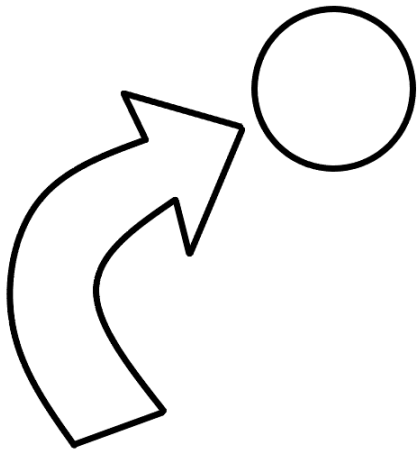
Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie

Predmet : **Inteligentné systémy a mobilná robotika**



Zadanie – pracovná skupina č. 6.

Vyhýbanie sa prekážkam pomocou DRL

kľúčové slová: Robotics, Navigation, Deep Reinforcement Learning, Obstacle Avoidance, DQN

Spracovali:
Ivan Tkachenko, Roman Dzhulai, Dmytro Varich, Anfisa Konycheva

Obsah

1 Úvod

Navigácia autonómnych robotov je kľúčovou súčasťou mnohých technologických aplikácií, od zásobovania až po prieskum neznámych prostredí. Hlavným cieľom je umožniť robotom efektívne sa pohybovať z počiatočného bodu do cieľa, zatiaľ aj sa vyhýbať statickým a dynamickým prekážkam.

V posledných rokoch bola veľká pozornosť venovaná vývoju algoritmov pre túto úlohu, pričom špeciálny dôraz bol kladený na využitie Hlbokého Zosilneného Učenia (Deep Reinforcement Learning, DRL)[1][2], ktoré umožňuje robotom učiť sa z vlastných skúseností a zlepšovať svoje rozhodovacie procesy v rámci procesu Markovovho rozhodovacieho procesu (Markov Decision Process, MDP)[3]. Hlboké zosilňovanie učenia, čo je časť strojového učenia, spájajúca hlboké učenie s zosilňovaním učenia, získalo význam pre svoj potenciál pri výcviku takýchto autonómnych agentov. Jedna z hlavných metód, používajúca Deep Q-siete (DQN)[4], sa zameriava na navigáciu mobilných robotov na základe vizuálnych pozorovaní. Tieto siete aproximujú Q-hodnoty pre rôzne akcie a stavy, čo umožňuje robotom robiť informované rozhodnutia v komplexných prostrediach.

Niekoľko výskumníkov sa snažilo aplikovať techniky hlbokého zosilňovania učenia na reálne navigačné situácie mobilných robotov, predovšetkým v interiéroch. Ich cieľom je zmenšiť medzeru medzi simuláciami a praktickými situáciami, zlepšujúc prispôsobivosť a účinnosť autonómnych robotov v reálnom svete. [5][6]

Tradičné metódy ako Simultánna lokalizácia a mapovanie (SLAM)[7] a optimizačné prístupy ako A-star[8] či RRT (Rapid exploration random tree)[9] boli efektívne na statických mapách, ale čelia obmedzeniam v dynamických prostrediach.

V tomto projekte sa sústredíme na vývoj a optimalizáciu algoritmu DRL pre navigáciu autonómneho robota v rôznorodých scenároch, kde sa kombinujú statické a dynamické prekážky, a to s cieľom zlepšiť úspešnosť navigácie, minimalizovať mieru kolízií a dosiahnuť efektívnejšiu cestu k stanovenému cieľu. Takéto systémy majú široké uplatnenie v priemyselnej automatizácii a reakcii na katastrofy, kde schopnosť navigácie v komplexných a nepredvídateľných prostrediach predstavuje kľúčový faktor.

1.1 Použite pojmy

Hlboké zosilnené učenie (DRL), často nazývané aj hlboké zosilňovanie učenia, je oblasť strojového učenia, ktorá kombinuje techniky hlbokého učenia s metódami zosilneného učenia. Hlavným cieľom DRL je vytvoriť agentov, ktorí sa môžu naučiť robiť rozhodnutia v dynamickom a zmenlivom prostredí s cieľom maximalizovať kumulatívne odmeny.

Markovov rozhodovací proces (MDP) je matematický model, ktorý sa používa na popis rozhodovacích problémov v situáciách, kde výsledok rozhodnutia závisí od aktuálneho stavu a vykonaných akcií, ale existuje aj istá miera náhodnosti.

Deep Q-Network (DQN) je algoritmus strojového učenia, ktorý sa používa na učenie hodnotiacich funkcií pre rozhodovacie procesy so správaním sa. Konkrétne ide o algoritmus používaný pre učenie zásad hlbokého zosilnenia (DRL), ktorý kombinuje hlboké neurónové siete s metódami zosilneného učenia, najmä s metódou Q-učenia.

Q-sieť (Q-network) je typ neurónovej siete používanej v hlbokom zosilňovaní učenia (DRL) na aproximáciu funkcie Q, ktorá hodnotí očakávanú odmenu pre vykonanie danej akcie v danom stave prostredia. Funkcia Q je kľúčovým konceptom v zosilnenom učení

a používa sa na rozhodovanie o optimálnych akciách, ktoré má agent vykonať v každom stave.

OpenAI Gym (Gymnasium) je knižnica, ktorá je zameraná na poskytovanie simulovaných prostredí pre tréning a testovanie algoritmov zosilneného učenia.

Rectified linear unit (ReLU) je aktivačná funkcia používaná v hlbokých neurónových sieťach. Jedná sa o jednoduchú a efektívnu aktivačnú funkciu, ktorá sa často používa v moderných architektúrach neurónových sietí. ReLU funkcia funguje tak, že vstupné hodnoty menšie alebo rovné nule sú transformované na nulu, zatiaľ čo všetky kladné hodnoty sú ponechané nezmenené. Matematicky môže byť ReLU definovaná ako f(x) = max(0, x), kde x je vstupná hodnota do aktivačnej funkcie.

Simultánne lokalizovanie a mapovanie (SLAM) je technika používaná v robotike a počítačovom videní na súčasné vytváranie mapy prostredia a určovanie polohy robota alebo kamery v tomto prostredí. Hlavným cieľom SLAM je umožniť robote alebo kamere pohybovať sa a súčasne si vytvárať mapu neznámeho prostredia bez predchádzajúcej znalosti o ňom.

Rapid Random Tree (RRT) je algoritmus používaný v robotike a automatizovanom plánovaní pohybu na riešenie problému plánovania cesty pre roboty a iné pohyblivé agenty. Jeho hlavným cieľom je rýchlo generovať rozsiahle stromové štruktúry, ktoré reprezentujú možné trajektórie pohybu v danom prostredí.

2 Popis konkrétneho problému

Tu predstavujeme detailný matematický rámec na riešenie navigačnej úlohy v 2D prostredí.

2.1 Problém

Reprezentácia prostredia je základom pre plánovanie a rozhodovanie mobilného robota. Prostredie je definované, ako 2D karteziánsky rovinový systém, kde každý objekt je geometricky reprezentovaný ako kruh. Robota aj ostatné entity v prostredí charakterizujú ich polohy, rýchlosti a ďalšie relevantné parametre.

Stav robota a ostatných entít je kritický pre proces rozhodovania. Stav robota zahŕňa jeho polohu, rýchlosť, natočenie a cieľovú pozíciu. Stav ostatných entít obsahuje ich polohy, rýchlosti, rozmery a minimálne vzdialenosti od robota.

Akcie robota sú zamerané na nastavenie jeho rýchlosti na navigáciu v prostredí. Funkcia odmeny vyhodnocuje následky týchto akcií, odmeňuje dosiahnutie cieľa, vyhýbanie sa kolíziám, udržiavanie bezpečnej vzdialenosti od prekážok a minimalizovanie nepohodlných situácií.

Cieľom je nájsť optimálnu riadiacu stratégiu, ktorá maximalizuje kumulatívne odmeny v čase. Hodnota funkcie V^* vyhodnocuje očakávané kumulatívne odmeny spojené s nasledovaním tejto optimálnej riadiacej stratégie, pričom zohľadňuje sa rôzne faktory, ako sú časový interval medzi akciami a faktor zľavy.

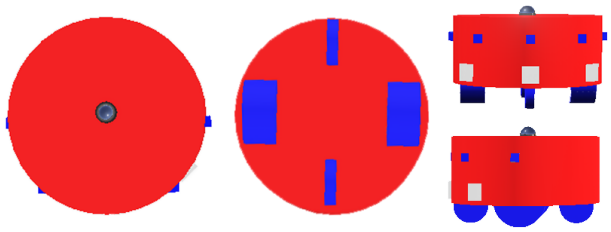
2.2 Metóda riešenia

Na riešenie navigačného problému využívame architektúru hlbokých Q-sietí (DQN), silný nástroj v oblasti zosilneného učenia. DQN sa iteratívne učí zo spätnej väzby prostredia, aby upravil svoj rozhodovací proces. Počas trénovacích epizód sa DQN model prispôsobuje svojej riadiacej stratégie, čo umožňuje robotovi účinne navigovať v

zložitých prostrediach, pričom maximalizuje kumulatívne odmeny.
Na trénovanie DQN siete budeme používať algoritmus zosilňovacieho učenia s názvom "Experience Replay". Tento algoritmus umožňuje robotovi učiť sa z minulých skúseností a zlepšovať svoje stratégie v priebehu času.

2.3 Model robota

Pri vytváraní modelu robota našou hlavnou úlohou bolo vyvinúť optimálneho robota v simulačnom prostredí Webots, aby sa mohol bez problémov pohybovať v svojom prostredí pomocou pohyblivých motorov, ktoré sú integrované vo forme kolies, a tiež ho vybaviť všetkými potrebnými senzormi, aby mohol analyzovať situáciu vo svojom prostredí, a tak prijímať ďalšie kroky pomocou hlbokého učenia.
Stojí za zmienku, že počas celej práce bol robot podrobený približne 15 zmenám pri tvorbe konečnej verzie robota, počas ktorých boli pridané nové senzory a ďalšie motory.
Konečná verzia robota má valcovitý tvar s červeným telom, vybaveným množstvom senzorov a pripojeným k telu modrými kolesami. Robot je vybavený 5 senzormi vzdialenosti, pričom každý z nich má dva lúče, a 3 senzormi nárazu, umiestnenými na prednej bočnej časti tela a rovnomerne rozmiestnenými po celej šírke tejto časti. Každý senzor zaberá približne rovnaké množstvo priestoru od okraja tela po jeho stred, čo zabezpečuje rovnomerné pokrytie prednej časti robota na detekciu prekážok a navigáciu. Okrem toho je k dispozícii jeden senzor umiestnený na vrchu v strede: GPS na určenie geografických súradníc robota vo svete a kompas na určenie smeru, ktorým sa pozerá robot, s cieľom zistiť odchýlku od cieľa.
Robot má štyri modré valcové kolesá pripevnené k telu pozdĺž okrajov a umiestnené paralelne navzájom. Slúžia na zabezpečenie jeho pohybu po povrchu. Bočné kolesá sú robustnejšie, čo im umožňuje pohybovať sa rýchlejšie, zatiaľ čo predné a zadné kolesá sú tenšie a poskytujú dostatok podpory a manévrovateľnosti pri pohybe. Na obrázku 1 je možné vidieť kompletný model konečnej verzie robota z rôznych uhlov pohľadu.



Obr. 1: Pohľad na model nášho robota

Aby náš robot vykonával príslušné funkcie svojho určenia, je v ňom zabudovaný personalizovaný kontrolér, ktorý nielen inicializuje naše zariadenia, ale aj priamo vykonáva funkcie a modely pre hlboké učenie. Pri inicializácii našich motorov sme sa rozhodli uľahčiť prácu vytvorením vlastnej triedy MotorSystem, v ktorom sa inicializuje každé jednotlivé koleso a tiež maximálnu rýchlosť pre každé koleso. V tejto triede sú tiež niekoľko metód, ktoré definujúce pohyb dopredu, otočenia doľava a doprava, zastavenie, ako aj funkcia apply_action, ktorá vykonáva určitú akciu v závislosti od argumentu akcie. Po inicializácii všetkých senzorov ich prenášame do triedy Environment, ktorá je určená na interakciu robota s okolitým prostredím.

2.4 Simulácia

Ako simulátor prostredia sme si vybrali Webots. Ako programovací jazyk sme zvolili Python, pretože to bol najvhodnejší jazyk na prácu s robotom, simuláciou a DRL

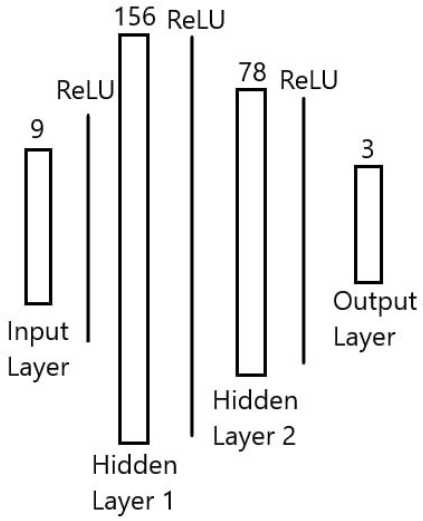
neurónovou sieťou súčasne. Model bol vyhynutý pomocou známej knižnice PyTorch. Všetky trénovania a experimenty prebiehali na počítači s grafickou kartou NVIDIA GeForce GT1030, ktorá umožňovala simuláciu v zrýchlenom režime. Približne 2,5-krát rýchlejšie, ako na priemernom CPU.

3 Návrh riešenia

3.1 Dáta zo senzorov

Každý typ senzorov implementovaný nami do modelu robota má svoj vlastný typ údajov, podľa ktorého je možné určiť jeho účel. V tabuľke 1 je uvedené, aký konkrétny senzor disponuje akým typom údajov, ako aj rozsah hodnôt od minimálnej po maximálnu hodnotu v nastaveniach senzorov pre naše účely. Senzory vzdialenosti sú reprezentované, ako desatinné čísla a ich hodnoty sa menia v závislosti od priblíženia objektu k lúčom, ktoré zachytili. Inými slovami, keď sa prekážka priblíži k senzoru, hodnota sa blíži k 0, a keď sa od neho vzdiali, hodnota sa vráti k 1000. Kolízne senzory alebo bumpery sú reprezentované, ako booleovská premenná, kde keď sa senzor zrazí s objektom, senzor sa spustí a vráti hodnotu True, inak False. Satelitný GPS vracia aktuálne merania 3D vektora vo forme zoznamu, ktorý udáva absolútnu polohu GPS zariadenia v nekonečnom rozsahu. Posledným senzorom, ktorý tiež vracia aktuálne 3D vektorové merania, ako zoznam, ako je GPS, je kompas. Jeho úlohou je vypočítať odchýlky od cieľa robota a vrátiť ich v radiánoch pomocou metódy get_deviation v triede Environment.

3.2 Model neurónovej siete



Obr. 2: Topológia modelu

Vstupná vrstva: Skladá sa z 9 neurónov, ktoré prijímajú vstupné dáta.
ReLU vrstvy: Nezávislé vrstvy typu ReLU aplikujú práh na vážený súčet vstupov z predchádzajúcej vrstvy pred prenosom na nasledujúcu vrstvu. ReLU neuróny uplatňujú prah na vážený súčet vstupov z predchádzajúcej vrstvy. V prípade kladného súčtu je výstup neurónu rovnaký, ako súčet, v prípade záporného súčtu je výstup nulový.
Skrytá vrstva 1: Obsahuje 78 neurónov, každý využívajúci ReLU aktivačnú funkciu.
Skrytá vrstva 2: Obsahuje 156 neurónov, každý využívajúci ReLU aktivačnú funkciu.
Výstupná vrstva: Obsahuje 3 neuróny, ktoré tvoria konečný výstup siete.

3.3 Funkcia odmeňovania

Pseudokód funkcie odmeňovania:

IF is_at_finish() **THEN**
  RETURN 400

Senzory	Typ dat	Rozsah hodnôt
Senzor vzdialenosti	float	$[0, 1000]$
Bumper	bool	$\{0, 1\}$
GPS	list[float]	$(-\infty, \infty)$
Kompas	list[float]	$(-\infty, \infty)$

Tabuľka 1: Popis senzorov a ich parametrov

```
ENDIF

IF is_collision() THEN
   RETURN -60
ENDIF

prev_distance = get_dist_to_finish(prev_pos)
curr_distance = get_dist_to_finish(curr_pos)

turn_coef = 1.0

reward = 10 * (prev_distance - curr_distance)

rad_deviation = get_deviation()
reward = (1/ABS(rad_deviation)) * 0.1

distances = get_values_from_distance_sensors()
d_effects = [d/2.5 FOR d IN distances]

FOR i=0 TO get_length(d_effects)
   IF d_effects[i] < 0.98:
      IF i == 0 OR i == 4 THEN
         d_effects[i] = d_effects[i] * 0.9 // almost
normal effect
      ENDIF
      IF i == 1 OR i == 3 THEN
         d_effects[i] = d_effects[i] * 0.8 // bigger
effect
      ENDIF
      IF i == 2 THEN
          d_effects[i] = d_effects[i] * 0.7 // the
biggest effect
      ENDIF
   ENDIF
ENDFOR

IF ABS(reward) < 0.03 THEN
   reward = 0.0
ELSE
   IF reward > 0.7 THEN
      a_sum = SUM(action_history)
      a_length = get_length(action_history)
      IF a_sum/a_length > 0.4 THEN
         turn_coef = (a_length/2.5) / a_sum
         reward = 0.7 * turn_coef
      ELSE:
         reward = 0.7
      ENDIF
   ENDIF
ENDIF

FOR effect IN d_effects
   reward = reward * e
ENDFOR

RETURN reward
```

V konkrétnom prípade, ak robot nedosiahne cieľ v danom počte krokov, dostane penalizáciu -650.

3.4 Environment

Environment je špecifická trieda, ktorá reprezentuje prostredie a funguje veľmi podobne, ako trieda Environment z knižnice Gymnasium. Naša trieda však namiesto simulácie celého prostredia hlavne ovláda robota, pozná aktuálny a predchádzajúci stav

robota a vypočíta odmenu za každý krok simulácie.

3.4.1 Stav

Obsahuje všetky senzory robota na získavanie informácií z nich. Na získanie aktuálneho stavu robota v simulácii máme metódu get_state(). Vráti zoznam hodnôt, ktorý obsahuje súradnice X a Y polohy robota, získané pomocou GPS modulu, uhlovú odchýlku od cieľa v radiánoch, 5 hodnôt zo senzorov vzdialenosti a hodnotu, ktorá ukazuje, či je robot práve v kolízii.

3.4.2 Odmena

Pomocou metódy get_reward() vypočítame výslednú odmenu pre robota pomocou aktuálneho stavu robota vo svete. Podrobný pseudokód funkcie odmeňovania je popísaný v bode 3.3.

3.4.3 Krok

Používame metódu step(action), aby robot vykonal danú akciu. Metóda využíva motorový systém robota na konfiguráciu motorov potrebným spôsobom, ktorý je definovaný parametrom action. Aktualizuje uloženú polohu robota. Používa metódu get_state() na získanie nového stavu robota po akcii, metódu get_reward() na výpočet odmeny pre robota za vykonanú akciu a výsledný stav robota.

3.5 Trénovanie modelu

Naše trénovacie dáta sme získali prostredníctvom simulácií v platforme Webots. Počas trénovania sme robota umiestnili do rôznych prostredí so zvyšujúcou sa mierou komplexnosti. Tento prístup nám umožnil vytvoriť model, ktorý je schopný efektívne navigovať v rôznych situáciách.

Najprv sme inicializovali konkrétny počet iterácií, ktoré robot dostal na dosiahnutie cieľa. Počet povolených iterácií sa časom znižuje, aby sa robot pokúsil dosiahnuť ciele efektívnejšie a nezneužíval špecifické mechanizmy vyhľadávania cieľov. Ak robot nedosiahne cieľ s daným počtom opakovaní, dostane penalizáciu.
Každou iteráciou uložíme aktuálny stav robota a pomocou neho získame novú akciu z Environment. Pomocou akcie robíme ďalší krok Environment. Tento krok dáva robotovi príkazy na pohyb vo zvolenom smere, ako aj vypočítava nový stav robota, ktorý sa teraz nazýva pozorovanie, odmenu, ktorú robot dostal za vykonanú akciu a boolovskú hodnotu terminated, ktorá ukazuje, či robot dosiahol cieľ.
Uložíme do pamäte predchádzajúci stav, zvolenú akciu, nasledujúci stav a odmenu na použitie modelom DQN. Optimalizujeme model pomocou uložených informácií a ukladáme spojenia do stavových slovníkov modelu. Ak robot v tejto iterácii dosiahol cieľ, ktorý môžeme skontrolovať pomocou premennej terminated, zastavíme aktuálnu epizódu tréningu a spustíme ďalšiu epizódu.

4 Experimenty a ich vyhodnotenie

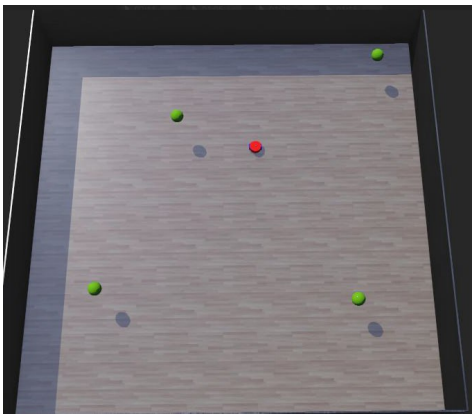
Naša práca zahŕňala niekoľko experimentov, ktoré sme vykonali na overenie výkonu nášho modelu.

4.1 Scenáre pre trénovanie a experimentovanie

4.1.1 Scenár bez prekážok

V scenári bez prekážok obr. 3 je robot umiestnený vo virtuálnom priestore, ktorý

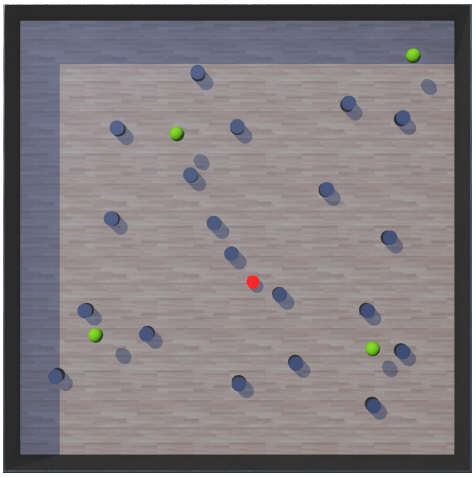
simuluje otvorenú plochu pre jeho tréning. Hlavným cieľom tohto scenára je umožniť robotu sústrediť sa na základné navigačné úlohy bez zbytočného vyrušenia. Červený objekt reprezentuje samotného robota, ktorý sa musí pohybovať v priestore. Okrem neho, v rôznych bodoch sú umiestnené zelené objekty, ktoré predstavujú ciele, ku ktorým bude robot smerovať.



Obr. 3: Scenár bez prekážok.

4.1.2 Jednoduchší scenár so statickými prekážkami

V druhom scenári obr. 4 sa robot ocitá vo svete so statickými prekážkami. Scenár zahŕňa tmavosivé kruhové objekty rozmiestnené po celej ploche, ktoré predstavujú statické prekážky. Robot, označený červeným objektom, musí navigovať priestorom a účinne sa vyhýbať týmto prekážkam. Zelené objekty, rovnaké, ako v prvom scenári, zostávajú v priestore, ako elementy, ktoré sú ciele, ku ktorým bude robot smerovať. Tento typ scenára je určený na tréning rozlišovania medzi navigovateľnými a nenavigovateľnými objektmi a zdokonaľovanie stratégií obchádzania prekážok.



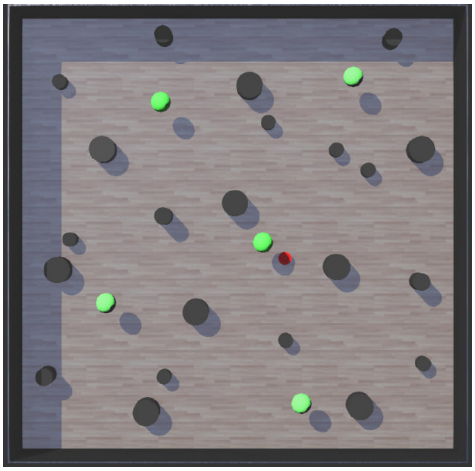
Obr. 4: Jednoduchší scenár so statickými prekážkami.

4.1.3 Scenár so statickými prekážkami rôznych veľkostí

V tomto scenári obr. 5 sa robot nachádza v prostredí so statickými prekážkami rôznych veľkostí. Tieto prekážky, zobrazené, ako šedé kruhy rozmiestnené náhodne po celej ploche, predstavujú rôznorodé objekty, ktoré robot musí identifikovať a obísť. Cieľom tohto scenára je vystaviť robota výzve navigácie v prostredí, kde je potrebné rozpoznať a prispôsobiť sa prekážkam rozličných rozmerov, čo je kľúčové pre aplikácie v dynamicky sa meniacich reálnych prostrediach.

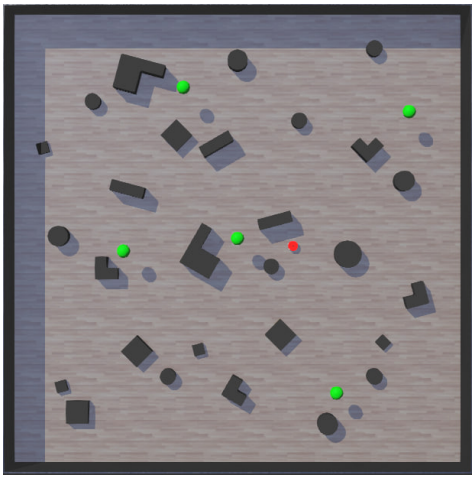
4.1.4 Scenár s vážnymi komplexnými statickými prekážkami

V tomto pokročilom scenári obr. 6 je robot postavený do komplexnejšieho prostredia so zložitejšími statickými prekážkami. Tieto prekážky sú reprezentované rôznymi šedými blokmi a kruhmi, ktoré sú rozmiestnené náhodne po celej ploche, vytvárajúc náročnejší labyrint pre navigáciu robota. Cieľom tohto scenára je otestovať schopnosť robota manévrovať v prostredí s viacerými



Obr. 5: Scenár so statickými prekážkami rôznych veľkostí.

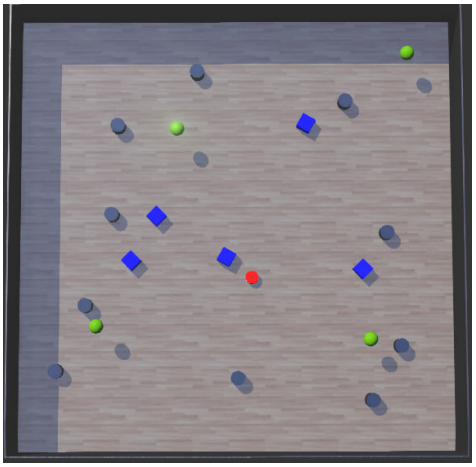
a zložitejšími statickými prekážkami, ktoré vyžadujú sofistikovanejšie stratégie plánovania trasy a rozhodovania.



Obr. 6: Scenár s vážnymi komplexnými statickými prekážkami.

4.1.5 Jednoduchší scenár s dynamickými prekážkami

V tomto scenári obr. 7 robot vstupuje do prostredia s dynamickými prekážkami. Modré bloky v priestore predstavujú prekážky schopné pohybu, ktoré sa robot musí naučiť predvídať a obchádzať. Tmavosivé kruhové objekty zostávajú statickými. Zelené objekty zase reprezentujú ciele pre robota. Takéto prostredie poskytuje robotu komplexnejší tréning, kde musí uplatniť rozšírené schopnosti navigácie, ako sú predikcia trajektórií pohybujúcich sa objektov a pružná adaptácia na meniace sa podmienky.

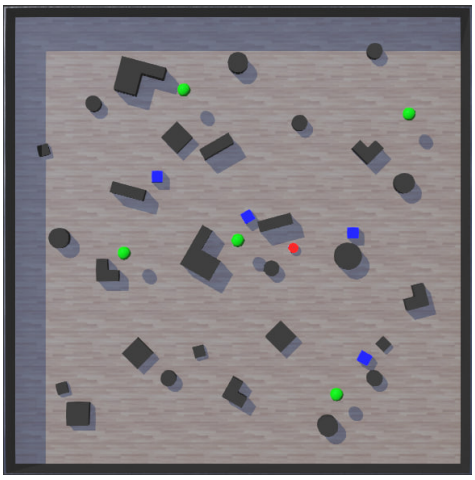


Obr. 7: Jednoduchší scenár s dynamickými prekážkami.

4.1.6 Najnáročnejší scenár s komplexnými statickými a dynamickými prekážkam

V tomto najnáročnejšom scenári obr. 8 robot čelí komplexnému prostrediu, kde sa kombinujú statické a dynamické prekážky. Tmavé bloky a kruhy predstavujú statické objekty, s ktorými musí robot počítať pri plánovaní trasy. Modré bloky naznačujú dynamické prekážky, ktoré môžu zmeniť svoju pozíciu alebo natočenie, čím pridávajú ďalšiu úroveň neistoty do procesu navigácie. Cieľom tohto scenára je vycvičiť robota, aby sa vedel prispôsobiť a efektívne navigovať v dynamickom prostredí, ktoré napodobňuje reálne

svetové podmienky s rôznymi nečakanými prekážkami.



Obr. 8: Najnáročnejší scenár s komplexnými statickými a dynamickými prekážkam.

4.2 Výsledky experimentov

Na základe našich experimentov z tabuľky 2 sme dospeli k záveru, že náš model dosahuje vysokú úspešnosť pri navigácii a vyhýbaní sa prekážkam v simulovanom prostredí ale stále má problémy v najťažších scenároch. Existujú isté oblasti, kde by sa mohol ešte zlepšiť, ako napríklad rýchlosť reakcie na neočakávané udalosti a niektoré ťažšie prípady s kombinovanými statickými a dynamickými prekážkami. Každý experiment trval približne 5-7 minút bez zrýchlenia.

5 Záver

Naša práca predstavuje dôležitý krok v oblasti využitia hlbokého zosilňovania učenia pri navigácii mobilných robotov. Navrhnutý model je schopný efektívne navigovať cez rôzne prostredia a vyhýbať sa prekážkam no zároveň má ťažkosti v konkrétnych prípadoch. Veríme, že naše výsledky môžu byť užitočné pre budúci výskum v tejto oblasti.

Jedným z možných spôsobov, ako vylepšiť náš model, je integrácia viacerých snímačov nárazov do celého robota, aby detegoval nárazy aj v prípadoch, keď nie sú čelné alebo v prípade dynamických prekážok nie sú priame. Je možné, že len zmeny princípov vo funkcii odmeňovania môžu viesť k lepším výsledkom. Tiež, ak je k dispozícii oveľa väčší výpočtový výkon, bolo by skvelé skúsiť použiť lidar s celým zorným polem a väčší model DQN. Mohlo by to viesť nielen k oveľa lepším výsledkom v scenároch s dynamickými prekážkami, ale aj k úplne iným princípom rozhodovania v rámci modelu. Výber cesty, kde je oveľa menej prekážok, absolútne vyhýbanie sa najťažším miestam scenára a oveľa viac. Tiež lepší výpočtový výkon by mohol viesť k oveľa rýchlejšiemu vývoju modelu, pretože tréning a testovanie by boli oveľa rýchlejšie.

Takže naša práca ponecháva veľa rôznych možností na zlepšenie pomocou väčšieho výpočtového výkonu, väčšieho modelu, nových rôznych senzorov a modulov alebo dokonca úplne iných prístupov.

Literatúra

[1] H. Liu, Y. Shen, S. Yu, Z. Gao, and T. Wu, "Deep reinforcement learning for mobile robot path planning," 2024.

[2] K. Li, Y. Xu, J. Wang, and M. Meng, "Sarl*: Deep reinforcement learning based human-aware navigation for mobile robot in indoor environments," 12 2019, pp. 688–694.

[3] F. Garcia and E. Rachelson, "Markov decision processes," *Markov Decision Pro-*

Scenár	**1**	**2**	**3**	**4**	**4**	**5**	**6**
Predtrénovaný model	Nie	Nie	Áno	Nie	Áno	Áno	Áno
Počet prekážok	0	20	22	28	28	20	32
Náročnosť scenára	Ľahký	Ľahký	Stredný	Ťažký	Ťažký	Stredný	Veľmi ťažký
Počet nárazov	0	0	1	8	2	0	6

Tabuľka 2: Porovnanie výsledkov v rôznych scenároch. Predtrénovaný model ukazuje, či bol na testovanie použitý predtrénovaný model na mape 2 (Áno/Nie). Počet prekážok udáva počet prekážok v testovanom prostredí. Náročnosť scenára značí náročnosť podmienok v simulovanom prostredí (Ľahký, Stredný, Ťažký, Veľmi ťažký). Počet nárazov vyjadruje, koľkokrát robil robot náraz do prekážky počas testovania.

cesses in Artificial Intelligence, pp. 1–38, 2013.

[4] M. Roderick, J. MacGlashan, and S. Tellex, "Implementing the deep q-network," 2017.

[5] N. Pico, B. Lee, E. Montero, M. Tadese, E. Auh, M. Doh, and H. Moon, "Static and dynamic collision avoidance for autonomous robot navigation in diverse scenarios based on deep reinforcement learning," in *2023 20th International Conference on Ubiquitous Robots (UR)*, 2023, pp. 281–286.

[6] S. Nam, C. Woo, S. Kang, T. A. Nguyen, and D. Min, "Slam-drlnav: A slam-enhanced deep reinforcement learning navigation framework for indoor self-driving," in *2023 International Conference on Mechatronics, Control and Robotics (ICMCR)*, 2023, pp. 44–48.

[7] H. Taheri and Z. C. Xia, "Slam; definition and evolution," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104032, 2021.

[8] S. Kuswadi, J. W. Santoso, M. Nasyir Tamara, and M. Nuh, "Application slam and path planning using a-star algorithm for mobile robot in indoor disaster area," in *2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, 2018, pp. 270–274.

[9] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1396–1402.