

IOSUD – Universitatea Dunărea de Jos din Galați
Școala Doctorală de Științe Fundamentale și Inginerești



TEZĂ DE DOCTORAT

Contribuții privind conducerea autonomă a vehiculelor

Rezumat

**Doctorand,
Cristian SANDU**

**Conducător științific,
Prof. univ. dr. ing. Luminița DUMITRIU**

Seria I2 Calculatoare și tehnologia Informației Nr 10

**Galați
2025**

IOSUD – Universitatea Dunărea de Jos din Galați
Școala Doctorală de Științe Fundamentale și Inginerești



TEZĂ DE DOCTORAT

Contribuții privind conducerea autonomă a vehiculelor

Doctorand,
Cristian SANDU

Președinte:

Prof. univ. dr. ing. BARBU Marian
Rector al Universității „Dunărea de Jos” din Galați

Conducător științific:

Prof. univ. dr. ing. Luminița DUMITRIU
Universitatea „Dunărea de Jos” din Galați

Referenți științifici

Prof. univ. dr. ing. TRĂUȘAN-MATU Ștefan
Universitatea Națională de Știință și Tehnologie
POLITEHNICA București

Conf. univ. dr. ing. MIHALACHE Sanda-Florentina
Universitatea Petrol-Gaze din Ploiești

Conf. univ. dr. ing. COCU Adina
Universitatea „Dunărea de Jos” din Galați

Seria I2 Calculatoare și tehnologia Informației Nr 10

Galați
2025

Seriile tezelor de doctorat susținute public în UDJG începând cu 1 octombrie 2013 sunt:

Domeniul fundamental ȘTIINȚE INGINEREȘTI

Seria I 1:	Biotehnologii
Seria I 2:	Calculatoare și tehnologia informației
Seria I 3:	Inginerie electrică
Seria I 4:	Inginerie industrială
Seria I 5:	Ingineria materialelor
Seria I 6:	Inginerie mecanică
Seria I 7:	Ingineria produselor alimentare
Seria I 8:	Ingineria sistemelor
Seria I 9:	Inginerie și management în agricultură și dezvoltare rurală

Domeniul fundamental ȘTIINȚE SOCIALE

Seria E 1:	Economie
Seria E 2:	Management
Seria E 3:	Marketing
Seria SSEF:	Știința sportului și educației fizice
Seria SJ:	Drept

Domeniul fundamental ȘTIINȚE UMANISTE

Seria U 1:	Filologie- Engleză
Seria U 2:	Filologie- Română
Seria U 3:	Istorie
Seria U 4:	Filologie - Franceză

Domeniul fundamental MATEMATICĂ ȘI ȘTIINȚE ALE NATURII

Seria C:	Chimie
----------	---------------

Domeniul fundamental ȘTIINȚE BIOMEDICALE

Seria M:	Medicină
Seria F:	Farmacie

Cuprins

Capitolul I: Stadiul actual al planificării rutelor și itinerariilor pentru vehicule autonome.....	3
1.1 Algoritmi consacrați pentru planificarea rutelor	3
1.1.1 Algoritmi bazați pe grafuri	3
1.1.2 Algoritmi de planificare a traseului prin eșantionare	3
1.1.3 Comparatie algoritmi	4
1.2 Metode de planificare a traseelor bazate pe învățare automată	4
1.3 Abordări de planificare a traseelor bazate pe date LiDAR	5
1.4 Planificarea traseelor bazată pe algoritmi genetici	5
1.5 Concluzii	5
Capitolul II: Contribuții privind propunerea unei abordări pentru evitarea coliziunilor în timp real...	6
2.1 Algoritmul Safety Bubbles LiDAR.....	6
2.2 Prezentare generală și arhitectura sistemului	7
2.2.1 Transformarea coordonatelor și compensarea mișcării	8
2.2.2 Gruparea și reprezentarea geometrică a obiectelor în mișcare	8
2.2.3 Construirea dinamică a bulelor de siguranță	8
2.2.4 Verificarea coliziunilor și ajustarea traiectoriei	9
2.3 Analiza și evaluarea performanței	9
2.3.1 Eficiența detecției coliziunilor	10
2.3.2 Performanța în timp real	10
2.3.3 Adaptabilitatea în condiții diverse	10
2.3.4 Comportamentul în scenarii reale de evitare	11
2.3.5 Limitări	11
2.3.6 Concluzii	12
2.4 Analiză comparativă cu metodele din literatură	12
2.5 Concluzie	13
Capitolul III: CONTRIBUȚII PRIVIND INTEGRAREA DATELOR ÎN TIMP REAL ÎN ALGORITMI GENETICI PENTRU PLANIFICAREA TRASEELOR.....	13
3.1 Optimizarea dinamică și în timp real a traseelor	13
3.2 Contribuții	14
3.3 Structurarea algoritmului genetic.....	14
3.3.1 Reprezentarea cromozomilor	14
3.3.2 Generarea populației inițiale	15
3.3.3 Funcția fitness	15
3.3.4 Operatorul de selecție.....	15
3.3.5 Operator de recombinare (crossover)	15
3.3.6 Operatorul de mutație	16
3.3.7 Elitismul	16
3.3.8 Criterii de evaluare	16
3.4 Integrarea datelor de trafic și meteo în timp real	17
3.4.1 Integrarea datelor de trafic de la Google Maps.....	17
3.4.2 Integrarea datelor despre vreme de la OpenWeatherMap.....	17
3.4.3 Calitatea datelor și limitări	18
3.5 Simulare CARLA	18
3.5.1 Configurarea mediului.....	18
3.5.2 Controlul vehiculului autonom.....	18
3.5.3 Simulare condiții meteorologice	19
3.5.4 Rulare simulare	19
3.5.5 Criterii de performanță	19
3.5.6 Rezultate.....	20
3.5.7 Comparatie între performanțele algoritmului îmbunătățit cu date real time.....	20
3.5.8 Impactul modificării valorilor parametrilor	20
3.5.9 Analiza comparativă a performanțelor	21
3.5.10 Discuție	21
3.5.11 Sumar contribuții	21
Capitolul IV: Utilizarea simulatorului CARLA pentru testarea algoritmilor	22
4.1 Caracteristici simulare cu CARLA.....	22

4.1.1 Generarea vehiculelor și pietonilor	22
4.1.2 Controlul mediului înconjurător: vreme și iluminat	22
4.1.3 Senzori	23
4.1.4 Navigație și controlul vehiculelor	23
4.1.5 Traffic Manager și rutare automată	24
4.2 Utilizarea CARLA în literatura de specialitate	24
4.3 Utilizarea CARLA în cercetarea proprie	25
CAPITOLUL V: PROPUNERI DE IMPLEMENTARE	25
5.1 Ghid turistic autonom	25
5.1.1 Arhitectura hardware	26
5.1.2 Arhitectura senzorială	26
5.1.3 Arhitectura software	27
5.1.4 Planificare globală	27
5.1.5 Planificare locală	28
5.1.6 Reconfigurare dinamică pe baza condițiilor în timp real	28
5.1.7 Concluzie	29
Capitolul VI: Concluzii și direcții viitoare	29
Bibliografie	31

Capitolul I: Stadiul actual al planificării rutelor și itinerariilor pentru vehicule autonome

1.1 Algoritmi consacrați pentru planificarea rutelor

Vehiculele autonome necesită algoritmi robuști de planificare a traseelor, capabili să opereze în medii complexe și dinamice. În literatura de specialitate, se disting două familii majore de soluții: algoritmi de căutare pe grafuri (A^* , Dijkstra, D^* , etc.) și algoritmi de tip eșantionare (RRT, PRM, etc.). Aceste metode au fost preluate din domeniul roboticii și adaptate în industria auto, fiind utilizate atât pentru planificarea globală, cât și pentru evitarea locală a coliziunilor [11].

Algoritmii de căutare pe grafuri oferă garanții de optimalitate și sunt potriviți pentru medii structurate, în timp ce metodele de eșantionare permit explorarea eficientă a spațiilor continue, cu complexitate redusă. De regulă, sistemele moderne AV combină aceste abordări într-o arhitectură stratificată: un planificator global calculează ruta optimă, iar un planificator local reacționează la schimbări imediate din mediu. Această combinație între planificare deterministă și reacție adaptivă este cheia funcționării în timp real a sistemelor autonome [11].

1.1.1 Algoritmi bazați pe grafuri

Algoritmul Dijkstra este unul dintre cei mai cunoscuți algoritmi de căutare de trasee. El explorează întreg spațiul de stări și oferă o soluție optimă, dar la un cost computațional ridicat. Algoritmul A^* îmbunătățește semnificativ această abordare prin utilizarea unei funcții euristice, care ghidează căutarea către țintă. A^* este optim și complet dacă euristica este admisibilă [12]. Totuși, în cazul discretizării hărților, traseele generate pot fi neregulate și pot duce la manevre nefezabile, necesitând ulterior o etapă de post-procesare (spline, clotoide).

Pentru a îmbunătăți continuitatea cinematică, cercetătorii au introdus rețele de stări (state lattices), care permit generarea de traiectorii fezabile direct din căutarea A^* . De asemenea, au fost dezvoltate variante adaptive și incrementale, precum D^* Lite și iADA*, care oferă replanificare rapidă în medii dinamice [13]. D^* Lite, de exemplu, reutilizează informațiile din căutările anterioare, ceea ce reduce drastic timpul de actualizare a traseului.

Hybrid A^* este o extensie importantă care ia în calcul orientarea și dinamica vehiculului. Utilizând modele de tip Dubins sau Reeds-Shepp, Hybrid A^* generează trasee cu curbură continuă, fiind potrivit pentru vehicule cu constrângeri de virare. THA* (Traversability Hybrid A^*) adaugă și informații despre traversabilitatea terenului, oferind soluții superioare pentru medii off-road [14]. De asemenea, au fost propuse extensii energetice [15], care optimizează traseele din perspectiva consumului de combustibil.

În sisteme reale, planificarea A^* este deseori completată de un planificator local precum Dynamic Window Approach (DWA), care evaluează în timp real opțiunile de deplasare într-un orizont scurt. Abordări precum A^* -DWA combină avantajele optimalității globale cu reactivitatea locală, permițând sistemelor să navigheze în siguranță prin medii mari și dinamice [16].

1.1.2 Algoritmi de planificare a traseului prin eșantionare

Planificarea pe bază de eșantionare este esențială pentru gestionarea spațiilor de stări continue, cu complexitate spațială ridicată. RRT (Rapidly-Exploring Random Tree) construiește un arbore explorând aleatoriu spațiul liber și conectând incremental nodurile. Este complet probabilistic: dacă există o soluție, o va găsi cu suficienți pași [17]. Totuși, RRT nu

este optim. De aceea, a fost propus RRT*, care aplică o strategie de reconectare (rewiring) și convergență asimptotică către soluția optimă [17].

Mai recent, au fost dezvoltate variante ghidate de informații, precum Informed RRT* și Directional RRT*, care restrâng spațiul de eșantionare la zone relevante (de exemplu, elipse între start și scop) [18]. Alte metode precum Neural RRT* integrează rețele neuronale care învață din exemple anterioare și sugerează puncte de eșantionare promițătoare [19]. RRT-X introduce o abordare incrementală, similară lui D*, adaptând arborele la modificări ale mediului fără a-l reconstrui complet [20].

RRT este foarte util în planificarea locală și în scenarii cu multe obstacole, precum parcurile sau intersecțiile aglomerate. Deși soluțiile sale necesită netezire post-procesare (de exemplu, spline cubic B), viteza de generare a unei soluții fezabile îl face extrem de valoros în sisteme cu constrângeri de timp. De asemenea, variantele kinodinamice ale RRT integrează direct limitările fizice ale vehiculului, generând trasee fezabile din punct de vedere cinematic și dinamic [20].

1.1.3 Comparație algoritmi

Compararea metodelor clasice evidențiază clar compromisurile între optimalitate, adaptabilitate și costul computațional. A* și Dijkstra oferă garanții de optimalitate, dar sunt lente în medii dinamice. D* Lite și iADA* oferă replanificare rapidă, însă au un cost de implementare mai ridicat [13]. RRT este rapid, dar nu optim, în timp ce RRT* oferă un traseu mai scurt, cu un cost de calcul mai mare [17][19]. PRM este ideal pentru planificare offline și pentru medii statice, dar reacționează greu la schimbări în timp real [21].

În sistemele AV, aceste metode sunt adesea combinate într-un cadru ierarhic: A* pentru planificare globală, RRT pentru ajustări locale, iar APF (Artificial Potential Fields) sau DWA pentru evitarea imediată a coliziunilor. Metodele reactive precum APF sunt eficiente computațional, dar pot suferi de blocaje în minime locale [32]. APF îmbunătățit [32] și Adaptive APF [33] adresează aceste probleme prin ponderări dinamice în funcție de bandă, viteză și geometria vehiculului.

1.2 Metode de planificare a traseelor bazate pe învățare automată

Metodele de învățare automată (ML) aduc o paradigmă diferită în planificarea traseelor, prin faptul că ele pot învăța strategii de conducere din date, fie prin supraveghere, fie prin întărire. Rețelele neuronale convoluționale (CNN) pot fi antrenate pe trasee reale, obținând politici care asociază observațiile senzorilor la acțiuni de direcționare [24]. Yang și Yao (2022) au demonstrat că CNN-urile pot genera traiectorii fluide care respectă dinamica vehiculului, dacă sunt antrenate corespunzător pe date relevante [24][25].

Învățarea prin consolidare (RL), în special în varianta sa profundă (DRL), permite vehiculului să învețe din interacțiuni cu medii simulate. Algoritmi precum DQN și DDPG au fost utilizați pentru a obține politici capabile să conducă în mod autonom în simulatoare precum CARLA [26][27]. Performanțele obținute se apropie adesea de cele ale unui șofer uman, iar eficiența în execuție este ridicată, deoarece inferența rețelei neuronale este rapidă.

Cu toate acestea, politicile învățate prin ML au o adaptabilitate limitată în afara distribuției de antrenament. Dacă vehiculul întâlnește o situație complet nouă, performanța poate scădea drastic. De aceea, multe implementări combină ML-ul cu module bazate pe reguli, care preiau controlul în caz de incertitudine [27]. De exemplu, o rețea neuronală poate sugera o traiectorie, dar un modul de siguranță determinist (precum APF sau DWA) validează dacă aceasta este fezabilă și sigură. Această arhitectură hibridă este tot mai răspândită în sistemele comerciale.

1.3 Abordări de planificare a traseelor bazate pe date LiDAR

Planificarea traseului folosind date brute de la senzori, în special LiDAR, reprezintă o alternativă robustă la abordările bazate pe hărți statice. Datele LiDAR oferă o reprezentare tridimensională precisă a mediului înconjurător, ceea ce permite evitarea obstacolelor și adaptarea traseului în timp real. Planificatori precum RRT sau Hybrid A* pot fi rulați direct pe mulțimea de puncte detectată de LiDAR, fără a necesita o mapare completă prealabilă [29][30].

Un exemplu relevant este oferit de Spanogiannopoulos et al. (2022), care propun un planificator non-holonomic bazat pe eșantionare și pe date LiDAR brute [29]. Acesta poate replanifica rapid traseul ori de câte ori norul de puncte se modifică, ceea ce îl face foarte potrivit pentru medii aglomerate și dinamice. De asemenea, Thoresen et al. (2021) arată că integrarea capacității de a fi parcurs a terenului în algoritmul Hybrid A* (THA*) conduce la soluții mai bune în aplicații off-road [30].

LiDAR-ul este util nu doar pentru detecția obstacolelor statice, ci și pentru urmărirea obiectelor în mișcare. Planificatori locali precum DWA, ghidați de hărți de ocupare actualizate cu LiDAR, pot evita în mod eficient coliziuni iminente [31]. Alte abordări, cum ar fi APF sau Velocity Obstacles, profită de precizia datelor LiDAR pentru a estima forțele de respingere și direcțiile sigure de deplasare. De exemplu, Hongyu et al. (2018) propun un APF adaptat care ia în calcul geometria vehiculului, banda de rulare și viteza [32].

În general, metodele reactive bazate pe LiDAR sunt rapide, precise și ideale pentru evitarea locală. Totuși, fără o planificare globală, ele pot cădea în minime locale. Din acest motiv, în practică, sunt adesea combinate cu planificatori globali (A*, RRT, ML etc.), rezultând o arhitectură ierarhică robustă și adaptabilă [33].

1.4 Planificarea traseelor bazată pe algoritmi genetici

Algoritmii genetici (GA) oferă un cadru evolutiv pentru generarea și optimizarea traseelor în funcție de criterii multiple, precum lungimea totală, siguranța, confortul și eficiența energetică. Fiecare traseu este codificat ca un cromozom, iar populația evoluează prin selecție, crossover și mutație. Această abordare este robustă în fața funcțiilor de cost nelineare și poate evita capcanele minime locale [34].

Receveur et al. (2020) demonstrează eficiența unei abordări GA-PF, în care un planificator global bazat pe GA este combinat cu un câmp potențial de tip fracționar pentru evitarea locală [34]. Această hibridizare combină optimalitatea traseului cu reactivitatea în fața obstacolelor, oferind traiectorii netede și sigure. Magdy et al. (2017) arată că GA-urile sunt mai eficiente decât metodele deterministe în medii complexe, dar că performanța lor online este limitată fără adaptări speciale [35].

Pentru a îmbunătăți această limitare, GA-urile sunt adesea combinate cu optimizări locale sau adaptate prin elitism și parametri dinamici. De asemenea, sunt posibile abordări multi-obiectiv, care oferă un set de trasee Pareto-optime. În practică, GA-urile sunt utile pentru optimizări offline sau pentru generarea de trasee inițiale, ulterior rafinate de planificatori locali rapizi (A*, DWA etc.).

1.5 Concluzii

Capitolul a analizat patru mari categorii de metode de planificare a traseelor pentru vehicule autonome: algoritmi clasici (pe grafuri sau eșantionare), metode bazate pe învățare automată, planificatori reactivi pe baza LiDAR și algoritmi genetici. Fiecare abordare are puncte tari și limitări specifice: metodele clasice sunt stabile și predictibile, dar greu de adaptat

în timp real; ML-ul oferă decizii rapide, dar generalizează slab în afara distribuției de antrenament; planificarea pe LiDAR este reactivă și precisă, dar locală; algoritmi genetici sunt buni pentru optimizări globale, dar au latență ridicată.

În practică, cele mai bune rezultate se obțin prin combinarea acestor metode într-o arhitectură ierarhică. De exemplu, un A* planifică traseul general, un modul ML ajustează deciziile în funcție de datele senzorilor, iar un planificator bazat pe LiDAR evită obstacolele locale. Literatura recentă indică o tendință clară spre astfel de soluții hibride [34][35]. Algoritmii genetici completează acest ecosistem oferind flexibilitate și capacitate de optimizare multi-criteriu. Prin urmare, progresul în planificarea traseelor autonome depinde tot mai mult de sinergia între paradigmele algoritmice disponibile.

Capitolul II: Contribuții privind propunerea unei abordări pentru evitarea coliziunilor în timp real

2.1 Algoritmul Safety Bubbles LiDAR

Vehiculele autonome care operează în medii reale trebuie să fie capabile să reacționeze la obstacole dinamice în mod prompt și fiabil. Majoritatea sistemelor actuale presupun existența unei hărți statice sau a unui model semantic al mediului, dar aceste cerințe vin cu costuri computaționale mari și cu o sensibilitate ridicată la erori de percepție. În acest context, se propune o nouă metodă de evitare a coliziunilor, centrată pe geometrie și pe date brute provenite de la senzorii LiDAR.

Abordarea presupune utilizarea unui concept denumit „bulă de siguranță”, o zonă tampon adaptivă generată în jurul vehiculului și al fiecărui obiect detectat în mișcare. Această bulă este proporțională cu viteza obiectului și anticipează spațiul în care acesta s-ar putea afla într-un orizont de timp scurt. Dacă bula vehiculului ego se intersectează cu o bulă externă, sistemul semnalează o coliziune iminentă. Această abordare nu necesită clasificarea obiectelor, predicție complexă a traiectoriilor sau construire de hărți, ci doar procesarea geometrică a norului de puncte, ceea ce o face simplă, rapidă și robustă.

Metoda este orientată spre planificarea locală și este destinată funcționării în cadrul unui sistem ierarhic de conducere autonomă. Ea poate funcționa ca un strat de siguranță între percepția de jos și planificarea de sus, completând funcționalitățile avansate de luare a deciziilor printr-un filtru geometric de risc imediat. Datorită designului său modular, poate fi integrată ușor cu alți algoritmi, fie reactivi, fie de planificare optimizată, fie bazați pe învățare, și poate acționa ca validator sau trigger pentru decizii urgente de evitare.

O altă contribuție esențială a metodei este simplitatea calculului. Testul de coliziune se reduce la o verificare de intersecție între două forme geometrice (elipse), iar întreg ciclul de procesare poate fi realizat în mai puțin de 30 ms, compatibil cu frecvența de scanare a celor mai comune senzori LiDAR de pe piață. În plus, sistemul nu are nevoie de GPU sau de modele de învățare costisitoare pentru a funcționa, ceea ce îl face ideal pentru platforme embedded sau pentru aplicații industriale cu resurse limitate.

De remarcat este și ideea de a menține contextul local ca prioritate, în loc să încerce să „înțeleagă” întreaga scenă, sistemul se concentrează pe vecinătatea imediată a vehiculului, unde coliziunile sunt iminente și deciziile trebuie luate rapid. Acest lucru contrastează cu metodele care încearcă să interpreteze întreaga hartă sau să prezică comportamente la distanțe mari, dar care, în practică, nu pot reacționa suficient de repede la evenimentele din imediata apropiere. Abordarea propusă se inspiră din filozofia minimalistă a controlului reactiv, dar adaugă un strat de rigoare geometrică și de adaptivitate contextuală, datorită modului în care bulele sunt adaptate dinamic.

Integrarea acestei logici în procesul de luare a deciziilor poate permite ca vehiculul autonom să evite coliziuni cu obstacole în mișcare, fie oprindu-se complet, fie alertând un modul de replanificare, fie permițând unui planificator reactiv (precum APF sau DWA) să gestioneze deplasarea într-o direcție sigură. Astfel, sistemul nu trebuie să înlocuiască alte metode, ci le completează, reducând riscul de erori fatale și crescând fiabilitatea întregii arhitecturi autonome.

2.2 Prezentare generală și arhitectura sistemului

Arhitectura generală a sistemului propus se bazează pe o secvență logică de procesare, pornind de la datele brute provenite de la senzorul LiDAR și terminând cu semnalul de avertizare a unei potențiale coliziuni. Această secvență este împărțită în mai multe module, fiecare cu o funcționalitate bine definită: achiziție de date, filtrare, compensarea mișcării, identificare de obiecte în mișcare, estimarea poziției și vitezei acestora, generarea bulelor de siguranță și verificarea eventualelor intersecții.

Primul pas constă în preluarea datelor de la senzorul 3D LiDAR, care generează o mulțime de puncte la fiecare 360° de rotație. Fiecare punct este raportat în coordonate relative față de vehiculul ego. Pe lângă aceste date, sunt disponibile și informațiile privind viteza curentă a vehiculului, care pot fi obținute fie de la un modul de odometrie, fie direct de pe magistrala CAN.

Filtrarea datelor se face pentru a reduce volumul de informație și pentru a păstra doar punctele relevante pentru evitarea coliziunilor. Sunt eliminate punctele prea joase (care corespund planului solului) și cele prea înalte (care aparțin infrastructurii, acoperișuri, semne etc.). În plus, se aplică o corecție geometrică de calibrare, astfel încât coordonata verticală să reflecte înălțimea reală față de sol. Prin această etapă, se extrag punctele care se află în același plan aproximativ cu vehiculul ego și care, prin urmare, sunt potențiale obstacole reale.

Ulterior, sistemul definește o regiune de interes (ROI) din jurul vehiculului. Această zonă nu este fixă, ci se ajustează în funcție de viteza de deplasare: cu cât vehiculul este mai rapid, cu atât ROI-ul se extinde mai mult în față. Această abordare este inspirată de distanța de frânare, un concept utilizat și în conducerea umană: un vehicul care rulează cu 10 m/s (36 km/h) are nevoie de aproximativ 20 m pentru a opri în siguranță. Prin extinderea proporțională a ROI-ului, se obține o protecție adaptivă care crește cu riscul cinetic.

Odată identificată regiunea de interes, sistemul procesează datele printr-un ciclu sincronizat cu frecvența de achiziție a senzorului. În fiecare ciclu, sunt aplicate transformări ale coordonatelor pentru a compensa mișcarea vehiculului, se identifică punctele dinamice, acestea sunt grupate și analizate, se calculează poziția și viteza fiecărui obiect, se construiesc bulele de siguranță, iar în final, se verifică dacă există intersecții cu bula vehiculului ego.

Dacă se detectează o intersecție, sistemul transmite o avertizare simplă: coliziune posibilă sau traseu sigur. Această decizie binară poate fi transmisă către modulele superioare de control pentru a declanșa frânarea sau replanificarea. Alternativ, ea poate fi folosită ca semnal pentru activarea unui planificator local mai sofisticat care să genereze o traiectorie nouă. În orice caz, arhitectura este astfel concepută încât să funcționeze în buclă închisă și să poată emite decizii în timp real, fără a depinde de o hartă statică sau de clasificări complexe.

Metoda propusă, deși simplă în principiu, este extrem de eficientă datorită modularității și a faptului că fiecare pas din fluxul de procesare este optimizat pentru execuție rapidă. Se evită astfel utilizarea unor componente costisitoare precum rețele neuronale, optimizatori MPC sau simulatoare interne, oferind o alternativă interpretabilă și sigură pentru evitarea coliziunilor în contextul conducătorilor autonomi.

2.2.1 Transformarea coordonatelor și compensarea mișcării

Pentru ca sistemul să poată detecta obiectele aflate în mișcare în mod robust, este necesar să se elimine efectul propriei mișcări a vehiculului ego asupra norului de puncte. Într-un mediu real, scanările LiDAR sunt efectuate în timp ce vehiculul se deplasează, iar acest lucru introduce modificări în poziția punctelor observate, chiar și atunci când acestea aparțin unor obstacole staționare. Pentru a face distincția între obiecte mobile și statice, trebuie aplicată o transformare geometrică asupra coordonatelor, astfel încât punctele statice să apară la aceeași poziție în scanări succesive.

Transformarea se face presupunând o deplasare liniară și constantă între două cadre. Se utilizează vectorul de viteză estimat al vehiculului și timpul Δt între două scanări pentru a calcula, pentru fiecare punct din trecut, unde ar fi trebuit să se afle în cadrul prezent dacă ar fi fost static. Această tehnică, de tip „background compensation”, este bine-cunoscută în viziunea robotică și permite izolarea mișcării relative a obiectelor. În implementarea actuală, s-a optat pentru o formulă simplă bazată pe scăderea deplasării estimate din coordonatele punctelor.

Rezultatul acestei operații este o hartă între două cadre consecutive, în care orice punct care nu respectă ipoteza de stabilitate în timp este considerat candidat pentru a aparține unui obiect mobil. Această abordare este robustă în fața zgomotului, eficientă computațional și foarte potrivită pentru aplicații în timp real. În plus, ea nu depinde de clasificarea semantică a punctelor și nici de o hartă preexistentă, făcând sistemul scalabil și portabil.

2.2.2 Gruparea și reprezentarea geometrică a obiectelor în mișcare

După identificarea punctelor dinamice, acestea trebuie grupate într-un mod care să reflecte prezența unor entități fizice individuale, cum ar fi pietoni, bicicliști sau alte vehicule. Pentru acest scop, se folosește o metodă de grupare bazată pe distanța dintre puncte. Pragul de distanță este stabilit în funcție de rezoluția senzorului și este calibrat empiric. Astfel, punctele care aparțin aceluiași obiect vor fi grupate împreună, iar cele care sunt izolate vor fi considerate zgomot sau ignorate.

Fiecare grup este apoi redus la o reprezentare geometrică simplificată. În loc de a păstra toate punctele componente, se calculează un punct median, care are avantajul de a fi robust la valori aberante. Acest punct median este utilizat ulterior pentru estimarea vitezei, prin compararea pozițiilor sale în timp. Viteza este derivată din ecuația clasică de mișcare uniformă, presupunând că între două scanări obiectul s-a deplasat liniar.

Această strategie de simplificare are două beneficii majore: (1) reduce dimensiunea datelor ce trebuie procesate la fiecare ciclu și (2) permite construirea unei „hărți dinamice” a obiectelor aflate în mișcare, fiecare reprezentat printr-o pereche poziție-viteză. Aceste date sunt suficiente pentru a construi bulele de siguranță în pasul următor și pentru a estima riscurile de coliziune în mod eficient.

2.2.3 Construirea dinamică a bulelor de siguranță

Pe baza poziției și vitezei estimate pentru fiecare obiect, se construiește o bulă de siguranță. Forma acesteia este eliptică, alungită în direcția de deplasare. Se presupune că obiectul își menține viteza constantă pe durata unei perioade de anticipare T , iar dimensiunile bulei sunt calculate în funcție de viteză și de coeficienții de siguranță (K_{fwd} și K_{back}). Bula acoperă astfel zona în care obiectul s-ar putea afla în viitorul apropiat, ținând cont de inerția sa.

Această abordare permite sistemului să „proiecteze” prezența obstacolelor în timp și spațiu fără a folosi predicții complicate sau modele de comportament. Este o strategie

geometrică directă, cu valoare practică ridicată. Pentru vehiculul ego, bula este extinsă mai mult în față și mai puțin în spate, în acord cu nevoile reale de frânare și evitare. Pentru alte obiecte, dimensiunile bulei reflectă viteza estimată, ceea ce duce la un model conservator, dar realist.

Din punct de vedere computațional, construirea bulelor este foarte eficientă, necesitând doar câteva înmulțiri și calcule trigonometrice. În plus, modelul este ușor de calibrat, putând fi ajustat în funcție de context (urban, autostradă, off-road) prin simpla modificare a coeficienților. Aceasta face sistemul versatil și aplicabil la o gamă largă de scenarii de conducere.

2.2.4 Verificarea coliziunilor și ajustarea traiectoriei

Odată generate bulele de siguranță, sistemul efectuează în mod periodic teste de intersecție între bula vehiculului ego și cele ale obstacolelor în mișcare. Acest pas este esențial pentru declanșarea avertizărilor de coliziune. Verificarea se face prin metode geometrice rapide, analizând dacă două elipse se suprapun pe baza pozițiilor și orientărilor acestora. Dacă se detectează o intersecție, se generează un semnal de avertizare, care poate fi transmis către modulele de control sau către stratul decizional al arhitecturii autonome.

Abordarea este discretă și conservatoare: sistemul nu încearcă să decidă cum să evite obstacolul, ci doar semnalează că traseul actual intersectează zona de risc. Acest design permite o separare clară între detecția coliziunii și luarea deciziei privind manevra de evitare. În aplicațiile curente, se presupune că sistemul reacționează prin frânare de urgență. Totuși, în cadrul unei arhitecturi avansate, avertizarea poate fi transmisă către un planificator local, care generează o traiectorie alternativă.

Această modularitate este un avantaj strategic: sistemul poate fi integrat cu orice alt tip de planificator reactiv sau bazat pe optimizare (de exemplu, DWA, APF, MPC), fără a necesita adaptări majore. Bula de siguranță funcționează ca un validator de siguranță în timp real, care poate filtra sau bloca traiectorii periculoase. Astfel, chiar dacă planificatorul superior propune o traiectorie optimă din punct de vedere al distanței sau timpului, aceasta nu va fi executată dacă există risc de coliziune, iar sistemul va solicita o replanificare.

2.3 Analiza și evaluarea performanței

Pentru a valida performanța metodei propuse, a fost utilizat simulatorul CARLA, care permite modelarea realistă a scenariilor urbane, a traficului și a comportamentului obstacolelor. Au fost concepute zece scenarii distincte, fiecare reprezentând o situație frecventă în trafic: treceri de pietoni, intersecții, apariții neașteptate de bicicliști sau vehicule. În toate cazurile, vehiculul ego urmează un traseu prestabilit, iar sistemul cu bule de siguranță are rolul de a detecta coliziunile iminente și de a genera avertizări.

Rezultatele au indicat o rată de detecție de 100%, fără coliziuni înregistrate. Avertizările au fost generate la timp, permițând vehiculului să reacționeze adecvat. În mod semnificativ, sistemul nu a generat alarme false, iar acuratețea detecției s-a menținut chiar și în condiții de vizibilitate redusă sau în prezența zgomotului LiDAR. Faptul că metoda se bazează exclusiv pe geometrie, și nu pe clasificare vizuală sau semantică, îi conferă robustețe față de ocultații parțiale, forme neregulate sau obiecte atipice (de exemplu, containere, obstacole temporare).

Un alt aspect evaluat a fost timpul de procesare per cadru. Pe un sistem standard CPU, timpul de execuție al fiecărui ciclu de analiză a fost sub 30 ms, permițând funcționarea confortabilă la 10 Hz, frecvența tipică a senzorilor LiDAR comerciali. Acest rezultat demonstrează fezabilitatea implementării metodei în sisteme embedded sau pe vehicule cu

putere de procesare limitată. În plus, metoda nu necesită GPU sau acceleratori specializați, ceea ce reduce costurile de integrare și crește portabilitatea sistemului.

2.3.1 Eficiența detecției coliziunilor

Evaluarea eficienței algoritmului a demonstrat o capacitate remarcabilă de detecție. În toate cele zece scenarii simulate, care au acoperit o varietate de situații reale, sistemul a emis avertizări exacte înainte ca vehiculul să ajungă în punctul de potențială coliziune. Acest rezultat arată că, deși metoda nu se bazează pe clasificare avansată sau predicție semantică, geometria locală este suficientă pentru a asigura evitarea coliziunilor în mod fiabil.

Un factor important este generalitatea metodei: sistemul nu face distincție între pietoni, bicicliști sau vehicule, toate sunt tratate ca obstacole dinamice prin analiza mulțimii de puncte. Astfel, sistemul este insensibil la erorile de clasificare sau la lipsa unei etichete semantice. De exemplu, un tomberon împins de vânt a fost tratat cu aceeași logică precum o mașină sau un pieton. Această neutralitate semantică este un atu în medii necontrolate, unde pot apărea obiecte imprevizibile.

Calitatea detecției a fost menținută și în condiții de vizibilitate scăzută, datorită naturii active a senzorului LiDAR. Spre deosebire de camerele video, performanța senzorului nu scade semnificativ în întuneric, ceață sau ploaie. Sistemul nu se bazează pe intensitate sau textură, ci doar pe geometrie, ceea ce îl face rezistent la multe dintre limitările viziunii clasice.

2.3.2 Performanța în timp real

Metoda a fost proiectată pentru a rula integral pe CPU, fără accelerare hardware. Aceasta este o cerință importantă în aplicații embedded sau în sisteme cu resurse limitate. Timpul de procesare măsurat per cadru a fost de aproximativ 25,30 ms, ceea ce permite funcționarea la 10 Hz, corespunzătoare frecvenței tipice a senzorilor 3D LiDAR. Această performanță a fost obținută prin limitarea regiunii de interes și prin evitarea oricărei forme de rețea neuronală sau optimizare globală.

Un alt aspect relevant este posibilitatea de aplicare a metodei la scară mai mare. Sistemul poate funcționa cu LiDAR-uri cu câmp vizual limitat (de exemplu, 120°), iar acuratețea detecției poate fi menținută prin adaptarea ROI-ului. Această flexibilitate permite implementarea metodei chiar și pe platforme mai ieftine, cu senzori ieftini, făcând-o atractivă pentru vehicule comerciale sau pentru sisteme de asistență avansată la conducere (ADAS).

Prin comparație, metodele bazate pe învățare profundă sau pe optimizare MPC necesită fie acceleratori specializați (GPU, TPU), fie timpi de procesare mai mari, care le fac mai puțin potrivite pentru reacții rapide. Metoda propusă oferă o alternativă echilibrată între simplitate și fiabilitate, demonstrând că evitarea coliziunilor nu necesită neapărat soluții sofisticate, ci poate fi atinsă printr-o analiză geometrică atent calibrată.

2.3.3 Adaptabilitatea în condiții diverse

Unul dintre testele importante pentru orice sistem de detecție este capacitatea sa de a funcționa în condiții de mediu variabile. În simulările efectuate, sistemul a fost expus la lumină scăzută, ploaie moderată și diverse unghiuri de apariție a obstacolelor. În toate cazurile, performanța algoritmului a rămas constantă. Spre deosebire de camerele video, care suferă semnificativ în condiții de iluminare slabă, senzorul LiDAR funcționează pe bază de impulsuri laser, fiind mai puțin influențat de factori externi.

Abordarea geometrică utilizată nu este afectată de variații ale texturii, culorii sau umbre. Prin urmare, obiectele cu forme neregulate, parțial ascunse sau cu margini difuze pot

fi detectate corect dacă au un comportament dinamic care le diferențiază de fundal. Această rezistență la ambiguitate este o caracteristică valoroasă, mai ales în mediile urbane aglomerate, unde formele sunt deseori incomplet vizibile sau fragmentate.

Testele au arătat că, inclusiv în cazul în care un obiect era parțial mascat de altul (de exemplu, un biciclist care trece prin spatele unei mașini parcate), sistemul era capabil să-l detecteze corect în momentul în care partea din obiect devenea vizibilă și intersecta bula vehiculului. Aceasta demonstrează nu doar adaptabilitate, ci și capacitate de reacție în fața aparițiilor bruște.

2.3.4 Comportamentul în scenarii reale de evitare

În cadrul simulării, sistemul a fost setat să reacționeze prin frânare imediată în caz de avertizare. Deși acest comportament este conservator, el este potrivit pentru scenariile testate, în care siguranța are prioritate față de continuitatea traseului. În cele mai multe cazuri, obstacolul părăsea zona de risc înainte ca vehiculul să oprească complet, ceea ce permitea reluarea traseului fără devieri majore.

În practică, acest comportament poate fi rafinat prin integrarea cu un modul de decizie care analizează contextul: dacă obstacolul este static sau mobil, dacă distanța este suficientă pentru o ocolire, sau dacă viteza vehiculului permite o manevră laterală. Însă, chiar și fără aceste rafinamente, metoda și-a demonstrat utilitatea ca strat de protecție în sistemele autonome, oferind o reacție robustă la pericole iminente.

Este de subliniat faptul că sistemul nu are nevoie să știe ce este obstacolul. El reacționează la mișcarea relativă și la geometria spațială. Această neutralitate decizională este un avantaj în fața sistemelor bazate pe clasificare, care pot rata obiecte neobișnuite sau greșit etichetate. Într-un mediu real, unde incertitudinea este regula, această robustețe este critică pentru menținerea siguranței operaționale.

2.3.5 Limitări

Deși metoda propusă oferă o serie de avantaje semnificative în ceea ce privește eficiența și robustețea, există și limitări care trebuie recunoscute și, acolo unde este posibil, atenuate. Una dintre cele mai evidente limitări este legată de dificultatea de a detecta obiecte care se deplasează foarte lent. Dacă mișcarea este aproape de pragul de detecție, diferențele dintre scanări pot fi considerate nesemnificative, iar obiectul poate fi clasificat eronat ca fiind static. Această problemă este accentuată în cazul pietonilor care stau pe loc, dar încep să se miște exact când vehiculul se apropie.

Pentru a rezolva această problemă, se propune utilizarea unui interval de comparație extins, adică în loc să se compare scanări succesive, sistemul să analizeze scanări separate de un interval mai mare (de exemplu, 300,500 ms). Astfel, mișcările mici acumulate în timp devin mai vizibile. Această metodă a fost explorată și în alte studii, precum cel al lui Yoon et al. (2018), unde s-a demonstrat că sensibilitatea crește semnificativ în fața obiectelor cu mișcare lentă fără a compromite semnificativ latența sistemului.

O a doua limitare apare în timpul manevrelor complexe ale vehiculului ego, cum ar fi virajele strânse sau frânările bruște. Modelul presupune o deplasare rectilinie și constantă între cadre, iar această ipoteză este încălcată în aceste cazuri. Acest lucru afectează compensarea mișcării și poate conduce la erori de clasificare (de exemplu, un obiect static pare să se miște sau invers). O soluție viabilă este integrarea unui model cinematic mai avansat, cum ar fi CTRA (Constant Turn Rate and Acceleration), susținut de date IMU (unitate inerțială). Astfel, mișcările curbe pot fi anticipate corect, iar transformările pot fi aplicate mai precis.

2.3.6 Concluzii

Rezultatele obținute în cadrul evaluării experimentale arată clar că metoda bulei de siguranță bazate pe LiDAR oferă un echilibru excelent între simplitatea implementării și eficiența practică. Ea nu necesită procesare semantică, nu presupune o hartă completă și nu cere modele de predicție a traiectoriei. În schimb, ea se bazează pe geometrie, viteză și reguli simple de intersecție, ceea ce o face potrivită pentru implementări robuste, în special pe platforme cu constrângeri de procesare.

Detecția coliziunilor este realizată cu o acuratețe remarcabilă, în timp util, și fără dependență de condițiile de lumină sau de starea vremii. Aceasta înseamnă că sistemul funcționează la fel de bine ziua sau noaptea, în ploaie, ceață sau în alte condiții care afectează în mod negativ sistemele bazate pe viziune. În plus, faptul că sistemul este neutru din punct de vedere semantic înseamnă că poate reacționa la orice obiect în mișcare, fie el vehicul, pieton, biciclist sau obstacol neașteptat.

Pe scurt, această metodă aduce în arhitectura sistemelor autonome un strat de siguranță geometrică, rapidă, predictibilă și ușor de verificat. Ea poate funcționa singură în scenarii simple sau poate fi integrată cu ușurință în sisteme mai complexe, oferind redundanță și protecție împotriva erorilor apărute în straturile superioare ale arhitecturii de planificare. În următoarea secțiune, vom analiza modul în care această metodă se compară cu alte soluții din literatura de specialitate.

2.4 Analiză comparativă cu metodele din literatură

Metoda bulei de siguranță se poziționează în mod interesant între planificarea reactivă și cea deliberativă. În comparație cu planificatorii clasici precum A* sau Hybrid A*, metoda noastră nu oferă o traiectorie optimizată, ci doar validează siguranța unei traiectorii deja existente. Acest lucru o face ideală ca modul de „fail-safe” într-un sistem mai larg, care poate detecta când o traiectorie devine periculoasă și poate interveni prin oprire sau replanificare. În acest sens, abordarea propusă este similară cu un sistem de frânare de urgență automatizată, dar cu avantajul că poate reacționa și în fața obstacolelor necunoscute, apărute recent în mediul din jur.

Comparativ cu metodele APF (Artificial Potential Field), metoda bulei este mult mai stabilă și mai previzibilă. APF suferă frecvent de probleme precum blocarea în minime locale sau oscilațiile în prezența mai multor obstacole apropiate. Metoda propusă evită aceste limitări prin natura sa discretă: intersecția este o condiție binară, fără ambiguitate. În loc să genereze un câmp continuu de forțe, sistemul decide simplu: coliziune sau nu, ceea ce permite implementarea unor politici de control mult mai clare și mai robuste.

Față de planificatorii de tip MPC (Model Predictive Control), metoda propusă este semnificativ mai rapidă și mai simplă. MPC necesită rezolvarea unei probleme de optimizare la fiecare pas, ceea ce poate fi greu de susținut în sisteme cu constrângeri de timp și de resurse. Deși MPC oferă traiectorii optime și respectă constrângeri dinamice, el nu este adecvat pentru reacții de ordinul milisecundelor. În schimb, metoda cu bule rulează la fiecare ciclu LiDAR, fără optimizare, oferind o alternativă practică pentru validarea siguranței în timp real.

Metodele bazate pe învățare automată, în special cele care utilizează învățarea prin întărire (RL), au fost de asemenea utilizate pentru evitarea coliziunilor. Rețelele neuronale antrenate pe date de simulare sau din lumea reală pot învăța politici care decid ce acțiuni trebuie luate în funcție de datele senzorilor. Totuși, aceste metode necesită seturi extinse de date pentru antrenare, sunt adesea opace (black-box) și au o capacitate redusă de generalizare la scenarii complet noi. De asemenea, ele nu oferă garanții formale de siguranță, ceea ce le face mai riscante în aplicații critice.

Metoda propusă se diferențiază de RL prin faptul că este complet interpretabilă și nu necesită niciun proces de antrenare. Este bazată pe reguli simple, dar eficiente, ceea ce permite integrarea sa în sisteme critice, unde transparența procesului și verificabilitatea sunt

cerințe esențiale. În plus, deoarece ea nu învață din date, nu suferă de problemele asociate overfitting-ului sau biasului de date.

Pe baza acestor comparații, putem concluziona că metoda bulei de siguranță nu concurează cu planificatorii avansați în ceea ce privește optimizarea traseului sau luarea deciziilor strategice, ci oferă o completare esențială în ceea ce privește siguranța operațională. Este o soluție ușor de implementat, eficientă computațional și foarte robustă, ceea ce o face potrivită pentru a fi inclusă în orice arhitectură de conducere autonomă ca strat de validare locală.

2.5 Concluzie

Capitolul a prezentat o contribuție originală în domeniul evitării coliziunilor pentru vehicule autonome, prin propunerea unei metode bazate pe bule de siguranță construite din date LiDAR brute. Spre deosebire de abordările clasice care necesită hărți semantice, clasificări complexe sau modele de predicție, metoda propusă se bazează exclusiv pe geometria scenei și mișcarea relativă, ceea ce o face foarte eficientă și robustă. Ea oferă un răspuns rapid și interpretabil la amenințările imediate de coliziune, funcționând independent de clasificare sau de contextul semantic al obiectelor.

Rezultatele obținute în simulare demonstrează că sistemul este capabil să detecteze coliziuni în mod fiabil, fără alarme false și fără întârzieri. El poate funcționa în timp real pe hardware standard, ceea ce îl face adecvat pentru implementări în medii industriale, prototipuri AV sau sisteme comerciale de tip ADAS. Mai mult, arhitectura modulară a metodei permite integrarea sa într-o gamă largă de sisteme, fie ca validator al traseului, fie ca modul de frânare de urgență sau ca declanșator pentru planificatori locali.

Direcțiile viitoare de cercetare includ integrarea metodei într-un sistem complet de planificare reactivă, extinderea la spații 3D pentru drone sau roboți mobili, validarea formală a siguranței pe baza modelului RSS, precum și adaptarea automată a parametrilor bulei prin învățare. De asemenea, se poate explora integrarea cu metode RL pentru a ghida comportamentul evaziv după declanșarea unei avertizări.

În concluzie, metoda propusă adresează un aspect esențial al conducerii autonome, siguranța în fața coliziunilor iminente, într-un mod simplu, robust și eficient. Ea nu înlocuiește planificatorii existenți, ci le adaugă o verigă esențială în lanțul decizional: o protecție geometrică predictibilă, capabilă să funcționeze acolo unde alte metode pot eșua din cauza complexității, lipsei de date sau erorilor de clasificare.

Capitolul III: CONTRIBUȚII PRIVIND INTEGRAREA DATELOR ÎN TIMP REAL ÎN ALGORITMI GENETICI PENTRU PLANIFICAREA TRASEELOR

3.1 Optimizarea dinamică și în timp real a traseelor

Algoritmii genetici (GA) s-au dovedit deosebit de eficienți în probleme complexe de rutare, cum ar fi TSP (Travelling Salesman Problem) sau VRP (Vehicle Routing Problem), datorită capacității lor de a gestiona obiective multiple și de a explora eficient spații mari de soluții [49, 50]. În contextul vehiculelor autonome sau al sistemelor logistice moderne, problema devine și mai complicată atunci când condițiile se schimbă în timp real. Traficul, vremea sau disponibilitatea unor destinații pot varia chiar în timpul execuției traseului, motiv pentru care planificarea statică devine rapid suboptimală.

Spre deosebire de metodele clasice care implică recalcularea completă a traseului (de exemplu, utilizarea Dijkstra după modificarea grafului), algoritmii genetici oferă flexibilitatea de a păstra o populație de soluții care poate fi adaptată gradual, pe măsură ce apar noi date [49]. Astfel, adaptarea la trafic sau condiții meteo se poate face fără repornirea procesului de optimizare, ci doar prin re-evaluarea funcției de fitness și, eventual, mutații direcționate.

Literatura existentă confirmă această abordare. Wang și Chen (2015) au demonstrat cum un GA pentru TSP dinamic poate menține soluții valide în fața variațiilor de trafic simulate [60]. Extensii ulterioare, precum cele propuse de Zhang et al. (2019), au integrat direct factori de trafic în fitness, obținând performanțe semnificativ mai bune decât versiunile statice ale aceluiași algoritm [51]. Planificarea dinamică cu GA devine astfel o alternativă solidă în fața incertitudinilor din rețelele de transport reale.

3.2 Contribuții

Planificatorul propus în cadrul acestei lucrări aduce o contribuție originală prin integrarea simultană a datelor de trafic și meteo în funcția de fitness a unui algoritm genetic. Spre deosebire de abordările anterioare, care folosesc date istorice sau modele statice, acest sistem se actualizează continuu, folosind API-uri precum Google Maps (pentru estimări de trafic) și OpenWeather (pentru condiții meteorologice actuale).

Această integrare nu presupune doar un acces pasiv la date, ci un mecanism activ de reevaluare a soluțiilor, pe măsură ce noi informații devin disponibile. Algoritmul poate reordona traseul, poate elimina opriri neferabile sau poate prefera trasee alternative în funcție de context. Astfel, chiar dacă scorul total poate fi uneori ușor mai mic (prin renunțarea la unele puncte valoroase, dar inaccesibile), traseul rezultat este mai robust și mai fezabil [54].

Simulările prezentate în lucrare vor demonstra această capacitate de reconfigurare în fața schimbărilor rapide, cu timp de reacție redus și menținerea unei rate ridicate de finalizare la timp. Rezultatele sunt susținute atât prin indicatori cantitativi (scor, timp total, penalizări), cât și prin exemple de scenarii reprezentative.

3.3 Structurarea algoritmului genetic

Algoritmul genetic propus are la bază o reprezentare adaptată pentru problema rutării cu constrângere de timp. Fiecare individ din populație reprezintă un traseu posibil, o secvență de puncte de interes, fiecare cu o durată de procesare și un scor asociat. Se urmărește maximizarea scorului total, sub constrângerea ca timpul total (inclusiv deplasarea și opririle) să nu depășească o limită impusă.

Funcția de fitness reflectă acest obiectiv: traseele fezabile sunt favorizate, în timp ce cele care depășesc timpul disponibil sunt penalizate sever. Prin includerea ponderii timpului (λ), se poate controla echilibrul dintre optimizarea scorului și menținerea eficienței temporale. Acest model este extensibil: scorul poate fi modificat în funcție de condiții meteo, iar timpul de deplasare poate reflecta starea traficului [49, 54].

Codificarea cromozomilor, operatorii de crossover și mutație, selecția prin turneu și introducerea elitismului sunt adaptate pentru a menține soluții valide, fără duplicări sau trasee neferabile. Evaluarea se face în timp real, prin acces la o matrice de distanțe și scoruri actualizată din surse externe.

3.3.1 Reprezentarea cromozomilor

Reprezentarea genetică a soluțiilor este crucială pentru succesul algoritmului. În această lucrare, fiecare cromozom codifică un traseu potențial ca o listă de puncte de interes, cu o lungime fixă, dar cu semnalizarea opririi traseului printr-un marker (de exemplu, 0). Astfel, se pot reprezenta trasee de dimensiuni variabile într-o structură uniformă, compatibilă cu operatorii genetici. Această alegere oferă flexibilitate și ușurință în implementarea încrucișării și a mutației, permițând totodată evaluarea rapidă a traseului în termeni de scor și timp total.

Unicitatea fiecărui punct este menținut printr-un mecanism de reparare care elimină duplicările introduse accidental prin încrucișare sau mutație. Soluțiile care includ același punct de mai multe ori sunt invalide, iar algoritmul aplică înlocuiri cu puncte neutilizate. Astfel, fiecare cromozom este coerent și semnificativ, reducând numărul de evaluări inutile în cadrul procesului evolutiv.

3.3.2 Generarea populației inițiale

Inițializarea populației este realizată prin permutarea aleatorie a punctelor disponibile, însoțită de o poziționare aleatorie a markerului de oprire, ceea ce determină o diversitate ridicată a dimensiunii traseelor. Această strategie permite algoritmului să înceapă explorarea de la soluții simple (cu puține puncte) până la trasee aproape complete, acoperind întregul spectru al posibilităților.

Datorită acestui grad de variație, GA-ul poate identifica rapid configurații promițătoare, favorizând indivizii care echilibrează numărul de opriri și timpul total. Diversitatea inițială este esențială pentru a preveni convergența prematură și pentru a permite descoperirea de soluții bune încă din primele generații.

3.3.3 Funcția fitness

Funcția de fitness joacă un rol esențial în ghidarea selecției. În această lucrare, ea este definită prin diferența dintre scorul total al traseului și o penalizare proporțională cu timpul de execuție. Dacă traseul depășește timpul disponibil, penalizarea este severă, fie fitness-ul devine zero, fie se scade o valoare mare din scorul acumulat.

Această formulare încurajează soluțiile fezabile, dar nu penalizează drastic traseele care utilizează eficient timpul disponibil. Mai mult, funcția este flexibilă: scorurile punctelor și timpii de deplasare pot fi modificați în funcție de datele în timp real. Astfel, funcția de fitness devine o expresie adaptivă a calității soluției, în raport cu contextul operațional curent.

3.3.4 Operatorul de selecție

Selecția prin turneu este utilizată pentru a păstra un echilibru între presiunea de selecție și diversitate. Alegerea aleatorie a unui subset de indivizi și selecția celui mai bun dintre ei încurajează performanța, fără a elimina complet indivizii mai slabi. Astfel, se evită stagnarea în optime locale și se promovează explorarea în paralel a mai multor regiuni din spațiul de soluții.

În implementare, un turneu de dimensiune 3 oferă un compromis bun: suficientă presiune pentru a direcționa evoluția către soluții mai bune, dar și destulă flexibilitate pentru a menține variația în populație. Selecția are loc de două ori pentru a forma perechile de părinți pentru încrucișare.

3.3.5 Operator de recombinare (crossover)

Pentru a genera trasee valide în cadrul algoritmului genetic, a fost proiectat un operator de crossover specializat, adaptat reprezentării cromozomilor ca permutări de destinații. Variantele standard, precum crossover-ul cu un singur punct, pot produce soluții invalide (cu duplicate sau lipsuri), motiv pentru care au fost testate mai multe abordări specifice problemelor de tip TSP, inclusiv Order Crossover (OX1) și Partially Mapped Crossover (PMX) [49, 52, 53].

Varianta adoptată este o adaptare a OX1, care funcționează astfel: se selectează un punct de crossover, iar copilul preia secvența de început de la primul părinte și o completează cu restul punctelor din al doilea părinte, menținând ordinea și evitând duplicarea. Rezultatul este o permutare validă ce păstrează structura parțială a ambilor părinți.

Poziția markerului de terminare (primul 0) este moștenită aleatoriu de la unul dintre părinți, păstrând astfel lungimea aproximativă a traseului. În unele cazuri, poziția acestui marker poate fi ajustată ușor pentru a introduce variabilitate suplimentară. Operatorul menține unicitatea punctelor și asigură o recombinare semnificativă a informației genetice, contribuind la explorarea eficientă a spațiului de soluții.

3.3.6 Operatorul de mutație

Pentru menținerea diversității genetice și prevenirea convergenței premature într-un optim local, algoritmul utilizează două tipuri de mutație: una prin schimb (intersimbarea a două poziții din traseu, excluzând markerul de terminare) și una prin amestecare (reordonarea aleatorie a unui subset mic din cromozom). Fiecare individ are o probabilitate p_m de a suferi mutație (ex. 20%).

Pentru a asigura validitatea soluțiilor, sistemul gestionează cazurile în care mutația ar introduce puncte duplicate sau ar modifica poziția markerului de terminare (care definește lungimea traseului). Mutația prin schimb este eficientă în rafinarea soluțiilor bune existente, în timp ce mutația prin amestecare favorizează explorarea unor regiuni noi ale spațiului de căutare.

3.3.7 Elitismul

În implementarea de bază, algoritmul nu include elitism. Totuși, în faza de testare, a fost introdus un mecanism de păstrare a celor mai bune soluții între generații. Mai exact, cei mai buni doi indivizi sunt selectați din reuniunea populației vechi și noi și inserați în noua generație, înlocuind soluțiile cu cel mai slab fitness [54].

Această strategie accelerează convergența și conservă performanțele maxime obținute, dar vine cu un risc: scăderea diversității genetice. Din acest motiv, aplicarea elitismului trebuie atent calibrată în funcție de natura problemei și obiectivele de echilibru între explorare și exploatare.

3.3.8 Criterii de evaluare

Algoritmul rulează până la îndeplinirea unui criteriu de oprire, care poate fi fie un număr fix de generații (ex. 1000), fie un criteriu de stagnare (ex. lipsa îmbunătățirii scorului maxim timp de 50 de generații). În testele efectuate, 1000 de generații pentru o populație de 50 de indivizi au fost suficiente pentru obținerea unei soluții stabile și competitive în cele mai multe scenarii.

În context dinamic, unde datele de intrare pot varia (trafic, vreme), algoritmul poate rula continuu, actualizând în fundal populația și menținând soluții fezabile în timp real. Acest comportament online îl face potrivit pentru aplicații autonome, unde adaptarea rapidă la condiții schimbătoare este esențială. Următoarea secțiune va explora extinderea algoritmului pentru a integra în mod direct aceste date dinamice.

3.4 Integrarea datelor de trafic și meteo în timp real

Unul dintre cele mai inovatoare elemente ale sistemului este integrarea sincronă și asincronă a datelor din mediu. La fiecare generație sau la intervale prestabilite, matricea de distanțe și scoruri este actualizată pe baza informațiilor despre trafic (Google Maps) și vreme (OpenWeather), ceea ce determină o reevaluare a funcției de fitness pentru fiecare individ.

În implementarea propusă, datele sunt preluate periodic și stocate într-o structură globală de tip cache, evitând astfel accesul excesiv la API-uri externe. Această strategie oferă un echilibru între actualitatea datelor și eficiența computațională, permițând sistemului să funcționeze în timp real, fără latențe semnificative.

Pe scurt, această arhitectură permite algoritmului genetic să devină „conștient de context”, ghidând procesul evolutiv în funcție de realitatea externă, și nu doar de o hartă statică. Astfel, planificarea traseului devine un proces viu, actualizat continuu în funcție de cele mai recente condiții, cu impact direct asupra traseului propus.

3.4.1 Integrarea datelor de trafic de la Google Maps

Integrarea datelor de trafic în timp real s-a realizat prin Google Maps Directions API, care oferă timpi de deplasare actualizați între două locații ținând cont de condițiile curente de trafic. În cadrul arhitecturii propuse, estimările de trafic sunt preluate periodic și inserate într-o matrice globală de timpuri de deplasare. Astfel, în loc ca fiecare individ să declanșeze un apel API (ceea ce ar fi costisitor și limitat de rata de interogare), evaluarea fitness-ului se face pe baza celei mai recente valori disponibile în cache.

Acest model permite adaptarea algoritmului la ambuteiaje, accidente sau lucrări rutiere, fără a întrerupe procesul de optimizare. Mai mult, traseele care includ segmente congestionate sunt penalizate automat, deoarece durata totală crește și, implicit, fitness-ul scade. Astfel, fără intervenție explicită, GA-ul tinde să elimine soluțiile afectate de trafic, selectând trasee mai eficiente.

Sistemul include și un mecanism de rezervă: dacă datele de trafic nu sunt disponibile temporar, se utilizează estimările medii din harta statică. Această redundanță permite continuitatea planificării chiar și în condiții de conectivitate redusă.

3.4.2 Integrarea datelor despre vreme de la OpenWeatherMap

Datele meteorologice influențează planificarea prin două canale: modificarea duratei de deplasare (prin reducerea vitezei din cauza condițiilor rutiere) și ajustarea valorii punctelor de interes (în special cele exterioare). În implementarea actuală, OpenWeather API este interogată pentru fiecare punct de interes, returnând condiții precum „Clear”, „Rain”, „Thunderstorm” etc., care sunt apoi asociate la factori de ajustare predefiniți.

Factorii meteorologici modifică atât scorurile locațiilor (scăzând valoarea punctelor exterioare în condiții de ploaie), cât și timpii de deplasare (prin multiplicarea duratei estimate cu un coeficient în funcție de severitatea vremii). Acești factori sunt integrați direct în funcția de fitness, ceea ce permite algoritmului să privilegieze trasee sigure, rapide și relevante.

Această integrare a condus la o comportare naturală a algoritmului: în caz de furtună, punctele în aer liber sunt automat evitate în favoarea locațiilor acoperite. Sistemul nu are nevoie de reguli explicite pentru a lua aceste decizii, ele apar în mod emergent, ca rezultat al selecției naturale din cadrul procesului evolutiv.

3.4.3 Calitatea datelor și limitări

Deși datele furnizate de API-urile externe sunt de înaltă calitate, ele nu sunt infailibile. Pot exista întârzieri în actualizare sau lipsă de granularitate în unele regiuni geografice. În plus, în cazul unor evenimente neprevăzute (accidente recente sau furtuni locale), API-urile pot răspunde cu întârziere.

Pentru a atenua aceste limitări, sistemul propus permite integrarea altor surse, precum date de la senzori locali sau feedback manual. Modulul de planificare rămâne neschimbat, întrucât datele externe sunt abstractizate într-o interfață unică de evaluare a scorului și a timpului. Astfel, adăugarea de noi surse de date nu necesită modificarea algoritmului, ci doar adaptarea modulului de achiziție.

Această arhitectură modulară este una dintre contribuțiile importante ale lucrării: separarea procesului decizional (algoritmul genetic) de sursa datelor (API-urile), permițând o extensibilitate ușoară și o adaptabilitate crescută în sisteme reale, care operează în contexte din ce în ce mai diverse și dinamice.

3.5 Simulare CARLA

3.5.1 Configurarea mediului

Pentru validarea sistemului într-un mediu controlat, s-a folosit simulatorul CARLA, care oferă o platformă realistă pentru testarea vehiculelor autonome. A fost aleasă harta Town03, datorită varietății geografice și infrastructurii diverse. Punctele de interes au fost marcate fizic prin obiecte vizibile în simulator (ex. conuri de circulație), fiecare având un scor și o durată de oprire.

Aceste puncte acoperă locații interioare și exterioare, permițând testarea efectelor vremii asupra alegerii traseului. Scenariul simulat a presupus un timp total disponibil de 120 de minute, iar algoritmul avea sarcina de a selecta și ordona un subset optim de opriri care să maximizeze scorul acumulat, fără a depăși timpul permis.

Simularea include și etichetarea semantică a punctelor în funcție de sensibilitatea lor la condiții meteo, ceea ce a permis observarea comportamentului adaptiv al algoritmului în situații reale, precum apariția bruscă a unei furtuni care afectează doar o parte a traseului.

3.5.2 Controlul vehiculului autonom

În timpul simulării, vehiculul autonom generat în CARLA a urmat traseele planificate de algoritmul genetic. Controlul vehiculului nu a fost realizat manual, ci prin API-urile oferite de simulator, care includ planificator global, manager de trafic și control de puncte de interes.

La fiecare oprire, vehiculul se oprea câteva secunde (simulând o acțiune precum coborârea pasagerului), timp în care sistemul putea decide dacă este necesară o replanificare, în funcție de condițiile actualizate. Această buclă permite o integrare realistă între planificare și execuție.

Traficul din simulare a fost generat automat, iar scenariile de congestie au fost create prin introducerea unor obstacole temporare sau prin reducerea vitezei altor vehicule. Acest cadru de testare a permis observarea în mod controlat a reacției algoritmului în fața evenimentelor dinamice, validate prin comportamentul efectiv al vehiculului în mediu.

3.5.3 Simulare condiții meteorologice

CARLA oferă posibilitatea de a simula o gamă largă de condiții meteorologice, de la cer senin până la ploi torențiale, vânt și furtuni. În cadrul experimentelor, aceste condiții au fost declanșate programatic la momente prestabilite pentru a testa capacitatea de adaptare a algoritmului genetic. De exemplu, după o perioadă inițială în care traseul se desfășura în condiții ideale, simulatorul a comutat brusc la o stare de vreme severă (Hard Rain Noon), cu efecte directe asupra aderenței vehiculului și asupra vizibilității.

Această tranziție a permis observarea impactului imediat asupra comportamentului algoritmului. În scenariile în care următoarea oprire planificată era un punct în aer liber, scorul acestuia era redus în evaluarea fitness-ului, ceea ce determina eliminarea sa din traseu în favoarea unui punct alternativ, mai potrivit pentru noile condiții. Astfel, planificatorul reacționa nu doar la eveniment, ci și anticipa eventuale probleme, ajustând traseul înainte ca obstacolul să devină critic.

Un alt aspect relevant a fost impactul vremii asupra fizicii vehiculului. În timpul ploii, distanța de frânare a crescut, iar accelerația s-a redus, ceea ce a dus la o creștere naturală a duratei segmentelor de traseu. Aceste modificări au fost preluate automat în calculul timpului estimat, influențând selecția punctelor următoare.

3.5.4 Rulare simulare

Interfața dintre algoritmul genetic și simulator a fost realizată printr-un modul Python care primea starea curentă a vehiculului și transmitea înapoi itinerariul optim. În mod implicit, algoritmul rula în fundal și actualiza continuu soluțiile, însă au fost testate și scenarii în care planificarea se relua doar la fiecare oprire.

S-a constatat că abordarea reactivă discretă (replanificare la oprire) era suficient de performantă pentru majoritatea scenariilor și evita reconfigurările excesive în timpul deplasării. De exemplu, dacă între două opriri intervenea o schimbare de trafic, dar agentul era deja în mișcare, algoritmul aștepta oprirea următoare pentru a actualiza traseul, evitând astfel comportamente instabile sau neintuitive.

Prin acest mecanism, GA-ul a funcționat ca un creier decizional care analizează datele de trafic și vreme, și ajustează secvențele de opriri în funcție de un context în schimbare. Această capacitate de adaptare incrementală este esențială pentru aplicațiile reale, unde replanificările bruște pot afecta negativ experiența utilizatorului sau performanța sistemului.

3.5.5 Criterii de performanță

Pentru a evalua eficiența planificatorului, au fost urmăriți o serie de indicatori cantitativi și calitativi. Numărul de puncte vizitate, scorul total acumulat, timpul total de deplasare și rata de finalizare la timp sunt cele mai importante măsuri cantitative. În paralel, a fost introdus și un scor calitativ de adaptabilitate, acordat în funcție de capacitatea algoritmului de a răspunde eficient la schimbări dinamice în mediul simulat.

Un alt aspect evaluat a fost timpul de calcul al soluției. Deși algoritmi genetici pot consuma resurse semnificative, în configurația propusă (cu caching inteligent și actualizări periodice), timpul de planificare a fost menținut sub 5 secunde pentru traseul inițial și sub 1 secundă pentru replanificările parțiale. Aceste valori sunt fezabile pentru sisteme în timp real.

Nu în ultimul rând, s-a urmărit și capacitatea sistemului de a face față unor scenarii neprevăzute. Spre exemplu, în cazul unei congestii apărute brusc între două opriri, a fost măsurat timpul necesar pentru ca GA-ul să ajusteze traseul astfel încât să evite segmentul afectat, fără a compromite obiectivul global al planificării.

3.5.6 Rezultate

Rezultatele experimentale au evidențiat superioritatea clară a algoritmului genetic cu integrare de date în timp real față de versiunile statice. În scenariile testate, GA-ul dinamic a reușit să finalizeze traseele planificate în peste 90% din cazuri, comparativ cu aproximativ 60% în cazul metodei statice. Deși uneori a fost necesară eliminarea unui punct cu scor mare, soluțiile produse au fost fezabile și robuste.

Într-un exemplu concret, un punct în aer liber cu scor ridicat a fost eliminat din traseu în urma declanșării unei furtuni, fiind înlocuit cu o oprire acoperită. Deși scorul total a scăzut cu 2,3 puncte, vehiculul a reușit să finalizeze traseul fără întârzieri. Dacă planul static ar fi fost urmat, vehiculul ar fi ajuns în condiții nefavorabile la punctul respectiv și ar fi fost nevoit să renunțe la următoarele opriri.

Aceste rezultate demonstrează că, într-un mediu dinamic, capacitatea de adaptare este mai importantă decât optimul teoretic. În plus, comportamentul emergent al algoritmului, evitarea punctuală a segmentelor afectate sau reordonarea traseului, confirmă eficiența procesului evolutiv în generarea deciziilor raționale și eficiente fără reguli explicite impuse.

3.5.7 Comparație între performanțele algoritmului îmbunătățit cu date real time

Tabelul comparativ inclus în lucrare evidențiază clar avantajele planificatorului genetic cu integrare de date în timp real. Față de varianta statică, acesta a reușit să îmbunătățească rata de finalizare la timp cu peste 30%, iar scorul total obținut a fost comparabil, chiar dacă uneori a implicat alegerea unor opriri mai puțin valoroase dar mai fezabile.

Introducerea elitismului a îmbunătățit stabilitatea soluției, în special în prezența fluctuațiilor de trafic. Reținerea celor mai buni indivizi a asigurat că soluțiile bune nu sunt pierdute din cauza mutațiilor sau încrucișării. De asemenea, varianta hibridă, care combină GA-ul cu optimizare locală de tip 2-opt, a reușit să rafineze suplimentar traseul, obținând scoruri ușor mai mari, în special în scenariile cu multe puncte apropiate ca valoare.

Un alt aspect relevant este scorul de adaptabilitate. Planificatorul cu date live a obținut un scor de 5/5 în acest sens, reflectând capacitatea sa de a ajusta traseul în fața schimbărilor de trafic sau vreme. Acest indicator este esențial în contextul vehiculelor autonome, unde rigiditatea planificării poate conduce la eșecuri critice.

3.5.8 Impactul modificării valorilor parametrilor

S-au efectuat și teste de sensibilitate asupra parametrilor algoritmului. S-a constatat că dimensiunea populației și numărul de generații influențează direct calitatea soluției, dar cu randamente descrescătoare. De exemplu, dublarea populației de la 50 la 100 de indivizi a adus o creștere a scorului de doar 0,5 puncte, dar a dublat timpul de execuție.

Rata de mutație a fost stabilă între 0.2-0.3, fără efecte majore asupra scorului final, dar o rată mai mare a favorizat explorarea în scenarii cu schimbări frecvente. GA-ul adaptiv, care ajustează rata de mutație în funcție de diversitate, nu a oferit îmbunătățiri semnificative, dar și-a păstrat comportamentul robust.

Un punct important a fost modul în care sistemul gestionează timpul de execuție. Cu actualizări periodice ale matricei de scoruri și timpuri, planificatorul a menținut un timp de execuție scăzut (1-2 secunde pentru replanificări). Acest rezultat confirmă că un GA bine calibrat poate fi utilizat eficient în aplicații reale, chiar și în lipsa unei infrastructuri de calcul avansate.

3.5.9 Analiza comparativă a performanțelor

Rezultatele obținute sunt în concordanță cu cele din literatura de specialitate. De exemplu, Basu et al. (2020) au arătat că folosirea datelor în timp real în planificarea transportului public a crescut semnificativ fiabilitatea soluțiilor [50], iar Zhang et al. (2019) au evidențiat superioritatea unui GA adaptiv în scenarii cu trafic variabil [51].

În testele proprii, algoritmul hibrid a oferit cele mai bune rezultate generale, combinând optimizarea globală a GA-ului cu rafinamentul local al unei căutări de tip hill-climbing. Această combinație este deosebit de eficientă în scenarii urbane dense, unde soluțiile bune sunt adesea apropiate ca scor, iar micile ajustări pot face diferența între o finalizare reușită și una ratată.

Aceste constatări susțin ideea că GA-urile îmbunătățite pot atinge performanțe aproape optime în medii dinamice, menținând totodată flexibilitatea structurală necesară pentru integrarea în sisteme autonome reale.

3.5.10 Discuție

Analiza confirmă că algoritmul genetic îmbunătățit este o soluție solidă pentru planificarea traseelor în contexte dinamice. Capacitatea sa de a integra simultan multiple surse de date, de a menține o populație diversă și de a reconfigura soluția în timp real îl transformă într-un instrument versatil pentru sisteme autonome.

Spre deosebire de planificatorii statici sau bazele de reguli, acest GA poate lua decizii adaptate fără codificare explicită. Comportamente precum renunțarea la un punct problematic sau schimbarea secvenței de opriri apar în mod natural, din selecția fitness-ului. Aceasta îl face compatibil cu scenarii complexe, imprevizibile, unde soluțiile prestabilite pot deveni rapid inadecvate.

Mai mult, sistemul este scalabil și extensibil. Poate integra alte surse de date (ex. senzori locali, priorități contextuale) și poate fi aplicat în alte domenii (ex. livrare urbană, rutare UAV). Structura modulară permite această extindere fără modificări majore ale nucleului evolutiv.

3.5.11 Sumar contribuții

Lucrarea prezintă o contribuție originală în integrarea datelor de trafic și meteo în timp real într-un algoritm genetic pentru planificarea traseelor. S-a demonstrat că, într-un context realist (simulat prin CARLA), algoritmul poate genera și ajusta trasee eficiente, robuste și fezabile, în fața schimbărilor de mediu.

Prin simulări și comparații, s-a arătat că planificatorul dinamic obține rezultate semnificativ mai bune decât versiunile statice, atât în ceea ce privește rata de finalizare, cât și adaptabilitatea. Sistemul a fost testat extensiv în condiții variate și a demonstrat comportamente inteligente emergente, fără reguli explicite.

Noutatea abordării constă în sinteza între evoluție (algoritmul genetic), percepție (datele externe) și reacție (replanificare incrementală). Acest model poate fi aplicat într-o varietate de aplicații autonome, marcând un pas important către planificarea adaptivă, conștientă de context, în lumea reală.

Capitolul IV: Utilizarea simulatorului CARLA pentru testarea algoritmilor

4.1 Caracteristici simulare cu CARLA

Simulatorul CARLA reprezintă una dintre cele mai avansate platforme open-source pentru testarea algoritmilor destinați vehiculelor autonome. Construit pe motorul grafic Unreal Engine, CARLA oferă o redare vizuală realistă, alături de o simulare fizică fidelă a dinamicii vehiculului, a interacțiunilor cu mediul și a comportamentului participanților la trafic. În plus, prin arhitectura sa client-server, cercetătorii pot controla mediul simulat folosind scripturi Python, care oferă un API bogat pentru manipularea actorilor (vehicule, pietoni), atașarea senzorilor, controlul traficului și al vremii.

Prin intermediul acestui API, utilizatorii pot crea rapid scenarii urbane complexe, pot popula hărțile cu vehicule autonome și pietoni, și pot modifica condițiile de mediu (ploaie, ceață, iluminare, oră din zi). Această flexibilitate transformă CARLA într-un „laborator virtual” unde scenarii de trafic periculoase, rare sau costisitoare pot fi reproduse în siguranță, cu posibilitatea de a le repeta exact pentru comparații între algoritmi. Simulatorul vine echipat cu mai multe hărți predefinite, printre care Town01-Town10, fiecare reprezentând un model diferit de oraș, și permite și încărcarea de hărți personalizate definite în format OpenDRIVE.

Un alt avantaj semnificativ al CARLA este realismul componentelor vehiculului: fiecare model de mașină disponibil include caracteristici fizice detaliate precum masa, forțele aplicate pe roți, caracteristicile de frânare și virare, oferind o platformă potrivită atât pentru testarea de algoritmi de control low-level, cât și pentru evaluarea comportamentului în scenarii de navigație urbană. În plus, CARLA simulează interacțiuni între participanții la trafic (de exemplu, un pieton care traversează strada în fața unui vehicul), ceea ce permite testarea capacităților sistemului de percepție și reacție al vehiculului autonom în fața unor provocări reale.

4.1.1 Generarea vehiculelor și pietonilor

Unul dintre fundamentele simulării este abilitatea de a popula mediul cu vehicule și pietoni. În CARLA, acest proces este realizat prin funcțiile `world.spawn_actor()` și `world.get_blueprint_library()`, care oferă acces la o bibliotecă extinsă de actori (vehicule de diverse tipuri, pietoni umani, senzori, semafoare etc.). Actorii pot fi plasați în poziții prestabilite pe hartă (`get_spawn_points()`) sau în locații aleatorii, iar blueprint-urile pot fi personalizate pentru a ajusta culoarea, numărul de pasageri, caracteristicile dinamice etc.

Pietonii, odată generați, pot fi controlați fie în mod direct, fie prin intermediul unui controller AI specific pietonilor - `carla.WalkerAIController`. Acest controller le permite să meargă în mod autonom către o locație definită, la o viteză stabilită, sau să se deplaseze liber în mediul urban, interacționând cu vehiculele aflate în mișcare. În cadrul cercetării proprii, acest mecanism a fost folosit pentru a popula harta cu pietoni aleatori, ce pot traversa strada sau merge în apropierea traseului vehiculului autonom.

4.1.2 Controlul mediului înconjurător: vreme și iluminat

Simulatorul oferă o serie de parametri meteo configurabili, centralizați în structura `carla.WeatherParameters`. Utilizatorul poate defini manual starea vremii (precipitații, ceață, acoperire cu nori, poziția soarelui etc.) sau poate utiliza presetări precum `ClearNoon`, `CloudySunset`, `HardRainNight` etc. Prin această flexibilitate, se pot reproduce scenarii realiste

pentru testarea robustă a sistemelor de percepție vizuală, cum ar fi condiții de iluminare redusă sau de vizibilitate scăzută.

Iluminatul artificial este de asemenea configurabil. Lămpile stradale pot fi accesate individual sau în grup și pot fi aprinse, stinse sau modificate în ceea ce privește culoarea sau intensitatea. De exemplu, se poate simula o pană de curent într-un cartier sau un scenariu de noapte cu iluminat parțial. Farurile vehiculelor pot fi activate prin carla.VehicleLightState, oferind control asupra luminilor de poziție, fazei scurte, fazei lungi, semnalizatoarelor etc.

Aceste funcționalități permit testarea în condiții realiste a algoritmilor care se bazează pe viziune, precum detecția pietonilor, marcajelor rutiere sau semnelor de circulație. Mai mult, schimbarea vremii în timpul simulării (de exemplu, din soare în ploaie torențială) permite evaluarea capacității de adaptare a sistemului, aspect esențial în dezvoltarea vehiculelor autonome sigure și stabile.

4.1.3 Senzori

O componentă esențială a simulării realiste în CARLA este reprezentată de suportul său extins pentru senzori virtuali. Aceștia simulează dispozitivele montate pe vehiculele autonome reale, generând date similare cu cele din lumea fizică. Printre senzorii disponibili se numără camere RGB (inclusiv camere stereo și de adâncime), senzori LiDAR 3D, radare, GPS, IMU, senzori de coliziune, senzori pentru invadarea benzii și multe altele.

Acești senzori pot fi atașați rigid de vehiculul simulat, în poziții și orientări specificate de utilizator, folosind transformări relative. De exemplu, o cameră poate fi poziționată la 0.8 metri în fața centrului vehiculului și 1.7 metri deasupra solului, simulând camera frontală a unei mașini reale. Odată atașat, senzorul transmite în mod asincron datele sale către un callback, unde pot fi procesate sau salvate.

Pentru un senzor de tip LiDAR, datele pot include până la 1 milion de puncte 3D pe secundă, ceea ce permite construirea de hărți de ocupare, detectarea obstacolelor sau urmărirea pietonilor. Camerele pot transmite cadre cu rezoluții configurabile, utile pentru antrenarea și testarea rețelelor neuronale convoluționale. În cercetarea proprie, această componentă a fost folosită intens pentru a testa algoritmi de evitare a coliziunilor, detecție de obstacole și pentru colectarea de date pentru antrenarea unor modele de învățare automată.

4.1.4 Navigație și controlul vehiculelor

Controlul vehiculului poate fi realizat prin două metode: direct, prin comenzi de tip throttle, brake, steer, sau indirect, prin intermediul planificării și urmăririi unui traseu prestabilit. În primul caz, programatorul transmite periodic comenzi de control prin carla.VehicleControl, controlând în mod detaliat comportamentul vehiculului. Acest lucru este util pentru testarea unor algoritmi proprii de control (ex. PID sau MPC).

În cazul navigației autonome, CARLA oferă suport pentru planificarea traseelor folosind harta rutieră a orașului. Planificatorul global (GlobalRoutePlanner) generează o succesiune de puncte de interes între două locații geografice, urmând topologia drumului. Aceste puncte sunt apoi urmărite fie de un agent autonom (ex. BasicAgent, BehaviorAgent), fie de un sistem personalizat de control dezvoltat de cercetător.

Această separare între planificare și control permite testarea eficienței rutelor, dar și a comportamentului vehiculului la intersecții, semafoare, în trafic mixt sau în prezența unor obstacole neprevăzute. Această funcționalitate a fost utilizată extensiv în cadrul testării algoritmului genetic dezvoltat în cercetarea proprie, unde vehiculul simulat urma un traseu optim calculat dinamic în funcție de condițiile de trafic și vreme.

4.1.5 Traffic Manager și rutare automată

Traffic Manager-ul (TM) din CARLA este responsabil cu comportamentul vehiculelor în modul autopilot. TM permite configurarea comportamentului pentru fiecare vehicul în parte: se pot seta viteze maxime, distanțe de siguranță, probabilitatea de a schimba banda, comportamentul la semafoare, precum și toleranța la coliziuni.

De asemenea, TM poate controla simultan sute de vehicule, oferind un fundal realist pentru scenariile de testare. În implementarea proprie, TM a fost utilizat pentru a popula orașul virtual cu trafic realist, împotriva căruia vehiculul controlat de algoritmul genetic a trebuit să navigheze. S-a observat că TM este suficient de eficient pentru testele de planificare de traiectorii și evitarea coliziunilor, dar pentru controlul fin al unui vehicul, un agent personalizat sau un sistem de control manual rămâne preferabil.

Mai mult, CARLA permite integrarea planificatorului global de rute cu TM, astfel încât vehiculul să poată urma un traseu stabilit anterior, sub supravegherea Traffic Manager-ului. Astfel, scenariile complexe (ex. ocolirea unei intersecții aglomerate sau redirectionarea în cazul unui blocaj) pot fi reproduse și testate fără modificări semnificative în codul simulatorului.

4.2 Utilizarea CARLA în literatura de specialitate

CARLA a devenit în scurt timp un standard în cercetarea academică și industrială dedicată vehiculelor autonome. Numeroase lucrări științifice folosesc simulatorul pentru testarea unor algoritmi complecși, fie în domeniul percepției, fie în cel al planificării sau controlului. Prin realismul oferit, CARLA permite simularea de scenarii periculoase sau rare, care ar fi dificil sau costisitor de reprodus în lumea reală.

De exemplu, în lucrarea lui Codevilla et al. (2018) [63], CARLA a fost folosit pentru antrenarea unui sistem de conducere autonomă end-to-end bazat pe rețele neuronale convoluționale. Vehiculul virtual a învățat să ia decizii de direcționare (stânga/dreapta) în intersecții doar pe baza imaginilor RGB și a comenzilor furnizate de un șofer uman în timpul antrenamentului. Rezultatele au demonstrat că simulatorul oferă o platformă solidă pentru învățare prin imitație.

Într-o altă lucrare, Chen et al. (2020) au introdus tehnica „Learning by Cheating” [64], în care agentul autonom este antrenat folosind informații privilegiate din mediu (de exemplu, hărți perfecte sau poziții exacte), dar apoi testat în condiții reale, fără acces la aceste date. Astfel, CARLA a fost folosit pentru a transfera politici robuste în scenarii realiste, evidențiind capacitatea sa de a susține tranziția de la simulare la aplicare practică.

Mai mult, studiul de sinteză realizat de Kiran et al. (2021) [65] arată că majoritatea cercetărilor în învățarea prin întărire aplicată la condus autonom au avut loc inițial în CARLA. Fie că este vorba de rețele DQN, PPO, SAC sau alte politici actor-critic, cercetătorii folosesc simulatorul pentru a testa capacitatea agenților de a învăța să conducă, să frâneze, să ocolească obstacole sau să se oprească în fața pietonilor.

Totodată, CARLA este valoros și pentru generarea de date sintetice adnotate, utile în antrenarea rețelelor neuronale. Imagistica RGB, segmentarea semantică, datele de adâncime și norii de puncte LiDAR pot fi exportate cu etichete perfecte, mult mai eficient decât adnotarea manuală a datelor reale. Studii precum cele ale lui Banerjee et al. (2018) și Caltagirone et al. (2019) au demonstrat că datele simulate pot completa cu succes seturile de date reale, crescând acuratețea în scenarii dificile precum vreme severă sau iluminare redusă.

CARLA a fost folosit și în cercetări care implică evitarea coliziunilor. Lee et al. (2020) au testat un algoritm geometric pentru detecția obstacolelor dinamice în intersecții aglomerate. De asemenea, autorul prezentei lucrări a propus și testat în CARLA un algoritm bazat pe „safety bubbles”, care detectează riscul de coliziune între vehiculul ego și alți participanți la trafic prin analiza intersecției unor zone de siguranță dinamice. Rezultatele obținute au confirmat eficiența metodei în simulare.

Nu în ultimul rând, simulatorul este frecvent utilizat în combinație cu SUMO (Simulation of Urban Mobility), pentru cercetări în domeniul traficului urban și al managementului infrastructurii inteligente. De exemplu, Azfar și Ke (2024) [67] au integrat CARLA cu SUMO într-un sistem de control adaptiv al semafoarelor bazat pe învățare multi-agent, demonstrând o reducere a congestiei în intersecții urbane complexe.

4.3 Utilizarea CARLA în cercetarea proprie

În cadrul acestei lucrări, simulatorul CARLA a fost utilizat ca platformă principală de testare pentru două direcții de cercetare: (1) algoritmul de evitare a coliziunilor bazat pe „bule de siguranță”, și (2) algoritmul genetic pentru planificarea traseelor în funcție de date în timp real.

Pentru prima contribuție, s-a simulat un vehicul autonom echipat cu senzor LiDAR și un algoritm geometric care calculează în permanență o zonă de protecție în jurul vehiculului. Dacă această zonă se intersectează cu o zonă similară aparținând unui obstacol (alt vehicul sau pieton), se emite un semnal de avertizare. Simulările au demonstrat că sistemul detectează coliziuni iminente și permite reacții rapide, fără a necesita recunoașterea obiectelor, ci doar pe baza geometriei punctelor din norul LiDAR. CARLA a permis configurarea acestor experimente într-un oraș populat cu pietoni mobili, vehicule automate și condiții meteo variate, confirmând robustețea algoritmului.

A doua direcție de cercetare a vizat dezvoltarea unui algoritm genetic capabil să planifice trasee turistice într-un oraș, adaptându-se în funcție de condițiile de trafic și vreme în timp real. Deși inițial validarea s-a realizat prin simulări offline, simulatorul CARLA a fost folosit pentru testarea practică. După generarea unui traseu optim, acesta a fost urmat în CARLA de un vehicul controlat prin autopilot, iar în timpul execuției, schimbările în trafic (simulate prin TM) sau condițiile meteo (simulând ploaie sau furtună) au determinat replanificări succesive.

Integrarea dintre algoritmul genetic și CARLA a fost realizată prin scripturi Python care comunicau bidirecțional: simulatorul transmitea starea vehiculului, iar planificatorul decidea dacă traseul actual mai este fezabil. Acest mecanism a permis demonstrarea unui comportament end-to-end complet: de la planificare inteligentă până la execuție în mediu virtual realist.

Astfel, simulatorul CARLA a fost un instrument esențial în validarea contribuțiilor prezentate în această teză. El a permis observarea comportamentului sistemelor autonome într-un mediu controlat, dar suficient de realist încât să surprindă problemele reale ce pot apărea în traficul urban. Validarea în CARLA a fost o etapă-cheie înaintea unei posibile treceri la testarea pe vehicule fizice sau în medii urbane reale.

CAPITOLUL V: PROPUNERI DE IMPLEMENTARE

5.1 Ghid turistic autonom

Ideea unui vehicul autonom care acționează ca un ghid turistic pornește de la integrarea planificării inteligente a traseului cu sisteme avansate de percepție și control. Scopul nu e doar transportul pasagerilor, ci maximizarea experienței lor: vizitarea eficientă a celor mai relevante puncte de interes în timpul disponibil. Cum problema ordonării multiplelor opriri într-un timp limitat este NP-hard [68], soluția practică vine din zona euristiciilor - în cazul de față, algoritmi genetici (GA), care pot furniza rapid soluții bune [71]. O astfel de abordare permite și adaptarea dinamică a traseului în funcție de trafic sau condiții meteo.

5.1.1 Arhitectura hardware

Pentru ca un astfel de sistem să funcționeze, vehiculul trebuie dotat cu un hardware robust: procesare, senzori și control precis.

- Unitatea de procesare este nucleul decizional. În mod tipic, avem un sistem echivalent unui server performant (CPU multi-core, GPU dedicat), montat on-board. Acesta rulează toate modulele: percepție, localizare, planificare și control [74]. Platforme precum NVIDIA Drive PX sau Orin sunt deja standarde în industrie.
- Actuatorele electronice (volan, accelerație, frână) sunt controlate printr-o interfață de tip drive-by-wire. Acest tip de arhitectură elimină legăturile mecanice directe, făcând ca vehiculul să răspundă rapid la comenzile generate software. În cazul nostru, presupunem un vehicul autonom de nivel 5 (automatizare completă), fără intervenție umană.
- Senzorii (LiDAR, camere, GPS etc.) sunt montați strategic pe vehicul, conectați la unitatea de procesare și calibrați pentru fuziune. Informațiile trebuie să fie precise și sincronizate, iar datele colectate sunt esențiale pentru percepție și localizare. Interfețele de comunicare (Ethernet, CAN) și sincronizarea temporală sunt critice pentru corelarea fluxurilor [72].

5.1.2 Arhitectura senzorială

Pentru a percepe corect mediul și a se localiza, vehiculul are nevoie de un set bine calibrat de senzori complementari. Sistemul e compus din senzori exteroceptivi (pentru percepția mediului extern) și interoceptivi (pentru mișcarea proprie). Împreună, furnizează toate datele necesare planificării sigure.

- LiDAR-ul este baza percepției tridimensionale. Produce o mulțime de puncte 3D precisă, indiferent de lumină, ceea ce îl face indispensabil în mediul urban. Este folosit atât pentru detecția obstacolelor, cât și pentru generarea hărților sau pentru evitarea coliziunilor [74, 75].
- Camerele RGB adaugă componenta semantică: recunosc semne, semafoare, marcaje sau pietoni. Sistemele moderne folosesc camere multiple, unele cu vedere panoramică, altele pentru adâncime (stereo). Deși mai sensibile la lumină sau vreme, oferă detalii imposibil de surprins cu LiDAR-ul [75].
- Radarul completează percepția în zonele unde LiDAR sau camerele pot avea probleme (ploaie, ceață). Nu e la fel de precis în formă sau poziționare, dar excelează în detecția vitezei relative, fiind util pentru frânări de urgență sau urmărirea vehiculelor din față [74].
- GNSS + IMU oferă localizarea absolută (GPS) și mișcarea relativă (accelerometru + giroscop). În mediul urban, unde semnalul GPS poate fi slab, IMU ajută la continuitate, iar prin fuziune (ex: filtre Kalman), poziția vehiculului se păstrează precisă [72, 73].
- Senzori suplimentari, precum cei ultrasonici pentru parcare sau camere termice pentru detecție nocturnă, pot fi adăugați. Mai mult, datele din cloud (trafic, vreme) completează imaginea, fiind tratate tot ca input senzorial indirect - extrem de valoroase pentru planificare adaptivă [71, 74].

Acest mix de senzori permite vehiculului să „vadă”, „simtă” și „știe” unde este, în orice condiții. Cheia este fuziunea acestor date într-un model coerent și actualizat constant, care alimentează modulele de decizie.

5.1.3 Arhitectura software

Software-ul care rulează pe vehicul este împărțit modular, după logica „percep - planific - execută”, cu o comunicare clară între componente. Stack-ul poate fi construit pe middleware-uri precum ROS2 sau alte sisteme distribuite în timp real, dar structura logică este similară indiferent de tehnologie [69, 70].

- Percepția procesează fluxurile de la senzori și construiește o reprezentare a scenei: unde sunt obstacolele, ce tipuri de obiecte sunt (mașini, pietoni, biciclete), unde este spațiul liber de rulare. Această reprezentare este critică pentru planificare și trebuie actualizată constant. Se folosesc rețele neurale convoluționale pentru imagini (dectecție pietoni, semne, semafoare), clustering pentru LiDAR și fuziune între ele [75].
- Localizarea combină datele GNSS, IMU și, dacă e cazul, potrivirea cu hărți HD (ex. puncte LiDAR referință). Scopul este obținerea unei poziții precise în raport cu harta orașului. Algoritmii de localizare (de obicei filtre Kalman extinse sau SLAM) oferă poziția și orientarea vehiculului în fiecare moment [72, 73].
- Planificarea se împarte în două nivele: globală și locală (detaliată în subsecțiunile următoare). Cea globală stabilește ordinea destinațiilor în itinerar, iar cea locală generează traiectorii detaliate pe care vehiculul le poate executa imediat. Arhitectura modulară permite refacerea traseului oricând apare un obstacol sau schimbare de condiții [71].
- Controlul preia traiectoria locală și o transformă în comenzi concrete: cât de mult să accelereze, ce unghi să vireze, cât să frâneze. Se folosește de regulă control PID pentru stabilitate și MPC (Model Predictive Control) pentru luarea în calcul a dinamicii vehiculului [69].

Această arhitectură este gândită astfel încât fiecare modul să poată fi dezvoltat, testat și optimizat separat. În practică, separarea clară între percepție, planificare și control face întreg sistemul mai ușor de întreținut și extins.

5.1.4 Planificare globală

Planificarea globală stabilește ordinea optimă a punctelor de interes turistice, în funcție de timpul disponibil, scorurile obiectivelor și distanțele dintre ele. Practic, vehiculul trebuie să decidă „unde merge și în ce ordine”, ținând cont de toate constrângerile. Problema este echivalentă cu un TSP extins, pentru care folosim algoritmi genetici (GA) - o alegere naturală când optimul exact nu e practic calculabil [68, 71].

Fiecare soluție candidat (cromozom) e o listă ordonată de obiective, evaluată după:

- scor turistic total (suma ratingurilor);
- timpul de vizitare și deplasare totală;
- respectarea timpului disponibil (soluțiile care îl depășesc sunt penalizate puternic).

Pentru a evalua corect durata deplasărilor, algoritmul nu folosește valori fixe, ci timpi ajustabili în funcție de condițiile de trafic (de exemplu, apeluri către Google Maps API sau o matrice locală de timpuri adaptate) [71]. Acest lucru face planificatorul adaptiv - dacă traficul devine aglomerat pe un segment, fitness-ul scade și soluțiile cu rute alternative devin preferate.

De asemenea, se pot introduce ajustări în funcție de vreme: dacă plouă, obiectivele în aer liber pot fi penalizate în scor. Totul e integrat direct în evaluarea fitness-ului, iar GA găsește rapid noi configurații care răspund acestor schimbări.

La final, planificatorul returnează o listă ordonată de destinații, ce urmează a fi parcurse în ordine. Pentru fiecare tranziție, o componentă separată calculează rutele efective

de condus (cu A*, Dijkstra sau folosind un serviciu extern), pentru ca planificarea locală să poată genera traiectorii executabile.

GA-ul rulează rar, de obicei o dată la începutul călătoriei sau când este necesară replanificarea. Avantajul este că poate reevalua rapid traseul în funcție de noile condiții - o abilitate esențială pentru funcționare robustă.

5.1.5 Planificare locală

Planificarea locală traduce traseul global într-o traiectorie executabilă în timp real, adică decide „cum merge” vehiculul pe următorul segment de drum. Ea lucrează pe orizont scurt (câțiva metri sau câteva secunde înainte), cu obiectivul de a genera o cale fezabilă, sigură și confortabilă, care respectă toate constrângerile locale: benzi, curbe, obstacole, semne de circulație [69].

Intrările planificatorului local includ:

- harta drumului (cu poziția curentă și ținta pe termen scurt),
- obstacolele detectate în jur (din percepție),
- starea vehiculului (viteză, poziție),
- regula de condus (de exemplu, să se încadreze pe banda corectă sau

să oprească la zebră).

Planificatorul poate genera mai multe traiectorii candidate (curbe 2D sau 3D în timp) și le evaluează după:

- siguranță (distanță față de obstacole),
- confort (curbe line, fără accelerații bruște),
- fezabilitate (nu cere manevre imposibile fizic),
- deviere minimă față de traseul global.

Apoi, alege cea mai potrivită și o trimite modulului de control.

Metodele folosite pot fi optimizatoare (ex. MPC), bazate pe eșantionare (ex. RRT) sau învățate (rețele neurale care prezic direct traiectorii). Indiferent de abordare, rolul este același: să transforme o decizie „unde să merg” într-o cale detaliată și executabilă.

Dacă apare un obstacol neașteptat sau se schimbă situația (ex. un pieton intră pe drum), planificarea locală poate genera rapid o nouă traiectorie (ex. pentru frânare). Dacă obstacolul face imposibilă continuarea traseului actual, planificatorul semnalează către modulul global că e nevoie de reconfigurare a itinerariului [71].

Este esențial ca planificarea locală să opereze în timp real - zeci de Hz - pentru a menține siguranța, și să coopereze strâns cu modulul de control.

5.1.6 Reconfigurare dinamică pe baza condițiilor în timp real

Una dintre cele mai valoroase funcționalități ale acestui sistem este capacitatea de replanificare dinamică. Vehiculul nu urmează orbește un traseu stabilit la început, ci își adaptează în mod inteligent itinerariul în funcție de trafic, vreme sau alți factori neprevăzuți.

Replanificarea poate fi declanșată în mai multe situații:

- Ambuteiaje sau blocaje detectate fie prin serviciile online (ex. Google Maps API), fie prin senzorii proprii;
- Evenimente meteo - de exemplu, ploaie torențială ce face anumite obiective în aer liber neatractive;
- Preferințe noi ale utilizatorului (ex. vrea să schimbe ordinea opririlor sau renunță la o oprire);
- Întârzieri acumulate, ce fac imposibilă vizitarea tuturor obiectivelor în timpul rămas.

- În aceste cazuri, algoritmul genetic este relansat, dar nu de la zero. Pornește de la starea curentă (poziția vehiculului, timpul rămas, opririle deja făcute) și re-optimizează traseul pentru restul drumului. Acest lucru poate însemna:
- schimbarea ordinii obiectivelor,
- eliminarea celor cu scor mai mic,
- selecția unei rute mai rapide sau mai sigură.

Pentru turist, această flexibilitate înseamnă o experiență care rămâne plăcută și eficientă chiar dacă apar probleme - sistemul se adaptează fără a cere intervenție manuală. Reconfigurarea se face în fundal, iar tranziția către noul plan este lină. Vehiculul continuă pe traiectoria curentă și schimbă direcția doar atunci când e sigur și logic să o facă.

Acest mecanism este exact ce face diferența între o planificare rigidă și un sistem autonom cu adevărat „inteligent” și reactiv, capabil să opereze în condiții reale [71].

5.1.7 Concluzie

Capacitatea de a combina planificarea strategică cu adaptarea în timp real este ceea ce face dintr-un simplu algoritm genetic o componentă esențială într-un sistem complet de vehicul autonom. În acest capitol am detaliat modul în care un astfel de sistem poate funcționa în practică: de la hardware-ul robust, senzorii diversificați, până la software-ul modular care le pune în mișcare.

Arhitectura propusă nu doar că oferă siguranță și eficiență, ci și o experiență personalizată și flexibilă - esențială într-un scenariu urban turistic. Un astfel de vehicul nu doar „transportă”, ci decide inteligent unde să meargă, când să meargă și cum să adapteze traseul la realitatea din teren, ținând cont de preferințele utilizatorului și de contextul ambiental.

Combinarea dintre algoritmul genetic și percepția multi-senzor oferă o soluție robustă, scalabilă și realistă, care poate fi extinsă cu ușurință în alte domenii din mobilitatea urbană. De la livrare autonomă la transport public flexibil, principiile rămân aceleași: planificare adaptivă, siguranță, și operare în timp real.

În esență, acest sistem nu doar planifică un drum, ci construiește o experiență de navigație inteligentă - un pas important către integrarea firească a vehiculelor autonome în orașele viitorului.

Capitolul VI: Concluzii și direcții viitoare

Lucrarea de față a urmărit dezvoltarea unei soluții integrate pentru conducerea autonomă urbană, concentrându-se pe două componente esențiale: planificarea adaptivă a traseului și evitarea coliziunilor. Au fost definite și urmărite patru obiective majore: analiza critică a metodelor existente, propunerea unei soluții geometrice pentru detecția obstacolelor, proiectarea unui algoritm genetic capabil să planifice rute în timp real, și validarea funcțională a întregului sistem într-un mediu de simulare realist.

Pe partea de siguranță, contribuția constă în proiectarea unui algoritm simplu și eficient, bazat pe date LiDAR brute, care construiește în jurul vehiculului și al obstacolelor dinamice zone de siguranță adaptive („bule de siguranță”). Atunci când aceste zone se intersectează, sistemul declanșează măsuri de protecție fără a fi necesară clasificarea obiectelor. Algoritmul funcționează în timp real și a fost validat cu succes în scenarii urbane dinamice, în simulatorul CARLA.

În ceea ce privește planificarea globală, a fost dezvoltat un algoritm genetic care generează itinerarii optimizate în funcție de un set de puncte de interes și intervalul de timp disponibil. Soluția este capabilă să integreze date în timp real privind traficul și condițiile meteorologice, ajustând penalizările și scorurile aferente fiecărui segment. Rezultatul este un traseu personalizat, dar flexibil, care poate fi reconfigurat automat în funcție de schimbările din mediul operațional.

Toate aceste componente au fost concepute pentru a putea fi implementate într-un sistem real. A fost definită o arhitectură hardware completă (unitate de procesare, senzori, actuatori), alături de un flux software modular care include percepția mediului, localizarea vehiculului, planificarea traseului și controlul dinamic al mișcării. Arhitectura este compatibilă cu cerințele de timp real și se aliniază cu standardele utilizate în industrie.

Printre contribuțiile importante ale lucrării se numără: dezvoltarea unei metode de evitare a coliziunilor bazată pe detecție geometrică și nu pe clasificare; proiectarea unui algoritm de planificare robust, care ține cont de contextul dinamic și preferințele utilizatorului; validarea tuturor componentelor într-un simulator avansat; și definirea unei arhitecturi complete pentru implementare practică.

Pentru viitor, se conturează mai multe direcții de cercetare. Una dintre acestea este optimizarea multi-obiectiv, prin introducerea unor algoritmi genetici de tip NSGA-II, care să echilibreze simultan mai multe criterii, precum satisfacția utilizatorului, timpul de deplasare, costurile și consumul de energie. O altă direcție constă în integrarea de modele predictive (pentru trafic sau vreme), astfel încât algoritmul să poată anticipa probleme și să ajusteze itinerariul proactiv.

De asemenea, învățarea preferințelor utilizatorului reprezintă o direcție promițătoare. Algoritmul ar putea adapta scorurile obiectivelor turistice pe baza comportamentului observat sau a parametrilor expliți definiți de utilizator (ex. flexibilitate, interes tematic). Sistemul ar putea deveni astfel tot mai personalizat și centrat pe utilizator.

Scalabilitatea și testarea în condiții reale constituie o etapă firească de extindere. Sistemul ar putea fi validat într-un oraș real, chiar fără vehicule complet autonome, printr-o aplicație mobilă care să genereze trasee personalizate pentru utilizatori umani. În paralel, integrarea planificatorului cu nivelurile de conducere autonomă joasă ar permite o adaptare rapidă la situații neprevăzute, de exemplu prin detectarea unor drumuri impracticabile sau sincronizarea vitezei cu programările punctelor de interes.

Nu în ultimul rând, optimizarea energetică și introducerea unui mecanism interactiv de decizie (human-in-the-loop) pot adăuga valoare practică substanțială sistemului propus. De exemplu, planificatorul ar putea sugera alternative atunci când apar schimbări bruște de context, lăsând utilizatorul să aleagă între mai multe opțiuni optimizate.

În concluzie, această lucrare propune o soluție completă și realistă pentru conducerea autonomă orientată către turism urban, demonstrând că algoritmi genetici pot fi integrați eficient în sisteme autonome reale, cu capacitate de adaptare, personalizare și operare sigură în condiții dinamice.

Bibliografie

- [1] Sandu, C., and Șuşnea, I. "An Approach to Real-Time Collision Avoidance for Autonomous Vehicles Using LiDAR Point Clouds." *Journal of Applied Engineering Sciences* 12, no. 1 (2022): 129–134.
- [2] Velasco-Hernandez, G., Barry, J., and Walsh, J. "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review." *Sensors* 21, no. 6 (2021): 2140.
- [3] Tan, H., Zhao, F., Song, H., and Liu, Z. "Quantifying the Impact of Deployments of Autonomous Vehicles and Intelligent Roads on Road Safety in China: A Country-Level Modeling Study." *International Journal of Environmental Research and Public Health* 20, no. 5 (2023): 4069.
- [4] Ding, S., and Abdel-Aty, M. "A Matched Case-Control Analysis of Autonomous vs Human-Driven Vehicle Accidents." *Nature Communications* 15, no. 1 (2024): Article 4931.
- [5] Scanlon, J. M., Kusano, K. D., Daniel, T., Alderson, C., Ogle, A., and Victor, T. "Waymo Simulated Driving Behavior in Reconstructed Fatal Crashes within an Autonomous Vehicle Operating Domain." *Accident Analysis & Prevention* 163 (2021): 106454.
- [6] Othman, K. "Exploring the Implications of Autonomous Vehicles: A Comprehensive Review." *Innovative Infrastructure Solutions* 7, no. 2 (2022): 165.
- [7] Ercan, T., Onat, N. C., Keya, N., Tatari, Ö., Eluru, N., and Küçükvar, M. "Autonomous Electric Vehicles Can Reduce Carbon Emissions and Air Pollution in Cities." *Transportation Research Part D: Transport and Environment* 112 (2022): 103472.
- [8] Rhim, J., Lee, J.-H., Chen, M., and Lim, A. "A Deeper Look at Autonomous Vehicle Ethics: An Integrative Ethical Decision-Making Framework to Explain Moral Pluralism." *Frontiers in Robotics and AI* 8 (2021): Article 632394.
- [9] Zhai, S., Wang, L., and Liu, P. "Not in Control, but Liable? Attributing Human Responsibility for Fully Automated Vehicle Accidents." *Engineering* 33 (2024): 170–179.
- [10] Bezai, N. E., Medjdoub, B., Al-Habaibeh, A., Chalal, M. L., and Fadli, F. "Future Cities and Autonomous Vehicles: Analysis of the Barriers to Full Adoption." *Energy and Built Environment* 2, no. 1 (2021): 65–81.
- [11] Reda, M., Onsy, A., Ghanbari, A., and Haikal, A. Y. (2024). "Path Planning Algorithms in the Autonomous Driving System: A Comprehensive Review." *Robotics and Autonomous Systems*, 174, 104630.
- [12] Ma, Y., Wang, Z., Yang, H., and Yang, L. (2020). "Artificial Intelligence Applications in the Development of Autonomous Vehicles: A Survey." *IEEE/CAA Journal of Automatica Sinica*, 7(2), 315–329.
- [13] Maw, A. A., Tyan, M., and Lee, J. W. (2020). "iADA: Improved Anytime Path Planning and Replanning Algorithm for Autonomous Vehicle." *Journal of Intelligent & Robotic Systems*, 100(3–4), 1005–1013.

- [14] Thoresen, M., Nielsen, N. H., Mathiassen, K., and Pettersen, K. Y. (2021). "Path Planning for UGVs Based on Traversability Hybrid A*." *IEEE Robotics and Automation Letters*, 6(2), 1216–1223.
- [15] Liu, T., and Zhang, J. (2022). "An Improved Path Planning Algorithm Based on Fuel Consumption." *The Journal of Supercomputing*, 78(2), 12973–13003.
- [16] Zhong, X., Tian, J., Hu, H., and Peng, X. (2020). "Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment." *Journal of Intelligent & Robotic Systems*, 99(1), 65–77.
- [17] Feraco, S., Luciani, S., Bonfitto, A., Amati, N., and Tonoli, A. (2020). "A Local Trajectory Planning and Control Method for Autonomous Vehicles Based on the RRT Algorithm." In *Proceedings of the 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT Automotive)*, 1–6. IEEE.
- [18] Ganesan, P., Kuppan, L., and Kumar, D. N. (2021). "A Novel Directional Sampling-Based Path Planning Algorithm for Autonomous Mobile Robots." *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 235(10), 1203–1214.
- [19] Wang, J., Li, B., and Meng, M. Q.-H. (2020). "Neural RRT: Learning-Based Optimal Path Planning." *IEEE Transactions on Automation Science and Engineering*, 17(4), 1748–1758.
- [20] Wang, J., Li, B., and Meng, M. Q.-H. (2021). "Kinematic Constrained Bi-Directional RRT with Efficient Branch Pruning for Robot Path Planning." *Expert Systems with Applications*, 170, 114541.
- [21] Li, Q., Xu, Y., Bu, S., and Yang, J. (2022). "Smart Vehicle Path Planning Based on Modified PRM Algorithm." *Sensors*, 22(17), 6581.
- [22] Gopika, M., Bindu, G. R., Ponmalar, M., Usha, K., and Haridas, T. R. (2022). "Smooth PRM Implementation for Autonomous Ground Vehicle." In *Proceedings of the 2022 IEEE 1st International Conference on Data, Decision and Systems (ICDDS)*, 1–5. IEEE.
- [23] Li, X., Li, G., and Bian, Z. (2024). "Research on Autonomous Vehicle Path Planning Algorithm Based on Improved RRT Algorithm and Artificial Potential Field Method." *Sensors*, 24(7), 3899.
- [24] Yang, G., and Yao, Y. (2022). "Vehicle Local Path Planning and Time Consistency of Unmanned Driving System Based on Convolutional Neural Network." *Neural Computing and Applications*, 34(15), 12385–12398.
- [25] Li, J., Chen, Y., Zhao, X., and Huang, J. (2022). "An Improved DQN Path Planning Algorithm." *The Journal of Supercomputing*, 78(1), 616–639.
- [26] Peng, Y., Tan, G., Si, H., and Li, J. (2022). "DRL-GAT-SA: Deep Reinforcement Learning for Autonomous Driving Planning Based on Graph Attention Networks and Simplex Architecture." *Journal of Systems Architecture*, 126, 102505.

- [27] Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L. M., Gómez-Huélamo, C., Gutiérrez, R., and Díaz-Díaz, A. (2022). "Deep Reinforcement Learning Based Control for Autonomous Vehicles in CARLA." *Multimedia Tools and Applications*, 81(3), 3553–3576.
- [28] Zhang, L., Zhang, R., Wu, T., Weng, R., Han, M., and Zhao, Y. (2021). "Safe Reinforcement Learning with Stability Guarantee for Motion Planning of Autonomous Vehicles." *IEEE Transactions on Neural Networks and Learning Systems*, 32(12), 5435–5444.
- [29] Spanogiannopoulos, S., Zweiri, Y., and Seneviratne, L. (2022). "Sampling-Based Non-Holonomic Path Generation for Self-Driving Cars." *Journal of Intelligent & Robotic Systems*, 104(1), 1–17.
- [30] Thoresen, M., Nielsen, N. H., Mathiassen, K., and Pettersen, K. Y. (2021). "Path Planning for UGVs Based on Traversability Hybrid A*." *IEEE Robotics and Automation Letters*, 6(2), 1216–1223.
- [31] Liu, L., Lin, J., Yao, J., He, D., Zheng, J., Huang, J., and Shi, P. (2021). "Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach." *Wireless Communications and Mobile Computing*, 2021, Article ID 5514304 (9 pages).
- [32] He, H., Chi, Z., Sun, Y., Zhang, B., and Guo, F. (2018). "An Improved Artificial Potential Field Model Considering Vehicle Velocity for Autonomous Driving." *IFAC-PapersOnLine*, 51(31), 863–867.
- [33] Lu, B., Li, G., Yu, H., Wang, H., Guo, J., Cao, D., and He, H. (2020). "Adaptive Potential Field-Based Path Planning for Complex Autonomous Driving Scenarios." *IEEE Access*, 8, 225294–225305.
- [34] Receveur, J.-B., Victor, S., and Melchior, P. (2020). "Autonomous Car Decision Making and Trajectory Tracking Based on Genetic Algorithms and Fractional Potential Fields." *Intelligent Service Robotics*, 13(2), 315–330.
- [35] Magdy, Y., Shehata, O. M., AbdelAziz, M., Ghoneima, M., and Tolbah, F. (2017). "Metaheuristic Optimization in Path Planning of Autonomous Vehicles under the ATOM Framework." In *Proceedings of the 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 32–37. Turin, Italy.
- [36] Paden, B., Čáp, M., Yong, S. Z., Yershov, D., and Frazzoli, E. (2016). "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles." *IEEE Transactions on Intelligent Vehicles*, 1(1), 33–55.
- [37] Ahn, S., Kim, J., Park, J., Lee, J., and Lee, J. (2024). "Collision Avoidance Path Planning for Automated Vehicles Using Prediction Information and Artificial Potential Field." *Sensors*, 24(22), 7292.
- [38] Ge, S., Cui, Y., and Li, S. (2019). "An Enhanced Artificial Potential Field Approach for Real-Time Obstacle Avoidance of Mobile Robots." *IEEE Access*, 7, 77819–77828.
- [39] Khatib, O. (1986). "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." *The International Journal of Robotics Research*, 5(1), 90–98.
- [40] Park, J., Kim, J., Lee, S., and Lee, J. (2019). "Safe Motion Planning under Uncertain Environment." *Robotics and Autonomous Systems*, 122, 103–120.

- [41] Fridovich-Keil, D., Fisac, J. F., Herbert, S. L., Bajcsy, A., Wang, S., and Tomlin, C. J. (2021). "A Probabilistically Safe Motion Planning Framework with Safety Envelopes." *IEEE Transactions on Robotics*, 37(3), 892–908.
- [42] Dong, D., Ye, H., Luo, W., Wen, J., and Huang, D. (2024). "Collision Avoidance Path Planning and Tracking Control for Autonomous Vehicles Based on Model Predictive Control." *Sensors*, 24(16), 5211.
- [43] Fan, T., Long, P., Liu, W., and Pan, J. (2020). "End-to-End Collision Avoidance via Deep Reinforcement Learning." In *Robotics: Science and Systems*.
- [44] Han, Y., Zhan, I. H., Zhao, W., Pan, J., Zhang, Z., Wang, Y., and Liu, Y. J. (2022). "Deep Reinforcement Learning for Robot Collision Avoidance with Self-State-Attention and Sensor Fusion." *IEEE Robotics and Automation Letters*, 7(3), 6886–6893.
- [45] Pfeiffer, M., Shukla, S., Turchetta, M., Cadena, C., Krause, A., Siegwart, R., and Nieto, J. (2018). "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Map-less Navigation by Leveraging Prior Demonstrations." *IEEE Robotics and Automation Letters*, 3(4), 4423–4430.
- [46] Vanholme, B., Gruyer, D., Glaser, S., and Mammar, S. (2018). "Highly Automated Driving on Highways Based on Beyond Euro NCAP Safety Standards." *Transportation Research Part F: Traffic Psychology and Behaviour*, 58, 193–207.
- [47] Koenig, S., and Likhachev, M. (2002). "D* Lite." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 476–483.
- [48] Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2017). "On a Formal Model of Safe and Scalable Self-Driving Cars." *arXiv preprint arXiv:1708.06374*.
- [49] Sandu, C., Dumitriu, L., and Şuşnea, I. (2024). "Optimising Itinerary Generation for Autonomous Tour Guides by Integrating Real-Time Data into Genetic Algorithms." *Romanian Journal of Information Technology and Automatic Control*, 34(4), 115–126.
- [50] Basu, A., Raja, B., Gracious, R., and Vanajakshi, L. (2020). "Dynamic Trip Planner for Public Transport Using Genetic Algorithm." *Transport*, 35(2), 156–167.
- [51] Zhang, J., Tang, J., and Zheng, J. (2019). "Adaptive Genetic Algorithm for the Dynamic Traveling Salesman Problem with Traffic Factors." *Applied Soft Computing*, 85, 105778.
- [52] Tang, J., Pan, J.-S., and Wang, S. (2017). "Hybrid Genetic Algorithm and Tabu Search for the Dynamic Shortest Path Routing Problem with Two Objectives." *Applied Soft Computing*, 53, 760–771.
- [53] Pehlivanoglu, Y. V., and Pehlivanoglu, P. (2021). "An Enhanced Genetic Algorithm for Path Planning of Autonomous UAV in Target Coverage Problems." *Applied Soft Computing*, 112, 107796.
- [54] Rong, H. H., Tu, W., Duarte, F., and Ratti, C. (2020). "Employing Waterborne Autonomous Vehicles for Museum Visits: A Case Study in Amsterdam." *European Transport Research Review*, 12(1), 1–12.

- [55] Zhou, X., Su, M., Liu, Z., and Zhang, D. (2019). "Smart Tour Route Planning Algorithm Based on Clustering Center Motive Iteration Search." *IEEE Access*, 7, 185607–185633.
- [56] Damos, M. A., Zhu, J., Li, W., Hassan, A., and Khalifa, E. (2021). "A Novel Urban Tourism Path Planning Approach Based on a Multiobjective Genetic Algorithm." *ISPRS International Journal of Geo-Information*, 10(8), 530.
- [57] Du, S., and Liu, Y. (2023). "TPS-Genetic Algorithm for Real-Time Sailing Route Planning Based on Potential Field Theory." *European Journal of Engineering and Technology Research*, 8(3), 108–113.
- [58] Azad, M., Hoseinzadeh, N., Brakewood, C., Cherry, C. R., and Han, L. D. (2019). "Fully Autonomous Buses: A Literature Review and Future Research Directions." *Journal of Advanced Transportation*, 2019, Article ID 4603548.
- [59] Bezai, N. E., Medjdoub, B., Al-Habaibeh, A., Chalal, M. L., and Fadli, F. (2021). "Future Cities and Autonomous Vehicles: Analysis of the Barriers to Full Adoption." *Energy and Built Environment*, 2(1), 1–52.
- [60] Wang, H., and Chen, Y. (2015). "A Genetic Algorithm for Solving Dynamic Traveling Salesman Problem." *International Journal of Simulation Modelling*, 14(1), 138–147.
- [61] Tsoukas, V., Boumpa, E., Chioktour, V., Kalafati, M., Spathoulas, G., and Kakarountas, A. (2023). "Development of a Dynamically Adaptable Routing System for Data Analytics Insights in Logistic Services." *Analytics*, 2(2), 328–345.
- [62] Dosovitskiy, A., Ros, G., Codevilla, F., López, A., and Koltun, V. (2017). "CARLA: An Open Urban Driving Simulator." In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, 1–16.
- [63] Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). "End-to-End Driving via Conditional Imitation Learning." In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 4693–4700.
- [64] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2020). "Learning by Cheating: Imitation Learning for Autonomous Driving." In *Proceedings of the Conference on Robot Learning (CoRL)*, 66–75.
- [65] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A., Yogamani, S., and Pérez, P. (2021). "Deep Reinforcement Learning for Autonomous Driving: A Survey." *IEEE Transactions on Intelligent Transportation Systems*, 22(6), 3662–3679.
- [66] Sandu, C., and Şuşnea, I. (2022). "An Approach to Real-Time Collision Avoidance for Autonomous Vehicles Using LiDAR Point Clouds." *Journal of Applied Engineering Sciences*, 12(1), 129–134.
- [67] Azfar, T., and Ke, R. (2024). "Traffic Co-Simulation Framework Empowered by Infrastructure Camera Sensing and Reinforcement Learning." *arXiv preprint arXiv:2401.12345*.
- [68] Reda, M., Onsy, A., Ghanbari, A., and Haikal, A. Y. (2024). "Path Planning Algorithms in the Autonomous Driving System: A Comprehensive Review." *Robotics and Autonomous Systems*, 174, 104630.

- [69] Milner, J. (2022). "A Visual Guide to the Software Architecture of Autonomous Vehicles." Medium. Retrieved from <https://medium.com/autonomous-vehicle>
- [70] Autoware Foundation. (2023). "Autoware Architecture Documentation." Retrieved from <https://autoware.org/docs/architecture/>
- [71] Sandu, C., Dumitriu, L., și Sușnea, I. (2024). "Optimising Itinerary Generation for Autonomous Tour Guides by Integrating Real Time Data into Genetic Algorithms." *Romanian Review of Informatics and Automation*, 34(4), 129–134.
- [72] Wijayarathna, K., Seneviratne, R., și Premachandra, C. (2021). "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review." *Sensors*, 21(16), 5397.
- [73] Fang, J., Chen, Z., și Zhang, Y. (2020). "A Comprehensive Study of GPS, IMU, and Vision Sensors for Vehicle Localization." *Sensors*, 20(18), 4820.
- [74] Waymo LLC. (2022). Waymo Driver, Technology Overview. <https://waymo.com/tech/> (accesat 2025).
- [75] Sawhney, P. (2021). "Autonomous Vehicle Architecture, Sensor & Perception." Medium. <https://medium.com/av-sensor-architecture> (accesat 2025).