Automatska navigacija mobilnog vozila kroz nemapirane šumske puteve korišćenjem neuralnih mreža

Nikola Jovičić

***Sadržaj* — U ovom radu je predstavljen sistem za autonomnu navigaciju nemapiranih šumskih puteva u simulaciji, zajedno sa novim simulatorom za testiranje i trening pomenutog sistema. Za navigaciju, koristi kombinaciju planiranja putanje i dubokih neuralnih mreža treniranih uz pomoć imitacionog učenja. Pokazano je da ovaj jednostavan sistem može uspešno da navigira šumskim putevima u simulaciji.**

***Ključne reči* — imitaciono učenje, šuma, duboko učenje, simulacija, planiranje puta**

I. Uvod

Prolazak kroz šume i njihovo mapiranje je spor i naporan proces za ljude. Morali bi da koriste i ažuriraju detaljnu mapu ručno ili uz pomoć GPS uređaja, a ukoliko bi još morali dodatno na to da skupljaju neku vrstu podataka (vlažnost vazduha, temperaturu, stanje puteva) to bi učinilo napor još većim.

U ovom radu predstavljen je sistem za automatsku navigaciju nemapiranih šumskih puteva u simulaciji koji koristi neuralne mreže. Takođe je predstavljen simulator koji je specifično napravljen za treniranje i testiranje ovog sistema.

Iako već postoje sistemi koji koriste neuralne mreže za navigaciju šumskih puteva oni su obično fokusirani na dronove. [1] Većina dronova nema dužu autonomiju leta od 30 minuta, stoga je ovaj sistem napravljen da radi sa mobilnim roverom koji se pokreće na točkove ili gusenice. Pomenuti sistemi za dronove ne rade nikakvo planiranje, stoga ih nije moguće koristiti bez

ljudskog nadzora. Sistem u ovom radu je potpuno automatizovan da za zadatu koordinatu isplanira putanju do nje ukoliko je već zna, ili ukoliko je ne zna da pokuša da pronađe na najbrži put do iste. Ukoliko ga neka putanja odvede do prepreke sistem će ažurirati unutrašnju mapu sa novim informacijama i prestati da koristi tu putanju.

II. POVEZANI RADOVI

Postoji mnogo pristupa koji se bave praćenjem šumskih puteva, ali se većina njih zasniva na niskoletećim dronovima [2, 3]. Ne postoji mnogo objavljenih kompletnih sistema o istraživanju šumskih puteva [3, 8]. Većina se ne bavi autonomnim istraživanjem ili planiranjem i bavi se samo mapiranjem [4], ili mapiranjem prečnika stabala [5] ili restauracijom šuma [6]. Praćenje šumske staze kopnenim vozilom je slično zadatku autonomne vožnje, stoga je u ovom radu korišćena neuralna mreža obučena slično kao u [7]. U poređenju sa sličnim rešenjima [8], ovaj sistem prati putanju od početka do kraja bez ikakve globalne mape, putanje ili prethodnog znanja o okruženju, koristi podatke samo jedne RGB kamere za trening i izvršavanje, i lako se obučava jer su mu potrebni samo podaci o ljudskoj vožnji.

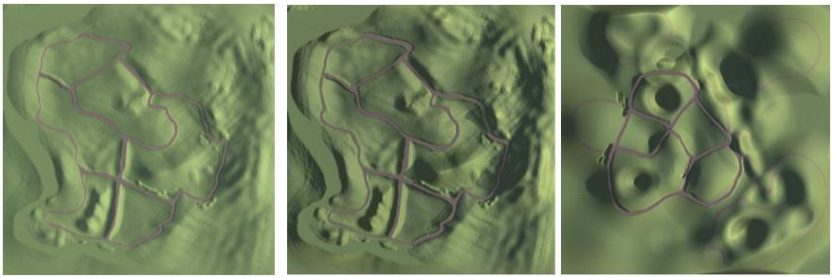
III. IMPLEMENTACIJA SIMULATORA

Prvi korak implementacije ovog sistema je pronalaženje odovarajućeg simulatora. Potrebne mogućnosti simulatora su:

- Realistični prikaz rastinja i drveća
- Detaljno podešavanje kamere
- Model rovera i njegova ručna kontrola
- Mogućnost snimanja podataka lokacije, kontrole i kamere
- Laka izmena terena i puteva za vožnju

U trenutku pisanja nije postojao simulator koji zadovoljava bar većinu ovih kriterijuma i zbog toga je bilo potrebno napraviti novi. Simulator za ovaj rad je napravljen u “Unity3D” editoru za video igre. Svuda gde je moguće korišćeni su besplatni dodaci, za modele drveća i rastinja, teskture zemlje i neba kao i za sam rover, a za komunikaciju sa pajtonom korišćen je “ML-agents” paket.

Za pravljenje terena korišen je ugrađen alat za teren u kojm su onda nasumično napravljeno dve trening mape za rovera, jedna manja i teža i jedna veća sa prostranijim putevima. Takođe je napravljena jedna testna mapa (Sl. 1.).



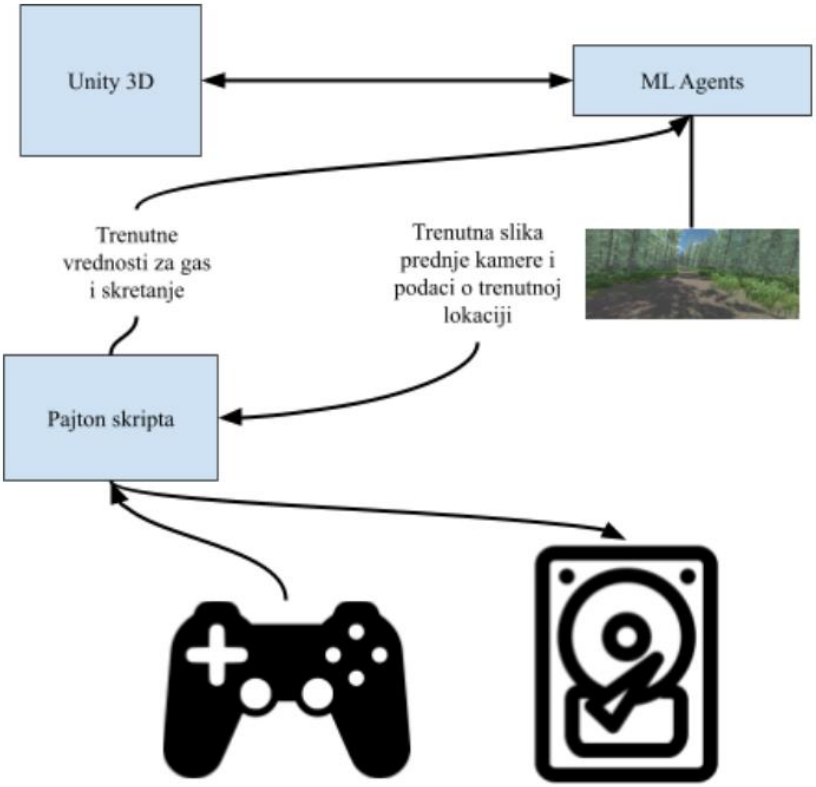
Sl. 1. Dve trening mape i testna mapa

Zatim je korišćen besplatan dodatak "Vegetation Spawner", koji omogućava stvaranje drveća i rastinja na terenu, ali samo na onim delovima koji nisu označeni kao put.

Onda je potrebno napraviti jednostavno vozilo koje bi se kretalo po mapi. Za to je korišćen besplatni "Realistic Buggy Kit", koji nudi nekoliko modela bagi vozila. Vozilu je dodata kamera, zatim je skalirano i ubačeno na mapu kao i podešeno da se može kontrolisati van igre iz pajtona us pomoć "ML-agents" paketa (Sl. 2.).

Sl. 2. Pogled sa kamere vozila unutar simulatora

Za kontrolu je korišćen džojstik preko koga je moguće slati komande za skretanje levo i desno, kao i za gas. Džojstik je korišćen umesto tastature jer je uz pomoć njega moguće slati analogne komande, što nam omogućava bolju kontrolu i kvalitetnije podatke za trening. Sakupljanje podataka je bilo vršeno uz pomoć pajton skripte koji se kačila na simulator, slala komande za upravljanje sa džojstika a zauzvrat dobijala informacije o trenutnoj lokaciji i sliku onoga što vidi prednja kamera vozila, zatim bi čuvala na hard disk: sliku, trenutnu lokaciju i položaj volana (Sl. 3.).



Sl. 3. Pregled sistema za sakupljanje podataka

IV. PROCESIRANJE PODATAKA

Kako bi neuralna mreža mogla da bude trenirana podaci se prvo moraju procesirati u odgovarajući format. Kako bi rešili problem nedostatka adaptacije sakupljene slike koje su širine 1024px a visine 432px je potrebno iseći na tri slike širine 700px: levu, desnu i centralnu, a kontrola volana vozila za te slike se mora pomeriti malo udesno za levu sliku i malo ulevo za desnu sliku. Kako mreža ne bi mogla da nauči da razaznaje koja je leva, desna i centralna slika, levo i desno isecanje se uzimaju nasumično. Leva slika se odseca za nasumično 0-137 piksela sa desne strane dok se desna slika odseca za nasumično 0-137 piksela sa leve strane, prema tome se i podesi

kontrola volana. Zatim su leva, desna i centralna slika sačuvane kao posebni elementi seta podataka.

Kontrola volana je ono što mreža treba da predvidi iz ulazne slike. Ona je pamćena kao broj od -1 do 1, gde je -1 maksimalno levo a +1 maksimalno desno.

Takođe kako bi mogli da kontrolišemo neuralnu mrežu potrebno je označiti gde je potrebno da skrene kada dođe na raskrsnicu, levo, desno ili da nastavi pravo. Komanda za skretanje se šalje konstantno mreži, iako se ona ne nalazi na raskrsnici, time sistem ne mora da pazi kada je poslao komandu, dovoljno je samo da označi mreži da skrene na sledećoj raskrsnici i mreža će skrenuti tek kada dođe na raskrsnicu.

Set podataka za ovaj ulaz u mrežu je sakupljen polu-automatski, ručno su označene sve koordinate raskrsnica, zatim je uz pomoć skripte prođen ceo set podataka. Kada je vozilo ušlo u domet raskrsnice pamćene su komande volana, kada je vozilo izašlo iz dometa raskrsnice uzima se prosek komande volana, ukoliko je prosek veći od -0.1 a manji od 0.1 onda su svi podaci do tog trenutka označeni kao idenje pravo, ukoliko je prosek manji od -0.1 onda su podaci označeni kao skretanje levo, a ako je veći od 0.1 onda su podaci označeni kao skretanje desno. U setu podataka su skretanja označena kao brojevi: 0 za pravo, 1 za levo i 2 za desno.

Kako bi mreža bila lakša za treniranje i kasnije upravljanje, ona je takođe trenirana da predvidi i da li se trenutno nalazi u raskrsnici, i taj podatak je sačuvan kao: 0 ako se ne nalazi i 1 ako se nalazi. Kako bi povećali prisustvo raskrsnica u setu podataka, ti podaci su duplirani. Ovo je urađeno jer su raskrsnice bile prisutne u samo 10% podataka.

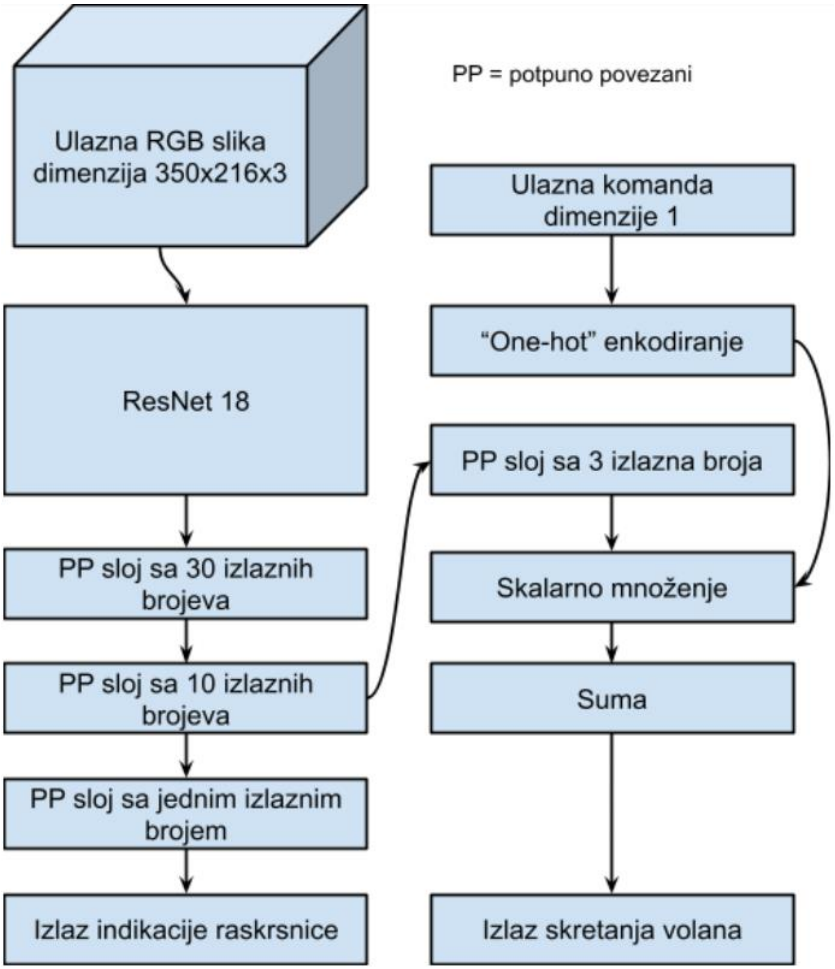
V. NEURALNA MREŽA

Pri dizajnu neuralne mreže korišćena je standardna Resnet18 [9] arhitektura koja se obično koristi za klasifikaciju slika.

Kod mreže je glava zamenjena novom glavom koja podržava dodatni ulaz za kontrolu smera i dva izlaza, jedan za kontrolu volana i drugi za indikaciju da li se trenutno nalazi u raskrsnici. Preciznije, poslednji potpuno povezani sloj mreže je zamenjen sa četiri nova potpuno povezana sloja, prvom je izlaz 30 brojeva, drugom je izlaz 10 brojeva, a treći i četvrti su paralelno povezani na izlaz drugog sloja, gde treći ima izlaz 3 broja a četvrtom je izlaz 1 broj. Time treći sloj predviđa skretanje a četvrti sloj predviđa da li se robot trenutno nalazi u raskrsnici. Ulaz za odabir smera skretanja se enkodira u nule i jedinice dužine 3 i skalarno množi sa izlazom trećeg sloja, dobijeni niz se sumira i time dobijamo komandu za skretanje.

Svi potpuno povezani slojevi na izlazu koriste ReLu [10] aktivaciju sem slojeva pre izlaza. Sloj za izlaz indikacije raskrsnice koristi sigmoid a sloj pre

množenja koristi linearnu aktivaciju. Za definiciju mreže i trening je korišćena PyTorch biblioteka [11] (Sl. 4.).



Sl. 4. Arhitektura neuralne mreže

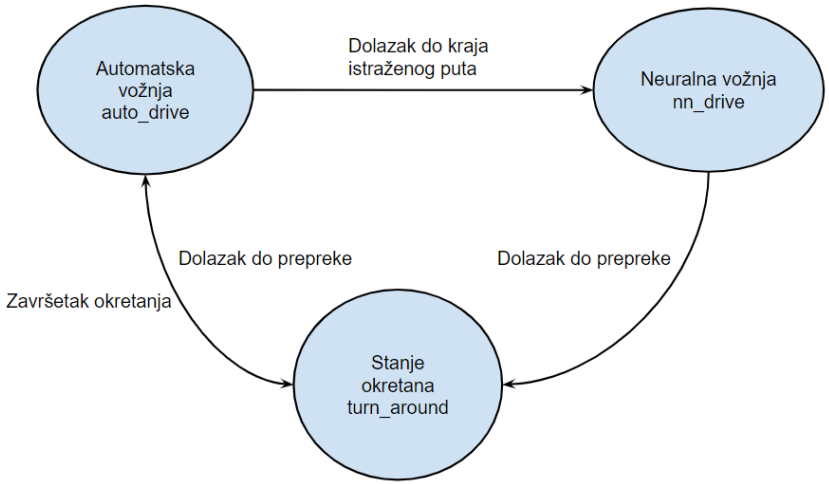
Za trening je korišćen Adam [12] optimizator sa 0.001 početnom stopom učenja, takođe je korišćen spuštač stope učenja, koji ukoliko gubitak ne napreduje 10 epoha duplo smanjuje stopu učenja. Trening je vršen u serijama od 64 podatka po koraku. Najbolji model je treniran 26 epoha na NVIDIA 1080TI grafičkoj kartici i to je trajalo oko 2 sata.

Time dobijamo neuralnu mrežu koja za zadatu sliku i informaciju o skretanju na ulazu predviđa za koliko je potrebno okrenuti volan robota kako bi on ostao na stazi i sledeo zadato skretanje i da li se trenutno nalazi u raskrsnici.

VI. KORIŠĆENJE NEURALNE MREŽE ZA NAVIGACIJU

Ukoliko bi se ovakva mreža koristila za navigaciju korisnik bi morao da joj ručno govori gde da skrene na svakoj raskrsnici. Takođe, pošto mreža nema unutršnje stanje ona ne bi mogla da razazna da li je već bila na nekoj lokaciji ili je već prošla kraći put do tamo. Stoga je potrebno da ima kontrolni sistem koji će dati mogućnost korisniku da komanduje sistemom preko ciljne tačke, pamtiti koje lokacije su već obiđene, kao i proračunava najbolji način da se stigne na neku lokaciju. Za potrebe ovog rada smo koristili precizne podatke o lokaciji koje daje simulator, ali u pravom svetu za tako nešto bi mogao da se koristi sistem vizualne odometrije, GPS ili njihova kombinacija [13].

Kontrolni sistem se sastorji iz tri stanja, neuralna vožnja, automatska vožnja i stanje okretanja (Sl. 5.).



Sl. 5. Konačni automat stanja sistema sa uslovima prelaska

A. *Stanje neuralne vožnje*

U ovom stanju se koristi neuralna mreža za kontrolu vozila. Kako bi stigla do zadate ciljne tačke automatski se zadaju komande za skretanje neuralnoj mreži prema trenutnoj lokaciji, već obiđenih lokacija i orjentaciji vozila. Kako bi se vozilo obeshrabrilo da ide na već obiđene tačke računa se ugao

između vozila i svih tačaka, kao i distanca od vozila do svih tačaka, inverzne vrednosti distance se upisuju u histogram prema uglu, zatim se u uglađenom nizu rotacija pronalazi dolina najbliža uglu između vozila i ciljne tačke, time će vozilo izbegavati već obiđene puteve. Pošto je neuralnoj mreži potrebno reći gde da skrene na raskrsnici dobijena vrednost rotacije se diskretizuje na levo, desno i pravo. Ova jednostavna kontrolna šema neće uvek naći najbrži put, ali će odvesti vozilo do destinacije u nekom trenutku. Dok se vozilo kreće sistem pamti lokaciju i pravi graf svih obiđenih tačaka. Ukoliko se vozilo nađe na lokaciji koja je već sačuvana u grafu, trenutna putanja će na tom mestu biti spojena sa već obiđenom. Ako se nađe ispred prepreke vozilo prelazi u stanje okretanja. Algoritam ovog stanja je opisan opisan kao (Algoritam 1)

Algoritam 1 Stanje neuralne vožnje
Input*: goal coordinate* G, *current map graph* M, *current image view from simulator* I, *current position* P, *current rotation* R, *neural network* N, *simulator* S
While P!=G: *// While goal not reached*
```
   rot_to_goal = rotation_between((P, R), G)
   all_distances = L2 norm(P-coordinates(M))
   inverse_distances = log2(map width – all_distances)
   all_rots = rotation_between((P, R), coordinates(M))
   hist = bincount(all_rots, weight=inverse_distances)
   smooth_hist = gaussian_filter1d(hist, sigma=6)
   extreme_locs = find_extreme_minima(smooth_hist)
   rot_to_goal = nearest(extreme_locs, rot_to_goal)
   If rot_to_goal > 0.3:
      command = 1
   Else if rot_to_goal < -0.3:
      command = 2
   Else:
      command = 0
   I = preprocess_img(I)
   M.add_and_connect_node(P)
   predicted_steering = N(I, command)
   P, R, I, obstacle = S(predicted_steering)
   If obstacle:
      change_state(turn_around_state)
End
```

B. Stanje neuralne vožnje

Pri svakom novom izdavanju komande sistem počinje u ovom stanju. Napraviće putanju između trenutne lokacije vozila i tačke u grafu najbliže ciljnoj tački. Kretaće se po putanji dok ne dođe do kraja putanje ili prepreke. Ukoliko dođe do kraja putanje a nije uspeo da dođe do ciljne tačke preći će u stanje neuralne vožnje. Ukoliko dođe do prepreke, prekinuće graf na tom mestu i preći u stanje okretanja. Algoritam ovog stanja je opisan kao (Algoritam 2)

Algoritam 2 Stanje automatske vožnje
Input*: goal coordinate* G, *current map graph* M, *current position* P, *current rotation* R, *simulator* S
While P!=G: *// While goal not reached*
    start_point = argmin(L2 norm(coordinates(M) – P))
    end_point = argmin(L2 norm(coordinates(M) – G))
    path = M.shortest_path(start_point, end_point)
    For point in path: *// Follow saved path*
        rot_to_target = rotation_between((P, R), point)
        P, R, I, obstacle = S(rot_to_target)
        If obstacle:
            change_state(turn_around_state)
    If P!=G: *// Reached end of explored path*
        change_state(neural_network_drive)
End

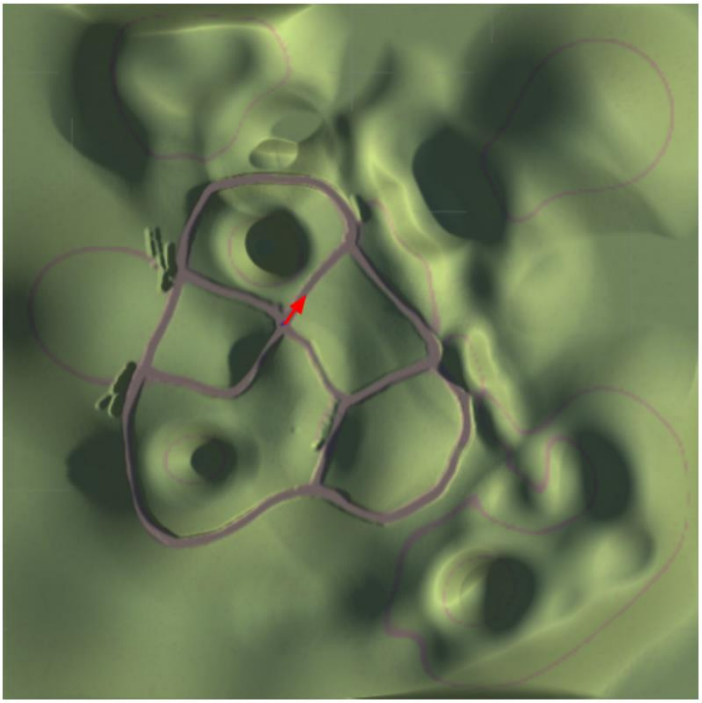
C. Stanje neuralne vožnje

Ovo stanje daje komande za okretanje vozila za 180 stepeni od trenutnog smera kretanja. Kada završi polukružni okret vozilo prelazi u stanje automatske vožnje.

VII. EKSPERIMENTALNI REZULTATI

Robot je testiran na potpuno novoj mapi koju nikada nije video. Testiran je na različitim konfiguracijama prepreka i ciljnih tačaka. Slučajevi testiranja su svrstani u tri kategorije, lake, srednje i teške.

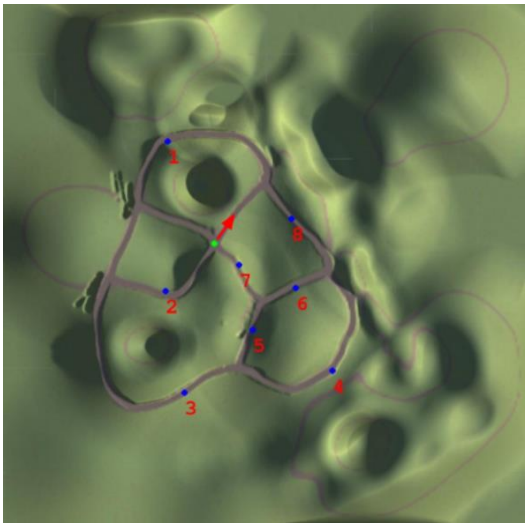
Laki slučajevi uključuju odlazak robota na neku tačku na mapi bez ikakvih prepreka, srednji slučajevi uključuju odlazak na tačku sa preprekama a teški slučajevi uključuju odlazak na više tački jednu posle druge sa preprekama koje menjaju mesto. Svi rezultati su sumirani na sledećem video snimku [14].



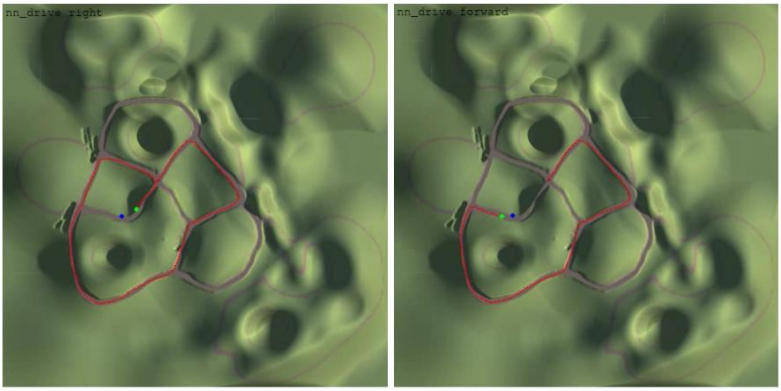
Sl. 6. Testna mapa sa početnom pozicijom označenom sa crvenom strelicom

A. Slučajevi lake težine

Za svaki od slučaja sa slike vozilo je uspelo da dođe na željenu lokaciju. Svaki slučaj je pokrenut 10 puta zbog nasumičnosti simulatora i neuralne mreže. Samo za slučaj broj 2 vozilo nije uspelo nijednom da nađe optimalnu putanju, dok je za ostale ili išlo uvek optimalnom putanjom ili više od pola pokretanja.



Sl. 7. Testni slučajevi lakse težine, plave tačke su ciljne tačke, brojevi označavaju broj slučaja



Sl. 8. Odabrane putanje slučaja broj 2

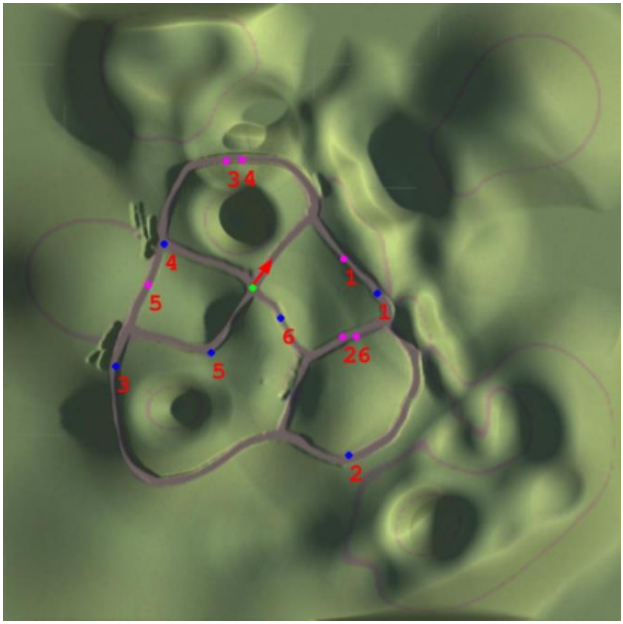
B. Slučajevi srednje težine

Čim se uvedu prepreke problem postaje komplikovaniji, vozilo više ne može da koristi najbrži put da stigne do cilja nego je potrebno da istražuje mapu kako bi pronašao pravo rešenje. Svaka prepreka i cilj su označeni brojem prema tome kome testnom slučaju pripadaju. U četvrtini testnih slučajeva (16 od 60), vozilo nije uspelo da nađe put do cilja, ili zbog predugog kretanja po mapi, ili zbog spadanja sa puta. Samo za slučajeve 3 i 5

model je uspeo da pronađe put do cilja svaki put. Vozilo je testirano 10 puta na svakom slučaju. (Tabela 1)

TABELA 1: USPEŠNOST SNALAŽENJA MODELA ZA SLUČAJEVE SREDNJE TEŽINE.

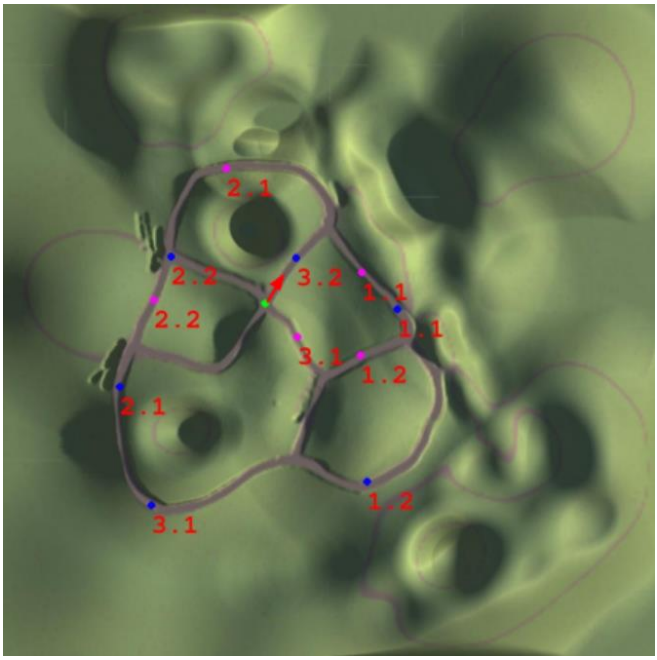
Broj slučaja	1	2	3	4	5	6
Procenat uspešnih pokušaja	30%	80%	100%	70%	100%	60%



Sl. 9. Testni slučajevi srednje težine, plave tačke su ciljne tačke, roze tačke su prepreke, brojevi označavaju broj slučaja kome tačka pripada

C. Slučajevi teške težine

Ovi slučajevi ispituju ceo sistem. Od njegovog snalaženja uz pomoć neuralne mreže do automatske navigacije kada je mapa napravljena, kao i ponašanje kada se prepreke pojave na već istraženim delovima mape. Ponovo su plave tačke ciljevi a roze tačke prepreke, svaki testni slučaj se izvršava iz dve faze, “X.1” i “X.2”. Sistem je pokrenut 20 puta za svaki slučaj.



Sl. 10. Testni slučajevi teške težine, plave tačke su ciljne tačke, roze tačke su prepreke, brojevi označavaju broj slučaja kome tačka pripada

TABELA 2: USPEŠNOST SNALAŽENJA MODELA ZA TEŠKE SLUČAJEVE.

Broj slučaja	1	2	3
Procenat uspešnih pokušaja 1. faze	40%	85%	100%
Procenat uspešnih pokušaja 2. faze	40%	70%	100%

VIII. PRONAĐENI PROBLEMI SISTEMA

U ovom poglavlju ćemo se pozabaviti najčešćim greškama sistema i mogućim načinima da se one isprave.

A. Spadanje sa puta

Ukoliko neuralna mreža ne uspeva da održi vozilo na putu, onda to može imati jako loše posledice. Vozilo može da se sudari sa drvećem ili stenom pored puta ili da potpuno spadne sa njega.

Jedan način da se ovaj problem izbegne je treniranje bolje neuralne mreže, koristeći više podataka i/ili veću mrežu. Drugi način je korišćenje boljeg senzora, npr dubinske kamere, kako bi mreža raspolagala sa više informacija. Sa trenutnom mrežom ovaj problem nije čest, ali za korišćenje mreže u realnom svetu bi bilo potrebno dosta više podataka za trening.

B. Polukružno okretanje

Pošto vozilo nema mogućnost okretanja u mestu, za okretanje je implementirano polukružno okretanje, ovo okretanje ne posmatra prepreke, niti ivicu puta. Stoga ukoliko je put uzak vozilo može da spadne sa njega ili da se udari u prepreku ili drvo pored puta.

Najlakši način da se ovo reši bi bio da se vozilo opremi sa jednokanalnim lidarom sa pogledom na 360 stepeni oko vozila, time bi vozilo moglo na siguran način da se okrene.

C. Mašenje mogućeg skretanja

Ponekad se dogodi da mreža ima instrukciju za skretanje, ali ne primeti da je skretanje moguće i stoga ne izvrši instrukciju. Ovaj problem bi mogao da se izbegne korišćenjem kamere koja je šireg ugla, time bi mreža lakše videla da je skretanje moguće.

IX. ZAKLJUČAK

U oblasti istraživanja robotizovanih vozila na četiri točka ovaj rad je doprineo simulator šumskog okruženja, kao i sistem koji uspešno rešava scenarije pomenutog simulatora. Sistem je detaljno testiran u simulatoru na različitim scenarijima prepreka i ciljnih tačaka. Uz pomoć napravljenog sistema pokazano je kako je moguće ostvariti jednostavno mapiranje šumskih puteva uz kombinaciju klasičnih metoda planiranja putanje i novih metoda neuralnih mreža. U radu su opisani problemi sistema kao i njihova potencijalna rešenja. Sledeći korak bi bio korišćenje ovog sistema u pravom svetu. Kako bi to izveli bilo bi potrebno istrenirati postojeću neuralnu mrežu sa slikama iz pravog sveta i dodati sistem odometrije. Ostatak sistema bi ostao nepromenjen.

LITERATURA

[1] A. Giusti et al., "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 661-667, July 2016 [2]

[2] Smolyanskiy N. et al “Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness”

[3] Ghamry, et al “Cooperative Forest Monitoring and Fire Detection Using a Team of UAVs-UGVs”, in ICUAS, 2016
[4] Pierzchała M. et al “Mapping forests using an unmanned ground vehicle with 3D LiDAR and graphSLAM”, in Computers and Electronics in Agriculture, Volume 145, 2018
[5] Chisholm R. et al “UAV LiDAR for below-canopy forest surveys”, in Journal of Unmanned Vehicle Systems 01(01):61-68, December 2013
[6] Mohan M. et al “UAV-Supported Forest Regeneration: Current Trends, Challenges and Implications”, in Remote Sensing, 2021
[7] Bojarski M. et al “End to End Learning for SelfDriving Cars”, in arXiv preprint arXiv:1604.07316, 2016
[8] Grigorescu S. “Embedded Vision for Self-Driving on Forest Roads”. on CVPR, Workshop on Embedded Vision, 2021
[9] He K et al “Deep residual learning for image recognition”, 2015
[10] Agarap, A. F. “Deep Learning using Rectified Linear Units (ReLU)”, 2018
[11] Paszk A. et al “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, 2019
[12] Diederik P. K., Ba J. “Adam: A Method for Stochastic Optimization”, in 3rd International Conference for Learning Representations, 2015
[13] Wang K. et al “Approaches, Challenges, and Applications for Deep Visual Odometry: Toward to Complicated and Emerging Areas”, in IEEE Transactions on Cognitive and Developmental Systems, 2020
[14] Jovičić N.: “Automated unmapped forest path navigation of mobile rover using neural networks”, video file, November 2021, retrieved from https://youtu.be/nr6jNAzg1TQG.

ABSTRACT

In this paper, a system for autonomous navigation of unmapped forest paths in a simulation, along with a new simulator for testing and training it is presented. For navigation, it uses a combination of path planning and deep neural networks trained with imitation learning.

AUTOMATED UNMAPPED FOREST PATH NAVIGATION OF MOBILE ROVER USING NEURAL NETWORKS

Nikola Jovičić