Universitatea Transilvania din Brașov

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea de INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

Bogdan TRĂSNEA (SIBIȘAN)

Integrarea fuziunii de senzori și a tehnicilor de inteligență artificială pentru planificarea avansată a traseului autovehiculelor autonome
Integrating sensor fusion and artificial intelligence techniques for advanced path planning of autonomous vehicles

REZUMAT

Conducător ştiinţific

Prof. dr. ing. Claudiu POZNA

BRAȘOV, 2024

Cuprins

Subiectul tezei de doctorat 1

Obiectivele cercetării 2

Structura tezei 4

Dezvoltarea mediilor de simulare 5
- GridSim . . . 6
- Robotul Raspberry PI . . . 11
- Agile Scout . . . 13

Planificarea traseului în 2D 16
- Definiția problemei . . . 17
- Arbitrarea comportamentului . . . 17
- Rețeaua neuronală NeuroTrajectory . . . 21

OctoPath - extinderea la reprezentarea 3D 23
- Arbori de tip OcTrees . . . 24
- Octomap . . . 26
- Arhitectura rețelei neuronale de tip encoder-decoder . . . 27
- Evaluarea performanței . . . 29

Contribuții personale 33
- Analiza literaturii de specialitate . . . 33
- Dezvoltarea mediilor de simulare . . . 34
- Dezvoltarea algoritmilor . . . 34
- Evaluare și benchmarking . . . 35

Diseminarea rezultatelor cercetării 35

Concluzii și direcții viitoare de cercetare 37

Subiectul tezei de doctorat

Lucrarea de față propune un algoritm bazat pe învățare care permite planificarea eficientă și precisă a traiectoriei pentru roboți mobili și vehicule autonome. Conducerea autonomă reprezintă o schimbare transformatoare în industria auto, promițând să îmbunătățească siguranța, eficiența și confortul în transport. Pe măsură ce tehnologia avansează, se preconizează că va revoluționa nu numai transportul personal, ci și logistica, transportul public și alte sectoare care depind de mobilitatea vehiculelor. Evoluția către vehicule complet autonome cuprinde un spectru larg de inovații tehnologice, inclusiv fuziunea senzorilor, învățarea automată, planificarea traseului și integrarea sistemelor în timp real.

Dezvoltările recente din domeniile deep learning și inteligenței artificiale au sprijinit avansarea rapidă a domeniului conducerii autonome. Vehiculele autonome (VA) sunt sisteme robotice care pot naviga fără intervenția umană. Implementarea vehiculelor autonome se preconizează că va avea un impact major asupra viitorului mobilității, aducând o varietate de beneficii în viața de zi cu zi, cum ar fi simplificarea conducerii, creșterea capacității rețelei rutiere și minimizarea accidentelor vehiculare.

Tehnologia conducerii autonome integrează o varietate de sisteme avansate pentru a permite vehiculelor să navigheze și să funcționeze fără intervenția umană. Aceste sisteme includ fuziunea senzorilor, care combină date de la mai mulți senzori (de exemplu, LiDAR, radar, camere) pentru a crea o înțelegere cuprinzătoare a mediului vehiculului. Învățarea automată și IA sunt utilizate pentru a interpreta datele senzorilor, a lua decizii și a învăța din experiențele de conducere pentru a îmbunătăți performanța în timp. Planificarea traseului implică dezvoltarea de rute optime și ajustări în timp real pentru a naviga prin medii dinamice. Integrarea sistemelor în timp real asigură că toate sistemele vehiculului funcționează armonios și răspund instantaneu la schimbările din mediu.

În ciuda progreselor semnificative, există mai multe provocări critice care împiedică adoptarea pe scară largă a vehiculelor autonome. Asigurarea robusteței și fiabilității sistemelor de percepție în diverse condiții de mediu, cum ar fi condițiile meteorologice variabile, iluminarea și condițiile rutiere, este o provocare semnificativă. Proiectarea interfețelor și protocoalelor pentru interacțiunea fără cusur între vehiculele autonome și utilizatorii umani, inclusiv șoferii, pasagerii și pietonii, rămâne complexă. Dezvoltarea standardelor de reglementare și a cadrelor legale cuprinzătoare care abordează responsabilitatea, siguranța și considerentele etice este, de asemenea, esențială.

Protejarea vehiculelor autonome împotriva amenințărilor cibernetice care ar putea compromite siguranța și confidențialitatea este o preocupare crucială. Abordarea dilemelor etice legate de luarea deciziilor în scenarii de accidente inevitabile și înțelegerea impactului societal mai larg al vehiculelor

autonome sunt necesare. În plus, înțelegerea și atenuarea efectelor automatizării asupra ocupării forței de muncă, planificării urbane și politicii publice sunt critice.

Dezvoltarea tehnologiei conducerii autonome este intrinsec legată de progresele în robotică mobilă. Roboții mobili au servit de mult timp ca fundament experimental pentru sistemele de navigație autonomă, oferind informații esențiale despre luarea deciziilor în timp real, percepția mediului și planificarea dinamică a traseului. Tehnicile de cartografiere și înțelegere a mediilor complexe dezvoltate pentru roboții mobili sunt direct aplicabile scenariilor de conducere urbană. Algoritmii pentru evitarea obstacolelor și optimizarea traseului în roboții mobili sunt fundamentali pentru autonomia vehiculelor. Strategiile de procesare a datelor senzorilor și luarea deciziilor într-un timp foarte scurt în roboții mobili sunt esențiale pentru conducerea autonomă în siguranță.

Privind înainte, viitorul conducerii autonome se așteaptă să fie modelat de mai multe progrese cheie. Îmbunătățirile continue ale algoritmilor IA vor spori capacitățile de luare a deciziilor și fiabilitatea sistemelor autonome. Progresele în tehnologia senzorilor, cum ar fi LiDAR-ul de înaltă rezoluție și sistemele radar mai sofisticate, vor îmbunătăți percepția mediului și siguranța. Comunicarea îmbunătățită între vehicule, infrastructură și alți utilizatori ai drumului va facilita rețele de transport mai eficiente și mai sigure.

Vehiculele autonome vor deveni componente integrale ale infrastructurilor orașelor inteligente, contribuind la gestionarea optimizată a traficului și reducerea congestiei urbane. Dezvoltarea continuă a standardelor de reglementare va oferi linii directoare mai clare și va facilita implementarea vehiculelor autonome. Colaborarea mai mare între cercetători, părțile interesate din industrie și factorii de decizie politică va stimula inovația și va aborda provocările multifacetate ale conducerii autonome.

Drumul către conducerea complet autonomă este atât interesant, cât și provocator, cu progrese substanțiale dependente de cercetarea și colaborarea interdisciplinară continuă. Prin valorificarea progreselor în robotică mobilă și abordarea provocărilor inerente, realizarea vehiculelor autonome sigure și eficiente va deveni mai aproape de realitate. Această transformare promite să redefinească transportul, să îmbunătățească siguranța și să creeze noi oportunități în diverse sectoare, anunțând o nouă eră în mobilitate.

Obiectivele cercetării

Principalul obiectiv al acestei cercetări este dezvoltarea, implementarea și testarea riguroasă a unui algoritm bazat pe învățare care să permită planificarea eficientă și precisă a traseelor pentru roboți mobili și vehicule autonome. Prin valorificarea celor mai recente progrese în domeniul deep learning și inteligenței artificiale, acest obiectiv își propune să îmbunătățească capacitatea vehiculului de a naviga autonom prin medii dinamice și imprevizibile. Algoritmul ar trebui să fie capabil să

proceseze datele senzoriale și să genereze rute optime în timp real, luând în considerare atât obstacolele statice, cât și cele dinamice.

Realizarea acestui obiectiv va duce la dezvoltarea unui algoritm de planificare a traseelor bazat pe învățare, validat, eficient și de încredere. Acest lucru va îmbunătăți semnificativ capacitatea vehiculului autonom de a naviga prin medii complexe, asigurând rute sigure și optime. Implementarea cu succes a unui astfel de algoritm este crucială pentru adoptarea și integrarea pe scară largă a vehiculelor autonome în sistemele de transport de zi cu zi, îmbunătățind în final siguranța rutieră, eficiența și convenienta.

Pe baza acestui obiectiv principal, au fost identificate următoarele obiective specifice, care au fost împărțite în patru categorii distincte:

1. Revizuirea completă a literaturii de specialitate
    - Realizarea unui studiu detaliat al literaturii existente în domeniile navigației autonome și planificării traseelor.
    - Identificarea metodologiilor, tehnologiilor și lacunelor actuale din cercetările legate de vehiculele autonome.
2. Dezvoltarea mediului de simulare
    - Crearea unui instrument de simulare pentru a stabili un mediu de testare controlat pentru algoritmii propuși.
    - Asigurarea că instrumentul de simulare reprezintă cu acuratețe scenariile reale de conducere pentru a valida eficacitatea algoritmilor.
3. Dezvoltarea algoritmilor
    - Combinarea datelor de la multiple senzori, precum LiDAR, radar și camere, pentru a crea o înțelegere cuprinzătoare a împrejurimilor vehiculului.
    - Asigurarea că abordarea de fuziune a senzorilor îmbunătățește acuratețea și fiabilitatea percepției vehiculului autonom.
    - Dezvoltarea, implementarea și testarea unui algoritm bazat pe învățare pentru a permite planificarea eficientă și eficace a traseelor pentru roboți mobili și vehicule autonome.
    - Dezvoltarea unui mediu de dezvoltare sigur pe placa NVidia AGX Xavier.
    - Concentrarea pe integrarea modelelor de deep learning și a tehnicilor de inteligență artificială pentru a spori robustețea și acuratețea planificării traseelor.
    - Dezvoltarea de rute optime și realizarea de ajustări în timp real pentru a naviga în medii dinamice.
4. Evaluare și benchmarking
    - Asigurarea că algoritmii de planificare a traseelor pot gestiona eficient atât obstacolele statice, cât și cele dinamice.

- Includerea diverselor metrici de performanță și scenarii de testare pentru a garanta fiabilitatea sistemului în diverse condiții.
- Asigurarea că algoritmii propuși funcționează optim pe diverse seturi de date și se adaptează la diferite condiții de conducere.

Aceste obiective urmăresc să asigure dezvoltarea unui sistem robust, fiabil și sigur pentru planificarea avansată a traseelor în vehicule autonome, valorificând cele mai recente progrese în fuziunea senzorilor și inteligența artificială.

Structura tezei

Teza este organizată în următorul mod:

Capitolul 1 oferă o prezentare generală a subiectului de cercetare în domeniul conducerii autonome, definind contextul și explicând semnificația sa științifică. De asemenea, capitolul subliniază problemele care trebuie abordate, obiectivele principale, precum și acțiunile și metodele utilizate pe parcursul activității de cercetare. În final, capitolul se încheie prin rezumarea structurii și a conținutului general al tezei.

O introducere amplă a tehnologiei de învățare profundă este prezentată în Capitolul 2. Designurile fundamentale ale rețelelor neuronale artificiale, cum ar fi rețelele neuronale convoluționale și recurente, precum și conceptul de învățare prin întărire profundă, sunt acoperite în acest capitol. În plus, tendințele actuale în domeniul învățării profunde pentru planificarea traseelor și arbitrajul comportamental, precum și pentru percepția și localizarea scenelor de conducere, sunt discutate în detaliu. La sfârșitul capitolului, este abordat aspectul programării în domeniul învățării profunde, cu accent pe două cadre Python bine cunoscute: Tensorflow și PyTorch.

Capitolul 3 descrie toate platformele experimentale utilizate și construite în timpul studiului tezei, detaliind dezvoltarea instrumentelor și sistemelor critice pentru cercetarea roboților autonomi. Capitolul începe cu o prezentare generală a mediului de simulare GridSim, un motor de simulare a conducerii autonome care folosește o arhitectură robotizată similară cu un vehicul pentru a crea grile de ocupare utilizând senzori simulați. De asemenea, GridSim este utilizat pentru a investiga performanța a două abordări de învățare profundă: învățarea prin întărire profundă și învățarea comportamentală în conducere utilizând algoritmi genetici. Prototipul Robotului Raspberry Pi, care este construit pe un sistem de tracțiune spate și direcție frontală, este apoi detaliat, comparabil cu modelele din competițiile de vehicule inteligente. Sunt documentați pașii de dezvoltare a prototipului, precum și testele efectuate cu versiunea finală echipată cu Lidar și cameră. În final, este prezentată principala platformă robotică utilizată pentru a testa și valida algoritmii descriși în această teză. Robotul este o platformă mobilă cu tracțiune diferențială, echipată cu camere quad e-Cam130A și un Lidar Hesai

Pandar de 360 de grade, 40 de canale. Ulterior, este oferită o cronologie a dezvoltării robotului și a experimentelor efectuate.

Algoritmii cheie creați în timpul acestei teze sunt prezentați în Capitolul 4 și Capitolul 5, care constituie esența tezei. Capitolul 4 începe cu o scurtă descriere a reprezentării pe bază de grile bidimensionale și continuă cu o prezentare a problemei, planificarea traseului pentru mașinile autonome, care indică capacitatea unui vehicul autonom de a găsi o rută între două puncte, și anume o poziție de plecare și o locație țintă. Ulterior, prezentarea se concentrează pe conceptul de arbitraj comportamental, care va fi discutat din perspectiva descrierii, analizei și modelării scenelor de conducere, precum și rezultatele simulării. În plus, este discutată perspectiva estimării traiectoriei ca problemă de învățare cognitivă. Conține informații despre cum să antrenezi o rețea neurală profundă pentru a anticipa traiectoriile locale ale stării vehiculului autonom prin antrenament neuroevolutiv. Soluția, denumită Neuro-Trajectory, este o metodă neuroevolutivă multiobiectiv pentru învățarea traiectoriilor de stare locală pentru conducerea autonomă, în care o rețea neurală profundă de percepție-planificare estimează traiectoria de stare dorită a vehiculului autonom pe un orizont de predicție limitat. Problema planificării mișcării este uneori denumită o provocare de mapare secvență-la-secvență sau o sarcină de generare a secvenței.

Capitolul 5 detaliază extinderea la o reprezentare 3D a măsurătorilor de intrare și fuziunea senzorilor, și anume OcTrees. Deoarece acestea includ bucle de feedback dependente de timp, design-urile rețelelor neuronale recurente sunt utilizate pentru a rezolva o astfel de problemă. Rezultatul este OctoPath, care este o rețea neurală profundă encoder-decoder auto-supervizată ce prezice cursul optim local pentru vehiculul autonom. De asemenea, este discutată instalarea OctoPath pe un Nvidia AGX Xavier, precum și scenariile și rezultatele măsurării performanței.

În final, Capitolul 6 rezumă principalele descoperiri și formulează concluziile finale. De asemenea, subliniază contribuțiile individuale, diseminarea rezultatelor cercetării, și anume brevetele și lucrările, și conturează direcțiile de cercetare viitoare, concentrându-se pe îmbunătățirea integrării senzorilor, sporirea robusteții algoritmilor și abordarea considerațiilor etice și de siguranță în conducerea autonomă.

Dezvoltarea mediilor de simulare

Simulatoarele oferă o scalabilitate incomparabilă, permițând dezvoltatorilor să ruleze mii de scenarii de testare simultan. Această scalabilitate este esențială pentru validarea riguroasă a algoritmilor de planificare a traseelor, deoarece permite testarea extinsă pe o gamă largă de condiții și parametri. Prin valorificarea puterii de calcul disponibile în simulatoarele moderne, dezvoltatorii pot efectua evaluări la scară largă care ar fi impracticabile sau imposibile în lumea reală. Această capaci-

tate accelerează procesul de dezvoltare și îmbunătățește robustețea și fiabilitatea algoritmilor.

În testările din lumea reală, reproducerea unor scenarii specifice cu condiții exacte este adesea dificilă. Simulatoarele, însă, oferă control precis asupra tuturor aspectelor mediului de testare, asigurând că scenariile pot fi replicate în mod constant. Această reproducibilitate este vitală pentru depanarea și rafinarea algoritmilor, deoarece permite dezvoltatorilor să testeze repetat condiții specifice și să compare rezultatele în mod fiabil. Reproducibilitatea constantă asigură că îmbunătățirile și optimizările pot fi evaluate cu acuratețe, conducând la soluții de planificare a traseelor mai eficiente și mai fiabile.

Simulatoarele oferă capacități cuprinzătoare de colectare a datelor, capturând informații detaliate despre fiecare aspect al performanței vehiculului și al mediului. Aceste date sunt neprețuite pentru analiza comportamentului algoritmilor de planificare a traseelor, identificarea slăbiciunilor și efectuarea de îmbunătățiri informate. Dezvoltatorii pot accesa seturi de date bogate care includ citiri ale senzorilor, dinamica vehiculului, condițiile de mediu și interacțiunea cu alte entități. O astfel de colectare detaliată a datelor este adesea dificilă și costisitoare de realizat în testările din lumea reală, dar este ușor disponibilă în mediile de simulare.

Capacitatea de a itera rapid și de a testa schimbările într-un mediu simulat accelerează semnificativ ciclul de dezvoltare. Dezvoltatorii pot implementa rapid noi funcționalități, testa impactul acestora și rafina algoritmii pe baza feedback-ului obținut din rezultatele simulării. Această iterație rapidă este crucială pentru a ține pasul cu avansurile rapide în tehnologia vehiculelor autonome. Simulatoarele permit dezvoltatorilor să experimenteze idei inovatoare și să încorporeze tehnici de ultimă oră în algoritmii de planificare a traseelor mai eficient.

Utilizarea simulatoarelor pentru dezvoltarea algoritmilor de planificare a traseelor oferă numeroase avantaje, inclusiv un mediu controlat, siguranță sporită, eficiență a costurilor, scalabilitate, reproducibilitate, colectare cuprinzătoare de date și un ciclu de dezvoltare accelerat. Aceste beneficii contribuie colectiv la dezvoltarea unor algoritmi de planificare a traseelor robuști, fiabili și eficienți pentru vehiculele autonome. Prin valorificarea puterii simulării, dezvoltatorii pot depăși multe dintre provocările asociate cu testarea în lumea reală și pot avansa capacitățile sistemelor de conducere autonomă.

GridSim

Simulatorul de conducere dezvoltat este numit GridSim și este descris ca un simulator de conducere autonomă care utilizează cinematica robotului auto non-holonomic. A fost dezvoltat de la zero pentru a sprijini dezvoltarea și validarea sistemelor de conducere autonomă. Acesta conține un meniu care permite comutarea între multiple scenarii, care sunt ușor reprezentate și încărcate în simulator ca fundaluri.

Senzorii simulați au un câmp vizual (FOV) de 120 de grade. Aceștia reacționează atunci când detectează un obstacol, marcându-l ca zonă ocupată. Obstacolele statice sunt mapate a priori pe fundaluri ca liste de poligoane. Senzorii simulați verifică continuu dacă razele de percepție se ciocnesc cu poligoanele date. Obstacolele dinamice sunt reprezentate de mașini de trafic, ale căror traiectorii sunt generate aleatoriu dintr-o distribuție uniformă a spațiilor libere posibile în cadrul scenariului

dat. Viteza longitudinală și rata de schimbare a unghiului de direcție sunt incluse în traiectorie. Toate caracteristicile GridSim au fost construite folosind Python și biblioteca PyGame.

Modelul cinematic cu pistă unică este descris mai detaliat. Pozițiile roților din față și din spate sunt p_f și p_r, respectiv, în timp ce α este unghiul de orientare care descrie direcția vehiculului, definit de unghiul dintre vectorii $\hat{e}_x$ și $p_f - p_r$. Pentru claritate, s-a considerat o ipoteză de "no-slip" pentru roțile pe suprafața de conducere. Alunecarea roților poate fi modelată prin adăugarea efectelor inerțiale generate de sol asupra anvelopelor vehiculului, similar modelului de anvelopă al lui Pacejka.

Roțile se rotesc liber în jurul axelor lor de rotație, în timp ce direcția este modelată prin unghiul δ ca un grad suplimentar de libertate pe roata din față. Vehiculul respectă ipoteza "non-holonomic", exprimată ca o constrângere diferențială asupra mișcării mașinii. Constrângerea non-holonomică împiedică vehiculul să facă deplasări laterale fără a se mișca simultan înainte.

Viteza de deplasare înainte este:

$$v_r = \dot{p}_r \cdot \frac{(p_f - p_r)}{||p_f - p_r||}, \tag{1}$$

unde v_r este magnitudinea lui $\dot{p}_r$ cu semnul corect pentru a indica deplasarea înainte sau înapoi.

Constrângerile diferențiale scrise în termeni de mișcare a lui p_f, unde viteza de deplasare înainte a roții din față v_f este utilizată:

$$\dot{x}_f = v_f \cos(\alpha + \delta), \tag{2}$$

$$\dot{y}_f = v_f \sin(\alpha + \delta), \tag{3}$$

$$\dot{\alpha} = \frac{v_f}{l} \sin(\delta). \tag{4}$$

Viteza roții din față v_f este legată de viteza roții din spate v_r prin:

$$\frac{v_r}{v_f} = \cos(\delta). \tag{5}$$

Problemele de planificare și control pentru acest model implică selectarea unghiului de direcție δ în limitele mecanice ale vehiculului $\delta \in [\delta_{min}, \delta_{max}]$ și viteza de deplasare înainte v_r într-un interval acceptabil $v_r \in [v_{min}, v_{max}]$.

Continuitatea unghiului de direcție poate fi impusă prin augmentarea modelului anterior cu rata de schimbare a direcției:

$$\dot{x}_f = v_f \cos(\alpha + \delta), \tag{6}$$

$$\dot{y}_f = v_f \sin(\alpha + \delta), \tag{7}$$

$$\dot{\alpha} = \frac{v_f}{l} \sin(\delta). \tag{8}$$

$$\dot{\delta} = v_{\delta}. \tag{9}$$

În plus față de limita unghiului de direcție, rata de schimbare a direcției poate fi acum limitată la:

$$v_{\delta} \in \left[\dot{\delta}_{min}, \dot{\delta}_{max}\right]. \tag{10}$$

GridSim a fost utilizat pentru a studia performanța a două abordări de conducere autonomă bazate pe simulare folosind grile de ocupare: învățarea prin întărire profundă și conducerea neuroevoluționară folosind algoritmi genetici. Datele sintetice utilizate ca intrare pentru ambele sisteme de control, agentul DQN și agentul neuroevoluționar, sunt reprezentate de Grilele de Ocupare (OGs) simulate. Algoritmii sunt evaluați în următoarele scenarii: autostradă (cu două modele diferite de senzori), drum curbat, oraș interior și model fără întreruperi, fiecare cu propriul set de constrângeri și dificultate crescândă. Comportamentul dorit al agentului neuroevoluționar propus este codificat într-o funcție de fitness cu două elemente, descriind o distanță maximă parcursă (definită ca distanța rămasă până la obiectivul definit anterior) și o viteză maximă înainte, limitată la un interval specific.

Am construit în GridSim o interfață utilizator potrivită pentru crearea și antrenarea diferitelor topologii de învățare profundă. Interfața suportă, de asemenea, preprocesarea datelor, etichetarea și anotarea. Pentru implementare, am evaluat trei biblioteci AI diferite, și anume Caffe2, Cognitive Neural Toolkit (CNTK) și TensorFlow. Am decis să folosim TensorFlow, deoarece are avantajul de a suporta straturi de rețea fine care permit utilizatorilor să construiască noi tipuri de straturi complexe fără a le implementa într-un limbaj de nivel scăzut. Acest back-end pentru reprezentările rețelelor neuronale este combinat cu API-ul Keras și este scris în Python. GridSim poate fi utilizat atât în configurații CPU, cât și GPU.

Pentru a reduce timpul necesar procesului de antrenament, am folosit următoarea configurație hardware: un computer desktop echipat cu un CPU Intel Core i7 7700K, 64 GB RAM și o placă grafică de înaltă performanță NVIDIA GeForce GTX 1080 Ti. A fost necesară și implementarea unui script pentru salvarea coordonatelor de ieșire ale datelor de intrare generate artificial. Aceste date sunt folosite în modul de redare pentru antrenament.

Interfața utilizator a fost integrată în meniul mediului GridSim, astfel încât modurile pot fi comutate între redare, înregistrare și antrenament, fiecare având acces la cele cinci scenarii diferite. Există un număr mare de parametri configurabili, cum ar fi rezoluția simulatorului, precizia grilei de ocupare, numărul de participanți la trafic, viteza maximă a mașinii ego și raza de întoarcere etc.

Rețelele neuronale profunde (DNN) sunt de obicei antrenate prin algoritmi de învățare bazată pe gradient, cum ar fi backpropagation. Strategiile de antrenament neuroevoluționare pot rivaliza cu algoritmii bazate pe backpropagation, cum ar fi Q-learning și gradientele politicii, în sarcini dificile de învățare prin întărire profundă (DRL). Ideea explorată în această lucrare este de a evolua greutățile unei rețele neuronale profunde folosind un algoritm genetic bazat pe populație, cu reguli de reproducere modificate.

Deoarece răspunsul rețelei profunde este cuantificat folosind o funcție de pierdere multi-obiectiv, greutățile acesteia sunt învățate utilizând calculul evoluționar. Antrenamentul are ca scop calcularea

greutăților optime pentru o colecție de rețele neuronale profunde $\boldsymbol{\varphi}(\cdot;\boldsymbol{\Theta})$ prin optimizarea simultană a funcțiilor lor de fitness. Această procedură de învățare a fost propusă pentru prima dată de autori pentru antrenarea unui clasificator generativ cu o singură învățare.

Abordările tradiționale de anrenare folosesc algoritmi precum backpropagation și o funcție de pierdere scalară pentru a calcula valorile optime ale greutăților unei singure rețele. În cazul antrenamentului evolutiv, $\boldsymbol{\varphi}(\cdot;\boldsymbol{\Theta})$ reprezintă o colecție de K rețele neuronale profunde, fiecare rețea având un set corespunzător de greutăți Θ_i:

$$\boldsymbol{\varphi}(\cdot;\boldsymbol{\Theta}) = \Big[\varphi_1(\cdot;\Theta_1), \varphi_2(\cdot;\Theta_2), ..., \varphi_i(\cdot;\Theta_i), ..., \varphi_K(\cdot;\Theta_K),\Big]^T \tag{11}$$

Greutățile unei singure rețele neuronale profunde sunt stocate într-un așa-numit vector de soluție $\Theta = \Big[\theta_1, \theta_2, ..., \theta_n\Big]^T$, compus din n variabile de decizie θ_i, cu $i = 1, ..., n$ și $\theta \in \mathbb{R}^n$. θ_i reprezintă un parametru de greutate într-o singură rețea.

Algorithm 0.1 Agent neuroevoluționar în GridSim

```
procedure Train(G)
    encoded_w ← encode_w(net_topology)
    ppl ← init_gen(encoded_w)
    while nu s-au terminat toate generațiile do
        while nu s-a terminat generația do
            weights ← decode_weights()
            dist, vel ← fitness_eval(weights)
            gen_score[i] ← dist, vel
        end while
        ppl ← tournament_sel(gen_score, ppl)
        ppl ← uniform_mate(α, ppl)
        ppl ← gaussian_mutate(β, ppl)
    end while
    elite_first ← k_best_selection(1, ppl)
    validate_elite(elite_first)
end procedure
```

Folosind algoritmi genetici, greutățile $\boldsymbol{\Theta}$ ale unei populații de rețele neuronale profunde $\boldsymbol{\varphi}(\cdot;\boldsymbol{\Theta})$, unde un individ Θ este un vector de soluție conținând greutățile unei rețele $\varphi(\cdot;\Theta)$, au fost evoluate. Primul pas în antrenament constă în rularea unui forward pass prin populația de rețele, obținând astfel valori pentru funcția de fitness multi-obiectiv $L(\cdot)$. După completarea forward pass-urilor pentru toată populația de rețele, folosim algoritmul de selecție prin turneu pentru a selecta cei mai buni indivizi.

Algoritmul de selecție prin turneu asigură că un număr de indivizi de elită, cu cea mai bună acuratețe, continuă ne modificați în generația următoare. Pentru explorarea spațiului de decizie S, am folosit un crossover uniform între 2 indivizi dintr-un lot de indivizi, cu o probabilitate independentă α

de a avea loc. În plus, o operație de mutație este aplicată pe același lot, dar cu o probabilitate independentă β. Genele noii generații de indivizi sunt supuse unui zgomot Gaussian aditiv σ:

$$\theta' = \theta + \sigma; \sigma \in [-3, 3] \tag{12}$$

Intervalul $[-3, 3]$ pentru zgomotul Gaussian a fost ales în raport cu funcția sigmoid utilizată pentru activarea neuronilor în rețelele neuronale profunde. Mai exact, intervalul are scopul de a nu suprasatura valorile greutăților prin atingerea valorilor maxime sau minime returnate de o sigmoidă. Noua populație este evaluată și procesul se repetă pentru G generații.

În primul rând, spațiul de decizie al Agentului Neuroevoluționar a fost redus la trei acțiuni (viraj la stânga, viraj la dreapta și nicio acțiune), pornind de la spațiul de decizie original de opt acțiuni (accelerare, viraj la stânga, viraj la dreapta, frânare, accelerare + viraj la stânga, accelerare + viraj la dreapta, mers înapoi/accelerație negativă și nicio acțiune).

Această primă versiune simplificată a mediului menține, de asemenea, viteza mașinii simulate la o rată constantă și folosește un model de drum continuu, care nu are intersecții în T sau intersecții de alt tip. Prin utilizarea mediului menționat, algoritmul genetic selectează cea mai bună combinație de parametri de greutate pentru o rețea neuronală feed forward.

La început, funcția de fitness a fost definită cu o singură metrică, care este distanța parcursă. Rețeaua neuronală efectuează o acțiune în simulator și funcția de fitness este actualizată cu noua valoare. Această distanță este calculată ca distanța euclidiană (măsurată în pixeli) între poziția de pornire și noua poziție, obținută după efectuarea acțiunii prezise. Criteriul de selecție al optimizării genetice s-a bazat pe distanța parcursă, având ca obiectiv maximizarea acesteia:

$$\rho = f(\delta) \tag{13}$$

Pentru a construi următoarea generație de indivizi, a fost utilizată următoarea ecuație:

$$\Theta_{i+1} = max(\rho) + rand(\Theta_i) + Cx(\Theta_i) + \mu(\Theta_i) \tag{14}$$

Intrarea rețelei neuronale este un vector descris de valorile grilei de ocupare generate de fasciculele sintetice ale modelului de senzor radar. Numărul de fascicule ale senzorului este, de asemenea, configurabil și poate fi crescut la orice rezoluție necesară.

După ce comportamentul dorit a fost atins și mașina a fost capabilă să navigheze modelul generat continuu de una singură, am efectuat actualizări incrementale ale spațiului de decizie, până la atingerea a cinci acțiuni (accelerare, viraj la dreapta, viraj la stânga, frânare și nicio acțiune). De asemenea, am eliminat viteza constantă și, pornind de la acest model, am inclus un nou obiectiv de maximizat în funcția de fitness, viteza mașinii:

$$\rho = f(\delta, \theta) \tag{15}$$

Metrica internă utilizată de simulator este ppu (pixeli pe unitate). Aceasta definește câți pixeli intră într-o singură unitate și folosim această valoare pentru a corela viteza simulatorului cu măsurătoarea reală în metri/secundă. Unitatea poate fi definită ca un număr fix de cadre sau un număr

fix de secunde. Noi am definit unitatea noastră ca un singur cadru, cu simulatorul rulând la 30 cadre pe secundă, datorită renderer-ului intern al PyGame.

Am impus un prag de viteză maximă de $30ppu$ și apoi am antrenat modelul să adapteze acțiunea de frânare la mediu, pentru a nu se prăbuși în curbele abrupte ale mediului generat continuu. După 26 de generații, modelul putea naviga fără a se prăbuși, cu o viteză medie de $15ppu$.

După trecerea fiecărui pas incremental de complexitate, spațiul de acțiune a fost mărit până la dimensiunea de opt acțiuni, împreună cu dimensiunea vectorului de intrare a senzorului, ajungând astfel la complexitatea originală a mediului DQN.

Robotul Raspberry PI

Robotul Raspberry Pi este bazat pe șasiul ER-SER85080C. Acesta dispune de un mecanism de direcție pe roțile din față și tracțiune pe roțile din spate, similar cu modelele din competițiile de mașini inteligente. Plăcile de podea sunt realizate din aliaj de aluminiu, ceea ce îmbunătățește considerabil rezistența caroseriei. Este compus din două motoare DC de 1500 RPM pentru fiecare roată din spate, cu o viteză maximă de 4,5 m/s și un raport de decelerare al motorului de 1:10. Direcția se realizează printr-un motor de comutare DS3119, cu un cuplu de direcție de 20 kg. De asemenea, vine cu encodere separate pentru fiecare motor din spate, cu următoarele setări de precizie: roata face o rotație completă, raportul de reducere a vitezei de 30 poate genera 1560 margini de salt, în timp ce raportul de reducere a vitezei de 10 poate genera 520 margini de salt.

Testarea algoritmilor de planificare a traiectoriei pe un robot mobil controlat de un Raspberry Pi implică o serie de experimente atent concepute pentru a evalua capacitatea robotului de a naviga și a îndeplini sarcini eficient. Aceste teste ar trebui să acopere diverse aspecte ale planificării traiectoriei, inclusiv evitarea obstacolelor, acuratețea navigării, procesarea în timp real și adaptabilitatea la medii dinamice. Mai jos sunt prezentate câteva teste cheie care pot fi efectuate pentru a evalua capacitățile de planificare a traiectoriei unui astfel de robot mobil.

Testul de navigare de bază are scopul de a verifica capacitatea robotului de a naviga de la un punct de plecare la un punct de destinație folosind traiectoria planificată. Pentru a efectua acest test, se amenajează un mediu simplu cu un punct de plecare și unul de destinație prestabilite. Algoritmul de planificare a traiectoriei este programat pentru a genera o traiectorie între aceste puncte. La execuție, se observă capacitatea robotului de a urma traiectoria, cu măsurători pentru timpul necesar și acuratețea atingerii destinației. Acest test asigură că robotul poate atinge destinația cu precizie, cu o deviere minimă de la traiectoria planificată și într-un interval de timp rezonabil.

Evitarea obstacolelor este crucială pentru navigarea autonomă. În acest test, obstacole de diverse forme și dimensiuni sunt plasate aleatoriu în mediul robotului. Algoritmul de planificare a traiectoriei trebuie să țină cont de aceste obstacole și să replanifice traiectoria după cum este necesar. Prin executarea algoritmului și observarea comportamentului robotului în timp real, dezvoltatorii pot evalua dacă robotul evită cu succes obstacolele în timp ce ajunge la destinație. Criteriile de evaluare includ capacitatea robotului de a evita coliziunile, de a reruta fluent în jurul obstacolelor și de a menține un timp rezonabil pentru a ajunge la destinație.

Evaluarea capacității robotului de a se adapta la obstacole în mișcare este obiectivul testului de obstacole dinamice. Obstacolele în mișcare, cum ar fi alți roboți mobili sau obiecte controlate manual, sunt introduse în mediu. Algoritmul de planificare a traiectoriei trebuie să includă capacități de replanificare în timp real pentru a se adapta la aceste schimbări. Prin executarea algoritmului și observarea adaptărilor robotului în timp real, testul evaluează cât de eficient evită robotul obstacolele în mișcare și își menține cursul. Factorii cheie includ eficiența manevrelor de evitare, eficiența replanificării în timp real și minimizarea întreruperilor sau întârzierilor în atingerea destinației.

Testul de navigare de precizie evaluează capacitatea robotului de a naviga prin pasaje înguste și medii complexe. Se creează un mediu de testare cu pasaje înguste, colțuri strâmte și căi complexe. Algoritmul de planificare a traiectoriei este programat pentru a naviga prin acest mediu provocator. La execuție, se observă performanța robotului, concentrându-se pe capacitatea sa de a menține traiectoria planificată în spații înguste și complexe. Criteriile de evaluare includ navigarea precisă prin pasaje înguste, coliziuni minime cu limitele și continuitatea și fluiditatea mișcării.

Testul de fiabilitate a senzorilor verifică fiabilitatea și acuratețea senzorilor utilizați pentru planificarea traiectoriei. Robotul este echipat cu diverși senzori, cum ar fi ultrasonici și infraroșii pentru detectarea mediului. Fiecare senzor este testat individual pentru a asigura citiri precise și performanță fiabilă. Acești senzori sunt apoi integrați în algoritmul de planificare a traiectoriei. Prin executarea algoritmului în diferite medii și condiții de iluminare, testul evaluează fiabilitatea senzorilor. Citirile constante și precise ale senzorilor, performanța fiabilă în diverse medii și integrarea și utilizarea corectă a datelor senzorilor în planificarea traiectoriei sunt criteriile cheie de evaluare.

Măsurarea consumului de energie al robotului în timpul planificării și navigării traiectoriei este obiectivul testului de eficiență energetică. Consumul de energie al Raspberry Pi și al motoarelor robotului este monitorizat în timpul funcționării. Se execută o serie de sarcini de planificare a traiectoriei de complexitate variată și se înregistrează consumul de energie pentru fiecare sarcină. Analiza se concentrează pe corelația dintre complexitatea traiectoriei, evitarea obstacolelor și consumul de energie. Criteriile de evaluare includ consumul eficient de energie în timpul navigării, impactul complexității planificării traiectoriei asupra consumului de energie și optimizarea planificării traiectoriei pentru eficiența energetică.

Testul de scenariu real evaluează performanța robotului într-un mediu real cu obstacole și scenarii din viața reală. Se amenajează un mediu de testare care imită un scenariu operațional tipic pentru robot, cum ar fi un mediu de casă sau de birou. Algoritmul de planificare a traiectoriei este programat pentru a naviga prin acest mediu. Prin executarea algoritmului și observarea comportamentului robotului în condiții reale, testul evaluează robustețea și adaptabilitatea robotului. Navigarea și îndeplinirea sarcinilor cu succes într-un mediu real, gestionarea eficientă a obstacolelor neașteptate și adaptabilitatea la schimbări sunt criteriile cheie de evaluare.

Testarea algoritmilor de planificare a traiectoriei pe un robot mobil controlat de un Raspberry Pi implică un set cuprinzător de experimente concepute pentru a evalua diverse aspecte ale capacităților de navigare ale robotului. Aceste teste asigură că algoritmul de planificare a traiectoriei este robust, eficient și fiabil în medii și condiții diverse. Prin realizarea sistematică a acestor teste, dezvoltatorii pot rafina și optimiza algoritmii lor, îmbunătățind în final performanța roboților mobili autonomi.

Agile Scout

Robotul Agile Scout este un sistem robotic mobil avansat, proiectat pentru o varietate de aplicații, inclusiv explorare, supraveghere, căutare și salvare, și monitorizarea mediului. Dezvoltarea sa se concentrează pe combinarea agilității, adaptabilității și robusteței pentru a funcționa eficient în medii diverse și adesea provocatoare. Secțiunile următoare oferă o descriere detaliată a caracteristicilor și capabilităților sale cheie.

În centrul designului robotului Agile Scout se află mobilitatea sa excepțională. Acesta este de obicei echipat cu un set de roți sau șenile foarte versatile, care îi permit să navigheze pe teren accidentat, obstacole și pante abrupte cu ușurință. Unele modele pot avea, de asemenea, picioare articulate sau designuri hibride care combină roțile și picioarele pentru a îmbunătăți manevrabilitatea. Această flexibilitate permite robotului să se miște rapid și eficient atât în medii interioare, cât și exterioare, făcându-l potrivit pentru sarcini care necesită desfășurare rapidă și acoperire extinsă a zonei.

Robotul Agile Scout este echipat cu un set complet de senzori care îi oferă o înțelegere bogată a mediului său. Acești senzori includ adesea LIDAR, senzori ultrasonici, camere infraroșii și camere vizuale de înaltă rezoluție. LIDAR este deosebit de util pentru crearea de hărți 3D detaliate ale împrejurimilor, ceea ce este crucial pentru navigare și evitarea obstacolelor. Combinația acestor senzori permite robotului să detecteze și să recunoască obiecte, să măsoare distanțe și să identifice potențiale pericole în timp real.

Una dintre caracteristicile remarcabile ale robotului Agile Scout este autonomia sa avansată. Comunicarea eficientă este esențială pentru robotul Agile Scout, mai ales în aplicații precum căutarea și salvarea sau supravegherea. Robotul este echipat cu sisteme de comunicații robuste care îi permit să transmită date înapoi la un centru de control în timp real. Aceste date includ fluxuri video, citiri ale senzorilor și actualizări de stare, permițând operatorilor să monitorizeze progresul robotului și să ia decizii informate. Interfața de control este de obicei ușor de utilizat, oferind atât opțiuni de control manual, cât și moduri de operare autonomă. Această capacitate dual asigură că operatorii umani pot interveni atunci când este necesar, beneficiind în același timp de funcțiile autonome ale robotului.

Versatilitatea robotului Agile Scout îl face potrivit pentru o gamă largă de aplicații. În explorare și monitorizarea mediului, poate fi utilizat pentru a colecta date în zone periculoase sau inaccesibile, oferind informații valoroase fără a pune în pericol viețile umane. În misiunile de căutare și salvare, agilitatea și capabilitățile senzoriale îi permit să localizeze și să asiste victimele în zonele afectate de dezastre. Aplicațiile de supraveghere și securitate beneficiază de capacitatea robotului de a patrula zone mari în mod autonom, de a detecta intruziuni și de a transmite informații în timp real personalului de securitate.

Robotul Agile Scout reprezintă un avans semnificativ în robotica mobilă, combinând agilitatea, senzorizarea avansată, autonomia și comunicarea robustă pentru a îndeplini o varietate de sarcini critice. Capacitatea sa de a naviga în medii complexe, împreună cu capacitățile sale autonome, îl fac un instrument valoros în domenii variate, de la explorare și monitorizarea mediului până la căutare și salvare și supraveghere. Pe măsură ce tehnologia continuă să evolueze, robotul Agile Scout este pregătit să devină și mai capabil, extinzându-și și mai mult gama de aplicații și eficacitatea în scenarii

critice.

Agile Scout este un SSWMR (robot mobil cu roți cu direcție prin patinare). Sunt luate în considerare următoarele ipoteze ale modelului:

1. Centrul de masă al robotului este la centrul geometric al cadrului corpului;
2. Ambele roți de pe fiecare parte se rotesc la aceeași viteză;
3. Robotul operează pe o suprafață fermă, cu toate cele patru roți în contact permanent cu aceasta.

Definim un cadru inerțial (X, Y) (cadru global) și un cadru local (corp de robot) (x, y). Presupunem că robotul se mișcă pe o suprafață plană cu o viteză liniară $v = (v_x, v_y, 0)^T$ și se rotește cu o viteză unghiulară $\omega = (0, 0, \omega_z)^T$, ambele exprimate în cadrul local. Dacă $q = (X, Y, \theta)^T$ este vectorul de stare care definește coordonatele generalizate ale robotului (pozițiile X și Y, precum și orientarea θ a cadrului local în raport cu cadrul inerțial), atunci $\dot{q} = (\dot{X}, \dot{Y}, \dot{\theta})^T$ este vectorul vitezelor generalizate.

Relația dintre vitezele robotului în ambele cadre se calculează după cum urmează:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}. \tag{16}$$

Deoarece specifică doar cinematica corpului liber, Ecuația (16) nu impune limite asupra mișcării planului SSWMR. Ca urmare, trebuie analizată relația dintre vitezele roților și vitezele locale. Pentru simplitate, grosimea roții este neglijată și se presupune că este în contact cu planul în punctul P_i, conform ipotezei inițiale nr. 3. În comparație cu alte vehicule pe roți, SSWMR are o viteză laterală nenulă. Această proprietate provine din structura mecanică a SSWMR, care necesită patinaj lateral dacă orientarea vehiculului se schimbă. Ca urmare, roțile sunt tangente la traseu doar când $\omega = 0$, adică atunci când robotul se deplasează în linie dreaptă. Este important să se ia în considerare toate roțile împreună la dezvoltarea modelului cinematic.

Fie ωi și vi, cu $i = 1, 2, 3, 4$ denotând vitezele unghiulare ale roților și vitezele liniare ale centrului pentru roțile din față-stânga, spate-stânga, față-dreapta și spate-dreapta, respectiv. Astfel, avem:

$$\omega_L = \omega_1 = \omega_2, \qquad \omega_R = \omega_3 = \omega_4. \tag{17}$$

Putem folosi ecuația anterioară pentru a stabili cinematica directă pe plan:

$$\begin{bmatrix} v_x \; v_y \; \omega_z \end{bmatrix} = f \begin{bmatrix} \omega_l r \; \omega_r r \end{bmatrix}, \tag{18}$$

unde $v = (v_x, v_y)$ este viteza de translație a vehiculului în raport cu cadrul său local, ω_z este viteza sa unghiulară, iar r este raza roții.

Centrele instantanee de rotație ale părții stânga, părții dreapta și corpului robotului sunt denumite ICR_l, ICR_r și ICR_G, respectiv, în timp ce robotul mobil se deplasează. ICR_l, ICR_r și ICR_G sunt toate cunoscute ca fiind pe o linie paralelă cu axa x. Definim coordonatele x-y pentru ICR_l, ICR_r și

ICR_G ca (x_{ICR_l}, y_{ICR_l}), (x_{ICR_r}, y_{ICR_r}) și (x_{ICR}, y_{ICR}), respectiv. Viteza unghiulară a părților este egală cu viteza corpului robotului ω_z. Obținem în continuare următoarele relații geometrice:

$$x_{ICR} = x_{ICR_l} = x_{ICR_r} = -\frac{v_y}{\omega_z} \tag{19}$$

$$y_{ICR} = \frac{v_x}{\omega_z}, \tag{20}$$

$$y_{ICR_l} = \frac{v_x - \omega_l r}{\omega_z}, \tag{21}$$

$$y_{ICR_r} = \frac{v_x - \omega_r r}{\omega_z}. \tag{22}$$

Din ecuațiile (19)–(22), relația cinematică (18) poate fi reprezentată ca:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = J_\omega \begin{bmatrix} \omega_l r \\ \omega_r r \end{bmatrix}, \tag{23}$$

unde elementele matricei J_ω sunt determinate de coordonatele ICR pe partea stângă și dreaptă:

$$J_\omega = \frac{1}{y_{ICR_l} - y_{ICR_r}} \begin{bmatrix} -y_{ICR_r} & y_{ICR_l} \\ x_{ICR} & -x_{ICR} \\ -1 & 1 \end{bmatrix}. \tag{24}$$

Deoarece SSWMR este simetric în cazul nostru, putem obține un model cinematic simetric. Ca rezultat, ICR-urile sunt distribuite simetric pe axa x, iar matricea J_ω poate fi scrisă după cum urmează:

$$J_\omega = \frac{1}{2y_{ICR_0}} \begin{bmatrix} y_{ICR_0} & y_{ICR_0} \\ 0 & 0 \\ -1 & 1 \end{bmatrix}, \tag{25}$$

unde $y_{ICR_0} = y_{ICR_l} = -y_{ICR_r}$ reprezintă valorile ICR-urilor laterale. Având în vedere că, pentru modelul nostru simetric, $v_l = \omega_l r$ și $v_r = \omega_r r$, relațiile dintre vitezele unghiulare ale roților și vitezele robotului sunt următoarele:

$$\begin{cases} v_x = \frac{\omega_l r + \omega_r r}{2} = \frac{v_l + v_r}{2} \\ v_y = 0 \\ \omega_z = \frac{-\omega_l r + \omega_r r}{2y_{ICR_0}} = \frac{-v_l + v_r}{2y_{ICR_0}} \end{cases}. \tag{26}$$

Pe baza ecuației (26), semnalul de control u poate fi scris ca:

$$u = \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} = r \begin{bmatrix} \frac{\omega_l + \omega_r}{2} \\ \frac{-\omega_l + \omega_r}{2y_{ICR_0}} \end{bmatrix}. \tag{27}$$

Ultima ecuație arată că perechea de viteze unghiulare ω_l și ω_r, precum și vitezele v_x și ω_z, pot fi tehnic privite ca un semnal de intrare cinematic de control. Precizia relației (27), pe de altă parte, este puternic dependentă de derapajul longitudinal și poate fi utilizată doar dacă acest fenomen nu

este dominant. În plus, parametrii r și y_{ICR_0} pot fi calculați experimental pentru a asigura că viteza unghiulară a robotului este estimată cu exactitate în raport cu vitezele unghiulare ale roților.

Planificarea traseului în 2D

Capacitatea unei mașini autonome de a găsi o rută între două puncte, adică o poziție de început și o locație dorită, reprezintă planificarea traiectoriei. Conform procesului de planificare a traiectoriei, o mașină autonomă ar trebui să ia în considerare toate posibilele obstacole prezente în mediul înconjurător și să calculeze o traiectorie de-a lungul unei rute fără coliziuni. După cum s-a menționat anterior, conducerea autonomă este un cadru multi-agent în care vehiculul gazdă trebuie să aplice abilități sofisticate de negociere cu alți utilizatori ai drumului atunci când depășește, cedează, se încadrează, face viraje la stânga și la dreapta, toate acestea în timp ce navighează pe drumuri urbane nestructurate. Descoperirile din literatură indică o politică netrivială care ar trebui să gestioneze siguranța în conducere. Având în vedere o funcție de recompensă $R(\bar{s}) = -r$ pentru un eveniment de accident care trebuie evitat și $R(\bar{s}) \in [-1, 1]$ pentru restul traiectoriilor, scopul este de a învăța să execute manevre dificile într-un mod lin și sigur.

Tradițional, DNN-urile sunt antrenate folosind metode diferențiabile bazate pe funcții de cost cu un singur obiectiv. Deși mai multe pierderi pot fi agregate într-o singură funcție de cost prin ponderare, pasul de gradient descent în algoritmul de backpropagation va ajusta greutățile rețelei doar pe baza pierderii cu un singur obiectiv. Ponderarea mai multor pierderi introduce, de asemenea, hiperparametri suplimentari (de obicei definiți manual) necesari pentru a pondera fiecare pierdere individuală.

O altă abordare a optimizării multi-obiectiv în învățarea profundă este Învățarea Multi-Sarcină (MTL), unde obiectivele sunt date ca sarcini. Modelul de rețea își împarte fie straturile între sarcinile ponderate (împărțirea dură a parametrilor), fie fiecare sarcină este utilizată pentru a antrena un model separat (împărțirea moale a parametrilor). În cel de-al doilea caz, parametrii între modelele corespunzătoare sarcinilor date sunt regularizați pentru a încuraja parametrii să fie similari. După cum s-a menționat anterior, paradigma de împărțire dură a parametrilor este încă pervasivă pentru MTL bazată pe rețele neuronale. Spre deosebire de MTL, noi folosim o tehnică de optimizare multi-obiectiv Pareto care optimizează independent o populație de DNN-uri, unde antrenarea fiecărui individ nu este influențată de ceilalți indivizi. În acest fel, asigurăm o explorare mai bună a spațiului de parametri în timpul antrenamentului, evitând în același timp hiperparametrii suplimentari de ponderare.

Definiția problemei

Dată fiind o secvență de grile de ocupare 2D (OG) $\vec{X} : \mathbb{R}^2 \times \tau_i \rightarrow \mathbb{R}^2 \times \tau_o$, poziția vehiculului ego $\vec{p}ego^{<t>} \in \mathbb{R}^2$ în $\vec{x}^{<t>}$ și coordonatele destinației $\vec{p}dest^{<t>} \in \mathbb{R}^2$ în spațiul grilei de ocupare la timpul t, sarcina este să învățăm o traiectorie locală pentru a naviga vehiculul ego către coordonatele destinației $\vec{p}_{dest}^{<t+\tau_o>}$. τ_i este lungimea secvenței de intrare OG, în timp ce τ_o este numărul de pași de timp pentru care este estimată traiectoria vehiculului ego.

Cu alte cuvinte, având $\vec{p}_0^{<t>}$ ca o coordonată în observația OG curentă $\vec{x}^{<t>}$, căutăm o traiectorie de navigație locală dorită a vehiculului ego de la orice punct de pornire arbitrar $\vec{p}0^{<t>}$ la $\vec{p}dest^{<t+\tau_o>}$, cu următoarele proprietăți:

- calea parcursă $||\vec{p}0^{<t>} - \vec{p}dest^{<t+\tau_o>}||$ este minimă;
- viteza laterală, dată de rata de schimbare a unghiului de direcție $v_\delta \in \left[\dot{\delta}min, \dot{\delta}max\right]$ este minimă, semnificând o valoare minimă pentru $v_\delta^{<t,t+\tau_o>}$;
- viteza înainte, cunoscută și ca viteză longitudinală, $v^{<t,t+\tau_o>}$ este maximă și limitată la un interval acceptabil $[v_{min}, v_{max}]$.

Vehiculul este modelat pe baza modelului cinematic cu o singură pistă a unui robot, cu starea poziției $\vec{y}^{<t>} = (p_x^{<t>}, p_y^{<t>})$ și presupunerea de fără derapaj. p_x și p_y reprezintă poziția vehiculului în planul de conducere 2D, respectiv. Direcția nu este luată în considerare pentru estimarea traiectoriei.

Observăm mediul de conducere folosind OG-uri construite din datele combinate LiDAR și radar. Pixeli verzi și roșii reprezintă spațiul liber și obstacolele, respectiv, în timp ce negru semnifică ocuparea necunoscută. O singură OG corespunde unei instanțe de observație $\vec{x}^{<t>}$, în timp ce o secvență de OG-uri este denumită $\vec{X}^{<t-\tau_i,t>}$. Aceste observații sunt secvențe de grile discrete aliniate pe axe, achiziționate pe intervalul de timp $[t - \tau_i, t]$ și centrate pe secvența stărilor vehiculului $\vec{Y}^{<t-\tau,t>}$.

Arbitrarea comportamentului

Vehiculele Autonome (AV) sunt sisteme robotice care se pot ghida singure fără operatori umani. Astfel de vehicule sunt echipate cu componente de Inteligență Artificială (AI) și se așteaptă să schimbe dramatic viitorul mobilității, aducând o varietate de beneficii în viața de zi cu zi, cum ar fi facilitarea conducerii, îmbunătățirea capacității rețelelor rutiere și reducerea accidentelor legate de vehicule.

Cel mai probabil, din cauza lipsei garanțiilor de siguranță și a legislației, precum și a lipsei de scalabilitate a sistemelor de Conducere Autonomă (AD), vehiculele complet autonome nu vor circula pe străzi în viitorul apropiat. Cu toate acestea, în ultimii ani, progresul realizat în domeniul AI, precum și disponibilitatea comercială a Sistemelor Avansate de Asistență pentru Șoferi (ADAS), ne-au adus mai aproape de obiectivul autonomiei complete în conducere.

Societatea Inginerilor de Automobile (SAE) a definit standardul J3016 pentru diferite niveluri de autonomie, care împarte conceptul de conducere autonomă în 5 niveluri de automatizare. Nivelurile 1 și 2 sunt reprezentate de sisteme în care șoferul uman trebuie să monitorizeze scena conducerii,

în timp ce nivelurile 3, 4 și 5, cu nivelul 5 fiind complet autonom, consideră că componentele de conducere automată monitorizează mediul.

Un vehicul autonom trebuie să fie capabil să își perceapă împrejurimile și să formeze un model de mediu care să includă obiecte în mișcare și staționare. Ulterior, folosește aceste informații pentru a învăța strategii de conducere pe termen lung. Aceste politici de conducere guvernează mișcarea vehiculului și generează automat semnale de control pentru volan, accelerație și frână. La cel mai înalt nivel, sistemul de luare a deciziilor al vehiculului trebuie să selecteze o rută optimă de la poziția curentă la destinație.

Principalul motiv din spatele capacității umane de a conduce mașini este abilitatea noastră de a înțelege mediul de conducere, sau contextul conducerii. În cele ce urmează, ne vom referi la contextul conducerii ca la scenă și îl vom defini ca modele legate de obiecte. În această lucrare, introducem AIBA (AI Behavior Arbitration), un algoritm conceput pentru a arbitra între diferite strategii de conducere sau comportamente ale vehiculului, pe baza înțelegerii AIBA a relațiilor dintre obiectele din scenă.

O scenă de conducere constă din obiecte precum benzi de circulație, trotuare, mașini, pietoni, biciclete, semne de circulație etc., toate fiind conectate între ele într-un mod particular (de exemplu, un semn de circulație afișează informații pentru un șofer). Predicția comportamentului de conducere este o parte esențială a procesului de luare a deciziilor al unui vehicul autonom. Sarcina de a identifica comportamentul viitor al obiectelor din scenă nu este una trivială, deoarece acest tip de informație nu poate fi măsurat sau comunicat direct, fiind considerată informație latentă. Pentru a putea efectua predicția comportamentului vehiculelor din jur, un vehicul autonom poate folosi modele matematice care iau în considerare variația în mișcarea obiectelor și descriu scenariul de conducere din punctul de vedere al mașinii ego.

Aceste tipuri de modele folosesc mai multe tipuri de informații, adică cinematică vehiculului, relația dintre mașina ego și entitățile din jur, interacțiunile cu alte vehicule și cunoștințele a-priori. Cinematica vehiculului și relațiile cu entitățile rutiere au fost considerate de aproape toate studiile existente.

De obicei, cele mai multe modele pentru arbitrajul comportamentului sunt adaptate pentru un anumit scenariu specific. Cu toate acestea, vehiculele autonome trebuie să conducă prin medii care se schimbă dinamic, în care o diversitate de scenarii apar de-a lungul timpului. Mai multe modele specifice scenariului activează un model corespunzător în funcție de caracteristicile scenariului. AIBA este un sistem pentru arbitrajul comportamentului în vehicule autonome, care construiește un model de descriere și înțelegere a scenei de conducere. Ideea noastră este să modelăm procesul de înțelegere al șoferului uman (HDr) al scenei de conducere, pentru a obține o soluție optimă de arbitraj al comportamentului. Descriem mecanismul gândirii HDr și îl transpunem într-un model aproximativ.

Descrierea scenei de conducere este dată din perspectiva șoferului uman și formulează proprietăți derivate din definițiile claselor, subclaselor și obiectelor care reprezintă nucleul unui model de abstractizare, bazat pe lucrările anterioare ale autorilor. Ideea principală din spatele AIBA este de a modela, sau de a formaliza, procesul de înțelegere al HDr și apoi de a-l transforma într-un model formal pentru arbitrajul comportamentului în vehicule autonome.

Un șofer uman este capabil să perceapă obiectele din scenă și să le observe. Aceasta înseamnă

că HDr identifică conceptele și diferitele proprietăți ale obiectelor. Diferitele obiecte din scenă sunt legate între ele, iar definirea legăturilor este un prim pas în procesul de cunoaștere, ceea ce înseamnă că stabilește subiectele de interes și, de asemenea, nivelul de importanță al fiecărui subiect. Sinteza înțelegerii scenei de către HDr conține următorii pași: identificarea legăturilor între obiecte, alocarea modelelor fiecărei legături, rularea modelelor și, ulterior, găsirea unei strategii de acțiune. De fapt, HDr creează un sistem implicit și îl simulează.

Pot fi observate două tipuri de legături: legături interne, considerate a fi setul de legături între observator (HDr) și toate obiectele observate, și legături externe, reprezentate de setul de legături între obiectele prezente în scenă. Din punctul de vedere al HDr, legăturile au semnificații și importanțe diferite. Specific, un șofer uman cunoaște regulile de circulație și cum să ajungă la destinația sa. Aceste reguli vor determina prioritățile de trafic, făcând astfel ca HDr să acorde o importanță mai mare acelor legături care sunt mai importante pentru strategia sa.

În pasul următor, se stabilește o descriere pentru fiecare legătură. În timpul conducerii, HDr adaptează, sau rafinează, descrierea menționată prin observație. Șoferul, folosind importanța legăturilor, simulează o descriere a scenei. Dacă analizăm scopul înțelegerii scenei de conducere, vom observa că originile sale sunt stabilitatea în timp și spațiu. Mai precis, șoferul uman are o sarcină de conducere care poate fi îndeplinită dacă posibilitatea de locomoție este păstrată în poziția curentă și pe durata unui anumit timp. Intuitiv, stabilitatea este legată de obiectele din scenă și poate fi atinsă prin înțelegerea scenei. Această înțelegere nu rezolvă problema conducerii, dar oferă informațiile pe baza cărora șoferul uman decide să acționeze un anumit comportament.

Analiza anterioară a înțelegerii scenei de conducere poate fi descrisă ca un proces care constă din următorii pași:

1. Perceperea obiectelor (mașini, semne de circulație, benzi de circulație, pietoni etc.) și obținerea proprietăților obiectelor (intenția pietonului este de a traversa drumul).

2. Definirea legăturilor (rețeaua scenei) între obiecte și a importanței acestora (de exemplu, mașina din față este importantă, pietonul care intenționează să traverseze drumul este foarte important etc.).

3. Adăugarea modelelor la legăturile menționate.

4. Simularea modelului celor mai semnificative legături și propunerea comportamentelor de conducere care vor fi verificate cu celelalte legături semnificative până când se găsește comportamentul adecvat (care permite îndeplinirea sarcinii de conducere).

Entități precum obiectele, legăturile sau rețelele, care au fost introduse în secțiunea anterioară, au corespondențe în procesul de modelare. Intenția noastră este să aproximăm procesul de înțelegere al șoferului uman (HD) printr-o reprezentare formală. Mai precis, această sarcină imită un proces de intrare/ieșire: folosind o scenă percepută, modelul AIBA trebuie să producă o descriere care oferă toate informațiile necesare în cadrul sistemului AD pentru a arbitra comportamentul de conducere.

În cadrul AIBA, prima acțiune este transformarea scenei într-o colecție de obiecte. Această operație este realizată de următoarea funcție generativă:

$$obj\_gen : \Sigma \times K \rightarrow \Omega \tag{28}$$

unde Σ este scena percepută; K este setul de clase de obiecte cunoscute și Ω este setul de obiecte.

Având o colecție inițială de clase, funcția generator din Eq. 45 transformă entitățile scenei într-o colecție de obiecte. Următoarele clase de obiecte sunt luate în considerare: participanți la trafic, pietoni și clădiri. Setul de clase este stabilit a priori în cadrul AIBA.

Complexitatea acestui proces, chiar și pentru setul de clase de drumuri, este evidentă în numeroasele tipuri de drumuri care există în întreaga lume. Pentru a reduce complexitatea scenei, le împărțim în două clase majore: obiecte statice (benzi de circulație, semne de circulație, clădiri etc.) și obiecte dinamice (mașini, pietoni etc.).

Două tipuri de definiții sunt semnificative aici: definiția generică, unde sunt menționate proximitatea și proprietățile, și definiția extensivă, unde este indicată sau prezentată definiția obiectului (sau o imagine a acestuia). Această observație ne permite să asociem metodele de recunoaștere a imaginilor (care corespund definiției extensive) cu o colecție de definiții generice.

Eq. 45 poate fi generalizată atunci când măsurătorile (viteza mașinilor, distanța dintre mașini, dimensiunea semnelor de circulație etc.) sunt asociate cu descrierea scenei:

$$obj\_gen : \Sigma \times K \times M \rightarrow \Omega, \tag{29}$$

unde M este setul de măsurători. Obiectele generate Ω conțin o structură de proprietăți și metode specifice clasei. Metodele reflectă interacțiunile posibile ale obiectelor cu vehiculul ego.

Al doilea pas în fluxul de lucru AIBA definește legăturile dintre un HDr și un obiect, precum și între obiectele însele, respectiv:

$$link\_gen : \Omega \times T \rightarrow \Lambda, \tag{30}$$

unde T este setul de proprietăți ale traiectoriei sarcinii și Λ este setul de semnificații ale legăturilor.

Deoarece legăturile sunt calculate în termeni de stabilitate în jurul unui punct particular al traiectoriei sarcinii, semnificația este corelată cu amenințări specifice. Conducerea unei mașini este supusă unei negocieri implicite bazate pe regulile de circulație. Cele mai importante legături sunt cele legate de agenții care, conform acestor reguli, au prioritate. Eq. 30 poate fi imaginată ca un sistem expert care va analiza toate aceste legături din punctul de vedere menționat, generând diferite marcaje:

$$obj\_gen : \Sigma \times K \times M \times T \rightarrow \Omega \times \Lambda. \tag{31}$$

Fiecare obiect recunoscut oferă metode care se referă la interacțiunile dintre vehiculul ego și obiect. Informațiile despre posibilele amenințări la stabilitatea sarcinii pot fi obținute prin simularea comportamentelor și, de asemenea, despre cum să selectăm comportamentul de conducere adecvat pentru a evita aceste amenințări:

$$thr\_sim : \Omega_S \times H \rightarrow \Theta \times B, \tag{32}$$

$$\Omega_S = \left\{ O_i | O_i \in \Omega; S_{O_i,O_j} \geq S_{min} \right\} \tag{33}$$

$$H = \begin{bmatrix} t_c & t_c + \delta \end{bmatrix} \tag{34}$$

unde Ω_S este setul de obiecte importante O_i, care sunt legate de alte obiecte O_j cu o semnificație $S_{O_i,O_j} \in \Lambda$, mai mare decât o semnificație minimă (impusă a priori) S_{min}. H este orizontul de timp, t_c este timpul curent, δ este timpul de simulare, Θ este setul de niveluri de amenințare și B este setul de comportamente recomandate pentru vehicul.

thr_sim va simula, pentru fiecare obiect important O_i, o colecție de comportamente $O_i\_M_{i,k}(P,H)$ și un nivel de amenințare prezis $\Theta_{i,k}$:

$$O_i\_M_{i,k}(P,H) = \begin{bmatrix} \Theta_{i,k} \\ \beta_{i,k} \end{bmatrix} \tag{35}$$

unde P este setul de proprietăți ale obiectului și $\beta_{i,k}$ este comportamentul vehiculului ego care va elimina amenințarea.

Pentru a rezolva toate amenințările legăturilor, pot fi alese mai multe strategii. Dacă adoptăm descrierea înțelegerii HDr din secțiunea precedentă, comportamentul care rezolvă amenințarea maximă este simulat pentru celelalte amenințări, obținând comportamentul optim al vehiculului ego $\Theta_{max} = \Theta_{i^*,k^*}$:

$$(i^*, k^*) = \arg\max_{i,k} \Theta_{i,k} \tag{36}$$

unde $\Theta_{i,k}$ sunt nivelurile de amenințare.

Ultimul pas în sistemul de modelare AIBA este transformarea comportamentului optim Θ_{max} într-o explicație în limbaj natural:

$$dsc : \Lambda \times B \rightarrow \Delta \tag{37}$$

$$\Delta = \Omega_S \times E \tag{38}$$

$$E = e_1 \times e_2 \times ... \times e_{n_S} \tag{39}$$

unde Δ este setul de descrieri, e_i este explicația semnificației asociate unui obiect și n_S este numărul de obiecte semnificative.

Rețeaua neuronală NeuroTrajectory

Abordarea acestei teze pentru învățarea traiectoriei locale de stare este de a reformula problema conducerii autonome ca o sarcină de învățare cognitivă. Problema de mai sus poate fi modelată ca un Proces de Decizie Markovian (MDP) $M = (S, A, T, L)$, unde:

- S reprezintă un set finit de stări, $\vec{s}^{<t>} \in S$ fiind starea agentului la timpul t. Pentru a codifica locația agentului în spațiul OG de conducere la timpul t, definim $\vec{s}^{<t>} = \vec{X}(\vec{p}ego^{<t-\tau_i,t>})$, care denotă o secvență de grilă discretă aliniată pe axe în intervalul $[t-\tau_i, t]$, centrată pe pozițiile vehiculului ego $\vec{p}ego^{<t-\tau_i,t>}$.

- A agentul ar trebui să o urmeze în intervalul de timp viitor $[t+1, t+\tau_o]$. O traiectorie $\vec{Y}^{<t+1,t+\tau_o>}$ este definită ca o colecție de puncte de referință ale stării traiectoriei estimate:

$$\vec{Y}^{<t+1,t+\tau_o>} = [\vec{y}^{<t+1>}, \vec{y}^{<t+2>}, ..., \vec{y}^{<t+\tau_o>}]. \tag{40}$$

- $T : S \times A \times S \to [0, 1]$ este o funcție de tranziție stocastică, unde $T^{\vec{s}^{<t+\tau_o>}}_{s^{<t>}, \vec{Y}^{<t+1,t+\tau_o>}}$ descrie probabilitatea de a ajunge în starea $\vec{s}^{<t+\tau_o>}$, după efectuarea unei mișcări de-a lungul traiectoriei $\vec{Y}^{<t+1,t+\tau_o>}$.

- $\vec{L} : S \times A \times S \to \mathbb{R}^3$ este o funcție vectorială de fitness multi-obiectiv care cuantifică calitatea traiectoriei vehiculului:

$$\vec{L}^{\vec{s}^{<t+\tau_o>}}_{\vec{s}^{<t>}, \vec{Y}^{<t+1,t+\tau_o>}} = \begin{bmatrix} l_1^{<t+\tau_o>} & l_2^{<t+\tau_o>} & l_3^{<t+\tau_o>} \end{bmatrix}. \tag{41}$$

  Fiecare element din Eq. 41 este definit astfel:

$$l_1^{<t+\tau_o>} = \sum_{i=1}^{\tau_o} ||\vec{p}_{ego}^{<t+i>} - \vec{p}_{dest}^{<t+i>}||_2^2 \tag{42}$$

$$l_2^{<t+\tau_o>} = \sum_{i=1}^{\tau_o} v_\delta^{<t+i>} \tag{43}$$

$$l_3^{<t+\tau_o>} = \sum_{i=1}^{\tau_o} v_f^{<t+i>} \in [v_{min}, v_{max}] \tag{44}$$

Intuitiv, $l_1^{<t+\tau_o>}$ reprezintă un feedback bazat pe distanță, care este mai mic dacă mașina urmează o traiectorie cu energie minimă către $\vec{p}_{dest}^{<t+\tau_o>}$ și mare în caz contrar. $l_2^{<t+\tau_o>}$ cuantifică mișcările periculoase și disconfortul pasagerilor prin însumarea vitezei laterale a vehiculului. Funcția de feedback $l_3^{<t+\tau_o>}$ este viteza longitudinală a vehiculului ego, limitată la viteze adecvate pentru diferite sectoare de drum, cum ar fi $v^{<t,t+\tau_o>} \in [80kmh, 130kmh]$ în cazul conducerii pe autostradă.

Având în vedere schema de estimare a stării propusă, obiectivul este să antrenăm un aproximator optim, definit aici printr-o rețea neurală profundă, care poate prezice traiectoria optimă a stării $\vec{Y}^{<t+1,t+\tau_o>}$ a vehiculului ego, dată fiind o secvență de observații ale grilei de ocupare $\vec{X}^{<t-\tau_i,t>}$ și vectorul de fitness multi-obiectiv din Eq. 41.

Învățăm o traiectorie optimă a stării combinând Rețele Neurale Convoluționale (CNN) cu predicțiile temporale robuste ale rețelelor Long Short-Term Memory (LSTM). Cele două tipuri de arhitecturi neurale sunt combinate după cum urmează. O observație $\vec{x}^{<t>}$ este procesată inițial de un CNN, implementat ca o serie de straturi convoluționale, având scopul de a extrage caracteristici spațiale relevante din datele de intrare. CNN-ul produce o reprezentare în spațiul caracteristicilor pentru fiecare

observație din $\vec{X}^{<t-\tau_i,t>}$. Fiecare observație spațială procesată în intervalul de intrare $[t-\tau_i, t]$ este aplatizată și trecută prin două straturi complet conectate de 1024 și 512 unități, respectiv. Secvența de intrare într-un bloc LSTM este reprezentată de o secvență de observații procesate spațial, denumită $CNN^{<t-\tau_i,t>}$. Aceeași topologie de rețea poate fi antrenată separat pe date sintetice, precum și pe date din lumea reală. Ca parametri de rețea antrenabili, considerăm atât greutățile rețelelor LSTM, cât și greutățile straturilor convoluționale.

Pentru a calcula traiectoria stării vehiculului ego, am proiectat rețeaua neurală profundă, unde secvențele OG sunt procesate de un set de straturi convoluționale, înainte de a fi introduse în diferite ramuri ale rețelei LSTM. Fiecare ramură LSTM este responsabilă pentru estimarea punctelor de referință ale traiectoriei pe intervalul de timp $[t+1, t+\tau_o]$. Alegerea pentru un teanc de ramuri LSTM în locul unei singure rețele LSTM care ar prezice toate punctele de referință ale stării viitoare vine din experimentele noastre cu diferite arhitecturi de rețea. Mai exact, am observat că performanța unei singure rețele LSTM scade exponențial odată cu orizontul de predicție τ_o. Valoarea maximă pentru care am putut obține o traiectorie stabilă folosind o singură rețea LSTM a fost $\tau_o = 2$. După cum se arată în secțiunea de rezultate experimentale, acest lucru nu este cazul cu teancul nostru propus de LSTM-uri, unde fiecare ramură este responsabilă pentru estimarea unui singur punct de referință al stării.

Pentru a antrena rețeaua profundă pe cât mai multe cazuri limită posibil, am construit un set de date de antrenament bazat pe mostre reale de grilă de ocupare $\vec{X}^{<t-\tau_i,t>}$, precum și pe secvențe sintetice $\hat{\vec{X}}^{<t-\tau_i,t>}$. Datele sintetice sunt generate în simulatorul nostru OG GridSim. Ca parametri de rețea antrenabili, considerăm atât greutățile rețelei LSTM, cât și greutățile straturilor convoluționale. Fluxurile de date sintetice și din lumea reală sunt procesate de o rețea convoluțională, înainte de a fi introduse în rețelele LSTM prin două straturi complet conectate de 1024 și 512 unități, respectiv. Aceeași topologie de rețea poate fi antrenată separat pe date sintetice sau din lumea reală.

OctoPath - extinderea la reprezentarea 3D

Deoarece planificarea mișcării poate fi privită și ca o problemă de mapare secvență-la-secvență sau ca o sarcină de generare de secvențe, RNN-urile au fost propuse pentru modelarea traiectoriilor de conducere. Diferit de rețelele neuronale convenționale, RNN-urile conțin o buclă de feedback dependentă de timp în celula de memorie. Pentru a utiliza RNN-urile în predicția unei traiectorii viitoare, fiecare punct separat este considerat un stat, ceea ce implică în continuare că întreaga traiectorie este reprezentată ca o secvență. Tranziția de la un stat la altul este strict constrânsă de topologia rețelei.

Cele mai multe soluții RNN propuse pentru rezolvarea sarcinii de estimare a traiectoriei necesită un model de mediu discret. În această teză, modelul de mediu propus se bazează pe octree-

uri și utilizează estimarea probabilistică a ocupării. Principalele avantaje ale utilizării acestui model sunt că reprezintă explicit nu doar spațiul ocupat, ci și zonele libere și necunoscute și că permite o reprezentare compactă a memoriei și rezoluții configurabile. Spre deosebire de Neuro-trajectory, care a fost prezentat în secțiunea anterioară, lumea este acum percepută în 3D folosind o reprezentare octree și nu mai folosim straturi convoluționale pentru procesarea secvențelor de intrare, deoarece această reprezentare intermediară a fost preluată de vectorul de stare fix între encoder și decoder al arhitecturii noastre.

În această teză, este abordată componenta de planificare a traiectoriei din pipeline-ul percepție-planificare-acțiune. Metoda propusă în prezent, denumită OctoPath, este auto-supervizată și are ca scop combinarea rezoluției configurabile a unui model de mediu bazat pe octree cu o arhitectură RNN encoder-decoder bazată pe clasificare. Aceasta ia ca intrare o secvență de măsurători de senzori, împreună cu segmentul curent al unei traiectorii de referință, bazându-se pe arhitectura RNN encoder-decoder care a arătat performanțe excelente pentru sarcinile de tip secvență-la-secvență.

Arbori de tip OcTrees

Cele mai multe aplicații robotice necesită un model de mediu care include zone libere, ocupate și nemapate și care este eficient din punct de vedere al timpului de execuție și al utilizării memoriei. Greșelile de măsurare a distanței sunt comune în modelele de senzori, iar reflecțiile sau barierele dinamice pot genera rezultate aparent aleatorii. Când se construiește un model precis al mediului din date zgomotoase, trebuie luată în considerare incertitudinea subiacentă. Multiple citiri eronate pot fi apoi combinate pentru a produce o estimare credibilă a condiției reale a mediului.

Structura de date octree reprezintă obiecte tridimensionale. Un octree este o decompoziție spațială în care rădăcina arborelui este divizată recursiv în două pe fiecare direcție coordonată până când fiecare celulă poate conține un număr maxim de elemente. Cu alte cuvinte, este o structură de date ierarhică pentru subdiviziunea spațială 3D care este cel mai frecvent utilizată pentru a subdivide recursiv o regiune 3D dată în opt octanți. Octree organizează elementele sale ierarhic, evitând reprezentarea spațiului gol.

$$O_k : [0,1]^d \rightarrow D_k \subset \mathbb{R}^3 \tag{45}$$

Nodul sau celula rădăcină este nodul inițial al unui octree. Nodul sau celula rădăcină indică opt elemente sau celule, fiecare dintre acestea putând indica alte opt elemente sau celule și așa mai departe. Fiecare nod dintr-un octree reprezintă spațiul unui volum cubic cunoscut sub numele de voxel, iar dimensiunea minimă a voxelului determină rezoluția octree-ului. Ultimul nivel atins este cunoscut sub numele de nivelul frunzei și conține componentele sau celulele frunzei. Dacă fiecare celulă de deasupra nivelului frunzei indică o celulă, octree-ul este considerat complet. Dacă nodurile interne sunt păstrate, arborele poate fi redus la orice nivel pentru a obține o subdiviziune mai grosieră. Octree-urile evită unul dintre defectele majore ale sistemelor de grilă fixă în cartografierea robotică: faptul că mediul nu ar trebui să fie cunoscut în prealabil și că modelul de mediu cuprinde doar volumul măsurat.

De exemplu, când ne referim la un telemetru laser, punctele de capăt ale senzorului generează spațiu ocupat, în timp ce regiunea detectată între senzor și punctul final este considerată spațiu liber. Spațiul ocupat este cartografiat din pachetele de date ale norului de puncte la distanța corespunzătoare în spațiu pentru datele noastre LiDAR de intrare. Ca rezultat, folosim datele LiDAR pentru a genera un model de mediu octree, care descrie spațiul liber (zona de deplasare) și zonele ocupate în trei dimensiuni.

O proprietate centrală a abordării noastre este că permite eficiența spațiului ocupat și liber, menținând în același timp consumul redus de memorie, ceea ce este esențial pentru hardware-ul mașinii noastre model. Octree-urile au dimensiuni fixe, așa cum este necesar pentru intrarea rețelei neuronale, bazate pe câmpul vizual al senzorului LiDAR. Nodurile care nu sunt nici ocupate, nici libere (acestea sunt întotdeauna dincolo de obstacolele detectate) sunt marcate ca necunoscute și inițializate cu zero pentru a preveni influențarea rezultatului inferenței. În plus, putem configura rezoluția la o valoare mai mică, pentru a reduce și mai mult timpii de procesare și utilizarea memoriei. Structurile de arbori sunt metodele primare utilizate în algoritmii anteriori de compresie a norului de puncte. Numeroase abordări stochează date într-un octree și efectuează codificare a entropiei cu modele de entropie realizate manual, cum ar fi histograme adaptative, contextul părinților și estimări bazate pe aproximații planare sau proximitatea vecinilor.

În forma sa cea mai simplă, octree-urile pot fi utilizate pentru a modela o proprietate Boolean. În contextul cartografierii robotice, aceasta este de obicei ocuparea unui volum. Dacă un anumit volum este măsurat ca fiind ocupat, nodul corespunzător din octree este inițializat. Orice nod neinițializat ar putea fi liber sau necunoscut în acest context Boolean. Pentru a rezolva această ambiguitate, reprezentăm explicit volumele libere în arbore. Acestea sunt create în zona dintre senzor și punctul final măsurat, de exemplu, de-a lungul unei raze determinate prin raycasting. Zonele care nu sunt inițializate modelează implicit spațiul necunoscut. Utilizarea stărilor de ocupare Boolean sau a etichetelor discrete permite reprezentări compacte ale octree-ului: Dacă toți copiii unui nod au aceeași stare (ocupat sau liber), aceștia pot fi eliminați. Acest lucru duce la o reducere substanțială a numărului de noduri care trebuie întreținute în arbore.

În ceea ce privește complexitatea accesului la date, octree-urile necesită un overhead comparativ cu o grilă 3D de dimensiune fixă datorită structurii arborelui. O interogare aleatorie pe o structură de date a unui arbore care conține n noduri cu o adâncime a arborelui de d poate fi realizată cu o complexitate de O(d) = O(logn). Parcurgerea completă a arborelui într-o manieră în adâncime necesită o complexitate de O(n). Rețineți că, în practică, octree-ul nostru este limitat la o adâncime maximă fixă dmax. Acest lucru rezultă într-o complexitate de căutare a unui nod aleatoriu de O(dmax), cu dmax fiind constant. Prin urmare, pentru o adâncime fixă dmax, overhead-ul comparativ cu o grilă 3D corespunzătoare este constant.

Octree-urile pot fi reprezentate într-o structură de arbore ierarhică sau într-o structură de arbore liniară fără pointeri. Lista de celule a unei structuri de arbore ierarhice include celula rădăcină, celulele intermediare și celulele frunze. Fiecare celulă include pointeri către celulele părinte și celulele copil. Operațiunile de localizare a punctelor și de căutare a vecinilor ar urma pointerii pentru a parcurge arborele. Lista de celule a unei structuri de arbore liniară include doar celulele frunze. Fiecare celulă are

un cod locațional care este folosit ca cheie de căutare pentru localizarea celulei. Codurile locaționale sunt compuse din coordonatele minime ale celulelor.

Octomap

OctoMap folosește o abordare probabilistică pentru a cartografia mediul. În loc de grile de ocupare binare care marchează zonele ca fiind fie libere, fie ocupate, OctoMap atribuie o valoare de probabilitate fiecărui nod, indicând probabilitatea ca volumul să fie ocupat. Această reprezentare probabilistică ține cont de zgomotul senzorilor și de incertitudine, ducând la hărți mai robuste și mai precise.

Natura ierarhică a octree-ului permite OctoMap să reprezinte eficient mediile mari. Nodurile de nivel superior acoperă volume mai mari și pot fi subdivizate în noduri mai mici și mai detaliate, după cum este necesar. Această abordare permite cadrului să își adapteze dinamica rezoluției, oferind detalii mari în zonele de interes, menținând în același timp o reprezentare grosieră în regiunile mai puțin critice.

Cadrul matematic al OctoMap se bazează pe teoria probabilității bayesiene, care oferă un mod sistematic de a actualiza probabilitățile de ocupare ale nodurilor pe baza măsurătorilor senzorilor.

Probabilitatea de ocupare a unui nod este actualizată folosind regula de actualizare bayesiană. Dată fiind o probabilitate inițială $P(O)$ și o nouă măsurătoare cu probabilitatea $P(M|O)$, probabilitatea actualizată $P(O|M)$ este calculată astfel:

$$P(O|M) = \frac{P(M|O)P(O)}{P(M)} \tag{46}$$

Unde:

- $P(O)$ este probabilitatea a priori ca nodul să fie ocupat.
- $P(M|O)$ este verosimilitatea măsurătorii dat fiind că nodul este ocupat.
- $P(M)$ este probabilitatea măsurătorii.

Pentru a simplifica calculul, OctoMap folosește reprezentarea log-odds a probabilităților. Valoarea log-odds l a unei probabilități p este definită astfel:

$$l = \log\left(\frac{p}{1-p}\right) \tag{47}$$

Folosind log-odds, regula de actualizare bayesiană devine o operațiune simplă de adunare:

$$l(O|M) = l(O) + l(M|O) - l(M) \tag{48}$$

Această reprezentare permite actualizări eficiente și numeric stabile. Implementarea OctoMap implică mai multe componente și algoritmi cheie pentru a construi și menține structura octree. Când este primită o nouă măsurătoare de la senzor, aceasta este inserată în octree. Măsurătoarea este de obicei un nor de puncte 3D obținut de la senzori precum LIDAR sau camere stereo. Fiecare punct din

nor actualizează probabilitățile de ocupare ale nodurilor pe care le intersectează, pe baza modelului senzorului.

Raycasting-ul este folosit pentru a actualiza probabilitățile de ocupare de-a lungul razelor senzorului. Pentru fiecare punct din norul de puncte, o rază este trasată de la originea senzorului până la punct. Nodurile de-a lungul razei sunt actualizate pentru a reflecta probabilitatea de a fi libere, în timp ce nodul de la punctul final este actualizat pentru a reflecta probabilitatea de a fi ocupat. Această abordare asigură că spațiul liber este modelat cu acuratețe. OctoMap suportă actualizări dinamice, permițând hărții să se schimbe pe măsură ce mediul se schimbă. Nodurile pot fi adăugate, eliminate sau actualizate pe baza noilor măsurători. Această capacitate este crucială pentru mediile în care obiectele pot să se miște sau să apară/dispară în timp.

Pentru navigarea autonomă, OctoMap oferă o reprezentare 3D detaliată a mediului, permițând planificarea traiectoriei și evitarea obstacolelor. Roboții pot folosi harta pentru a naviga prin spații complexe și aglomerate, evitând obstacolele și găsind traiectorii optime. Structura ierarhică a octree-ului permite OctoMap să reprezinte eficient mediile mari, utilizând mai puțină memorie și resurse computaționale comparativ cu grilele voxel. Capacitatea de a ajusta dinamic rezoluția îmbunătățește și mai mult eficiența.

Abordarea probabilistică a OctoMap ține cont de zgomotul senzorilor și de incertitudine, ducând la hărți mai precise. Utilizarea log-odds pentru actualizarea probabilităților asigură stabilitate numerică și calcule eficiente. Capacitatea OctoMap de a gestiona mediile dinamice și de a integra diverse modele de senzori o face extrem de flexibilă. Poate fi adaptată la diferite tipuri de senzori și scenarii, oferind performanțe robuste într-o gamă largă de aplicații.

OctoMap este un cadru puternic și versatil pentru cartografierea 3D în robotică și sisteme autonome. Reprezentarea sa probabilistică și ierarhică a mediului permite cartografierea eficientă, precisă și flexibilă, făcând-o potrivită pentru o gamă largă de aplicații. Fie că este utilizată pentru navigare autonomă, manipulare robotică, explorare sau SLAM, OctoMap oferă instrumentele necesare pentru a construi și menține hărți 3D detaliate în timp real. Pe măsură ce cercetarea și dezvoltarea în domeniul roboticii continuă să avanseze, OctoMap rămâne o componentă crucială pentru crearea de sisteme autonome inteligente și capabile.

Arhitectura rețelei neuronale de tip encoder-decoder

Spre deosebire de rețelele neuronale tradiționale, celula de memorie a unui RNN cuprinde o buclă de feedback dependentă de timp. O rețea neuronală recurentă în sine poate fi „desfășurată" de $\tau_i + \tau_o$ ori pentru a produce o arhitectură fără buclă care să corespundă lungimii intrării, dacă considerăm o secvență de intrare $[x^{<t-\tau_i>}, ..., x^{<t>}]$ care este dependentă de timp, împreună cu o secvență de ieșire $[y^{<t+1>}, ..., y^{<t+\tau_o>}]$. Rețelele desfășurate au $\tau_i + \tau_o + 1$ straturi similare sau chiar identice, ceea ce înseamnă că fiecare strat are aceleași greutăți învățate.

Această arhitectură este compusă din două modele: un teanc de mai multe unități recurente pentru citirea secvenței de intrare și codificarea acesteia într-un vector de lungime fixă, și un al doilea pentru decodificarea vectorului de lungime fixă și producerea secvenței prezise. Modelele combinate

sunt cunoscute sub numele de RNN Encoder-Decoder, care este conceput special pentru problemele de tip secvență la secvență. Dată fiind secvența de intrare $\vec{X}^{<t-\tau_i,t>}$, un encoder RNN de bază calculează secvența stărilor ascunse $(h_1, ; h_2, ; h_3, ; \ldots, ; h_N)$:

$$h_t = \tanh\left(U_{xh}x_t + U_{hh}h_{t-1}\right), \tag{49}$$

unde cele două matrice Uxh și Uhh sunt matricea de greutăți dintre stratul de intrare și stratul ascuns, și matricea de greutăți ale conexiunilor recurente într-un strat ascuns dat, respectiv.

Problema gradientului care dispare în timpul antrenării este provocarea majoră atunci când se folosesc RNN-uri simple. Semnalul gradientului poate fi multiplicat de un număr infinit de ori, până la numărul de pași de timp. Ca rezultat, un RNN clasic nu poate captura dependențele pe termen lung în datele secvențiale. Gradientul ieșirii rețelei va avea dificultăți în a se propaga înapoi pentru a afecta greutățile straturilor anterioare dacă rețeaua este foarte adâncă sau procesează secvențe lungi. Greutățile rețelei nu vor fi modificate cu succes din cauza dispariției gradientului, rezultând în valori foarte mici ale greutăților.

Pentru a contracara aceste provocări, în lucrarea noastră folosim un set de rețele Long Short-Term Memory (LSTM) atât pentru encoder, cât și pentru decoder. LSTM-urile rezolvă problema gradientului care dispare prin adăugarea a trei porți care controlează starea de intrare, ieșire și memorie, spre deosebire de rețelele neuronale recurente clasice.

$\Theta = [W_i, U_i, b_i]$ parametrizează o rețea LSTM, unde W_i încorporează greutățile porților și celulelor de memorie înmulțite cu starea de intrare, U_i reprezintă greutățile care controlează activările rețelei și b_i conține valorile de bias ale neuronilor. O secvență de ieșire a rețelei este definită ca o traiectorie optimă dorită pentru vehiculul ego:

$$Y^{<t+1,t+\tau_o>} = [y^{<t+1>}, y^{<t+2>}, ..., y^{<t+\tau_o>}], \tag{50}$$

unde $y^{<t+1>}$ este un punct de referință prezis al traiectoriei la momentul $t+1$. τ_i și τ_o nu sunt neapărat egale: $\tau_i \neq \tau_o$.

Encoder-ul LSTM ia cele mai recente eșantioane octree $\vec{X}^{<t-\tau_i,t>}$, precum și secvența traiectoriei de referință $\vec{Z}_{ref}^{<t-\tau_i,t+\tau_o>}$ pentru pasul de timp curent t, și produce un vector intermediar de dimensiune fixă c_t care păstrează corelația temporală a observațiilor anterioare. Starea ascunsă a encoder-ului LSTM h_t este calculată folosind următoarele ecuații:

$$z_t = \sigma(U_{xz}x_t + U_{hz}h_{t-1}), \tag{51}$$

$$r_t = \sigma(U_{xr}x_t + U_{hr}h_{t-1}), \tag{52}$$

$$\tilde{h}_t = \tanh(U_{xh}x_t + U_{rh}(r_t \otimes h_{t-1})), \tag{53}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t, \tag{54}$$

unde σ reprezintă funcția de activare sigmoidă. zt, rt și $\tilde{h}t$ sunt poarta de actualizare, poarta de resetare și activarea candidatului, respectiv. Uxz, Uxr, Uxh, Uhz, Uhr și U_{rh} sunt matricile de greutăți asociate. Notația $\otimes$ reprezintă un operator de înmulțire element cu element.

Decoder-ul LSTM ia eșantionul de traiectorie prezis pentru a produce eșantioanele de traiectorie ulterioare, producând întreaga traiectorie viitoare $\vec{Y}^{<t+1,t+\tau_o>}$ pentru pasul de timp curent, dat fiind vectorul de context c_t ca intrare. $\vec{Y}^{<t+1,t+\tau_o>}$ este definită ca o variabilă de secvență Y cu instanțe de date $[y^{<t+1>}, ..., y^{<t+\tau_o-1>}, x^{<t+\tau_o>}]$ într-un interval de timp specific $[t+1, t+\tau_o]$. Probabilitatea fiecărei variabile de secvență prezise este calculată astfel:

$$p(y_t|X, y_{t-1}) = g\left(U_o(Ey_{t-1} + U_s s_t + U_c c_t)\right), \tag{55}$$

unde g este o funcție de activare softmax. st este starea ascunsă curentă a decoder-ului, iar $yt-1$ reprezintă simbolul țintă anterior, în timp ce E denotă matricea de încorporare.

Variabila de secvență țintă anterioară y_{t-1} și vectorul de context c_t sunt, de asemenea, intrări pentru decoder, care folosește un singur strat unidirecțional pentru a calcula starea ascunsă s_t:

$$z'_t = \sigma(U_{yz}Ey_{t-1} + U_{sz}s_{t-1} + C_{cz}c_t), \tag{56}$$

$$r'_t = \sigma(U_{yr}Ey_{t-1} + U_{sr}s_{t-1} + C_{cr}c_t), \tag{57}$$

$$\tilde{s}_t = \tanh(U_{ys}Ey_{t-1} + U_{rs}(r'_t \otimes s_{t-1}) + C_{cs}c_t), \tag{58}$$

$$s_t = (1 - z'_t) \otimes s_{t-1} + z'_t \otimes \tilde{s}_t, \tag{59}$$

unde z'_t, $r't$ și $\tilde{s}t$ sunt poarta de actualizare, poarta de resetare și activarea candidatului, respectiv. Uxx și Cxx sunt matricile de greutăți asociate.

Decoder-ul păstrează cei mai buni candidați de secvență în algoritm atunci când creează eșantionul de traiectorie viitoare pentru fiecare pas de timp. Ca rezultat, folosind cadrul de intrare octree, modelul propus ar prezice cele mai probabile ipoteze ale traiectoriei vehiculului. Prin analogie cu problemele de traducere automată, o coordonată a unui punct într-un octree este un caracter, un octree este un cuvânt, iar secvența de octree de intrare reprezintă o propoziție. Experimentele noastre arată că un RNN encoder-decoder produce o traiectorie acceptabilă și că acuratețea predicției sale este îmbunătățită în comparație cu metodele tradiționale de predicție.

Evaluarea performanței

OctoPath a fost comparat cu algoritmul de referință hibrid A*, cu o abordare bazată pe regresie și cu o abordare bazată pe învățare CNN. Am pus algoritmul OctoPath la test în două medii distincte: (I) în simulatorul GridSim și (II) într-un mediu de navigare real, atât în interior, cât și în exterior, folosind robotul AMTU. AMTU este o platformă AgileX Scout 2.0 care acționează ca o mașină la scară 1:4, echipată cu un Lidar Hesai Pandar 40 de 360° , 4 camere e-130A care oferă o percepție vizuală de 360° a împrejurimilor, o unitate de măsurare inerțială, GPS și o placă NVIDIA AGX Xavier pentru procesarea datelor și control. Starea vehiculului a fost măsurată folosind odometria roților și unitatea de măsurare inerțială (IMU).

Toate experimentele au avut ca scop rezolvarea problemei estimării traiectoriei, care presupunea calcularea unei traiectorii pentru a naviga în siguranță mediul de conducere fără a realiza sarcina de

control al mișcării. Pentru a implementa controlul mișcării, stările prezise au fost folosite ca intrare pentru un controler predictiv de model, care a calculat semnalele de control v_x și ω_z necesare pentru AMTU. Proiectarea și implementarea controlerului de mișcare sunt în afara scopului acestei lucrări.

Algoritmul hibrid A* utilizează o regulă modificată de actualizare a stării pentru a aplica o variantă a bine-cunoscutului algoritm A* la modelul de mediu octree al vehiculului. Spațiul de căutare (x, y, θ) este discretizat, la fel ca în A* tradițional, dar spre deosebire de A*, care permite doar vizitarea centrelor celulelor, versiunea hibridă a algoritmului asociază o stare mai continuă a mașinii cu fiecare celulă a grilei, permițând, de asemenea, puncte de traiectorie care nu sunt exact în centrul celulei octree.

În cazul predicției traiectoriei ca problemă de regresie, obiectivul este de a obține o predicție directă a pozițiilor continue viitoare fără nicio discretizare. Deoarece predicția medie minimizează eroarea de regresie, astfel de metode au o tendință de a produce media mai multor opțiuni, făcându-le astfel inexacte.

Algoritmul Neural RRT* este un algoritm nou de planificare optimă a traiectoriei bazat pe rețele neuronale convoluționale. A folosit algoritmul A* pentru a genera date de antrenament, considerând informațiile hărții ca intrare și traiectoria optimă ca referință. Având o nouă problemă de planificare a traiectoriei, modelul poate determina rapid distribuția probabilității traiectoriei optime, care este apoi folosită pentru a direcționa operația de eșantionare a planificatorului RRT*. Performanța algoritmului variază în funcție de diferite valori ale distanței de siguranță față de obstacole și ale mărimii pasului. O distanță de siguranță mai mare indică faptul că ruta planificată este departe de obstacole, în timp ce o distanță de siguranță mai mică indică faptul că traseul planificat este mai aproape de acestea. Am folosit o mărime fixă a pasului de 2 și o valoare a distanței de siguranță de 4.

Folosim RMSE între traiectoria prezisă și cea înregistrată în planul de conducere 2D:

$$RMSE = \sqrt{\frac{1}{\tau_o}\sum_{t=1}^{\tau_o}\left[(\hat{p}_x^{<t>} - p_x^{<t>})^2 + (\hat{p}_y^{<t>} - p_y^{<t>})^2\right]}, \tag{60}$$

unde $\hat{p}_x^{<t>}$, $\hat{p}_y^{<t>}$ sunt punctele de pe traiectoria prezisă, iar $p_x^{<t>}$, $p_y^{<t>}$ sunt punctele de pe traiectoria de referință, respectiv. Am stabilit orizontul de predicție $\tau_o = 10$.

Fluxul de lucru al experimentelor este următorul:

- colectarea datelor de antrenament din înregistrările de conducere;
- generarea octree-urilor și formatarea datelor de antrenament ca secvențe;
- antrenarea rețelei OctoPath;
- evaluarea în scenarii de conducere simulate și reale.

Acest setup experimental a rezultat în 15 km de conducere în GridSim, peste 1 km de navigare interioară buclată și peste 2 km de navigare exterioară în afara Institutului pentru Cercetare al Universității Transilvania din Brașov. Robotul a navigat în medii interioare și exterioare, evitând obstacole statice și dinamice.

Table 1: Erori între traiectoriile estimate și cele de referință în scenarii de testare a navigației în simulare și în lumea reală.

Scenariu	Metoda	$\bar{e}_x[m]$	max$(e_x)[m]$	$\bar{e}_y[m]$	max$(e_y)[m]$	$RMSE[m]$
Simulare	A* hibrid	1.43	3.21	2.71	4.01	2.71
GridSim	Regresie	3.51	7.20	4.71	8.53	5.10
	Neural RRT	1.27	3.01	2.35	2.98	2.48
	Octopath	1.16	2.31	1.72	2.75	2.07
Navigare	A* hibrid	1.21	4.33	1.33	3.88	1.74
interioară	Regresie	1.90	5.73	2.31	4.98	2.75
	Neural RRT	1.01	3.29	0.98	2.16	1.44
	Octopath	0.55	1.08	0.44	0.87	0.69
Navigare	A* hibrid	1.35	4.67	1.44	4.44	1.98
exterioară	Regresie	2.41	8.42	2.77	8.98	3.01
	Neural RRT	1.05	2.52	1.06	3.24	1.17
	Octopath	0.71	1.46	0.57	1.17	0.88

GridSim este un motor de simulare pentru conducere autonomă care generează grile de ocupare sintetice din senzori simulați folosind modele cinematice, care sunt apoi utilizate pentru a produce date de intrare pentru octree. Interfața utilizator a fost integrată în meniul mediului GridSim, astfel încât modurile pot fi comutate între redare, înregistrare și antrenament, fiecare având acces la cele cinci scenarii diferite. Există un număr mare de parametri configurabili, cum ar fi rezoluția simulatorului, precizia grilei de ocupare, numărul participanților la trafic, dimensiunea vehiculului ego, viteza maximă sau raza de viraj.

Obiectivul este de a ajunge dintr-o poziție de plecare la o destinație dată evitând coliziunile și conducând la viteza dorită. Coordonata Z a tuturor punctelor de obstacole și spații libere este setată la zero pentru a ajusta datele de intrare ale rețelei encoder-decoder la mediul GridSim. Scenariile de testare generate folosind mediul de simulare GridSim nu au fost utilizate în timpul antrenării rețelei.

Pentru diferitele tipuri de drumuri și medii de trafic găsite în baza de date de testare sintetică, evaluarea performanței algoritmilor de referință este rezumată în partea de sus a Tabelului 1. Erorile medii de poziție ($\bar{e}_x$, $\bar{e}_y$), precum și metrica RMSE din Ecuația (60) sunt ilustrate.

Experimentul de navigare interioară a fost realizat folosind vehiculul robot mobil AMTU cu tracțiune diferențială, cu diferite sarcini de navigare interioară. Rutele de referință pe care mașina trebuia să le urmeze erau compuse din linii drepte, curbe în S, cercuri și un traseu de 75 m pe holul principal al Institutului pentru Cercetare al Universității Transilvania din Brașov.

Camera de testare pentru experimentul interior a fost aceeași cu cea utilizată pentru colectarea datelor de antrenament, dar rutele de referință și obstacolele au fost amplasate diferit. Holul principal nu a fost utilizat pentru colectarea datelor de antrenament.

Primul set de 10 încercări a fost realizat fără obstacole prezente pe rutele de referință, în timp

ce al doilea set de 10 încercări a conținut obstacole statice și dinamice. Au fost colectate cincizeci și patru de mii de mostre de antrenament sub formă de date LiDAR și stări ale vehiculului. Traseul parcurs la colectarea datelor a fost considerat ca o traiectorie de referință și a fost creat într-un mod auto-supervizat.

Experimentul de navigare exterioară a fost realizat în afara Institutului pentru Cercetare al Universității Transilvania din Brașov. Traseul de referință pe care mașina trebuia să-l urmeze era compus dintr-un circuit complet în jurul institutului și a fost creat folosind un instrument GPS. Traseul în sine are aproximativ 500 m lungime și a fost parcurs de 4 ori.

Traseul de referință exterior, care a fost utilizat pentru antrenarea rețelei, a fost înregistrat ca traseu parcurs în timpul colectării datelor senzoriale. La testarea rețelei, traseul de referință a fost generat folosind instrumentul nostru de planificare a misiunilor vehiculului. Obstacolele statice au fost în principal mașini parcate, în timp ce obstacolele dinamice au fost mașini în mișcare sau oameni.

Media și deviația standard a erorii de poziție (calculată ca RMSE), partea stângă pentru navigarea interioară și partea dreaptă pentru navigarea exterioară. Erorile de poziție sunt prezentate în Tabelul 1 pentru toate scenariile: simulare, navigare interioară și navigare exterioară. Media ($\bar{e}_x$, $\bar{e}_y$) și maximul (max(e_x), max(e_y)) erorilor de poziție, precum și metrica RMSE din Ecuația (60), sunt afișate. Comparativ cu OctoPath, Neural RRT are cele mai mici deviații, dar, dintre abordările non-învățare, Hybrid A* performează cel mai bine, indicând că este un bun candidat pentru estimarea traiectoriei non-învățare.

În experimentele realizate, algoritmul hibrid A* s-a comportat mai bine decât abordarea de regresie, în mare parte datorită structurii datelor de intrare ale modelului de mediu octree. Acest lucru face ca A* să fie strict dependent de precizia reprezentării obstacolelor în mediul înconjurător. În plus, efectul de tremurat al OctoPath poate fi un efect secundar al naturii discrete a ieșirii decodorului. Totuși, acesta va oferi o predicție fiabilă a traiectoriei vehiculului pe un orizont de timp dat.

Camerele sincronizate quad e-CAM130A vor fi utilizate în cercetările viitoare pentru a realiza o segmentare semantică completă a norului de puncte primit și pentru a extinde validarea abordării noastre la mai multe cazuri de utilizare. Abordările bazate pe învățare au demonstrat că pot oferi rezultate mai bune pe termen lung decât metodele convenționale. Această îmbunătățire ar fi realizată prin antrenarea pe mai multe date, care ar include un număr mai mare de cazuri limită.

Contribuții personale

Pe baza obiectivelor menționate anterior, contribuțiile personale se încadrează în aceleași patru categorii, detaliate după cum urmează:

1. Analiza literaturii de specialitate: accent deosebit pe tehnicile de învățare profundă și evidențierea impactului lor transformator asupra domeniului conducerii autonome.

2. Dezvoltarea mediilor de simulare: oferirea unui mediu controlat pentru testarea și validarea algoritmilor, asigurând experimente sigure și repetabile.

3. Dezvoltarea algoritmilor: fuziunea senzorilor pentru o percepție îmbunătățită și algoritmi de planificare a traseului bazați pe învățare.

4. Evaluare și benchmarking: evaluare cuprinzătoare și benchmarking în raport cu metodele consacrate, demonstrând eficacitatea și fiabilitatea metodelor dezvoltate.

Analiza literaturii de specialitate

Revizuirea evidențiază cu succes potențialul transformator al conducerii autonome, subliniind impactul acesteia asupra siguranței, eficienței și comodității în transport. Se subliniază avansurile rapide în tehnologie, cum ar fi fuziunea senzorilor, învățarea automată și planificarea traseelor, care sunt critice pentru dezvoltarea vehiculelor autonome. Capitolul identifică, de asemenea, provocările cheie, inclusiv robustețea mediului, interacțiunea om-vehicul, cadrele de reglementare și securitatea cibernetică. Prin abordarea acestor provocări prin cercetare interdisciplinară și colaborare, capitolul stabilește o fundație solidă pentru teza, având ca scop avansarea domeniului conducerii autonome prin algoritmi inovatori de planificare a traseelor și tehnici de integrare a senzorilor.

Pentru a rezuma, principalele contribuții ale acestei categorii sunt:

- Realizarea unei revizuiri complete a literaturii pentru a identifica și evalua metodologiile și tehnologiile existente în navigația autonomă și planificarea traseelor, evidențiind avansurile cheie și provocările persistente.

- Identificarea lacunelor și limitărilor critice în cercetările actuale, oferind o bază pentru dezvoltarea de soluții inovatoare în navigația vehiculelor autonome.

Dezvoltarea mediilor de simulare

Această secțiune detaliază platformele experimentale dezvoltate și utilizate în timpul cercetării tezei. Acoperă crearea mediului de simulare GridSim, care a fost esențial pentru testarea abordărilor propuse de învățare profundă pentru conducerea autonomă. De asemenea, descrie dezvoltarea și testarea prototipului Robot Raspberry Pi și a platformei AMTU, subliniind rolurile lor în validarea algoritmilor propuși. Prin furnizarea unui mediu robust de simulare și testare, se asigură că algoritmii de planificare a traseelor pot fi evaluați riguros în condiții realiste, sporind astfel fiabilitatea și performanța acestora în scenarii reale.

Contribuțiile principale ale acestei categorii sunt:

- Dezvoltarea unui instrument de simulare robust (numit GridSim) proiectat pentru a emula condițiile reale de conducere, facilitând testarea și validarea controlată a algoritmilor propuși.
- Asigurarea că instrumentul de simulare reprezintă cu acuratețe scenariile reale diverse și dinamice, sporind fiabilitatea testării algoritmilor.
- Dezvoltarea prototipului Robot Raspberry-PI și a principalei platforme robotice utilizate pentru experimente, robotul mobil AMTU cu tracțiune diferențială.

Dezvoltarea algoritmilor

Această secțiune acoperă dezvoltarea algoritmilor de planificare a traseelor și de arbitraj comportamental pentru vehicule autonome. Introduce conceptul de reprezentare pe bază de grilă și abordează provocările navigării în medii complexe. Detaliază mecanismul de arbitraj comportamental, care implică descrierea, analiza și modelarea scenelor de conducere. Abordarea Neuro-Trajectory, o metodă neuroevolutivă multiobiectivă, este prezentată ca o soluție pentru învățarea traiectoriilor locale de stare. Prin combinarea eficientă a sarcinilor de învățare cognitivă cu antrenamentul neuroevolutiv, acest capitol contribuie semnificativ la avansarea capacităților vehiculelor autonome în medii dinamice și imprevizibile.

Ulterior, cercetarea este extinsă la o reprezentare 3D folosind OcTrees, îmbunătățind percepția spațială și capacitățile de planificare a traseelor vehiculelor autonome. Este discutată implementarea sistemului OctoPath, o rețea neurală encoder-decoder auto-supervizată, concepută pentru a prezice traseele optime ale vehiculului în medii 3D. Prin tranziția de la reprezentările 2D la cele 3D, acest capitol demonstrează o îmbunătățire semnificativă a acurateței și fiabilității algoritmilor de planificare a traseelor, contribuind la soluții de conducere autonomă mai robuste și eficiente.

În rezumat, principalele contribuții ale acestei categorii sunt:

- Proiectarea și implementarea unui cadru cuprinzător de fuziune a senzorilor, integrând date de la LiDAR, camere și IMU-uri.
- Demonstrarea unor îmbunătățiri semnificative ale acurateței și fiabilității percepției, contribuind la conștientizarea situațională a vehiculului autonom în arbitrajul comportamental.
- Dezvoltarea, implementarea și testarea riguroasă a unui algoritm nou bazat pe învățare pentru planificarea eficientă și precisă a traseelor în grile 2D.
- Dezvoltarea OctoPath, o rețea neurală profundă encoder-decoder auto-supervizată care prezice traseul optim pentru vehicul, gestionând eficient datele spațiale 3D, permițând predicții precise și scalabile ale traiectoriei în medii complexe.

Evaluare și benchmarking

Principalele contribuții ale acestei categorii sunt:

- Dezvoltarea unui mediu de dezvoltare sigur și eficient pe placa NVidia AGX Xavier, asigurând performanța și securitatea optimă a algoritmilor implementați.
- Stabilirea diverselor metrici de performanță și scenarii de testare pentru a evalua riguros fiabilitatea și eficiența algoritmilor dezvoltați.
- Asigurarea că algoritmii propuși funcționează optim pe diverse seturi de date și se adaptează eficient la diferite condiții de conducere, demonstrând versatilitate și robustețe.
- Evidențierea implicațiilor practice ale cercetării, subliniind potențialul acesteia de a îmbunătăți siguranța, eficiența și fiabilitatea vehiculelor autonome în aplicații reale.

Diseminarea rezultatelor cercetării

Diseminarea rezultatelor cercetării a fost esențială pentru avansarea domeniului vehiculelor autonome și a sistemelor inteligente. Au fost depuse mai multe brevete referitoare la unele dintre tehnologiile și metodologiile dezvoltate în timpul acestei cercetări:

- Grigorescu, S., Trasnea, B., Vasilcoi A., Training of a convolutional neural network, World Intellectual Property Organization, Patent no. WO2021008798A1, 2020.
- Grigorescu, S., Macesanu, G., Cocias, T., Trasnea, B., Ginerica, C., Generating training images for machine learning-based object recognition systems, European Patent Office, Patent no. EP3343432A1, 2018.
- Vasilcoi A., Radu P., Marina L., Trasnea, B., Grigorescu, S., Convolutional neural network with reduced complexity, European Patent Office, Patent no. EP3343432A1, 2020.
- Trasnea, B., Grigorescu, S., Trajectory estimation for vehicles, European Patent Office, Patent no. EP3839830A1, 2021.

Mai multe publicații în reviste științifice internaționale și în volumele conferințelor au rezultat din activitatea desfășurată pe parcursul studiilor de doctorat. Trei lucrări de cercetare au fost publicate ca prim autor, și au fost aduse contribuții importante la alte opt, după cum urmează:

- Trasnea, B., Ginerica, C., Zaha, M., Macesanu, G., Pozna, C. and Grigorescu, S., OctoPath: An OcTree-Based Self-Supervised Learning Approach to Local Trajectory Planning for Mobile Robots. Sensors, 21(11), p.3606, 2021. DOI: 10.3390/s21113606

- Trasnea, B., Pozna, C. and Grigorescu, S.M., AIBA: An AI Model for Behavior Arbitration in Autonomous Driving. In Multi-disciplinary Trends in Artificial Intelligence: 13th International Conference, MIWAI 2019, Kuala Lumpur, Malaysia, Proceedings (Vol. 11909, p. 191). Springer Nature, November 17–19, 2019. DOI: 10.1007/978-3-030-33709-417

- Trasnea, B., Marina, L.A., Vasilcoi, A., Pozna, C.R. and Grigorescu, S.M., GridSim: A vehicle kinematics engine for deep neuroevolutionary control in autonomous driving. In 2019 Third IEEE International Conference on Robotic Computing (IRC) (pp. 443-444). IEEE, February 2019. DOI: 10.1109/IRC.2019.00091

- Grigorescu, S., Trasnea, B., Cocias, T. and Macesanu, G., A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3), pp.362-386, 2020. DOI: 10.1002/rob.21918

- Marina, L.A., Trasnea, B. and Grigorescu, S.M., A multi-platform framework for artificial intelligence engines in automotive systems. In 2018 22nd International conference on system theory, control and computing (ICSTCC) (pp. 559-564). IEEE, October 2018. DOI: 10.1109/icstcc.2018.8540753

- Marina, L.A., Trasnea, B., Cocias, T., Vasilcoi, A., Moldoveanu, F. and Grigorescu, S.M., Deep Grid Net (DGN): A deep learning system for real-time driving context understanding. In 2019 Third IEEE International Conference on Robotic Computing (IRC) (pp. 399-402). IEEE, February 2019. DOI: 10.1109/IRC.2019.00073

- Grigorescu, S.M., Trasnea, B., Marina, L., Vasilcoi, A. and Cocias, T., Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles. IEEE Robotics and Automation Letters, 4(4), pp.3441-3448, 2019. DOI: 10.1109/lra.2019.2926224

- Grigorescu, S., Cocias, T., Trasnea, B., Margheri, A., Lombardi, F. and Aniello, L., Cloud2Edge Elastic AI Framework for Prototyping and Deployment of AI Inference Engines in Autonomous Vehicles. Sensors, 20(19), p.5450, 2020. DOI: 10.3390/s20195450

- Grigorescu, S., Zaha, M., Trasnea, B. and Ginerica, C., Embedded Vision for Self-Driving on Forest Roads. IEEE Computer Vision and Pattern Recognition conference (CVPR), Workshop Demo on Embedded Vision. June 2021.

- Grigorescu, S., Ginerica, C., Zaha, M., Macesanu, G. and Trasnea, B., LVD-NMPC: A Learning-based Vision Dynamics Approach to Nonlinear Model Predictive Control for Autonomous Vehicles. International Journal of Advanced Robotic Systems. May 2021. DOI: 10.1177/17298814211019544

- Ginerica, C., Zaha, M., Gogianu, F., Busoniu, L., Trasnea, B. and Grigorescu, S. ObserveNet Control: A Vision-Dynamics Learning Approach to Predictive Control in Autonomous Vehicles. IEEE Robotics and Automation Letters, 6(4), pp.6915-6922, 2021. DOI: 10.1109/LRA.2021.3096157

Părți din rezultatele cercetării au fost, de asemenea, prezentate la Forumul European de Robotică din 2019, desfășurat la București, România, unde peste 50 de expozanți și-au prezentat prototipurile, produsele și serviciile, iar vizitatorii au avut ocazia să descopere cea mai avansată industrie de robotică, institute de cercetare și proiecte din Europa.

Concluzii și direcții viitoare de cercetare

Cercetarea a început cu o revizuire extinsă a avansărilor tehnologice și metodologiilor care stau la baza sistemelor de conducere autonomă, concentrându-se pe impactul transformator al tehnicilor de deep learning. Aceasta a detaliat aplicarea Rețelelor Neuronale Convoluționale (RNC) pentru detectarea obiectelor, recunoașterea imaginilor și segmentarea scenelor, și a Rețelelor Neuronale Recurente (RNR) pentru predicția secvențelor și modelarea comportamentului. În plus, Deep Reinforcement Learning (DRL) este explorată pentru dezvoltarea politicilor care permit vehiculelor autonome să învețe strategii optime de conducere prin interacțiunea cu mediul.

Aplicările acestor tehnologii au fost examinate în detaliu, în special în înțelegerea scenelor de conducere, planificarea traseului, arbitrajul comportamentului și fuziunea senzorilor. Implementarea practică utilizează cadre Python precum TensorFlow, Keras și PyTorch pentru dezvoltarea și implementarea modelelor de deep learning. Studiul a trecut apoi la platforme experimentale, începând cu dezvoltarea Simulatorului GridSim pentru testarea algoritmilor de conducere autonomă într-un mediu controlat.

Principala platformă robotică, AMTU, echipată cu camere e-Cam130A quad și un LiDAR Hesai Pandar de 360 de grade, a validat algoritmii în condiții realiste. Teza detaliază procesul de dezvoltare iterativă și experimentală, integrând și testând algoritmii cheie pe aceste platforme pentru a acoperi decalajul dintre cercetarea teoretică și implementarea practică.

Cercetarea a culminat cu avansarea predicției traiectoriei vehiculelor autonome prin algoritmii NeuroTrajectory și OctoPath. NeuroTrajectory folosește o abordare neuroevolutivă cu optimizare multiobiectivă Pareto pentru a învăța traiectorii locale de stare din grile de ocupare. OctoPath, utilizând învățarea auto-supervizată cu un model de mediu bazat pe octree, prezice traiectorii locale eficient, demonstrând funcționarea în timp real pe Nvidia AGX Xavier.

Adaptabilitatea OctoPath la diferite rezoluții de mediu și capacitatea sa de a gestiona date spațiale tridimensionale au evidențiat eficiența sa în medii dinamice. Analizele comparative au validat OctoPath ca o metodă robustă pentru predicția traiectoriei locale, subliniind aplicabilitatea sa practică atât în simulare, cât și în scenarii reale.

Proiectele de cercetare finalizate pe parcursul programului de doctorat demonstrează potențialul semnificativ al deep learning în industriile auto și robotică. Cu toate acestea, pe baza cercetărilor

actuale și a contribuțiilor acestei teze, există câteva provocări și constrângeri care trebuie soluționate. Planificarea traseului rămâne o provocare critică în navigația vehiculelor autonome. Cercetările viitoare ar trebui să se concentreze pe dezvoltarea unor algoritmi mai sofisticați care pot gestiona medii dinamice și complexe. Domeniile potențiale includ:

- Planificarea traseelor multi-agent: explorarea strategiilor de coordonare între mai multe vehicule autonome pentru optimizarea fluxului de trafic și reducerea congestiei.
- Tehnici avansate de fuziune a senzorilor: investigarea unor metode noi de combinare a datelor provenite de la senzori diverși (de exemplu, LiDAR, radar, camere) pentru îmbunătățirea percepției mediului și detectarea obiectelor.
- Algoritmi eficienți din punct de vedere energetic: proiectarea de algoritmi care să maximizeze eficiența computațională și să minimizeze consumul de energie, esențial pentru vehiculele electrice.
- Integrarea calculului de înaltă performanță: valorificarea avansurilor hardware, cum ar fi GPU-urile și FPGA-urile, pentru a îmbunătăți capacitățile de procesare ale sistemelor încorporate.
- Reducerea latenței: dezvoltarea de protocoale de comunicare cu latență scăzută pentru a asigura schimbul prompt de date între vehicule și infrastructură.
- Robustețe în condiții adverse: dezvoltarea algoritmilor care să mențină performanțe ridicate în condiții dificile, cum ar fi ploaie torențială, ceață sau zăpadă.
- Algoritmi de întreținere predictivă: dezvoltarea de modele de învățare automată care să prezică și să prevină defecțiunile sistemului înainte ca acestea să apară.

În plus, ar trebui acordată o atenție mai mare combinării și optimizării designurilor rețelelor neuronale. De asemenea, ar trebui investigate funcții de pierdere hibride alternative și tehnici de antrenament, cum ar fi setup-ul adversarial. În concluzie, viitorul cercetării în domeniul vehiculelor autonome este plin de oportunități pentru inovație și îmbunătățire. Abordând aceste direcții, cercetătorii pot contribui la dezvoltarea unor sisteme autonome mai sigure, mai eficiente și mai prietenoase cu utilizatorul.