Navigacija mobilnih robota u gužvama pomoću dubokog potpornog učenja

Pavlović, Barbara

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 714

NAVIGACIJA MOBILNIH ROBOTA U GUŽVAMA POMOĆU DUBOKOG POTPORNOG UČENJA

Barbara Pavlović

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 714

NAVIGACIJA MOBILNIH ROBOTA U GUŽVAMA POMOĆU DUBOKOG POTPORNOG UČENJA

Barbara Pavlović

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 714

Pristupnica:	**Barbara Pavlović (0036526060)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentorica:	prof. dr. sc. Marija Seder

Zadatak: **Navigacija mobilnih robota u gužvama pomoću dubokog potpornog učenja**

Opis zadatka:

Autonomna navigacija mobilnih agenata izazovan je zadatak za robote koji rade u okolinama s dinamičkim objektima. Mobilnost mora biti učinkovita te društveno usklađena s obzirom na prirodu okoline. Prvenstveno, potrebno je da robot dospije do predodređenog cilja, poštivajući gibanja preostalih objekata, što je moguće modeliranjem interakcija i predviđanjem budućih trajektorija. Algoritmi dubokog potpornog učenja omogućuju odabir akcije koja približava robot cilju uz prilagođavanja implicitno predviđenih trajektorija preostalih objekata. Cilj diplomskog rada je implementirati algoritam dubokog potpornog učenja koji omogućuje autonomnu navigaciju uz prisutnost dinamičkih objekata. Algoritam je potrebno trenirati i validirati na raznim scenarijima kreiranim u simulaciji, uz potencijalnu implementaciju i testiranje na mobilnom robotu Husky.

Rok za predaju rada: 14. veljače 2025.

Zahvaljujem svojoj obitelji koja mi je omogućila školovanje i pružala bezuvjetnu podršku.

Zahvaljujem prijateljima koji su mi olakšali i uljepšali studiranje.

Sadržaj

1. Uvod

Svakodnevni razvoj umjetne inteligencije kojemu svjedočimo rezultira mogućnošću brze obrade podataka koja na mnoge načine može olakšati ljudski život. Neki od brojnih doprinosa umjetne inteligencije su da nam ona omogućuje prepoznavanje govora, zaštitu podataka i raspoznavanje različitih predmeta na slikama. Ovo tehničko područje svoju važnost dokazuje i time što umanjuje potrebnu količinu potrošenog vremena na repetitivni ljudski rad, ali i unaprjeđuje metode dijagnostike u medicini. Međutim, umjetna inteligencija svoju primjenu nije zaobišla ni u području robotike, što će ovaj rad i pokazati. Također će u radu biti predstavljeno duboko učenje i demonstrirana njegova primjena. Riječ je o grani umjetne inteligencije čiji algoritmi omogućuju autonomnim robotima nesmetanu i brzu navigaciju kroz gužvu, uz izbjegavanje bilo kakvog opasnog kontakta. Unatoč impresivnim rezultatima koji u tom području već i danas postoje, vidljive su prepreke na putu prema pronalasku idealnog rješenja, stoga je važno usredotočiti se na zaobilaženje istih. Glavni je cilj ovog rada demonstrirati neka od najpoznatijih i najuspješnijih postojećih rješenja za problem navigacije robota kroz gužve, pokazati glavne razlike između tih rješenja te njihove prednosti i nedostatke. Uspješnost rješenja koja će ovaj rad analizirati samo su dio dokaza da je daljnji razvoj umjetne inteligencije ključan za postojanje funkcionalnog suživota robota i ljudi.

1.1. Problematika i predmet istraživanja

Težak je zadatak ostvariti mobilnost robota koja je istovremeno učinkovita i prihvatljiva svim ljudima u njegovoj okolini. Elementarni je cilj robotu da se u gužvama kreće na udaljenosti ugodnoj ljudima, a istovremeno brzo stiže do svog cilja. On bi u ovakvim situacijama morao razumjeti ponašanje ljudi oko sebe kako bi ih znao zaobići. Kako bi to ostvario, mora naučiti predvidjeti kretnje koje bi mu potencijalno mogle dovesti

prepreku na put te se u skladu s time ponašati na način da izbjegne susrete na neugodnoj udaljenosti ili, u najgorem slučaju, kolizije. Očito je, zbog spomenutih izazova, da ovaj problem nije isključivo inženjerske prirode, već je potrebno razmotriti i uvide iz polja psihologije, kao i sociologije.

U početcima razvoja mobilnih robota u gužvama, u pristupima rješavanju ovog problema često se javljala pretpostavka da je gibanje ljudi u gužvi neovisno o sljedećem koraku robota. Kasnije se ipak u obzir počela uzimati činjenica da robot i ljudi jedni na druge utječu svojim radnjama, stoga se uvodi koncept predviđanja zajedničkog gibanja. Međutim, ako se pristup u slučaju prevelike kompleksnosti simuliranog okruženja previše usredotoči na neizvjesnost ljudskog ponašanja, postoji mogućnost da robot zapne u stanju takozvanog "zamrznutog robota" (engl. freezing robot problem). Robot u tom slučaju zaključi kako mu je najsigurnija opcija ostati na mjestu jer su sve radnje koje bi mogao poduzeti preopasne. Iz navedenog je razloga teško primjenjivati metode zasnovane na planiranju putanje, a one su i jako računski zahtjevne, posebno u slučaju velike gustoće gužve. Danas se, u svrhu rješavanja ovakvih problema, koriste metode potpornog učenja koje implicitno kodiraju interakcije među agentima, i na taj način postižu bolje rezultate i manju računsku složenost. Implicitni pristup problematici kojom se bavi ovaj rad, svijest o interakciji koja je ostvarena s okolinom ugrađuje u funkcije nagrade te na taj način usmjerava donošenje odluka. Navedena je problematika motivirala razvoj velike količine istraživanja, a to je dovelo do značajnog napretka u području robotike unutar posljednjih nekoliko desetljeća [1]. Modeli koji koriste ovakav pristup bit će predstavljeni i komparativno analizirani u ovom radu.

2. Strojno učenje

Strojno učenje je programiranje računala na način da optimiziraju neki kriterij uspješnosti na temelju podatkovnih primjera ili prethodnog iskustva [2]. Radi se o jednom od najbrže rastućih tehničkih područja danas koje predstavlja samu srž umjetne inteligencije i na sjecištu je računalne znanosti i statistike. Današnja široka dostupnost velike količine podataka na internetu potaknula je napredak u razvoju algoritama strojnog učenja [3]. Cilj je strojnog učenja optimizacijom parametara na temelju dobivenih podataka, izgraditi tim parametrima definirani model koji će dobro generalizirati. Model bi morao na temelju naučenoga iz dobivenih podataka, moći predvidjeti i objasniti još neviđene podatke, odnosno njihova svojstva. Primjena strojnog učenja prisutna je u raznim područjima života, kao što su: raspoznavanje uzoraka, obrada prirodnog jezika, prepoznavanje govora, računalni vid, ali doprinos je vidljiv i u području zdravstva, marketinga, proizvodnje, prometa, bankarstva i sl.

Razlikujemo tri temeljna pristupa strojnom učenju: nadzirano učenje, nenadzirano učenje i potporno učenje. Kod nadziranog učenja, model se uči uzorcima ulazno-izlaznih podataka te je njegov zadatak za dani ulaz pronaći odgovarajući izlaz. Nenadzirano učenje analizira neoznačene skupove podataka te iz njih izvlači najznačajnije uzorke. Treći se pristup naziva potporno (podržano/ojačano) učenje, a njega karakterizira učenje optimalne strategije na temelju pokušaja s određenom nagradom. Algoritmi potpornog učenja omogućuju procjenu vrijednosti poduzimanja pojedinih radnja u određenom okruženju, odnosno postizanje optimalnog ponašanja kako bi agenti bili sposobni gibati se na siguran i učinkovit način. Ovoj je vrsti učenja krajnji cilj iskoristiti znanje dobiveno istraživanjem okoline kako bi poduzimali radnje koje će im omogućiti dobitak najveće nagrade, odnosno maksimalno smanjenje kazne za pogreške prilikom gibanja. Navedeni način učenja iskorištava se, osim u robotici, i u zadacima proizvodnje i logistike u opskr-

bnom lancu [4]. Primjena potpornog učenja u robotici bit će istražena i predstavljena u ovom radu.

2.1. Duboko učenje

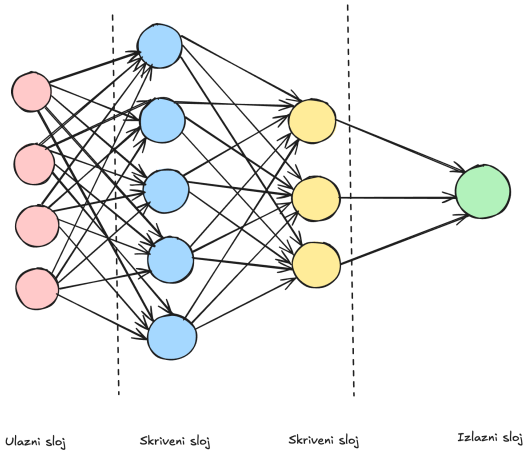
Algoritmi strojnog učenja nailaze na probleme u slučaju prisutnosti nestrukturiranih podataka, kao i kod rada s iznimno velikom količinom podataka, stoga se razvila grana učenja koja je za takve slučajeve specijalizirana te je u takvim situacijama uspješnija. Duboko učenje je grana strojnog učenja koja je posebno prikladna za rješavanje problema iz područja umjetne inteligencije. U algoritmima dubokog učenja, podaci se obrađuju kroz više slojeva, svaki sloj je zaslužan za izvlačenje značajki iz podataka, a onda te iste šalje daljnjim slojevima. Početni slojevi izvlače značajke niže razine, a kasniji ih kombiniraju te na taj način stvaraju potpunu reprezentaciju.

Duboko učenje može se implementirati koristeći različite arhitekture, kao što su: unaprijedne neuronske mreže, konvolucijske neuronske mreže, rekurzivne neuronske mreže i povratne neuronske mreže [5]. U ovome radu, modeli koji se analiziraju koriste unaprijedne neuronske mreže, kao što je višeslojni perceptron. Višeslojni perceptroni imaju jedan ulazni sloj, jedan izlazni sloj te jedan ili više slojeva u skrivenom sloju neurona (2.1.). Vidljivo je također da su izlazi neurona jednog sloja zapravo ulazi u drugi sloj mreže. Ovaj tip neuronske mreže ima tri glavna svojstva. Prvo je svojstvo nelinearna glatka izlazna karakteristika, koju omogućuje korištenje sigmoidne aktivacijske funkcije, no u ovom radu, neki su višeslojni perceptroni izrađeni s ReLu funkcijom (engl. Rectified Linear Unit). Radi se o aktivacijskoj linearnoj funkciji definiranoj izrazom:

$$A(Y) = \begin{cases} 0 & \text{ako } Y < 0, \\ Y & \text{ako } Y \geq 0 \end{cases} \tag{2.1}$$

U radu će biti spomenuta i primjena Leaky ReLu (engl. Leaky Rectified Linear Unit) funkcije. Drugo je svojstvo već spomenuto, mreža sadrži jedan ili više skrivenih slojeva neurona. Činjenica da je mreža dobro povezana čini treće svojstvo. Proces učenja u ovoj mreži zbog navedenih je karakteristika otežan, veliki broj neurona ga usporava, a mreža sama mora odlučiti što moraju naučiti skriveni neuroni, dok je za izlazni sloj

to zadano. Također, na izlazu mreže, za kriterij uspješnosti koristi se metoda računanja srednje kvadratne pogreške. Cilj učenja ovom metodom, odnosno povratnom propagacijom pogreške (engl. error back-propagation), minimizirati je srednju kvadratnu pogrešku ispravnim određivanjem težina u mreži.



Slika 2.1. Skica višeslojnog perceptrona

2.1.1. Duboko potporno učenje

Potporno učenje (engl. deep reinforcement learning) vrsta je strojnog učenja koju za potrebe navigacije koristi agent: robot kojemu je cilj kretati se kroz gužve ili čovjek koji je dio te gužve. Robotima potporno učenje služi kako bi naučili optimalnu politiku gibanja prema cilju uz izbjegavanje prepreka, metodom pokušaja i pogrešaka. Navedeni model učenja ima korijene u psihologiji i predstavlja temelj potpornog učenja [6]. Agent stupa u interakciju s okolinom poduzimajući radnju a_t u stanju s_t. Trenutačno stanje okoline služi tome da pruži informacije o okruženju u kojem se agent nalazi. Kada agent poduzme neku radnju, i okolina i agent prelaze u novo stanje s_{t+1} na temelju trenutačnog stanja i odabrane radnje. Okolina robota usmjerava prema odabiru najbolje (optimalne) radnje na način da prilikom prelaska u novo stanje, agent od svoje okoline dobiva i povratnu informaciju u obliku nagrade r_{t+1}.

Potporno učenje opisuje se kao Markovljev proces odlučivanja (MDP) koji se sastoji od: skupa stanja S, skupa radnji A, funkcije prijelaza $T(s_{t+1}|s_t, a_t)$, funkcije nagrade $R(s_t, a_t, s_{t+1})$ te faktora popusta $\gamma \in [0, 1]$ koji uravnotežuje odnos između trenutačnih nagrada i nagrada u budućnosti. Funkcija prijelaza određuje vjerojatnost prijelaza iz stanja s_t u stanje s_{t+1} odabirom radnje a_t, dok funkcija nagrade određuje dobivenu nagradu

navedenim prelaskom [6]. Agentu je cilj pronaći politiku koja postiže maksimalni povrat iz svih stanja, a njen je izraz sljedeći:

$$\pi^* = argmax_\pi E[R|\pi]. \tag{2.2}$$

U potpornom učenju agent pokušava maksimizirati nagradu koju dobiva kroz dvije temeljne faze: istraživanje (engl. exploration) i iskorištavanje (engl. exploitation). U fazi istraživanja, robot pretražuje stanja za koja mu nije poznato kakvu bi nagradu mogao dobiti, dok je u fazi iskorištavanja više usredotočen na poduzimanje radnji kako bi, na temelju informacija koje trenutačno posjeduje, mogao maksimizirati nagradu koju će dobiti. Faza iskorištavanja za robota predstavlja minimalnu količinu rizika, za razliku od istraživanja gdje poduzimanje rizika donosi znanje o okolini te se sprječava učenje pristrano malom broju iskustava.

Duboko potporno učenje, istraženo i predstavljeno u ovom radu, naziv je za vrstu potpornog učenja kod koje se za optimizaciju politike i funkcije prijelaza stanja koriste duboke neuronske mreže. Duboke mreže specifične su po tome što se svaki njihov sloj trenira zasebno. Posljedica navedenog svojstva je ta da su svi slojevi dobro istrenirani, čime se postiže i bolje izbjegavanje lokalnih optimuma. Spomenute činjenice omogućuju stvaranje kompleksnih mreža koje su sposobne za rješavanje težih zadataka [7]. Takve mreže u mogućnosti su otkriti bitne značajke u ulaznim podacima, koji najčešće sadrže informaciju o trenutačnom stanju okoline. Ovo je svojstvo važno u slučaju kompleksnih scenarija s velikim brojem ljudi u gužvi, čije značajke interakcije duboke neuronske mreže imaju zadatak izvući.

3. LM-SARL

Količina uslužnih robota u javnim ustanovama i na otvorenim prostorima se, uslijed napretka umjetne inteligencije i strojnog učenja, znatno povećava. U slučaju da se robot nađe u prostoru s velikom količinom ljudi u gužvi, brojni modeli kojima se on trenira kako bi tu mogao navigirati nailaze na probleme te podbacuju. Razlog tome je pojednostavljenje navedenog problema isključivo na jednosmjernu interakciju između čovjeka i robota. Pritom se zanemaruju interakcije među ljudima unutar same gužve, a one također mogu utjecati na robota, iz razloga što one mogu nagovijestiti buduća događanja u okolini. U prvome radu [8] koji spomenuti problem uzima u obzir te ga pokušava riješiti, implementiran je algoritam SARL (engl. Socially Attentive Reinforcement Learning). U spomenutom radu fokus izlazi izvan granica interakcije čovjeka i robota te se prelazi na interakciju čovjeka i gužve. Interakcija čovjeka i gužve modelirana je ugradnjom mehanizma samo-pažnje (engl. self-attention), pomoću kojega se važnost pojedinca u gužvi određuje na temelju procjene njegovih budućih stanja. Težine koje se na temelju navedenog mehanizma dodjeljuju svakom pojedincu (engl. attention weights), omogućuju robotu da procijeni koji bi mu se ljudski agenti u nekom trenutku mogli naći na putu, odnosno da bolje obrati pažnju na pojedine dinamičke prepreke koje mu predstavljaju potencijalnu opasnost. S obzirom na to da se interakcije između dvaju čovjeka modeliraju pomoću lokalnih mapa, puni naziv implementiranog modela naziva se LM-SARL (Local Map-SARL). Također, navedeni model, kako bi dozvolio ostvarenje realistične situacije u simulaciji i omogućio bolje razumijevanje ponašanja ljudi unutar gužve, može uzeti u obzir gužvu proizvoljne gustoće, odnosno bilo koji broj mobilnih agenata. Za svakoga je agenta (čovjeka i robota), poznata pozicija $p = [p_x, p_y]$, brzina $v = [v_x, v_y]$ i radijus r. Izrazom s_t označavamo stanje robota u trenutku t, a izrazom $w_t = [w_t^1, w_t^2, ..., w_t^n]$ označavamo stanje ljudskih agenata u trenutku t, stoga zajedničko stanje robota i ljudi u trenutku t označavamo izrazom: $s_t^{jn} = [s_t, w_t]$ [8].

LM-SARL je prvi put predstavljen na Međunarodnoj konferenciji o robotici i automatizaciji (engl. International Conference on Robotics and Automation, ICRA) 2019. godine te je u eksperimentima pokazano kako se radi o boljem algoritmu od tadašnjih najuspješnijih metoda (engl. state-of-the-art). Prednost algoritma je već spomenuto fokusiranje ne samo na odnos između čovjeka i robota, već i na odnose između svaka dva ljudska agenta te na predviđanje mogućih stanja u kojima se ti ljudski agenti mogu naći u idućem koraku.

Optimalna politika za navigaciju robota u ovoj okolini računa se prema sljedećem izrazu:

$$\pi^* = \underset{a_t}{\operatorname{argmax}}\, R(s_t^{jn}, a_t) + \gamma^{\Delta t * v_{pref}} * \int_{s_{S_t+\Delta t}} P(s_t^{jn}, a_t, s_{t+\Delta t}^{jn}) V^*(s_{t+\Delta t}^{jn}) d s_{t+\Delta t}^{jn}, \tag{3.1}$$

gdje je $R(s_t^{jn}, a_t)$ nagrada primljena u trenutku t, dok je V^* funkcija optimalne vrijednosti te se računa prema izrazu:

$$V^*(s_t^{jn}) = \sum_{t'=t}^{T} \gamma^{t' v_{pref}} R_{t'}(s_{t'}^{jn}, \pi^*(s_{t'}^{jn})). \tag{3.2}$$

Izraz $P(s_t^{jn}, a_t, s_{t+\Delta t}^{jn})$ predstavlja vjerojatnost prijelaza od trenutka t do trenutka $t+\Delta t$. Funkcija nagrade određuje za pojedine radnje iznos dobivene nagrade na način da kažnjava kolizije i preveliku blizinu drugom agentu, a nagrađuje dostizanje cilja na sljedeći način:

$$R_t(s_t^{jn}, a_t) = \begin{cases} -0.25 & \text{ako } d_t < 0, \\ -0.1 + d_t/2 & \text{ako } d_t < 0.2 \\ 1 & \text{ako } p_t = p_g \\ 0 & \text{inače} \end{cases} \tag{3.3}$$

gdje je d_t minimalna udaljenost između robota i ljudi u gužvi u periodu: $[t - \Delta t, t]$

[8]. Vrijednost 0 za nagradu robot dobiva u slučaju kada mu istekne dozvoljeno vrijeme navigacije.

Model je inicijaliziran učenjem oponašanjem koristeći skup iskustava od demonstratora. U ovom se slučaju demonstrator kreće prema politici algoritma ORCA. Inicijalizira se mreža vrijednosti, mreža ciljnih vrijednosti te memorija za ponavljanje iskustva. Nakon toga, model se konstantno poboljšava nadodavanjem znanja dobivenog iz iskustva samog robota u interakciji s okolinom (preprekama). Za razliku od načina na koji su to implementirali radovi koji su prethodili ovome, vrijednost sljedećeg stanja robot ne saznaje aproksimacijom linearnim modelom gibanja, već on od okoline dobiva informaciju u kojem se stanju nalazi na temelju radnje koju je poduzeo. Također, vrlo je važno precizno procijeniti optimalnu vrijednosnu funkciju V^*, koja implicitno kodira društvene odnose između pojedinih ljudskih agenata [8]. Dakle, LM-SARL se ne oslanja ni na kakva predefinirana pravila ljudskog ponašanja, već ih sam pokušava naučiti iz lokalnih mapa koje sadrže prikaz njihovih interakcija.

U ovome radu predstavljena je "društveno pažljiva" mreža koja je sastavljena od tri modula: interakcijski modul (engl. Interaction module), modul za grupiranje (engl. Pooling module) i modul za planiranje (engl. Planning module). Svaki od navedenih modula ima drukčiju, ali važnu ulogu. Interakcijski modul zaslužan je za modeliranje odnosa između agenata u simulaciji. Odnos između čovjeka i robota modelira se eksplicitno, a odnosi između pojedinih ljudskih agenata zabilježeni su u obliku lokalnih mapa. Razlog korištenja lokalnih mapa je zbog smanjenja kompleksnosti modeliranja odnosa među ljudskim agentima. Eksplicitno modeliranje odnosa među ljudima u gužvi dovodi do složenosti $O(N^2)$, što nije idealan slučaj, pogotovo kada je u simulaciji prisutan veliki broj ljudskih agenata. Ovdje korišteni mehanizam samo-pažnje, pomoću kojega se računaju vrijednosti pažnje usmjerene na ljudske agente, omogućuje modulu za grupiranje sažimanje svih primijećenih interakcija te njihovo prikazivanje pomoću vektora fiksne duljine. Na kraju, modul za planiranje važan je zbog procjene vrijednosti prethodno spomenutog zajedničkog stanja robota i ljudi (gužve) : $s_t^{jn} = [s_t, w_t]$ [8].

Prethodno spomenute lokalne mape grade se na način da, ako imamo susjedstvo koje se sastoji od L susjeda, gradimo tenzor dimenzije LxLx3 koji označavamo sa M_i koji je centriran u svakom ljudskom agentu i :

$$M_i(a, b, \cdot) = \sum_{j \in N_i} \delta_{ab}[x_j - x_i, y_j - y_i] w_y', \tag{3.4}$$

gdje je $w_y' = (v_{xj}, v_{yj}, 1)$ lokalni vektor stanja za ljudskog agenta j (jedan od susjeda agenta i), a brzine ljudskih agenata u navedenom izrazu računaju se isključivo u slučaju da je veličina kanala lokalne mape postavljena na vrijednost veću od 1. U ovome je radu za eksperimente veličina kanala postavljena na vrijednost 3, a ona određuje količinu informacije koja se zabilježava u lokalnoj mapi. Dakle, u ovom se slučaju bilježi informacija o tome je li određeni susjed nekog ljudskog agenta prisutan u zadanom prostoru unutar mape, ali bilježe se i brzine tog susjeda (u x i y smjeru), relativne u odnosu na promatranog agenta. Kanal veličine 2 bilježio bi samo informacije o relativnim brzinama. $\delta_{mn}[x_j - x_i, y_j - y_i]$ predstavlja indikatorsku funkciju koja iznosi 1 ako se relativna pozicija $(\Delta x, \Delta y)$ nalazi u prostoru mape zadanom koordinatama (a, b), a N_i je skupina susjeda oko i-te osobe. Konačno, stanje čovjeka i, tenzor mape i stanje robota združujemo u jedan vektor koristeći višeslojni perceptron (engl. multi-layer perceptron, MLP):

$$e_i = \phi_e(s, w_i, M_i; W_e), \tag{3.5}$$

gdje je $\phi_e(\cdot)$ funkcija sa ReLu svojstvima, dok su W_e pripadne težine. Navedeni vektor zatim se predaje sljedećem višeslojnom perceptronu kako bi se dobila značajka interakcije između robota i osobe i:

$$h_i = \psi_h(e_i; W_h), \tag{3.6}$$

,

gdje je $\psi_h(\cdot)$ potpuno povezani sloj sa ReLu nelinearnošću, a W_h su težine mreže [8].

Odbacivši tezu da je najbliži ljudski agent ujedno i najopasniji za moguću koliziju te da se njemu mora pridodati najveća pažnja, odnosno težina, ovaj rad koristi mehanizam samo-pažnje. On služi tome da se ipak u obzir uzmu i smjer i brzina gibanja nekog drugog, možda udaljenijeg, ali potencijalno opasnijeg pojedinca. Prethodno izračunati vektor e_i koristi se za računanje koeficijenta pažnje α_i na sljedeći način:

$$e_m = \frac{1}{n}\sum_{k=1}^{n} e_k, \tag{3.7}$$

,

$$\alpha_i = \psi_\alpha(e_i, e_m; W_\alpha), \tag{3.8}$$

gdje je e_m vektor fiksne duljine koji je dobiven prosječnim grupiranjem (engl. mean pooling) individualnih agenata, $\psi_\alpha(\cdot)$ je višeslojni perceptron (MLP) sa ReLu aktivacijama, dok su W_α težine. Konačna reprezentacija gužve donosi se sljedećim izrazom:

$$c = \sum_{i=i}^{n} softmax(\alpha_i)h_i. \tag{3.9}$$

Procjena vrijednosti (v) stanja olakšana je uz danu reprezentaciju gužve:

$$v = f_v(s, c; W_v), \tag{3.10}$$

gdje je $f_v(\cdot)$ višeslojni perceptron sa ReLu aktivacijama, dok W_v predstavlja težine [8].

3.1. Originalna implementacija

Treniranje modela započinje, kao što je već spomenuto, s 3000 epizoda učenja oponašanjem demonstratora. Ovo se postiže na način da se robot kreće kroz simuliranu okolinu vođen ORCA algoritmom, kasnije opisanim u 3.2., i nakon toga ažurira svoje pamćenje prikupljenim iskustvima. Politika je u sklopu učenja oponašanjem trenirana u 50 epoha stopom učenja jednakom vrijednosti 0.01. Slijedi potporno učenje koje se odvija treniranjem u nizu od 10.000 epizoda, sa stopom učenja 0.001 i faktorom popusta γ koji je jednak 0.9.

Nakon svake epizode izvršava se optimizacija grupa podataka (engl. batch optimization). Iz skupa podataka o iskustvima koji su spremljeni u memoriji, uzima se određeni

dio (u ovom slučaju uzima ih se 100), a onda se na temelju tih podataka uspoređuju predviđanja trenutačnog modela s ciljnim modelom. Podaci sadržavaju informacije o stanjima, poduzetim radnjama i nagradama, ali i informaciju o tome je li navigiranje završeno (cilj je dosegnut, isteklo je vrijeme ili se dogodila kolizija). Srednja kvadratna pogreška računa razliku između predviđanja tih dvaju modela, a onda se na temelju toga i ažuriraju težine mreže trenutačnog modela. Provodi se i validacija modela u 100 koraka, nakon svake 1000. epizode treniranja. U fazi validacije provjerava se koliko je dobra izvedba modela u trenutku kada se ona provodi. Nakon 50 epizoda treniranja odvija se i ažuriranje trenutačnog ciljnog modela, odnosno ciljni model svoje težine usklađuje s trenutačnim. Uloga je ciljnog modela da, prilikom predviđanja najbolje moguće sljedeće radnje, trenutačni model uspoređuje svoju odluku s njegovom te pokušava tu razliku minimizirati. Na taj se način trenutačni model konstantno poboljšava, učeći od vlastitih iskustava.

Vremensko ograničenje u kojemu agent (robot) ima vremena za doći do svog cilja postavljeno je na 25 sekundi. Za preveliku blizinu ljudskim agentima, robota se kažnjava faktorom vrijednosti 0.5. Agent koristi epsilon-pohlepnu metodu za balansiranje između istraživanja i iskorištavanja, dakle slučajno odabire između ta dva procesa. Vrijednost epsilona ovdje označava vjerojatnost da će agent odabrati istraživanje. Iz tog se razloga vrijednost epsilona smanjuje s brojem epizoda, odnosno cilj je ovog pristupa da, nakon što robot prikupi dovoljno iskustva i znanja o okolini, iskorištava te informacije kako bi znao gdje se kretati da bi dobio veći iznos nagrade. U početku treniranja ta vrijednost postavljena je na 0.5, a zatim linearno prelazi na 0.1 u prvih 4000 epizoda, nakon toga ostaje na 0.1. Prostor radnji koje robot smije poduzimati sastoji se od 80 diskretnih radnji: 5 brzina eksponencijalno su razmještene u intervalu : $(0, v_{pref}]$ te 16 mogućih orijentacija, odnosno usmjerenja podjednako su razmještena između vrijednosti: $[0, 2\pi)$. Navedeno vrijedi za holonomski model robota, a za model neholonomskog robota, na kojemu se i provode eksperimenti u ovom radu, dozvoljena usmjerenja su postavljena na interval: $[-\pi/6, \pi/6]$. Za neholonomskog robota originalno (u izvornom radu) je bilo zamišljeno da su mu dozvoljena usmjerenja postavljena na interval: $[-\pi/4, \pi/4]$.

3.2. Originalna simulacija

Za eksperimente koristi se simulirano okruženje (CrowdNav) u kojemu se robotski agent kreće uz zadane početne i ciljne pozicije pritom nastojeći izbjeći sudar s ljudskim agentima u gužvi. Ljudski su agenti smješteni u takozvanom "kružnom" scenariju (engl. circle crossing scenario), tako da je svaki od njih generiran na slučajno određenoj poziciji unutar kružnice zadanog radijusa (u slučaju implementacije testirane u ovom radu, iznos tog radijusa postavljen je na 4 m). Također je prilikom računanja njihovih početnih pozicija, vrijednostima dodan slučajan šum. Zadatak ljudskih agenata je gibanje prema suprotnoj strani iste kružnice. Prva implementacija LM-SARL-a, uspoređena je s tada najuspješnijim algoritmima za navigaciju robota, a to su: ORCA [9], LSTM-RL [10] i CADRL [11].

ORCA (engl. Optimal Reciprocal Collision Avoidance) je algoritam čija se glavna ideja sastoji u tome da svaki agent koji sudjeluje u interakciji, preuzima polovicu odgovornosti za izbjegavanje kolizije (sudara). Robot iskorištava poznatu informaciju o brzinama okolnih agenata kako bi izbjegao koliziju s njima te bira svoju brzinu iz skupa dozvoljenih brzina u kojemu za neke odabir nije dozvoljen zbog prisutnosti drugih agenata [9].
CADRL (engl. Collision Avoidance with Reinforcement Learning) je bio inspiracija za scenarije u kojima komunikacija ne može biti pouzdano uspostavljena, kao što je to slučaj, na primjer, u interakciji ljudskog i robotskog agenta. Prvi je pristup problemu kojim se bavi ovaj rad, u kojemu se primjenjuje potporno učenje te podržava zadavanje kinematskih ograničenja robotskog agenta. Međutim, navedeni model predstavlja algoritam za izbjegavanje kolizije između samo dva agenta te se fokusira samo na taj odnos, pritom ignorirajući ostale interakcije, što automatski dovodi do lošijih rezultata [11].
LSTM-RL, kao što ime sugerira, u svom pristupu koristi povratnu neuronsku mrežu LSTM (engl. Long Short-Term Memory). LSTM model važan je zbog korištenja memorijske ćelije koja predstavlja spremnik informacija, stoga se one tu mogu sačuvati na duži period vremena i tako učiti dugotrajne ovisnosti u slijednim podacima. LSTM-RL u svom pristupu odbacuje sve pretpostavke o tome kakav bi model ponašanja mogli slijediti ljudski agenti. Tako razvija bolju mogućnost generaliziranja jer ne očekuje pravila u radnjama drugih agenata. Ovaj model također omogućuje i promatranje proizvoljnog broja

agenata u okolini tako da se njihove pozicije kodiraju u vektor fiksne duljine i koristeći spomenutu memorijsku ćeliju te podatke dovoljno dugo pamtiti [10].

Temeljna razlika između ovih metoda i LM-SARL-a leži u primjeni modula grupiranja i interakcijskog modula, dok je modul planiranja ostao isti za sve metode u svrhu poštene usporedbe [8].

4. SG-D3QN

Brojna su postojeća rješenja za problem navigacije robota kroz gužvu, unatoč svojoj uspješnosti, računski vrlo zahtjevna. Ovime inspirirano bilo je potaknuto rješenje pod nazivom SG-D3QN (engl. Social Graph-based Dueling Double Deep Q-Network) model, originalno implementiran u [12]. Ovaj model dubokog podržanog učenja kreira učinkovitu grafičku reprezentaciju stanja robota i gužve (engl. graph attention network, GAT), a Q-vrijednosti računa učenjem dvostruke duboke Q-mreže (engl. deep Q network, DQN). Tako dobivene Q-vrijednosti su grube, stoga ih se modificira 'online' planiranjem, na temelju predviđanja zadanog broja mogućih budućih stanja.

Prostor dozvoljenih radnji za robota prilikom kretanja jednaki su kao i u prethodno opisanoj implementaciji u 3.1.

Kao što je već spomenuto, Q-mreža se koristi za računanje najboljih Q-vrijednosti te, samim time, odabir najbolje radnje. Ulaz u mrežu predstavljaju zajednička stanja robota i gužve. Cilj je, naravno, postići što bolju politiku navigacije, a njen je izraz u ovom slučaju sljedeći:

$$\pi^*(s_t) = \underset{a_t}{\operatorname{argmax}}\ Q^*(s_t, a_t), \tag{4.1}$$

gdje je a_t radnja u trenutku t, a s_t stanje u trenutku t. $Q^*(s_t, a_t)$ predstavlja optimalnu funkciju radnje i stanja, a njen je izraz sljedeći:

$$Q^*(s_t, a_t) = \sum_{s_{t+1}, r_t} P(s_{t+1}, r_t | s_t, a_t)[r + \gamma^{\Delta t * v_p} \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})], \tag{4.2}$$

gdje je s_t sljedeće stanje, a r_t je trenutačna nagrada. γ je faktor popusta koji je nor-

maliziran preferiranom brzinom v_p i vremenskim razmakom Δt. $P(s_{t+1}, r_t | s_t, a_t)$ opisuje vjerojatnost prijelaza iz jednoga stanja u drugo [12].

4.1. Funkcija nagrade

Funkcija nagrade dizajnirana je na način da se sastoji od tri dijela: R_g, R_c i R_s. R_g predstavlja dio nagrade koji je to veći što je robot bliže cilju pa ga na taj način potiče da nastavi navigirati prema njemu. R_c je dio funkcije nagrade koji služi za kažnjavanje u slučaju da se dogodila kolizija, a kazna je u originalnoj implementaciji algoritma [12] postavljena na vrijednost -2.5. R_s je dio funkcije dizajniran na način da nagrađuje robota kada uspije održati sigurnu udaljenost od svih ljudskih agenata. Funkcija nagrade implementirana za treniranje i testiranje modela u ovom radu zbroj je prethodno spomenutih komponenti, a definirana je sljedećim izrazom:

$$R^t = R_g^t + R_c^t + R_s^t, \tag{4.3}$$

a izrazi za R_g^t, R_c^t i R_s^t dani sljedećim izrazima:

$$R_g^t = \begin{cases} r_g & \text{ako } ||p_0^t - g_0|| < 0.2, \\ 0.1 * (||p_0^{t-1} - g_0|| - ||p_0^t - g_0||) & \text{inače} \end{cases} \tag{4.4}$$

$$R_c^t = \begin{cases} r_c & \text{ako } ||p_0^t - p_i^t|| < \rho_0 + \rho_i, \\ 0 & \text{inače ,} \end{cases} \tag{4.5}$$

$$R_s^t = \sum_{i=1}^{N} f(d_i^t, d_s) \tag{4.6}$$

$$f(d_i^t, d_s) = \begin{cases} d_i^t - d_s & \text{ako } d_i^t < 0.2, \\ 0 & \text{inače ,} \end{cases} \tag{4.7}$$

gdje su p_0 i g_0 pozicija i cilj robota, a ρ_0 i ρ_i redom radijus robota i radijus ljudskog

agenta s indeksom i. d_s označava minimalnu udaljenost na kojoj je robotu dozvoljeno biti od ljudskih agenata, a d_i^t označava njegovu stvarnu udaljenost od ljudskog agenta s indeksom i u trenutku t [12]. U ovom je radu r_g postavljen na vrijednost 0.25, r_c na -0.25, a d_s na 0.2.

4.2. Pristup

Predloženo razvojno okruženje u ovom modelu podijeljeno je na tri dijela. Već je spomenut dvoslojni GAT za reprezentaciju zajedničkog stanja robota i gužve, a drugi dio čini dvostruki DQN za procjenu Q-vrijednosti. Treći dio služi za predviđanje mogućih događaja u budućnosti na temelju kojih se donosi odluka o sljedećoj radnji, a taj zadatak obavlja takozvani 'online' planer [12].

U grafičkoj reprezentaciji stanja robota i ljudi kreiranoj pomoću GAT-a, čvorovi predstavljaju agente, a veze između čvorova opisuju kako su ti agenti povezani. Najprije se koriste višeslojni perceptroni kako bi generirali takozvano latentno (skriveno) stanje fiksne duljine, koje zatim predstavlja ulaz slijednih grafičkih konvolucijskih slojeva. Ovo se izvodi kako bi taj ulaz, neovisno o količini elemenata u stanju određenog agenta, bio kod svih jednake dimenzije. Latentno stanje čvora i prikazuje se sljedećim izrazima:

$$h_0 = \Psi_r(w_0; W_r) \tag{4.8}$$

$$h_i = \Psi_p(w_i; W_p) \tag{4.9}$$

gdje je w_0 puno stanje robota, w_i stanje ljudskog agenta, Ψ_r i Ψ_p su višeslojni perceptroni sa pripadnim težinama W_r i W_p. Ulaz u prvi konvolucijski sloj predstavlja izraz: $H^0 = [h_0^0, h_1^0, ..., h_N^0]$, a eksponent u izrazu označava redni broj sloja. Slijedi postupak koji služi modeliranju interakcija među agentima [12]. Iz dobivenog latentnog stanja se računa matrica relacija, odnosno povezanosti između agenata, pomoću funkcije sličnosti koja se računa na temelju dobivenih značajki iz prethodno dobivenih latentnih stanja. Za funkciju sličnosti između značajki uzima se postupak konkatenacije. Rezultat ove operacije predaje se aktivacijskoj funkciji Leaky ReLu, koja računa koeficijente pažnje

e_{ij}, što opisuje sljedeći izraz:

$$e_{ij} = LeakyReLu(A), \tag{4.10}$$

gdje je A matrica sličnosti izračunata na temelju spomenutog latentnog stanja. LeakyReLu (engl. Leaky Rectified Linear Unit) je aktivacijska funkcija te je varijacija poznate i prethodno spomenute funkcije ReLu. Ona je ovdje korištena kako se negativnim vrijednostima na ulazu ne bi dodijelila nula, već negativna vrijednost te se time postiže nelinearnost u neuronskoj mreži. Takva je aktivacijska funkcija važna jer pomaže pri učenju kompleksnih uzoraka u podacima. Vrijednosti e_{ij} zatim su normalizirane funkcijom $softmax$, čime nastaju težine pažnje (engl. attention weights) α_{ij}. Navedeno je vidljivo iz sljedećeg izraza:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})} \tag{4.11}$$

Prikazana težina označava važnost čvora j čvoru i. Svaki čvor zatim ažurira svoje stanje h_i^{l+1} dodavanjem stanja svojih susjeda ($N(i)$), množeći ih sa izračunatim težinama pažnje α_{ij}:

$$h_i^{l+1} = \sigma(\sum_{j \in N(i)} \alpha_{ij} h_j^l) \tag{4.12}$$

σ je također nelinearna aktivacijska funkcija.

GAT je dvoslojan zbog potrebe modeliranja indirektnih, ali i direktnih interakcija [12], tako da se njegov konačni izlaz zapisuje na sljedeći način:

$$H = h^0 + h^1 + h^2 \tag{4.13}$$

Za računanje Q-vrijednosti koristi se dvojna duboka Q-mreža. Kažemo da je dvojna jer se sastoji od dva odvojena toka, odnosno grane mreže. Jedan tok služi za procjenu vrijednosti stanja, a drugi se sastoji od funkcija za procjenu prednosti nekog stanja nad

drugima. Funkcija stanja i radnji prikazana je sljedećim izrazom:

$$Q(H, a; \alpha, \beta, \eta) = V(H; \alpha, \beta) + D(H, a; \alpha, \eta) \tag{4.14}$$

Za računanje skalara $V(H; \alpha, \beta)$ i $|A|$-dimenzionalnog vektora $D(H, a; \alpha, \eta)$ koristi se dvoslojni MLP (njegovi slojevi označavaju se sa: $\Psi_v(\cdot; \beta)$ i $\Psi_d(\cdot; \eta)$). Ovdje je H konačna grafička reprezentacija iz izraza 4.13, a α, β i η su parametri dvojne arhitekture [12]. U svakoj epizodi treniranja je stanje robota i gužve slučajno inicijalizirano. Dalje se treniranje odvija kao što je opisano u 3.1., osim postupka učenja oponašanjem i s drukčijim vrijednostima određenih parametara, što je detaljnije opisano kasnije, u poglavlju 6.

Problem postojanja grubih Q-vrijednosti postoji zbog nesavršenosti DQN modela u rješavanju problema navigacije kroz gužve. Uzrok ovome je nepoznavanje točne politike kretanja ljudskih agenata. Kako bi se navedeni problem riješio, naučeni se DQN kombinira sa online planiranjem, na način da se simuliraju moguća buduća stanja u odnosu na trenutačno. Dubina i širina takvog simuliranja označavaju se sa d i k, slijedno. Vrijednost k označava razinu na kojoj se navigacijska politika oslanja na potporno učenje bez modela, gdje vrijednost 1 predstavlja čisto potporno učenje bez modela [12]. Vrijednost parametra d jednako utječe na navigacijsku politiku, a označava razinu uravnoteženosti važnosti trenutačnog i budućih stanja. Razlog ovome je činjenica da, ako parametre k i d postavimo na manje vrijednosti, tada robot samo traži trenutačnu najbolju vrijednost, ne istražuje suboptimalna rješenja i ne iskorištava mogućnost modela da procjenjuje moguće vrijednosti određenih stanja u budućnosti.

Naučeni DQN model i model okoline primjenjuju se rekurzivno za izgradnju stabla predviđanja. Rekurzija ima prethodno spomenutu dubinu te, što je ona veća, predviđa se veći broj mogućih stanja u budućnosti. Poboljšanje Q-vrijednosti opisuje se sljedećim izrazom:

$$Q^d(s_t, a_t; \theta) = \begin{cases} Q(s_t, a_t; \theta), & \text{ako } d = 0, \\ \frac{d}{d+1} Q(s_t, a_t; \theta) + \frac{1}{d+1}(r^{d-1} + \gamma^{\Delta t v_p} \max_{a_{t+1}} Q^{d-1}(s_{t+1}, a_{t+1}; \theta)), & \text{inače.} \end{cases} \tag{4.15}$$

Skriveni neuroni $\Psi_r(\cdot)$, $\Psi_h(\cdot)$, Ψ_r, Ψ_v, i Ψ_d imaju sljedeće dimenzije: (64, 32), (64, 32), (128), (128), i (128). Izlazne dimenzije u svakom sloju od $\Psi_q(\cdot)$ i $\Psi_k(\cdot)$ su jednake 32. U korištenoj arhitekturi, Ψ_r, Ψ_v, i Ψ_d imaju sljedeće dimenzije izlaza: 128, 1, i 81 [12].

5. Intrinzične nagrade

U takozvanoj Teoriji samoodređenja (Self-Determination Theory), navodi se da ljude na radnju potiču dvije vrste motivacija, a to su intrinzične i ekstrinzične [13]. Intrinzična se odnosi na motiviranost time što će nas poduzeta radnja zabaviti, odnosno trenutno učiniti sretnima, poduzimamo je iz znatiželje. U slučaju ekstrinzične motivacije, radnja se poduzima jer je poznata "nagrada" koju ćemo dobiti ako se na nju odlučimo, ona će nam donijeti dobro koje nam je poznato.
Čak i najčešće korišteni algoritmi za navigaciju robota kroz gužve, poput prethodno opisanog LM-SARL-a, robota znaju dovesti u stanje zarobljenosti u lokalnom optimumu, zbog premalo istraživanja prostora i nepoznatih stanja koja donose još nepoznate nagrade. Iz tog je razloga važno potaknuti ga na istraživanje i u trenucima kada je blizu dostizanja svog cilja. Na taj način robot može naučiti generalizirati, odnosno usvojiti vještine u kretanju koje će mu u budućnosti pomoći da se snađe u nepoznatim, dotad neviđenim okolinama. Postoje brojni pristupi kojima se pokušava postići ravnoteža između istraživanja i iskorištavanja poticanjem robota na istraživanje nepoznatih područja dodjeljivanjem intrinzičnih nagrada. Neki od njih su: strategija VIME (engl. Variational Information Maximizing Exploration) [14], strategija ICM (engl. Intrinsic Curiosity Module) [15], tehnika destilacije nasumične mreže (engl. random network distillation (RND)) uz kombiniranje intrizičnih i ekstrinzičnih nagrada [16], RE3 (engl. Random Encoders for Efficient Exploration) koji kao intrinzičnu nagradu koristi entropiju stanja [17].

5.1. Ideja i implemetacija

U ovom radu istražena je implementacija dviju strategija računanja intrinzičnih nagrada: ICM [15] i RE3 [17]. Radi se o nadogradnji algoritama unutar široko korištene simula-

cije okruženja (CrowdNav), već opisane u 3.2. U navedenom je radu tadašnjim najčešće korištenim i najuspješnijim algoritmima, među kojima je i LM-SARL, implementirana tehnika nagrađivanja robota izmijenjena dodavanjem intrinzičnih nagrada spomenutim dvama strategijama [18].

ICM metoda generira intrinzičnu nagradu s obzirom na to koliko je teško robotu predvidjeti sljedeće stanje na temelju trenutačnog stanja i radnje koju poduzima. Predstavlja se učenje prostora značajki treniranjem duboke neuronske mreže sa dva podmodula: prvi kodira stanje s_t u vektor značajki $\phi(s_t)$ i pokušava točno predvidjeti vektor značajki stanja $\hat{\phi}(s_{t+1})$ do kojeg dovodi radnja a_t, a drugi koristi stanja u prostoru značajki $\phi(s_t)$ i $\phi(s_{t+1})$ za predviđanje radnje koja je dovela do tog stanja, $\hat{a}_t$. Redom ih nazivamo unaprijednim i inverznim dinamičkim modelom. Dakle, inverzni model služi za učenje prostora značajki koji kodira informaciju važnu za predviđanje radnji robota, a unaprijedni model radi predviđanja unutar ovog prostora značajki [18]. Intrinzična nagrada se, prema tome, računa sljedećim izrazom:

$$r_{in} = MSE(\phi(s_{t+1}), \hat{\phi}(s_{t+1})), \tag{5.1}$$

gdje je MSE srednja kvadratna pogreška [15].

RE3 metoda koristi neuronsku mrežu f_θ inicijaliziranu slučajnim težinama θ, koje se ne ispravljaju prilikom treniranja algoritma potpornog učenja. Za računanje intrinzične nagrade koristi se metoda k-najbližih susjeda koja služi za procjenu entropije određenog stanja. "Susjedi" u ovom kontekstu predstavljaju prethodna iskustva robota u stanjima koja su, unutar reprezentacijskog prostora stanja, bliska promatranome. Temeljna je pretpostavka ovog pristupa da, prostor stanja kreiran neuronskom mrežom sa slučajno inicijaliziranom težinama, učinkovito zabilježava informacije o sličnostima stanja i bez učenja reprezentiranja stanja [17]. Što je veća izračunata entropija stanja, dakle, što je stanje različitije od svojih susjeda, to će biti veća intrinzična nagrada za robota jer to znači da je stanje manje poznato. Ovdje se intrinzična nagrada računa prema sljedećem izrazu:

$$r_{in}(s_i) = log(||f_\theta(s_i) - f_\theta(s_i)^{k-NN}||_2 + 1), \tag{5.2}$$

gdje izraz $f_\theta(s_i)^{k-NN}$ predstavlja k najbližih susjeda (engl. k-nearest neighbours) stanja s_i.

Kod obiju metoda, sveukupna se nagrada računa prema izrazu:

$$r = r_{ex} + \beta * r_{in}, \tag{5.3}$$

gdje je r_{ex} ekstrinzična nagrada dobivena od okoline, a β je hiperparametar koji kontrolira utjecaj intrinzične nagrade na ukupnu i on je postavljen kod obiju metoda na vrijednost 0.2, s kojom su postignuti najbolji rezultati.

Ovdje je iskorištena CrowdNav okolina za simulaciju navigacije robota kroz gužvu, uz određene izmjene, kao što su uklanjanje procesa učenja oponašanjem, izmjena vrijednosti određenih parametara te promjena određivanja ekstrinzične nagrade (opisano u 4.1.). Sve su izmjene treniranja detaljnije opisane u 6. poglavlju. Također, ovdje se za robota pretpostavlja model monocikla (engl. unicycle), dakle ne može se slobodno gibati u svim smjerovima. Skup dozvoljenih radnji ovdje je jednak kao i prethodno navedeni u 3.1. Prijelazni model je za svakog agenta definiran sljedećim izrazima:

$$\theta_{t+1} = \theta_t + \delta_t \tag{5.4}$$

$$v_t = [v_t \cos\theta_t,\ v_t \sin\theta_t] \tag{5.5}$$

$$p_{t+1} = [p_t + v_t \Delta t], \tag{5.6}$$

gdje je vremenski interval postavljen između dva vremenska koraka jednak Δt, θ je kut smjera, a p_t je pozicija u trenutku t. v_t predstavlja brzinu agenta u trenutku t [18].

6. Provedeni eksperimenti

Koristeći različite eksperimente, uspoređuje se originalna implementacija LM-SARL-a s modelima koji ne primjenjuju učenje oponašanjem, a imaju funkciju nagrade iz potpoglavlja 4.1. Postupak treniranja kod novijih modela, izuzev učenju oponašanjem, sličan je opisanome u potpoglavlju 3.1., uz izmjenu vrijednosti parametara koji će sada biti navedeni. Validacija modela izvršava se nakon svako 500. epizode treniranja, kao i ažuriranje trenutačnog ciljnog modela. Postupak optimizacije grupe podataka izvršava se na skupu od 50 podataka. Prezentira se i testira unaprjeđenje algoritma LM-SARL implementacijom intrinzičnih nagrada, a zatim se i uspoređuje izvedba takvih modela s originalnim. Međutim, uspoređuje se i izvedba SG-D3QN algoritma u odnosu na različite verzije LM-SARL algoritma. Modeli se treniraju na jednakom skupu podataka, odnosno početne i ciljne pozicije, kao i brzine agenata te svi ostali njihovi atributi u svim su epizodama treniranja i testiranja postavljeni jednako, kako bi se provela poštena usporedba. Također, unatoč tome što je za originalnu implementaciju LM-SARL algoritma postavljeno vremensko ograničenje za dolazak do cilja na 25 sekundi, u eksperimentima je svim modelima zadano ograničenje od 30 sekundi. Također, kutevi dozvoljenih rotacija za robota postavljeni su u interval $[-\pi/6, \pi/6]$. Modeli su potpuno neovisni o okolini u kojoj ih se trenira, odnosno promjene u simulaciji ne zahtijevaju nikakve promjene u samim modelima. Ovu činjenicu iskorištavamo kako bismo testirali izvedbu ispitanih modela na različitim postavkama okoline. U radu se prezentiraju prednosti i nedostaci ispitanih modela, ističu ključne razlike i donosi zaključak o mogućnosti primjene u stvarnom životu.

Provodi se nekoliko različitih postupaka treniranja: robot je u jednom postupku vidljiv ljudskim agentima, u drugome nije, a treći postupak ljudskim agentima slučajno dodjeljuje preferirane brzine gibanja i radijus. Na modelima treniranima prvim navedenim

postupkom izvode se svi eksperimenti, dok ostali postupci služe za usporedbu izvedbe modela u drukčijim okolnostima. Kada robot nije vidljiv ljudskim agentima, oni se kreću na način neovisan o poziciji robota i njegovim mogućim budućim kretnjama. Veća realističnost postignuta je u scenariju s vidljivim robotom, gdje robot mora ostvariti interakciju s ljudskim agentima, a ne samo razumjeti ponašanje gužve, kao u prethodnom slučaju. Scenarij sa slučajnim ljudskim atributima također pruža veću količinu realističnosti u simuliranim situacijama, stoga je i on koristan za detaljniju i točniju evaluaciju izvedbe modela. Izvodi se i analiza postupka treniranja pomoću grafova (engl. learning curve), a zatim i testiranje izvedbi modela u različitim, još neviđenim scenarijima te se na taj način ispituje sposobnost pojedinog modela da generalizira. U odnosu na originalne postavke u simulaciji, testira se izvedba modela uvođenjem promjena ponašanja ljudskih agenata na način da se postigne simulacija gibanja sličnijeg prirodnome, a uvode se i statičke prepreke. Simuliraju se situacije u kojoj svi ljudski agenti imaju različite, slučajno generirane atribute, a i robotu se postavljaju drukčije početne i ciljne pozicije. Izvedba se testira i na način da se 'senzoru' koji robot ima dodaje šum.

Osim testiranja izvedbi modela u različitim scenarijima, provodi se i analiza zaglađenosti putanje kojom se robot kreće u jednoj od simulacija, a kasnije se ispituje u kojoj se mjeri robot kreće na siguran način za svoju okolinu, odnosno koliko se često našao na opasno maloj udaljenosti od ljudskog agenta. Na temelju rezultata dobivenih iz opisanih testova i eksperimenata, zaključuje se koji je model najuspješniji i mogu li se neki od njih primijeniti na fizičkom robotu.

6.1. Implementacijski detalji i korišteni alati u simulaciji

Za provođenje eksperimenata, kako bi se usporedile prethodno predstavljene metode za navigaciju robota kroz gužve, korištena je već spomenuta CrowdNav okolina implementirana u programskom jeziku Python. Za programsku izgradnju dubokih neuronskih mreža, koristi se sučelje PyTorch. Podaci o okolini (pozicija i brzina agenata) određeni su programski, odnosno izvedeni iz programski simuliranog gibanja kroz zadani prostor. Svi atributi agenata bilježe se kao varijable klasa koje opisuju robota i čovjeka. Unutar tih klasa razlikujemo skup varijabli koji mogu vidjeti i drugi agenti (engl. observable

state) te onaj koji je poznat samo tom agentu (engl. full state). Zadavanjem različitih argumenata prilikom pokretanja treniranja moguće je bilo zadati određene karakteristike simulacije, kao što je raspored ljudskih agenata u prostoru (kružni ili kvadratni), broj ljudskih agenata u gužvi, politika koju slijedi robot prilikom navigacije (algoritam potpornog učenja) i slične postavke za okolinu. Scenariji za ispitivanje generalizacije također su bili aktivirani na način da se zadaju određeni argumenti prilikom pokretanja testiranja.

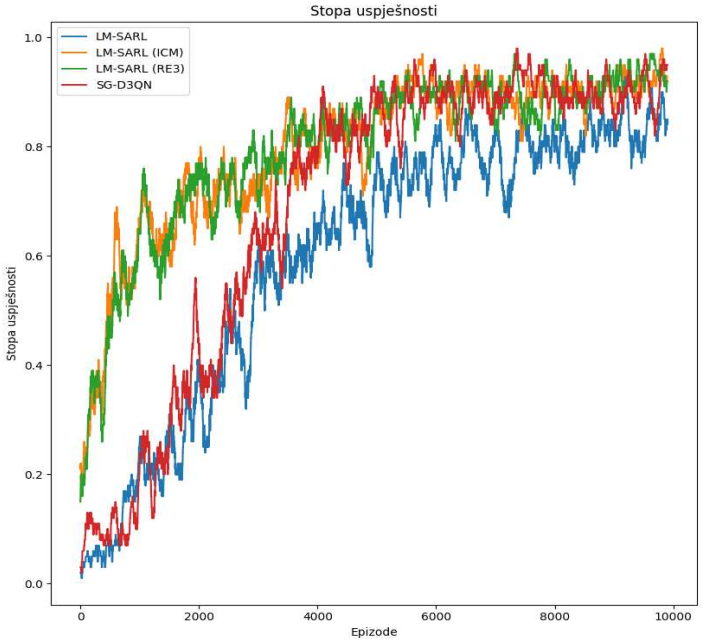
U svakoj epizodi treniranja i testiranja, ljudski agenti imaju slučajno postavljene pozicije, a eksperimenti se provode u "kompleksnom scenariju". U svakoj epizodi treniranja 10 se ljudskih agenata postavlja na slučajne početne pozicije te im se slučajno određuju i ciljne pozicije, u takozvanom "kružnom scenariju" (engl. circle-crossing scenario). 5 je ljudskih agenata postavljeno unutar kružnice radijusa 4 m i moraju proći kroz njen centar, a ostalih 5 agenata smješteno je unutar granica kvadrata sa zadanim stranicama duljine 10 m (engl. square-crossing scenario). U ovom kvadratnom scenariju, ciljne se pozicije također određuju unutar zadanih dimenzija kvadrata te su također zadane slučajno. Ljudski agenti u simulaciji se nikada ne prestaju gibati i navedeno je postignuto na način da, kada dođu do svog zadanog cilja, zadaje im se nova, slučajna ciljna pozicija prema kojoj se kreću. U simulaciji je moguće zadati proizvoljan broj ljudskih agenata, kako bi se eksperimentiralo s različitim gustoćama gužve. U ovom radu provedeni su samo eksperimenti s 10 ljudskih agenata, s obzirom na činjenicu da su modeli prilagođeni toj (ili manjoj) gustoći gužve. Također, radijus ljudskih agenata postavljen je na vrijednost 0.3 m. Oni u ovdje korištenoj simulaciji izbjegavaju međusobne kolizije koristeći ORCA algoritam. Točnije, koristi se centralizirana verzija ove metode, što znači da svaki ljudski agent ne računa svoju brzinu na temelju položaja i brzine drugih agenata, već postoji središnji sustav koji koordinira sve agente istovremeno.

Robotu su početna i ciljna pozicija u svakoj epizodi treniranja jednake, nalazi se na granici zadane kružnice (njegove su koordinate $(0, -r)$, gdje je r radijus kružnice), a cilj mu je prijeći na suprotnu stranu kružnice (dakle, na poziciju $(0, r)$). U testovima je ispitana učinkovitost modela u slučaju kada su mu početne i krajnje pozicije zadane drukčije, odnosno na slučajno generiranim pozicijama na zadanoj kružnici. 'Senzoru' kojim robot promatra okolinu, odnosno ljudske agente oko sebe, doseg vidljivosti postavljen je

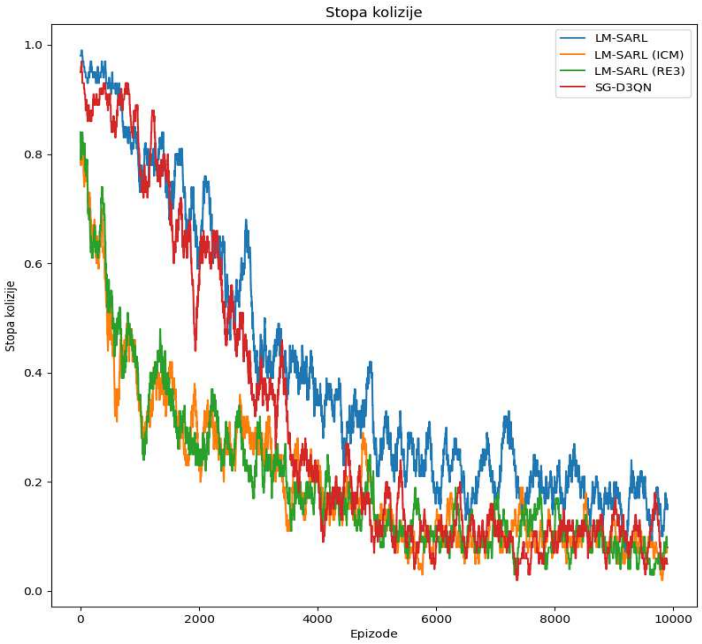
na 4 m. Kako bi se za pojedinog ljudskog agenta provjerilo nalazi li se unutar dosega senzora, računa se udaljenost između robota i agenta te ako je ona manja od dosega, bilježi se stanje primijećenog ljudskog agenta. Radijus robota postavljen je, kao i kod ljudskih agenata, na vrijednost 0.3 m. Svim je agentima postavljena ciljna brzina gibanja te je jednaka 1 m/s. Robotu je, dakle, zadano da mora do cilja doći u roku od 30 sekundi.

6.2. Analiza treniranja

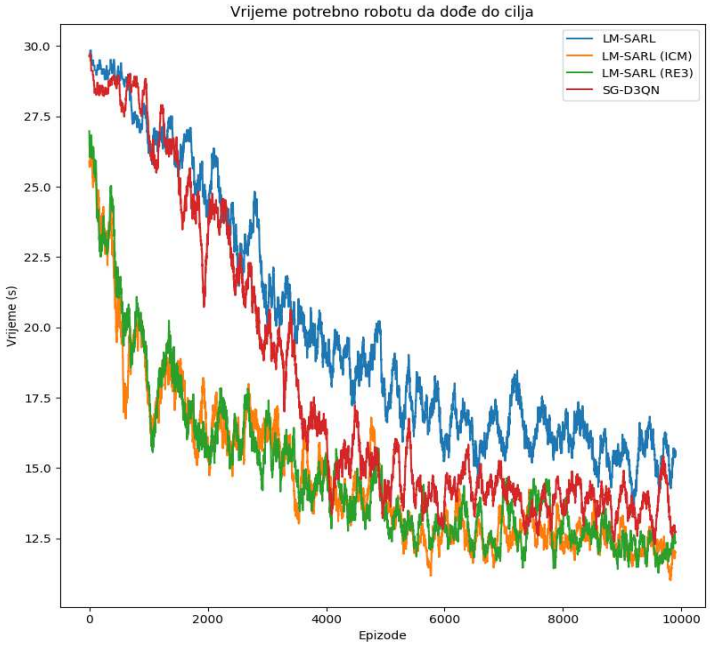
U ovom poglavlju prikazani su grafovi koji prezentiraju kvalitetu modela u smislu stope uspješnosti, učestalosti kolizija, trajanja navigacije robota i dobivene nagrade kroz period od 10.000 epizoda treniranja. Prikazuju se krivulje učenja modela s različitim varijantama LM-SARL algoritma, odnosno s originalnim algoritmom (ali iz [18]) i onim s intrinzičnim nagradama (ICM i RE3 pristup), a i krivulja učenja za SG-D3QN algoritam. Svi navedeni modeli za koje se prikazuju krivulje trenirali su se postupkom opisanim u poglavlju 6. Ovdje se analiziraju treniranja izvedena u simulaciji u kojoj je robot nevidljiv ljudskim agentima, a zatim se prikazuje i rezultat testa u istoj simulaciji nakon treniranja u varijanti s vidljivim robotom. Iako scenarij u kojem robot nije vidljiv ljudskim agentima ne predstavlja realističnu situaciju, na taj smo način osigurali da se robot sam mora pobrinuti za sigurnu kretnju i izbjegavanje kolizija, bez suradnje s ljudskim agentima u tome. Također, to znači da se ljudski agenti nikada neće pomaknuti robotu kako bi mu "raščistili" put prema cilju, što mu u konačnici otežava navigaciju.
Treniranje modela s algoritmom LM-SARL, postupkom treniranja opisanim u 3.2. [8] trajalo je $\approx$ 24 sata, a pristup bez učenja oponašanjem i s funkcijom nagrade iz 4. [18] rezultira nešto kraćim treniranjem, $\approx$ 10 sati, kao i model s pristupom ICM, a s RE3 pristupom za intrinzične nagrade $\approx$ 8 sati. Treniranje SG-D3QN modela trajalo je $\approx$ 80 sati, što dokazuje znatno veću kompleksnost tog modela naspram ostalih. CPU korišten za treniranje je AMD Ryzen 7 5800H s Radeon grafikom.



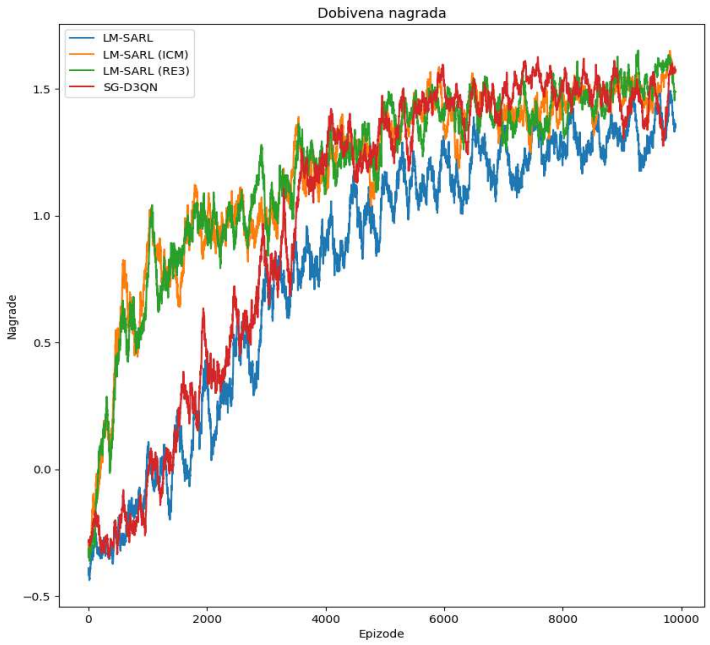
Slika 6.1. Stopa uspješnosti prilikom treniranja različitih modela s algoritmima LM-SARL i SG-D3QN u 10.000 epizoda



Slika 6.2. Stopa kolizije prilikom treniranja različitih modela s algoritmima LM-SARL i SG-D3QN u 10.000 epizoda



Slika 6.3. Vrijeme potrebno robotu da dođe do cilja prilikom treniranja različitih modela s algoritmima LM-SARL i SG-D3QN u 10.000 epizoda



Slika 6.4. Dobivene nagrade prilikom treniranja različitih modela s algoritmima LM-SARL i SG-D3QN u 10.000 epizoda

Slika 6.1. prikazuje stope uspješnosti kod različitih metoda implementacije LM-SARL algoritma. Od osnovne metode brže učenje vidimo kod verzija LM-SARL-a kod kojih robot dobiva intrinzične nagrade, a i imaju bolju konačnu stopu uspješnosti. Algoritam SG-D3QN sporije dostiže visoke stope uspješnosti, no na kraju treniranja pokazuje jednaku uspješnost kao i intrinzični modeli.

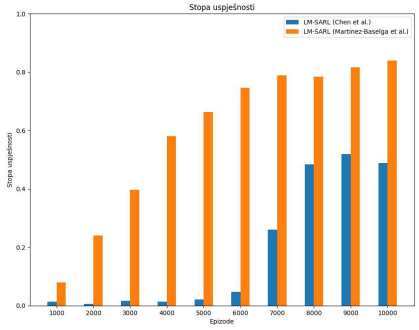
Slika 6.2. prikazuje učestalost kolizije kod proučavanih implementacija. Model s čistim LM-SARL algoritmom, bez intrinzičnih nagrada, od početka treniranja uzrokuje češće kolizije, kao i algoritam SG-D3QN. Ponovno intrinzične metode ranije pokazuju niže učestalosti kolizije, a konačno stanje je približno jednako dobro kao i kod algoritma SG-D3QN.

Vrijeme koje je bilo potrebno robotu da dođe do cilja koristeći različite metode koje su ga vodile prikazano je na slici 6.3. Brzo učenje, ali generalno lošiji rezultat od intrinzičnih metoda postiže čisti LM-SARL. Algoritam SG-D3QN brže uči od prethodno spomenutog algoritma i krajnji je rezultat sličan intrinzičnim modelima.

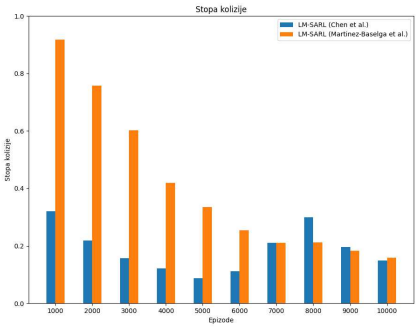
Graf na slici 6.4. prikazuje količinu nagrade koju je robot dobivao u epizodama treniranja. Metode s intrinzičnim nagradama i u ovom slučaju postižu bolji rezultat od čistog LM-SARL-a. SG-D3QN na kraju treniranja pokazuje rezultat jednako dobar kao i modeli s intrinzičnim nagradama.

Modeli s intrinzičnim nagradama za LM-SARL algoritam prikazuju najbrže učenje, ali ne i bolje od modela s algoritmom SG-D3QN. Prikazana je i usporedba modela LM-SARL iz originalne implementacije opisane u 3.2. i novijeg modela opisanog u 6. na slici 6.5. Na dijagramima su prikazane prosječne vrijednosti stopa uspješnosti, stopa kolizije, trajanja navigacije i dobivenih nagrada unutar svakih 1000 epizoda treniranja. Iz prikazanog dijagrama na slici 6.5.(a) vidimo da noviji model kontinuirano ostvaruje veću uspješnost, i na kraju treniranja pokazuje koliko je kvalitetniji. Vidimo da novija implementacija LM-SARL-a brže uči i postiže visoke stope uspješnosti, što primjećujemo i na slici 6.5.(b), gdje vidimo da taj model u početku ima znatno veću učestalost kolizije od originalnog modela. Međutim, razlog ovome je što originalnom modelu često ističe vremenski rok za navigaciju, stoga mu je učestalost kolizije manja. Robotu je korištenjem originalnog modela za kretanje potrebno više vremena do cilja, što vidimo na slici

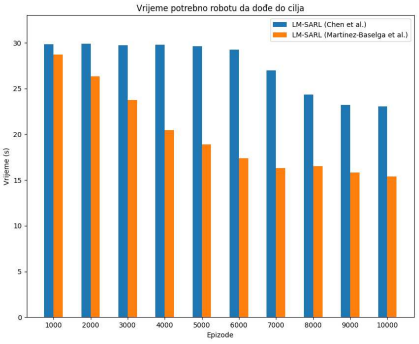
6.5.(c) U početku treniranja, modelima je trajanje navigacije do cilja približno jednako, ali na kraju vidimo da je noviji model znatno brži. Dijagrami na slici 6.5.(d) prikazuju velike razlike u vrijednostima nagrade, a razlog tome leži u drugačijim funkcijama nagrade implementiranima u dvama modelima. Iz ovih slika možemo zaključiti da je prednost novijeg modela LM-SARL-a činjenica da je istreniran koristeći funkciju nagrade prilagođenu korištenim eksperimentima. Također, originalni je model poprimio i neke negativne karakteristike algoritma ORCA zbog učenja oponašanjem, a za neholonomskog robota postavlja drukčija ograničenja za rotaciju u odnosu na ona kojima je model izvorno prilagođen, stoga ne čudi njegova lošija izvedba.



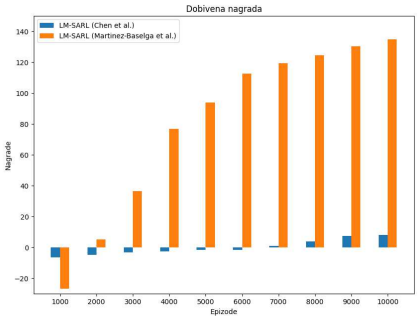
(a) Stope uspješnosti dvaju LM-SARL modela tijekom treniranja



(b) Učestalosti kolizije dvaju LM-SARL modela tijekom treniranja



(c) Trajanje navigacije robota do cilja kod dvaju LM-SARL modela tijekom treniranja



(d) Dobivene nagrade kod dvaju LM-SARL modela tijekom treniranja

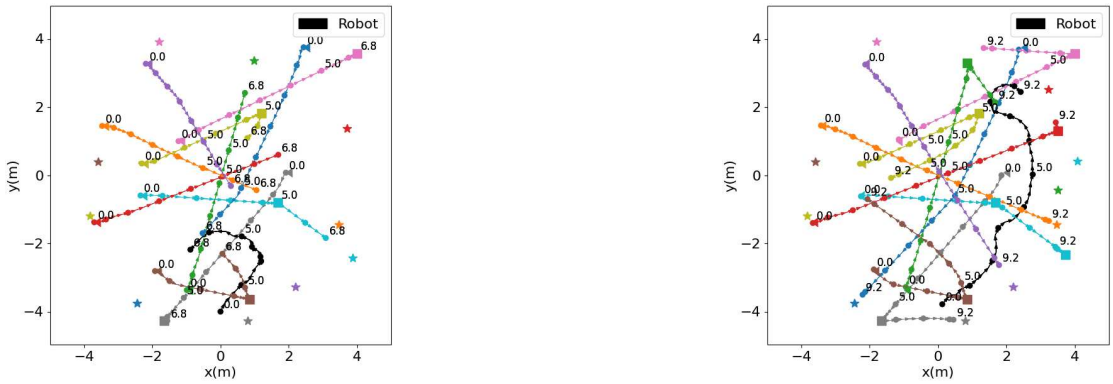
Slika 6.5. Usporedba statistike kod treniranja dvaju LM-SARL modela implementiranih u [8] i u [18]

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.530	0.090	15.130	495.1218	**0.020**
LM-SARL [18]	0.897	0.103	11.727	1483.4659	0.066
LM-SARL (ICM)	0.909	0.091	**10.653**	1487.5209	0.081
LM-SARL (RE3)	0.928	0.071	10.666	1533.7953	0.068
SG-D3QN [18]	**0.964**	**0.036**	11.031	**1650.5357**	0.039

Tablica 6.1. Izvedba svih modela u kružnom scenariju s nevidljivim robotom i 10 ljudskih agenata u gužvi.

Tablica 6.1. prikazuje rezultate testova na simulaciji u kojoj je izvedeno i treniranje modela, dakle u kružnom scenariju. Iz podataka možemo zaključiti da je u takvom scenariju model SG-D3QN bio primjetno najbolji, s uspjehom dolaska robota do cilja u čak 96.4% slučajeva. Možemo primijetiti da je originalni model LM-SARL-a najmanje uspješan, u samo 53% slučajeva. Međutim, vidimo da se kolizija događa u samo 9% slučajeva, na temelju čega dolazimo do zaključka da model prilikom treniranja ne konvergira, odnosno konačni model ne usmjerava robota dovoljno dobro prema cilju i jako se često događa istek dozvoljenog trajanja navigacije.

Na slici 6.6. vidimo prikaz kako se odvija navigacija robota vođenog različitim modelima. Originalna implementacija LM-SARL algoritma [8] ne uspijeva dovesti robota do cilja, kao ni noviji model s istim algoritmom [18]. Vidimo da spomenuti modeli nisu uspjeli navigirati robota na način da izbjegne koliziju s agentima. Međutim, vidimo da LM-SARL model u kojemu se potiče znatiželjnost (intrinzične nagrade s ICM i RE3 pristupom) uspješno vodi robota do zadanog cilja, kao i SG-D3QN model.



(a) LM-SARL [8]

(b) LM-SARL [18]

(c) LM-SARL (ICM)

(d) LM-SARL (RE3)

(e) SG-D3QN

Slika 6.6. Prikaz putanji robota i ljudskih agenata kod navigacije vođene različitim modelima u jednoj od testnih epizoda u kružnom scenariju.

Zbog usporedbe su u nastavku (tablica 6.2.) prikazani rezultati testova modela istreniranih na scenarijima u kojima je robot bio vidljiv ljudskim agentima. U ovakvom

scenariju vidimo očekivano bolje rezultate kod svih modela, uključujući i originalnu implementaciju LM-SARL algoritma koja pokazuje velik skok u uspješnosti u odnosu na kružni scenarij s nevidljivim robotom. Stopa uspješnosti je, unatoč tome, primjetno niža nego kod drugih modela, no rezultati pokazuju veliki napredak za ovaj model. Vidimo utjecaj učenja oponašanjem s algoritmom ORCA, koji pretpostavlja da će svi agenti preuzeti polovicu odgovornosti za izbjegavanje kolizije u svakoj interakciji. Međutim, stopa kolizije je i ovdje niska za spomenuti model, dakle učestalost isteka vremenskog ograničenja od 30 sekundi je i u ovom slučaju nezanemariva (11%). Dakle, ni u ovom slučaju model ne konvergira, a razloga za ovo ima nekoliko. U originalnom se modelu za neholonomskog robota koriste ograničenja za rotaciju kojima model nije prilagođen, stoga možemo zaključiti da se iz tog razloga, uz korišteni dizajn nagrade i postupak treniranja, ne događa konvergencija. Neholonomski se robot, koristeći ovaj model, ne snalazi dovoljno dobro u ovakvim scenarijima, on se previše odmiče od cilja i ne stiže se vratiti do njega. Možemo pretpostaviti da funkcija nagrade očito ne usmjerava robota dovoljno dobro, stoga često zaluta na pogrešan put.

Metoda	**Stopa uspješnosti**	**Učestalost kolizije**	**Vrijeme navigacije**	**Ukupna nagrada**	**Učestalost opasnih situacija**
LM-SARL [8]	0.850	0.040	11.330	796.0399	0.080
LM-SARL [18]	0.947	0.053	11.421	1509.0642	0.077
LM-SARL (ICM)	0.962	0.036	**10.429**	1542.9560	0.089
LM-SARL (RE3)	0.963	0.037	10.775	1521.9126	0.085
SG-D3QN [18]	**0.977**	**0.023**	10.712	**1612.4454**	**0.062**

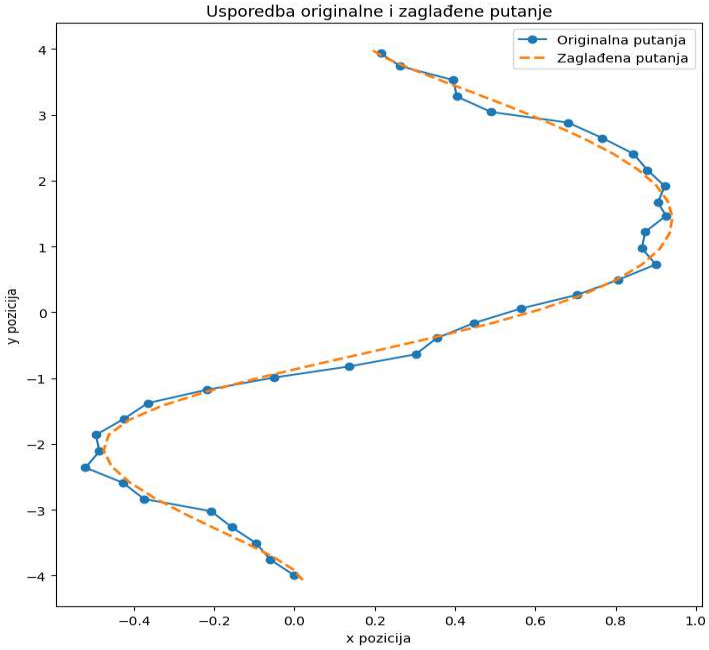
Tablica 6.2. Izvedba svih modela u kružnom scenariju s vidljivim robotom i 10 ljudskih agenata u gužvi.

6.3. Analiza zaglađenosti i duljine putanje

Od velike je važnosti osigurati da putanja kojom se robot kreće kroz prostor bude glatka. Takvu putanju ima robot koji se prirodnije kreće, što je vrlo važno u dinamičnim okruženjima jer se time ostvaruje sigurnost više razine za okolinu. Također, kada bi robot imao nagle pokrete i neprirodna skretanja, to bi u njegovoj okolini ljudima stvorilo nelagodu i

razlog za zabrinutost i zbunjenost. Takvo bi ponašanje robota moglo i ljude učiniti nesigurnima, stoga bi i njihovo gibanje postalo manje prirodno i naglo. Manje glatko gibanje uzrokuje i češće kolizije i susrete na maloj udaljenosti, što se pokušava izbjeći.

Za ispitivanje zaglađenosti putanje, u ovom se radu računa B-spline, odnosno kombinaciju krivulja koje prolaze kroz određene točke. U ovome se radu pronalazi aproksimacija B-spline krivulje u odnosu na zadanu krivulju, a to je ovdje krivulja koju čini putanja robota do njegovog cilja. Putanju robota čine nove pozicije robota nakon svake izvedene radnje. Za računanje točaka B-spline krivulje koristi se biblioteka *scipy*, odnosno njezin paket *scipy.interpolate* koji nudi razne prikladne funkcije, među kojima je i *splprep*. Spomenuta funkcija služi nam za dobivanje reprezentacije opisane B-spline krivulje. Nakon što se izračunala zaglađena krivulja u odnosu na originalnu, računaju se udaljenosti između točaka koje pripadaju krivulji putanje robota i pripadnih točaka B-spline krivulje. Na temelju zbroja tih udaljenosti, određuje se zaglađenost putanje, odnosno veća udaljenost jednaka je manjoj zaglađenosti. U tablici 6.3. vidimo prosječne razine zaglađenosti putanja tijekom 1000 epizoda testiranja. Stopa zaglađivanja postavljena je na vrijednost 0.1, a stupanj krivulje postavljen je na vrijednost 3. Na slici 6.7., plavom bojom prikazana je putanja kojom je robot navigirao kroz okolinu u jednoj od epizoda testiranja, a narančastom bojom prikazana je zaglađena verzija (B-spline) iste putanje. Kao što je prethodno spomenuto, zaglađenost je to veća što je vrijednost prikazana u tablici bliža nuli. Možemo primijetiti da su zaglađenosti putanja takve da ne možemo uočiti model koji se u pozitivnom smislu ističe. Najveću prosječnu zaglađenost putanja ima LM-SARL s ICM pristupom za intrinzične nagrade, a najmanju izvorna implementacija LM-SARL algoritma. SG-D3QN algoritam ne rezultira pretjerano zaglađenim putanjama jer, iako je najuspješniji i najbolje se snalazi u izbjegavanju dinamičkih prepreka, robot koristeći takav model češće ima nešto naglije, oštrije pokrete kako bi ih izbjegao.



Slika 6.7. Usporedba zaglađene putanje i putanje kojom je robot navigirao do cilja

Kako bismo na još jedan način usporedili modele, a i ispitali njihovu izvedbu, uspoređuje se i duljina putanje robota od početka do cilja kod svih ovdje istreniranih modela. Duljina se računa tako da su se dijelovi putanje bilježili svakim potezom robota, na način da se računala kvadratna razlika između njegove nove i prethodne pozicije. U nastavku u tablici 6.3. su prikazani podaci o prosječnoj zaglađenosti i duljini putanje u 1000 epizoda testiranja i duljina putanje mjeri se u metrima. U tablici 6.3. vidimo da LM-SARL sa intrinzičnim nagradama (pristup ICM) rezultira s najkraćim putanjama, a originalna implementacija LM-SARL-a s najduljim. Iz dobivenih rezultata možemo zaključiti da putanje s najboljim karakteristikama ima model sa ICM pristupom za intrinzične nagrade, odnosno najzaglađenije su, a ujedno i najkraće.

Metoda	Zaglađenost putanje	Duljina putanje (m)
LM-SARL [8]	2.0397	11.0455
LM-SARL [18]	1.8114	9.2864
LM-SARL (ICM)	**1.7592**	**9.1098**
LM-SARL (RE3)	1.7657	9.2017
SG-D3QN [18]	1.8059	9.7504

Tablica 6.3. Prosječna zaglađenost i duljina putanje kod treniranih modela u kvadratnom scenariju.

6.4. Usporedba sposobnosti modela za generalizaciju

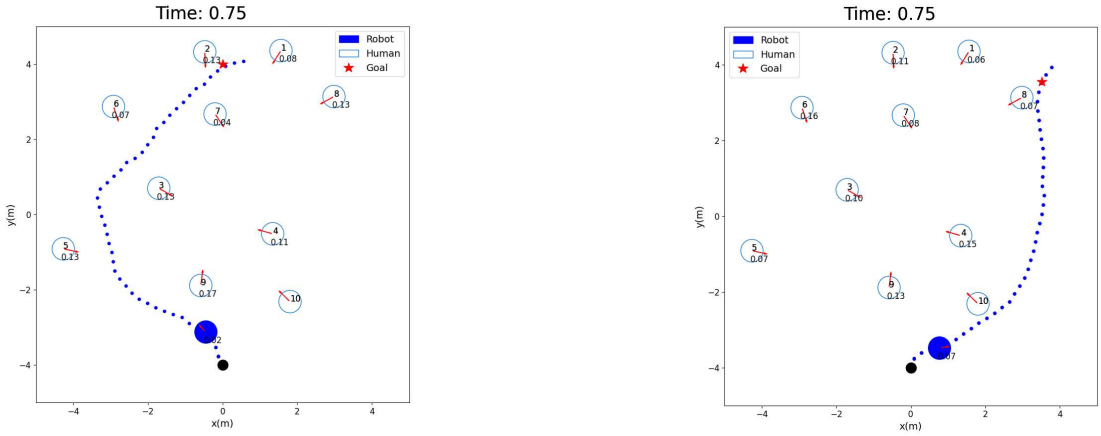
Kako bi uspješno donijeli zaključak o tome može li se model primijeniti u stvarnom životu, odnosno na pravome robotu, potrebno je ispitati izvedbu robota koji ga koristi za navigaciju u različitim situacijama koje su bliske onima iz stvarnog života. U sljedećim su potpoglavljima prikazani rezultati testiranja prethodno opisanih modela na različitim scenarijima koji pokušavaju u što većoj mjeri simulirati okruženje u stvarnom životu. Prikazane su tablice sa postocima uspješnosti, vremenom navigacije do cilja, ukupnom nagradom, ali i učestalošću kolizije i situacija u kojima se robot našao na opasno maloj udaljenosti od ljudskog agenta. Recimo, tablica 6.4. prikazuje uspješnost svih modela u kvadratnom scenariju. U toj tablici vidimo da je i dalje SG-D3QN najbolji model, unatoč padu u stopi uspješnosti, metoda LM-SARL sa ICM pristupom za intrinzične nagrade bilježi visoki skok u uspješnosti, a originalni model s LM-SARL algoritmom ponovno pokazuje primjetno lošije rezultate od novijeg modela s istim algoritmom. U tablicama su naglašene podebljanim brojkama najbolje vrijednosti, uspoređujući izvedbu svih modela u istom scenariju.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.700	0.100	12.720	653.1510	**0.040**
LM-SARL [18]	0.885	0.115	10.856	1457.1531	0.066
LM-SARL (ICM)	0.937	0.063	**9.974**	1544.9926	0.076
LM-SARL (RE3)	0.933	0.066	10.072	1554.5090	0.065
SG-D3QN [18]	**0.957**	**0.043**	10.391	**1628.2031**	0.043

Tablica 6.4. Izvedba svih modela u kvadratnom scenariju s 10 ljudskih agenata u gužvi.

6.4.1. Slučajno generiranje početnih i ciljnih pozicija robota

Robot neće uvijek biti moći postavljen na jednaku početnu poziciju te imati zadatak doći do iste ciljne pozicije koja mu je zadana u simulacijama prilikom treniranja, stoga ovaj rad testira uspješnost modela kada su robotu te pozicije zadane na slučajan način.



(a) Klasično postavljena početna pozicija i cilj

(b) Promijenjen cilj robotu

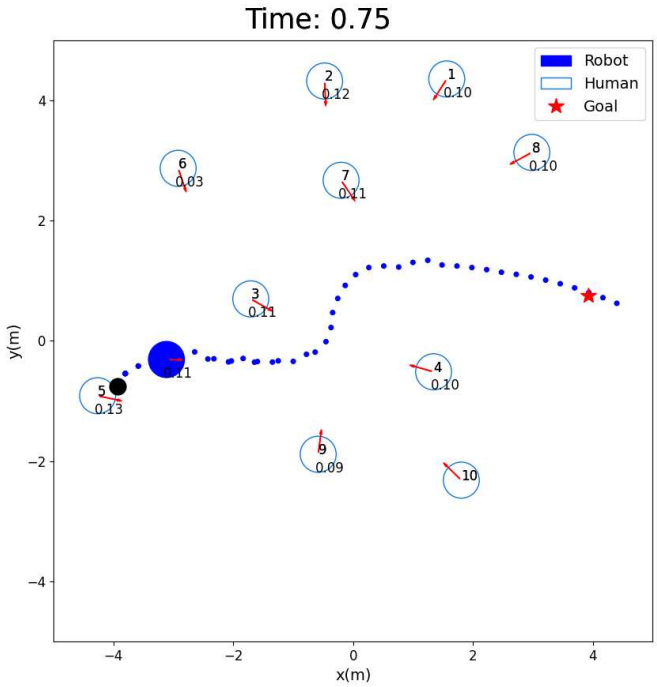
Slika 6.8. Usporedba klasičnog i slučajnog postavljanja cilja za robota u simulaciji

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.780	0.160	9.140	714.1823	0.080
LM-SARL [18]	0.889	0.111	8.741	1177.0720	0.074
LM-SARL (ICM)	0.933	0.065	**8.111**	1259.2761	0.076
LM-SARL (RE3)	0.940	0.060	8.253	1304.3096	0.061
SG-D3QN [18]	**0.956**	**0.044**	8.292	**1321.2754**	**0.051**

Tablica 6.5. Izvedba svih modela u kvadratnom scenariju sa slučajno određenim ciljnim pozicijama robota.

Prema rezultatima testiranja iz tablice 6.5., vidimo da originalni model s LM-SARL algoritmom postiže najlošije rezultate u simulaciji s promjenom lokacije cilja prikazanoj na slici 6.8.(b), ali primjetno bolje nego u simulaciji bez promjene cilja. Robotu kod ovog modela pomaže činjenica da su nekada ciljevi postavljeni bliže u odnosu na cilj iz simulacije tijekom treniranja. Vidimo da je i u ovoj simulaciji algoritam SG-D3QN uspješniji od ostalih, bez važnije promjene u uspješnosti. Možemo uočiti da modeli ne pokazuju značajno lošije rezultate dodavanjem ove promjene u simulaciju.

Na slici 6.9. prikazan je opisani scenarij u kojemu su i početna i ciljna pozicija robotu izmijenjene u odnosu na one koje je imao u svakoj epizodi treniranja.



Slika 6.9. Simulacija sa promijenjenom početnom i ciljnom pozicijom robota.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.500	0.260	15.950	412.1295	**0.040**
LM-SARL [18]	0.702	0.294	12.326	1066.8938	0.076
LM-SARL (ICM)	0.814	0.186	**10.421**	1244.0759	0.105
LM-SARL (RE3)	0.899	0.100	11.136	1480.2280	0.063
SG-D3QN [18]	**0.954**	**0.046**	10.769	**1618.9983**	0.043

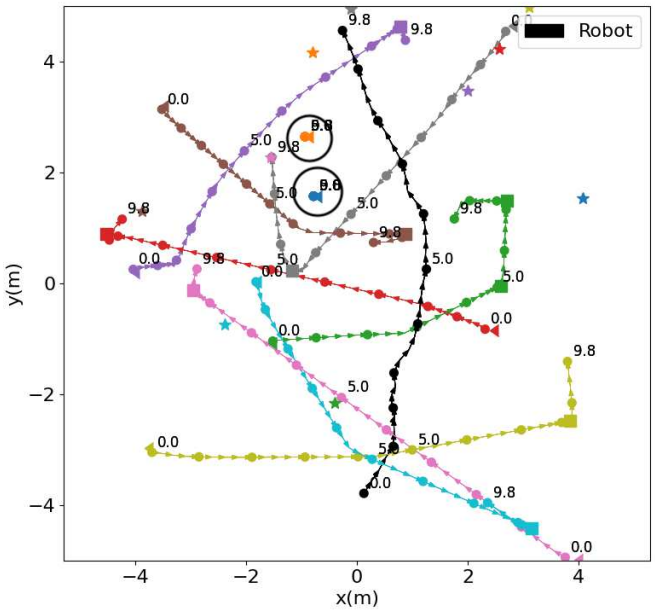
Tablica 6.6. Izvedba modela u kvadratnom scenariju sa slučajno određenim početnim i ciljnim pozicijama robota.

U tablici 6.6. vidimo da svi modeli s LM-SARL algoritmom pokazuju lošije rezultate u odnosu na simulaciju iz treniranja, što bi značilo da, kada bismo postavili robota na neko drugo mjesto u prostoru i zadali mu drukčiji cilj, primjetno bi se loše snalazio i uzrokovao češće kolizije. Ovi rezultati pokazuju da je izvedba LM-SARL algoritma pretjerano ovisna o simulaciji u kojoj je algoritam treniran, odnosno ovaj algoritam dobro radi samo za putanju između određene početne pozicije i cilja. Navedeno bi značilo da, ako bismo ga htjeli primijeniti na pravome robotu i u stvarnoj situaciji, ovakav algoritam ne bismo

mogli koristiti. SG-D3QN pokazuje dobre rezultate, dakle njegova izvedba ne ovisi o početnoj i ciljnoj poziciji robota u prostoru.

6.4.2. Uvođenje statičkih prepreka

Svi pristupi koji se istražuju u ovom radu imaju istu pogrešnu pretpostavku da će se ljudi u gužvi neprestano gibati, i to u praznom prostoru. Međutim, ako bismo htjeli simulirati nešto prirodniju situaciju, prikazali bismo ljude koji ponekad zastanu i uveli statičke prepreke, stoga i u takvim simulacijama testiramo analizirane modele. Statičke prepreke predstavljaju ljudski agenti koji stoje na početnoj poziciji koja im je zadana slučajno i ne kreću se.

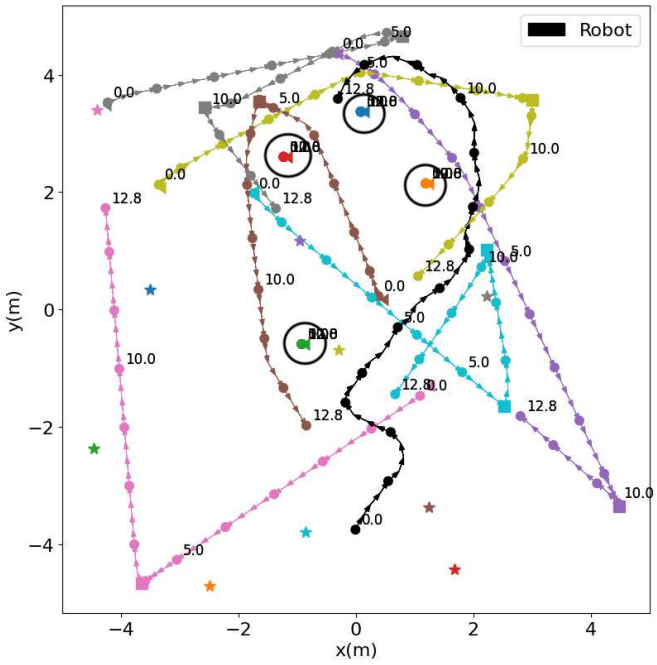


Slika 6.10. Putanje robota i ljudskih agenata uz slučajno odabrana 2 statička agenta. Kružnicama su označeni statički ljudski agenti.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.620	0.070	12.650	591.3918	**0.030**
LM-SARL [18]	0.909	0.090	10.754	1482.2472	0.072
LM-SARL (ICM)	0.921	0.079	**10.142**	1491.8337	0.079
LM-SARL (RE3)	0.920	0.079	10.183	1521.1004	0.067
SG-D3QN [18]	**0.963**	**0.037**	10.763	**1656.7837**	0.034

Tablica 6.7. Izvedba modela u kvadratnom scenariju sa dva slučajno postavljena statička ljudska agenta

Tablica 6.7. pokazuje statistiku izvedbi modela u simulaciji u kojoj su, od 10 ljudskih agenata, dva statička. U ovom slučaju, algoritmi sa intrinzičnim nagradama imaju lošije rezultate u odnosu na originalnu simulaciju. Ni u ovoj simulaciji nema promjene što se tiče pitanja koji je model najuspješniji.



Slika 6.11. Putanje robota i ljudskih agenata uz slučajno generiranim brojem statičkih agenata. Kružnicama su označeni statički ljudski agenti.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.590	0.060	12.870	562.0211	**0.030**
LM-SARL [18]	0.918	0.078	11.755	1538.7115	0.051
LM-SARL (ICM)	0.946	0.053	**10.552**	1572.9187	0.062
LM-SARL (RE3)	0.941	0.057	10.749	1580.8840	0.059
SG-D3QN [18]	**0.951**	**0.049**	10.940	**1631.3751**	0.038

Tablica 6.8. Izvedba modela u kvadratnom scenariju sa slučajnim brojem statičkih ljudskih agenata

Iz navedenih podataka u tablicama 6.7. i 6.8. možemo zaključiti da, iako je i u ovakvim scenarijima sa statičkim agentima algoritam SG-D3QN najuspješniji i robot vođen njime dobiva najveću ukupnu nagradu, uspješnost mu se u scenariju u kojemu je broj dinamičkih agenata slučajno određen nije povećala kao i kod svih ostalih modela, osim originalne implementacije LM-SARL algoritma. Iz navedenoga možemo zaključiti da se algoritam pretjerano oslanja na pretpostavku konstantne dinamičnosti u gužvi. Q-mreža vjerojatno dobro ne uči pronaći značajke za statičke prepreke jer ih tretira kao dinamičke. Problem može biti i u korištenom GAT-u, koji ne prioritizira dinamičke prepreke nad statičkim, iz razloga što ih ne zna razlikovati. Originalna implementacija LM-SARL-a pokazuje najrjeđu prisutnost u opasnim situacijama, međutim, razlog ovome je to što se robot pretjerano udaljava od cilja, ali i od ljudskih agenata. Dakle, ovaj model još jednom dokazuje da se konvergencija u treniranju nije dogodila. Kod svih ostalih modela možemo uočiti da imaju općenito bolje rezultate kada su u simulaciji statički ljudski agenti nego kad nisu, a posebno kada ih je više statičkih prisutno.

Za očekivati je da se modeli u prisustvu statičkih agenata bolje snalaze zbog generalne pretpostavke da je dinamičke prepreke teže zaobići uslijed nepredvidivosti njihovog gibanja. Iz tog razloga možemo zaključiti da je algoritam SG-D3QN pretjerano prilagođen situaciji u kojoj se očekuje da su svi ljudski agenti u pokretu. Morali bismo uvesti promjene u model u kojima bismo učili mrežu da razlikuje dinamičke od statičkih agenata.

6.4.3. Zaustavljanje ljudskih agenata tijekom gibanja do cilja

Imajući na umu prirodu ljudske interakcije, moramo uzeti u obzir i postojanje mogućnosti da će ljudi u gužvi nekada i zastati kako bi razmislili gdje točno moraju ići, razgledali prostor oko sebe ili kako bi popričali s poznanikom ili možda grupom poznanika. Iz tog razloga u ovom se radu provode i testovi na simulacijama u kojima se ljudski agenti ne gibaju neprestano, već u (slučajno odabranim) trenucima zastanu, prije nego nastave s gibanjem prema cilju.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.690	0.110	12.840	638.5153	**0.040**
LM-SARL [18]	0.883	0.117	10.892	1437.4295	0.068
LM-SARL (ICM)	0.885	0.115	**9.808**	1447.4805	0.080
LM-SARL (RE3)	**0.908**	**0.092**	9.976	1514.4363	0.067
SG-D3QN [18]	**0.908**	**0.092**	10.339	**1528.5533**	0.050

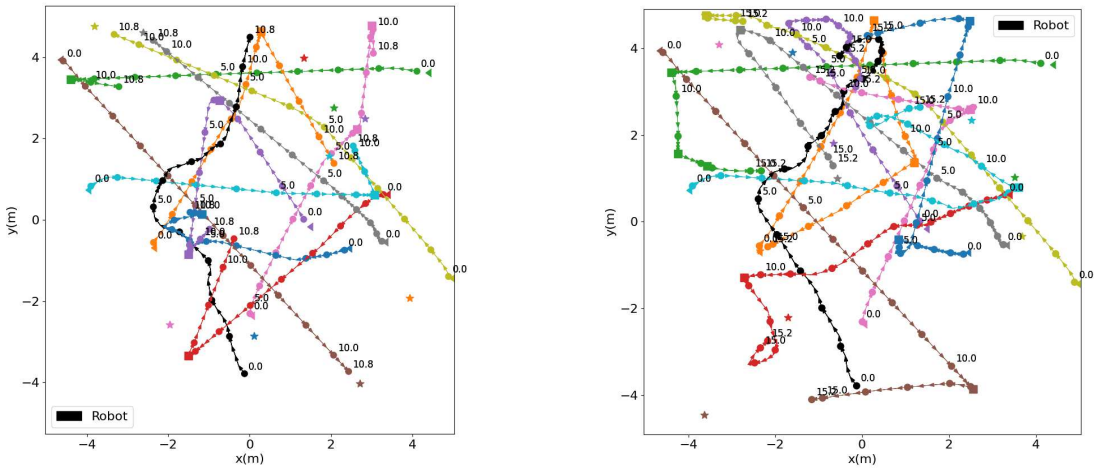
Tablica 6.9. Izvedba svih modela u kvadratnom scenariju s 10 ljudskih agenata u gužvi u kojoj ljudi prilikom gibanja zastajkuju sa vjerojatnošću od 10%.

U ovako postavljenoj simulaciji, s vjerojatnošću zaustavljanja ljudskih agenata od 10%, iz rezultata u tablici 6.9. možemo zaključiti da je smanjena kvaliteta izvedbe svih ispitanih modela, što ponovno duguju prenaučenosti postavkama simulacije na kojoj su trenirani. Niti jedan od ispitanih modela nije neovisan o karakteristikama ponašanja ljudskih agenata koje zaobilazi, stoga bismo morali trenirati modele na skupinama ljudskih agenata koji se ponašaju na različite načine. Mreže koje se koriste za treniranje, prilikom učenja izvlače karakteristike koje uoče u ponašanju ljudskih agenata u simulaciji. Iz tog razloga ne iznenađuje činjenica da ovako trenirani modeli podbacuju kada se u simulaciji ljudski agenti ponašaju drukčije od onoga na što su naučeni.

6.4.4. Dinamička promjena ciljeva ljudskih agenata

U stvarnosti, ljudi ponekad prilikom gibanja odluče da ipak žele otići negdje drugdje, pa promijene cilj do kojega žele ići. Simulacije u kojima su se trenirali ovdje ispitivani mo-

deli, pogrešno pretpostavljaju da se takva situacija ljudskim agentima ne može dogoditi pa ne možemo reći da je robot istreniran u dovoljno realističnim uvjetima. U ovom radu iz navedenog se razloga testiraju izvedbe modela u situacijama kada se (nekim) ljudskim agentima slučajno mijenja ciljna pozicija prilikom gibanja.



(a) Obični scenarij

(b) Dinamička promjena ciljeva ljudskih agenata

Slika 6.12. Prikaz jednakog testnog scenarija u dva slučaja 6.12.(a) prikazuje odvoženi uobičajeni kvadratni scenarij, a 6.12.(b) prikazuje istu simulaciju, ali u ovom se scenariju slučajno mijenjaju ciljevi ljudskim agentima tijekom gibanja do cilja

Na slici 6.12. možemo vidjeti razliku u scenariju u kojemu se ljudski agenti kreću normalno prema zadanom cilju i u scenariju u kojemu ljudski agenti usred gibanja promijene svoj cilj. Vidimo da je putanja kojom se robot kreće prema cilju u tom dinamičnom scenariju drukčija, manje je zaglađena. Uzrok tome je jasan, robot reagira na iznenadnu promjenu cilja ljudskih agenata na način da naglo mijenja svoju putanju kako bi izbjegao koliziju. Tablica 6.10. prikazuje uspješnost svih modela u ovakvom scenariju.

Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.640	0.100	12.930	595.0532	**0.030**
LM-SARL [18]	0.881	0.119	10.872	1444.1950	0.065
LM-SARL (ICM)	0.921	0.079	**9.957**	1516.3134	0.077
LM-SARL (RE3)	0.941	0.059	10.091	1577.3505	0.063
SG-D3QN [18]	**0.947**	**0.053**	10.468	**1609.1727**	0.044

Tablica 6.10. Izvedba modela u kvadratnom scenariju sa dinamičnim promjenama ciljeva ljudskih agenata.

Unatoč tome što su robotu u ovom slučaju putanje manje zaglađene, prema rezultatima iz tablice 6.10., vidimo da su modeli, osim originalnog LM-SARL-a, dobro prilagođeni promjenama u zadanim ciljevima ljudskih agenata, brzo se i lako privikavaju na nastalu promjenu te ona na njihovu navigaciju ne utječe u prevelikoj mjeri. Razlog ovome leži u činjenici da su ljudski agenti i u originalnoj simulaciji programirani na način da im se, kada dođu do svog prvotno određenog cilja, zadaje novi cilj. Robot je, prema tome, naučen nositi se sa promjenama ciljeva ljudskih agenata. Neki su modeli uspješniji uz ove promjene, a neki manje uspješni, no ta razlika nije značajna da bismo na temelju ovog testiranja uočili nedostatke ili prednosti nekog od modela.

U originalnom modelu s LM-SARL algoritmom, kao što smo već vidjeli, neuronske mreže tijekom treniranja nisu bile uspješne u ekstrakciji značajki, stoga vidimo da je i u ovom slučaju taj model manje uspješan.

6.4.5. Dodavanje šuma senzoru

Senzori koje će robot imati u stvarnosti i njime se služiti kako bi registrirao prepreke u svom okruženju, neće uvijek savršeno točno moći robotu prenijeti informacije o pozicijama navedenih prepreka, što simulacije u kojima je robot treniran pretpostavljaju. Modele se testira na simulacijama u kojima informacije dobivene od robotovog "senzora" sadržavaju šum. Pozicijama ljudskih agenata za koje robot zna, odnosno primijetio ih je, dodaje se slučajno generirana mala vrijednost koja predstavlja šum u informaciji.

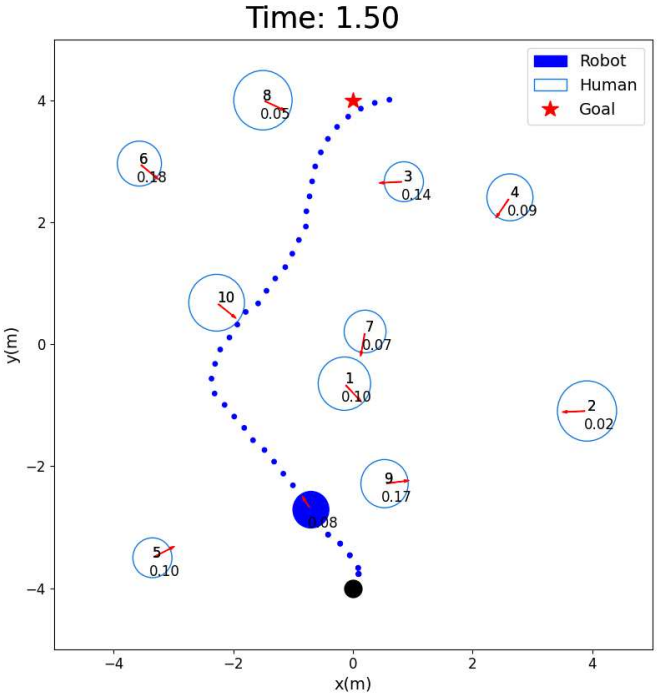
Metoda	Stopa uspješnosti	Učestalost kolizije	Vrijeme navigacije	Ukupna nagrada	Učestalost opasnih situacija
LM-SARL [8]	0.690	0.100	12.530	639.2867	**0.040**
LM-SARL [18]	0.871	0.129	10.862	1431.2230	0.068
LM-SARL (ICM)	0.931	0.069	**10.023**	1535.3955	0.076
LM-SARL (RE3)	0.931	0.069	10.151	1557.4338	0.066
SG-D3QN [18]	**0.951**	**0.049**	10.407	**1625.7144**	0.042

Tablica 6.11. Izvedba modela u kvadratnom scenariju u kojemu se robotovom senzoru dodaje šum

Iz tablice 6.11. vidimo da modelima ovakva izmjena u scenariju ne utječe na izvedbu. Ovdje je razina šuma postavljena na vrijednost 0.01, odnosno u tom iznosu odstupaju prave lokacije ljudskih agenata od onih koje robot vidi. Možemo zaključiti da je taj iznos šuma premalen i da su modeli dobro prilagođeni tolikim pogreškama senzora. Iako je ovakva situacija pozitivna, morali bismo ispitati koliko točno odstupanje, odnosno šum modelima počne utjecati na izvedbu.

6.4.6. Slučajno generiranje atributa ljudskih agenata

U stvarnom životu ljudi u robotovom okruženju neće imati jednake fizičke karakteristike niti će se svi kretati jednakom brzinom, zbog čega je važno ispitati izvedbu modela u scenariju u kojemu su ljudskim agentima vrijednosti ovih atributa slučajno postavljene. Brzine agenata uniformnom se distribucijom generiraju iz raspona vrijednosti $(0.5, 1.5)$, a njihovi radijusi iz raspona $(0.3, 0.5)$



Slika 6.13. Prikaz simulacije sa slučajno postavljenim radijusima ljudskih agenata.

Metoda	**Stopa uspješnosti**	**Učestalost kolizije**	**Vrijeme navigacije**	**Ukupna nagrada**	**Učestalost opasnih situacija**
LM-SARL [8]	0.560	0.170	12.930	485.6166	**0.050**
LM-SARL [18]	0.827	0.171	11.427	1251.9866	0.093
LM-SARL (ICM)	0.858	0.141	**10.353**	1285.1186	0.123
LM-SARL (RE3)	**0.888**	**0.112**	10.577	**1370.9872**	0.103
SG-D3QN [18]	0.838	0.162	10.712	1246.0060	0.106

Tablica 6.12. Izvedba modela u kvadratnom scenariju u kojemu se atributi ljudskih agenata (preferirana brzina i radijus) postavljaju slučajnim generiranjem vrijednosti

Iz tablice 6.12. možemo uočiti kako modeli pokazuju manju kvalitetu izvedbi kada nije okružen skupom identičnih ljudskih agenata te mu zbog toga njihove karakteristike više nisu predvidive, što znatno otežava navigaciju. Po prvi put algoritam SG-D3QN primjetno podbacuje, u odnosu na rezultate u prethodnim eksperimentima. Možemo zaključiti da se niti jedan model se snalazi dobro u ovakvoj simulaciji i pokazuju pretjeranu prilagođenost scenarijima na kojima su se trenirali. Ovdje se vidi važnost izrade modela sa snažnom mogučnošću generalizacije, u svrhu izbjegavanja ovoliko lošijih rezultata u

drukčijim, kompleksnijim simulacijama.

Iako većina ispitanih modela u ovom radu prikazuje konvergenciju u treniranju, a kasnije i većinom dobre reultate testiranja, oni spomenute uspjehe duguju funkciji nagrade koja je podešena upravo za simulacije na kojima se modeli treniraju. Iz tog razloga ne iznenađuje činjenica da, ako modele treniramo koristeći iste funkcije nagrade, ali u simulaciji sa slučajno generiranim preferiranim brzinama i radijusom ljudskih agenata, testovi u ovakvom scenariju i tada pokazuju primjetno lošije rezultate, što prikazuje tablica 6.13.

Metoda	**Stopa uspješnosti**	**Učestalost kolizije**	**Vrijeme navigacije**	**Ukupna nagrada**	**Učestalost opasnih situacija**
LM-SARL [8]	0.550	0.080	13.400	510.5826	**0.030**
LM-SARL [18]	0.858	0.142	12.045	1352.8143	0.071
LM-SARL (ICM)	0.894	0.106	**11.021**	1411.7221	0.080
LM-SARL (RE3)	0.887	0.113	11.154	1438.3092	0.073
SG-D3QN [18]	**0.923**	**0.077**	11.541	**1532.8970**	0.056

Tablica 6.13. Izvedba modela u kvadratnom scenariju u kojemu se atributi ljudskih agenata (preferirana brzina i radijus) postavljaju slučajnim generiranjem vrijednosti, nakon treniranja u takvim scenarijima

6.5. Analiza sigurnosti gibanja

Kako bismo model koji se koristi za navigaciju kroz gužve mogli primijeniti na stvarnom robotu, morali bismo biti uvjereni u činjenicu da je njegova navigacija, vođena takvim modelom, na dovoljno visokoj razini sigurna za okolinu u kojoj se nalazi. Ako se robot prečesto približava ljudskim agentima na udaljenosti koja se smatra neugodnom ili s njima sudara, tada takav model koji kontrolira robota ne možemo smatrati iskoristivim. Potrebno je provesti analizu sigurnosti u navigaciji robota koje ovi modeli kontroliraju i ako je moguće, pronaći način da se ta navigacija učini još sigurnijom. Ovako možemo razviti modele koji će robotima omogućiti siguran suživot s ljudima u kućanstvima, uredima, ali i raznim javnim mjestima. Kako bi se provela analiza sigurnosti u navigaciji robota koristeći analizirane modele, uzima se prosječna učestalost situacija u kojima se

robot nalazi na opasno maloj udaljenosti od ljudskog agenta, uzevši u obzir sve ispitne scenarije iz potpoglavlja 6.2. i 6.4. U tablici 6.14. su prikazani rezultati analize.

Metoda	**Učestalost opasnih situacija**	**Minimalna udaljenost od ljudskog agenta**
LM-SARL [8]	**0.0423**	0.1285
LM-SARL [18]	0.0695	0.1415
LM-SARL (ICM)	0.0837	0.1475
LM-SARL (RE3)	0.0699	**0.1483**
SG-D3QN [18]	0.0509	**0.1483**

Tablica 6.14. Analiza sigurnosti modela prikazom podataka o učestalosti opasnih situacija tijekom navigacije te prosječnoj minimalnoj udaljenosti od ljudskog agenta u tim situacijama

Vidimo da originalni LM-SARL ostvaruje najnižu učestalost opasnih situacija, međutim, znamo i da je razlog tome činjenica da se robot pretjerano udaljava od ljudi, ali i od cilja jer model ne konvergira, što objašnjava rezultate u tablici. Iz navedenog razloga ne možemo smatrati taj model najsigurnijim. Posljedica navedenih činjenica i rezultata koje tablica prikazuje je zaključak da je algoritam SG-D3QN najuspješniji i u kontekstu sigurnosti u navigaciji, što možemo zaključiti na temelju učestalosti prisustva robota u opasnim situacijama od samo 5.09%. Opasnim situacijama definiramo one u kojima se robot približava ljudskom agentu na udaljenost manju od postavljene najmanje sigurne udaljenosti, a to je u ovom slučaju 0.2 m. S druge strane, LM-SARL algoritam sa ICM postupkom za dodjelu intrinzičnih nagrada najčešće dovodi robota u opasne situacije, u 8.37% slučajeva. U slučaju kada se stvori prethodno opisana nelagodna situacija, zabilježavamo i najmanju udaljenost na kojoj se robot u nekom trenutku našao od ljudskog agenta. Model s originalnom implementacijom algoritma LM-SARL, prema podacima se u takvim situacijama približava ljudskim agentima na prosječno najmanjoj udaljenosti, dok modeli s algoritmima SG-D3QN i LM-SARL uz intrinzične nagrade s RE3 postupkom postižu za robota najveću udaljenost u spomenutim okolnostima. Navedeno dokazuje da je SG-D3QN algoritam za navigaciju robota kroz gužve najsigurniji. Očigledno SG-D3QN algoritam implementira najbolju reprezentaciju gužve, što vidimo u njegovoj sposobnosti predviđanja nadolazeće interakcije, stoga se robot uspijeva držati na najsigurnijoj udaljenosti od ljudi, u usporedbi s ostalim modelima.

7. Zaključak

Cilj ovog rada bio je demonstrirati na simulacijama izvedbe nekih od najpoznatijih postojećih rješenja za problem navigacije robota kroz gužve. Originalna implementacija LM-SARL algoritma uspoređena je s modelom koji ima isti algoritam, ali za čije se treniranje nije koristilo učenje oponašanjem i s drukčijim dizajnom funkcije nagrade te je pokazano kako takav, noviji model, postiže bolje rezultate. Originalna implementacija LM-SARL-a iz [8] pokazuje određene važne nedostatke. Njezina funkcija nagrade izrađena je originalno za rješavanje problema izbjegavanja kolizije između dvaju međusobno nekomunicirajućih agenata. Također, učenje oponašanjem algoritma ORCA uzrokuje da mreža konvergira prema suboptimalnoj politici navigacije, i to u ranoj fazi treniranja. Osim toga, originalni je model s LM-SARL algoritmom bio, prema izvornoj ideji, prilagođen karakteristikama neholonomskog robota s drukčijim restrikcijama u kretanju u odnosu na one koje su korištene u ovom radu, stoga i tome pripisujemo lošije rezultate originalnog modela kada ga koristimo za navigiranje neholonomskog robota, točnije, monocikla.

Navedeni se modeli uspoređuju i s onim koji također implementira algoritam LM-SARL, no robota potiče na istraživanje nepoznatih mu područja u okolini, tako da mu za to dodjeljuje tzv. intrinzične nagrade. Oba ispitana postupka za dodjelu intrinzičnih nagrada, ICM i RE3, dokazuju da njihova primjena predstavlja napredak za modele dubokog učenja. Kada usporedimo LM-SARL bez učenja oponašanjem, iz [18], istreniran bez intrinzičnih nagrada i s njima, vidimo bolje rezultate kada ih se dodaje u funkciju nagrade. LM-SARL algoritam je primjetno uspješniji u modelima u kojima u većoj mjeri riskira istražujući prostor oko sebe. Također, kada robot koristi LM-SARL s ICM postupkom za nagrade, najbrže dolazi do cilja, odnosno, s obzirom na to da je brzina robota postavljena kod svih na jednaku maksimalnu vrijednost, njegova je putanja često najkraća. Osim toga, iako neznatno u odnosu na ostale modele, ta je putanja također i najzaglađe-

nija. Međutim, takav model LM-SARL algoritma robota pri navigaciji najčešće dovodi u opasne situacije, odnosno dovodi ga na nelagodnu udaljenost do ljudskih agenata. Sa svim navedenim modelima još je uspoređen i onaj s algoritmom SG-D3QN, koji također koristi funkciju nagrade kao i unaprijeđeni model s LM-SARL-om. Međutim, ovaj model koristi dvojnu duboku Q-mrežu i predviđa zadani broj budućih događaja u simulaciji, stoga je računski najzahtjevniji, iako je najuspješniji. SG-D3QN pokazao je sveukupno najbolju izvedbu prilikom testiranja sposobnosti za generalizaciju, u simulacijama s raznim karakteristikama koje ih razlikuju od one u kojoj su svi modeli trenirani. Prema rezultatima analize, taj model pruža robotu i najveću sigurnost u navigaciji, odnosno najrjeđe ga dovodi u pretjerano nelagodnu blizinu ljudskim agentima.

Unatoč brojnim pozitivnim rezultatima i čestim visokim stopama uspješnosti u testovima kojima se ispituje generalizacija analiziranih modela, zabrinjava činjenica da se svi modeli lošije snalaze u postavkama simulacije u kojima se karakteristike ili ponašanje ljudskih agenata razlikuju od onoga kako su bili postavljeni u epizodama treniranja. Svoje nedostatke modeli posebno pokazuju kada se ljudskim agentima u svakoj epizodi slučajno odrede preferirana brzina i radijus (6.4.6.). U ovom scenariju svi modeli imaju primjetno lošiju izvedbu, uključujući i onaj s algoritmom SG-D3QN, koji je u svim scenarijima najuspješniji. Činjenicu da se radi o problemu prenaučenosti, dokazuju i lošiji rezultati testiranja modela koji su trenirani na simulacijama sa slučajno generiranim atributima agenata. Simulacija u kojoj ljudski agenti s malom vjerojatnošću zastajkuju prilikom gibanja do cilja (6.4.3.), također je robotima dovoljno zakomplicirala navigaciju da kod svih modela i ovdje vidimo primjetno lošije snalaženje. Navedene činjenice pokazuju visoku razinu pristranosti u uzorkovanju karakteristika ljudskog ponašanja, zbog čega vidimo mjesto za napredak kod svih modela.

Kada bismo razmatrali mogućnost primjene nekoga od analiziranih modela na stvarnome robotu, morali bismo uzeti u obzir koliko se model dobro snalazi na eksperimentima koji sadrže realistične karakteristike. Na temelju dobivenih rezultata i prethodno donesenih zaključaka, u obzir bismo najprije mogli uzeti model s algoritmom SG-D3QN. Međutim, zbog loših rezultata prilikom navigacije robota u simulaciji s izmjenama karakteristika ljudskih agenata, morali bismo model trenirati u kompleksnijim okruženjima. Također, ne bismo smjeli pojednostaviti ljudske fizičke karakteristike kružnicom

određenog radijusa, a tako ni fizičke karakteristike robota.

S obzirom na to da je odabrani model već i sada računski vrlo zahtijevan, možemo pretpostavit da bi uvedene prilagodbe za snalaženje u kompleksnijim simulacijama rezultirale još sporijim treniranjem i većim opterećenjem korištenog CPU-a. Unatoč velikoj količini resursa koja bi za ovo bila potrebna, važno je u korištene simulacije uvesti znatno veću dozu realističnosti, kako bismo premostili prepreku između simulacije i stvarnosti te kako bi ti modeli jednog dana mogli robotima omogućiti nesmetano funkcioniranje u suživotu s ljudima.

Literatura

[1] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, i J. Oh, "Core challenges of social robot navigation: A survey", *ACM Transactions on Human-Robot Interaction*, sv. 12, br. 3, str. 1–39, 2023.

[2] E. Alpaydin, *Introduction to Machine Learning*, 2. izd. The MIT Press, 2010.

[3] M. I. Jordan i T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects", *Science*, sv. 349, br. 6245, str. 255–260, 2015.

[4] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions", *SN Computer Science*, sv. 2, br. 3, str. 1–21, 2021.

[5] A. Mathew, P. Amudha, i S. Sivakumari, "Deep learning techniques: An overview", u *Advanced Machine Learning Technologies and Applications*, ser. Advances in Intelligent Systems and Computing, A. E. Hassanien *et al.*, Ur. Singapore: Springer Nature Singapore Pte Ltd., 2021., sv. 1141, str. 599–600. https://doi.org/10.1007/978-981-15-3383-9_54

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, i A. A. Bharath, "Deep reinforcement learning: A brief survey", *IEEE Signal Processing Magazine*, sv. 34, br. 6, str. 26–38, 2017.

[7] F. Stamenković, "Duboke neuronske mreže", *Unknown Journal*, 2015., "Accesed: 2024-11-08".

[8] C. Chen, Y. Liu, S. Kreiss, i A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning", 2019. [Mrežno]. Adresa: https://arxiv.org/abs/1809.08835

[9] J. van den Berg, S. J. Guy, M. Lin, i D. Manocha, “Reciprocal n-body collision avoidance”, u *Robotics Research*, C. Pradalier, R. Siegwart, i G. Hirzinger, Ur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011., str. 3–19.

[10] M. Everett, Y. F. Chen, i J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning”, *CoRR*, sv. abs/1805.01956, 2018. [Mrežno]. Adresa: http://arxiv.org/abs/1805.01956

[11] Y. F. Chen, M. Liu, M. Everett, i J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning”, *CoRR*, sv. abs/1609.07845, 2016. [Mrežno]. Adresa: http://arxiv.org/abs/1609.07845

[12] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, i Z. Zhou, “Robot navigation in a crowd by integrating deep reinforcement learning and online planning”, *CoRR*, sv. abs/2102.13265, 2021. [Mrežno]. Adresa: https://arxiv.org/abs/2102.13265

[13] E. L. Deci i R. M. Ryan, *Self-determination theory*. Springer Science & Business Media, 1985.

[14] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, i P. Abbeel, “Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks”, *CoRR*, sv. abs/1605.09674, 2016. [Mrežno]. Adresa: http://arxiv.org/abs/1605.09674

[15] D. Pathak, P. Agrawal, A. A. Efros, i T. Darrell, “Curiosity-driven exploration by self-supervised prediction”, *CoRR*, sv. abs/1705.05363, 2017. [Mrežno]. Adresa: http://arxiv.org/abs/1705.05363

[16] Y. Burda, H. Edwards, A. J. Storkey, i O. Klimov, “Exploration by random network distillation”, *CoRR*, sv. abs/1810.12894, 2018. [Mrežno]. Adresa: http://arxiv.org/abs/1810.12894

[17] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, i K. Lee, “State entropy maximization with random encoders for efficient exploration”, *CoRR*, sv. abs/2102.09430, 2021. [Mrežno]. Adresa: https://arxiv.org/abs/2102.09430

[18] D. Martinez-Baselga, L. Riazuelo, i L. Montano, “Improving robot navigation in crowded environments using intrinsic rewards”, 2023. [Mrežno]. Adresa:

https://arxiv.org/abs/2302.06554

Sažetak

Navigacija mobilnih robota u gužvama pomoću dubokog potpornog učenja

Barbara Pavlović

U ovom su radu demonstrirana, testirana i uspoređena neka od najpoznatijih rješenja za problem navigacije robota kroz gužve. Svi analizirani modeli temeljeni su na dubokom potpornom učenju, stoga se njihovo treniranje izvodi principom pokušaja i pogrešaka. Uspoređuju se izvedbe LM-SARL algoritma u modelima s različitim postupcima treniranja i funkcijama nagrade, a radi se i analiza utjecaja intrinzičnih nagrada na model s istim algoritmom. Analizira se izvedba modela dubokog potpornog učenja s Q-mrežama, ispitivanjem implementacije algoritma SG-D3QN. Svi su modeli trenirani koristeći jednake eksperimente u okolini kreiranoj programskim jezikom Python, u kojemu su implementirani i ovdje analizirani modeli dubokog učenja. U simulaciji se koristi neholonomski model robota, a testiranja na raznim različitim eksperimentima u simulaciji izvode se na modelima treniranima u okruženju s ljudskim agentima nevidljivim robotom. Radi se analiza zaglađenosti putanje robota kod svih ispitanih modela, kao i analiza razine sigurnosti u navigaciji kroz gužvu. Istakuti su uspjesi i kvaliteta ispitanih modela, no naglašeni su i uočeni nedostaci te mjesto za napredak.

Ključne riječi: robot; gužva; navigacija; duboko učenje; duboko potporno učenje; nagrada; izbjegavanje kolizije; prepreke; LM-SARL; SG-D3QN; ICM; RE3;

Abstract

Crowd-aware mobile robot navigation via deep reinforcement learning

Barbara Pavlović

In this paper, some of the most popular solutions to robot navigation through crowds have been demonstrated, tested and compared. All analyzed models are based on deep reinforcement learning and therefore their training is carried out by the principle of trial and error. Performances of LM-SARL algorithm in models with different training approaches and reward functions are compared, and the impact of the intrinsic rewards on models with the same algorithm is also evaluated. The performance of deep reinforcement learning model with Q-network is analyzed, by examining the implementation of SG-D3QN algorithm. All models were trained using the same set of experiments in an environment created with Python programming language, which was used for the implementation of deep reinforcement learning models as well. A nonholonomic robot was used in the simulation, and tests with various experiments were performed on the models trained in an environment with robot invisible to humans. The analysis of path smoothness and the level of navigation safety is done. The success and quality of the evaluated models have been highlighted, while the identified shortcomings and room for improvement have also been emphasized.

Keywords: robot; crowd; navigation; deep learning; deep reinforcement learning; reward; collision avoidance; obstacles; LM-SARL; SG-D3QN; ICM; RE3;