UNIVERSITATEA NAŢIONALĂ DE ŞTIINŢĂ ŞI TEHNOLOGIE POLITEHNICA BUCUREŞTI

FACULTATEA DE INGINERIE MECANICĂ ȘI MECATRONICĂ

Programul de studii: Mecatronică Avansată

PROIECT DE DISERTAȚIE

CERCETARI PRIVIND UNELE SOLUȚII CONSTRUCTIVE DE ROBOȚI MOBILI BIOINSPIRAȚI

ÎNDRUMĂTOR ŞTIINŢIFIC,

Ș.l.dr.ing.Edgar MORARU

MASTERAND,

Alexandru GRĂMADĂ

Bucureşti

2026

Cuprins

I. Introducere

În studiul soluțiilor constructive ale roboților se remarcă cei cu inspirație animală, imitând sistemele avansate de locomoție ale creaturilor din mediul înconjurător - preluând avantajele a milioane de ani de evoluție genetică de adaptare în mediul lor de supraviețuire. Utilitatea s-a demonstrat în timp prin algoritmi precum cei de optimizare, exemplu fiind cea a coloniilor de furnici[6], dar și prin reproducerea sistemelor musculo-scheletice (recent progrese în roboții cu segmente flexibile).

Soluția ce va fi propusă este o continuare a unui proiect de cercetare trecut - un robot patruped pentru explorarea mediului - se vor cerceta modalități de simulare a robotului și a senzorilor acestuia într-un spațiu virtual și de implementare a algoritmilor de Reinforcement Learning pentru dezvoltarea unui mers eficient, utilizând apoi soluții deja existente pentru transmiterea datelor.

Sistemul studiat va deservi desfășurării operațiunilor de căutare și recuperare a persoanelor pierdute sau rănite din spațiul forestier, prin cartografierea premeditată de la nivelul solului - utilă pentru descoperirea căilor de acces terestru, cât și pentru identificarea zonelor înguste (posibilitatea prezenței unei persoane în pericol este mai mare în zonele puțin accesibile și dense), dar și facilitarea transportului de echipament medical până la victimă. Astfel, acesta necesită un design robust și capacitatea de a se adapta rapid mediului.

Se vor observa avantaje și dezavantaje ale construcției, logica computațională necesară pentru simularea eficientă într-un software 3D, cât și a parcurgerii automate ale robotului într-un mediu simulat.

Se propune studiul simulării folosing Godot, un software specializat pe crearea jocurilor 2D, dar mai nou foarte eficient pentru dezvoltarea 3D, deci capabil și de simulari rapide. Acest software are numeroase avantaje precum prototiparea rapidă, un limbaj ușor de înțeles, în stilul Python, cu metode integrate de a comunica cu alte programe, fiind de asemenea Open Source - astfel fiind dezvoltat de comunitate, iar codul de baza putând fi manipulat.

La scară largă apare problema de eficiență de procesare și ulterior de stocare a punctelor și imaginilor ce reconstruiesc mediul străbătut. Se propune utilizarea unui sistem de înregistrare parțială a mediului, folosind mai puține puncte din mediu în timpul operațiunii de topografiere și reconstrucția logică utilizând utilizând mai puține poligoane. Propunerea nu este de a scădea rezoluția de înregistrare pentru toate obstacolele întâlnite (distanța dintre puncte și numărul lor),

ci simplificarea geometrica a anumitor elemente din natură ce nu sunt de interes / a caror simplificare nu afectează parcurgerea eficientă a robotului de explorare și ulterior a unui potențial salvamontist.

Înregistrarea informațiilor din mediu se propune a fi executată utilizând o combinație de camere stereo, GPS și un sistem LiDAR (Light Detection and Ranging), prin metoda SLAM (Simultaneous Localization and Mapping), ce va fi explicată în capitolul următor.

Această propunere vine deodată cu reglarea parametrilor de dinaintea misiunii de explorare sau chiar în timpul ei, unde costul rezoluției suplimentare este o viteză scăzută de deplasare a robotului pentru permiterea procesării topologice.

II. Stadiu Actual

Pentru studiul unui sistem, trebuiesc luați în considerare pașii parcurși în ultimii ani ce sunt legați de subiectul propus. Astfel s-au studiat numeroase surse științifice ce acoperă laturile de interes ce compun robotul bioinspirat propus:

1. Design structural eficient

Roboții mobili se clasifică în trei tipuri: cu roți, cu picioare și cu șenile. Cei cu roți sunt eficienți și ușor de manevrat pe terenuri ușor înclinate, dar au dificultăți pe suprafețe accidentate. Roboții cu șenile pot traversa terenuri denivelate sau instabile datorită mecanicii speciale, însă sunt lenți pe suprafețe netede. Roboții cu picioare oferă o adaptabilitate bună și mențin viteze de deplasare constante. Traiectoria lor este discretă, necesitând o continuitate redusă a suprafeței, ceea ce îi face potriviți pentru medii neamenajate. Cei mai folosiți sunt roboții bipezi, patrupezi și hexapozi. Dintre aceștia, roboții patrupezi combină stabilitatea și dinamica bună cu o mecanică mai simplă, conformându-se cerințelor industriale moderne, motiv pentru care sunt intens studiați.

Roboții patrupezi pot fi împărțiți în trei categorii, în funcție de metoda de acționare: hidraulică, electrică sau pneumatică. Acționările hidraulice permit manipularea sarcinilor mari, dar implică o greutate totală excesivă; acționările electrice oferă precizie ridicată în control, dar au o capacitate de încărcare redusă; acționările pneumatice prezintă caracteristici de linearitate slabe, ceea ce le face dificil de controlat și, prin urmare, au puține aplicații practice.

În prezent, structura mecanică a roboților patrupezi este relativ matură, cu numeroase designuri eficiente și stabile care respectă cerințele de control. Principala provocare constă în cadrul de control și algoritmii utilizați, mai degrabă decât în structura mecanică. [1]

Se analizează structural roboții patrupezi cu 12 grade de libertate (figura 1), structură ce va fi folosită pentru design-ul final al proiectului.

Picioarele robotului patruped intră frecvent în contact cu solul, fiind o sursă principală de forță, astfel încât designul acestora este esențial pentru o locomotie reușită. Mișcarea de balans înainte a articulației șoldului coapsei utilizează rulmenți pentru a susține forțele axiale.

În locul unei conexiuni directe cu servomecanism la articulația genunchiului, se utilizează o articulație cu cap sferic, iar partea inferioară a piciorului este acționată indirect de brațul basculant al servomecanismului. Acest lucru permite amplasarea servomecanismului pe partea superioară a coapsei, lângă articulația șoldului, unde inerția rotațională a piciorului este relativ scăzută.

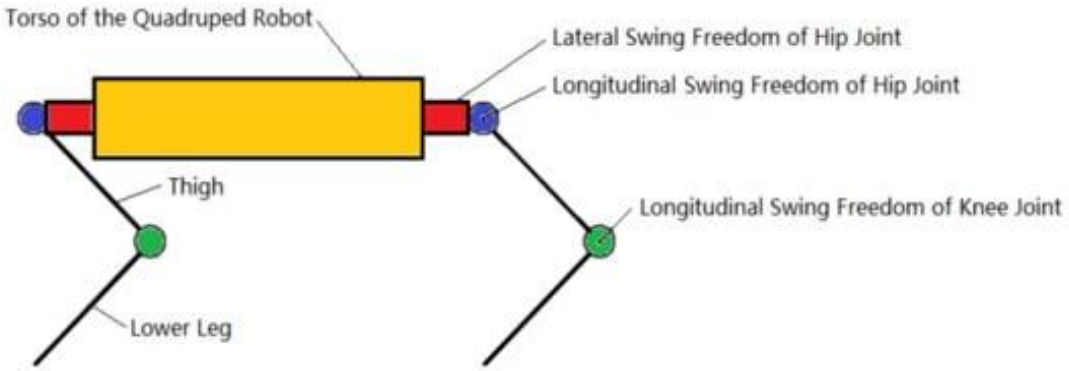

Figura II.1 - Structura robotului patruped, vizualizată din lateral[1]

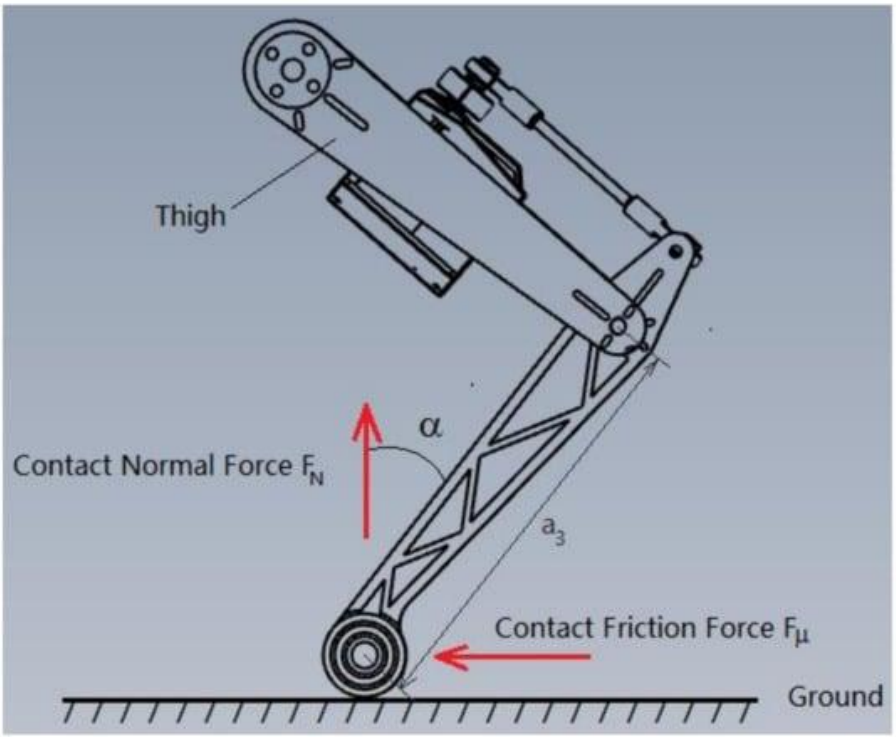

Figura II.2 - Structura piciorului robotic analizat[1]

Figura II.3 - Întregul sistem propus în articolul studiat[1]

2. Control optimal

Începând cu anii 1960, numeroase studii experimentale au fost realizate în diverse laboratoare, dezvoltându-se multe idei de control mainstream, rezumate succint mai jos.

Teoria controlului ZMP (Zero Moment Point) ia în considerare efectele forțelor inerțiale generate de accelerația mișcării robotului atunci când acesta se deplasează cu un mers static și introduce conceptul de ZMP. Deși această teorie a fost eficientă în primele etape ale roboticii, cercetările ulterioare au evidențiat limitările sale. De exemplu, implică un volum mare de calcule, ceea ce afectează performanța în timp real, și nu este aplicabilă pe terenuri complexe.

Algoritmul CPG (Central Pattern Generator) este o rețea neuronală capabilă să genereze un output specific, imitând circuitele neuronale din sistemul nervos central al unor organisme inferioare. Fiecare articulație controlată are un neuron, iar rețeaua CPG formată controlează mișcările robotului. Cu toate acestea, parametrii CPG sunt complexi și dificil de ajustat, iar transformările necesare pentru controlul unghiurilor articulațiilor reduc eficiența, mai ales pe terenuri dificile care necesită ajustări rapide.

Modelul SLIP (Spring-Loaded Inverted Pendulum) este cea mai utilizată abordare de control pentru roboții cu picioare și este în continuă perfecționare. Acesta împarte controlul mișcării robotului în trei componente: „înălțimea săriturii - viteza înainte - poziția corpului”. Fiecare componentă este controlată printr-un regulator PD, iar coeficienții pot fi ajustați individual. Odată optimizați, aceștia asigură echilibrul dinamic al unui robot monopod prin sărituri constante. Modelul a evoluat ulterior, introducând conceptul de „picioare virtuale”, care simplifică controlul unui robot cu mai multe picioare, reducându-l la cel al unui robot monopod cu stabilitate dinamică ridicată.

Pe baza modelului SLIP, Pratt și colaboratorii au propus conceptul VMC (Virtual Model Control). Acesta se concentrează pe mișcarea centrului de masă al robotului, presupunând existența unor elemente virtuale precum arcuri și amortizoare. Forțele virtuale orizontale „trag” corpul în direcția dorită, cele verticale „susțin” corpul la o anumită înălțime, iar momentele virtuale ajustează atitudinea pe axele de tangaj, ruliu și girație. Totuși, parametrii PD necesari acestui model sunt sensibili la variațiile modelului și dificil de optimizat.

Implementarea cu succes a acestor controale necesită modelarea adecvată a interacțiunii complexe dintre picior și sol. În timpul deplasării pe orice teren, interacțiunea de frecare dintre picioare și sol joacă un rol crucial în viteza și eficiența locomotiei. Este esențial să se modeleze și să se compenseze alunecările, deoarece acestea pot modifica gradele de libertate ale picioarelor de sprijin și pot duce la instabilitate sau răsturnare pe terenuri denivelate, pante abrupte sau suprafețe alunecoase. [1]

3. Mișcarea într-un regim optim

În cazul unui robot motorizat, prin aranjarea motoarelor care efectuează o mișcare specifică, fie în sensul acelor de ceasornic, fie invers, se poate obține un mers particular. De exemplu, pentru un robot cu patru picioare, dacă fiecare picior este controlat de un motor pas cu pas, motoarele pot fi aranjate astfel încât piciorul stâng frontal să se miște inițial, urmat de piciorul drept posterior, apoi piciorul drept frontal, urmat de piciorul stâng posterior. Acest tip de mișcare a picioarelor este numit „mers”. În mod similar, diferite aranjamente pot genera diverse tipuri de mersuri.

Una dintre metodele avansate utilizate până în prezent este implementarea unui algoritm complex. Ea implică programarea mersului prin calcularea atentă a poziționării unghiulare a segmentelor piciorului. În funcție de tipul de sistem utilizat pentru acționare, se creează un program care este introdus într-un microcontroller ce guvernează mersul robotului. Utilizarea unui algoritm împarte operațiunea de furnizare a intrării către sistem, analizarea acesteia și generarea mersului între dispozitivele auxiliare. Aceasta elimină supraîncărcarea și permite funcționarea lină a fiecărei componente, indiferent de celelalte. Algoritmii oferă, de asemenea, un răspuns imediat al robotului în cazul apariției unor noi obstacole. Acest grad de flexibilitate pentru optimizarea mersului în funcție de fiecare obstacol în schimbare este mai mare comparativ cu orice altă metodă de generare a mersului. [2]

Se disting diverse tipuri de mers, cu avantaje și dezavantaje, pentru stabilitate statică sau dinamică.

In stabilitate statică, sunt reprezentate în figura 3 combinațiile posibile de alternare ale pașilor pentru menținerea echilibrului. Un alt nume al deplasării din 3 este mers târâș, fiecare picior fiind poziționat în ordine, unul pe rând, doar atunci când celălalte membre sunt în contact cu solul.

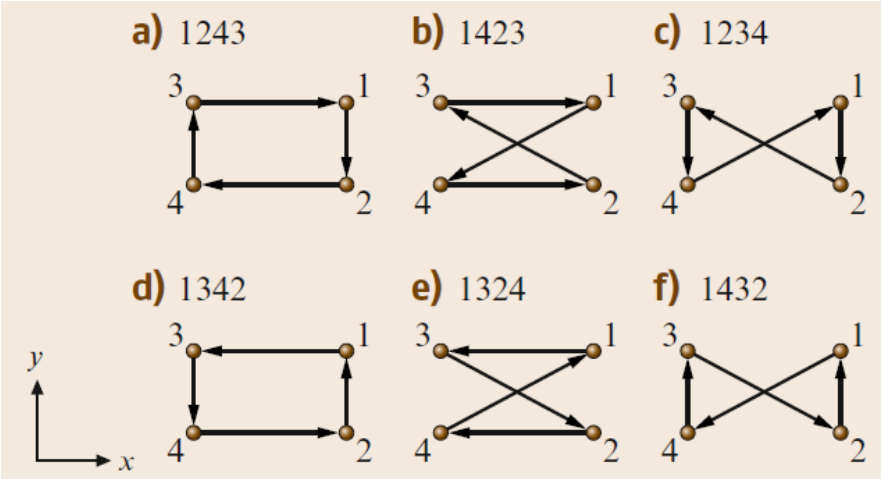

Figura II.4 - configurații ale mersului cu echilibru static

În urma unor analize, mersul de tipul b) 1423 oferă cea mai mare stabilitate pentru mersul pe axa X, așa cum e) 1324 oferă aceeași stabilitate pentru deplasarea inversă pe axa X, iar c) 1234 și f) 1432 pe axa Y, respectiv -Y. Se notează că deși mai putin stabile, a) 1243 și d) 1342 sunt optime pentru întoarcerea robotului. [3]

Se observă în următoarea figură mișcarea unui alt picior după ce precedentul a făcut contact cu solul

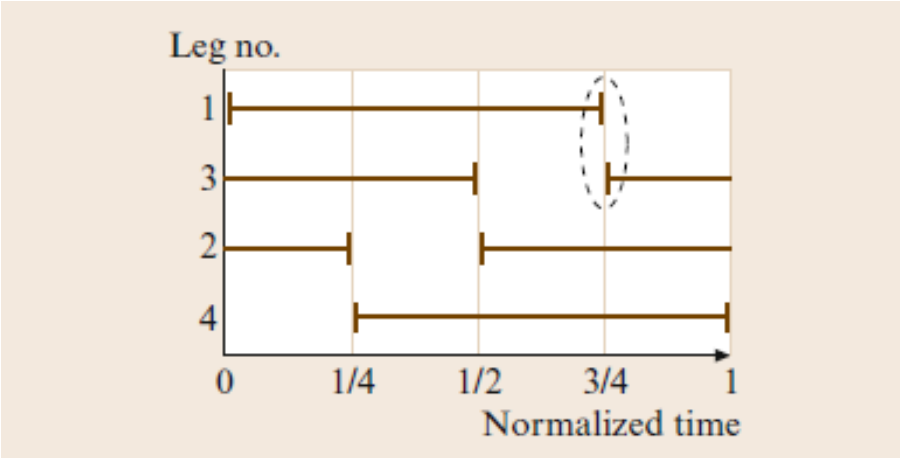

Figura II.5 - Plasarea imediată a următorului picior după ridicarea celui anterior

Cel mai studiat model pentru picioarele robotice destinate mersului alergător este modelul SLIP (Spring Loaded Inverted Pendulum) – figura 5. În faza de zbor, arcul nu are efect și, prin urmare, nu este luat în considerare, astfel încât dinamica piciorului este reprezentată luând în calcul doar masa punctuală.[4]

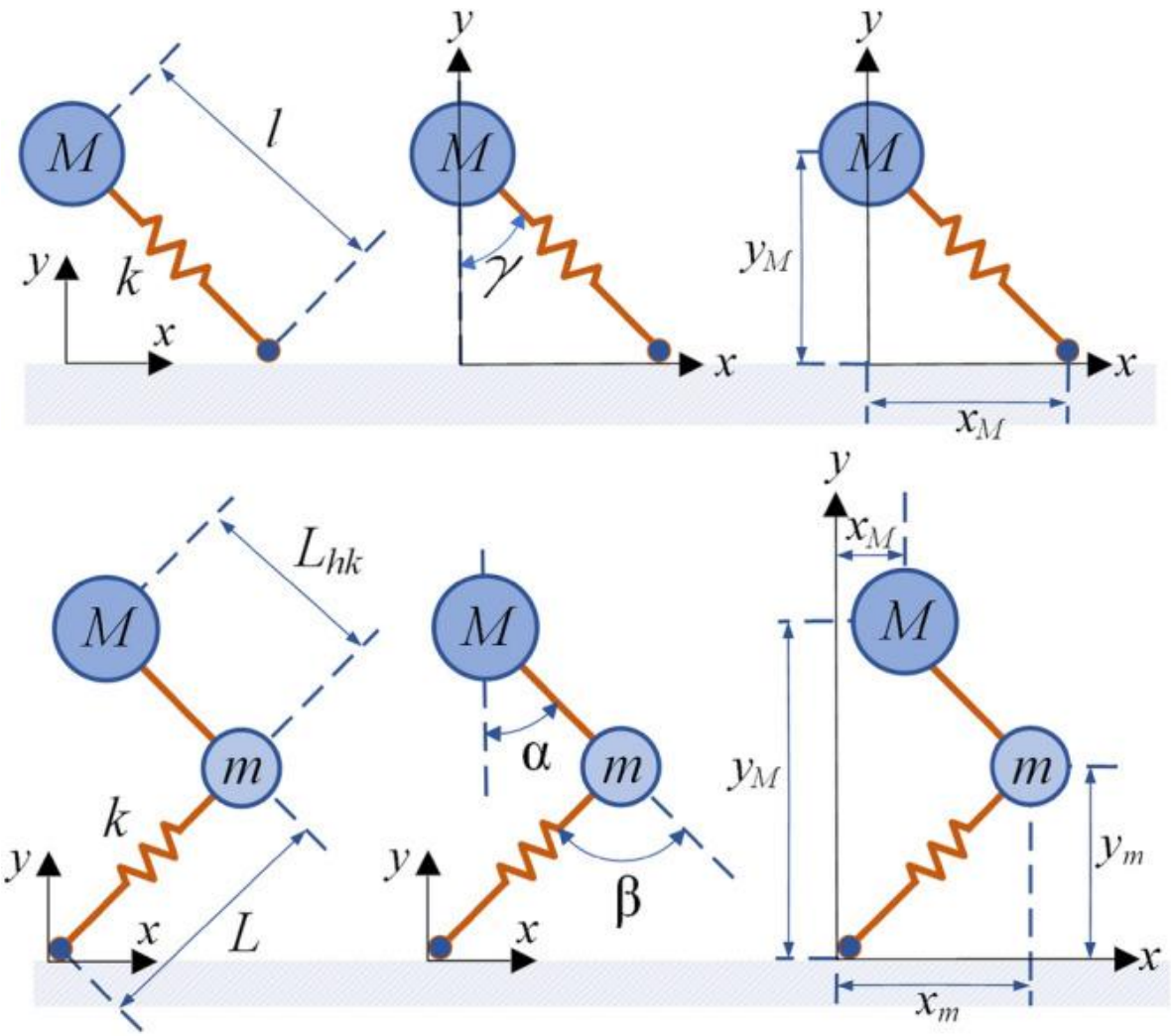

Figura II.6 - Diagrama modelului de picior SLIP (sus) și a modelului Masă-Masă-Arc (MMS) (jos)

Modelul SLIP este compus dintr-o masă punctuală, M, care reprezintă șoldul, și un arc liniar, k, care transmite forțele de reacție între sol și șold, acționând ca un sistem de stocare a energiei în timpul fazei de sprijin. Pe lângă masa șoldului, modelul MMS ia în considerare și masa piciorului, localizată la articulația genunchiului, precum și un arc care stabilește contactul între masa piciorului și sol, așa cum este ilustrat în figura următoare. Datorită masei suplimentare m, este posibilă modelarea forțelor care permit trunchiului să se îndoaie în faza de zbor.

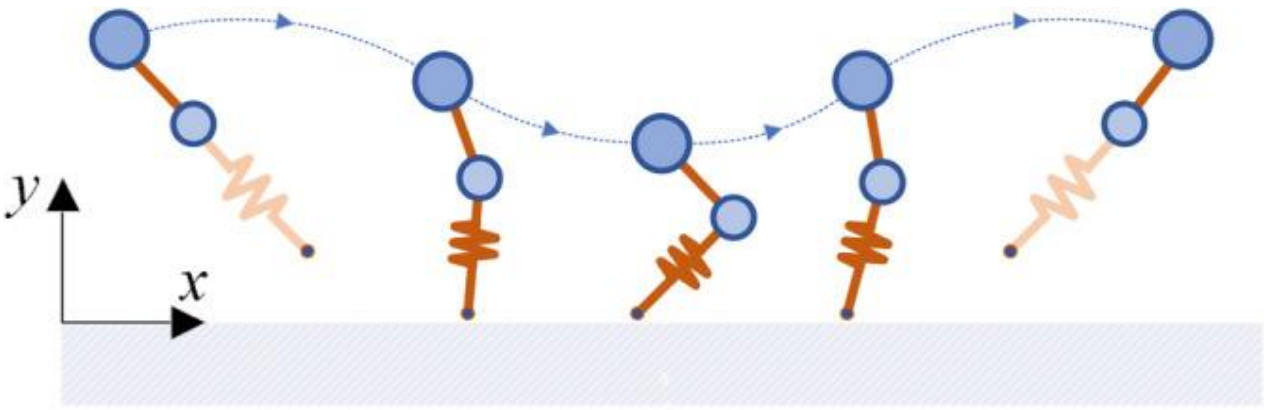

Figura II.7 - Secvența mișcării modelului MMS. Faza de sprijin este în centrul imaginii, iar faza de zbor este pe părțile laterale ale imaginii.

4. Categorisirea automată a obiectelor din mediu

Detectarea marginilor este un instrument de bază utilizat în procesarea imaginilor, în principal pentru detectarea și extragerea caracteristicilor, cu scopul de a identifica punctele dintr-o imagine digitală unde luminozitatea imaginii se schimbă brusc și de a găsi discontinuități. Scopul detectării marginilor este de a reduce semnificativ cantitatea de date dintr-o imagine și de a păstra proprietățile structurale pentru procesarea ulterioară a imaginii. Într-o imagine de nivel gri, marginea este o caracteristică locală care, într-o vecinătate, separă regiunile în fiecare dintre care nivelul de gri este mai mult sau mai puțin uniform, având valori diferite pe cele două părți ale marginii. Pentru o imagine zgomotoasă, este dificil să se detecteze marginile, deoarece atât marginea, cât și zgomotul conțin componente de frecvență înaltă, ceea ce duce la un rezultat neclar și distorsionat.[5]

Detectorul de margini Canny are un algoritm avansat derivat din munca anterioară a lui Marr și Hildreth. Este o tehnică optimă de detectare a marginilor, deoarece oferă o detectare bună, un răspuns clar și o localizare precisă. Este utilizat pe scară largă în tehnicile actuale de procesare a imaginilor, cu îmbunătățiri suplimentare.

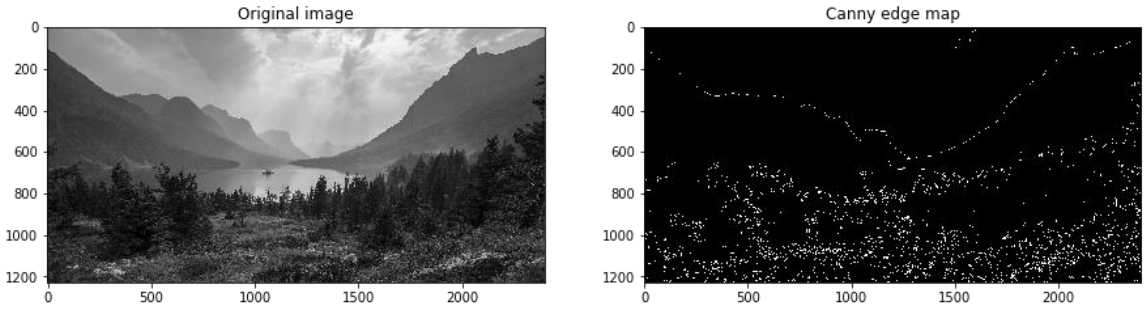

Figura II.8 - Algoritm de detecție Canny

În urma detecției marginilor și a reconstrucției mediului, utilizând tehnici cunoscute de clasificare a obiectelor, precum tehnica Single Shot Detector (SSD) - cea mai rapidă metodă de detectare a obiectelor dintr-o imagine, cu un singur strat al unei rețele convoluționale[7], pentru catalogarea obstacolelor întâlnite în mediu.

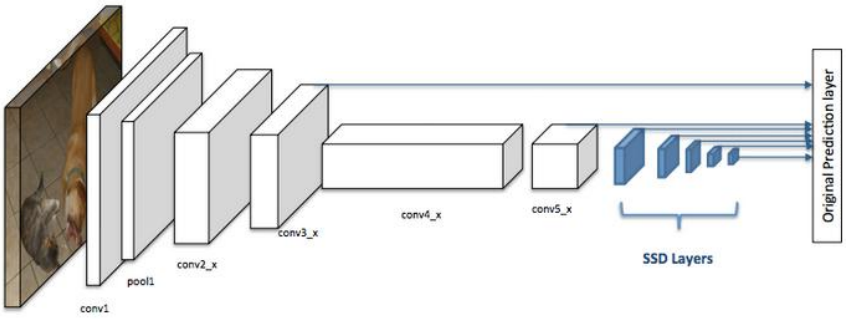

Figura II.9 - Algoritmul de detecțe SSD

În figura de mai sus se pot identifica două componente: un model “backbone” și un “SSD head”. Modelul backbone este de obicei o rețea pre-antrenată pentru clasificarea imaginilor, utilizată ca extractor de caracteristici. În mod tipic, aceasta este o rețea precum ResNet, antrenată pe ImageNet, din care stratul complet conectat de clasificare final a fost eliminat. Astfel, rămânem cu o rețea neurală profundă capabilă să extragă semnificația semantică din imaginea de intrare, păstrând în același timp structura spațială a imaginii, deși la o rezoluție mai mică.

5. Captarea punctelor de interes

Una dintre cele mai interesante tehnici de poziționare dezvoltate este ceea ce se numește în robotică Localizare și Cartografiere Simultană (SLAM). Soluția problemei SLAM a cunoscut o îmbunătățire rapidă în ultimele decenii, fie prin utilizarea senzorilor activi precum Radio Detection And Ranging (Radar) și Light Detection and Ranging (LiDAR), fie a senzorilor pasivi precum camerele.[8]

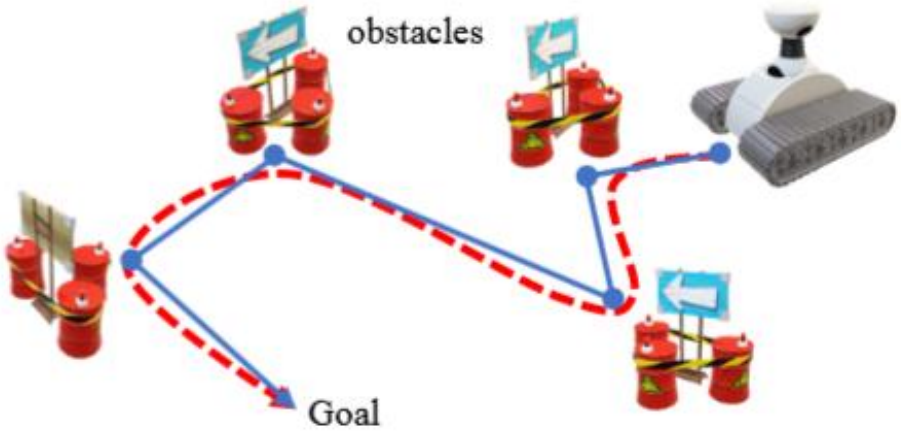

Figura II.10 - navigarea robotului dintr-un articol studiat utilizând SLAM

6. Planificarea rutei de deplasare și evitarea obstacolelor

Se identifică două tipuri de planificare a rutei optime – globală și locală.

Planificarea globală a traseului constă în proiectarea unui traseu sigur și lipsit de coliziuni pentru robot, de la punctul de plecare la punctul final, pe baza informațiilor disponibile despre mediu. Precizia traseului planificat depinde de acuratețea informațiilor despre mediu. Procesul de planificare globală a traseului poate fi considerat ca o căutare a soluției optime.[9]

O soluție studiată este GM (Grid Method), ce, în esență, presupune o împărțire a spațiului de lucru al unui robot mobil într-o serie de celule de rețea cu informații binare și de dimensiuni egale. Atunci când nu există obstacole într-o celulă, aceasta este numită celulă liberă, iar robotul mobil se poate deplasa liber. Când există un obstacol într-o celulă, chiar dacă obstacolul nu umple întreaga suprafață a celulei, cercetătorii folosesc, în general, metoda de extindere pentru a extinde obstacolul astfel încât să umple întreaga celulă și o numesc celulă cu obstacol. Celulele libere sunt de obicei marcate cu 0, iar celulele cu obstacole cu 1 (așa cum se arată în Fig. 3). Dimensiunea celulei este, în general, determinată în funcție de dimensiunea reală a robotului. Dacă dimensiunea celulei este mică, modelul de mediu va fi foarte clar, iar traseele planificate vor fi sigure. Cu toate

acestea, va ocupa mult spațiu de stocare al sistemului și va genera mai multe semnale de interferență, ceea ce va duce la un timp de planificare a traseului mai lung.[9]

Desigur, după detectarea obstacolelor, apoi vor fi aplicați algoritmi de detecție a rutei optime.

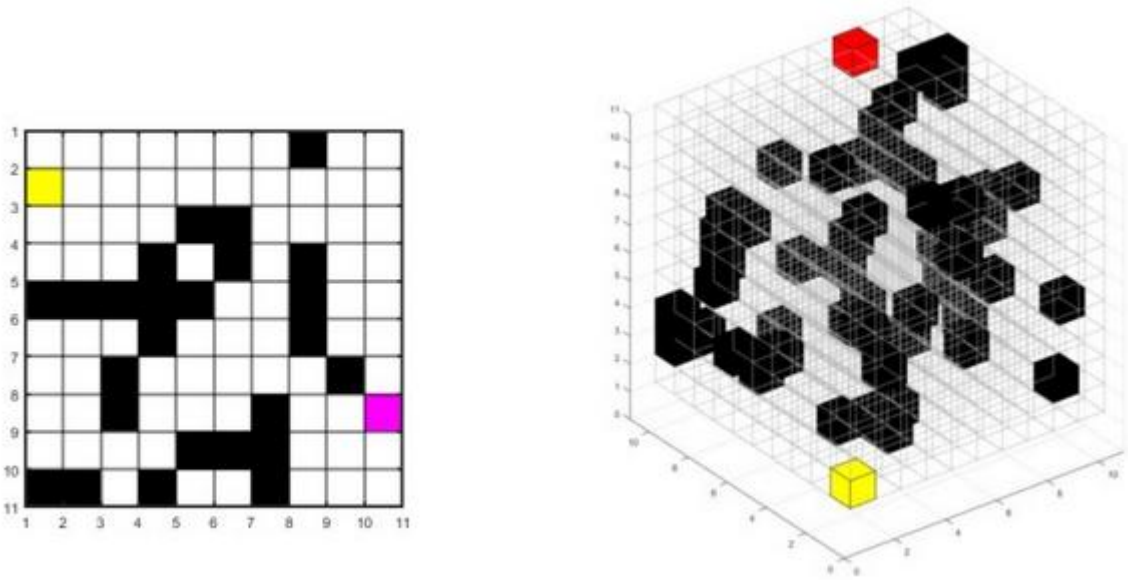

Figura 1- GM în mediul 2D și 3D

7. Simularea 3D

Pentru dezvoltarea unui robot mobil capabil să se adapteze mediului, acesta trebuie trecut printr-un proces de testare continuă pentru evaluarea performanțelor în diverse situații folosind dispozitive fizice. Pe langâ acestea, mai nou, a devenit standard folosirea simulărilor, astfel încât, dacă interacțiunea cu mediul virtual este destul de aproape de comportamentul real, procesul iterativ de evaluare poate fi făcut majoritar în software-ul 3D.[10] În timp, au apărut diverse soluții ce au fost adoptate la scară largă și sunt în continuă dezvoltare, precum ROS (Robot Operating Software) Gazebo, Webots, RoboDK, MATLAB/Simulink, NVIDIA Isaac și chiar și software ce a fost inițial dezvoltat pentru crearea jocurilor video - Unity și Unreal Engine.

În momentul în care un robot are o clona digitală cu scop de vizualizare sau chiar antrenare, aceasta se numește „geamăn digital”, sau „digital twin”.

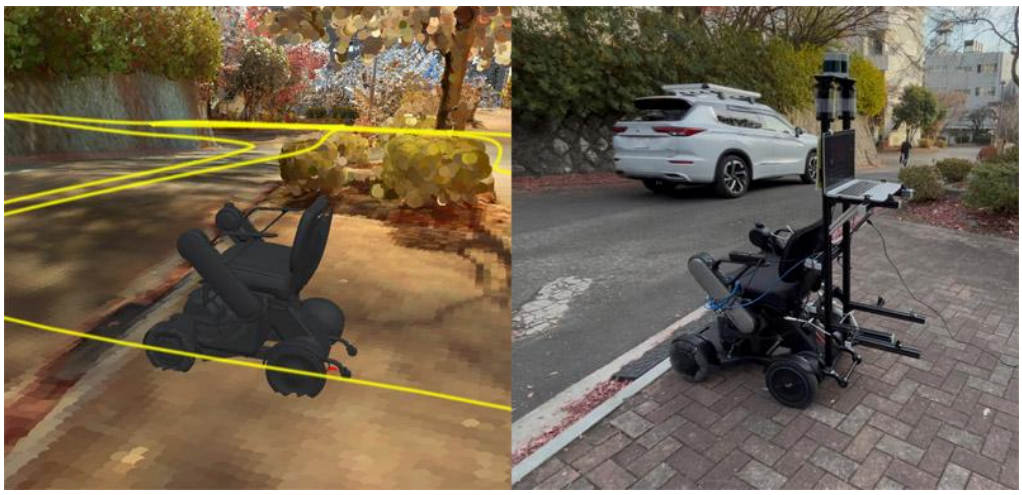

Figura II.11 - sincronizarea unui robot virtual și al celui fizic[10]

Utilă testării performanțelor și antrenării este o resursă capabilă să simuleze robotul în factori de mediu variabili. Pentru asta, special pentru software-ul ales, s-a dezvoltat Godot RL Agents, ce interfațează RLLib, o bibliotecă pentru antrenarea de tip Reinforcement Learning printr-o metodă nouă, ierarhică cu centre de control[11] și Stable Baselines, un alt tip de algoritm de Reinforcement Learning[12] cu Godot.[13]

Mai jos se pot observa diverse medii, 2D și 3D, în care un sistem poate fi antrenat.

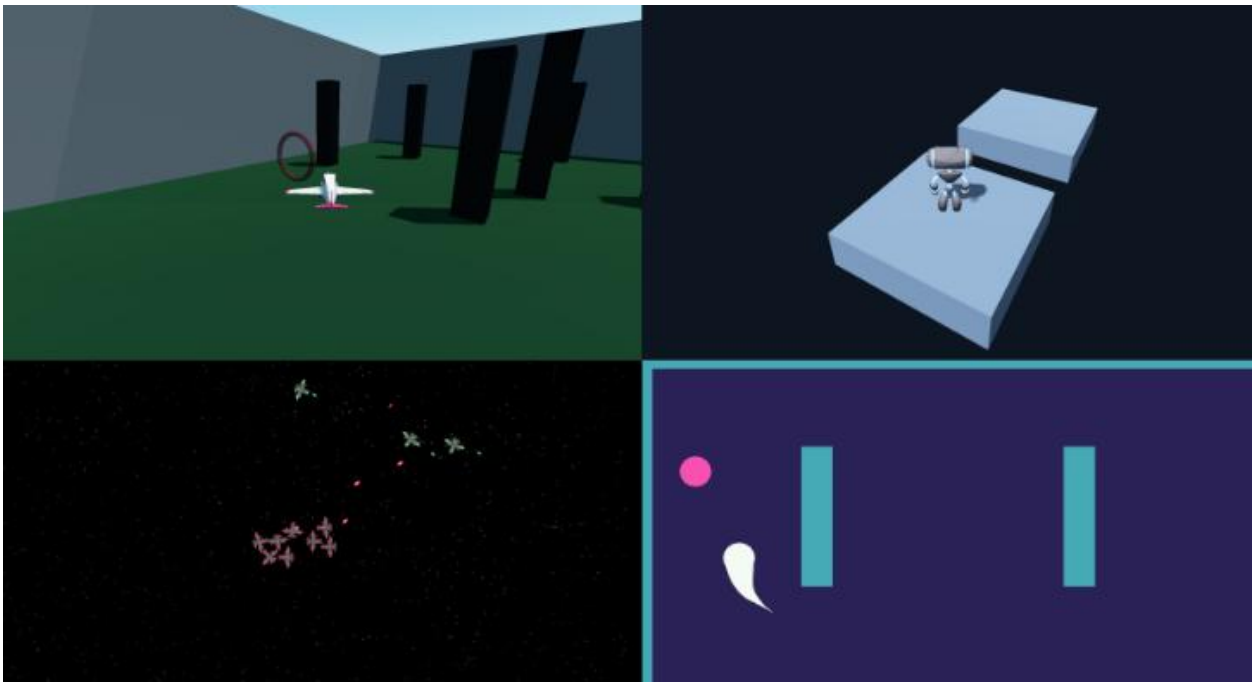

Figura II.12 - 4 cazuri de simulare - zbor autonom, navigarea terenului, evitarea inamicilor și țintirea automată de către o navă spațială și evitarea obstacolelor și culegerea recompenselor în timpi optimi[13]

III. Soluție proprie

Acest capitol detaliază conceptul, arhitectura și funcționalitatea soluției propuse - se va prezenta, precum a fost mentionat in introducere, un robot mobil bioinspirat, conceput pentru navigarea eficientă în mediul forestier, având ca scop principal facilitarea misiunilor de căutare și recuperare, precum și o extindere a capabilităților pentru cartografiere, monitorizarea calității solului și a vegetației. Accentul va fi pus pe inovația adusă în integrarea tehnologiilor de percepție, locomoție și inteligență artificială, acestea fiind simulate pentru dezvoltarea eficientă a unui sistem real cu o arhitectură de calcul distribuită, unde un Raspberry Pi 5 ar fi destul de eficient pentru gestiunea operațiunilor critice la bord, iar un sistem remote puternic preluând sarcini de analiză complexă. Se va prezenta întai conceptul robotului real și apoi metodologia simulării acestuia pentru a-i defini dimensiunile și metodele de adaptare și strabatere a mediului.

1. Concept general și arhitectura propusă

Arhitectura propusă se bazează pe o abordare stratificată, cu module distincte pentru percepție, control, planificare și interacțiune. O componentă cheie este arhitectura de calcul distribuită, unde Raspberry Pi 5 de la bordul robotului va îndeplini rolul de "creier" local, gestionând sarcinile critice în timp real, iar un sistem de calcul remote, mult mai puternic, va prelua procesarea intensivă a datelor, asigurând o scalabilitate și o profunzime de analiză superioare.

2. 1. Funcțiile produsului final, capabilități avansate pentru misiuni complexe

Robotul mobil bioinspirat este conceput pentru a îndeplini o serie de funcții esențiale, care îl diferențiază de platformele tradiționale și îl fac extrem de valoros pentru aplicațiile vizate:

- **Capacitate de Traversare pe Teren Neregulat**: Abilitatea de a naviga cu ușurință pe suprafețe denivelate, cu rădăcini expuse, bolovani, trunchiuri căzute și pante abrupte. Sistemul de locomoție va fi adaptabil la diverse morfologii ale terenului.
- **Agilitate în Spații Înguste**: Designul compact și controlul precis al mișcărilor picioarelor îi vor permite robotului să se strecoare prin vegetație densă, printre copaci apropiați și în locuri greu accesibile oamenilor sau vehiculelor pe roți/șenile.

- **Stabilitate și Echilibru Dinamic**: Indiferent de dificultatea terenului, robotul își va menține echilibrul, compensând înclinările și recuperându-se după perturbări, utilizând algoritmi avansați de control.
- **Deplasare Optimizată Energetic**: Algoritmii de generare a mersului și de control vor fi optimizați pentru a minimiza consumul energetic, maximizând autonomia robotului în medii izolate.

Se menționează că toate aspectele legate de controlul direct al locomoției, menținerea echilibrului, planificarea rutei pe termen scurt (evitarea obstacolelor imediate) și fuziunea senzorială esențială pentru odometrie vor fi gestionate de Raspberry Pi 5, asigurând un răspuns rapid și fiabil în mediu.

2.2. Cartografiere 3D inteligentă și reconstrucție parțială a mediului (pre-procesare locală)

Pentru a gestiona eficient cantitatea mare de date și a asigura relevanța informațiilor pentru misiune, abordarea propusă este una de "cartografiere inteligentă", cu o împărțire a sarcinilor:

- **Pre-procesare pe Raspberry Pi 5**: Norul de puncte generat de LiDAR va fi supus unei prime etape de filtrare și reducere a zgomotului. Se va realiza o cartografiere 3D locală, de rezoluție suficientă pentru navigația pe termen scurt, cu prioritizarea informațiilor utile pentru traversarea obstacolelor imediate (ex: crearea unei *costmap* locale). Localizarea robotului (SLAM) va fi efectuată continuu, bazându-se pe fuziunea datelor de la LiDAR, camere stereo și IMU.
- **Optimizarea densității norului de puncte (local)**: Se va realiza o sub-eșantionare inteligentă a norului de puncte, păstrând detaliile esențiale în zonele de interes apropiate robotului și reducând densitatea în zonele mai puțin relevante sau la distanțe mari, pentru a reduce volumul de date transmise.
- **Reconstrucție geometrică simplificată (local)**: Se va crea o reprezentare geometrică a mediului imediat înconjurător, suficient de precisă pentru a permite planificarea locală a mișcării. Aceasta poate fi un *grid map* 2.5D sau un *voxel map* simplificat.

2.3. Identificarea, clasificarea și înțelegerea obstacolelor

Procesarea imaginii va fi împărțită pentru a eficientiza resursele:

- **Pre-procesare pe Raspberry Pi 5**:
    - **Detecția marginilor (Canny)**: Aplicată imaginilor pentru a extrage contururile clare ale obiectelor, reducând zgomotul și concentrarea pe informațiile structurale.
    - **Filtrări și normalizări**: Optimizarea imaginilor pentru o transmisie eficientă și o pre-procesare ulterioară pe server.
    - **Segmentare de bază**: O segmentare simplă pentru a diferenția rapid între "teren traversabil" și "obstacol major", informând planificatorul local.
- **Analiză detaliată remote**: Datele vizuale și norii de puncte pre-procesați vor fi transmise către calculatorul puternic pentru:
    - **Clasificarea Obiectelor prin Deep Learning (SSD/YOLO)**: Recunoașterea detaliată și clasificarea speciilor de copaci, identificarea trunchiurilor căzute, a stâncilor, a corpurilor de apă, a potecilor.
    - **Estimarea Atributelor Obstacolelor**: Calcularea precisă a dimensiunilor, texturii, stabilității și materialului.
    - **Recunoaștere Avansată Obiecte**: Detecția siluetelor umane, a obiectelor asociate (rucsacuri, haine, echipament) cu o precizie și o robustețe superioare.

2.4. Planificare dinamică și adaptativă a rutei

Navigația eficientă într-un mediu complex necesită o abordare hibridă de planificare:

- **Planificare locală**: Modulul de navigație de pe Raspberry Pi 5 va fi responsabil pentru generarea rapidă a traiectoriilor scurte, de evitare a obstacolelor imediate, bazându-se pe harta locală generată și pe segmentarea de bază a obstacolelor. Acesta va asigura o mișcare fluidă și sigură în timp real.
- **Planificare globală remote**: Harta 3D detaliată (generată și optimizată pe serverul puternic) va fi utilizată pentru a calcula trasee optime pe termen lung. Aceste planuri globale vor fi transmise robotului ca o serie de waypoint-uri sau un ghid general, pe care planificatorul local îl va urma și ajusta. Această diviziune permite luarea unor decizii de navigare la nivel înalt, complexe, fără a supraîncărca resursele de la bord.

2.5. Monitorizarea mediului forestier și analiza calității (exclusiv remote)

Pentru a maximiza utilitatea robotului, se pot integra senzori suplimentari care să îi extindă capabilitățile dincolo de SAR:

- **Culegere date pe robot**: Senzori pentru pH, umiditate, conductivitate electrică, și chiar micro-senzori pentru prezența anumitor nutrienți sau poluanți. De asemenea, captarea de imagini multispectrale sau hiperspectrale.
- **Analiză complexă remote**: Toate aceste date vor fi transmise către calculatorul puternic pentru:
    - **Analiza calității solului**: Corelarea datelor cu baze de date geospațiale.
    - **Identificarea speciilor de copaci și monitorizarea sănătății vegetației**: Utilizarea algoritmilor de clasificare bazați pe AI, cu acces la baze de date extinse de imagini și semnături spectrale.

3. Descrierea funcționării detaliate a produsului final

Funcționarea robotului este un proces iterativ și continuu, organizat în etape logice care se interconectează pentru a asigura autonomia și adaptabilitatea, cu o separare clară a responsabilităților între sistemul local (Raspberry Pi 5) și cel remote.

3.1. Inițializare și configurare misiune

1. **Diagnosticare sistematică (local)**: La pornire, calculatorul local efectuează o autodiagnosticare completă a componentelor hardware (baterie, actuatoare, senzori) și a modulelor software de la bord.
2. **Încărcare parametri misiune (remote)**: Operatorul configurează misiunea prin intermediul stației de control (calculatorul puternic): puncte de start/destinație/waypoint-uri, priorități de mapare, profil de deplasare - transmise către Raspberry Pi.
3. **Calibrare inițială senzori (local)**: Verificarea și calibrarea senzorilor cheie (calibrare IMU, verificare bias GPS).

3.2. Percepția mediului - percepție continuă și pre-procesare locală

Robotul colectează date în mod continuu și sincronizat de la toate sursele senzoriale:

1. **LiDAR**: Scanează mediul, generând nori de puncte 3D. Raspberry Pi 5 realizează filtrarea zgomotului și downsampling-ul pentru a reduce volumul de date.
2. **Camere Stereo**: Capturează fluxuri video simultane. Raspberry Pi 5 realizează detecția marginilor (Canny) și compresia video pentru transmitere eficientă.
3. **GPS/IMU**: GPS-ul furnizează localizarea globală absolută. IMU-ul măsoară mișcarea robotului (accelerație, viteză unghiulară, orientare).

3.3. Procesarea datelor și înțelegerea contextului

Datele brute de la senzori sunt supuse unor procese avansate de prelucrare, cu o diviziune clară a sarcinilor:

Pe Raspberry Pi 5:

1. **Fuziunea Senzorilor și SLAM Local**: Datele de la LiDAR, camere și IMU sunt integrate pentru a realiza odometria robotului și a construi o hartă locală 3D de tip voxel. Aceasta este esențială pentru navigația imediată și evitarea coliziunilor.
    - **LiDAR odometry**: Estimează mișcarea robotului pe baza modificărilor din norul de puncte LiDAR pre-procesat.
    - **Visual odometry**: Estimează mișcarea pe baza imaginilor stereo pre-procesate.
    - **Fuziune IMU**: Corectează driftul și stabilizează estimarea poziției.
2. **Segmentare de bază a obstacolelor**: Pe baza datelor vizuale și LiDAR locale, se realizează o segmentare rapidă pentru a clasifica "teren traversabil", "obstacol mic/mare" și "zonă periculoasă", informând planificatorul local.

Pe calculatorul remote:

1. **Fuziunea senzorilor și SLAM global**: Datele pre-procesate (nori de puncte, imagini comprimate, estimări de odometrie locală) sunt transmise wireless către calculatorul puternic. Acesta realizează un SLAM global de înaltă rezoluție, construind o hartă 3D detaliată și consistentă pe termen lung a întregului mediu explorat (utilizând *Cartographer* sau soluții similare). Corecțiile de buclă închisă (*loop closure*) și optimizările globale sunt efectuate aici.

2. **Reconstrucție 3D avansată și optimizare hartă**: Norul de puncte global este transformat într-un model 3D optimizat (mesh), cu algoritmi avansați de reducere a poligoanelor, dar cu păstrarea detaliilor importante pentru analiza la distanță.
3. **Procesare vizuală avansată și inteligență artificială**:
    - **Clasificarea detaliată a obiectelor (SSD/YOLO)**: Utilizând modele AI complexe, calculatorul identifică și clasifică cu precizie obiecte precum specii de arbori, roci, tipuri de vegetație, structuri umane, animale etc.
    - **Estimarea atributelor și analiză semantică**: Se estimează atribute precum diametrul copacilor, tipul de sol, starea de sănătate a vegetației (dacă sunt folosiți senzori adecvați).
    - **Recunoaștere avansată SAR**: Modele AI dedicate analizează fluxurile video și termice pentru a identifica cu mare fiabilitate prezența victimelor sau a semnelor asociate.
4. **Bază de date mediu globală**: Toate aceste informații - harta 3D detaliată, clasificările, datele de monitorizare - sunt stocate într-o bază de date pe calculatorul remote.

3.4. Planificarea rutei în timp real

Planificarea rutei este gestionată printr-o colaborare între cele două sisteme:

1. **Planificare globală (remote)**: La începutul misiunii sau la schimbarea obiectivului, calculatorul puternic generează un traseu optim de la punctul curent la destinație, utilizând harta 3D globală și clasificarea detaliată a obstacolelor. Acest plan global este transmis către Raspberry Pi 5 ca o serie de waypoint-uri sau un traseu de referință.
2. **Planificare locală și evitarea obstacolelor dinamice (local)**: Raspberry Pi 5 utilizează harta locală actualizată și segmentarea de bază a obstacolelor pentru a genera traiectorii scurte și sigure. Acesta ajustează dinamic planul global în timp real pentru a ocoli obstacolele neprevăzute sau a gestiona terenul imediat. Dacă este necesară o deviere majoră, Raspberry Pi 5 poate semnala calculatorului puternic necesitatea unei replanificări globale.
3. **Adaptarea strategiei de mers (local)**: Planificatorul local colaborează cu modulul de control al locomoției pentru a alege mersul optim (crawl, trot) și parametrii acestuia

(lungimea pasului, frecvența) în funcție de caracteristicile locale ale terenului și de obstacolele întâlnite.

3.5. Controlul locomoției și interacțiunea cu terenul (local)

Acest modul traduce planul de mișcare în acțiuni fizice, asigurând o deplasare stabilă și eficientă. Toată această logică este implementată pe Raspberry Pi 5 pentru un răspuns rapid:

1. **Generare mers**: Algoritmi specifici generează pattern-uri ritmice de mișcare pentru cele 12 articulații ale picioarelor, coordonând ridicarea și plasarea fiecărui picior.
2. **Controlul echilibrului și stabilității (ZMP / SLIP / VMC)**: Sistemul monitorizează continuu centrul de masă al robotului și punctul de moment zero (ZMP). Algoritmii de control ajustează poziția picioarelor și forțele de contact pentru a menține stabilitatea.
3. **Cinematică inversă și directă**: Calculul unghiurilor articulațiilor pentru a plasa vârful piciorului în poziția dorită.
4. **Controlul forței și compensarea alunecării**: Senzori de forță și/sau odometrie vizuală permit robotului să detecteze alunecarea și să ajusteze rapid traiectoria piciorului sau cuplul aplicat.
5. **Comenzi actuatori**: Comenzile calculate sunt transmise direct servomotoarelor digitale.

3.6. Execuția misiunii și raportare continuă (comunicare hibrida)

Pe parcursul întregii misiuni, robotul operează autonom, menținând o legătură constantă cu operatorul prin sistemul remote.

1. **Culegere date specifice misiunii:** Robotul culege datele brute și pre-procesează informațiile critice.
2. **Transmisie wireless:** Datele pre-procesate (nori de puncte reduși, imagini comprimate, alerte preliminare, odometrie) sunt transmise wireless către punctul de procesare.
3. **Analiză detaliată și generare alerte:** Calculatorul remote realizează analiza complexă (detecție precisă a victimelor, clasificare detaliată a mediului, monitorizare ecologică) și generează alerte precise.

4. **Transmitere informații critice către operator:** Odată ce o țintă este detectată, calculatorul remote transmite instantaneu către operator coordonatele GPS precise, imagini video (sau termice), și orice alte informații relevante.
5. **Streaming video/hărți**: Operatorul vizualizează hărțile 3D detaliate și fluxurile video live din locația stației remote, permițând o monitorizare completă.
6. **Raport final si analiză cost:** La finalizarea misiunii, calculatorul compilează un raport detaliat cu traseul parcurs, evenimentele cheie, datele colectate și detectările relevante, pentru o analiză ulterioară și pentru îmbunătățirea continuă a sistemului.

4. Echipamente Necesare

Acest capitol va detalia lista echipamentelor necesare pentru construcția și funcționarea robotului mobil bioinspirat, incluzând componente mecanice, electronice și software. Se vor oferi descrieri și, pe cât posibil, imagini și fișe tehnice sintetizate.

4.1. Componente electronice și senzori

Designul structural al robotului patruped, cu 12 grade de libertate, este esențial pentru o locomoție reușită și va fi folosit ca bază. Se vor utiliza materiale ușoare, dar rezistente, pentru a optimiza raportul greutate-rezistență. Deoarece ținta acestei cercetari este mai degrabă gasirea soluțiilor și optimizarea pentru detecție și navigare, întâi se vor prezenta componentele electrice necesare ce permit procesarea

1. **Unitate de Control Principală (Microcontroller/SBC):**

Descriere: Creierul robotului, responsabil pentru interpretarea datelor senzorilor, execuția algoritmilor de control (SLAM, planificare rută, generare mers), și comunicarea cu actuatori și alte module.

O combinație de Raspberry Pi 5 (pentru control general) și o placă de dezvoltare FPGA/GPU pentru accelerarea algoritmilor SLAM și procesare imagine.

Figura 2 - Raspberry Pi 5

Figura 3 - NVIDIA Jetson AGX Xavier

GPU: 512-core Volta GPU with 64 Tensor Cores

CPU: 8-core ARM v8.2 64-bit Carmel CPU

Memorie: 32 GB 256-bit LPDDR4x

Stocare: 32 GB eMMC 5.1

Conectivitate: Gigabit Ethernet, PCIe Gen4, USB 3.1, HDMI, DisplayPort

Capacitate de calcul AI: 32 TOPS (Tera Operations Per Second)

2. **Sistem de Senzori pentru Percepția Mediului:**

LiDAR (Light Detection and Ranging):

Un senzor activ care emite impulsuri laser și măsoară timpul necesar pentru ca lumina să se întoarcă, creând un nor de puncte 3D al mediului. Crucial pentru cartografierea precisă a topologiei forestiere.

Se propune crearea unui senzor LiDAR rudimentar de la 0.

Camere Stereo OV2640**:**

Două camere calibrate care, prin stereoviziune, permit calcularea distanței până la obiecte și crearea unei hărți de adâncime. Utilizate pentru detecția marginilor și clasificarea obiectelor.

Tabel 1 - specificații OV2640

Rezoluție maximă	1600 x 1200 pixeli (UXGA, 2 Megapixeli)
Format imagine	JPEG, RGB565, YUV422, YCbCr422, RAW10
Interfață	SCCB (similar I2C) + DVP (Digital Video Port, 8 bit paralel)
Dimensiune senzor	1/4 inch
Pixel size	2.2 µm x 2.2 µm
Lentilă	Focală fixă, opțională cu unghi larg sau IR-cut
Framerate (cadre/s)	15 fps la UXGA, până la 30 fps la rezoluții mai mici
Iluminare minimă	1.3 V/lux-sec

Control automat	Auto exposure, auto white balance, auto gain
Dimensiuni fizice	Aproximativ 20 x 20 mm (depinde de modul)
Tensiune operare	2.5V – 3.0V pentru nucleu, 1.8V sau 2.5V I/O (cu LDO)
Consum	Foarte redus (< 100 mW în funcționare normală)

Sistem GPS/IMU (Inertial Measurement Unit):

Descriere: GPS-ul oferă localizarea globală, iar IMU-ul (accelerometru, giroscop, magnetometru) măsoară orientarea, viteza unghiulară și accelerația, esențiale pentru estimarea poziției și controlul echilibrului.

Se poate utiliza Swift Navigation Piksi Multi GNSS RTK Receiver cu o unitate IMU integrată.

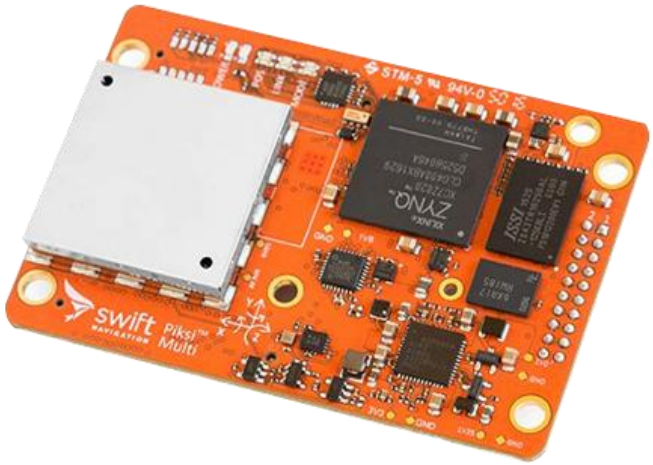

Figura 4 - Swift Navigation Piksi Multi GNSS RTK

Suport GNSS: GPS L1/L2, GLONASS L1/L2, BeiDou B1/B2, Galileo E1/E5b, QZSS L1/L2

Precizie RTK: Orizontal 1cm + 1ppm, Vertical 2cm + 1ppm

Rată de actualizare: Până la 50 Hz

IMU: Accelerometru, giroscop, magnetometru

3. Baterii și Sistem de Management al Energiei (BMS):

Asigură alimentarea cu energie a tuturor componentelor robotului, influențând autonomia. BMS-ul monitorizează și gestionează încărcarea, descărcarea și starea generală a bateriei.

Se propun baterii LiPo de înaltă densitate energetică, cu o capacitate adecvată pentru o autonomie de cel puțin 2-4 ore.

Specificații pachet LiPo 6S 22000mAh:

Tensiune nominală: 22.2V

Capacitate: 22000mAh

Curent de descărcare continuu: 25C (550A)

Greutate: ~2.5 kg

Conector: XT90 sau similar

Module de Comunicare Wireless:

Descriere: Permite comunicarea datelor și a comenzilor între robot și o stație de control la distanță.

Exemplu: Modem radio de înaltă frecvență (ex: XBee Pro 900HP) pentru telemetrie și control, și/sau modul 4G/5G pentru transmiterea datelor pe distanțe lungi.

Figura 5 - XBee Pro 900HP

Frecvență: 902-928 MHz

Putere de ieșire: 24 dBm (250 mW)

Rază de acțiune: Până la 14 km în linie vizuală (depinde de antenă și obstacole)

Rată de date: Până la 200 Kbps

Interfață: UART, SPI, I2C

4. Software și Medii de Dezvoltare

Godot comunicând cu programe create în Python, prin biblioteca Websockets.

Caracteristici:

- Modularitate - fiecare obiect creat este un “nod”
- Simplitate ce permite simularea senzorilor într-un mod foarte eficient
- Permite vizualizarea în timp real
- Comunitate vastă și resurse open-source.

Biblioteci pentru Procesare Imagine și Viziune Artificială:

Seturi de instrumente software pentru prelucrarea datelor vizuale: OpenCV (pentru detecția marginilor Canny), PyTorch/TensorFlow (pentru implementarea SSD și alte rețele neuronale de clasificare).

Medii de Simulare 3D:

Descriere: Platforme software care permit testarea și validarea algoritmilor de control și a designului robotului într-un mediu virtual, înainte de implementarea pe hardware real. Acest lucru reduce costurile și riscurile asociate dezvoltării.

Godot, Blender (pentru modelare 3D și prelucrare nori de puncte), MeshLab (pentru reconstrucție suprafață).

- Vizualizare și analiză a datelor.
- Simularea senzorilor (LiDAR, camere).
- Import de modele 3D (ex: .obj din Blender).

Limbaje de Programare:

Python (pentru prototipare rapidă, procesare imagine, AI), GdScript (limbaj specific Godot, utilizând la bază C++ și fiind structurat în același fel precum Python), micropython.

Se ia în considerare următorul grafic ce afișează ordinea proceselor și coordonarea dintre robot și stația remote de control și procesare

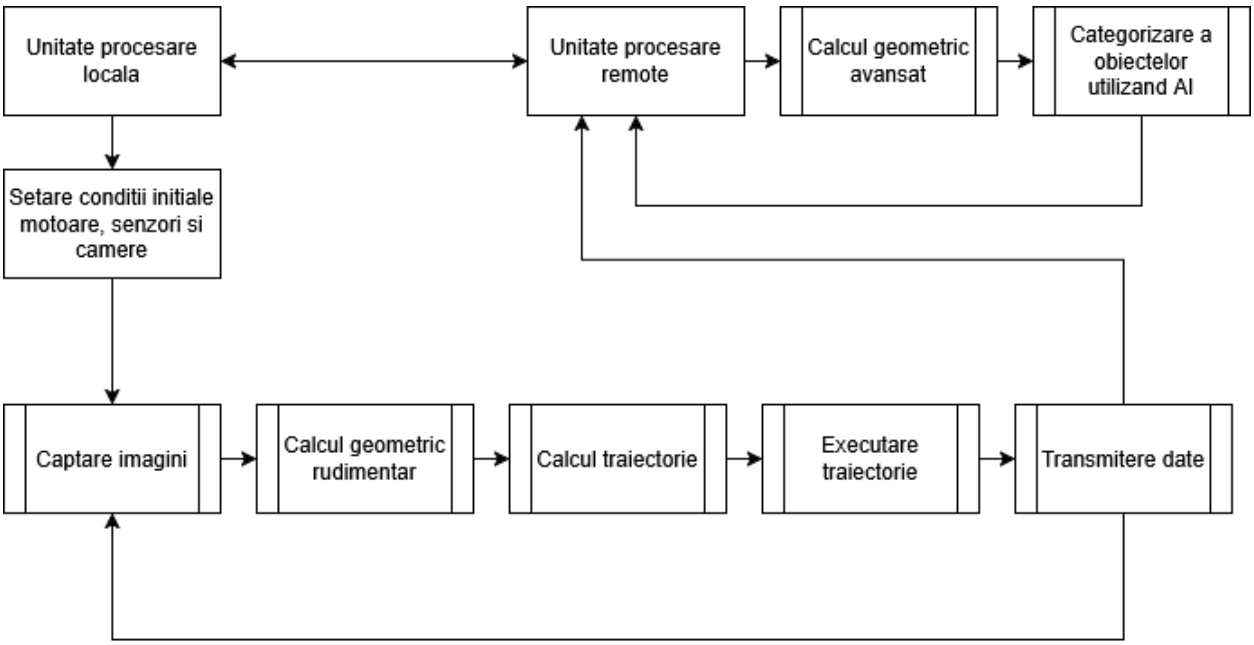

Figura III.1 - Coordonarea proceselor sistemului

IV. Rezultate experimentale

Întrucât este nevoie de procesarea automată a punctelor și transpunerea într-o suprafață, s-a inceput cu niște teste manuale - folosind aplicația 3D Forest și câteva fișiere de test ce conțin înregistrări ale unei secțiuni forestiere de 20x40m, s-a expus grafic un nor de puncte captat prin LiDAR. Ce se dorește este transformarea punctelor în fețe triunghiulare pentru a putea prelucra modelele și a extrage diverse caracteristici din mediu.

Figura IV.1 - Secțiunea de 20x40m de pădure

Figura IV.2 - un singur copac izolat. Acesta va fi transformat într-un model 3D

În urma extragerii unui singur copac din mediul de lucru, acesta a fost importat în Blender, exportat cu extensia .obj, ca apoi să fie importat în aplicația MeshLab, unde am aplicat un filtru ce recalculează normalele, astfel încât lumina să funcționeze corect când cade pe modelul 3D. Apoi am aplicat un filtru de reconstrucție numit Ball Mesh Pivoting. Se poate observa rezultatul mai jos.

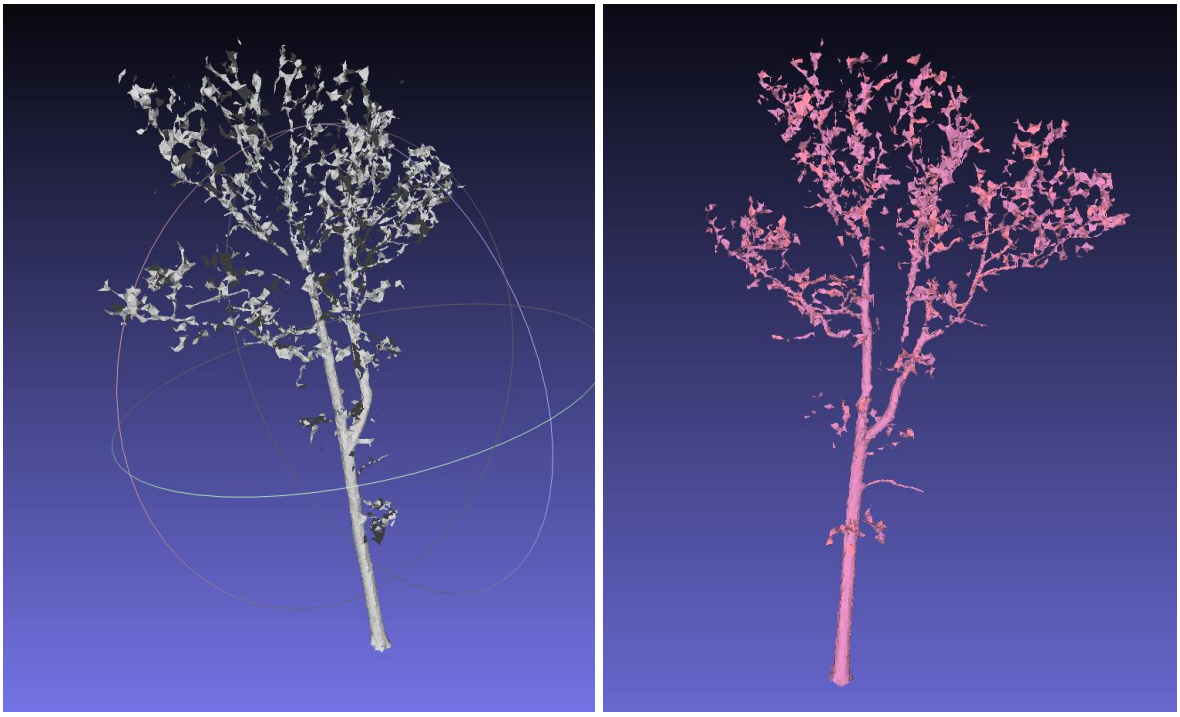

Figura 6 - Reconstrucție utilizând MeshLab 2023

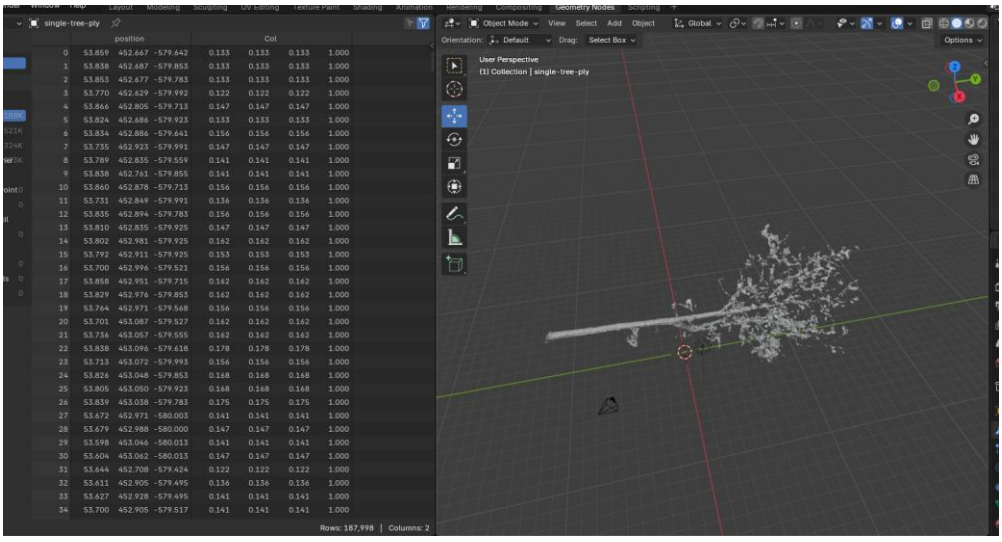

Figura 7 - Preview Blender

Următorii pași, în mod natural, sunt captarea și apoi procesarea automată a punctelor din mediu - procese ce vor fi executate în acest studiu exclusiv într-un mediu de simulare, pentru a permite un control mai ușor și rapid al factorilor de mediu, dar și pentru un proces de integrare și antrenare facil al sistemului bioinspirat. Aceste două procese vor fi executate simultan de un LiDar și un sistem de camere stereo, amândouă integrate digital în mediul de simulare, ca apoi să fie cuplate pentru a regenera mediul încojurător la o rezoluție utilă pentru parcurgerea terenului și înregistrarea informațiilor legate de mediu.

Mediul de simulare, fiind în mod normal un mediu de dezvoltare a jocurilor video, este mult mai rapid în etapa de inițializare și necesită puțini parametri, astfel fiind ideal pentru prototiparea rapidă, dezvoltarea unui sistem robotic și analiza comportamentului său.

Pentru ușurarea procesului de integrare a sistemelor în mediul de simulare, s-a executat următoarea schemă, ce propune urmarea unui curs stabilit de lucru:

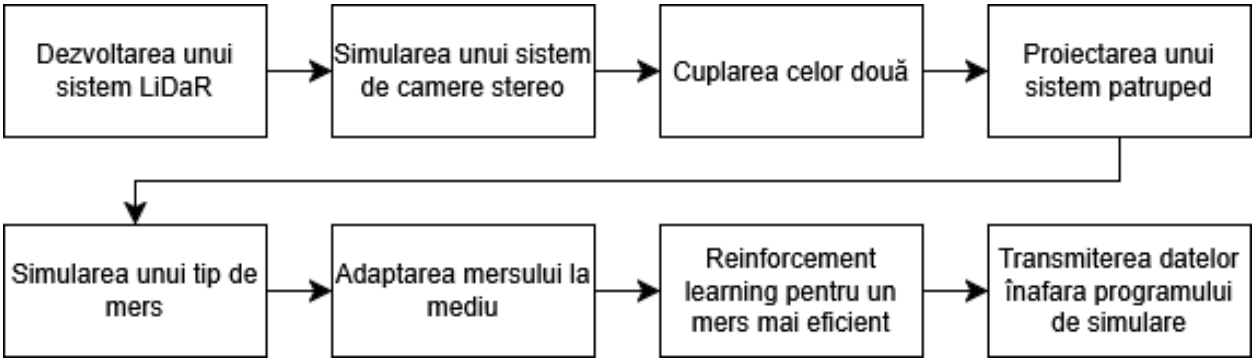

Figura IV.3 - Proces de integrare în mediul de simulare

1. **Dezvoltarea unui sistem LiDaR**

Replicarea unui modul lidar - viteza de rotație, număr de raze, distanță maximă pentru acuratețe, distanță minimă etc - 1 raycast sau un array ce se invarte la o viteză stabilită - undeva între 300 și 1200 rpm (5-20Hz), cu 32, 64 sau 128 de canale (raze verticale), 0.05grd - 0.2grd inclinatie de la un canal la altul

Metodologie

Pentru teste inițiale, se adaugă un cub “robot” într-un mediu pe care il vom numi “mediul activ”, sau prescurtat, “MA”. Acesta va avea atașată o cameră controlabilă, pentru debugging, ce îl va urma de aproape. Din locația robotului se poate proiecta, apăsând o tastă, o rază detectoare până în locația pe care este pus mouse-ul.

Mai jos se poate observa mediul de simulare, cubul ce reprezintă poziția robotului și un obstacol poziționat pe un plan.

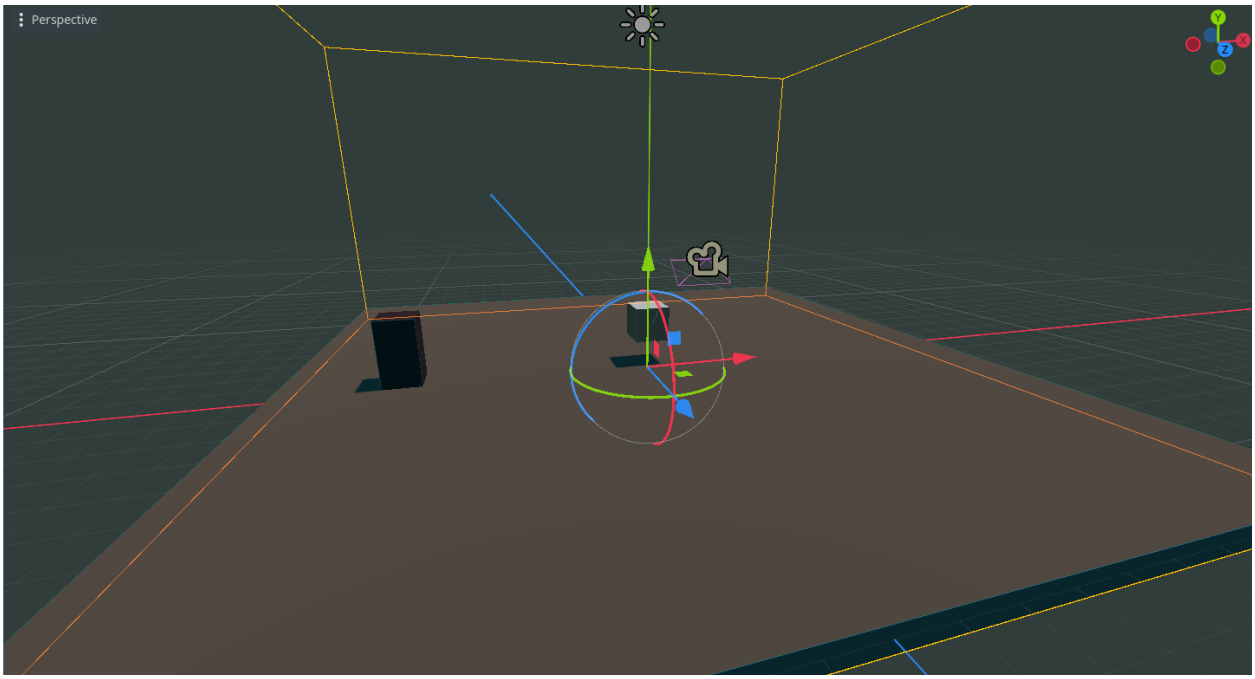

Figura IV.4 - Mediul de simulare

Pentru control optim al mediului de lucru, am creat un sistem de control al camerei principale ce se ocupa cu vizualizarea întregii scene, astfel încât poate fi poziționată în mod precis oriunde. De asemenea, robotul, deoarece coliziunile cu caracter realistic nu sunt înca de interes, prezintă o schemă simplă de control de tip vector între axele X și Z dependentă de rotația camerei, iar pe Y poate sări.

Am menționat proiectarea unei raze detectoare de la locația robotului până la locul în care se află mouse-ul. Aceasta simulează un singur puls al unui sistem LiDaR, dar nu este o abordare eficientă pentru stocarea și vizualizarea punctelor în momentul în care se doresc mai mult de 500 de puncte concomitent. Mai jos se va vorbi despre o metodă mult mai eficientă, scalabilă, ce va fi folosită până la finalul studiului.

În figura următoare se pot observa o sfera verde ce reprezintă cercul de pivot al camerei, ce este mobil, vizibil fiind din motive de control facil, obstacolul anterior, cubul robot, o rază verde ce afișează direcția în care se deplasează și raza roșie proiectată menționată anterior.

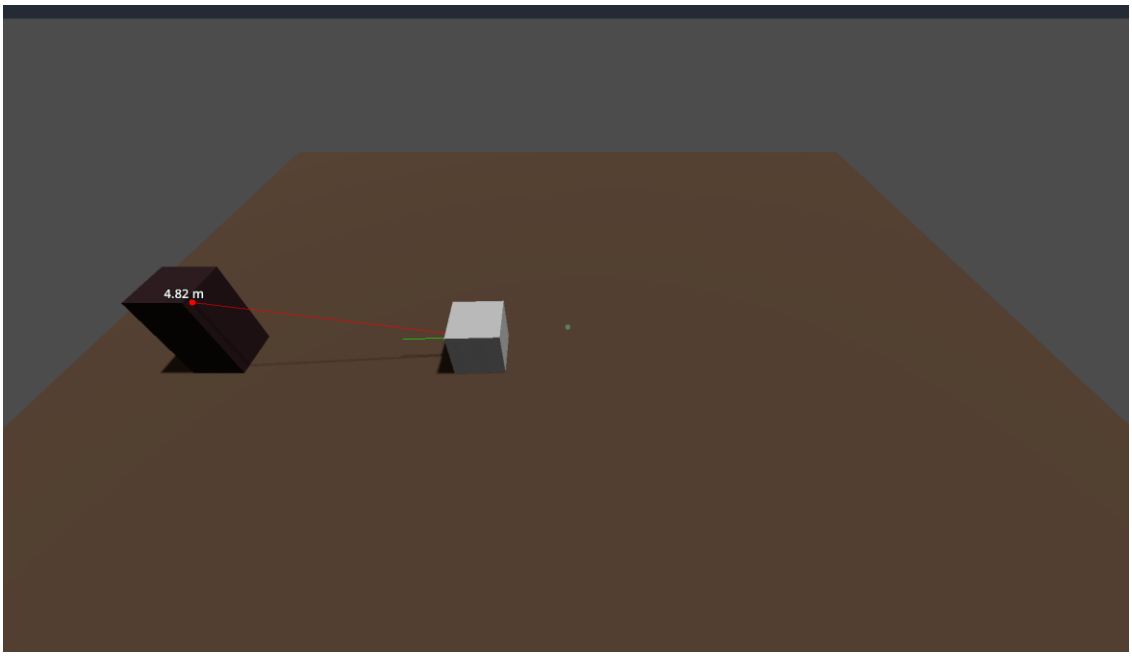

Figura IV.5 - Proiectarea razei

Pentru simularea funcționalității unui LiDaR, raza laser ar trebui să își schimbe unghiul pe axa Z o dată la fiecare rotație completă pe axa Y, astfel survolând terenul din imprejurimile robotului pentru a capta cât mai multe puncte din mediu, pentru a-l reconstrui apoi în memorie. Problema intervine când vrem o scanare fidelă realității - rapidă, cu multe puncte captate. În mod normal, punctele nu sunt vizualizate instantaneu și de aceea un astfel de modul poate în mod normal să ruleze la eficiență sporită.

Pentru o simulare eficientă, nu se pot folosi metode obișnuite de proiecție, deoarece fiecare punct înseamnă un obiect ce necesită un schimb de date individual între procesor și placa video. Acest proces este în schimb executat prin adăugarea unui set de instrucțiuni penru cum trebuiesc împachetate informațiile, ca ele să fie trimise către placa grafică laolaltă. După interceptarea fiecărui punct lovit, printr-o metodă convențională de scanare a împrejurimilor, unde aceeași rază precum cea roșie de adineauri survolează în jurul robotului, punctele sunt stocate și teleportate în spațiu instantaneu.

Mai jos se poate observa proiecția punctelor pe terenul plat și pe obstacol, fiecare punct galben fiind reprezentat de coordonatele globale XYZ și păstrând informație legată de reflectivitate / intensitate, I.

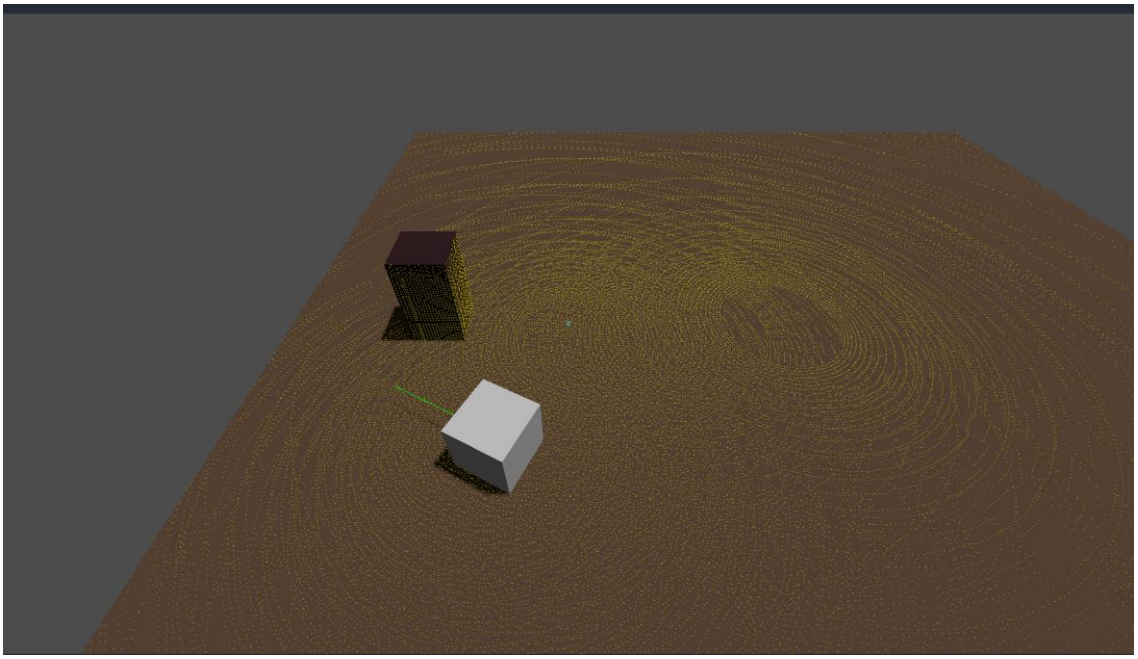

Figura IV.6 - Proiecția punctelor în mediu

Pe lângă captarea punctelor, am realizat și înregistrarea lor într-un fișier .txt. Aceasta a fost realizată printr-un pod de comunicare Websockets între un program Python bazat într-un mediu virtual (utilizarea consolei în următoarea imagine) și simularea din software-ul de simulare (în partea dreaptă).

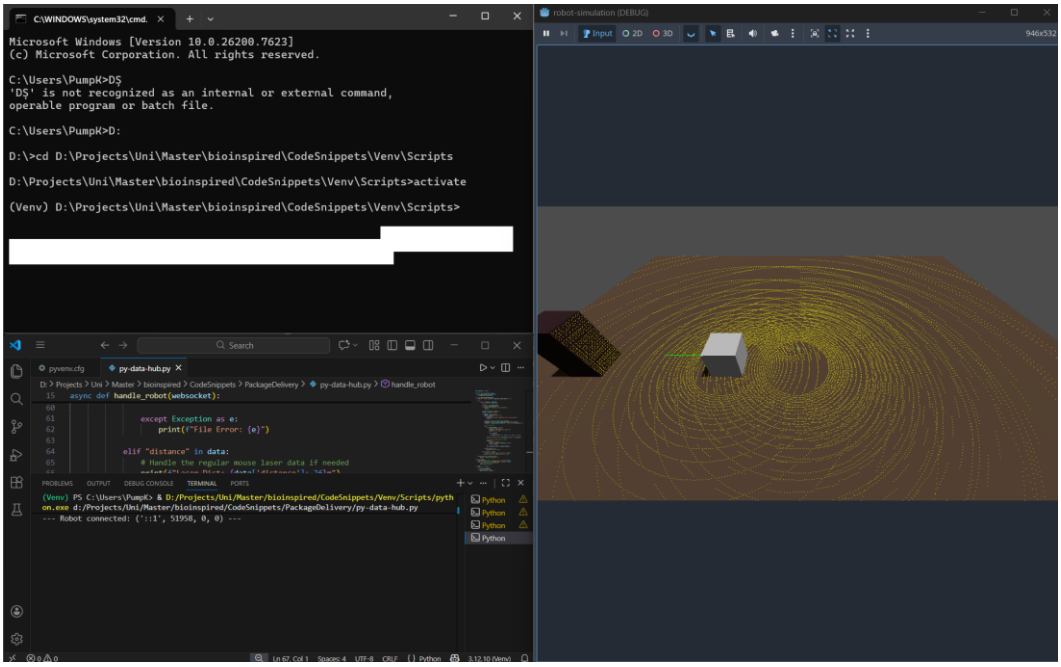

Figura IV.7 - comunicarea Python - Godot

Utilizând norul de puncte XYZI, se poate recompune mediul înconjurător și ulterior cupla cu o cameră stereo pentru potrivirea coordonatelor și identificarea caracteristicilor din mediu pentru procesul de recunoaștere automată și navigare mai eficientă. Camera stereo va fi simulată în același software și va integra algoritmul de detecție a marginilor Canny.

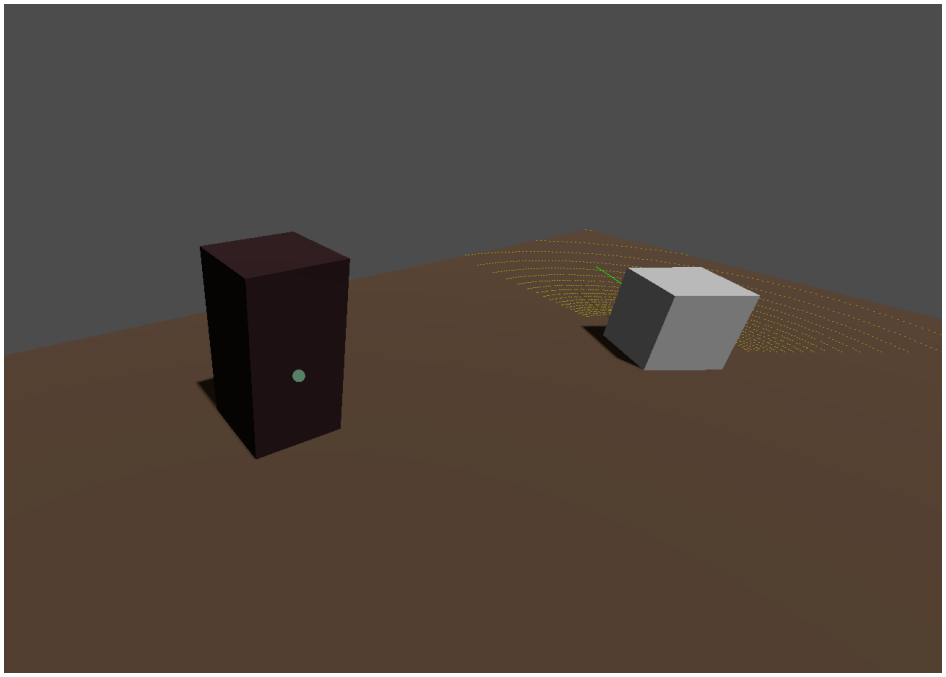

Figura IV.8 - imaginea de test

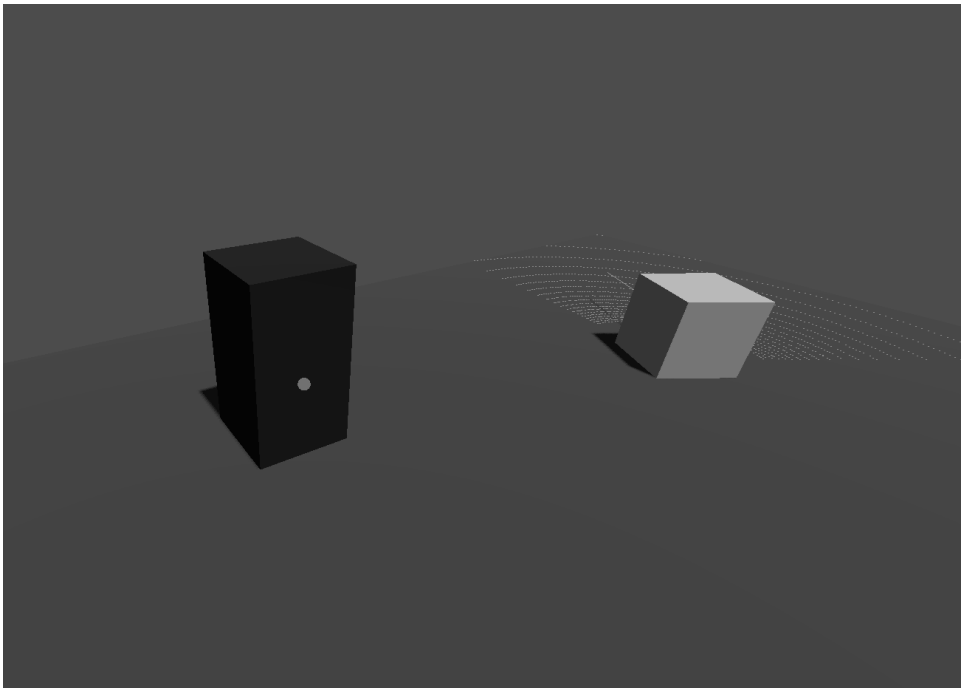

Figura IV.9 - imaginea de test fară culori

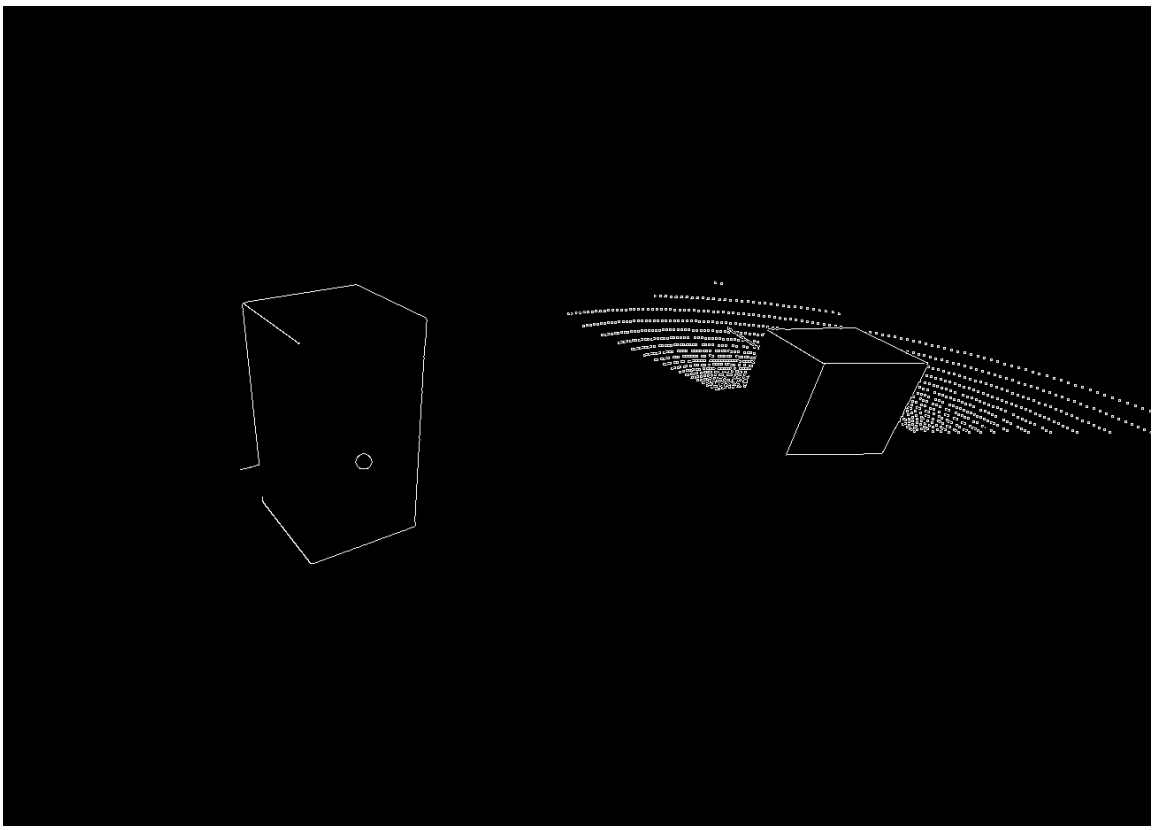

Figura IV.10 - imaginea de test, margini detectate utilizând Canny

Bibliografie

[1] Shi Y, Li S, Guo M, Yang Y, Xia D, Luo X. Structural Design, Simulation and Experiment of Quadruped Robot. *Applied Sciences*. 2021; 11(22):10705.

[2] Rajat Kolhe , Kanaiya Gadhavi, Nikhil Koli, Swati Gurav and Prasanna Raut. (2020). Overview of Different Techniques Utilized in Designing of a Legged Robot. *Global Journal of Enterprise Information System*, *9*(1), 36-41.

[3] Bruno Siciliano and Oussama Khatib. 2007. Springer Handbook of Robotics. Springer-Verlag, Berlin, Heidelberg.

[4] Parra Ricaurte EA, Pareja J, Dominguez S, Rossi C. Comparison of leg dynamic models for quadrupedal robots with compliant backbone. Sci Rep. 2022 Aug 26;12(1):14579. doi: 10.1038/s41598-022-18536-7.

[5] Rashmi, & Saxena, Rohini. (2013). Algorithm and Technique on Various Edge Detection : A Survey. Signal & Image Processing : An International Journal. 4. 65-75. 10.5121/sipij.2013.4306.

[6] Rezvanian, Alireza & Vahidipour, S Mehdi & Sadollah, Ali. (2023). An Overview of Ant Colony Optimization Algorithms for Dynamic Optimization Problems.

[7] Vaishnavi, K. & Reddy, G. & Reddy, T. & Iyengar, N. & Shaik, Subhani. (2023). Real-time Object Detection Using Deep Learning. Journal of Advances in Mathematics and Computer Science. 38. 24-32. 10.9734/jamcs/2023/v38i81787.

[8] Alsadik, Bashar & Karam, Samer. (2021). The Simultaneous Localization and Mapping (SLAM)-An Overview. Journal of Applied Science and Technology Trends. 2. 120-131. 10.38094/jastt204117.

[9] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, Pengfei Wang, Path planning techniques for mobile robots: Review and prospect, Expert Systems with Applications, Volume 227, 2023, 120254, ISSN 0957-4174.

[10] S. Kurebayashi, T. Tomizawa, and S. Tarao, “Game-Engine-Based 3D Simulation of Mobile Robot and its Application to Autonomous Navigation in Physical Environments,” J. Robot. Mechatron., Vol.37 No.6, pp. 1314-1326, 2025.

[11] Liang, Eric, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. "RLlib: Abstractions for distributed reinforcement learning." In *International conference on machine learning*, pp. 3053-3062. PMLR, 2018.

[12] Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines.

[13] Beeching, Edward, Jilles Debangoye, Olivier Simonin, and Christian Wolf. "Godot reinforcement learning agents." *arXiv preprint arXiv:2112.03636* (2021).