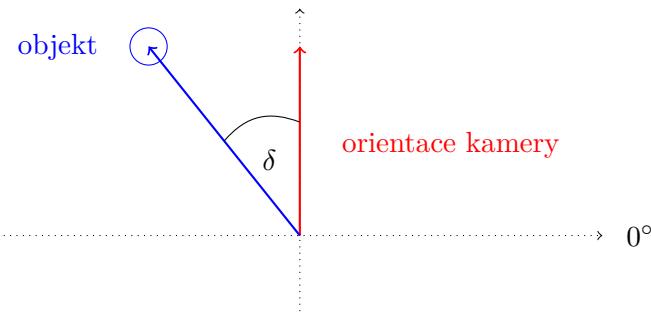
Získání odpovídajícího seznamu objektů, které jsou v daný moment viditelné kamerou ego vozidla, představuje jeden z komplikovanějších procesů v rámci tvorby doplňující datové sady. Simulátor CARLA umožňuje načtení seznamu všech aktérů aktuálně se vyskytujících v mapě světa, avšak tento seznam je nutno filtrovat tak, aby byl omezen pouze na objekty ve výhledu, a to pouze v dosahu kamery. Pro tento filtr bylo využito existující řešení CARLA Vehicle 2D Bounding Box Annotation Module [1].

Ze světa simulátoru je nejprve načten seznam všech aktivních objektů. Tento seznam je předán filtrovacím funkcím, které z něj nejprve odeberou objekty ve vzdálenosti od senzoru větší než zadaná maximální hodnota (zde bylo zvoleno 50 metrů). Toto filtrování je

provedeno za pomoci průchodu seznamem aktérů a využití standardní funkce simulátoru CARLA pro načtení jejich lokality v mapě (získanou hodnotou je objekt s určením jednotlivých souřadnic). Vzdálenost od vozidla je pak vypočítána jako vzdálenost souřadnic bodu, ve kterém se nachází posuzovaný objekt, a bodu, v němž se nachází kamera.

Následně jsou ze seznamu odebrány i objekty, které se nenachází v úhlu pokrytém kamerou (zde se jedná o úhel  $90^\circ$ ). Opět je využita standardní funkce API simulátoru, která umožňuje načtení aktuální orientace senzoru. V rámci filtrování je vypočítána odchylka orientace kamery a vektoru spojujícího kameru se středem posuzovaného objektu. Absolutní hodnota této odchylky je pak porovnána s polovinou zorného úhlu kamery.

Při využití značení z ilustrace 3.2 lze filtrovací podmínu zapsat jako:  $|\delta| < 45^\circ$ .



Obrázek 3.2: Ilustrace filtrování aktuálně viditelných objektů

Takto připravený seznam viditelných objektů je předán funkcím pro tvorbu popisků aktuálního snímku.

### 3.3.2 Soubory s popisky

Po získání seznamu objektů, které se aktuálně nachází v zorném poli kamery ego vozidla, je nutné pro každý z nich vytvořit anotační řádek v KITTI formátu. Tento formát je popsán v níže přiložené tabulce 2.3. Pro práci s jednotlivými řádky byla vytvořena třída `Label_Row`, která uchovává všechny hodnoty definované formátem a definuje metody pro práci s nimi, včetně finálního převodu všech hodnot na jeden textový řetězec pro účel jejich zápisu do textového souboru.

Tvorba anotačního souboru pro daný snímek je tvořena cyklem, procházejícím seznam aktuálně viditelných aktérů (objekty třídy `carla.Actor` v klientském API simulátoru). V následujících několika odstavcích je popsán postup získávání jednotlivých hodnot popisku za pomocí této třídy a jejích atributů (uváděny jsou v pořadí jejich zpracování klientským skriptem).

#### Typ objektu

Každý z objektů třídy `carla.Actor` nese jako atribut také identifikátor šablony, podle níž byl vytvořen, za pomocí kterého ji lze z knihovny šablon načíst a na základě jejích vlastností objekt klasifikovat do odpovídající třídy. Kromě typů *Car*, *Cyclist* a *Pedestrian* jsou využívány i typy *Van*, *Truck* a *DontCare*, přičemž poslední ze jmenovaných typů slouží

k označení objektů, které jsou podle pravidel definovaných formátem KITTI příliš malé<sup>3</sup>, ale zároveň není žádoucí, aby bylo nalezení objektu v této části obrazu počítáno jako falešné pozitivum (tedy nalezení objektu, který neexistuje).

## Rozměry

3D rozměry objektu v jednotkách metrů lze z instance třídy `carla.Actor` získat prostřednictvím jejího odkazu na reprezentaci ohraničujícího kvádru daného objektu třídou `carla.BoundingBox`, která má následující atributy:

- pozice – souřadnice odpovídající středu objektu
- rotace – úhly natočení, náklonu a valení
- rozměry – uloženy v podobě 3D vektoru vedoucího ze středu objektu do jednoho z vrcholů ohraničujícího kvádru (jedná se tedy o polovinu rozměru celého kvádru ve směru každé osy)

Označíme-li objekt hraničního kvádru proměnnou `bbox`, je pro určení rozměrů objektu v metrech využít vztah:

$$[x, y, z] = [2 * bbox.extent.x, 2 * bbox.extent.y, 2 * bbox.extent.z]$$

V případě popisování objektu cyklisty poskytuje simulátor pouze pozici a rozměry jízdního kola, je k nim tedy nutno započítat i výšku osoby, která na něm sedí. Zde se experimentálně osvědčilo násobení výšky jízdního kola koeficientem 1.5.

## Pozice

Pozici objektu lze určit více způsoby, první z nichž (získání souřadnic středu za pomocí ohraničujícího kvádru) byl představen v přechozím odstavci. Druhým způsobem je pak standardní metoda třídy aktéra, jejímž zavoláním je objekt se souřadnicemi pozice získán přímo. Soustavy souřadnic simulátoru CARLA a datové sady KITTI se však navzájem liší, je tedy nutno provést jejich převod<sup>4</sup>:

$$[x_K, y_K, z_K] = [-x_C, -y_C, z_C]$$

Jedná-li se o postavu chodce, je pro získání odpovídající pozice navíc k souřadnici  $y$  nutno přičíst polovinu výšky jeho hraničního kvádru.

## 2D ohraničení, míra překrytí a oříznutí

Pro získání souřadnic krajních bodů hraničního obdélníka je opět využito existující řešení CARLA Vehicle 2D Bounding Box Annotation Module [1].

Z dat poskytovaných simulátorem (souřadnic středu tělesa a jeho rozměrů) jsou nejprve určeny souřadnice všech osmi vrcholů hraničního kvádru. Ty jsou následně transformovány do pixelových souřadnic určujících jejich pozici v obrazu kamery, k čemuž je využita projekční matici.

---

<sup>3</sup>Minimem pro anotaci objektů v sadě KITTI je výška hraničního obdélníku 25 pixelů, přičemž objekt se musí zároveň z více než 50 % nacházet v hranicích obrazu [9].

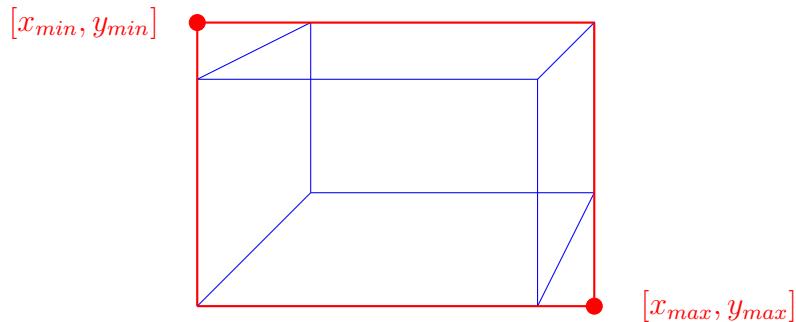
<sup>4</sup>Dolní index K označuje souřadnice datové sady KITTI, souřadnice simulátoru CARLA jsou označeny dolním indexem C.

Ze získaných 2D souřadnic všech osmi vrcholů prostorového ohraničení objektu je za využití hloubkové mapy získané pomocí hloubkové kamery nejprve vypočítána míra překrytí objektu jeho okolím. Zde byl zvolen zjednodušený přístup, kdy je zjištěna hloubka čtyř nejbližších pixelů obklopujících každý z vrcholů. Je-li hloubka každého z nich menší, než je vzdálenost vrcholu od kamery, je tento vrchol označen za překrytý.

Podle počtu překrytých vrcholů jsou podniknutý případné kroky:

- překryto 4-5 vrcholů – objekt je klasifikován jako částečně překrytý (v KITTI formátu nastavení hodnoty *occlusion* rovno 1)
- překryto 6-7 vrcholů – objekt je klasifikován jako obtížně viditelný (*occlusion* nastaveno na hodnotu 2)
- překryto 8 vrcholů – objekt není ani částečně viditelný, do souboru s popisky tedy jeho řádek nebude vůbec zapsán

Po určení míry překrytí objektu jsou souřadnice pixelů, v nichž se nachází vrcholy hraničního kvádru, využity i k určení vrcholů jeho 2D alternativy (ilustrováno na obrázku 3.3).

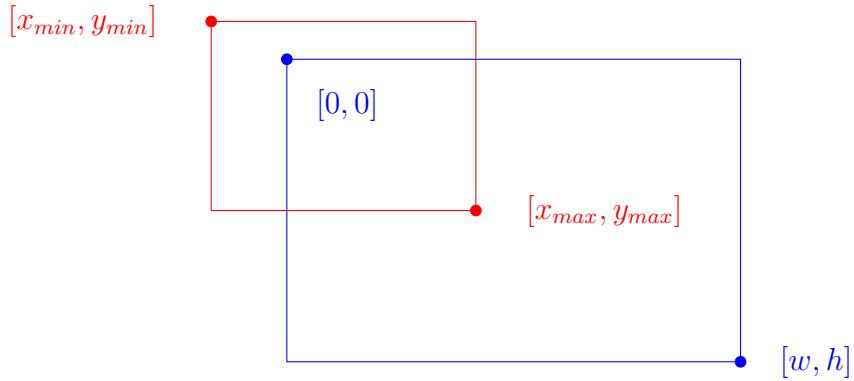


Obrázek 3.3: Převod hraničního kvádru na hraniční obdélník

Ohraničení objektu odpovídajícím obdélníkem je ve 2D souřadnicích reprezentováno pouze dvěma body – jeho horním levým a pravým dolním rohem. Jejich souřadnice jsou získány zjištěním minima a maxima pro obě souřadnice ( $x$  a  $y$ ) v rámci všech osmi vrcholů hraničního kvádru.

Spolu se souřadnicemi hraničního obdélníka je vypočítána i poměrná část objektu, která se nachází mimo hranice obrazu (míra jeho oříznutí).

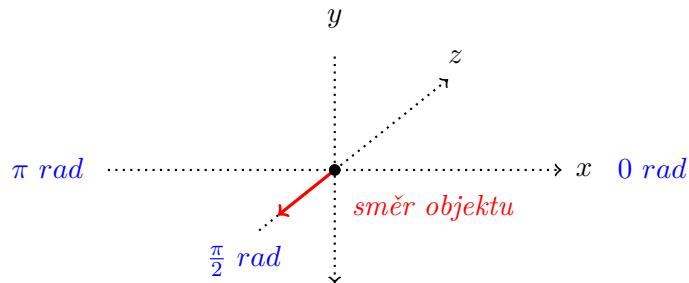
Jak je ilustrováno na obrázku 3.4, krajní body ohraničení objektu se nemusejí vždy nacházet uvnitř obrazu kamery. Poměrnou část objektu nacházející se v hranicích obrazu lze určit jako podíl průniku plochy hraničního obdélníka s plochou obrazu a obsahu celého hraničního obdélníka. Část nacházející se mimo hranice obrazu (v KITTI formátu označována názvem *truncated*) je určena jako zbytek po odečtení tohoto podílu od hodnoty 1.



Obrázek 3.4: Objekt částečně mimo hranice obrazu

### Rotace, observační úhel

Rotace objektu představuje v KITTI anotaci jeho otočení kolem osy  $y$  v kamerových souřadnicích nabývající hodnot  $[-\pi, \pi]$  v radiánech. Směr osy  $x$  představuje nulový úhel, kladná odchylka je pak dosažena rotací po směru hodinových ručiček okolo osy  $y$ . Objekt směřující čelně proti kameře by tak měl úhel rotace  $+\frac{\pi}{2} \text{ rad}$  (vyobrazeno na obrázku 3.5).



Obrázek 3.5: Rotace objektu při průmětu roviny os  $x$  a  $z$  do svislého směru

Z nyní již známého úhlu rotace a souřadnic pozice pozorovaného objektu je nakonec určen ještě observační úhel, v KITTI formátu označován jako *alpha*, který nabývá opět hodnot v intervalu  $[-\pi, \pi]$  v radiánech.

### 3.3.3 Kamerová a lidarová data, kalibrační matice

Kromě souborů s anotací detekovaných objektů zahrnuje datová sada KITTI další tři typy souborů:

- obraz RGB kamery (formát *.png*)
- lidarové mračno bodů (formát *.bin*)
- soubor s kalibračními maticemi senzorů (formát *.txt*)

Výstupy RGB kamery jsou za pomocí standardních funkcí jazyka Python ukládány v podobě téměř identické k té, v níž byly ze senzoru odebrány, tedy jako barevný obraz v požadované velikosti (výchozími hodnotami jsou šířka M pixelů a výška N pixelů).

## Lidarová data

Data lidaru je po jejich načtení ze senzoru nutno nejprve rozdělit na souřadnice jednotlivých bodů mračna (každé 4 po sobě jdoucí hodnoty), následuje pak jejich uspořádání do matice tvaru  $(N, 4)$ , kde  $N$  značí počet bodů.

V rámci předzpracování jsou lidarová data poté uvedena do tvaru, kdy je níže uvedenou transformací možné získat jejich pozici v obrazu (tedy převést bod do pixelových souřadnic). V rovnici jsou využita označení matic užívaná ve formátu datové sady KITTI –  $P_0$  značí projekční matici kamery (v syntetické datové sadě je užita pouze jedna),  $R$  značí rektifikační matici též kamery a  $TR\_velo\_to\_cam$  označuje matici převodu lidarových souřadnic na kamerové.

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = P_0 \cdot R \cdot TR\_velo\_to\_cam \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (3.1)$$

## Kalibrační matice

Formát souborů s kalibračními maticemi, definovaný pro podsadu KITTI určenou pro 3D detekci objektů, je uveden v tabulce 2.2.

Ukládání kalibračních matic je v této práci zjednodušeno, jelikož na rozdíl od originální datové sady KITTI využívá ke snímání obrazu pouze jednu kameru namísto čtyř. Potřebná není ani matice pro převod dat senzoru náklonu do souřadnic lidaru, jelikož tento senzor není v práci využit. Nezbytné je však korektní ukládání projekční matice kamery a matice převodu lidarových souřadnic na kamerové.

Projekční matice kamery  $K$  má tvar:

$$K = \begin{pmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{pmatrix} \quad (3.2)$$

Hodnoty  $u$ ,  $v$  představují souřadnice středu obrazu. Je-li šířka obrazu označena  $w$ , výška obrazu označena  $h$  a zorný úhel kamery ve stupních označen  $fov$ , lze výpočet jednotlivých proměnných zapsat následovně:

$$f = \frac{w}{2 \cdot \tan \frac{fov \cdot \pi}{360^\circ}} \quad (3.3)$$

$$u = \frac{1}{2} \cdot w \quad (3.4)$$

$$v = \frac{1}{2} \cdot h \quad (3.5)$$

Úlohou projekční matice je převod trojrozměrného vektoru bodu v souřadnicích kamery (v rovnících značeny dolním indexem  $k$ ) na vektor pixelových souřadnic (označeny dolním indexem  $p$ ). Po transformaci projekční maticí je nutná normalizace získaného vektoru.

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} \quad (3.6)$$

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} a/c \\ b/c \end{pmatrix} \quad (3.7)$$

Pro převod homogenních souřadnic lidaru do souřadnic kamery (opět značeny dolním indexem  $k$ ) slouží transformace:

$$P_k = TR\_velo\_to\_cam \cdot P_l \quad (3.8)$$

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_l \\ y_l \\ z_l \\ 1 \end{pmatrix} \quad (3.9)$$

### 3.4 Výsledná datová sada

Ke tvorbě datové sady byla využita verze simulátoru CARLA 0.9.13 ve verzi pro operační systémy Microsoft Windows.

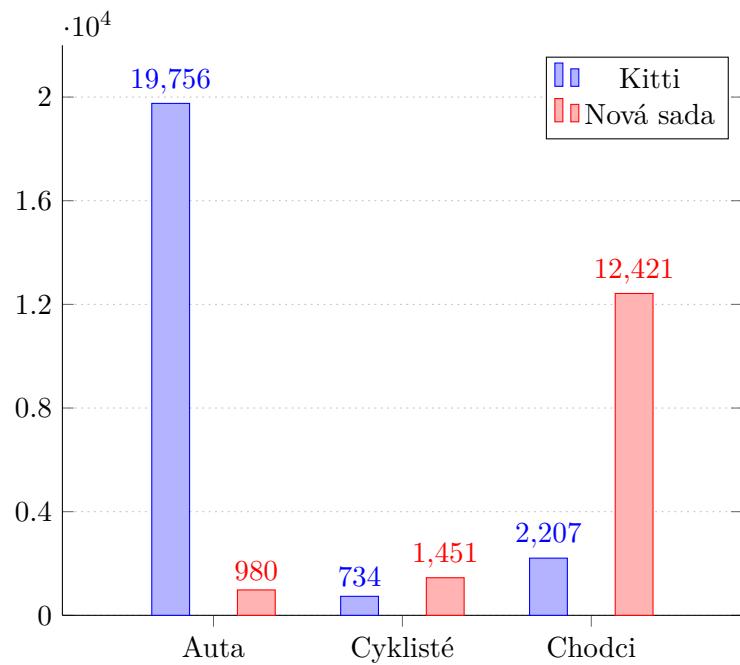
Celkem bylo vytvořeno 3200 vzorků ze čtyř různých map:

- Town02 – menší mapa městského prostředí s převážně jednoduchými křižovatkami ve tvaru písmene T
- Town03 – mapa většího města s víceproudovými cestami, zahrnující kruhový objezd v centru i obchvat po celém obvodu města
- Town04 – mapa většího města ve tvaru čísla 8 v horském prostředí, velkou část této mapy tvoří meziměstské cesty
- Town05 – mapa zahrnuje kromě industriálních budov ve svém centru také víceproudé cesty, velké křižovatky a několik podjezdů

Co se zastoupení jednotlivých tříd účastníků provozu týká, počty jejich výskytů ve vy- tvořené sadě v porovnání s trénovacími vzorky datové sady KITTI jsou zobrazeny v grafu 3.7. Ukázky scén z jednotlivých map simulátoru jsou přiloženy na obrázku 3.6.



Obrázek 3.6: Ukázky vzorků RGB obrazu z různých map simulátoru



Obrázek 3.7: Počet výskytů tří objektů v datových sadách

# Kapitola 4

# Experimenty

Cílem provedených experimentů bylo ověření, zda doplněním datové sady Kittí vzorky získanými pomocí simulátoru Carla dojde ke zlepšení spolehlivosti detekce objektů spadajících do tříd chodců a cyklistů testovacím modelem.

Testování probíhalo v iteracích, přičemž v průběhu každé z nich byly po vytvoření syntetických vzorků dat provedeny experimenty využívající model VoxelNet (viz podkapitola 2.2.2) a validace datové sady za pomoci vizualizačních nástrojů. Jednotlivé experimenty jsou spolu s výsledky dosaženými v poslední iteraci testování blíže popsány v následujících sekcích.

Pro větší přehlednost jsou kromě vlastních výsledků experimentů nejprve prezentovány také referenční výsledky dosažené modelem učeným čistě na vzorcích datové sady Kittí.

## 4.1 Experimenty využívající model VoxelNet

K provedení experimentů vlivu syntetické datové sady na úspěšnost detekce objektů byl zvolen model VoxelNet, který byl představen v podkapitole 2.2.2, respektive jeho volně dostupná neoficiální implementace [11] využívající knihovnu TensorFlow<sup>1</sup>. Parametry jeho učení byly ponechány výchozí, s výjimkou koeficientu učení (výchozí hodnotou je 0.001, nahrazena byla hodnotou 0.005) a počtu epoch, které se pro každý experiment liší.

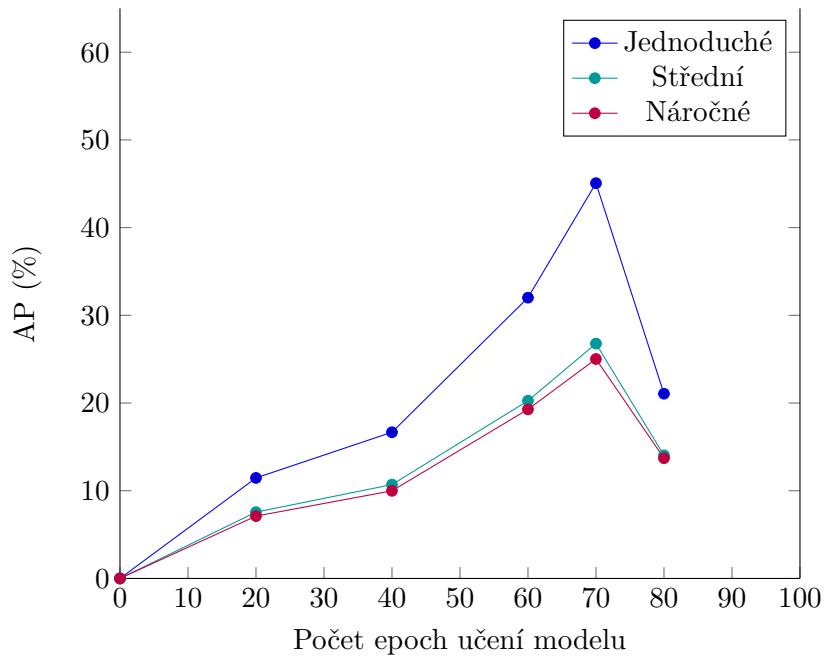
Inspirací pro zvolené experimenty byly především existující práce zabývající se využitím syntetických dat pro učení modelů detekce objektů [7, 13].

### 4.1.1 Výsledky referenčního modelu

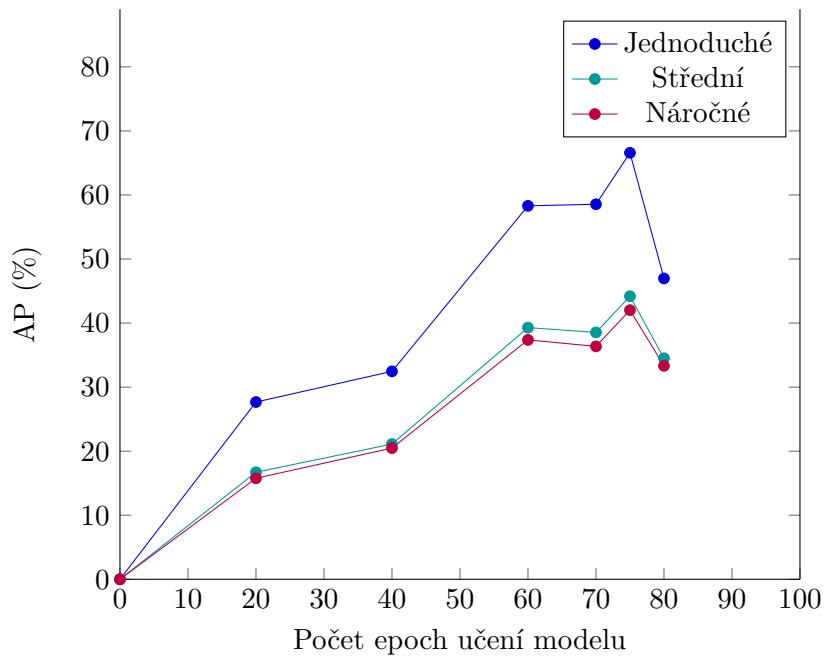
Před samotným prováděním experimentů s nově vytvořenou datovou sadou byly nejprve v průběhu 80 epoch učení modelu monitorovány jeho výsledky detekce cyklistů a chodců na vzorcích všech tří kategorií náročnosti definovaných sadou KITTI (2.2.4).

---

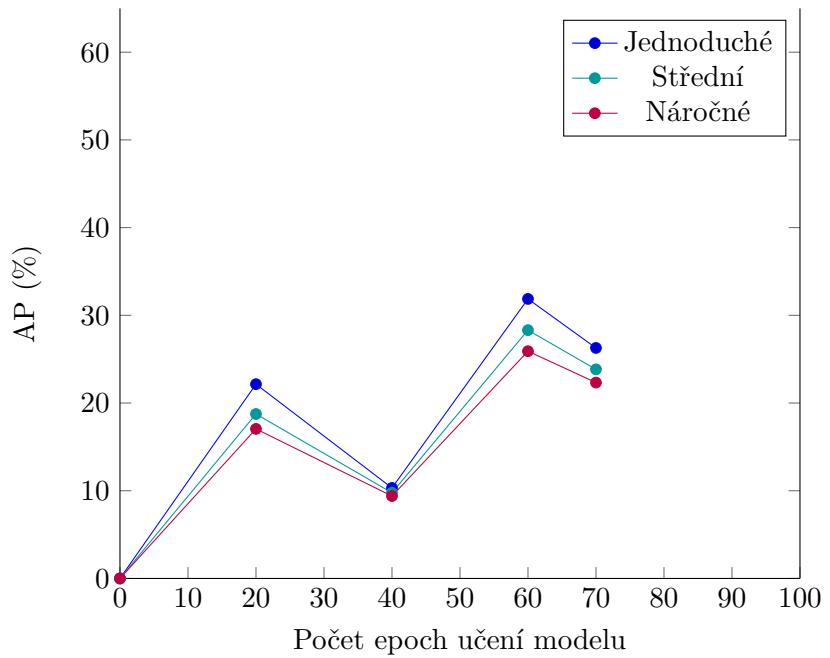
<sup>1</sup><https://www.tensorflow.org/>



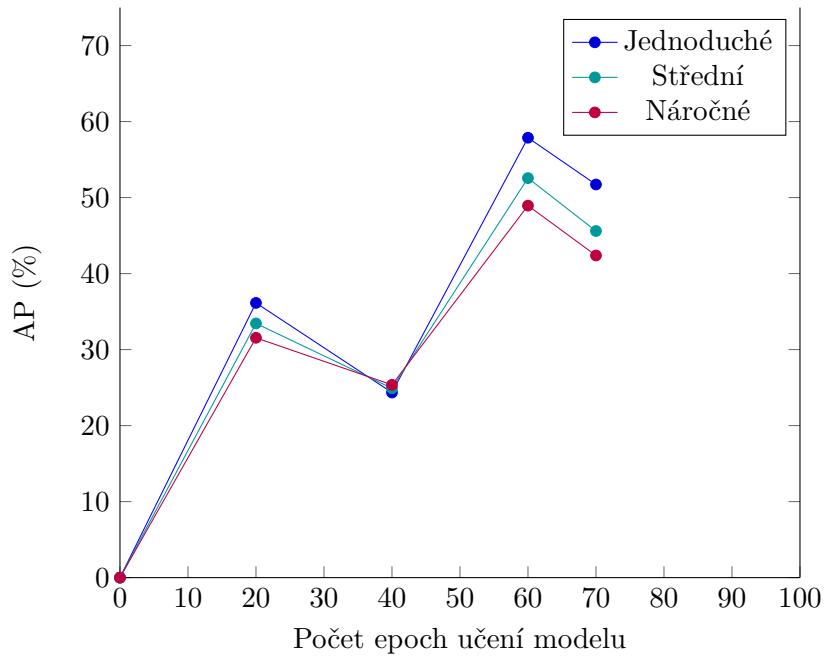
Obrázek 4.1: AP 3D detekce cyklistů referenčním modelem



Obrázek 4.2: AP 2D detekce cyklistů referenčním modelem



Obrázek 4.3: AP 3D detekce chodců referenčním modelem

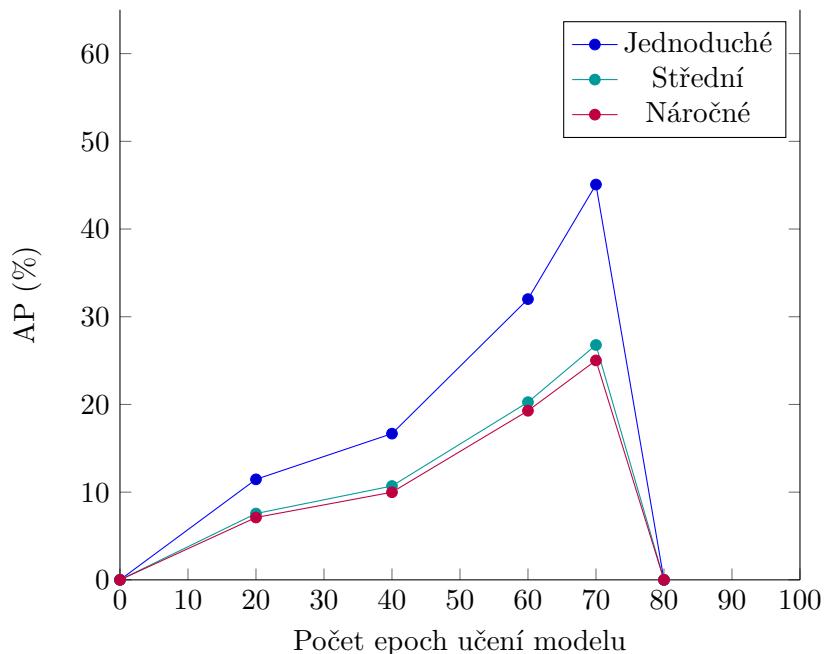


Obrázek 4.4: AP 2D detekce chodců referenčním modelem

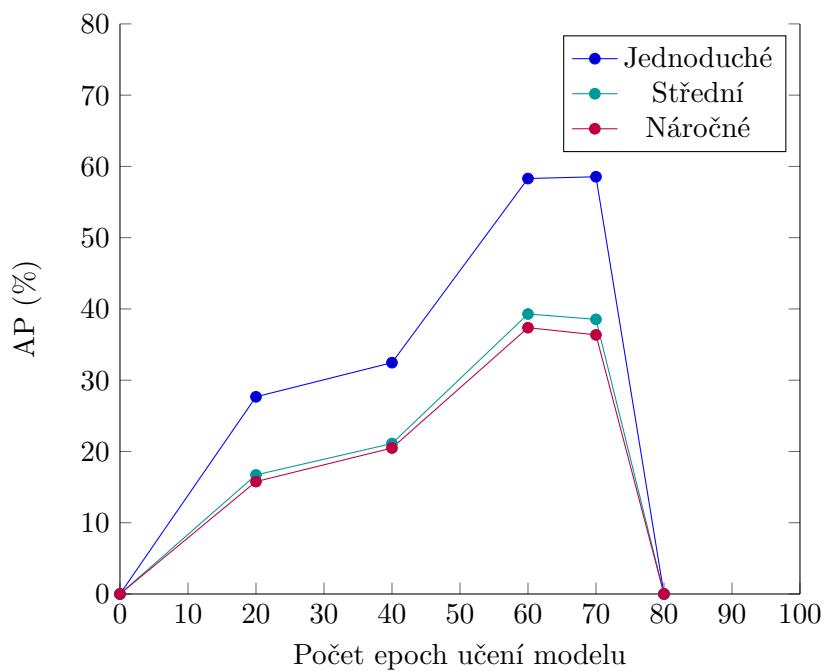
#### 4.1.2 Experiment 1

V rámci prvního experimentu bylo použitím doplňujících vzorků umělé datové sady navázáno na stav modelu po 70 epochách učení na vzorcích sady Kitti, kdy za testovacích podmínek dosahoval nejlepších výsledků prostorové detekce cyklistů.

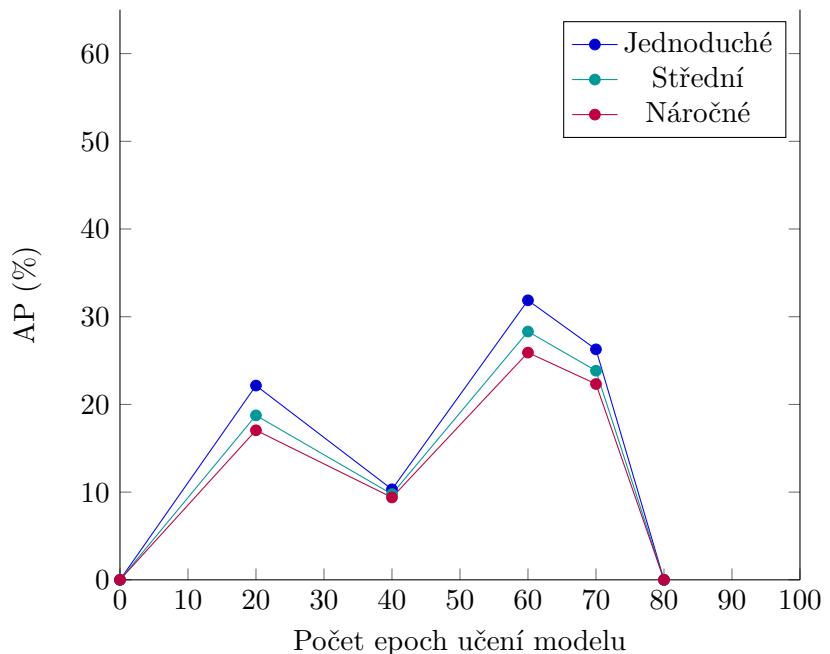
Výsledky takto připraveného modelu při testování detekce cyklistů na validačních vzorcích sady KITTI jsou vyobrazeny na grafech 4.5 a 4.6. Výsledky dosažené při detekci chodců ve stejných validačních vzorcích jsou pak vyobrazeny na grafech 4.7 a 4.8.



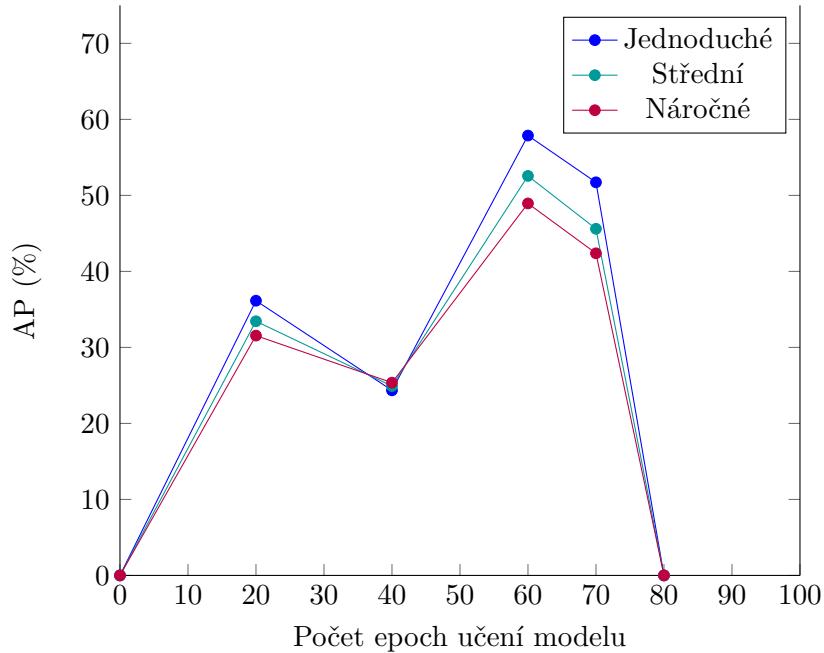
Obrázek 4.5: Průměrná přesnost 3D detekce cyklistů v průběhu experimentu 1



Obrázek 4.6: Průměrná přesnost 2D detekce cyklistů v průběhu experimentu 1



Obrázek 4.7: Průměrná přesnost 3D detekce chodců v průběhu experimentu 1

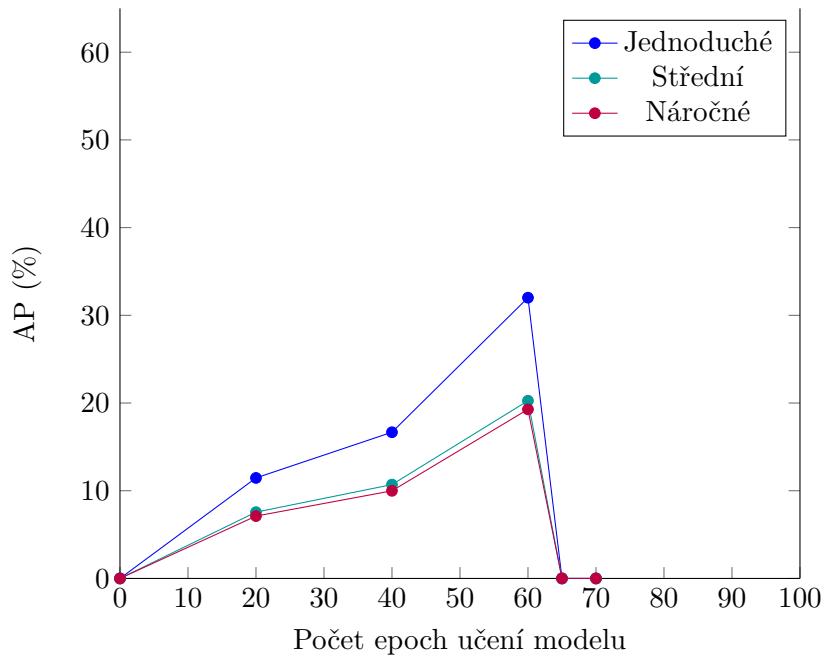


Obrázek 4.8: Průměrná přesnost 2D detekce chodců v průběhu experimentu 1

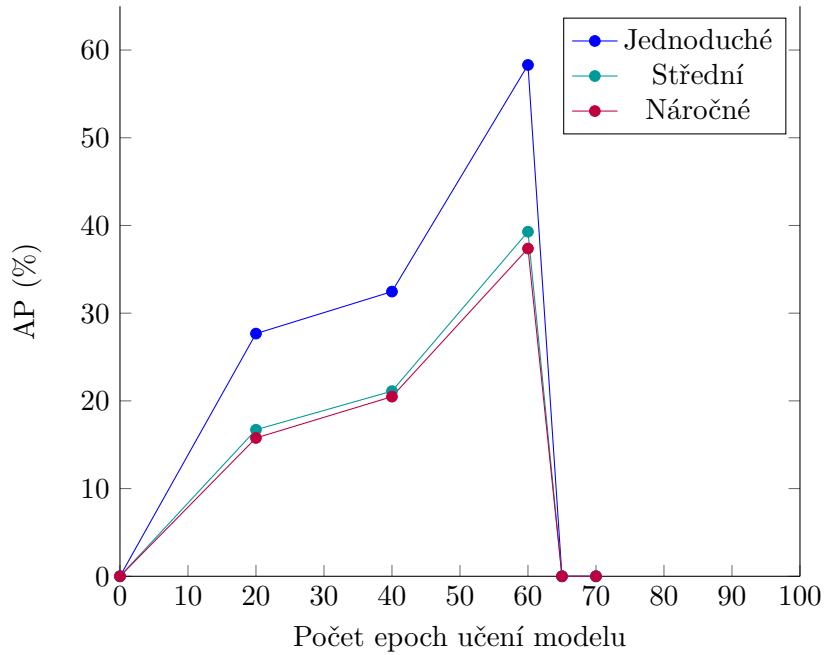
#### 4.1.3 Experiment 2

V rámci druhého experimentu bylo učení modelu na datové sadě Kitti proloženo několika epochami učení na vzorcích doplňující datové sady. První fáze učení na sadě Kitti je tentokrát ukončena v šedesáté epoše, na kterou následně navazuje 5 epoch učení na vzorcích umělé sady, a dalších 5 epoch na sadě Kitti.

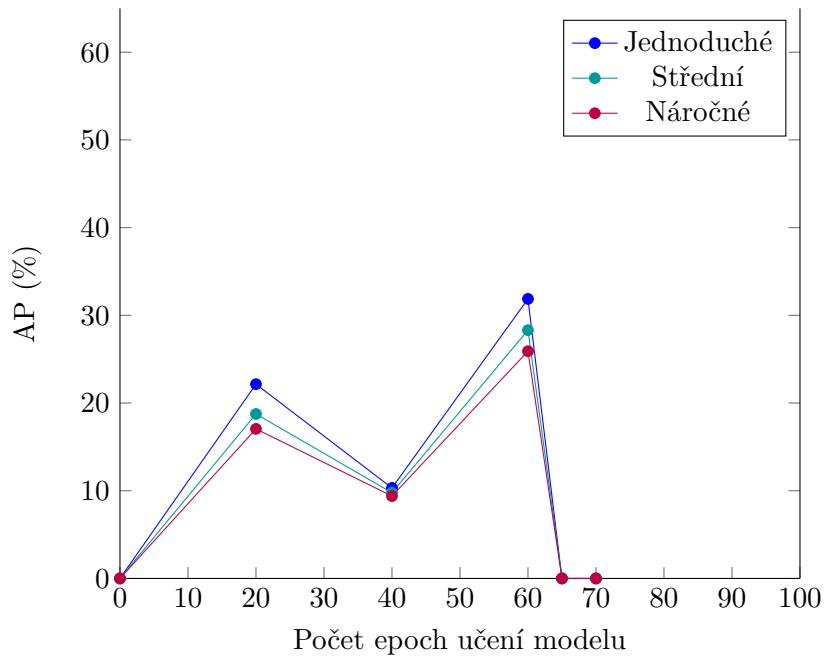
Výsledky takto připraveného modelu při testování detekce cyklistů na validačních vzorcích sady KITTI jsou vyobrazeny na grafech 4.9 a 4.10. Výsledky dosažené při detekci chodců ve stejných validačních vzorcích jsou pak vyobrazeny na grafech 4.11 a 4.12.



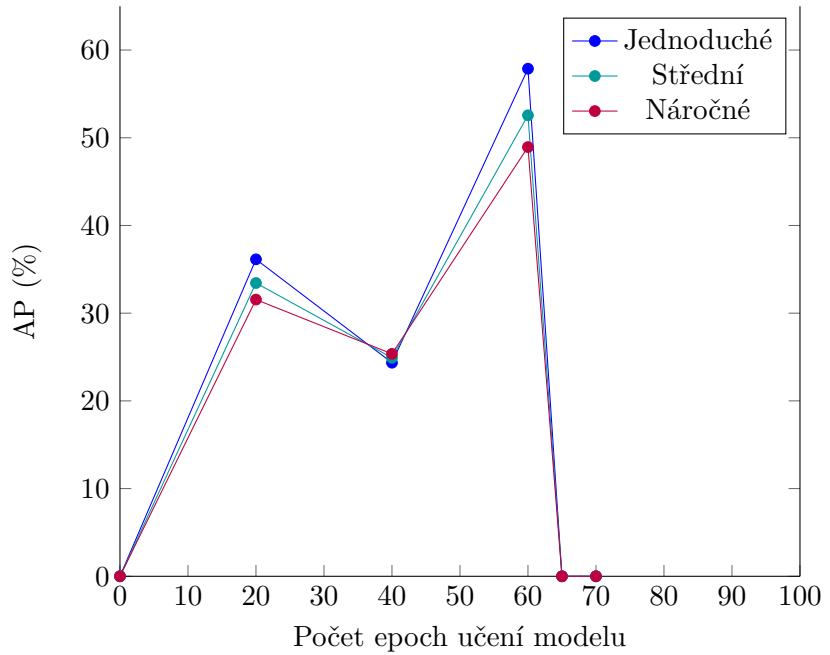
Obrázek 4.9: Průměrná přesnost 3D detekce cyklistů v průběhu experimentu 2



Obrázek 4.10: Průměrná přesnost 2D detekce cyklistů v průběhu experimentu 2



Obrázek 4.11: Průměrná přesnost 3D detekce chodců v průběhu experimentu 2



Obrázek 4.12: Průměrná přesnost 2D detekce chodců v průběhu experimentu 2

#### 4.1.4 Experiment 3

Vzhledem k neúspěšnosti prvních pokusů o zlepšení výsledků modelu VoxelNet pomocí vzorků uměle vytvořené sady byl po několika iteracích testování zaveden také experiment využívající pro učení a testování výsledků modelu pouze syntetická data získaná za pomoci simulátoru CARLA, ani ten však nepřinesl žádné pozitivní výsledky – přesnost detekce hledaných objektů při něm zůstávala nulová.

Model byl v průběhu tohoto experimentu učen pouze 20 epoch, jelikož jeho cílem nebylo dosažení nejlepších možných výsledků detekce, ale získání více informací pro zkoumání příčin neúspěšnosti předchozích experimentů.

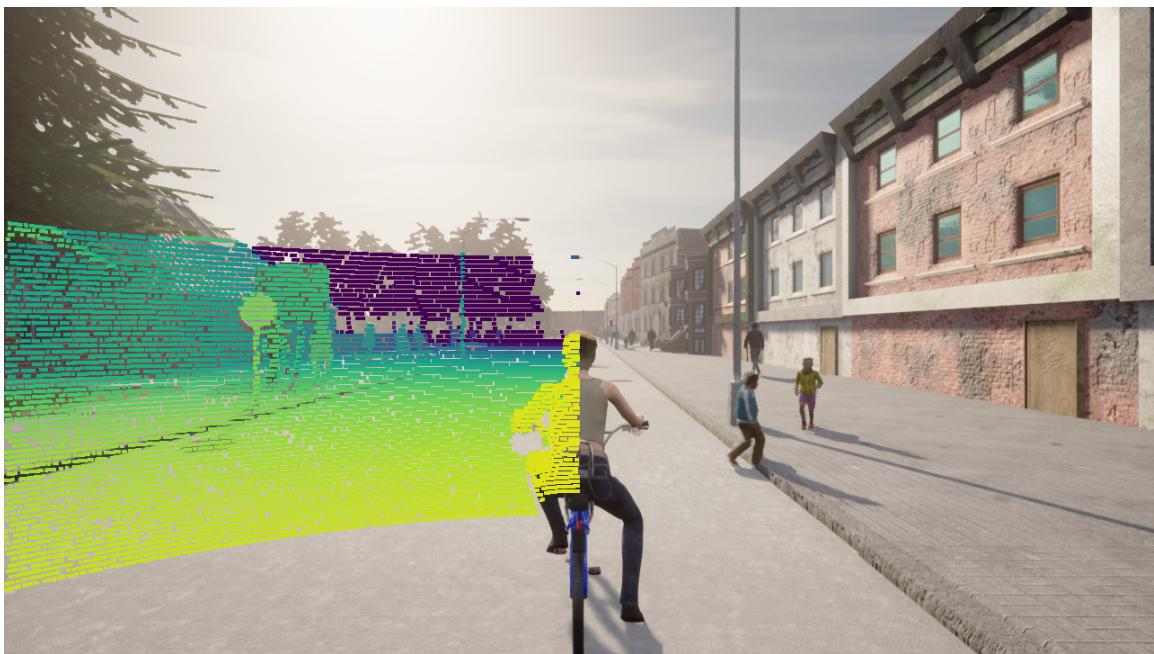
## 4.2 Validace vytvořené datové sady

V průběhu iterací tvorby umělé datové sady bylo v návaznosti na neúspěch experimentů využívajících model VoxelNet opakovaně prováděno testování korespondence mezi obrazovými a lidarovými daty s údaji ukládanými do souborů s popisky a kalibračními maticemi. Tyto testy byly prováděny za pomoci volně dostupného řešení pro vizualizaci vzorků sady v Kitti formátu [29].

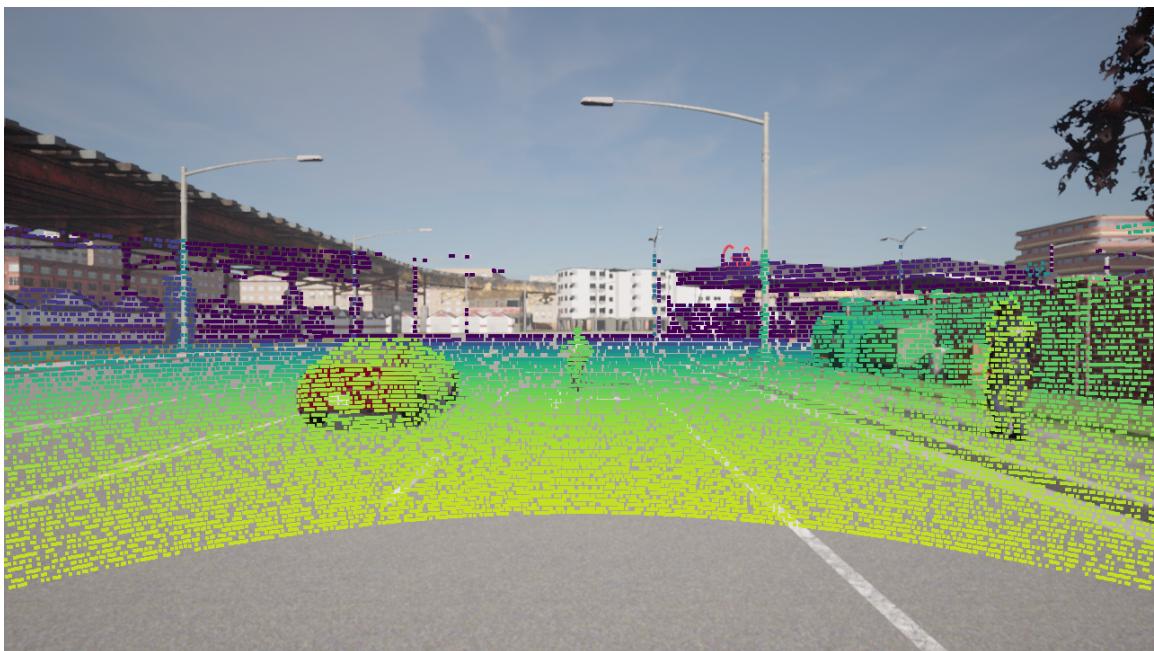
Vizualizací prostorového ohraničení objektů popsaných v anotačních souborech byl zjištěn a opraven chybný výpočet jejich rotace. Vizualizací hraničních obdélníků (2D ohraničení objektu) byly dále zjištěny nedostatky metody určování výšky chodců a cyklistů, které byly následně také opraveny.

Největším problémem, který se díky vizualizaci datové sady podařilo zachytit, byla nevhodně zvolená kombinace časového kroku simulátoru a rotační frekvence simulovaného lidarového senzoru. Původní kombinace, kdy časový krok  $\delta$  byl nastaven na 0.5 vteřiny a rotační frekvence lidaru na 10 Hz (odpovídá frekvenci rotace senzoru v datové sadě KITTI), měla za následek pokrytí pouze poloviny monitorovaného prostoru exportovaným mračnem bodů (obrázek 4.13). Tento problém byl vyřešen navýšením rotační frekvence na 20 Hz (obrázek 4.14).

Známým problémem vytvořené sady, který nebylo možné jednoduše vyřešit, jsou nepresná ohraničení chodců v případech, kdy jsou výrazně rozkročeni či při chůzi výrazně pohybují pažemi. Důvodem tohoto nedostatku je způsob získávání údajů o rozměrech chodce ze serveru simulátoru (dostupné údaje vychází ze statické stojící pozice šablony chodce, není tedy možno zjistit aktuální pozici chodce, ani polohu jeho končetin vůči středu těla).

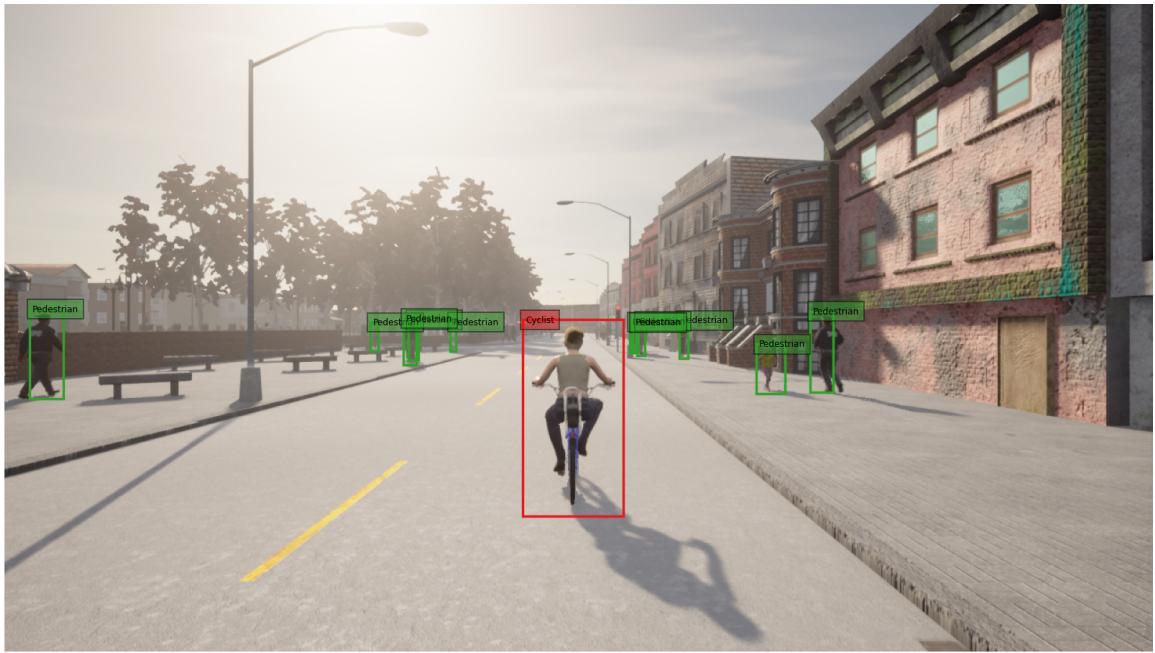


Obrázek 4.13: Rotační frekvence 10 Hz, vygenerováno vlastním skriptem



Obrázek 4.14: Rotační frekvence 20 Hz, vygenerováno vlastním skriptem

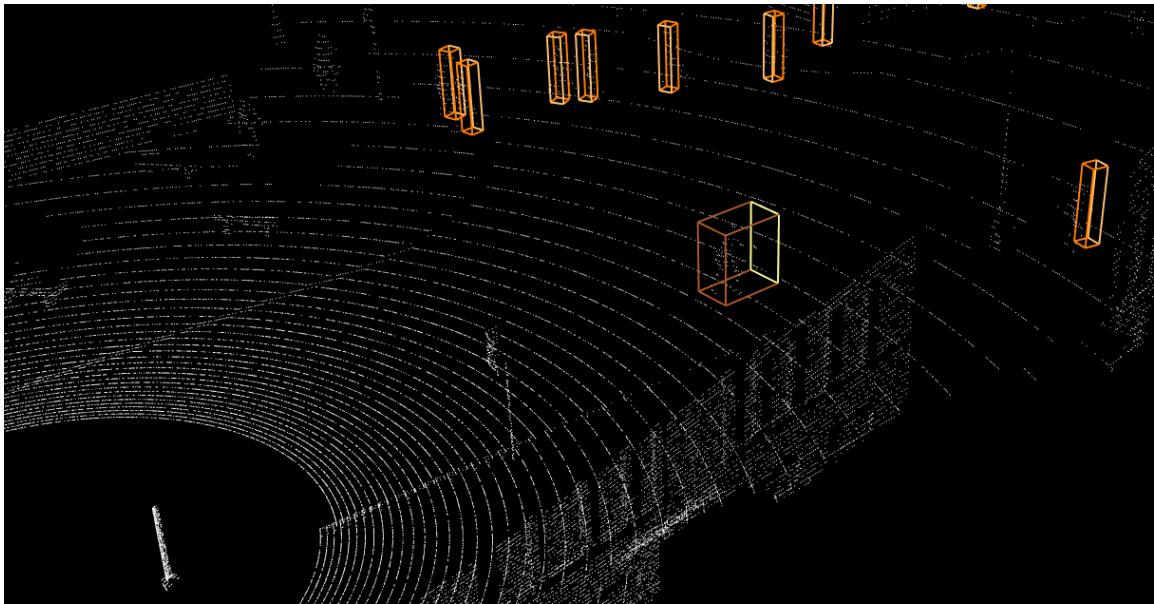
Ukázka průmětů ohraničujících obdélníků do obrazu kamery a ohraničujících kvádrů do obrazových a lidarových dat vytvořené sady je přiložena v uvedeném pořadí na obrázcích 4.15, 4.16 a 4.17.



Obrázek 4.15: Průměr ohraničujících obdélníků do obrazu simulátoru, vygenerováno za pomoci [29]



Obrázek 4.16: Průměr ohraničujících kvádrů do obrazu simulátoru, vygenerováno za pomocí [29]



Obrázek 4.17: Průmět ohraničujících kvádrů do syntetického mračna bodů, vygenerováno za pomocí [29]

### 4.3 Zhodnocení výsledků

Ačkoliv validace vzniklé syntetické sady pomocí vizualizačních skriptů potvrdila, že vzorky posuzované v poslední iteraci testování dodržují KITTI formát kalibračních i anotačních souborů a že s jejich využitím lze korektně promítnout rovinné i prostorové ohraničení objektů do vstupních dat, výsledky dosažené v rámci experimentů s modelem VoxelNet nelze zdaleka považovat za uspokojivé, ani neodpovídají výsledkům dosaženým v existující práci využívající tento model k experimentům s umělou datovou sadou [7].

Důvodů zde může být hned několik. První nabízející se možností je neodhalení další chyby či chyb ve vytvořené sadě. Zde by bylo možné manuálně projít a vizualizovat větší množství vytvořených vzorků a ověřit, zda se v nich nenachází nekonzistence.

Další možnou příčinou může být nevhodně zvolená konkrétní implementace modelu VoxelNet. Z důvodu ověření tohoto podezření byla využitá implementace již v průběhu experimentů důkladně prozkoumána a byly v ní nalezeny konstatní hodnoty kalibračních matic senzorů datové sady KITTI, namísto jejich načítání z kalibračních souborů. Tyto hodnoty byly tedy nahrazeny maticemi senzorů ze simulátoru, na výsledku experimentů se tato změna však neprojevila.

Při prozkoumání dalších neoficiálních implementací modelu navíc byly objeveny tytéž pevně zakódované hodnoty kalibračních matic a dalších údajů. Tuto skutečnost však nelze zcela považovat za chybu autorů implementací, jelikož i práce popisující vznik a princip samotného oficiálního modelu VoxelNet [26] zmiňuje jeho testování specificky na datové sadě KITTI.

# Kapitola 5

## Závěr

Zadáním práce bylo nejprve se seznámit se způsoby rozpoznávání objektů a s dostupnými implementacemi této úlohy pomocí neuronových sítí a následně navrhnut a vytvořit řešení pro vytvoření syntetické datové sady za účelem vylepšení výsledků detekce cyklistů a chodců dosahovaných existujícími modely.

V rámci zpracování tématu byly nejprve prostudovány zdroje porovnávající různé modely na bázi neuronových sítí využívané k detekci objektů v okolí autonomního vozidla a také několik studií zabývajících se využitím syntetických dat pro učení těchto modelů.

Na základě získaných poznatků byl pro tvorbu umělé sady zvolen simulátor autonomního řízení CARLA, čemuž následovalo seznámení se s jeho prostředím a nástroji. Implementace řešení pro tvorbu datové sady zahrnovala vytvoření několika skriptů v jazyce Python, které ze simulátoru získávají a v korektní formě ukládají data senzorů a informace nezbytné pro jejich využití.

Po dokončení jeho implementace prošlo řešení více iteracemi testování, během kterých byl posuzován vliv využití vytvořených vzorků na výsledky detekce cyklistů a chodců modelem VoxelNet a kontrolována správnost těchto datových vzorků. V posledním cyklu experimentů bylo vytvořeno a otestováno 3200 vzorků umělé sady, které jsou k práci přiloženy.

Výsledky dosažené při experimentech jednoznačně vypovídají o tom, že využití vytvořené sady má na přesnost detekce objektů zvoleným modelem negativní vliv. V návaznosti na toto zjištění byly identifikovány jeho možné příčiny.

Na práci by bylo v budoucnu jistě vhodné navázat dalšími experimenty, ať už zkoumajícími vzorky sady, nebo ověřujícími dopady jejího využití pro jiný model detekce objektů.

Samotný proces tvorby umělé datové sady by jistě bylo možné dále vylepsit například implementací přesnější metody pro určení míry překrytí popisovaných objektů, nebo přidáním filtru, který by kompletně překryté objekty ze seznamu pro anotaci ihned odebíral.

# Literatura

- [1] ADIB, M. *CARLA Vehicle 2D Bounding Box Annotation Module* [online]. [cit. 2023-04-15]. Dostupné z: <https://mukhlasadib.github.io/CARLA-2DBBox/>.
- [2] BEHERE, S. a TÖRNGREN, M. A Functional Architecture for Autonomous Driving. In: *Proceedings of the First International Workshop on Automotive Software Architecture*. New York, NY, USA: Association for Computing Machinery, 2015, s. 3–10. WASA ’15. DOI: 10.1145/2752489.2752491. ISBN 9781450334440. Dostupné z: <https://doi.org/10.1145/2752489.2752491>.
- [3] CAESAR, H., BANKITI, V., LANG, A. H., VORA, S., LIONG, V. E. et al. NuScenes: A Multimodal Dataset for Autonomous Driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [4] DABRAL, S., KAMATH, S., APPIA, V., MODY, M., ZHANG, B. et al. Trends in camera based Automotive Driver Assistance Systems (ADAS). In: *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2014, s. 1110–1115. DOI: 10.1109/MWSCAS.2014.6908613.
- [5] DESCHAUD, J. KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator. *CoRR*. 2021, abs/2109.00892. Dostupné z: <https://arxiv.org/abs/2109.00892>.
- [6] DOSOVITSKIY, A., ROS, G., CODEVILLA, F., LOPEZ, A. a KOLTUN, V. CARLA: An Open Urban Driving Simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, s. 1–16.
- [7] DWORAK, D., CIEPIELA, F., DERBISZ, J., IZZAT, I., KOMORKIEWICZ, M. et al. Performance of LiDAR object detection deep learning architectures based on artificially generated point cloud data from CARLA simulator. In: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2019, s. 600–605. DOI: 10.1109/MMAR.2019.8864642.
- [8] GEIGER, A., LENZ, P., STILLER, C. a URTASUN, R. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*. Sage Publications Sage UK: London, England. 2013, sv. 32, č. 11, s. 1231–1237.
- [9] GEIGER, A., LENZ, P. a URTASUN, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

- [10] GUO, Y., WANG, H., HU, Q., LIU, H., LIU, L. et al. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021, sv. 43, č. 12, s. 4338–4364. DOI: 10.1109/TPAMI.2020.3005434.
- [11] HUANG, Q. *Voxelnet* [<https://github.com/qianguih/voxelnet>]. GitHub, 2018.
- [12] HUANG, Y. a CHEN, Y. Autonomous driving with deep learning: A survey of state-of-art technologies. *ArXiv preprint arXiv:2006.06091*. 2020.
- [13] JANG, J., LEE, H. a KIM, J.-C. CarFree: Hassle-Free Object Detection Dataset Generation Using Carla Autonomous Driving Simulator. *Applied Sciences*. 2022, sv. 12, č. 1. DOI: 10.3390/app12010281. ISSN 2076-3417. Dostupné z: <https://www.mdpi.com/2076-3417/12/1/281>.
- [14] KHATAB, E., ONSY, A., VARLEY, M. a ABOUELFARAG, A. Vulnerable objects detection for autonomous driving: A review. *Integration*. Elsevier. 2021, sv. 78, s. 36–48.
- [15] MARTINEZ, M., SITAWARIN, C., FINCH, K., MEINCKE, L., YABLONSKI, A. et al. *Beyond Grand Theft Auto V for Training, Testing and Enhancing Deep Learning in Self Driving Cars*. 2017.
- [16] MÁČALA, S. *Historický vývoj a moderní trendy bezpečnostních prvků osobních automobilů*. Brno, 2011. Bakalářská práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav automobilního a dopravního inženýrství. Dostupné z: <http://hdl.handle.net/11012/6246>.
- [17] PADILLA, R., NETTO, S. L. a SILVA, E. A. B. da. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, s. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.
- [18] PANAGAKIS, Y., KOSSAIFI, J., CHRYSOS, G. G., OLDFIELD, J., NICOLAOU, M. A. et al. Tensor Methods in Computer Vision and Deep Learning. *Proceedings of the IEEE*. 2021, sv. 109, č. 5, s. 863–890. DOI: 10.1109/JPROC.2021.3074329.
- [19] QIAN, R., LAI, X. a LI, X. 3D Object Detection for Autonomous Driving: A Survey. *Pattern Recognition*. 2022, sv. 130, s. 108796. DOI: <https://doi.org/10.1016/j.patcog.2022.108796>. ISSN 0031-3203. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0031320322002771>.
- [20] SERBAN, A. C., POLL, E. a VISSER, J. A Standard Driven Software Architecture for Fully Autonomous Vehicles. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 2018, s. 120–127. DOI: 10.1109/ICSA-C.2018.00040.
- [21] SUN, P., KRETZSCHMAR, H., DOTIWALLA, X., CHOUARD, A., PATNAIK, V. et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [22] SUN, X. Space-based lidar systems. In: *2012 Conference on Lasers and Electro-Optics (CLEO)*. 2012, s. 1–2.

- [23] VELODYNE LIDAR, I. *Velodyne Lidar* [online]. Velodyne Lidar, Inc., 2023 [cit. 2023-04-24]. Dostupné z: <https://velodynelidar.com/>.
- [24] WANG, Y. a YE, J. An Overview Of 3D Object Detection. *CoRR*. 2020, abs/2010.15614. Dostupné z: <https://arxiv.org/abs/2010.15614>.
- [25] ZAMANAKOS, G., TSOCHATZIDIS, L., AMANATIADIS, A. a PRATIKAKIS, I. A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving. *Computers & Graphics*. 2021, sv. 99, s. 153–181. DOI: <https://doi.org/10.1016/j.cag.2021.07.003>. ISSN 0097-8493. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0097849321001321>.
- [26] ZHOU, Y. a TUZEL, O. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017.
- [27] ZIEBINSKI, A., CUPEK, R., GRZECHCZA, D. a CHRUSZCZYK, L. Review of advanced driver assistance systems (ADAS). *AIP Conference Proceedings*. Listopad 2017, sv. 1906, č. 1. DOI: 10.1063/1.5012394. ISSN 0094-243X. 120002. Dostupné z: <https://doi.org/10.1063/1.5012394>.
- [28] ZOU, Z., CHEN, K., SHI, Z., GUO, Y. a YE, J. Object detection in 20 years: A survey. *Proceedings of the IEEE*. IEEE. 2023.
- [29] ZZZXXXTTT. *Simple Kitti visualization scripts* [[https://github.com/zzzxxxttt/simple\\_kitti\\_visualization](https://github.com/zzzxxxttt/simple_kitti_visualization)]. GitHub, 2020.

## Příloha A

# Obsah přiloženého paměťového média

Soubory uložené na přiloženém paměťovém médiu jsou organizovány dle níže uvedené struktury. Dvě hlavní složky souborů jsou **scripts** obsahující skripty v jazyce Python použité pro tvorbu datové sady a **dataset**, v níž je uložena samotná datová sada. Ve složce **text\_src** je přiložen zdrojový kód technické zprávy.

```
./
└── dataset
    ├── scripts
    │   ├── generator_actors.py
    │   ├── generator_bbox.py
    │   ├── generator_kitti.py
    │   ├── generator_labels.py
    │   ├── generator_utils.py
    │   └── README
    ├── text_src
    ├── poster.pdf
    ├── README
    └── technical_report.pdf
```

Vzhledem ke členění jeho obsahu jsou na médiu přiloženy dva různé README soubory. První z nich slouží k obecné orientaci v přiložených souborech, druhý popisuje konkrétní skripty řešení a prostředí, v němž byly spouštěny. Výňatky kódu přejaté z existujících volně dostupných řešení jsou ve skriptech odpovídajícím způsobem vyznačeny.

# Příloha B

## Plakát



Autor: **Zuzana Kopčilová**  
Vedoucí práce: **doc. RNDr. PAVEL SMRŽ, Ph.D.**  
Akademický rok: 2022/2023

### Využití syntetických dat pro zlepšení detekce cyklistů a chodců v autonomním řízení

Motivace	Schéma řešení
<p>Mezi největší nástrahy autonomního řízení patří korektní rozpoznaní okolních zranitelných účastníků provozu. Tvorba kvalitních datových sad je však časově i finančně náročná, což vede ke zkoumání možnosti využití umělých dat.</p>	
Využité nástroje	
<p>Pro tvorbu umělé datové sady byl využit open-source simulátor autonomního řízení CARLA. Práce se simulátorem probíhá za pomocí skriptů v jazyce Python využívajících API simulátoru v tomto jazyce.</p>	
Vytvořená datová sada	

