VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Autonomní řízení vozidel

Autonomous driving for vehicles

Bc. Martin Krybus

Diplomová práce

Vedoucí práce: doc. Ing. Jan Platoš, Ph.D.

Ostrava, 2021

Abstrakt

Autonomní řízení vozidel má potenciál zvýšit účinnost dopravního systému tím, že zcela eliminuje lidský faktor. Autonomní řízení snižuje ekologickou stopu šetrnějším chováním k životnímu prostředí v podobě nižších emisí, kromě toho dělá dopravu dostupnější zejména pro osoby, které kvůli zdravotním omezením nemohou řídit vozidlo sami. Cílem práce je vytvořit simulátor automobilu pro zkoušení algoritmů reinforcement learningu s aplikací v autonomním řízení. Konkrétně v práci zkouším využít Q-Learning algoritmus v kombinaci s hlubokou neuronovou Q sítí k učení ovládání vozidla z jeho vlastních zkušeností získaných pomocí pokusů a omylů. Prováděny byly dva experimenty. První zkouší vozidlo naučit vozidlo závodnímu stylu jízdy a druhý naopak plynulému a bezpečnému stylu jízdy. Problém učení řízení byl rozdělen do dvou fází, přičemž každá fáze využívala svoji vlastní instanci konvoluční neuronové sítě a byly trénovány zvlášť ve zvoleném pořadí. Veškerý trénink byl realizován v procedurálně generovaném prostředí s vozidlem vsazeným do něj. Vozidlo s uživatelsky definovatelnými fyzikálními vlastnostmi a parametry umožňuje zkoušet algoritmus s různými typy vozidel. Vstupem modelů jsou snímky pořízené přední kamerou, která zachycuje bezprostřední okolí vozidla a simuluje výhled řidiče. Výstupem práce je systém pro simulaci jízdy různé složitosti, jenž je schopen sdílet informace o situaci kolem vozidla a zároveň umožní toto vozidlo řídit pomocí algoritmu reinforcement learningu nazývaného Q-Learning.

Klíčová slova

Hluboké Q Sítě, Q-Learning, Reinforcement learning, simulátor, procedurálně generované trasy

Abstract

Autonomous driving has the potential to increase the efficiency of the transport system by completely eliminating the human factor. Autonomous driving reduces the ecological footprint through more environmentally friendly behavior in the form of lower emissions, in addition to making transport more accessible, especially for people who cannot drive alone due to health restrictions. The aim of the work is to create a car simulator for testing reinforcement learning algorithms with an application in autonomous control. Specifically, in my work I try to use the Q-Learning algorithm in combination with a deep neural Q network to learn to control the vehicle from his own experience gained through trial and error. Two experiments were performed. The first tests the vehicle to teach the vehicle a racing driving style and the second, on the contrary, a smooth and safe driving

style. The control learning problem was divided into two phases, each phase using its own instance of a convolutional neural network and being trained separately in the chosen order. All training was carried out in a procedurally generated environment with the vehicle inserted into it. A vehicle with user-definable physical properties and parameters allows the algorithm to be tested with different types of vehicles. The input of the models are images taken by a front camera, which captures the immediate surroundings of the vehicle and simulates the driver's view. The output of the work is a system for simulating driving of various complexity, which is able to share information about the situation around the vehicle and at the same time allow to control this vehicle using a reinforcement learning algorithm called Q-Learning.

Keywords

Autonomous driving, Deep Q networks, Q-Learning, Reinforcement learning, simulator, procedurally generated tracks

Poděkování

Velmi rád bych na tomto místě poděkoval svému vedoucímu práce doc. Ing. Janu Platošovi, Ph.D. za odbornou pomoc při zpracování práce, cenné rady, připomínky a trpělivost při vedení práce. Velké poděkování patří také mé rodině a kamarádům za podporu při studiu.

Obsah

Seznam použitých zkratek a symbolů

SAE	–	Society of Automotive Engineers
GM	–	General Motors
RL	–	Reinforcement Learning
MDP	–	Markov Decision Process
DQN	–	Deep Q Network
MSE	–	Mean Squared Error
CUDA	–	Compute Unified Device Architecture
CPU	–	Central Processing Unit
RAM	–	Random Access Memory
GPU	–	Graphics Processing Unit

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Jízda automobilem po silnici je pro člověka relativně jednoduchý úkol, nicméně sebemenší drobná chyba při řízení vede obvykle k fatálním následkům, proto je vývoj pokročilých asistenčních systémů jízdy od počátku digitalizace automobilu neustálým procesem. V dnešní době automobily obsahují systém pro regulaci rychlosti, automatické parkování, systém hlídání mrtvého úhlu, systémy pro předcházení kolizím, systém pokročilé navigace, monitorovací systém řidiče, rozpoznávání dopravních značek a mnoho dalšího. Systémy zpracovávají informace ze senzorů a na základě nich konají svá rozhodnutí. Tyto typy senzorů často generují velké množství dat. Populárním přístupem řešení problémů s velkým množstvím dat je ve strojovém učení hluboké učení, které je založené na umělých neuronových sítích. Během posledních několika let využití neuronové sítě prudce vzrostlo, protože došlo k velkému pokroku ve vývoji výpočetní kapacity grafických procesorů. Vlivem modernizace světa, větších nároků na pohodlí, snížení ekologické stopy a minimalizace nehodovosti je autonomní řízení v posledních letech čím dál větší oblastí výzkumu a umělá inteligence ve formě neuronových sítí má velký potenciál tento problém efektivně řešit.

Organizace SAE International klasifikuje automatizované systémy řízení do 5 různých úrovní. V úrovních 0 - 2 se od řidiče očekává, že bude sledovat aktuální dění kolem vozidla a bude vždy připraven převzít kontrolu. Úrovně 3 – 5 jsou systémy, které mohou v omezených podmínkách a v závislosti na prostředí řídit samostatně bez dozoru.

Uplatnění vysoce automatizovaných systémů řízení ve vozidlech může mít mnoho výhod. Očekává se, že vyřazením lidského faktoru z řízení povede k méně smrtelným nehodám a lepší efektivnosti v přetíženém provozu. Může také poskytnout mobilitu a svobodu těm, kteří vozidla řídit nemohou. Navíc by lidé mohli strávený čas cestováním v automobilech využít produktivněji.

V první části diplomové práce je popsáno autonomní řízení, stupně autonomního řízení a metody využívané v autonomním řízení. V druhé části popisuji reinforcement learning a jeho základní principy včetně Markova rozhodovacího procesu, nezbytných výpočetních vztahů a rovnic a hluboké neuronové Q sítě. Třetí část se zabývá implementací algoritmů, architekturám neuronových sítí,

fyzikálnímu modelu vozidla a procesu učení. V závěru práce se věnuji experimentálnímu ověření algoritmů a jejich porovnání.

Kapitola 2

Autonomní řízení

Moderní automobily v dnešní době disponují několika senzory. Mezi tyto senzory patří radar, lidar, sonar, GPS a inerciální měřicí jednotky. Pokročilé řídící systémy interpretují informace ze senzorů o aktuální dopravní situaci, aby identifikovaly vhodné navigační cesty, překážky a příslušné značení. Získané informace slouží pro systémy pokročilé asistence řidiče, které poskytují důležité bezpečnostní funkce:

- varování před kolizí
- pomoc s řízením
- automatické brzdění

Vozidla s autonomním řízením tyto technologie posouvají na další úroveň tím, že zcela eliminují potřebu řidiče. Senzory automaticky snímají silniční provoz a systémy vyhodnocují dopravní situace za řidiče. Vozidlo je tedy schopno vnímat prostředí a navigovat se k cíli bez zásahu člověka, přičemž musí zajišťovat všechny bezpečnostní funkce.

Autonomní řízení přináší několik výhod pro společnost. Zejména zvyšuje účinnost dopravního systému díky plynulejší jízdě autonomních vozidel v konvoji, kde se vozidla pohybují těsně za sebou, díky čemuž se zvýší dopravní kapacita, průjezdnost a sníží dopravní kongesce. Dalším benefitem autonomního řízení je zvýšení bezpečnosti v dopravě, protože drtivá většina nehod je způsobená chybou řidiče. Nemalé přínosy přináší autonomní řízení v podobě snížení emisí a dalších dopadů na životní prostředí, kromě toho slibuje i zlepšení dostupnosti dopravy a služeb mobility zejména u osob, které ze zdravotních důvodů či jiných omezení nemohou řídit vozidlo.

2.1 Stupně automatizovaného řízení

Organizace SAE International (Society of Automotive Engineers) definovala 5 stupňů automatizovaného řízení počínaje stupněm 0.

2.1.1 Úroveň 0: Bez automatizace

V úrovni 0 systém nemá kontrolu nad vozidlem, ale poskytuje řidiči pouze základní informace o dopravní situaci. Řidič má nad vozidlem plnou kontrolu a vše ovládá sám. Typickým příkladem může být akustické či vizuální upozornění na vozidla nacházející se v mrtvém úhlu nebo upozornění při poklesu teploty na možnou námrazu na vozovce. Do této úrovně patří drtivá většina dnešních vozidel na silnici.

2.1.2 Úroveň 1: Asistence řidiče

Úroveň 1 znamená, že systém dokáže provádět úkony spojené s příčným pohybem nebo s podélným pohybem vozidla. Jinými slovy systém může mírně zasahovat do řízení na základě aktuální jízdní situace, konkrétně zrychlovat, zpomalovat nebo lehce zatáčet, ovšem vozidlo může vykonávat vždy jen jednu funkci, nikoli je kombinovat. Řidič sleduje aktuální dopravní situaci a v případě potřeby okamžitě přebírá kontrolu. Typickým příkladem je adaptivní automat udržující rychlost a zároveň hlídající bezpečný odstup od vozidla jedoucího vpřed, parkovací asistent s automatickým otáčením kol, asistent pro udržování vozidla v jízdním pruhu nebo asistent zabraňující kolizím.

2.1.3 Úroveň 2: Částečná automatizace

V úrovni 2 může systém navíc kombinovat jednotlivé úkony definované na úrovni 1. Automatizovaný systém může samostatně zrychlovat, brzdit i zatáčet. Řidič sleduje a kontroluje automatizované úkony a je připraven v případě potřeby okamžitě převzít kontrolu nad vozidlem. Typickým příkladem je systém automatického parkování nebo automatického brždění v případě možné kolize.

2.1.4 Úroveň 3: Podmíněná automatizace

Na úrovni 3 systém může přebrat za určitých podmínek plnou kontrolu nad vozidlem bez nutnosti vizuální a fyzické kontroly řidičem, nicméně řidič stále musí být připraven na upozornění systému okamžitě převzít řízení. Příkladem může být jízda po dálnici s dobře vyznačenými jízdními pruhy. Systém automaticky brzdí, zrychluje a dokonce se vyhýbá vozidlům.

2.1.5 Úroveň 4: Vysoká automatizace

V této úrovni jsou veškeré úkony prováděny automatizovaně a není nutný žádný zásah člověkem. Systém dokonce dokáže bezpečně zastavit a zaparkovat v případě, že řidiče vyzve k převzetí řízení, ale ten nereaguje. Autonomní jízda funguje jen ve vymezených oblastech nebo za zvláštních okolností. Mimo tyto oblasti a okolnosti musí být vozidlo schopné bezpečně přerušit jízdu.

2.1.6 Úroveň 5: Plná automatizace

Vozidla této úrovně jsou plně automatizovaná a není potřeba žádný lidský zásah do řízení bez vyjímky. Člověk zadá jednoduše jen cílovou destinaci a vozidlo se postará o vše ostatní. [1]

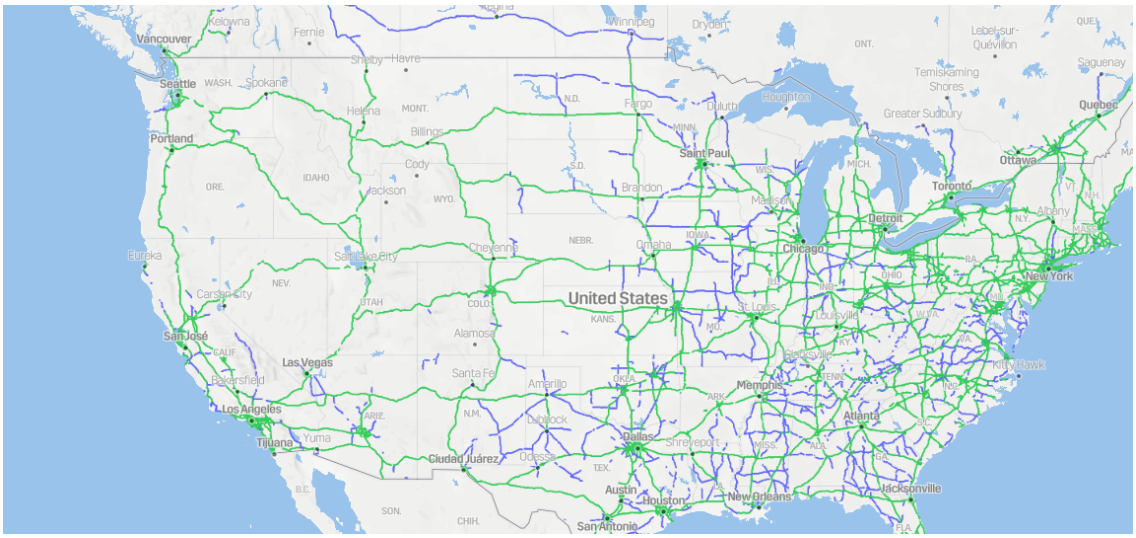
2.2 Metody využívané v autonomním řízení

Autonomní řízení je složitý úkol. Existují tři hlavní přístupy k řešení problému autonomního řízení:

- předpočítaná 3D mapa ve vysokém rozlišení
- zpracování obrazu + strojové učení
- přizpůsobení prostředí automobilu

2.2.1 Předpočítané mapy

Tato vozidla závisí na předem zaznamenané 3D mapě s vysokým rozlišením okolí, která je předem zachycena pomocí vozidel vybavených lidarem. Vozidla využívají přesné určování polohy v reálném čase, kamery, senzory a mapová data (zachycené lidarem), které pomáhají detekovat každou křivku a pomáhají dělat dlouhé jízdy a dojíždění pohodlnější. Vozidlo pak může použít mapu, zjistit zda se prostředí změnilo pomocí vlastního lidaru a poté převzít kontrolu při jízdě v mapované oblasti. V případě přiblížení se k nezmapované oblasti, vozidlo automaticky pomocí akustických a vizuálních výstrah upozorní řidiče, aby převzal kontrolu, v opačném případě vůz bezpečně zastaví. Spolehlivé využití autonomního řízení vyžaduje širší spolupráce mezi obcemi a výrobci automobilů. Mapy je potřeba vytvořit a udržovat aktuální. Tato metoda nabízí předvídatelnost a vysokou spolehlivost řízení, ale přichází s vyššími náklady potřebnými k záznamu map a výrobě vozidel vybavených lidarem. Tento přístup využívá například v současné době Cadillac Super Cruise od amerického automobilového výrobce General Motors. Jejich Cadillac Super Cruise je schopný vozidlo navigovat pouze po dálnicích, které byly zmapovány a prošly kontrolou konzistence a bezpečnosti (viz obrázek 2.1).



Obrázek 2.1: Trasy kompatibilní s GM Super Cruise

Na obrázku 2.1 jsou vyznačeny zmapované trasy modrou a zelenou barvou kompatibilní s modelem Super Cruise s celkovou délkou přes 320 000 kilometrů. Mimo automobilku GM na tento přístup vsadili i automobilky Mercedes Benz a Ford.

2.2.2 Realtime analýza prostředí

Spíše než spoléhat na předem zaznamenané mapy, je možné zvolit kombinaci zpracování obrazu a strojového učení tak, aby každé vozidlo vědělo o svém okolí co nejvíce informací v reálném čase. Jelikož vozidlo vyžaduje aktuální znalost prostředí, může se přizpůsobit měnícímu se stavu dopravní situace a dynamicky na ní reagovat. Nehrozí jim tedy riziko spoléhání na mapu, která možná obsahuje zastaralé informace. Neuronové sítě analyzují nezpracované snímky z jednotlivých kamer a provádějí sémantickou segmentaci, detekci objektů a odhad hloubky, kromě toho navíc pořizují video a vytvářejí rozložení silnic, statickou infrastrukturu a 3D objekty přímo v pohledu shora dolů. Neuronové sítě se učí z nejsložitějších a nejrůznějších scénářů na světě pocházejících z vozidel v reálném čase. Všechny získané informace a znalosti mezi sebou vozidla sdílejí. Tento druh zpracování v reálném čase však vozu přidává spoustu složitostí. Na přístup analýzy prostředí v reálnem čase například vsadil automobilový výrobce Tesla. Jejich Tesla Autopilot využívá 48 neuronových sítí, jejichž trénink trvá zhruba 70 000 hodin GPU a v každém časovém kroku predikují přes 1000 odlišných predikcí.

2.2.3 Adaptace prostředí a pozemní komunikace

Další přístup v autonomii řízení se tolik nezaměřuje na to, aby automobily byly inteligentnější a přizpůsobovaly se jejich prostředí, ale na vytváření intuitivního a lehce pochopitelného prostředí. Tím se snižuje komplexita vozidel a režie spojená s vývojem automobilu, nicméně vozidla by stejně

měla být schopná rozpoznat chodce, cyklisty, jiná auta, křižovatky, přednosti v jízdě, zaparkovaná vozidla a změny jízdního pruhu v pohybujícím se provozu. V tomto scénáři by prostředí upozornilo vozidlo na měnící se okolí a s větší přesností informovalo o okolních podmínkách (například kužely, které mohou říci vozu, kde přesně je oblast konstrukce a kde se nachází dočasné pruhy). Tento přístup není omezen na komunikaci s infrastrukturou. Auta mohou také komunikovat mezi sebou a informovat ostatní auta o důležitých událostech, jako jsou nehody a objekty na vozovce, kromě toho město má možnost posílat do vozidel informace a data o mimořádných situacích. Tento přístup řešení autonomního řešení zvolila například automobilka Volkswagen. [2]

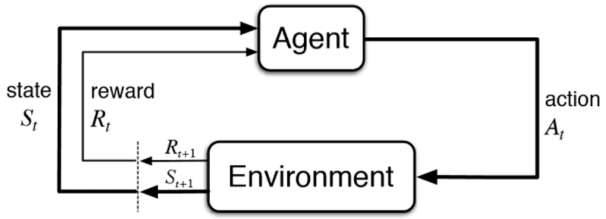
Kapitola 3

Reinforcement learning

Reinforcement learning je technika strojového učení, která se zaměřuje na učení algoritmu(funkce) metodou pokusů a omylů. Agent vyhodnotí aktuální situaci (stav), provede akci a po každém činu obdrží zpětnou vazbu (odměnu) z prostředí. Pozitivní zpětná vazba je odměna a negativní zpětná vazba je trest za chybu.

3.1 Markovův rozhodovací proces MDP

Prostředí ve kterém se agent vyskytuje lze formálně popsat jako Markovův rozhodovací proces (MDP). MDP je uspořádaná čtveřice $(S, A, P_a(s, s'), R_a(s, s'))$, kde S je konečná množina stavů, A je konečná množina akcí, $P_a(s, s')$ je pravděpodobnost, že akce a ve stavu s v čase t povede v čase $t + 1$ do stavu s' a $R_a(s, s')$ je okamžitá odměna dosažená po přechodu stavu na s' ze stavu s s pravděpodobností přechodu $P_a(s, s')$. Markovův rozhodovací proces demostruje obrázek 3.1.



Obrázek 3.1: Markovův rozhodovací proces

Problém MDP se skládá z funkce odměny R a funkce přechodu stavu P. V případě, že funkce R a P nejsou známy, Q-Learning algoritmus lze použít k řešení problému MDP s neznámými funkcemi odměny a přechodu. [3]

3.2 Q-Learning

Q-learning je posilovací algoritmus učení. Agent (umělá inteligence) se snaží najít nejlepší kroky, které je třeba provést vzhledem k současnému stavu tak, aby maximalizoval svoji celkovou odměnu. Za každý krok může být agent penalizován nebo odměněn. Pomocí tohoto systému odměn najde Q-Learning optimální chování ve smyslu maximalizace odměny. Model Q-Learning problému je dán stavy s souborem akcí (kroků) a. Provedením akce může agent přecházet z jednoho stavu do druhého. Cílem agenta je maximalizovat jeho celkovou odměnu za jednotlivé kroky tak, že se učí, která akce je optimální pro daný stav. [4]

3.3 Iterace Q-Hodnot

Algoritmus Q-Learning iterativně aktualizuje Q-Hodnoty pro každý pár stav-akce pomocí Bellmanovy rovnice, dokud Q-Funkce nebude konvergovat na optimální Q-Funkci. Tento přístup se nazývá iterace Q-Hodnot.

3.4 Q-Tabulka

K ukládání Q-Hodnot pro každý pár stav-akce slouží Q-Tabulka tvořená kombinací stavu a akce:

$$Q : s \times a \Longrightarrow R$$

Vodorovná osa tabulky představuje akce a svislá osa představuje stavy. Protože agent nemá žádné informace o tom jak prostředí vypadá a nemá žádnou informaci o očekávaných odměnách pro páry stav-akce, jsou všechny Q-Hodnoty v tabulce inicializovány na nulu. Agent postupně spustí několik epizod a vypočte nové Q-Hodnoty pro dvojice stav-akce, které agent nově objevil a použije je k aktualizaci Q-Hodnot uložených v Q-Tabulce. Agent později může hledat v aktualizovaných Q-Hodnotách a vytvořit další akci založenou na nejvyšší Q-Hodnotě pro aktuální stav.

3.5 Epizody

V každé epizodě agent začíná výběrem akce z počátečního stavu na základě aktuálních Q-Hodnot v Q-Tabulce. Agent vybírá akci na základě akce s nejvyšší Q-Hodnotu v Q-Tabulce pro aktuální stav. Na začátku tréninku jsou Q-Hodnoty nastaveny na nulu a agent nedokáže rozlišovat mezi nimi, nelze tedy rozhodnout, která z nich je lepší. Epsilon Greedy strategie dokáže tenhle problém vyřešit.

3.6 Epsilon Greedy strategie

Agent interaguje s prostředím dvěma způsoby. První je použití Q-Tabulky jako reference a zobrazení všech možných akcí pro daný stav. Agent poté vybere akci na základě maximální hodnoty těchto akcí. Jinými slovy využíváme to, co se už umělá inteligence naučila. Druhým způsobem jak jednat, je jednat náhodně. Tomu se říká průzkum. Průzkum zkoumá a získává informace o prostředí, ve kterém se agent pohybuje, proto místo výběru akcí na základě maximální budoucí odměny, vybírá akci náhodně. Náhodné chování je důležité, protože umožňuje agentovi prozkoumat a objevit nové stavy, které by jinak nemohly být vybrány v případě využití Q-Hodnoty ze získaných zkušeností. Rovnováha mezi průzkumem a využitím dosavadně naučených schopností se řídí Epsilon Greedy strategií. [5]

Epsilon Greedy strategie definuje míru průzkumu *epsilon* a udává s jakou pravděpodobností agent prozkoumá prostředí namísto využití Q-Hodnot z Q-Tabulky. Agent začíná objevováním prostředí ($epsilon = 1$). Postupem času je hodnota *epsilon* snižována a agent začíná využívat první dovednosti získané z předchozího zkoumání prostředí. Náhodně vygenerované číslo mezi 1 a 0 v každém časovém kroku rozhodne o tom, zda si agent zvolí průzkum nebo využije dosavadně získané zkušenosti. Pokud je toto náhodné číslo větší než *epsilon*, pak agent vybere svou další akci prostřednictvím exploatace, neboli vybere akci s nejvyšší Q-Hodnotou na základě aktuálního stavu z Q-tabulky, v opačném případě bude jeho další akce vybrána náhodně. [6]

3.7 Aktualizace Q-Hodnoty

K aktualizaci Q-Hodnoty pro akci převzatou z předchozího stavu použijeme následující Bellmanovu rovnici.

$$q*(s,a) = E[R_{\mathrm{t+1}} + gamma * max(q*(s_{\mathrm{i}}, a_{\mathrm{i}}))]$$

Myšlenkou je přiblížit Q-Hodnotu pro daný pár stav-akce co nejblíže pravé straně Bellmanovy rovnice, aby se Q-Hodnota nakonec přibližně rovnala optimální Q-Hodnotě $q*$. K tomu dojde v průběhu času iterativním porovnáním ztráty mezi Q-Hodnotou a optimální Q-Hodnotou pro daný pár stav-akce a následnou aktualizací Q-Hodnoty znovu a znovu pokaždé, když se setkáme se stejným párem stav-akce za účelem snížení chyby.

3.8 Rychlost učení

Rychlost učení neboli learning rate lr je číslo mezi 0 a 1, které definuje do jaké míry je nová Q-Hodnota oproti staré Q-Hodnotě přepsána pro daný pár stav-akce. Předpokládejme například, že máme v Q-Tabulce Q-Hodnotu pro nějaký pár stav-akce objeveného v předchozím časovém kroku. Pokud agent narazí na stejný pár stav-akce v pozdějším kroku a bude mít k dispozici více informací

o prostředí, bude potřeba Q-hodnotu aktualizovat tak, aby odrážela změnu očekávání pro budoucí odměny. Účelem rychlosti učení není přepsat starou Q-Hodnotu, ale spíše kontrolovat kolik informací o vypočítané Q-Hodnotě z minulosti pro daný pár stav-akce uchováváme oproti nové Q-Hodnotě vypočítané pro stejný pár stav-akce v pozdějším časovém kroku. Jinými slovy, čím vyšší je rychlost učení, tím rychleji agent příjme novou hodnotu Q. Například pokud je rychlost učení 1, odhad Q-Hodnoty pro daný pár stav-akce bude právě nově vypočítá Q-Hodnota a nebude brát v úvahu předchozí Q-Hodnoty, které byly pro daný pár stav-akce vypočítány v minulosti.

3.9 Výpočet nové Q-Hodnoty

Nové Q-Hodnoty pro dvojice stav-akce jsou vypočteny podle následující rovnice.

$$q_{\text{new}}(s,a) = (1-lr) * q_{\text{old}}(s,a) + lr * (R_{\text{t+1}} + gamma * max(q(s_{\text{i}}, a_{\text{i}})))$$

Parametr *gamma* definuje význam budoucích odměn. Parametr R je odměna získaná po dokončení určitého kroku v daném stavu. Nová Q-Hodnota se tedy rovná váženému součtu staré Q-Hodnoty a nové Q-Hodnoty vynásobeného $(1 - lr)$. Nově vypočítaná Q-Hodnota je uložena v Q-Tabulce a stejný proces se opakuje pro každý časový krok až do ukončení epizody.

3.10 Maximální počet kroků

Maximální počet kroků udává kolik kroků může agent maximálně podniknout, než se epizoda automaticky ukončí. Epizodu lze ukončit i před využitím všech kroků, které má agent k dispozici. Ukončení epizody v tomto případě znamená dosažení určitého koncového bodu agentem. Typicky to může být dosažení konce nějaké hry, dokončení požadovaného cíle atd. Maximální počet kroků se liší v závislosti na povaze a složitosti problému, jenž agent řeší.

3.11 Experience replay buffer

Simulace vozidla je výpočetně náročná. Výpočetní náročnost simulace lze snížit učením ze zkušeností, které agent získal v minulosti. Jednotlivé reakce agenta na prostředí jsou uloženy v datové struktuře, do které lze přistoupit s nižší výpočetní náročností. Velikost datové struktury umožňuje uložit více kroků, nicméně také zvyšuje paměťovou složitost algoritmu. Jednotlivé reakce na prostředí jsou definovány uspořádanou čtveřic (s_t, a_t, r_t, s_{t+1}), kde s_t je stav v kroku t, a_t je zvolená akce, r_t je odměna a s_{t+1} je nově dosažený stav. U některých algoritmů je důležité mít zřetězená data, která jsou oddělena jako časové sekvence. Replay buffer je v tomto případě rozšířen tak, aby umožňoval ukládat sekvence uspořádaných trojic $(s_{t+n}, a_{t+n}, r_{t+n})$, kde n je poloha čtveřice v časové

posloupnosti. Uložená data jsou náhodně navzorkována a následně použity k tréninku neuronové sítě.

Hlavním důvodem pro použití Replay bufferu je zmírnit korelace mezi po sobě jdoucími vzorky. Trénink nejlépe funguje s nezávislými a identicky distribuovanými vzorky, nicméně použitím Reinforcement learningu dostáváme sekvenční vzorky z interakcí s prostředím, tak jak se postupně vyskytly. To způsobí, že do neuronová sítě vstupuje příliš mnoho vzorků jednoho druhu a síť má tendenci zapomenou ty ostatní. Například agent by mohl dosáhnout druhé úrovně hry, která vypadá nebo se chová úplně jinak a zapomenout tak, jak hrát první úroveň. Ukládání veškerých zkušeností do replay bufferu umožňuje trénovat na méně závislých vzorcích. [7]

3.12 Funkce odměny

Funkce odměny implicitně určuje cíl optimalizace a je důležitou součástí, protože prostřednictvím funkce odměny je popsáno optimální chování. Je definována jako mapování párů stav-akce na reálné číslo. Funkce je někdy dána přirozeně (například peněžní náklady nebo energetická účinnost), ale v některých oblastech to není triviální vybrat tu správnou funkci vhodnou pro úspěšné učení. Správná volba vhodné funkce odměny během návrhu MDP pro problém v reálném životě je známým problémem v RL. Návrhář má obvykle představu o tom, jak by výsledná strategie měla vypadat a konstruuje funkci odměny, aby zdůraznil konkrétní cíle (například dosáhnout nějakého cíle a zůstat v bezpečí).

Funkci odměny lze občas snadno navrhnout ručně nebo je dána (například v počítačové hře nebo soutěži). V některých případech je však těžké najít takovou funkci. Funkce odměny, která dává pro člověka smysl, není vždy nejlepší pro výcvik agentů v RL. Funkce odměny může být v jejích limitních případech řídká (sparse), např. jeden signál po dokončení epizody nebo naopak hustá (dense). Funkce husté odměny odměňuje agenta za průběžné výsledky. Například to mohou být pozitivní signály pro dokončení podúrovně ve víceúrovňové hře. Funkce řídkých i hustých odměn jsou naučitelné. To znamená, že se agent může úspěšně naučit hodnotovou funkci pomocí daných odměn. Někdy je však funkce odměny příliš řídká a agent může dosáhnout lokálního maxima a navždy tam uvíznout. Taková situace nastává, když je obtížné dosáhnout stavu s velkou odměnou.

3.13 Mezní situace

Q-Learning je užitečný algoritmus, který přináší robustní řešení jednodušších problémů, nicméně jeho výkon značně klesá s velikostí prostředí. Větší velikost stavového prostoru, znamená více času potřebného k projití všech těchto stavů a aktualizaci Q-Hodnot. V případě reprezentace stavu sadou pixelů s několika akcemi z každého stavu se stává algoritmus výpočetně neefektivní a pravděpodobně ve většině případů neproveditelný kvůli výpočetním zdrojům a času, který to může trvat. V případě složitějšího prostředí, namísto použití iterace Q-Hodnot a nalezení optimální funkce, je časově vý-

hodné využít Q-Learning v kombinaci s hlubokou neuronovou sítí k aproximaci optimální funkce. Hluboká neuronová síť slouží k odhadu Q-Hodnot pro každý pár stav-akce v daném prostředí a pro aproximaci optimální funkce. Akt kombinování Q-Learning s hlubokou neuronovou sítí se nazývá Deep Q-Learning a hluboká neuronová síť, která odhaduje Q-Funkci se nazývá Deep Q-Network neboli DQN.

3.14 Deep Q-Networks

Deep Q-Networks neboli hluboké neuronové Q sítě využívají neuronovou síť k aproximaci Q-Funkce místo výpočetně náročné iterace všech Q-Hodnot a řeší tak problém časové neefektivnosti ve velkém stavovém prostoru v případě varianty Q-Learningu bez neuronové sítě. Hluboká neuronová Q síť přijímá jako vstup stavy z daného prostředí a pro každý daný stavový vstup síť vygeneruje odhad Q-Hodnot pro každou akci na základě tohoto stavu. Cílem této sítě je aproximovat optimální Q-Funkci s respektem k Bellmanové rovnici. S ohledem na tuto skutečnost se chyba (loss) sítě počítá porovnáním výstupních Q-Hodnot s Q-Hodnotami z pravé strany Bellmanovy rovnice. Cílem jakékoli neuronové sítě je minimalizovat tuto chybu. Po výpočtu chyby se váhy v síti aktualizují pomocí optimalizátoru a zpětné propagace stejně jako u jakékoli jiné typické sítě. Tento proces se provádí znovu a znovu pro každý stav v prostředí, dokud dostatečně neminimalizujeme chybu a nezískáme optimální Q-Funkci.

3.14.1 Vstupní data

V případě složitějšího prostředí například videohra, jsou typicky vstupem do neuronové sítě statické snímky zachycující stavy z komplexního prostředí. Standardní předzpracování provedené na snímcích obvykle zahrnuje převod dat RGB do dat ve stupních šedi, protože barva v obrázku pravděpodobně neovlivní stav prostředí. Kromě toho předzpracování vstupních snímků často zahrnuje i oříznutí estetických a méně důležitých informací z snímků včetně změny měřítka. Předzpracování optimalizuje a zrychluje celý proces díky menší velikosti vstupů. Obvykle se používá několik po sobě jdoucích snímků spojených do jednoho zachycující čtyři po sobě jdoucí kroky z videohry v pořadí, ve kterém se ve vyskytly. Jediný snímek obvykle nebude stačit pro síť nebo dokonce pro naše lidské mozky, abychom plně pochopili stav prostředí. Například pouhým pohledem na jediný snímek nemůžeme zjistit, jakým směrem a rychlostí se objekt pohybuje. Podíváme-li se však na čtyři po sobě jdoucí snímky, máme mnohem lepší představu o současném stavu prostředí.

3.14.2 Vrstvy

Hluboké Q sítě typicky obsahují několik za sebou naskládaných konvolučních vrstev následovaných nějakými nelineárními aktivačními funkcemi. Na konvoluční sítě jsou napojeny plně propojené sítě.

Na výstupní vrstvu navazuje lineární aktivační funkce, která zachová výstupní Q-Hodnoty v původním netransformovaném surovém formátu.

3.14.3 Výstup

Výstupní Q-Hodnoty pro každou akci, kterou lze provést z daného stavu, generuje plně propojená výstupní vrstva. Výstupní Q-Hodnoty generuje výstupní vrstva ve formě vektoru, kde pořadí jednotlivých složek vektoru představují Q-Hodnoty jednotlivých akcí z vstupního stavu. Na výstupní Q-Hodnoty se neaplikují žádné transformace a zůstávají v původních hodnotách.

Kapitola 4

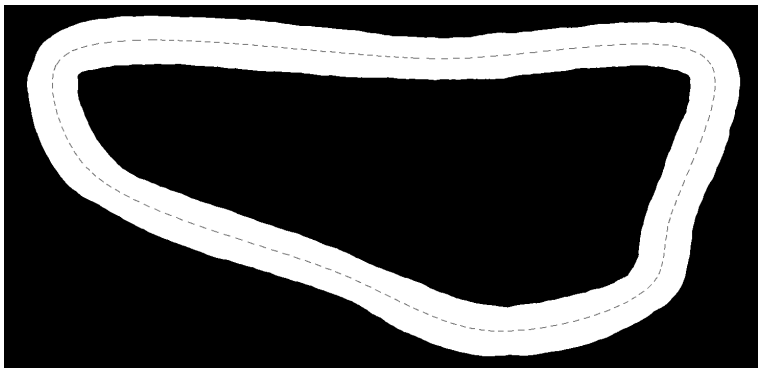
Návrh řešení a implementace

Celou aplikaci jsem implementoval ve vývojovém prostředí IntelliJ IDEA [8] od JetBrains s.r.o. v programovacím jazyce Java [9]. Pro sestavení projektu a doinstalování potřebných závislostí byl využíván nástroj pro správu projektů nazývaný Maven. Pro práci se strojovým učením byla využívána knihovna Deeplearning4j [10] disponující širokou podporou algoritmů hlubokého učení napsaných v Javě. Zdrojový kód je spravován verzovacím systémem Git [11] [12].

Program se skládá z prostředí reprezentovaného plátnem a jednoho vozidla s definovanými fyzikálními vlastnostmi a kamerovým senzorem. Vozidlo je možno řídit přímo uživatel pomocí kontroleru nebo Reinforcement learningem.

4.1 Procedurální generování tratí

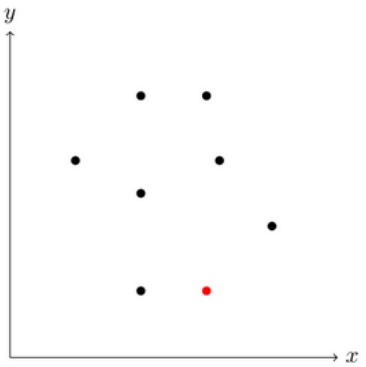
Simulátor umožňuje generovat procedurálně generované tratě různé složitosti. Základním stavebním blokem procedurálně generovaných tratí je konvexní obal. Konvexní obal množiny bodů je definován jako nejmenší konvexní polygon, který uzavírá všechny body v množině. K nalezení konvexního obalu využívám Andrewův monotónní řetězový algoritmus. [13] Kontrolní body tvořící konvexní polygon jsou transformovány do Catmull-Rom křivky. Jednou z nevýhod generátoru založeného na konvexním obalu je, že neumožňuje triviálně generovat komplexní trasy s konkávní topologií. Příklad procedurálně vygenerované tratě je vidět na obrázku č. 4.1.



Obrázek 4.1: Příklad procedurálně vygenerované tratě

4.1.0.1 Andrewův monotónní řetězový algoritmus

Andrewův monotónní řetězový algoritmus konstruuje konvexní obal množiny 2-dimenzionálních bodů v čase O (n log n). Algoritmus začíná seřazením bodů lexikograficky (podle souřadnice x a v případě rovnosti bodů podle souřadnice y). V druhém kroku je zvolen počáteční bod konvexního obalu. Počáteční bod je na obrázku 4.2 označen červeně. [14]



Obrázek 4.2: Příklad počátečního bodu konvexního obalu

Typicky se jedná o rohový bod s nejnižší souřadnicí x nebo y. Pokud existuje více bodů na stejné souřadnici x, tak se rozhoduje podle druhé souřadnice. Algoritmus pokračuje procházením seřazených bodů. Pro každou trojici bodů rozhodneme, zda tvoří konvexní nebo konkávní roh. Pokud je roh konkávní, pak střední bod z této trojice neleží na konvexním obalu. Z takto vygenerovaných bodů ležících na konvexním obalu je vykreslena trať na plátno, tak že jednotlivé části vozovky tvoří hrany mezi body obalu. Šířka vozovky je generovaná náhodně v definovaném rozsahu.

4.2 Protředí

Testovací prostředí je tvořeno plátnem, na které je vykreslená uzavřená procedurálně generovaná trať. V této práci uvažuji pouze trasu v otevřené krajině. Trasy jsou považovány za hladké bez ostrých zatáček nebo výškových rozdílů. V prostředí funguje jednoduchý systém detekce kolizí na základě barvy, kde bílá barva značí vozovku a černá nesjízdný povrch. Simulátor neumožňuje jízdu po nesjízdném povrchu. V případě, že vozidlo vyjede z tratě na nesjízdný povrch, dojde ke kolizi a tím k úplnému zastavení vozidla, kromě toho je trasa generovaná tak, aby vozidlo nebylo schopné se otočit během jízdy do protisměru. Detekce kolizí testuje pro každý časový krok, zda body tvořící ohraničující rámeček (bounding box) vozidla se nenacházejí na souřadnici v prostředí, kde se zároveň nachází nesjízdný povrch (černá barva). V případě použitého modelu vozidla v simulátoru je ohraničující rámeček reprezentovaný obdélníkem. Ohraňujíčí rámeček použitý v detekci kolizí je prezentován na obrázku 4.3.

Obrázek 4.3: Ohraničující rámeček použitý v detekci kolizí (bounding box)

Prostředí zobrazuje uživateli v reálném čase obraz kamerového senzoru, aktuální rychlost a informace o úhlu natočení kol v radiánech, kromě toho během tréninku uživateli zobrazuje aktuální informace o odměnách, které vozidlo v dané epizodě získává a informace o tom, jak vozidlo v příštím kroku upraví rychlost a natočení kol. Prostředí je vykreslované jednoduchým vykreslovacím modulem. Vykreslovací okno je zhruba 1500 pixelů široké a 750 pixelů vysoké. Poměr zmenšení vykreslovacího plátna, tedy poměr délky měřené na plátně k délce ve skutečnosti je nastaven na 1:7, neboli 1 pixel na plátně je zhruba 7 centimetrů ve skutečnosti.

4.3 Modelování fyziky automobilu

Vozidlo použité v simulátoru uvažuje pouze 1 rychlostní stupeň a disponuje pedálem brzdy a pedálem plynu. Vozidlo je poháněno použitím pedálu plynu, který simuluje zrychlení vozidla ve formě pseudo síly, která tlačí vozidlo vpřed. Velikost síly motoru $F_{traction}$ je řízena přímo uživatelem nebo reinforcement learningem. Brzda působí na vozidlo sílou v opačném směru než síla motoru.

4.3.1 První Newtonův zákon setrvačnosti

Newtonův první zákon setrvačnosti říká, že každé těleso setrvá v klidu nebo v rovnoměrném přímočarém pohybu, dokud není nuceno změnit svůj stav působením vnější síly (například působením jiného tělesa). To znamená, že pokud těleso setrvá v klidu nebo v rovnoměrném přímočarém pohybu

a nepůsobí na něj žádná síla, pak si těleso udrží konstantní rychlost. Pokud je tato rychlost nulová, pak objekt zůstane v klidu. Pokud na těleso působí vnější síla, rychlost se změní v důsledku síly.

Simulátor uvažuje první pohybový Newtonův zákon setrvačnosti. Vozidlo v rovnoměrném přímočarém pohybu při konstantní rychlosti udrží tento pohyb, dokud na něj nebude působit vnější síla. Jediným důvodem, proč vozidlo s vyřazenou rychlostí nebude navždy setrvat v této rychlosti, je ten, že tření a vnější síla, ho postupně zpomalují. Vnější síla vzniká třením mezi vozovkou a pneumatikami a vlivem proudícího vzduchu kolem vozu.

4.3.2 Druhý Newtonův zákon síly

Druhý zákon vysvětluje, jak se rychlost tělesa mění, když je vystaveno vnější síle. Jestliže na těleso působí síla, pak se těleso pohybuje zrychlením, které je přímo úměrné působící síle a nepřímo úměrné hmotnosti tělesa. Zákon lze vyjádřiv vztahem $F = m * a$, kde F je vektor síly, m je konstantní hmotnost tělesa a a je vektor zrychlení. Vektory zrychlení a síly mají podle této rovnice stejný směr.

Simulátor uvažuje druhý pohybový Newtonův zákon síly. V případě, že je na automobil aplikována síla motoru, je změna pohybu úměrná síle dělené hmotností vozidla. Silnější motor znamená rychlejší změny v pohybu a vyšší hmotnost vozidla způsobí, že vozidlo bude na síly reagovat pomaleji, proto jsou auta s nízkou hmotností rychlá a výkonná. Čím silnějším motorem vozidlo disponuje a čím méně váží, tím většího zrychlení lze dosáhnout.

4.3.3 Třetí Newtonův zákon akce a reakce

Třetí Newtonův pohybový zákon akce a reakce říká, že pro každou akci (síla) existuje stejná reakce opačného směru. Jinými slovy, pokud těleso $t1$ vyvíjí sílu na jiné těleso $t2$, pak i těleso $t2$ také vyvíjí stejnou sílu na těleso $t1$.

Simulátor využívá třetí Newtonův zákon ve formě brzd a zrychlení. V případě zrychlení působí síla pneumatik na vozovku a ta je vyrovnána stejnou silou v opačném směru působící směrem od vozovky na vozidlo. Podobně sešlápnutím pedálu brzdy způsobí, že pneumatiky vozidla tlačí silou proti vozovce a vozovka tlačí stejnou silou v opačném směru zpět, dokud vozidlo nezpomalí.

4.3.4 Odpor vozidla

Odpor vozidla je kombinace sil, které působí proti pohybu vpřed. Odpor ovlivňuje mnoho faktorů. Mezi tyto faktory patří hmotnost vozidla, gravitace, setrvačnost, množství tření v nápravách, množství tření mezi pneumatikami a povrchem vozovky a odpor vzduchu. Čím větší odpor, tím více síly je zapotřebí k pohonu vozidla. V osobních vozidlech je 3 až 11 procent plynu použito k překonání této síly. Snížení odporu na možné minimum znamená úsporu paliva.

Prvním a obvykle nejdůležitějším odporem je odpor vzduchu, neboli aerodynamický odpor. Odpor vzduchu je úměrný druhé mocnině rychlosti. Simulátor aproximuje sílu odporu vzduchu

následujícím vztahem

$$F_{\text{drag}} = -C_{\text{drag}} * v * |v|$$

Konstanta C_{drag} je dána vztahem $C_{\text{drag}} = 0.5 * Cd * A * rho * v^2$, kde Cd je koeficient tření, A je čelní plocha vozidla a rho je hustota vzduchu. Vektor v je aktuální rychlost vozidla. Dalším odporem je valivý odpor. Valivý odpor vzniká třením mezi pneumatikami a povrchem vozovky a třením v nápravách. Síla odporu je lineárně závislá na aktuální rychlosti vozidla. Při nízkých rychlostech je valivý odpor hlavní odporovou silou, naopak při vysokých rychlostech převažuje odpor vzduchu. Při zhruba 100 km/h jsou tyto dvě síly ekvivalentní. Sílá odporu je vypočtena následujícím vztahem

$$F_{\text{rr}} = -C_{\text{rr}} * v$$

Celková podélná síla vozidla je vypočtena vektorovým součtem těchto tří sil.

$$F_{\text{long}} = F_{\text{traction}} + F_{\text{drag}} + F_{\text{rr}}$$

Zrychlení vozidla je vypočteno podle druhého Newtonova pohybového zákonu $a = F/m$, kde a je zrychlení, m je hmotnost tělesa a F je síla působící na těleso. Eulerova metoda pro numerickou integraci $v = v + dt * a$, kde dt je časový přírůstek v sekundách od předchozího volání, aproximuje rychlost vozidla. Nová pozice vozu je zase určena integrací rychlosti v čase následujícím vzorcem $p = p + dt * v$, kde p je aktuální pozice vozidla. Pro simulaci brzdy je v tomto modelu použita konstanta F_{braking}. Brzda působí silou na vozidlo v opačném směru jízdy.

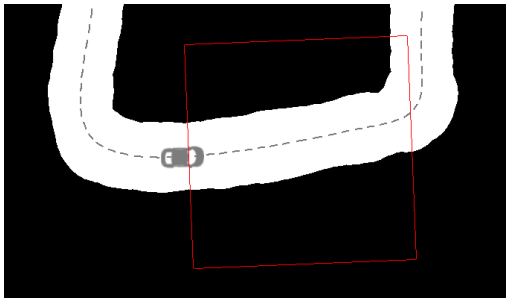
Pomocí výše zmíněných sil je možné poměrně přesně simulovat zrychlení vozidla. Společně také určují nejvyšší rychlost vozu pro daný výkon motoru. Není třeba nikde v kódu uvádět maximální rychlost. Maximální rychlost vyplývá z uvedených rovnic. Rovnice tvoří pomyslnou smyčku díky negativním odporovým silám. Pokud tažná síla (síla motoru) přetlačí všechny ostatní síly, vůz zrychluje, nicméně vyšší rychlost způsobuje nárůst odporových sil. Tažná síla je postupně vyvažovaná sílou odporu a v určitém okamžiku se odporové síly začnou navzájem rušit a v tomto bodě vozidlo dosáhlo nejvyšší rychlosti pro daný výkon motoru. [15]

4.3.5 Smyk

Aby se vůz udržel ve směru natočení kol, musí být třecí síla mezi vozovkou a pneumatikami dostředivá. Když je odstředivá síla větší než třecí síla, auto se začne dostávat do smyku. Bezpečnou rychlost v zatáčce lze zvýšit sklonem silnice v zatáčce. Pokud je silnice správně nakloněná, lze zabránit smyku bez použití třecích sil. Simulátor předchází vzniku smyku nastavením maximálního úhlu natočení kol v závislosti na rychlosti vozidla, tak aby bylo schopné projet ostré zatáčky pouze v bezpečné rychlosti.

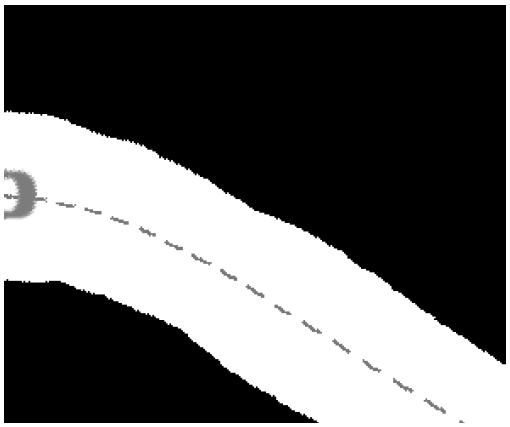
4.4 Kamerový senzor

Vozidlo disponuje kamerovým senzorem umístěným nad vozidlem, jenž snímá každý časový krok bezprostřední okolí vozidla a výhled řidiče. Senzor dokáže zachytit informace o dopravní situaci, zda se blíží k zatáčce, nebo jestli se například vozovka zužuje. Obraz senzoru je reprezentovaný bitmapou o velikosti 300x300 pixelů odpovídající ve skutečnosti zhruba viditelnosti 20 metrů vpřed. Díky dostatečně velké snímané oblasti má vozidlo daleko lepší představu o prostředí a složitosti trasy v dostatečném předstihu a umožňuje chápat souvislosti v širším kontextu. Senzorem snímanou oblast shora dolů a neboli simulaci výhledu řidiče demostruje obrázek 4.4, kde červená barva označuje hranice výhledu řidiče (snímané oblasti).



Obrázek 4.4: Snímaná oblast kamerovým senzorem

Skutečná viditelnost řidiče ve vozidle v reálném světě je v porovnání s kamerovým senzorem mnohonásobně větší, nicméně kvůli rostoucí paměťové složitosti s velikostí obrazu je velikost výhledu v simulátoru omezena. Barevný obraz není v tomto případě důležitý a nepřidává obrazu žádnou hodnotu a zbytečně zvyšuje paměťovou složitost algoritmu, proto kamera snímá okolí pouze ve stupních šedi. Snímaná oblast senzoru musí být dostatečně přiměřená maximální rychlosti vozidla, v opačném případě může být pro vozidlo nemožné rozpoznat stavy, kdy je nutné zareagovat včas a začít brzdit před ostrou zatáčkou v dostatečném časovém předstihu. Příklad obrazu senzoru je vidět na obrázku 4.5.



Obrázek 4.5: Přiklad obrazu kamerového senzoru

4.5 Normalizace a standardizace vstupních dat

Data z kamerového senzoru jsou v reálném čase posílána do neuronové sítě za účelem predikce nebo tréninku. Před vstupem do hluboké neuronové sítě jsou data normalizována a standardizována. Obrazu z kamery je změněna velikost na 40x40 pixelů a rozsah hodnot je přeškálován do rozsahu <0, 1>. Normalizace obrazu hraje důležitou roli pro správné trénování modelu a pozorování vyhodnocování výkonnostních metrik modelů při tréninku. Váhy modelů jsou na začátku inicializovány na malé náhodné hodnoty a aktualizují se pomocí optimalizačního algoritmu na základě odhadu chyby mezi predikovanými a skutečnými hodnotami. Nenormalizované vstupní hodnoty mohou mít za následek pomalý nebo nestabilní proces učení.

Augmentace obrazových dat se používá k rozšíření tréninkové datové sady za účelem zlepšení výkonu a schopnosti modelu se lépe zobecnit. Obvykle se na tréninková data aplikují různé transformace, zejména posuny ve směru jedné z os, zrcadlení nebo škálování obrazu. Simulátor rozšiřuje trénovací data tak, že vozidlo postupně střídá směr jízdy, díky tomu vozidlo pozná rozmanitější prostředí, které pomáhá modelu se zobecnit. Pokud by vozidlo jezdilo například ve směru hodinových ručiček pořád dokola, výsledný model by pravděpodobně byl schopný rozeznat pouze pravoúhlé zatáčky a při styku s levotočivou zatáčkou by došlo ke kolizi.

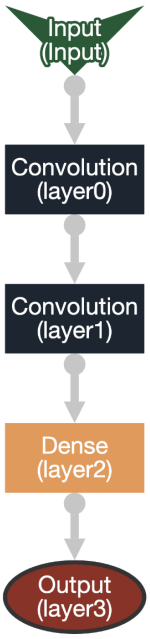
4.6 Architektura neuronové sítě

Architekturu neuronové sítě tvoří celkem 4 vrstvy. První dvě vrstvy jsou konvoluční a mají za úkol rozpoznat situaci vozidla. Vstupem konvoluční vrstvy je obraz, kterému vrstva přiřazuje důležitost různých aspektů / objektů a je schopna je odlišit jeden od druhého. Zatímco v primitivních metodách klasifikace jsou filtry vytvářeny ručně, konvoluční vrstvy s dostatečným tréninkem mají schopnost se tyto filtry a vlastnosti naučit. Paměťová a výpočetní složitost konvolučních vrstev roste s velikostí jejich vstupu, proto je vhodné velikost obrázku zmenšit a zbavit se zbytečných

estetických detailů obrazu. Změna velikosti vstupu na rozumnou velikost zrychlí výcvik a sníží paměťové nároky. Zpracované výsledky z konvolučních vrstev jsou mapovány na 256 výstupů pomocí třetí plně propojené vrstvy (dense layer). Čtvrtá vrstva výstupní mapuje předešlých 256 výstupů na výstupní akce. Přesnost odhadů je měřena pomocí střední kvadratické chyby (MSE). MSE je definovaná následující rovnicí.

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i, \theta))^2,$$

kde x_i je vstupní obraz, parametr θ jsou váhy neuronové sítě, y_i je Q-Hodnota z pravé strany Bellmanovy rovnice a $f(x_i - \theta)$ je výstup neuronové sítě. Použitá architektura v této práci je vidět na obrázku číslo 4.6.

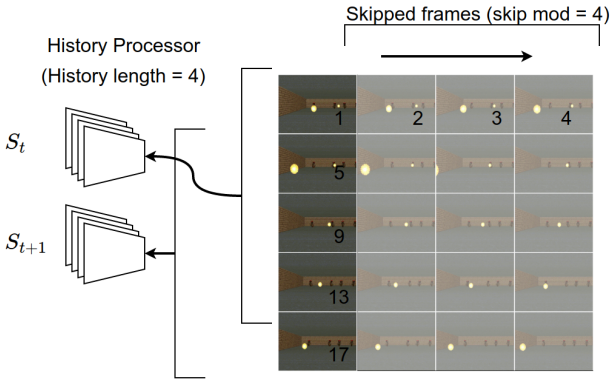


Obrázek 4.6: Architektura neuronové sítě

4.7 Zpracování historie

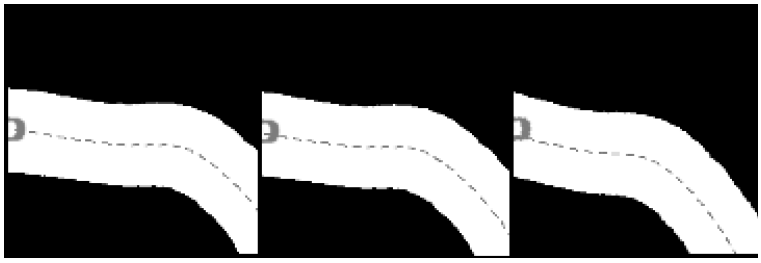
Zpracování historie má zásadní vliv na učení správných reakcí na ostré zatáčky, nerovnosti na vozovce, chodce, dopravní značení nebo protijedoucí vozidla. Procesor historie zpracovává posledních n snímků, které následně skládá do kanálů. Tyto snímky reprezentují momentální hybnost vozidla. Procesor každý časový krok vkládá aktuální snímky do fronty. Jakmile má procesor k dispozici ve

frontě dostatečné množství naakumulovaných snímků, jsou snímky spojeny do n kanálů v závislosti na konfiguraci. Obvykle není nutné neuronové síti předkládat veškeré informace o pohybu pouze o 1 krok napřed, proto je výhodné využít funkci přeskakování snímků. Přeskakování snímků (skip mod) může zrychlit učení za cenu relativně malé ztráty informací. Například využitím přeskakování 4 snímků je ve skutečnosti zpracován neuronovou sítí pouze 1 ze 4 snímků. U přeskočených snímků agent opakuje poslední použitou akci, dokud nedojde k akumulaci dalších snímků ve frontě. Procesor historie je zobrazen na obrázku č. 4.7.



Obrázek 4.7: Procesor historie

Příklad vstupu do neuronové sítě s využitím historie o délce 3 ve vybraném simulačním prostředí je vidět na obrázku 4.8.



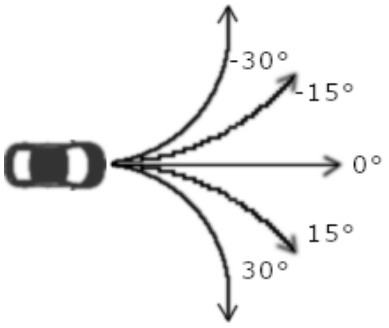
Obrázek 4.8: Příklad výstupu z procesoru historie (n = 3)

4.8 Návrh řešení

Problem lze rozdělit do dvou fází. V první fázi se vozidlo učí řídit pomocí volantu a v druhé fázi se vozidlo učí zrychlovat. Rozdělením problému do dvou kroků je výrazně snížena časová složitost v porovnání s tréninkem obou dovedností zároveň. Myšlenka rozdělení tréninků je inspirovaná člověkem. Lidé za volantem přirozeně ovládají paralelně pedál plynu i řízení nezávisle na sobě a lze je považovat za dvě odlišné dovednosti. Nevýhodou tohoto přístupu je silná závislost druhé fáze na

spolehlivosti modelu z první fáze. Po natrénování se oba modely vzájemně doplňují a každý časový krok jsou predikovány celkem dva výstupy. První výstup je výběr akce řízení a druhý výstup říká vozidlu jakou rychlost má zvolit v závislosti na prostředí. Obě fáze využívají Epsilon Greedy strategii. Učení začíná náhodnými kroky ($epsilon = 1$). V průběhu učení je hodnota *epsilon* postupně snižována. Hodnota epsilon určuje pravděpodobnost náhodného kroku místo využití dosavadně naučených schopností (predikce). Automobil využívá kamerový senzor. Obraz z kamerového obrazu vstupuje do konvoluční neuronové sítě. Obraz zároveň definuje stav, ve kterém se vozidlo v danou chvíli nachází. V průběhu učení jednotlivých fází se náhodně mění směr jízdy vozidla. Po určité době se epocha automaticky restartuje a vozidlo je vsazeno do prostředí v opačném směru. Počáteční poloha agenta se odvíjí od místa vniku kolize. Celá trať je rovnoměrně navzorkovaná body v oblasti středové dělící čáry. V případě kolize, prostředí automaticky najde nejbližší bod a využije jej jako počáteční polohu agenta.

V první fázi je vozidlo vsazeno do prostředí a je mu na startu nastavena konstantní rychlost. Konstantní rychlost je záměrně zvolená, tak aby vozidlo nebylo omezováno ve smyslu prevence smyku kvůli vysoké rychlosti a mohlo tak naplno využít zatáčení v plném rozsahu. Reinforcement learning řídí automobil natáčením volantu do 5 poloh. Mezi tyto polohy patří ostře doleva, mírně doleva, mírně doprava, ostře doprava a poloha, kdy vozidlo udržuje stejný směr, přičemž poloha ostře doprava odpovídá maximálnímu natočení kol o 30 stupňů doprava a ostře doleva odpovídá maximálnímu natočení kol o 30 stupňů doleva. Mírně doleva a mírně doprava se rovná polovině maximálního natočení kol na danou stranu. Natočení kol se vždy počítá jako relativní úhel mezi vozidlem a natočením kol. V případě, že vozidlo udržuje stálý směr je relativní úhel mezi vozidlem a koly rovný 0. Jednotlivé akce řízení jsou zobrazeny na obrázku 4.9.



Obrázek 4.9: Vizualizace akcí řízení

Algoritmus má možnost vybrat jakoukoliv strategii a neváže se na předchozí zvolenou akci. Algoritmus může přeskakovat libovolně mezi různými úhly natočení volantu bez ohledu na akci zvolenou v předchozím kroku. Mezi jednotlivými výběry akce je časová pomlka, jelikož vozidlo využívá principů fyziky a potřebuje čas na to, aby se jednotlivé změny projevily v čase, proto nemá smysl trénovat neuronovou síť na datech, ve kterých není žádná viditelná změna v reakci na zvolenou strategii. V případě chybného rozhodnutí, které vyústí v kolizi, je životní cyklus vozidla restartován

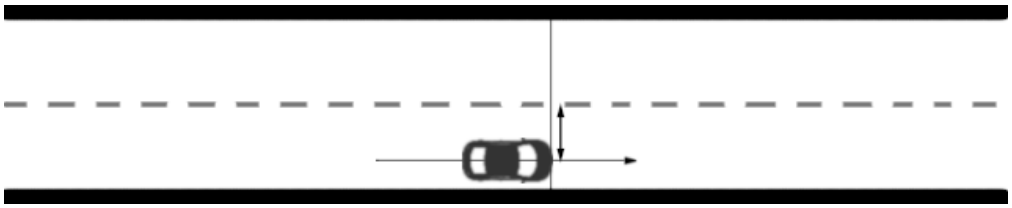
a vozidlo je přemístěno v místě kolize do středu vozovky a celý proces se opakuje. Maximální počet kroků epizody není nijak v tomto případě omezen, protože vozidlo v ranné fázi učení často chybuje, jelikož není schopné pomocí náhodných tahů udržet řízení uprostřed vozovky a většinou skončí epizoda po několika málo krocích kolizí.

Druhou fází učení je trénování zrychlení. Během tohoto procesu je využit natrénovaný model řízení. Jelikož každá fáze využívá svůj vlastní model, tak každý časový krok modely generují dvě predikce. Jedna predikce slouží k řízení a druhá k ovládaní zrychlení. Algoritmus řídí vozidlo třemi různými kroky. Vozidla má možnost mírně zrychlovat (1/4 plynu), plný plyn, a nebo brzdit. Posloupnosti vybraných akcí nejsou nijak omezeny a algoritmus je může dle libosti kombinovat. Mezi zvolenými kroky je časová prodleva, aby se rozhodnutí algoritmu stihli projevit v čase. Díky použitému fyzikálnímu modelu, jsou přechody mezi jednotlivými akcemi plynulé a celá jízda působí realisticky. Aby se byl model schopný správně naučit, kdy má brzdit a kdy zrychlovat, potřebuje informaci o tom, jak rychle se blíží k ostré zatáčce. Tuto informaci dokáže zachytit procesor historie, který naskládá posledních n snímků do n kanálů. V případě kolize vlivem nepřiměřené rychlosti je životní cyklus vozidla restartován a vozidlo je přemístěno do středu vozovky a celá epizoda se opakuje. Jednotlivé epizody jsou omezeny maximálním počtem kroků. Omezení maximálního počtu kroků ukončí v případě dlouhé bezchybné jízdy epizodu předčasně a tím optimalizuje celý proces učení.

4.9 Implementace odměn

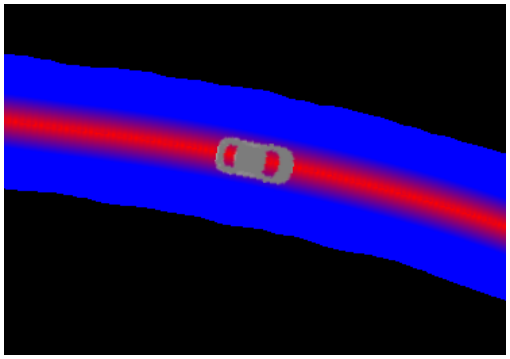
Navrhování funkce odměny, která popisuje optimální chování při řízení, je notoricky obtížné, protože řízení automobilu se skládá z mnoha různých typů scénářů, které lze jen těžko odvodit z explicitní rovnice. Jelikož je však rozsah této práce omezen na to, aby vozidlo bylo jediným účastníkem silničního provozu a prostředí se skládalo ze silnice, která tvoří okruh, lze ignorovat mnoho výzev navrhování funkce odměny za řízení. Abychom agenta vedli k optimálnímu chování, byly navržené funkce odměny zvoleny jako husté (dense).

V této práci jsou navrženy dvě odměňovací funkce pro první fázi učení (natáčení volantu) popisující dva typy chování. První počítá odměny jednotlivých epoch na základě vzdálenosti od středové dělící čáry. Vzdálenost od středové dělící čáry je počítána od středu přední části vozidla (přídi vozidla) (viz obrázek 4.10).



Obrázek 4.10: Vizualizace vzdálenosti od středové dělící čáry

Čím blíže středové čáře se vozidlo nachází, tím vyšší odměny získává. Odměny formou vzdálenosti od středové dělící čáry motivují vozidlo udržovat blízko středové dělící čáry za předpokladu, že vozidlo není schopné se vlastní vůlí otočit do protisměru. V takovém případě by vozidlo mohlo být natočené směrem ke svodidlům, přestože by se nacházelo blízko středové čáry. V následujícím kroku by pravděpodobně došlo k náraz a vozidlo by za to bylo ještě nesprávně odměněno. Vizualizace odměn na základě vzdálenosti od středové čáry je na obrázku 4.11.



Obrázek 4.11: Vizualizace odměn na základě vzdálenosti od středové čáry

Červená barva označuje body s vysokou odměnou a modrá barva označuje body s nízkou až zápornou odměnou. Odměna je vypočtena vztahem

$$R_{\text{distance}} = base - \sqrt{\sum_{i=0}^{n}(p1_i - p2_i)^2},$$

kde parametr *base* je základní odměna, $p1$ je pozice středové dělící čáry a $p2$ je pozice přídě vozidla.

Druhá funkce odměny pro trénink dovednosti řízení odměňuje vozidlo na základě odchylky směru od dělící středové čáry. Odměňovací funkce je definovaná jako

$$R_{\text{steering\_dev}} = \frac{abs(s - rs)}{0.5\pi} - 0.5,$$

kde s je natočení vozidla v radiánech a rs je přímý směr vozovky vyjádřený v radiánech.

Ve fázi učení schopnosti ovládat plyn jsou představeny dvě odměňovací funkce popisující dva typy chování. První funkce odměňuje vozidlo pouze za rychlost. Odměny za ryhlost je vypočítaná jako

$$R_{\text{speed}} = \frac{v}{max_v} - 0.5,$$

kde parametr max_v je maximální rychlost vozidla. Odměny za rychlost mají za úkol motivovat vozidlo ujet co nejdelší trasu v co nejkratším čase, jinými slovy čím rychleji se pohybuje, tím větší odměny získává. Vysoká rychlost znamená větší odměny, nicméně vysoká rychlost vede obvykle ke kolizi a k ukončení celé epizody. Vozidlo se tedy snaží najít pomyslnou rovnováhu mezi optimální

rychlostí, kterou si může vzhledem k topologii trasy dovolit, aby získalo co největší odměny, ale zároveň nedošlo ke kolizi s mantinely či vyjetí z vozovky. Odměny za rychlost lze optimalizovat nastavením hranice minimální rychlosti, za kterou je vozidlo odměňováno.

Druhá funkce odměny ve fázi učení zrychlení odměňuje vozidlo na základě aktuální rychlosti a odchylky od středové dělící čáry. Samotná funkce odchylky není pro toto řešení není dostatečně účinné, protože nijak nemotivuje vozidlo přidávat plyn, nýbrž jen minimalizovat odchylku řízení od středové dělící čáry, proto je tento typ odměny kombinovaný s odměnou za momentální rychlost. Kombinovaná odměňovací funkce je definovaná jako

$$R_{\text{steering\_dev\_speed}} = \begin{cases} R_{speed}, & \text{if } v <= 20 \\ R_{speed} + R_{steering\_dev}, & \text{if v} > 20, \end{cases} \tag{4.1}$$

kde R_{speed} je odměna za rychlost, $R_{steering\_dev}$ je odměna za odchylku směru a v je rychlost vozidla.

V této práci jsou dále představeny dvě sparse (řídké) odměny, které nebyly použité v experimentech. První řídká odměna ve fázi učení zrychlení odměňuje vozidlo na základě času, za který ujede jedno kolo normalizovaným délkou trasy. Vozidlo postupně zkouší, které posloupnosti provedených kroků jej v dané trase vedou k vysokým odměnám. Jelikož k získání odměny musí vozidlo ujet minimálně jeden okruh, je tento typ velice časově náročný a může potřebovat i desetitisíce kroků k naučení dovednosti. Odměňovací funkce je definovaná jako

$$R_{\text{laptime}} = t * rl,$$

kde t je doba, za kterou vozidlo zvládlo úspěšně ujet jedno celé kolo bez kolize a rl je délka trasy.

Druhá řídká odměna penalizuje vozidlo v první fázi učení řízení za kolize. Odměna zdůrazňuje vážnost fatálních následků vlivem kolize a motivuje vozidlo předcházet těmto situacím. Jednorázová penalizace za kolizi je definovaná jako

$$R_{\text{crash}} = \begin{cases} -50, & \text{if car crashed} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

Třetí a zároveň poslední řídkou odměnou je penalizace za použití brzdy při nulové rychlosti. Tento typ odměny učí vozidlo nepoužívat brzdu v případě, že se nepohybuje a předchází tak situacím, kdy vozidlo stojí nehybně na místě a brzdí. Penalizace za použití brzdy je definovaná jako

$$R_{\text{brake}} = \begin{cases} -1, & \text{if } v <= 0 \\ 0, & \text{if v} > 0, \end{cases} \tag{4.3}$$

kde v je rychlost vozidla.

4.10 Implementace MDP

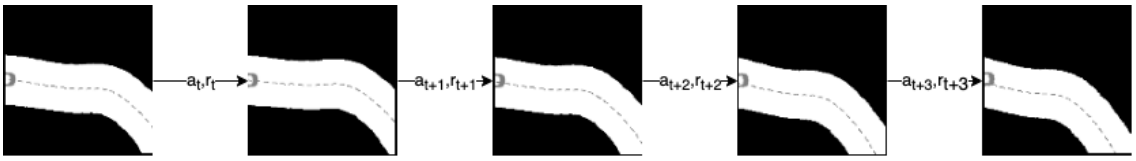
Simulátor implementuje Markovův rozhodovací proces. MDP poskytuje matematický rámec pro modelování rozhodování problému učení řízení a zrychlení v diskrétním prostoru. Tvar stavu z prostředí odpovídá dimenzi a hloubce obrazu z kamerového senzoru po aplikaci funkcí předzpracování. Funkce odměn se odvíjí od požadovaného chování a stylu jízdy a jednotlivé stavy jsou reprezentovány sadou pixelů. Tok řízení prostředí simulace je konstruován tak, že prostředí provede akci, o které agent rozhodne a pro pevný počet snímků je simulace pozastavena. Poté, co agent provedl jeden krok, prostředí vrací nový stav, odměnu a logickou proměnnou, která uvádí, zda aktuální stav prostředí odpovídá koncovému stavu nebo ne. Prostředí nastaví logickou proměnnou odpovídající koncovému stavu na 1 pouze v případě, že se agent nachází mimo vozovku nebo byl večerpán maximální počet kroků v epizodě. Navzorkované body v oblasti středové dělící čáry slouží jako záchytný bod v případě, že je agent v koncovém stavu a MDP je restartován a vozidlo mohlo navázat na předešlou epizodu. Je důležité, aby automobil nezačínal epizodu stále ze stejného místa a neuronové síti bylo předkládáno na vstupu co nejvíce různých dat, v opačném případě by data byla velmi nerovnoměrně distribuována a neuronová síť by se nebyla schopna se správně naučit.

4.10.1 MDP řízení

Akce jsou v případě rozhodovacího procesu řízení definovány jako

$$a_{\text{steer\_t}} = \begin{cases} 0, & \text{ostře doleva} \\ 1, & \text{mírně doleva} \\ 2, & \text{stejný směr} \\ 3, & \text{mírně doprava} \\ 4, & \text{ostře doprava.} \end{cases} \tag{4.4}$$

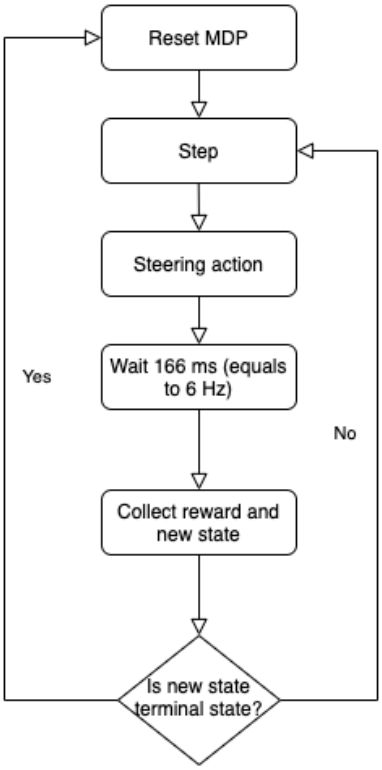
Stavy procesu řízení určují bitmapy tvořené předzpracovanými snímky z kamerového senzoru. Přechody mezi stavy v rozhodovacím procesu řízení demonstruje obrázek 4.12.



Obrázek 4.12: Přechody mezi stavy v rozhodovacím procesu řízení

Vykreslovací modul vykresluje v simulačním prostředí zhruba 60 snímků za sekundu, což se však může občas lišit, pokud vykreslení aktuální scény vyžaduje značné výpočetní zatížení. Pozice vozida je přepočítávána se stejnou frekvencí jako frekvence překreslování scény. Rychlost simulace

není závislá na výkonu hardwaru a počtu vykreslených snímků za vteřinu, protože velikost kroku je normalizovaná časovým rozdílem mezi předchozím vykresleným snímkem a aktuálním vykresleným snímkem (delta time). Na výkonějším hardwaru je velikost kroku menší a jízda působí vizuálně plynuleji, jelikož je schopný vykreslit větší počet snímků za vteřinu než méně výkonný hardware. Počet rámců vykreslených po provedené akci odpovídá počtu vykreslených rámců, které stihne daný hardware vykreslit za 166 milisekund, což pro rozhodnutí odpovídá frekvenci zhruba 6 Hz. Řídící tok MDP řízení znázorňuje následující diagram aktivit (viz obrázek 4.13).



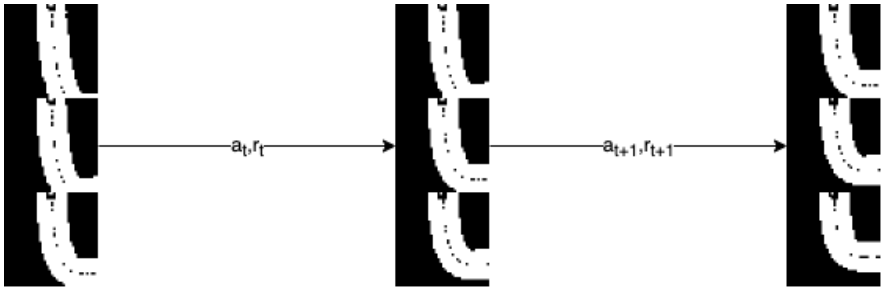
Obrázek 4.13: Diagram aktivit znázorňující řídící tok MDP řízení

4.10.2 MDP zrychlení

Každé akci rozhodovacího procesu zrychlení předchází akce z natrénovaného MDP řízení a následuje akce zrychlení. Akce zrychlení jsou definované jako

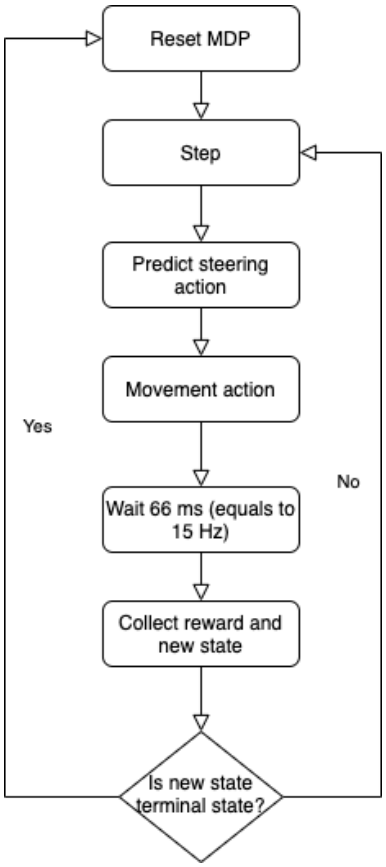
$$a_{\text{move\_t}} = \begin{cases} 0, & \text{plný plyn} \\ 1, & \text{mírně zrychlovat} \\ 2, & \text{brzda.} \end{cases} \tag{4.5}$$

Stavy procesu definují bitmapy tvořené sekvencí předzpracovaných snímků z kamerového senzoru. Přechody mezi stavy v rozhodovacím procesu zrychlení s délkou historie 3 a 2 přeskočenými snímky demonstruje obrázek 4.14.



Obrázek 4.14: Přechody mezi stavy v rozhodovacím procesu zrychlení ($n = 3, skipmod = 3$)

V simulaci zrychlení počet rámců vykreslených po provedené akci odpovídá počtu vykreslených rámců, které stihne daný hardware vykreslit za 66 milisekund, což pro rozhodnutí odpovídá frekvenci zhruba 15 Hz. V simulaci zrychlení, kde se o natočení kol stará natrénovaný model řízení z předchozí simulace, rozhodovací proces využívá prodlevy rovné prodlevě po učiněné akci zrychlení. Rozdílná frekvence kroků v tomto případě nebude mít žádný vliv na výkon modelu řízení, jelikož model počítá s jednoduchým vstupem bez žadných aditivních informací například rychlosti a úhel natočení kol lze předpovědět v jakékoliv situaci nezávisle na rychlosti, kterou se vozidlo pohybuje. Řídící tok MDP zrychlení modeluje následující diagram aktivit (viz obrázek 4.15).



Obrázek 4.15: Diagram aktivit znázorňující řídící tok MDP zrychlení

Kapitola 5

Experimentální ověření algoritmů a jejich porovnání

Experiment využívá Q-Learning algoritmus v kombinace s konvoluční neuronovou sítí. Učení bylo rozděleno do dvou fází a postupně spuštěno. Obě fáze učení využívají svou vlastní instanci výše zmíněné architektury vícevrstvé neuronové sítě. První fáze je fáze učení řízení, kdy se vozidlo učí natáčet volant při konstantní rychlosti. Vozidlo se tak nestará o zrychlení ani brzdu, nýbrž jen o udržení správného směru natáčením volantu. Délka učení v první fázi probíhala zhruba hodinu, kdežto druhá fáze probíhala asi 4x déle. Během učení byly měřeny epochy, vizualizované provedené posloupnosti jednotlivých kroků, chyby modelu (loss) a jednotlivé odměny za epochy. Výsledné váhy naučených modelů jsou uloženy ve formě souboru pro jejich pozdější využití.

5.1 Seznam experimentů

V této práci učím vozidlo dvou různým stylům jízdy ve dvou různých prostředích (trasách). První experiment byl proveden za účelem trénování vozidla závodnímu stylu jízdy a hledání nejrychlejší možné jízdy v dané topologii trasy při zachování bezpečné navigace a stability vozidla. Druhý experiment byl proveden za účelem trénování konfortního a bezpečného stylu jízdy s důrazem na stabilitu jízdy. Oba experimenty využívají dvoufázové učení a používají různé odměňovací funnkce.

5.2 Systémová konfigurace

Knihovna strojového učení DeepLearning4j umožňuje použít grafickou akceleraci a zrychlit tak proces učení, nicméně to umožňuje jen grafickým kartám s podporou architektury CUDA, proto v důsledku absence grafické karty s podporou architektury CUDA nebudu využívat v této práci grafickou akceleraci pro trénink. Alternativním řešením by bylo využít grafické akcelerace na AMD

grafické kartě, kterou mám k dispozici, ale tu bohužel knihovna v současné době nepodporuje. Zvolená systémová konfigurace je vidět v tabulce 5.1.

MacBook Pro (13-palcový, model 2019)	
CPU	Intel Core i5 (1,4 GHz, TB 3,9 GHz)
RAM	8 GB 2133MHz
GPU	Intel Iris Plus Graphics 645 1536 MB

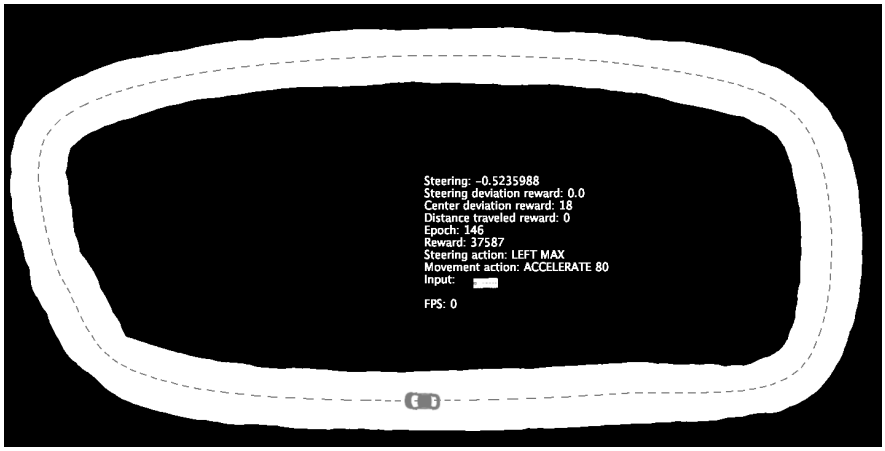
Tabulka 5.1: Zvolená konfigurace pro spuštění učení

5.3 Trénink závodního stylu jízdy

5.3.1 Učení řízení

V závislosti na využití modelu řízení v praxi se budou lišit odměny vozidla za požadovaný styl jízdy. Jiné odměny bude vozidlo dostávat za plynulé řízení a jiné za agresivní styl řízení. Problém řízení se přirozeně nachází ve spojitém prostoru, protože je řízen natáčením (úhlem) volantu v definovaném intervalu. Q-Learning řeší problémy v diskrétním prostoru, proto je nutné spojitý prostor diskretizovat s dostatečně malým krokem, tak aby nedocházelo k skokovému otáčení volantu. V tomto experimentu učím vozidlo udržovat v bezprostřední blízkosti středové dělící čáry.

Pokus byl prováděn v procedurálně generované trati s mírnou obtížností. Zvolené simulační prostředí použité pro učení modelu řízení je zobrazeno na obrázku 5.1. Zvolená konfigurace pro spuštění učení závodního vozidla je uvedeno v tabulce 5.2.



Obrázek 5.1: Zvolené simulační prostředí v rámci učení řízení závodního vozidla

Parametry simulace	
Maximální rychlost	35 km/h
Minimální rychlost	35 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	35
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost trasy	lehká
Délka trasy	230m
Maximální natočení kol	30°

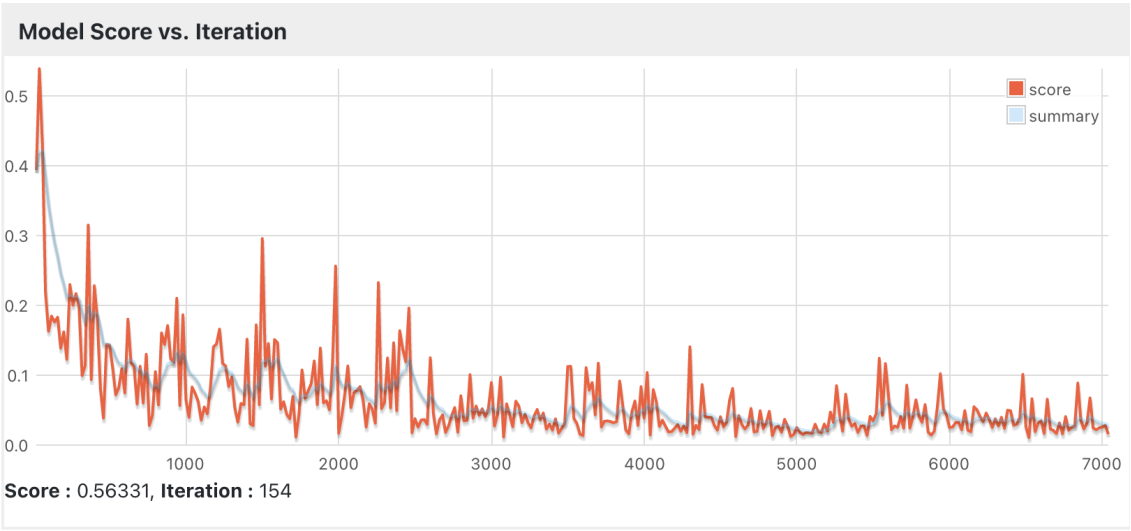
Parametry učení	
Počet epizod	150
Maximální počet kroků v epizodě	∞
Maximální počet kroků	7000
Rychlost učení	0.001
Batch size	10
Gamma	0.99
Minimální epsilon	0.01
Optimalizátor	SGD
Počet trénovatelných parametrů	86 581
Experience replay buffer	7 000

Tabulka 5.2: Zvolená konfigurace pro spuštění učení závodního vozidla

První 2500 iterací (kroků) se model choval velice nestabilně. Zhruba po 3000 iteracích začíná model konvergovat a stabilizovat se. Model bylo možné trénovat i déle a dosáhnout ještě lepší přesnosti, nicméně z hlediska poměru délky trénování a výkonu modelu přispělo nejvíce první 7000 iterací. Po skončení učení se chyba modelu pohybovala v průměru okolo 0.02. Tento výsledek se dá považovat za dokonalé zvládnutí řízení a je možné tento spolehlivý model použit v druhé fázi učení. Celková doba tréninku trvala zhruba 45 minut. Zvolená konfigurace nauronové sítě závodního vozidla je uvedená v tabulce 5.3. Skóre modelu (loss) v jednotlivých iteracích je zobrazen na obrázku 5.2.

Parametry neuronové sítě						
#	Vrstva	Vstupy	Výstupy	Kernel	Střída	Aktivační funkce
0	Konvoluční	3	16	8x8	4x4	Relu
1	Konvoluční	16	32	4x4	2x2	Relu
2	Plně propojená	32	256	-	-	Relu
3	Výstupní	256	5	-	-	Identity

Tabulka 5.3: Zvolená konfigurace neuronové sítě závodního vozidla



Obrázek 5.2: Skóre modelu (loss) v jednotlivých iteracích v učení řízení závodního vozidla

Během učení dominovaly akce udržování směru. Méně často volil agent akce ostře doprava a ostře doleva. Nejméně využívané akce byly mírně doprava a mírně doleva. Distribuci jednotlivých akcí během tréninku je uvedeno v tabulce 5.4.

Akce řízení		
#	Akce	Počet
0	Ostře doleva	1473
1	Ostře doprava	1483
2	Mírně doleva	1307
3	Mírně doprava	1210
4	Udržování směru	1527

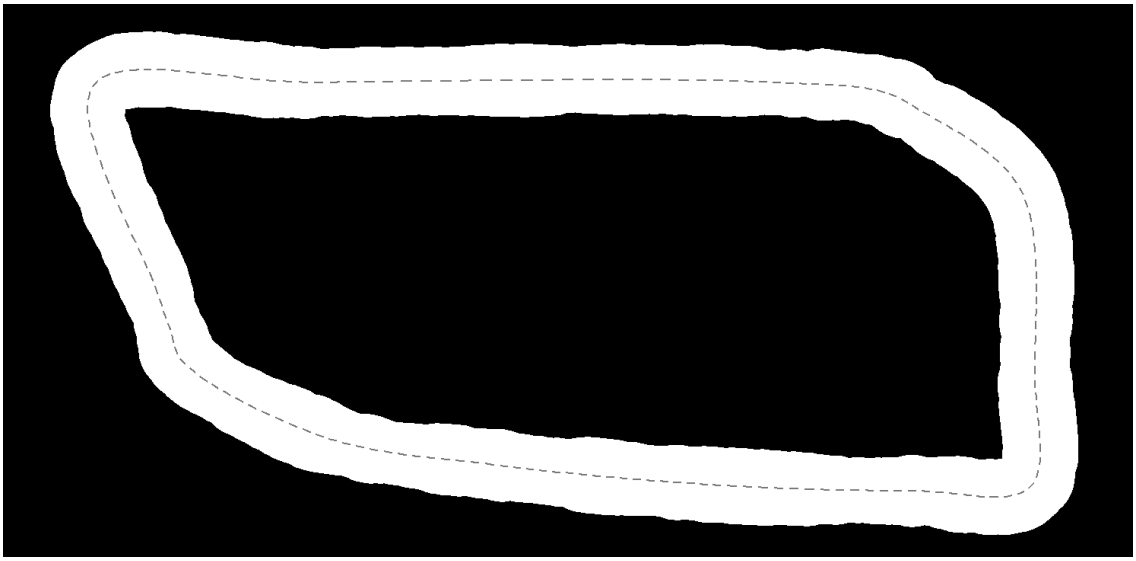
Tabulka 5.4: Distribuce akcí během tréninku řízení závodního vozidla

5.3.2 Učení zrychlování

V závislosti na využití modelu v praxi se liší funkce odměňující vozidla za požadovaný styl jízdy. Jiné odměny budou využity v případě agresivního stylu zrychlování a jiné v případě bezpečného ale zároveň dynamického stylu jízdy. Tak jako u učení řízení se problém zrychlení přirozeně nachází ve spojitém prostoru a je tedy v tomto případě také nutné problém zrychlení diskretizovat dostatečně malým krokem, tak aby bylo možné simulovat realistickou jízdu s ohledem na velikost stavového prostoru a dobu učení. V tomto experimentu učím vozidlo závodnímu stylu jízdy a nutím jej posunout limity až k hraniční rychlosti ve zvolené topologii trasy. Agent je odměňován funkcí R_{speed}. Trať pro trénink by měla ideálně obsahovat několik zatáček, které je možné projet jen v bezpečné rychlosti a

zároveň by měla obsahovat rovinku umožňující využít plný výkon motoru. Výsledný model by měl být schopen simulovat jízdu v maximálně možné rychlosti vzhledem k obtížnosti trasy a vzhledem k maximálnímu výkonu motoru.

Experiment byl prováděn v procedurálně generované trati se střední obtížností. Druhá fáze je časově delší na trénink a to z toho důvodu, že předchozí dokonale natrénovaný model řízení dokáže zvládnout ve většině případů průjezdy zatáček v průměrné rychlosti, proto byla zvolena trať s vyšší obtížností. Průměrná rychlost vozidla vyplývá z Epsilon Greedy strategie, kdy veškeré akce na startu jsou vybírány náhodně a lze tak předpokládat, že kombinace akcí maximálního zrychlení s brzdou vyústí v průměrnou rychlost a model získá poměrně vysoké odměny, i když se ještě nic ve skutečnosti nenaučil. Záměrem je donutit vozidlo udělat v co nejkratší době co nejvíce chyb, aby model začal v rozumném čase konvergovat, proto trať pro učení zrychlování obsahuje záměrně více ostrých zatáček, aby bylo zdůrazněno, kdy přesně má vozidlo přizpůsobovat rychlost pro bezpečnou jízdu a nedocházelo tak k tomu, že vozidlo bude bezchybně jezdit dokola a ve výsledku se nic nenaučí. Zejména by trať měla obsahovat rovinu pro využití plné rychlosti a ostré zatáčky pro plné využití brzdy a zkrácení celkové doby učení. Maximální počet kroků provedených v epizodě je omezený na 200. Limit 200 kroků je v případě bezchybné jízdy dostatečný na to, aby vozidlo stihlo objet alespoň jedno kolo. Zvolené simulační prostředí použité pro učení modelu zrychlení je zobrazeno na obrázku 5.3. Tabulky 5.5 a 5.6 zobrazují zvolenou konfiguraci simulace a prostředí a parametry neuronové sítě. [16]



Obrázek 5.3: Zvolené simulační prostředí v rámci učení zrychlení závodního vozidla

Parametry simulace	
Maximální rychlost	120 km/h
Minimální rychlost	0 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	140
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost trasy	střední
Délka trasy	240m
Maximální natočení kol	30°

Parametry učení	
Počet epizod	844
Maximální počet kroků v epizodě	200
Maximální počet kroků	100 000
Rychlost učení	0.001
Batch size	10
Gamma	0.99
Minimální epsilon	0.01
Optimalizátor	Adam
Počet trénovatelných parametrů	86 581
Experience replay buffer	7 000

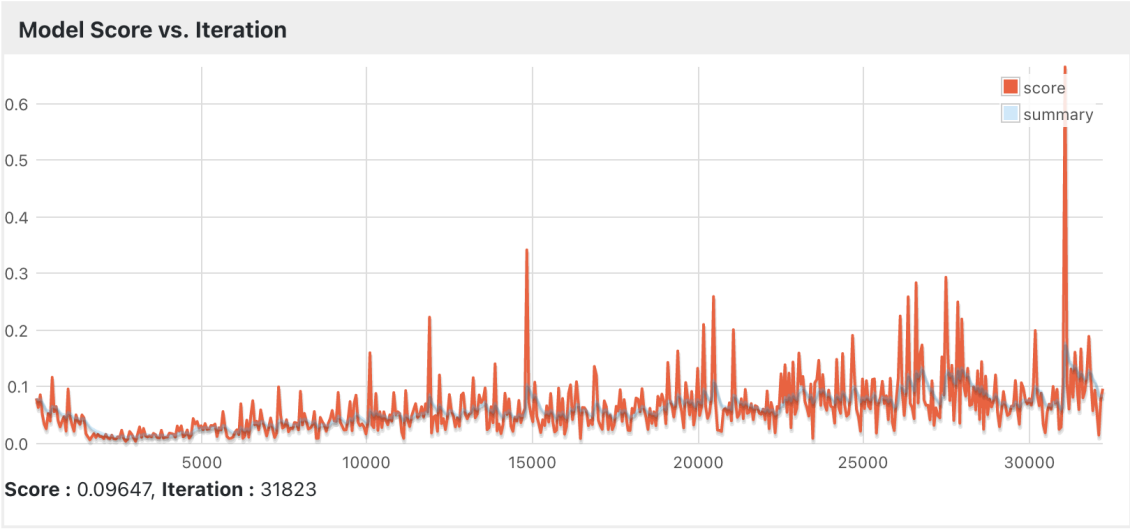
Tabulka 5.5: Zvolená konfigurace pro spuštění učení závodního vozidla

Parametry neuronové sítě						
#	Vrstva	Vstupy	Výstupy	Kernel	Střída	Aktivační funkce
0	Konvoluční	3	16	8x8	4x4	Relu
1	Konvoluční	16	32	4x4	2x2	Relu
2	Plně propojená	32	256	-	-	Relu
3	Výstupní	256	3	-	-	Identity

Tabulka 5.6: Zvolená konfigurace neuronové sítě závodního vozidla

V ranné části tréninku se model choval velmi stabilně. Stabilita modelu vyplývá z Epsilon Greedy náhodné strategie, jež vyústí v průměrnou rychlost vozidla. Postupem času dochází ke snížení *epsilon* a vozidlo začíná pomalu benefitovat z prvně naučených znalostí a začíná volit akce, které generují vyšší odměny. Přesnost modelu se zhoršuje, protože vozidlo zvyšuje průměrnou rychlost a začíná více chybovat za cenu vyšší odměny. Zhruba v polovině tréninku se průměrně ujetá vzdálenost v jedné epizodě zvyšuje a přináší menší spolehlivost, protože vozidlo začíná chápat fakt, že rychlost přináší bohaté odměny, nicméně příliš vysoká rychlost vede v ostrých zatáčkách k nárazu a tak dochází k velkým výchylkám chyb modelu, které jsou vidět na obrázku 1. V závěru učení se model začíná zpět stabilizovat a dochází k závěrečnému ladění modelu. V této fázi učení už vozidlo převazně volí dovednosti, které se během dosavadního tréninku naučilo a zřídka je doplňuje náhodnými kroky. Posledních 3000 iterací se model začíná zpět stabilizovat a zvyšuje se jeho spolehlivost. Po skončení učení se chyba modelu pohybovala v průměru okolo 0.08. Tento výsledek se dá považovat za zvládnutí dovednosti zrychlení. Model bylo možné trénovat i déle nicméně z hlediska poměru délky trénování a výkonu modelu přispělo nejvíce prvních 30000 iterací. Celková doba tréninku trvala zhruba 4

hodiny. V porovnání s modelem řízení, je trénink zrychlení asi 4x časově náročnější na trénink. Skóre modelu (loss) v jednotlivých iteracích je zobrazeno na obrázku 5.4.



Obrázek 5.4: Skóre modelu (loss) v jednotlivých iteracích v učení zrychlení závodního vozidla

Během tréninku převažovala akce plného plynu. Mnohokrát agent zvolil akci brzdy. Nejméně početné zastoupení měla akce mírného zrychlení. Distribuci jednotlivých akcí během tréninku je uvedeno v tabulce 5.7.

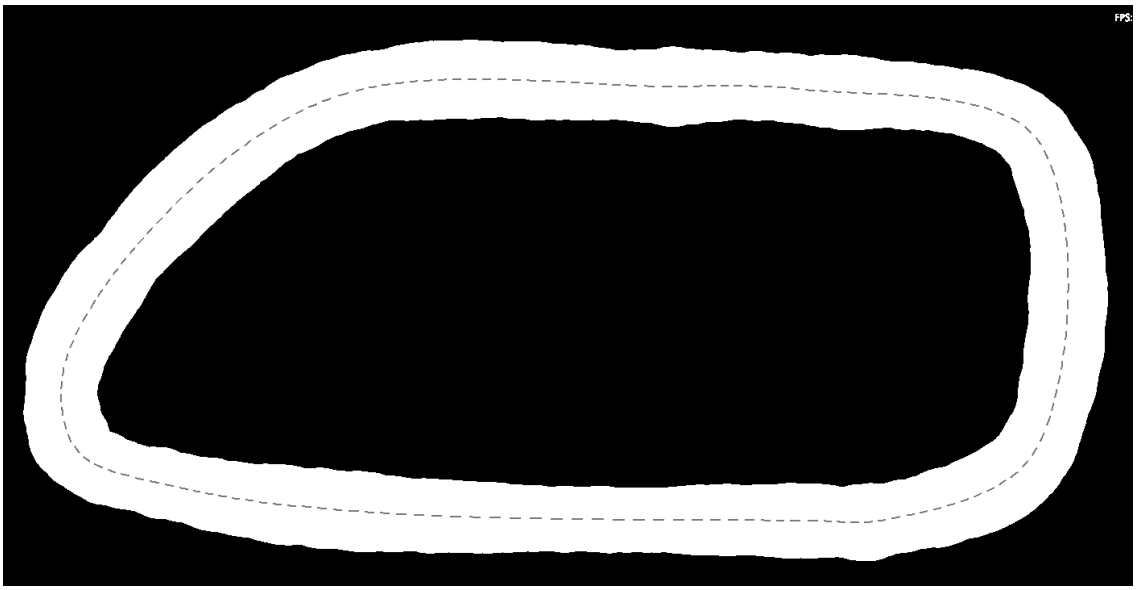
Akce zrychlení		
#	Akce	Počet
0	Plný plyn	46398
1	Mírně zrychlovat (1/4 plynu)	25143
2	Brzda	28337

Tabulka 5.7: Distribuce akcí během tréninku zrychlení závodního vozidla

5.4 Trénink spolehlivého stylu jízdy

5.4.1 Učení řízení

V tomto experimentu učím vozidlo se příliš nevychylovat od přímého směru a přitom zůstat jet co nejvyšší možnou rychlostí. Agent je odměňován funkcí $R_{steering\_dev}$. Pokus byl spuštěn v procedurálně generované trati se střední optížností. Zvolené simulační prostředí pro učení je na zachyceno na obrázku 5.5. Tabulka 5.8 zobrazuje zvolenou konfiguraci simulace a prostředí.



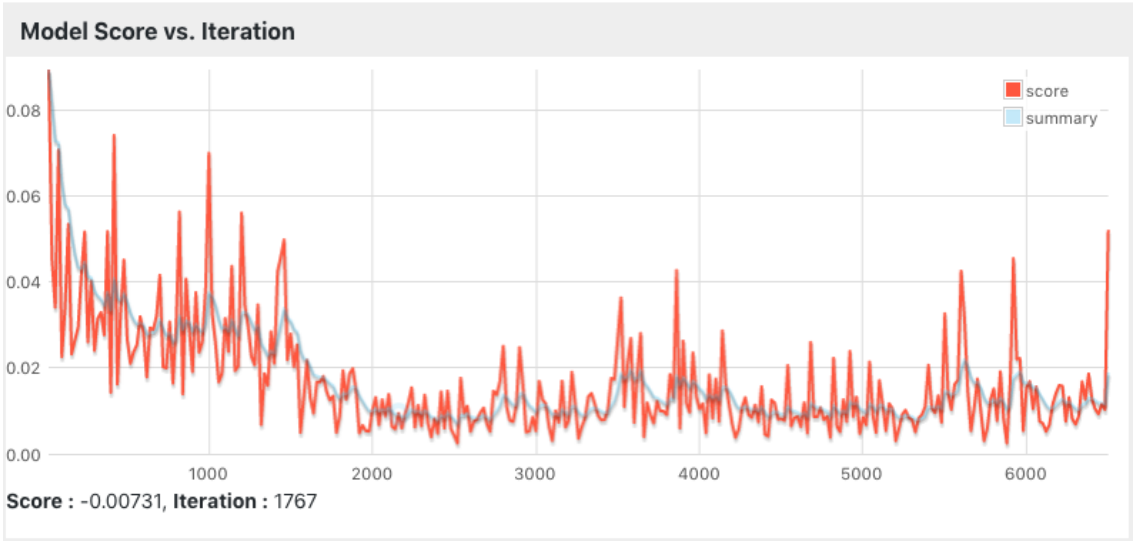
Obrázek 5.5: Zvolené simulační prostředí v rámci učení řízení spolehlivého vozidla

Parametry simulace	
Maximální rychlost	35 km/h
Minimální rychlost	35 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	35
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost trasy	lehká
Délka trasy	270m
Maximální natočení kol	30°

Parametry učení	
Počet epizod	200
Maximální počet kroků v epizodě	∞
Maximální počet kroků	7000
Rychlost učení	0.001
Batch size	10
Gamma	0.99
Minimální epsilon	0.01
Optimalizátor	SGD
Počet trénovatelných parametrů	57 075
Experience replay buffer	5 000

Tabulka 5.8: Zvolená konfigurace pro spuštění učení spolehlivého vozidla

První 1500 iterací byl trénink nestabilní. Doba trvání jedné epochy byla krátká. Počáteční chyba byla zhruba 0.1. Postupem času model začíná po malých krocích konvergovat a stabilizovat. Doba trvání jedné epochy se začíná zvyšovat a vozidlo začalo dostávat vyšší odměny. Největšího zlepšení dosáhl model v první polovině tréninku. Druhou polovinu se rychlost optimalizace modelu začala zpomalovat. V závěru tréninku se chyba pohybovala zhruba kolem hodnoty 0.01. Celková doba tréninku trvala zhruba 50 minut. Obrázek 5.6 zobrazuje skóre modelu v jednotlivých iteracích a v tabulce 5.9 je uvedena zvolená konfigurace neuronové sítě spolehlivého vozidla.



Obrázek 5.6: Skóre modelu (loss) v jednotlivých iteracích v učení řízení spolehlivého vozidla

Parametry neuronové sítě						
#	Vrstva	Vstupy	Výstupy	Kernel	Střída	Aktivační funkce
0	Konvoluční	3	16	10x10	4x4	Mish
1	Konvoluční	16	32	6x6	2x2	Mish
2	Plně propojená	32	256	-	-	Mish
3	Výstupní	256	5	-	-	Identity

Tabulka 5.9: Zvolená konfigurace neuronové sítě spolehlivého vozidla

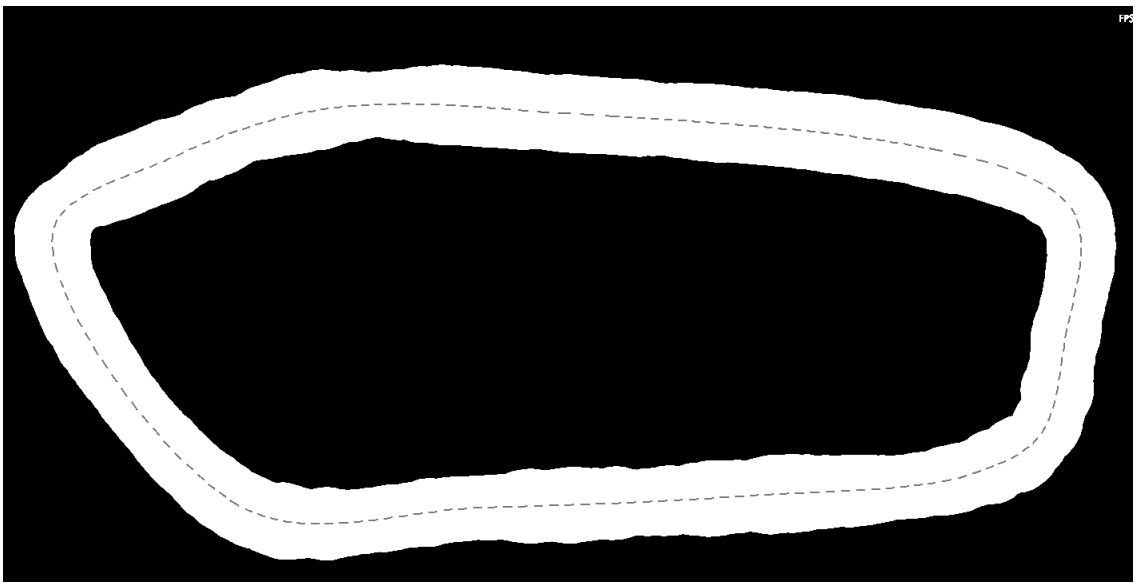
V průběhu učení agent převážně volil akci udržování směru. Druhou nejčastěji používanou akcí byla akce ostře doleva a ostře doprava. Akce mírně doleva a mírně doprava agent využíval nejméně. Distribuce akcí řízení během tréninku zobrazuje tabulka 5.10.

Akce řízení		
#	Akce	Počet
0	Ostře doleva	1405
1	Ostře doprava	1374
2	Mírně doleva	1357
3	Mírně doprava	1371
4	Udržování směru	1493

Tabulka 5.10: Distribuce akcí během tréninku řízení spolehlivého vozidla

5.4.2 Učení zrychlování

V tomto experimentu učím vozidlo příliš neměnit směr jízdy pokud to není nezbytně nutné. Agent je odměňován funkcí $R_{steering\_dev\_speed}$. Výsledkem by měla být simulace plynulé a bezpečné jízdy vzhledem k typu trasy. Trénink byl spuštěn v procedurálně generované trati s lehkou obtížností. Zvolené simulační prostředí je zobrazeno na obrázku 5.7. Konfigurace neuronové sítě a parametry simulace zobrazují tabulky 5.11 a 5.12.



Obrázek 5.7: Zvolené simulační prostředí v rámci učení zrychlení spolehlivého vozidla

Parametry simulace	
Maximální rychlost	90 km/h
Minimální rychlost	0 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	100
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost trasy	střední
Délka trasy	265m
Maximální natočení kol	30°

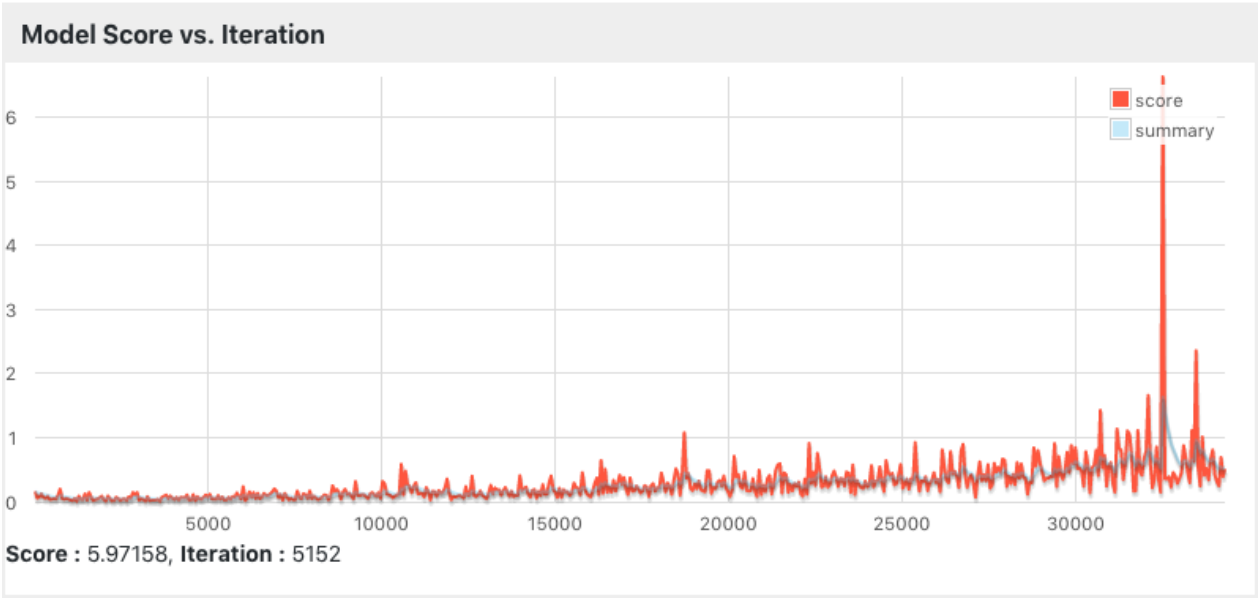
Parametry učení	
Počet epizod	844
Maximální počet kroků v epizodě	200
Maximální počet kroků	70 000
Rychlost učení	0.001
Batch size	10
Gamma	0.99
Minimální epsilon	0.01
Optimalizátor	Adam
Počet trénovatelných parametrů	57 075
Experience replay buffer	7 000

Tabulka 5.11: Zvolená konfigurace pro spuštění učení spolehlivého vozidla

Parametry neuronové sítě						
#	Vrstva	Vstupy	Výstupy	Kernel	Střída	Aktivační funkce
0	Konvoluční	3	16	10x10	4x4	Mish
1	Konvoluční	16	32	6x6	2x2	Mish
2	Plně propojená	32	256	-	-	Mish
3	Výstupní	256	3	-	-	Identity

Tabulka 5.12: Zvolená konfigurace neuronové sítě spolehlivého vozidla

V prvních krocích se model choval stabilně a chyba se pohybovala kolem hodnoty 0.2. Postupem času se chyba modelu začala zvětsovat. Čím déle se vozidlo učilo, tím docházelo k vyšším odchylkám v chybě modelu. Navzdory horší přesnosti modelu, vozidlo v průběhu konvergovalo k optimálnímu chování. Přesnost modelu se postupně zhoršuje, protože agent se v jízdě zlepšuje a tím zvyšuje postupně svoje odměny a průměrnou délku trvání jedné epizody. Postupným zvyšováním odměn roste i její rozptyl a úkol odhadování Q-Hodnot se stává těžší a těžší, proto má model tendenci se zhoršovat. V závěru učení se chyba modelu pohybovala kolem hodnoty 1.5 a vozidlo zvládalo bezpečnou a stabilní jizdu bez zbytečné oscilace v řízení. V tomto případě hodnota chyby modelu neodpovídá tomu, jak kvalitně se během tréninku agent naučil danou schopnost. Obrázek 5.8 zobrazuje skóre modelu v jednotlivých iteracích.



Obrázek 5.8: Skóre modelu (loss) v jednotlivých iteracích v učení zrychlení spolehlivého vozidla

Během tréninku agent nejvíce využil akci plného plynu. Méně často agent využíval brzdu v kombinaci s akcí mírného zrychlení (1/4 plynu). Distribuce akcí zrychlení během tréninku ukazuje tabulka 5.13.

Akce zrychlení		
#	Akce	Počet
0	Plný plyn	32531
1	Mírně zrychlovat (1/4 plynu)	18697
2	Brzda	18772

Tabulka 5.13: Distribuce akcí během tréninku zrychlení spolehlivého vozidla

5.5 Testování

Testy byly prováděny v procedurálně generovaných tratích se střední obtížností odlišných od tratí použitých pro trénink. Vozidlo bylo vsazeno na náhodné místo do středu vozovky a spuštěno. Pro autentičtější výsledky bylo vozidlo ve všech testech spuštěno postupně v obou směrech. Natrénované váhy modelu řízení a zrychlení byly postupně načteny do programu a použity k predikci. Jako vstup do natrénované neuronové sítě byly použity předzpracované kamerové snímky. Parametry simulace jsou uvedeny v tabulce 5.14.

Simulace spolehlivého agenta	
Délka simulace	1000 kroků
Maximální rychlost	90 km/h
Minimální rychlost	0 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	100
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost tras	střední
Počet testovaných tras	5
Maximální natočení kol	30°

Simulace závodního agenta	
Délka simulace	1000 kroků
Maximální rychlost	120 km/h
Minimální rychlost	0 km/h
Maximální šířka vozovky	6,9m
Minimální šířka vozovky	5,5m
Síla motoru	140
Rozměry vozidla	4m x 2m
Hmotnost vozidla v kg	800
Obtížnost tras	střední
Počet testovaných tras	5
Maximální natočení kol	30°

Tabulka 5.14: Parametry testovacích simulací

Každý časový krok oba modely generují celkem 2 hodnoty. První hodnota slouží pro výběr vhodné akce řízení a druhá hodnota rozhoduje o tom, zda vozidlo zrychlí, zpomalí, zabrzdí či bude udržovat současnou rychlost. U predikce zrychlení je důležité dodržet přesný postup transformací aplikovaných na obraz z kamery, zejména zpracovávat stejnou délku historie obrazů jako byla použita během tréninku včetně počtu přeskočených snímků a zachovat tak údaj o momentální hybnosti vozidla v původním formátu, v opačném případě vozidlo nebude schopno dosáhnout tak přesných výsledků jako při tréninku nebo v horším případě nebude schopno jízdy.

Pokud jsou modely správně naučené, měly by se umět generalizovat. Generalizovat v tomto případě znamená, že model není závislý na vstupních datech, na kterých byl naučen. Obě vozidla byly postupně spuštěny na 5 odlišných vygenerovaných tratích. Oba agenti bravurně zvládli bezchybné natáčení volantu v obou směrech ve všech zatáčkách, dokonce i v zatáčkách, které ještě neviděli. To je důkazem toho, že se modely perfektně zobecnily a dokáží si poradit i s neznámým prostředím. Dále bylo z jízd jasně vidět, že obě vozidla v ostrých zatáčkách dokázaly inteligentně využívat brzdu, kterou používaly jen v situacích, které to vyžadovaly a nebylo by možné současnou rychlostí bezpečně projet. Na rovinně vozidla zrychlovala až k hranici maximální rychlosti pro daný výkon motoru a správně zvládla situace, kdy bylo potřeba snížit rychlost a zabránit tak vzniku možného smyku.

5.6 Porovnání

V rámci učení řízení obě použité odměňovací funkce splnili svůj účel a modely konvergovaly k požadovanému chování. V případě závodního stylu jízdy agent udržoval příď vozidla v bezprostřední blízkosti středové dělící čáry a řízení působilo občas trhaně. Na rovince se agent snažil řízení udržet dokonale ve středu, což se zřídka projevovalo oscilací řízení ve středu vozovky z jedné strany na druhou. Oscilace je způsobena hrubou diskretizací spojitého prostoru s poměrně velkým krokem. Naopak v případě spolehlivého stylu jízdy se agent snažil příliš neměnit směr jízdy a řízení působilo plynuleji. Rozdílná délka tras neměla na učení výrazný vliv. Přestože obě učení měla nastavený stejný maximální počet kroků, tak závodní agent se choval spolehlivěji a způsobil méně kolizí než jeho konkurent. Navzdory rozdílnému počtu trénovatelných parametrů se oba modely dokázaly dovednost perfektně naučit.

V rámci učení zrychlení navrhnuté odměňovací funkce splnili svůj účel a agent se naučil požadované chovaní. Narozdíl od učení řízení oba grafy skóre (kvality odhadů) v jednotlivých iteracích měly stoupající trend a modely měly tendenci se zhoršovat. Ukázalo se, že učení spolehlivého agenta bylo rychlejší na natrénování a trénink nevyžadoval tolik kroků jako v případě tréninku závodního agenta. Malé rozdíly v délkách tras neměly na trénink žádný vliv. Spolehlivý agent volil raději průměrnou rychlost a nevychylovat se příliš od přímého směru, naopak závodní agent se choval agresivně a snažil se projet trasu maximální možnou rychlostí. Mezní situace nastávala pokud závodní vozidlo zrychlilo natolik, že nestihlo zareagovat včas na neznámou a ostrou zatáčku, která nebyla použitá při tréninku. Důvod proč nedokáže model správně zareagovat na neznámou ostrou zatáčku vyplývá přímo z povahy toho jak neuronové sítě fungují. Jedna z mála nevýhod neuronových sítí je fakt, že nedokážou generovat hodnoty vyšší než ty, které síť viděla při tréninku. V takovém případě je možné problém vyřešit snížením výkonu motoru nebo nastavením maximální rychlosti (omezovače) a nebo nechat model dál učit i po hlavní tréninku během predikce a dát mu tím možnost si v průběhu dodatečně upravovat váhy a naučit se tak nově vzniklé typy vozovek, jež nebyly dostupné při tréninku a vznikly až po nějakém čase.

Kapitola 6

Závěr

Vozidla s autonomním řízením mají potenciál zvýšit bezpečnost silničního provozu, redukovat dopravní zácpy a poskytovat mobilitu i těm, kteří nejsou schopni ručně vozidlo ovládat ze zdravotních důvodů. Organizace SAE International definovala 5 stupňů automatizovaného řízení. Většina dnešních vozidel je stále na úrovni 0 - bez automatizace. Vývoj bezpečných a účinných autonomních vozidel, jenž jsou schopny jízdy za všech podmínek je jednou z největších výzev dnešních automobilových výrobců. Největší výrobci automobilů zvolili odlišné přístupy řešení autonomního řízení. Automobilový výrobce Tesla zvolil přístup obdobný přístup použitý v této práci, kdežto výrobce GM vsadili na metodu předpočítaných map. Volkswagen přistupuje k řešení problému autonomního řízení tak, že přizpůsobuje prostředí a pozemní komunikaci, tak aby bylo pro vozidlo lépe pochopitelné.

Reinforcement learning algoritmy dokážou vyřešit komplexní problémy jako jsou videohry, hledání a rozpoznávání vzorů v obrazu a lokomoce robotů. Úkoly, které mohou být obtížně řešitelné klasickými metodami řízení, lze nyní řešit bez nutnosti plného porozumění všem aspektům problému. Jednou z nevýhod reinforcement learningu v případě řešení úkolu kritického z hlediska bezpečnosti je, že neexistuje žádná záruka chování podle naučených zásad. U autonomně řízeného vozidla může mít jeden špatný krok fatální následky. V rámci této práce je ukázáno, že současné algoritmy reinforcement learningu jsou schopné se naučit řízení vozidla v simulovaném prostředí. Myšlenka inspirovaná člověkem rozdělení problému řízení na dvě různé dovednosti vykonávané současně urychlila učení a umožnila se soustředit na dva dílčí problémy. Ukázalo se, že navržené konvoluční neuronové sítě s pouze 4 vrstvami jsou dostatečné k realizaci jednoduchého autonomního řízení v omezených podmínkách. Modely byly trénovány pomocí pouze obrazových dat z kamerového senzoru. Obraz z kamery realisticky odráží výhled řidiče a tím simuluje situaci, kdy lidský řidič musí na základě svého výhledu vyhodnotit situaci a dynamicky na ní reagovat. Namísto lidského mozku, obraz zpracovává konvoluční neuronová síť, která je schopná identifikovat objekty a odlišit je jeden od druhého.

Fyzikální model vozidla využívající Newtonovi pohybové zákony simuluje působení vnějších sil na vozidlo včetně síly větru a valivého odporu. Vozidla používaná v silničním provozu jsou různá a

liší se zejména velikostí, výkonem a rozměry. Uživatelsky definovatelné fyzikální vlastnosti vozidla umožňují využít simulátor pro všechny značky, výkony a rozměry vozidel.

Autonomního řízení je složitý úkol, který se skládá z mnoha scénářů, které lze jen těžko popsat rovnicí. Jeho řešení se může vy výsledku lišit v závislosti na požadovaném stylu jízdy. Styl jízdy a chování ovlivňuje odměňovací funkce. Navržením funkce odměny dokážeme popsat optimální chování při řízení. Funkce odměň mohou být husté (dense) nebo řídké (sparse). Oba typy odměn jsou naučitelné. V práci je definovaných několik odměňovacích funkcí, které lze využít k naučení autonomního řízení. Výsledky experimentů demonstrují, že Q-Learning algoritmus, lze použít k naučení dovednosti řízení různých stylů jízdy. Všechny naučené modely se dokázaly zobecnit a jsou schopny odhadnout rozhodnutí řízení i v neznámém prostředí. Hlavní přínosem práce je zkoušení Q-learningu algoritmu s aplikací v autonomním řízení. Andrewův monotónní řetězový algoritmus je užitečný algoritmus, díky němuž lze procedurálně generovat tratě založené na konvexním obalu. Procedurálně generované tratě umožňují rychle a jednoduše generovat prostředí a simulovat jízdy různé složitosti a přitom sdílet informace o situaci kolem vozidla.

Literatura

1. *Levels of driving automation* [online] [cit. 2021-03-17]. Dostupné z: `https://www.sae.org`.
2. LUKE, Katie. *Three Approaches to Solving the Autonomous Vehicle Orientation Problem* [online]. 2019-10-08 [cit. 2021-04-16]. Dostupné z: `https://www.connected.io`.
3. *Markov Decision Processes (MDPs) - Structuring A Reinforcement Learning Problem* [online] [cit. 2021-03-17]. Dostupné z: `https://deeplizard.com`.
4. *Q-Learning Explained - A Reinforcement Learning Technique* [online] [cit. 2021-03-17]. Dostupné z: `https://deeplizard.com`.
5. *Q-Learning - Choosing Actions With An Epsilon Greedy Strategy* [online] [cit. 2021-03-17]. Dostupné z: `https://deeplizard.com`.
6. *Exploration Vs. Exploitation - Learning The Optimal Reinforcement Learning Policy* [online] [cit. 2021-03-17]. Dostupné z: `https://deeplizard.com`.
7. *Replay Memory Explained - Experience For Deep Q-Network Training* [online] [cit. 2021-03-17]. Dostupné z: `https://deeplizard.com`.
8. *IntelliJ IDEA* [online]. 2001 [cit. 2021-04-19]. Dostupné z: `https://www.jetbrains.com/idea`.
9. *Java* [online]. 1995 [cit. 2021-04-19]. Dostupné z: `https://www.java.com`.
10. *Deeplearning4j* [online]. 2014 [cit. 2021-04-05]. Dostupné z: `https://deeplearning4j.konduit.ai`.
11. *Git* [online]. 2005 [cit. 2021-04-01]. Dostupné z: `https://git-scm.com`.
12. BLOCH, Joshua. *Java efektivně: 57 zásad softwarového experta*. Moderní programování. Praha: Grada, 2002. ISBN 80-247-0416-1.
13. *Generating Procedural Racetracks* [online] [cit. 2021-04-16]. Dostupné z: `https://www.gamasutra.com`.
14. *Algorithm Implementation/Geometry/Convex hull/Monotone chain* [online] [cit. 2021-03-17]. Dostupné z: `https://en.wikibooks.org`.
15. *Car Physics for Games* [online] [cit. 2021-04-16]. Dostupné z: `https://asawicki.info`.

16. *Learning to Drive Smoothly in Minutes* [online]. 2019-01-27 [cit. 2021-04-16]. Dostupné z: `https://towardsdatascience.com`.

Příloha A

Videozáznamy z testovacích jízd

Byly pořízeny dva krátké záznamy z testovacích jízd demonstrující dva odlišné typy chování a styly jízdy. Videozáznamy jsou dostupné z odkazů uvedených v tabulce A.1.

Videozáznamy	
Chování	Odkazy
Závodní	https://www.youtube.com/watch?v=Cs5S3zdyUnw
Spolehlivé	https://www.youtube.com/watch?v=pq5Hz_pg-pQ

Tabulka A.1: Odkazy na videozáznamy demonstrující dva typy chování