UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI
MINISTERUL MUNCII, FAMILIEI ȘI PROTECȚIEI SOCIALE
AMPOSDRU

Fondul Social European
POSDRU 2007-2013

Instrumente Structurale
2007-2013

MINISTERUL EDUCAȚIEI CERCETĂRII TINERETULUI ȘI SPORTULUI
OIPOSDRU

UNIVERSITAS GALATIENSIS

UNIVERSITATEA "DUNĂREA DE JOS" DIN GALAȚI

ŞCOALA DOCTORALĂ A FACULTĂȚII DE AUTOMATICĂ, CALCULATOARE, INGINERIE ELECTRICĂ ȘI ELECTRONICĂ

Teză de doctorat

CONTRIBUTII LA CONDUCEREA, NAVIGAȚIA ȘI EVITAREA DE OBSTACOLE A ROBOȚILOR MOBILI ȘI VEHICULELOR AUTONOME

Conducător ştiinţific
Prof. Univ.Dr. Ing. Adrian Filipescu

Doctorand
Ing. Bogdan Dumitrașcu

Galați
2012

Investeşte în oameni !

FONDUL SOCIAL EUROPEAN
Programul Operaţional Sectorial Dezvoltarea Resurselor Umane 2007 – 2013
Axa prioritară 1 ”Educaţie şi formare profesională în sprijinul creşterii economice şi dezvoltării societăţii bazate pe cunoastere”
Domeniul major de intervenţie 1.5 ”Programe doctorale şi postdoctorale în sprijinul cercetării”
Titlul proiectului: Eficientizarea activităţii studentilor din cadrul ciclului de studii doctorale - EFICIENT
Numărul de identificare al contractului: POSDRU/88/1.5/S/61445

TEZĂ DE DOCTORAT

CONTRIBUȚII LA CONDUCEREA, NAVIGAȚIA ȘI EVITAREA DE OBSTACOLE A ROBOȚILOR MOBILI ȘI VEHICULELOR AUTONOME

Domeniul: *Ingineria Sistemelor*

Doctorand: Ing. Bogdan DUMITRASCU

Componența Comisiei de doctorat:

PREȘEDINTE: Conf.univ.dr.ing. Emilia PECHEANU
Universitatea “Dunărea de Jos” din Galați

CONDUCĂTOR ȘTIINȚIFIC: Prof.univ.dr.ing. Adrian FILIPESCU
Universitatea “Dunărea de Jos” din Galați

REFERENT OFICIAL: Prof.univ.dr.ing. Corneliu LAZĂR
Universitatea Tehnică “Gheorghe Asachi” din Iași

REFERENT OFICIAL: Prof.univ.dr.ing. Octavian-Cezar PĂSTRĂVANU
Universitatea Tehnică “Gheorghe Asachi” din Iași

REFERENT OFICIAL: Conf.univ.dr.ing. Eugenia MINCĂ
Universitatea “Valahia” din Târgoviște

Cuprins

Prefață

Mulțumesc tuturor celor care m-au încurajat și ajutat la elaborarea tezei de doctorat.

În primul rând îi mulțumesc profesorului coordonator, dr. ing. Adrian Filipescu, pentru oportunitatea de a urma studiile de doctorat la Universitatea "Dunarea de Jos" din Galați și pentru ajutorul și cunoștintele fără de care această teză nu ar fi putut fi elaborată.

Îi mulțumesc domnului profesor Urbano Nunes, de la Universitatea din Coimbra, Institutul de Sisteme și Roboți pentru ajutorul acordat în perioada stagiului extern. Ajutorul dumnealui a făcut posibilă folosirea senzorilor inerțiali.

Mulțumesc domnului profesor Octavian-Cezar Păstrăvanu, domnului profesor Cornelui Lazăr, doamnei conferențiar Eugenia Mincă pentru ajutorul în elaborarea tezei de doctorat prin observațiile făcute în calitate de referenți oficiali.

Cercetarea și teza de doctorat s-a făcut cu suportul logistic și financiar din cadrul proiectului Eficientizarea activității studenților din cadrul ciclului de studii doctorale-EFICIENT, Cod proiect POSDRU/88/1.5/S/61445, Id proiect: 61445.

Cercetarea și teza de doctorat s-a facut cu suportul logistic și financiar din cadrul proiectului UEFISCDI-IDEI, număr proiect PN-II-ID-PCE-2011-3-0641.

Doresc să mulțumesc părinților pentru sprijinul și încurajările acordate.

Abstract

Mobile robots and autonomous vehicles have become very important research topics due to their multitude of possible uses and the technological progress that made it possible to build increasingly more complex robots at cheaper prices. Autonomous vehicles have the potential of eliminating the accidents caused by human errors, which represent about 90 percent of all accidents.

Both WMRs(Wheeled Mobile Robots) and AVs(Autonomous Vehicles) need reliable information about their position and heading, velocity and distance to obstacles. For this reason, sensors like the GPS, sonar sensors, encoders, lasers and IMU(Inertial Measurement Units) are used. An indirect Kalman filter for the estimation of the attitude of vehicles using the data from the IMU has been presented in this thesis.

The calculation of the kinematic models for 2DW/2FW robots and 4DW/SW autonomous vehicles has been presented.

The trajectory-tracking problem for WMRs and AVs has been solved using sliding-mode control in continuous time, sliding-mode control in discrete-time and backstepping control.

An algorithm for obstacle avoidance of WMRs using sonar sensors to detect obstacles and one of the proposed control methods to follow an imposed trajectory and only alter course to avoid an obstacle and return to the initial trajectory afterwards.

An algorithm for obstacle avoidance of autonomous vehicles using lasers to detect obstacles and one of the control methods that is proposed above, to follow an imposed trajectory and only alter course to avoid an obstacle and return to the initial trajectory afterwards.

Simulations and real-time tests have have been performed to prove the proposed algorithms.

Lista Figurilor

Lista Notațiilor și Abrevierilor

2DW/2FW - 2 Driving Wheels/ 2 Free Wheel Casters

3D – Tridimensional

4DW/SW – 4 Driving Wheels/ Steering Wheels

AV – Autonomous Vehicle

AVAR – Allan Variance

BRW – Bias Random Walk

IMU – Inertial Measurement Unit(Unitate de Măsură Inerțială)

FK – Filtru Kalman

MEMS - Micro-Electrical Mechanical Systems

PI – Controller Proporțional Integrativ

PID – Controller Proporțional Integrativ Derivativ

WMR – Wheeled Mobile Robot

SMC – Sliding-Mode Control

VFH – Vector Field Histogram

Capitolul 1.

1. Introducere

Roboții mobili și vehiculele autonome au devenit subiecte de cercetare foarte importante în ultimii ani. Progresul tehnologic a permis crearea de roboți din ce în ce mai complexi și la prețuri din ce în ce mai mici. Din acest motiv, roboții mobili sunt din ce în ce mai des folosiți în industrie și chiar de către utilizatorii casnici. În industrie roboții mobili sunt folosiți pentru a transporta componente la liniile de asamblare și în zonele periculoase, cum ar fi depozitele de deșeuri radioactive. Au avantajul că reduc costurile de producție pentru că pot realiza obiectivele fără a fi nevoie de supravegherea umană. Roboții mobili pot fi folosiți pentru patrularea unor obiective, asistarea persoanelor cu dizabilități și ajuta în gospodării.

Vehiculele autonome pot înlocui șoferii umani, astfel fiind eliminat comportamentul imprevizibil în trafic al șoferilor. În prezent cauza majoră a accidentelor rutiere este eroarea șoferilor. Vehiculele autonome prezintă avantajul că pot naviga în condiții maxime de siguranță. Progresul în domeniul vehiculelor autonome poate revoluționa întreg transportul uman prin introducerea vehiculelor total autonome capabile să transporte oameni sau obiecte, pe orice drum, capabile să calculeze traseul necesar pentru a ajunge la destinație și să efectueze deplasarea pe ruta calculată pentru a ajunge la destinație fără controlul operatorilor umani.

Roboții mobili și vehiculele autonome au nevoie de date despre poziția și orientarea lor, viteza de deplasare și distanța până la obstacole. Pentru a obține aceste date este nevoie de senzori, cum ar fi: sonarele, senzorii inerțiali, GPS-urile, encoderele. Aceste vehicule nu au un operator uman care să analizeze datele și să trimită comenzile necesare pentru realizarea obiectivului și beneficiază de un sistem de conducere în buclă închisă care elaborează comenzile pe baza informațiilor primite de la senzori.

Informația de la senzori este perturbată de zgomot și este necesară prelucrarea acestei informații pentru a se elimina erorile care ar putea impiedica buna funcționare a sistemului. Se pot estima datele reale cu ajutorul unei fuziuni a datelor primite de la mai mulți senzori folosind un filtru Kalman.

Funcționarea în regim autonom are nevoie de un modul capabil să detecteze și să planifice un traseu care să ocolească obstacolele detectate. Un alt aspect important este

prezența modulului de conducere, care furnizează comenzile necesare pentru ca vehiculele autonome să funcționeze în condițiile dorite.

1.1 Stadiul actual

Un număr mare de companii au construit diverse tipuri de roboți mobili și vehicule autonome, și un număr mare de cercetători au cercetat acest domeniu. Din acest motiv analiza tuturor articolelor științifice sau roboților mobili și vehiculelor autonome nu este posibilă. În această secțiune sunt prezentate câteva dintre realizările notabile din acest domeniu.

Mai multe companii au dezvoltat diverși roboți mobili autonomi capabili să execute diverse sarcini în gospodărie. Câteva exemple de astfel de roboți sunt prezentate mai jos.

Compania iRobot ,[42], comercializează gama Roomba de roboți-aspirator autonomi capabili să aspire autonom podelele, inclusiv sub și în jurul mobilei și de a-lungul pereților. Roomba (Fig. 8.1) poate detecta tipurile de podele și ajusta setările pentru cea mai eficientă curățare a suprafeței respective.

Compania iRobot, [42], comercializează și gama Scooba (Fig. 8.2) de roboți autonomi proiectați pentru spălarea podelelor, care funcționează similar roboților Roomba.

Husqvarna, [45], comercializează gama Automower (Fig. 8.3) de roboți autonomi de tuns iarba.

La institutul japonez pentru cercetarea roboților, ATR, se află în curs de dezvoltare robotul Robovie-II (Fig. 8.4). Robovie-II, [40], este un robot proiectat să realizeze cumpărăturile de la magazin. Robotul se deplasează în interiorul magazinului și amintește vânzătorului produsele care trebuie cumpărate, iar vânzătorul le depozitează în coșul robotului.

Unul dintre primele vehicule autonome a fost VaMp construită de echipa lui Ernst Dickmanns de la Universitatea Bundeswehr și Mercedes-Benz în 1990. Vehiculul poate parcurge distanțe lungi în trafic intens fără intervenția umană, și să recunoască obstacole în mișcare și să le evite automat. Este un Mercedes 500 SEL modificat pentru a primi comenzi de la calculator. Comenzile sunt elaborate considerând informațiile primite de la camerele video și nu folosește GPS.

Vehiculul autonom Sojourner (Fig. 8.5), [41], a fost folosit de NASA pentru a explora planeta Marte și este un vehicul proiectat să evite automat obstacolele. A fost urmat de vehiculele Spirit și Oportunity care foloseau același sistem de evitare a obstacolelor.

Vehiculul autonom TerraMax, [38], (Fig. 8.6) produs de Oshkosh Truck Corporation, Laboratorul de Viziune artificială și Sisteme inteligente al Universității din Parma și Rockwell Collins a terminat proba DARPA Grand Challenge din 2005. Vehiculul utilizează 3 lasere LIDAR, 3 camere și 2 sisteme de navigație GPS.

În 2010 a fost lansat testul Vislab Intercontinental Autonomous Challenge, [43], la care au participat 4 vehicule autonome, VIAC (Fig. 8.7), și care au străbatut distanța de 15000 km de la Parma, Italia, până la Shanghai, China, fără intervenția operatorului uman. Vehiculele au îndurat condiții foarte dificile: șosea, vreme, infrastructură, temperatură, trafic și chiar comportament periculos al participanților la trafic și ruta a fost necunoscută.

Pentru a rezolva aceste probleme VisLab a gândit 2 vehicule autonome care se deplasează în același timp dar cu scopuri diferite. Primul vehicul merge autonom în majoritatea timpului dar uneori este nevoie de intervenția umană pentru a corecta erorile, iar al doilea vehicul urmărește primul vehicul și se deplasează autonom pe întreg parcursul testului.

Roboții mobili și vehiculele autonome sunt sisteme nonholonomice, descrise de sisteme de ecuații neliniare. Din acest motiv conducerea folosind regulatoare PID nu dă rezultate satisfăcătoare. Pentru conducerea proceselor neliniare s-au dezvoltat mai multe metode, cum ar fi ,metodele backstepping, sliding-mode, fuzzy și rețelele neuronale.

Metodele de conducere a roboților mobili și vehiculelor autonome au ca scop proiectarea unor controllere capabile să calculeze viteza liniară și viteza unghiulară a vehiculului pentru ca acesta sa urmărească o traiectorie dorită. Conducerea sistemelor nonholonomice a fost cercetată de un număr mare de cercetători și diferite regulatoare au fost propuse.

Cercetătorii din [9], [51], [52], [61], [72], [80], [98] și [117] au propus conducerea backstepping pentru a rezolva problema conducerii sistemelor nonholonomice. Backstepping este o procedură recursivă care descompune o problemă de proiectare a întregului sistem în probleme de proiectare pentru sisteme de ordin inferior. Proiectarea backstepping are la bază integratorul backstepping, acesta fiind aplicat recursiv pentru rezolvarea problemei sistemelor complexe, procedură descrisă de Chen, [9], și Khalil, [57].

Altă metodă propusă pentru conducerea sistemelor nonholonomice este conducerea sliding-mode. Conducerea sliding-mode a fost analizată de mai mulți cercetători, cum ar fi Utkin, [101], [102], Gao, [31], [32], Khalil, [57], și a fost aplicată de mai mulți cercetători în [11], [12], [13], [14], [47], [49], [63], [91], [92], [93], [94], [95], [96] și [103], pentru conducerea roboților mobili și vehiculelor autonome, cu rezultate foarte bune.

O metodă de proiectare care integrează conducerea sliding-mode, conducerea fuzzy și gain schedulling este propusă în [1].

Conducerea utilizând un controller proporțional și un predictor Smith a unui vehicul autonom 4DW/SW este propusă de [4].

În [50] este prezentată o metodă de conducere bazată pe inteligența artificială și folosește o metodă de învățare pentru a îmbunătăți politicile de conducere cu ajutorul experienței și datelor de la sistem.

Un regulator neliniar, bazat pe liniarizarea modelului dinamic al vehiculului autonom, prezentat în [64] si [69], propune o conducere bazată pe metoda Lyapunov combinată cu principiul invariației LaSalle pentru urmărirea unei traiectorii.

Navigația constă în localizarea robotului și stabilirea unei traiectorii pentru ajungerea la destinația dorită.

W. Danwei și Qi Feng, [15], au propus o metodă de planificare a traiectoriilor vehiculelor cu 4 roți directoare bazat pe modelul cinematic și care utilizează la maxim flexibilitarea oferită de roțile directoare și ia în considerație limitările mecanismului.

Fraichard et al., [27], prezintă o metodă pentru planificarea traiectoriilor folosind continous curvature pentru vehiculele autonome.

O strategie de planificare a traseului pentru vehicule autonome aeriene este prezentată în [8]. Strategia consideră puncte intermediare 3D și calculează o traiectorie netedă compatibilă cu dinamica vehiculului, pe baza mișcărilor din librăria precalculată de mișcări.

În [85] este prezentată metoda pentru calculul traiectoriei sigure pentru vehicule în medii parțial cunoscute, folosind o arhitectură de control hibridă care combină mai multe moduri de conducere a vitezei cu un programator de manevre.

Kanarat [55] propune o metodă de planificare a traiectoriei care determină un traseu optimal pentru un robot mobil într-un mediu, bazat pe incertitudinea maximă permisă, care reprezintă un număr repartizat fiecărei configurații libere din mediu. Traseul optim conectează punctul inițial de puntul final, prin configurații care respectă constrângerile nonholonomice și cea mai mare incertitudine permisă.

Ge et al. , [34], prezintă o abordare hibridă pentru planificarea traiectoriei netede, cu convergență globală pentru roboți nonholonomici.

Garcia et al. [33] tratează problema planificării traiectoriei folosind optimizarea coloniilor de furnici.

Foka, [26], propune o formulare ierarhică a proceselor de decizie Markov parțial observabile pentru navigația roboților autonomi, care poate fi rezolvată în timp real și este folosită ca un cadru unificat pentru navigația în medii dinamice.

Velagic et al., [104], a propus un sistem de navigație bazat pe 3 subsisteme de navigație. Sistemul de nivel scăzut se ocupă de controlul vitezelor folosind un controller PI. Poziționarea este calculată de nivelul mediu și este neliniară. Subsistemul de nivel înalt utilizează logica fuzzy și teria Dempster-Shafer pentru proiectarea unei fuziuni de date de la senzori, construirea hărții și planificarea traiectoriei.

[70] prezintă algoritmul pentru generarea manevrelor dinamice posibile pentru vehiculele autonome care se deplasează cu viteze mari și pe distanțe mari. Generarea se bazează pe căutarea incrementală în mulțimea spațiului stărilor.

Menegatti et al., [71], propune o navigație bazată pe localizarea robotului prin compararea imaginilor din locația curentă cu imagini de referință din memorie. Singurul senzor necesar este o cameră omnidirecțională. Componentele Fourier ale imaginii omnidirecționale simplifică soluția problemei navigației robotului.

În timpul deplasării robotului pot apare obstacole neprevăzute și este nevoie de o componentă capabilă să evite obstacolele. Lumelsky et al., [66], a propus algoritmul „bug" pentru evitarea obstacolelor, care presupune înconjurarea obstacolului pentru determinarea punctului aflat la distanță minimă față de țintă și continuarea traseului din acest punct.

Frazzoli et al., [28], propune planificarea deplasării unui vehicul autonom în medii dinamice necunoscute folosind un automaton hibrid, al cărui stări reprezintă manevre fezabile.

Alt algoritm de evitarea obstacolelor este algoritmul câmpului de potențial din [5], [6] și [58]. Acest algoritm consideră că robotul este atras de forțe virtuale către țintă și respins de forțe virtuale generate de obstacole, iar traiectoria este rezultatul compunerii acestor forțe. Există situații în care soluția nu este găsită de acest algoritm. Situațiile în care nu este găsită soluția pot fi evitate folosind câmpurile potențiale adaptive din [2], care folosesc mai multe puncte auxiliare de atracție pentru a permite robotului să evite obstacolele mari sau grupate foarte aproape unul de altul. Configurația câmpului potențial optim este calculată de un algoritm genetic.

Navigarea și evitarea obstacolelor este realizată în [26] utilizând o structură ierarhică de procese Markov de decizie parțial observabile.

Algoritmul Vector Field Hostogram din [65] elimină problema erorilor de măsură ale senzorilor prin utilizarea unei histograme polare care conține mai multe măsurări succesive recente.

Shouwenaars, [85], propune o metodă de planificare a traiectoriei pentru vehicule în medii parțial cunoscute, care permite evitarea obstacolelor.

Un algoritm de planificare a traiectoriei în medii necunoscute este prezentat în [34]. Este folosită o abordare hibridă pentru a obține o traiectorie netedă cu convergență globală pentru roboți diferențiali nonholonomici.

1.2 Structura lucrării

Structura tezei este următoarea:

Capitolul 1 a evidențiat **Stadiul actual** al implementării structurilor robotice în diferite tipologii de vehicule autonome.

Capitolul 2, intitulat **Tipologii de senzori utilizați în conducerea roboților mobili și vehiculelor autonome**, a prezentat o descriere scurtă a celor mai importanți senzori folosiți de WMR și vehiculele autonome și a prezentat în detaliu caracteristicile senzorilor IMU. Scopul acestui capitol este prezentarea rezultatelor cercetării obținute prin utilizarea senzorilor IMU realizată cu sprijinul Institutului de Sisteme și Robotică din cadrul Universității din Coimbra, în timpul stagiului extern. Au fost prezentate erorile care caracterizează mărimile măsurate de senzorii IMU și a fost analizat senzorul IMU Xsens Mti aplicând metoda Allan. În secțiunea următoare modul de reprezentare a atitudinii vehiculelor utilizând quatenioni a fost analizat, apoi a fost prezentat FK, urmat de implementarea folosită prentru estimarea erorilor atitudinii furnizate de IMU. Filtrul corectează erorile furnizate de giroscoapele 3D utilizând măsurătorile de la accelerometre și senzorii magnetici.În secțiunea următoare sunt prezentate atitudinile estimate.

Capitolul 3, intitulat **Conducerea trajectory-tracking a roboților mobili și vehiculelor autonome**, a prezentat determinarea modelului cinematic al WMR cu 2DW/2FW și modelului cinematic al vehiculului autonom electric SEEKUR(4DW/SW). Modelele cinematice descriu mișcarea robotului sau a vehiculului și nu iau în calcul forțele care acționează asupra lor. Pentru simplificarea calculelor modelul vehiculului autonom SEEKUR a fost simplificat și s-a obținut un model similar modelului bicicletei. Aceste modele au fost folosite pentru proiectarea a 3 regulatoare pentru conducerea trajectory-tracking a roboților mobili 2DW/2FW și 3 regulatoare pentru conducerea trajectory-tracking

a vehiculelor autonome 4DW/SW: Regulatoarele au fost calculate aplicând metodologia sliding-mode în timp continuu, metodologia sliding-mode în timp discret și metodologia backstepping.

Capitolul 4, intitulat **Algoritmi de conducere a vehiculelor autonome și a roboților mobili, destinați evitării obstacolelor**, a prezentat o metodă de evitare de obstacole pentru roboții mobili care utilizează sonarele și o metodă de evitare a obstacolelor pentru vehiculele autonome care folosește detecția laser. Robotul urmărește o traiectorie impusă și folosește una din metodele de conducere menționate în capitolul 4. La fiecare pas se verifică dacă a fost detectat un obstacol, iar în cazul detectării unui obstacol se caută o traiectorie pentru evitarea obstacolului folosind Quintic equations și se calculează vitezele și accelerațiile care descriu traiectoria de evitare a obstacolelor. Noua traiectorie este urmărita de procedura de conducere, iar când traiectoria inițială nu mai este blocată robotul revine la traiectoria inițială.

În capitolul următor, Capitolul 5, intitulat **Implementarea conducerii în timp real. Rezultate experimentale**, a fost prezentată implementarea metodelor propuse și rezultate experimentale pentru validarea metodelor propuse.

Capitolul final prezintă **Concluzii**.

1.3 Lista publicațiilor

Această teză are la baza următoarele publicații:

1. [16] **Dumitrascu B**. and Filipescu A., Discrete-time sliding-mode controller for wheeled mobile robots, Proceedings of the 18th International Conference on Control Systems and Computer Science, vol. 1, Bucuresti, pag. 397-403, mai, 2011.
2. [17] **Dumitrascu B**. and Filipescu A., Sliding mode control of lateral motion for four driving-steering wheels autonomous *vehicle, Annals of the University* of Craiova, vol. 7 (34), pag 20-25, 2010.
3. [18] **Dumitrascu B**. and Filipescu A., Sliding mode controller for steering of four-wheels driving and steering *vehicle, Proceedings of the 14th International Conference on* System Theory and Control, pag. 202-206, Sinaia, 2010.
4. [19] **Dumitrascu B**., Filipescu A., Backstepping control of wheeled mobile *robots, Proceedings of the 15th International Conference on* System Theory and Control, Sinaia, pag. 206-211, 2011.

5. [20] **Dumitrascu B**., Filipescu A., Radaschin A., Filipescu A. Jr., Minca E., Discrete-time sliding mode control of wheeled mobile *robots, Proceedings of 2011 8th Asian Control Conference (*ASCC) ,Kaohiung, Taiwan, pag. 771-776, mai , 2011.
6. [21] **Dumitrascu B**., Filipescu A., Vasilache C., Minca E., Filipescu A. Jr., Discrete-time sliding-mode control of four driving/steering wheels mobile platform, Proceedings of the 19th Mediterranean Conference on Control and Automation, Corfu, Grecia, iunie, 2011, pag. 1076-1081.
7. [22] **Dumitrascu B**., Filipescu A., Minzu V., Voda A., Minca E.,Discrete-Time Sliding-Mode Control of Four Driving-Steering Wheels Autonomous Vehicle, *Proceedings of the 30th Chinesse Control Conference,* pag. 3620-3625, 2011.
8. [24]Filipescu A., Minzu V., Dumitrascu **B.** and Filipescu A., Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots, The 2011 IEEE International Conference on Information and Automation, iunie, 2011.
9. [93] Solea R., Filipescu A., Filipescu S, and **Dumitrascu B**., Sliding-mode controller for four- wheel-steering vehicle: trajectory-tracking *problem, Proceedings of the 8th World Congress on Intelligent Control* and Automation, Jinan, China, pag. 1185-1190, 2010.

Capitolul 2.

2. Tipologii de senzori utilizați în conducerea roboților mobili și vehiculelor autonome

Vehiculele autonome și roboții mobili sunt dotați cu mai multe tipuri de senzori pentru a fi posibilă determinarea poziției, orientării și distanței până la obstacole, aceste informații fiind vitale pentru funcționarea autonomă.

Cele mai importante tipuri de senzori sunt: encoderele, senzorii inerțiali (IMU), sonarele și laserele.

2.1. Encodere

Un encoder este un dispozitiv electro-mecanic care convertește mișcarea unghiulară a roților vehiculului în cod analog sau digital. Encoderele sunt printre senzorii cei mai folosiți pentru măsurarea vitezei de rotire a unui ax. Informația de la encoder este procesată pentru a se obține viteza, poziția și orientarea vehiculului. In calculul vitezei și poziției bazate pe datele de la encodere pot apare erori datorită alunecărilor.

Există metode de eliminare a erorilor furnizate de encodere, dintre acestea fiind prezentate în [75] ,[76] și [78]. Viteza și poziția robotului sunt calculate în funcție de deplasarea unghiulară a roții în jurul axului și din acest motiv nu toate erorile pot fi minimizate. Pot exista cazuri în care robotul se deplasază pe o suprafață alunecoasă chiar dacă roțile sale nu se învârt, caz în care encoderul raportează că robotul este staționar, sau în cazul în care roțile alunecă și encoderul raportează o deplasare mult mai mare decât cea reală. Pentru a obține informația reală este nevoie de un alt tip de senzor, cum ar fi senzorul IMU.

Fig. 2.1 Encoder incremental [69]

2.2. Senzori laser

Senzorii laser sunt folosiți pentru detectarea distanței până la obstacole sau pentru identificarea unor repere. Laserele măsoară durata de timp dintre emiterea unui puls laser și reflecția lui de către un obiect. Pentru că timpul dintre transmisia pulsului și recepția lui este proporțională cu distanța se poate calcula distanța la care se află obiectul. Informația este trimisă în timp real și laserele pot fi folosite în exterior datorită distanței mari de detectare a obstacolelor(până la 20 m pentru laserul SICK LMS 111).

Laserele pot măsura obstacole aflate la unghiuri de până la $0.25°$, în funcție de rezoluția laserului, oferind o detecție mai bună comparativ cu rezoluția sonarelor.

Fig. 2.2 Senzorul laser SICK LMS111

2.3. Unități de măsură inerțială

Unitățile de măsură inerțială(IMU) sunt compuse din accelerometre 3D și giroscoape 3D și uneori senzori magnetici 3D. Acești senzori pot detecta mișcarea chiar și în cazurile în care encoderele nu pot furniza corect datele.

IMU-urile sunt folosite pentru a calcula viteza, poziția și atitudinea vehiculelor, navelor și avioanelor. Pentru a se obține viteza semnalul de la accelerometru este integrat, iar pentru a afla poziția accelerația este dublu integrată. Prin integrarea datelor de la giroscop se obține unghiul de ruliu, unghiul de tangaj și orientarea. Semnalele primite de la IMU conțin erori care sunt integrate pentru a se obține datele dorite. Din acest motiv IMU reprezintă un

senzor bun pentru perioade de timp scurte, dar acumulează erori din ce în ce mai mari dacă timpul de funcționare este mai lung. Aceste erori trebuie identificate și corectate pentru a obține date corecte. Corectarea datelor de la IMU se realizează utilizând filtrul Kalman.

Cercetarea senzorilor IMU Xsens Mti a fost realizată cu sprijinul Institutul de Sisteme și Roboți din cadrul Universității din Coimbra, unde am efectuat stagiul extern din cadrul proiectului POSDRU.

2.3.1. Erorile înregistrate în funcționarea IMU

Un IMU oferă date precise despre atitudine și poziție pentru vehicule doar în cazul perioadelor scurte de timp, în cazul intervalelor de timp lungi integrarea micilor erori ale IMU conduc la erori mari. Soluția pentru o funcționare corectă este identificarea și eliminarea erorilor care afectează senzorul. Erorile înregistrate în funcționarea IMU și modelarea stohastică a IMU au fost prezentate în [25], [76]și [77]. O metodă pentru identificarea erorilor este metoda Allan. Această metodă a fost folosită de cercetătorii din: [23], [37],[107] și [111].

Metoda Allan este o tehnică de analiză în domeniul timp, care a fost inițial folosită pentru analiza stabilității oscilatoarelor, dar apoi a fost folosită pentru analiza senzorilor inerțiali. Are avantajul că se poate identifica caracteristica procesului aleator care generează perturbația(Fig. 2.3). Această metodă presupune că incertidudinea este cauzată de diferite caracteristici ale componentelor zgomotului. Covarianța fiecărei componente este evaluată folosind datele din analiza Allan.

Divergența Allan se calculează astfel:

1. Se obține un set lung de date și se împarte în grupuri de lungimea τ . Setul de date trebuie să poată fi împărțit în minim 9 grupuri pentru a păstra semnificația;

2. Se formează o listă cu valoarea medie din fiecare grup ($a(\tau)_1, a(\tau)_2, ..., a(\tau)_n$);

3. Se utilizeaza formula

$$AVAR(\tau) = \frac{1}{2(n-1)} \sum \left(a(\tau)_{i+1} - a(\tau)_i \right)^2 ; \tag{2.1}$$

4. Caracteristicile zgomotului se determină din graficul deviației Allan ca o funcție de τ pe o scală logaritmică. Fiecare componentă a zgomotului este reprezentată de o pantă cu un gradient diferit. Caracteristica generală a zgomotului conform metodei Allan din [48] este prezentată în Fig. 2.3.

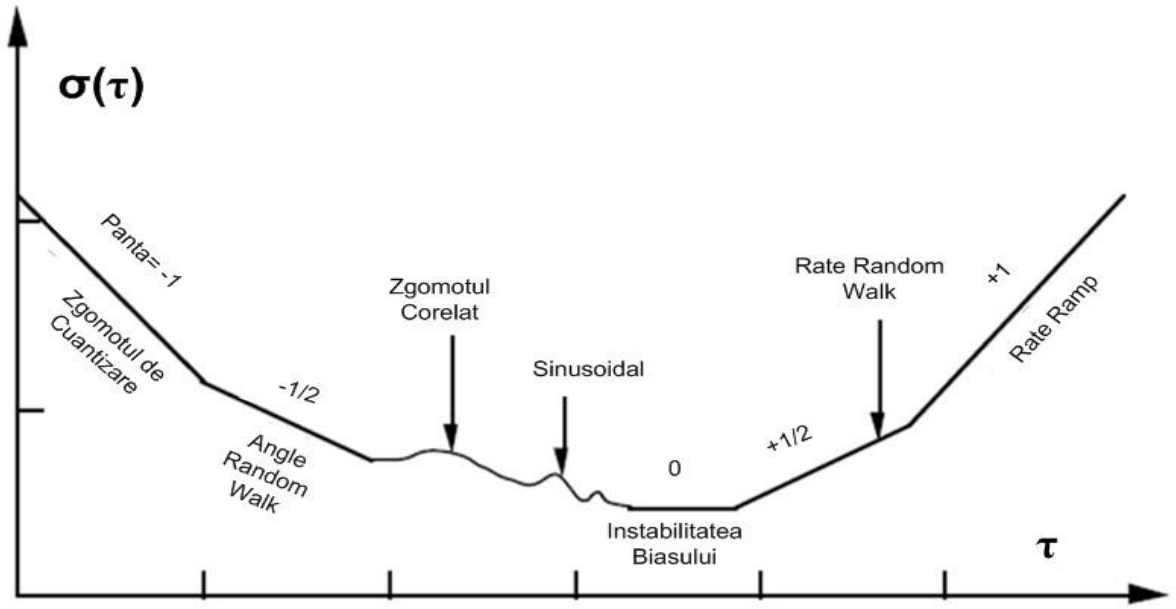

Fig. 2.3 Caracteristica zgomotului conform metodei Allan(adaptare după lucrarea[48])

Semnalul de la giroscoape conține viteza de rotație perturbată de zgomot și deviație. Cele mai semnificative erori ale giroscoapelor sunt biasul constant, angle random walk, flicker noise, efectul temperaturii și eroarea de calibrare. Biasul constant reprezintă valoarea medie măsurată când giroscopul nu se rotește și se exprimă în grade pe oră sau radiani pe secundă. Biasul constant produce o eroare care crește liniar cu timpu($\theta_e = \varepsilon \cdot t$).

Valoarea calculată a biasului giroscopului este:

$$Bgyrox = -9.37799e-4;$$

$$Bgyroy = -0.0011;$$

$$Bgyroz = -6.2526e-4.$$

Angle random walk este cauzat de integrarea zgomotului alb prezent în semnalul giroscopului. Această eroare introduce o deviație standard care crește proporțional cu $\sqrt{t}$ și se măsoară în $°/\sqrt{h}$. Pe graficul Allan este reprezentat de o pantă cu gradient -1/2. Valoarea numerică se citește din grafic la timpul $\tau = 1$.

Stabilitatea biasului unui giroscop se modifică în timp datorită zgomotului de comutație în componentele electronice. Zgomotul de comutație are frecvența scăzută și apare ca variații ale biasului în semnal. Se modelează ca random walk. Stabilitatea biasului specifică cum se poate schimba biasul într-o perioadă de timp, de obicei 100s. Valoarea lui este exprimată ca o valoare $1\sigma[°/h]$. Dacă B_τ este biasul la timpul τ , o stabilitate de 0.01

timp de 100s înseamnă că $B_\tau + 100$ este o variabilă aleatoare cu valoarea așteptată B_τ și deviația standard 0.01. Uneori se exprimă folosind bias random walk

$$BRW(°/\sqrt{h}) = \frac{BS(°/\sqrt{h})}{\sqrt{t(h)}}. \tag{2.2}$$

În realitate biasul nu crește nelimitat cu creșterea timpului și poate fi aproximat ca un random walk doar pentru intervale de timp scurte. Instabilitatea este reprezentată de zona plată din zona minimului și valoarea numerică este valoarea minimă din grafic.

Rate random walk este un proces cu origini incerte și e reprezentat cu o pantă de $1/2$, iar magnitudinea poate fi citită la $\tau = 3$.

Rate ramp este o eroare deterministică care apare în timpul intervalelor de timp mari și e definit[48]

$$\Omega = R \cdot t, \tag{2.3}$$

unde R este coeficientul rampei. Rampa este reprezentată de o pantă cu gradientul 1 și amplitudinea poate fi calculată la $\tau = \sqrt{2}$.

Quantization noise este cauzat de natura digitală a circuitului și e reprezentat cu panta -1 și magnitudinea poate fi citită la $\tau = \sqrt{3}$.

Graficul Allan pentru cele 3 giroscoape ale Xsens Mti sunt prezentate în Fig. 2.4.

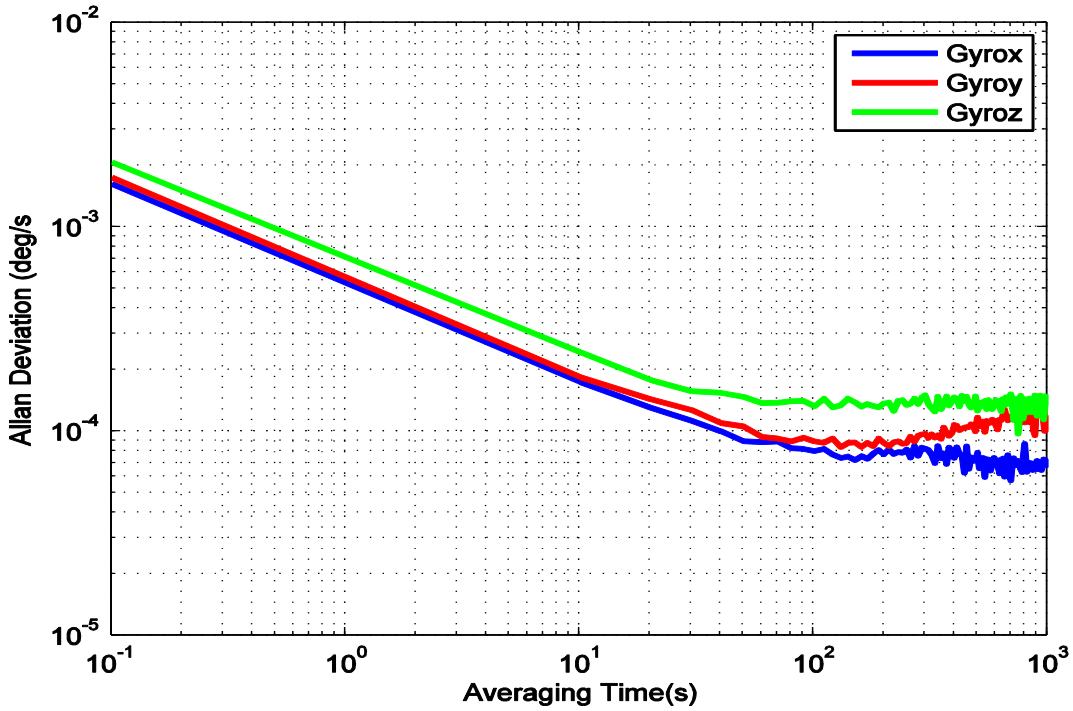

Fig. 2.4 Deviația Allan a giroscoapelor

Zgomotul alb pentru giroscop este:

$$Gnx = 0.00533;$$

$$Gny = 0.056;$$

$$Gnz = 0.00714.$$

Instabilitatea giroscopului este:

$$BIGx = 0.001283;$$

$$Bigy = 0.00179;$$

$$Bigz = 0.0171.$$

Biasul accelerometrului este valoarea medie a accelerației (m/s^2) când accelerometrul nu este supus accelerației. Valoarea medie a unui set lung de date dacă nu e afectată de gravitație este biasul. Poziția este rezultatul dublei integrări a valorilor de la accelerometru și se obține o eroare care evoluează astfel

$$p(t) = \varepsilon \cdot \frac{t^2}{2}. \tag{2.4}$$

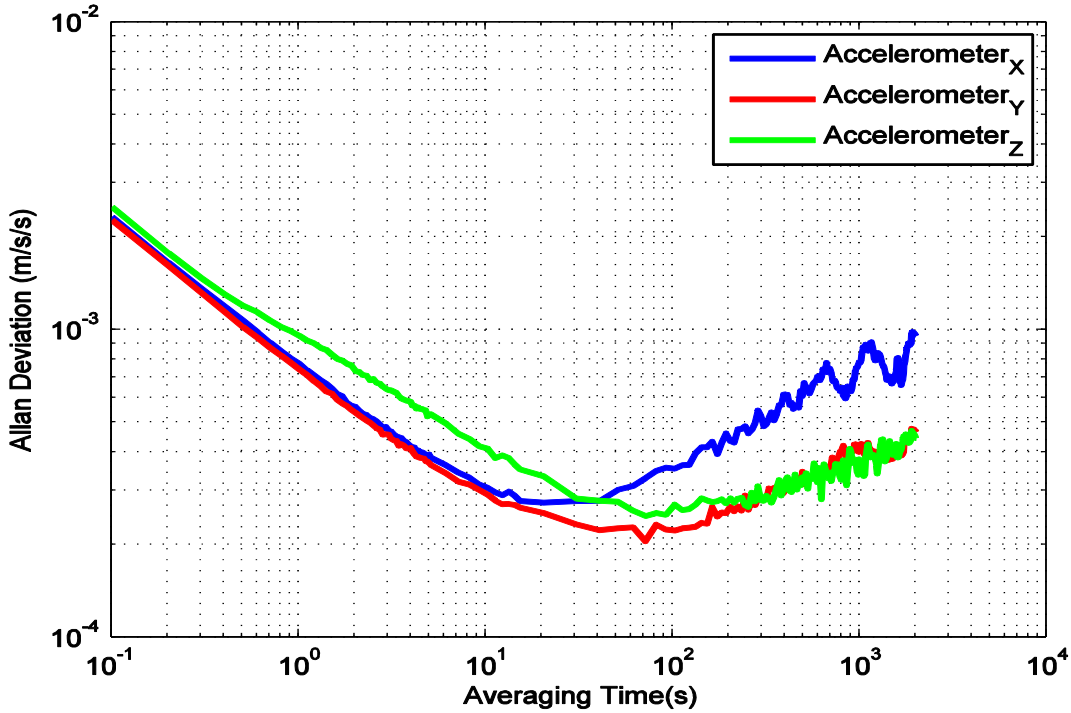

Fig. 2.5 Deviația Allan a accelerometrelor

Biasul calculat este:

$$B_x = -0.0781;$$

$$B_y = 0.0430;$$

$$B_z = 0.0091.$$

Velocity random walk se măsoară în $m/s/\sqrt{h}$ și e cauzat de integrarea zgomotului alb. Zgomotul alb generează random walk de ordinul doi în poziție, cu medie zero și deviație standard [107]

$$\sigma_p(t) = \sigma \cdot t^{3/2} \cdot \sqrt{\frac{\delta t}{3}}. \tag{2.5}$$

Stabilitatea biasului accelerometrului determină variația biasului în timp.

Deviația Allan pentru accelerometre este prezentată în Fig. 2.5. Zgomotul alb apare ca o pantă cu valoarea -0.5. Valoarea pentru random walk este citită la $\tau = 1$. Instabilitatea biasului apare ca regiunea plată din zona minimului și valoarea este valoarea minima a curbei.

Zgomotul alb al accelerometrelor este:

$$Ax = 0.000911;$$

$$Ay = 0.00092;$$

$$Az = 0.001429.$$

Instabilitatea biasului accelerometrelor este:

$$BIAx = 0.00021;$$

$$BIay = 0.00021;$$

$$BIAz = 0.00023.$$

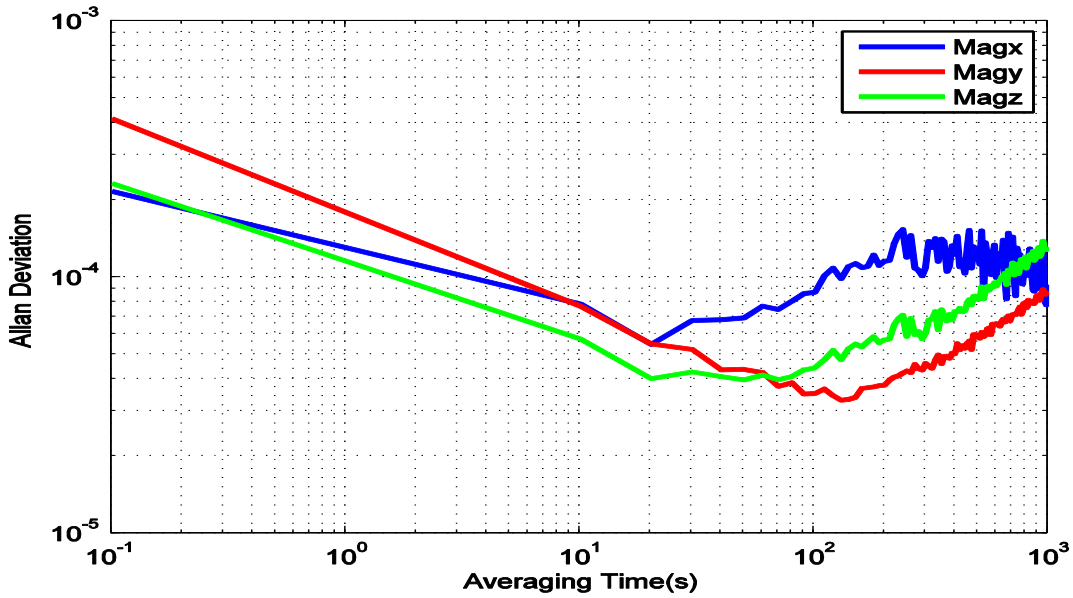

Fig. 2.6 Deviația Allan pentru senzorii magnetici

Deviația Allan pentru senzorii magnetici este prezentată în Fig. 2.6.

Zgomotul alb al senzorilor magnetici este:

$$Mx = 0.0001307;$$

$$My = 0.000179;$$

$$Mz = 0.0001156.$$

2.3.2. Reprezentarea atitudinii

Atitudinea este formată din unghiul de ruliu, unghiul de tangaj și orientarea. În cazul roboților mobili și vehiculelor autonome cel mai inportant unghi este unghiul care reprezintă orientarea, dar pentru o estimare mai bună a orientării trebuie folosite cele 3 unghiuri.

Atitudinea poate fi exprimată folosind unghiurile Euler, dar unghiurile Euler conțin o singularitate. Această problemă a dus la stabilirea altor reprezentări: direct cosine formulation, Euler-axis formulation, Euler-Rodrigues quaternion formulation. Aceste reprezentări sunt prezentate în [79].

Algebra quaternionilor elimină singularitatea și are nevoie de mai puțin timp de calcul, fiind o foarte bună metodă de reprezentare a atitudinii. Proprietățile quaternionilor sunt descrise de [44], [67], [86].

Un quaternion e reprezentat de un vector cu 4 elemente

$$q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} q_0 & \vec{q} \end{bmatrix}^T. \tag{2.6}$$

unde q_0 este partea scalară a quaternionului și $\vec{q}$ este partea vectorială, q_0, q_1, q_3, q_4 sunt numere reale.

Elementele sunt definite astfel

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ r_x \cdot \sin\frac{\theta}{2} \\ r_y \cdot \sin\frac{\theta}{2} \\ r_z \cdot \sin\frac{\theta}{2} \end{bmatrix}. \tag{2.7}$$

unde θ este unghiul necesar pentru a roti sistemul de coordonate A în jurul axei pentru a obține sistemul de coordonate B, r_x, r_y, r_z definesc componentele vectorului unitate r pe axele cadrului A.

Quaternionul descrie o atitudine cu 3 grade de libertate. Relația dintre componente este

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = \cos^2\frac{\theta}{2} + (r_x^2 + r_y^2 + r_z^2)\cdot\sin^2\frac{\theta}{2} = 1. \tag{2.8}$$

Matricea de rotație a vectorului v de la coordonatele A la coordonatele B este

$$C(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2\cdot(q_1\cdot q_2 - q_0\cdot q_3) & 2\cdot(q_1\cdot q_3 + q_0\cdot q_2) \\ 2\cdot(q_1\cdot q_2 + q_0\cdot q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2\cdot(q_2\cdot q_3 - q_0\cdot q_1) \\ 2\cdot(q_1\cdot q_3 - q_0\cdot q_2) & 2\cdot(q_2\cdot q_3 + q_0\cdot q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad 2.9)$$

Un quaternion poate fi definit ca

$$q = q_0 + q_1\cdot i + q_2\cdot j + q_3\cdot k. \tag{2.10}$$

Adunarea și scăderea quaternionilor p și q sunt definite prin adunarea și scăderea componentelor corespunzătoare:

$$q + p = (q_0 + p_0) + i\cdot(q_1 + p_1) + j\cdot(q_2 + p_2) + k\cdot(q_3 + p_3); \tag{2.11}$$

$$q - p = (q_0 + p_0) + i\cdot(q_1 - p_1) + j\cdot(q_2 - p_2) + k\cdot(q_3 - p_3). \tag{2.12}$$

Produsul este definit astfel:

$$q \otimes p = (q_0 + q_1\cdot i + q_2\cdot j + q_3\cdot k)\otimes(p_0 + p_1\cdot i + p_2\cdot j + p_3\cdot k); \tag{2.13}$$

$$\begin{aligned} q\otimes p = {} & q_0\cdot p_0 - q_1\cdot p_1 - q_2\cdot p_2 - q_3\cdot p_3 + i\cdot(q_0\cdot p_1 + q_1\cdot p_0 + q_2\cdot p_3 - q_3\cdot p_2) \\ & + j\cdot(q_0\cdot p_2 + q_2\cdot p_0 + q_3\cdot p_1 - q_1\cdot p_3) + k\cdot(q_0\cdot p_3 + q_3\cdot p_0 + q_1\cdot p_2 - q_2\cdot p_1) \end{aligned}; \tag{2.14}$$

$$q\otimes p = -q\cdot p + q\times p. \tag{2.15}$$

Magnitudinea unui quaternion este

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{2.16}$$

Conjugatul unui quaternion este:

$$q^* = q_0 - q_1\cdot i - q_2\cdot j - q_3\cdot k; \tag{2.17}$$

$$q\otimes q^* = |q|^2$$

Un vector poate fi rotit de un quaternion folosind

$$v' = q\otimes v\otimes q^*. \tag{2.18}$$

Relația dintre unghiurile Euler și quaternion este

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan(2\cdot\dfrac{q_2\cdot q_3 + q_0\cdot q_1}{q_0^2 - q_1^2 - q_2^2 + q_3^2}) \\ \arcsin[2\cdot(-q_1\cdot q_3 + q_0\cdot q_2)] \\ \arctan(2\cdot\dfrac{q_0\cdot q_3 + q_2\cdot q_1}{q_0^2 + q_1^2 - q_2^2 - q_3^2}) \end{bmatrix}. \tag{2.19}$$

2.3.3. Filtrul Kalman

Filtrul Kalman a fost dezvoltat în 1960. Este o tehnică de estimare recursivă bazată pe spațiul stărilor, în domeniul timp. Estimatorul optimal este un algoritm care procesează măsurătorile pentru a obține eroare minimă estimată a stării sistemului , folosind cunoștințe despre sistem și dinamica măsurătorilor, presupuse statistice și zgomotul sistemului, erorile de măsurare și condițiile inițiale din [3]. Filtrul Kalman a devenit tehnica principală de combinare în sistemele multisenzor. O prezentare detaliată a FK este prezentată în [35].

Modelul filtrului Kalman liniar în timp discret este

$$x_k = F_k \cdot x_{k-1} + B_k \cdot u_k + w_k . \tag{2.20}$$

Unde x_k este vectorul stărilor la momentul k, F_k este matricea de tranziție a stărilor de la vectorul stărilor la momentul k-1 la momentul K, u_k este vectorul comenzii, B_k reprezintă relația dintre vectorul de control și vectorul stărilor, w_k este secvența de zgomot alb necorelat Gaussian, cu media zero și covarianța Q_k , adică

$$E[w_j \cdot w_j^T] = \begin{cases} Q_k, i = j = k \\ 0, i \neq j \end{cases} . \tag{2.21}$$

Vectorul observațiilor este dat de

$$z_k = H_k \cdot x_k + v_k . \tag{2.22}$$

unde H_k este modelul de observație, și v_k este zgomotul de observație presupus a fi zgomot alb necorelat Gaussian, cu media zero și covarianța R_k, adică

$$E[v_j \cdot v_j^T] = \begin{cases} R_k, i = j = k \\ 0, i \neq j \end{cases} . \tag{2.23}$$

Filtrul furnizează o estimare a stării la momentul k, folosind observațiile până la momentul k. Algoritmul are două etape: etapa de predicție și etapa de estimare.

În etapa de predicție se evaluează starea folosind observațiile până la momentul k-1

$$\hat{x}_k^- = F_k \cdot \hat{x}_{k-1}^+ + B_k \cdot u_k \ . \tag{2.24}$$

Incertitudinea predicției este reprezentată de matricea de covarianță P_k^-

$$P_k^- = E[(x_k - \hat{x}_k^-) \cdot (x_k - \hat{x}_k^-)^T \big| Z^{k-1}] = F_k \cdot P_{k-1}^+ \cdot F_k^T + Q_k . \tag{2.25}$$

Ecuațiile (2.24), (2.25) se evaluează la fiecare pas. Când o estimare este disponibilă se trece la etapa de estimare, calculată astfel

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot \nu_k \,, \tag{2.26}$$

unde K_k este matricea de câștig a filtrului și ν_k este vectorul inovație egal cu diferența dintre valoarea observată și valoarea prezisă

$$\nu_k = z_k - H_k \cdot \hat{x}_k^- \,. \tag{2.27}$$

Câștigul este ales astfel încât să minimizeze eroarea pătrată a estimării

$$K_k = P_k^- \cdot H_k^T \cdot S_k^{-1} \,, \tag{2.28}$$

$$S_k = H_k \cdot P_k^- \cdot H_k^T + R_k \,. \tag{2.29}$$

Matricea de covarianță se actualizează folosind:

$$P_k^+ = E[(x_k - \hat{x}_k^-) \cdot (x_k - \hat{x}_k^-)^T | Z] \,; \tag{2.30}$$

$$P_k^+ = (I - K_k \cdot H_k) \cdot P_k^- \cdot (I - K_k \cdot H_k)^T + K_k \cdot R_k \cdot K_k^T \,. \tag{2.31}$$

2.2.4. Implementarea practică a filtrului Kalman

Atitudinea vehiculului este estimată folosind accelerometre 3D, giroscoape 3D și compasuri magnetice 3D conținute în pachetul Xsens Mti. Datele de la senzori sunt filtrate folosind filtrul Kalman. FK este folosit foarte pentru estimarea atitudinii în [10], [53], [54], [60], [66], [68], [82], [84], [100], [107], [109], [110], [113] și [116]. În [59] este prezentată detectarea inclinației senzorului IMU utilizând accelerometrele din senzorul IMU, și se constată posibilitatea corectării orientării cu ajutorul acceleratiilor înregistrate de accelerometre.

Pentru determinarea atitudinii se definește o stare nominală și o stare eroare. Schema bloc a algoritmului de estimare este prezentată în Fig. 2.7.

Folosim Filtrul Kalman indirect pentru a minimiza starea eroare. Primul pas este prezicerea stării nominale a quaternionului integrând valorile primite de la giroscoape. Formula pentru integrare este [53]

$$\dot{\hat{q}} = \frac{1}{2} \cdot \Omega(w) \cdot \hat{q} \,, \tag{2.32}$$

unde

$$\Omega(w) = \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix}. \tag{2.33}$$

Semnalele giroscoapelor sunt perturbate de zgomot și valoarea prezisă trebuie corectată utilizând

$$q = \tilde{q}_e \otimes \hat{q}, \tag{2.34}$$

unde $\tilde{q}_e$ este eroarea estimată dintre valoarea reală și cea prezisă.

Pentru că $\tilde{q}_e$ depinde doar de zgomotul și biasul semnalului și are valori foarte mici, care permit aproximarea:

$$\tilde{q}_e = \begin{bmatrix} 1 \\ q_e \end{bmatrix} \tag{2.35}$$

Filtrul Kalman estimează eroarea și corectează quaternionul q folosind ecuația (2.34).

Eroare este aproximată ca [53]:

$$\hat{q}_e = -y_g \times q_e - \frac{1}{2} \cdot \left(b_g + v_g\right) \tag{2.36}$$

unde y_g este rata unghiulară măsurată, b_g este biasul giroscoapelor, v_g este zgomotul.

Ieșirile accelerometrelor, giroscoapelor și senzorilor magnetic au forma

$$\begin{cases} y_a = C(q)\tilde{g} + b_a + v_a + a_b \\ y_g = \omega + b_g + v_g \\ y_m = C(q)\tilde{m} + v_m \end{cases}, \tag{2.37}$$

$$\tilde{g} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.39}$$

$$\tilde{m} = \begin{bmatrix} \cos i \cdot \cos d \\ \cos i \cdot \sin d \\ -\sin i \end{bmatrix}, \tag{2.40}$$

unde g este accelerația gravitațională și i este înclinația vectorului câmpului magnetic terestru, iar d declinația, i și d sunt valori specifice zonei în care se efectuează testul.

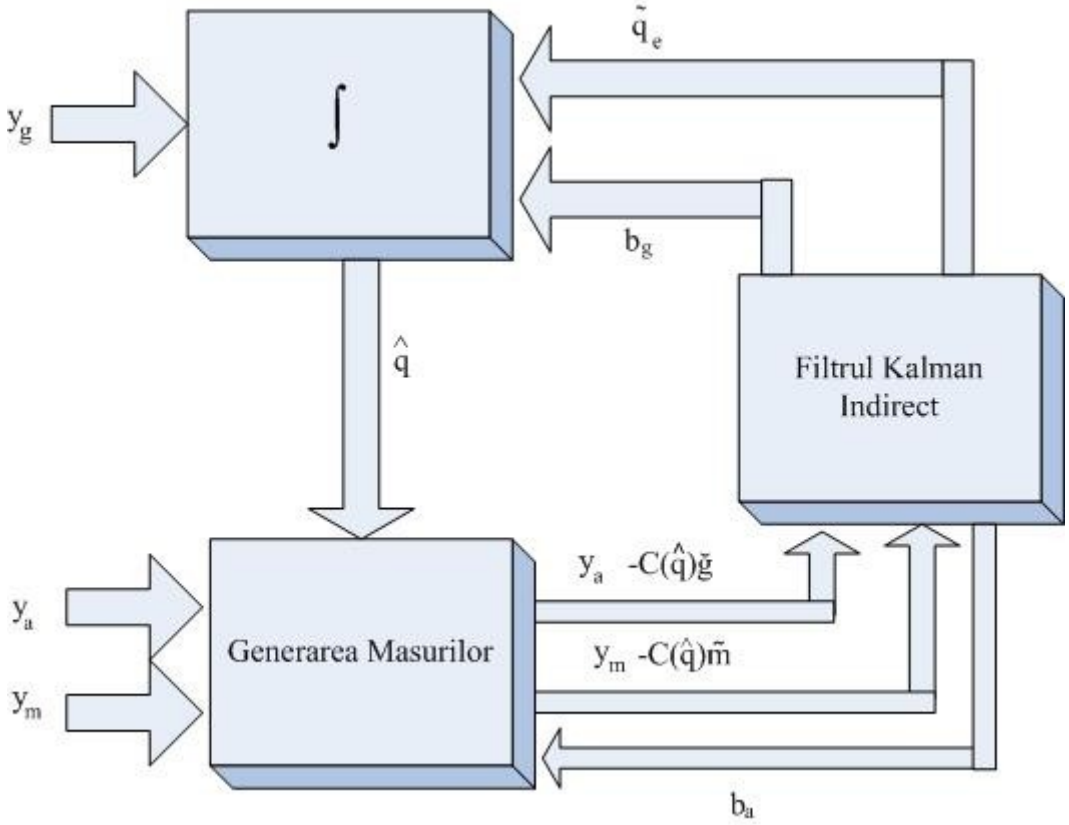

Fig. 2.7 Schema bloc a algoritmului de filtrare

Definim starea ca

$$x = \begin{bmatrix} q_e \\ b_g \\ b_a \end{bmatrix} \tag{2.41}$$

Ecuația procesului este

$$x_{k+1} = A_k x_k + \begin{bmatrix} -0.5 v_g \\ v_{bg} \\ v_{ba} \end{bmatrix}, \tag{2.42}$$

$$A_k = I + A \cdot T + \frac{1}{2} A^2 \cdot T^2, \tag{2.43}$$

$$A = \begin{bmatrix} -[y_g \times] & -0.5I & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.44}$$

Fie un vector $v \in R^3$, notația $[v\times]$ este definită astfel

$$[v\times] = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{2.45}$$

Ecuațiile măsurilor sunt [113]:

$$y_a - C(\hat{q})\tilde{g} = 2[C(\hat{q})\tilde{g}\times]q_e + a_b + v_a + b_a; \tag{2.46}$$

$$y_m - C(\hat{q})\tilde{m} = 2[C(\hat{q})\tilde{m}\times]q_e + v_m. \tag{2.47}$$

Predicția stării se face utilizând ecuația (2.42):

$$\hat{x}_{k+1}^{-} = A_k \hat{x}_k; \tag{2.48}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + Q. \tag{2.49}$$

Corecția se face în 2 etape pentru a minimaliza erorile magnetice mari care pot fi generate când senzorul este în apropierea unor obiecte metalice. Prima etapă asigură corecția utilizând informațiile de la accelerometre și în etapa a doua se aplică corecția magnetică.

- Etapa 1

Măsurătoarea $y_a - C(\hat{q})\tilde{g}$ este folosită penru corecta estimarea:

$$K_{a,k} = P_k^{-} \cdot H_{a,k}^T (H_{a,k} P_k^{-} H_{a,k}^T + R_a + \hat{Q}_{a,k})^{-1}; \tag{2.50}$$

$$\hat{x}_{a,k} = \hat{x}_k^{-} + K_{a,k}(z_{a,k} - H_{a,k}\hat{x}_k^{-}); \tag{2.51}$$

$$P_{a,k} = (I - K_{a,k}H_{a,k})P_k^{-}(I - K_{a,k}H_{a,k})^T + K_{a,k}\left(R_a + \hat{Q}_{a,k}\right)K_{a,k}{}^T; \tag{2.52}$$

$$P_k^{+} = \frac{1}{2}\left(P_{a,k} + P_{a,k}{}^T\right); \tag{2.53}$$

$$H_{a,k} = \left[2[C(\hat{q})\tilde{g}\times] \quad 0 \quad I\right]. \tag{2.54}$$

$\hat{Q}_{a,k}$ este definită conform [113]

$$\hat{Q}_{a,k} = U_k - (H_{a,k} P_k^{-} H_{a,k}^T + R_a). \tag{2.55}$$

U_k este simetrică și există vectori eigen $u_{i,k} \in R^{3\times 1}$ și valori corespunzătoare eigen $\lambda_{i,k} \in R$ astfel încât

$$U_k = \sum_{i=1}^{3} \lambda_{i,k} \cdot u_{i,k} \cdot u_{i,k}{}^T. \tag{2.56}$$

În cazul în care nu există accelerație externă ($\max(\lambda_{i,k} - u_{i,k}) < \varsigma$) $\hat{Q}_{a,k}$ este zero. Dacă există accelerații externe

$$\hat{Q}_{a,k} = \sum_{i=1}^{3} \max(\lambda_{i,k} - u_{i,k}, 0) u_{i,k} \cdot u_{i,k}{}^{T} . \tag{2.57}$$

După calculul estimării corectăm atitudinea astfel:

$$q_e = \hat{x}_{a,k}(1:3) \ ; \tag{2.58}$$

$$\hat{q} = \hat{q} \otimes \bar{q}_e \, ; \tag{2.59}$$

$$\hat{q} = \hat{q} / \|\hat{q}\| ; \tag{2.60}$$

$$\hat{x}_{a,k}(1:3) = 0. \tag{2.61}$$

- Etapa 2. Corecția magnetică:

$$P_{m,k}^{-} = P_k^{+} ; \tag{2.62}$$

$$K_{m,k} = P_{m,k}^{-} H_{m,k}^{T} (H_{m,k} P_{m,k}^{-} H_{m,k}^{T} + R_m)^{-1} ; \tag{2.63}$$

$$\hat{x}_k = \hat{x}_{a,k} + K_{m,k}(z_{a,k} - H_{m,k} \hat{x}_{a,k}) ; \tag{2.64}$$

$$H_{m,k} = \begin{bmatrix} 2[C(\hat{q})\tilde{m}\times] & 0 & 0 \end{bmatrix} ; \tag{2.65}$$

$$z_{m,k} = y_{m,k} - C(\hat{q}_k)\tilde{m} ; \tag{2.66}$$

$$q_e = \hat{x}_k(1:3) . \tag{2.67}$$

Se corectează atitudinea folosind (2.59),(2.60).

2.3.5. Rezultate obținute prin implementarea algoritmului de filtrare Kalman în estimarea atitudinii

Datele au fost colectate în urma rulării vehiculului Yamaha (Fig. 2.8) în parcarea laboratorului. Vehiculul Yamaha este dotat cu un calculator care folosește sistemul de operare ROS pentru a colecta datele de la toți senzorii echipați și pentru a funcționa în regim autonom. Datele colectate de la vehicul au fost prelucrate în MATLAB utilizând algoritmul prezentat în secțiunea anterioară. Programul principal MATLAB este trecut în Anexa 3. Programul principal folosit la estimarea atitudinii și în

Anexa 4. Funcția pentru calculul atitudinii utilizând filtrul Kalman este trecută funcția care implementează filtrul Kalman. Rezultatele unuia din testele efectuate pentru validarea algoritmului de estimare a atitudinii sunt prezentate în figurile următoare. În grafice atitudinea exprimată în quaternioni este convertită în atitudinea exprimată în grade pentru o mai bună înțelegere a rezultatelor. Conversia de la quaternioni la unghiuri Euler furnizează rezultate aflate în intervalul în intervalul $[-180°,180°]$.

Fig. 2.8 Vehiculul Yamaha colectând datele pentru fuziune

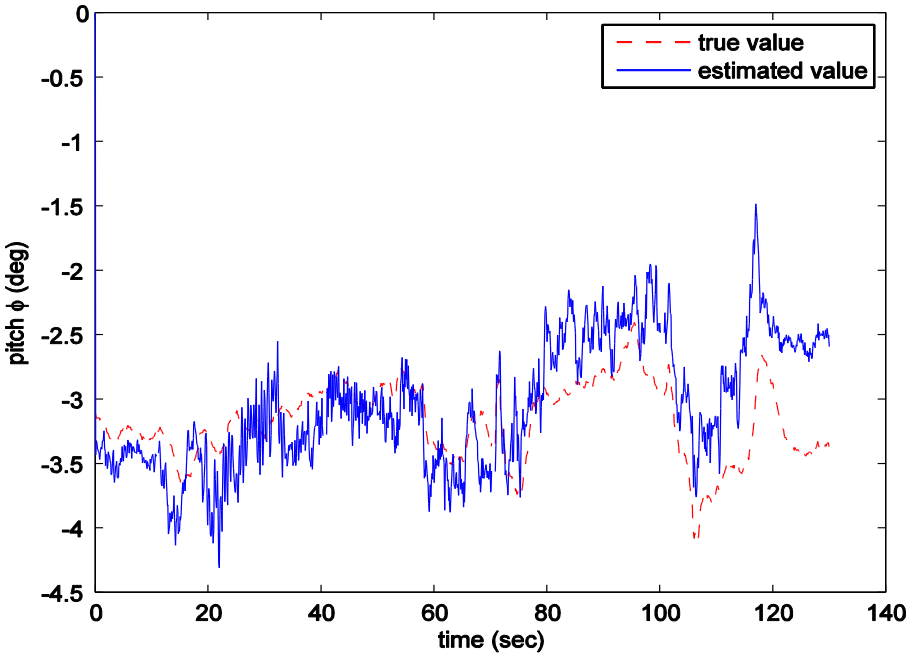

Fig. 2.9 Variația în timp a unghiului de tangaj

Fig. 2.9 prezintă variația în timp a unghiului de tangaj estimat de senzorul Xsens Mti(roșu) și variația în timp a unghiului de tangaj estimat ultilizând filtrul propus(albastru). Fig. 2.10 prezintă prezintă variația în timp a unghiului de ruliu estimat de senzorul Xsens(roșu) și variația în timp a unghiului de tangaj estimat ultilizând filtrul propus(albastru).

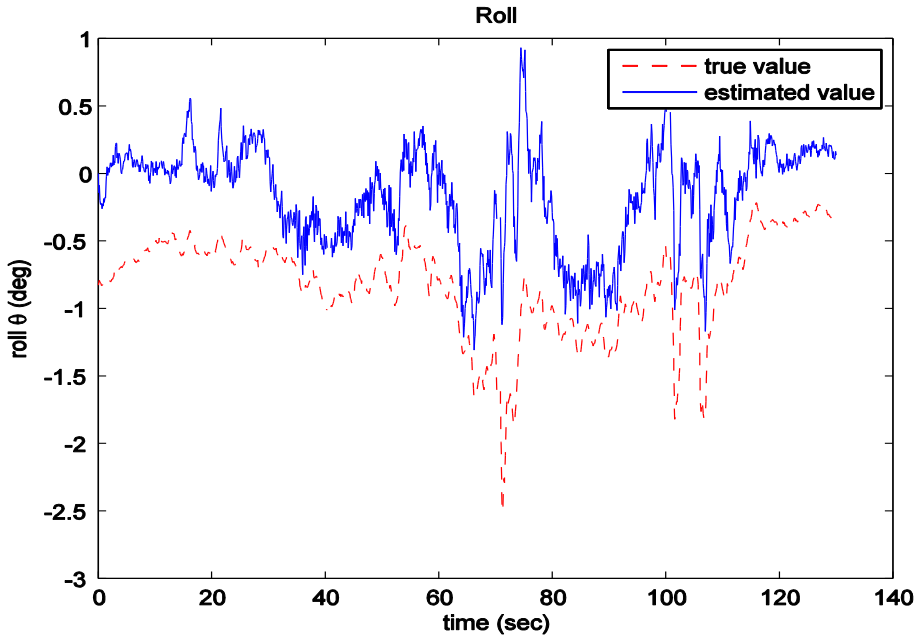

Fig. 2.10 Variația în timp a unghiului de ruliu

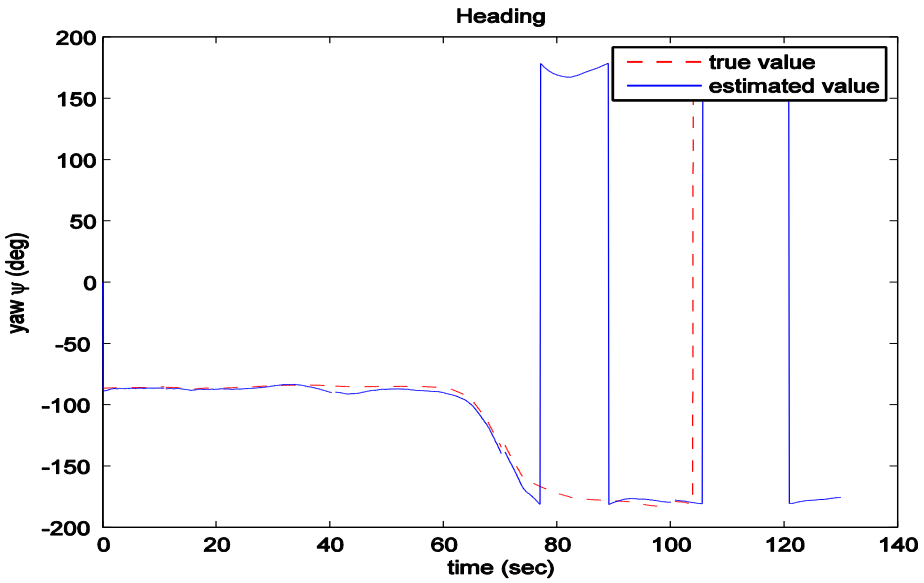

Fig. 2.11 Variația în timp a unghiului de direcție

Fig. 2.11 prezintă variația în timp a unghiului de direcție estimat de senzorul Xsens(roșu) și variația în timp a unghiului de tangaj estimat ultilizând filtrul propus(albastru). Rezulatele filtrului utilizat au furnizat grafice similare celor furnizate de Xsens, se observă că Xsens folosește valori diferite pentru condițiile inițiale, aceste valori afectând diferențele

dintre grafice. Pentru că unghiul de direcție are valori mai mici de $-180°$ se observă o trecere bruscă spre $180°$, cauzată de modul de reprezentare a quaternionilor. Din cele 3 grafice se observă că filtrul propus realizează o bună estimare a atitudinii.

2.4. Concluzii

Roboții mobili și vehiculele autonome au nevoie de senzori, cum ar fi sonarele, laserele, encoderele sau IMU, prentru a opera în regim autonom. Sonarele și laserele sunt folosite pentru a determina distanța dintre vehicul și obstacole, iar encoderele și IMU sunt folosite pentru calculul odometriei platformei.

Există situații în care encoderele oferă informații eronate despre odometria vehiculului și se pot obține informații corecte utilizând IMU. Integrarea semnalelor de la IMU oferă informații despre odometrie corecte doar în cazul intervalelor scurte de timp. În cazul intervalelor mari, micile erori ale semnalelor se acumulează datorită integrărilor și rezultă erori semnificative. Aceste erori sunt caracterizate utilizând variația Allan.

Pentru corectarea erorilor este necesară fuziunea datelor de la mai mulți senzori. Am propus filtrul Kalman indirect pentru a elimina erorile care afectează atitudinea platformei. Filtrul este aplicat stării eroare pentru a minimiza eroarea, apoi eroarea estimată este aplicată stării nominale a sistemului pentru a obține valorile reale ale atitudinii. Am calculat atitudinea prin integrarea datelor de la giroscoape și apoi am corectat atitudinea cu ajutorul erorii calculate de FK.

Am testat filtrul propus utilizând datele colectate, datele au fost colectate în urma rulării vehiculului Yamaha în parcarea laboratorului. Vehiculul Yamaha este dotat cu un calculator care folosește sistemul de operare ROS pentru a colecta datele de la toți senzorii echipați și pentru a funcționa în regim autonom. Am prelucrat în MATLAB datele colectate de la vehicul utilizând algoritmul prezentat în secțiunea anterioară. Am convertit atitudinea, în quaternioni, obținută în urma prelucrării în atitudine exprimată în grade pentru a ușura înțelegerea rezultatelor. Conversia de la quaternioni la unghiuri Euler furnizează rezultate aflate în intervalul în intervalul $[-180°,180°]$.

Am prezentat rezultate care au arătat funcționarea eficientă a filtrului Kalman propus.

Capitolul 3.

3. Conducerea trajectory-tracking a roboților mobili și vehiculelor autonome

3.1 Determinarea modelului cinematic al roboților mobili și al vehiculelor autonome

Există două modele folosite pentru conducerea WMR și vehiculelor: modelul cinematic și modelul dinamic. Modelul cinematic descrie mișcarea fără a considera forțele implicate. Modelul dinamic descrie mișcarea consiuderând și forțele care acționează asupra roboților.

În continuare vor fi prezentate modelele cinematice ale WMR și vehiculului SEEKUR, care vor fi folosite ulterior la elaborarea conducerii.

3.1.1 Determinarea modelului cinematic al roboților mobili

Se consideră un WMR cu 2 roți motoare și două directoare(2DW/2FW) prezentat în Fig. 3.1, caracterizat de variabila generalizată $q = (q_1, q_2, ..., q_n)$. Se presupune că roțile vehiculului se rotesc fără a aluneca, astfel robotul este supus constângerilor nonholonomice descrise de ecuația

$$A(q) \cdot \dot{q} = 0 \quad (3.1),$$

unde $A(q)$ este matricea asociată constrângerilor.

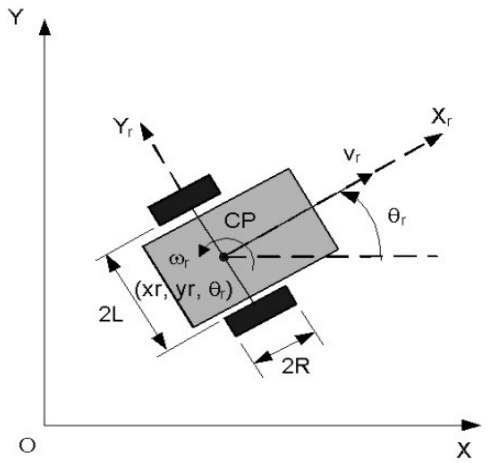

Fig. 3.1 Modelul cinematic al WMR cu 2DW/2FW

Considerăm modelul folosit în [30] și robotul mobil din Fig. 3.1. Oxy este sistemul de coordonate, CPX_rY_r este sistemul de coordonate atașat robotului, distanța dintre CP și centrul

de greutate este d, CP se află la mijlocul distanței dintre cele 2 roți motoare, în acest caz avem constrângerile:

$$\dot{y}_r \cdot \cos\theta_r - \dot{x}_r \cdot \sin\theta_r - d \cdot \dot{\theta} = 0 \; ; \tag{3.2}$$

$$x_r \cdot \cos\theta_r + y_r \cdot \sin\theta_r + L \cdot \dot{\theta} = R \cdot \dot{\phi}_d \; ; \tag{3.3}$$

$$x_r \cdot \cos\theta_r + y_r \cdot \sin\theta_r - L \cdot \dot{\theta} = R \cdot \dot{\phi}_s \; . \tag{3.4}$$

Matricea $A(q)$ devine

$$A(q) = \begin{bmatrix} \sin\theta_r & -\cos\theta_r & d & 0 & 0 \\ \cos\theta_r & \sin\theta_r & L & -R & 0 \\ \cos\theta_r & \sin\theta_r & -L & 0 & -R \end{bmatrix}. \tag{3.5}$$

Configurația robotului poate fi reprezentată utilizând cinci variabile generalizate $q = [x_r \quad y_r \quad \theta_r \quad \phi_r \quad \phi_l]$, unde (x_r, y_r) sunt coordonatele lui CP, θ_r este orientarea, iar ϕ_d, ϕ_s sunt unghiurile roților motoare. Fie matricea S(q) astfel încât

$$S^T(q) \cdot A^T(q) = 0, \tag{3.4}$$

$$S(q) = \begin{bmatrix} \frac{R}{2 \cdot L}(L \cdot \cos\theta_r - d \cdot \sin\theta_r) & \frac{R}{2 \cdot L}(L \cdot \cos\theta_r + d \cdot \sin\theta_r) \\ \frac{R}{2 \cdot L}(L \cdot \sin\theta_r + d \cdot \cos\theta_r) & \frac{R}{2 \cdot L}(L \cdot \sin\theta_r - d \cdot \cos\theta_r) \\ \frac{R}{2 \cdot L} & -\frac{R}{2 \cdot L} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.5}$$

Modelul cinematic al robotului devine

$$\dot{q} = S(q) \cdot \nu, \tag{3.6}$$

unde $\nu = [\nu_d \quad \nu_s]$ reprezintă vitezele unghiulare ale roților motoare.

Din (3.6) se obține

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \\ \dot{\phi}_d \\ \dot{\phi}_s \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\theta_r & \frac{R}{2}\cos\theta_r \\ \frac{R}{2}\sin\theta_r & \frac{R}{2}\sin\theta_r \\ \frac{R}{2 \cdot L} & -\frac{R}{2 \cdot L} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \nu_d \\ \nu_s \end{bmatrix}. \tag{3.7}$$

Se cunoaște relația dintre viteza liniară și viteza unghiulă și vitezele unghiulare ale roților motoare

$$\begin{bmatrix} v_d \\ v_s \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & \frac{L}{R} \\ \frac{1}{R} & -\frac{L}{R} \end{bmatrix} \cdot \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} . \tag{3.8}$$

Utilizând relația (3.8) în (3.7) modelul cinematic se poate simplifica astfel

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} v_r \\ \omega_r \end{bmatrix}, \tag{3.9}$$

în care:

x_r - reprezintă poziția robotului pe axa Ox;

y_r - reprezintă poziția robotului pe axa Oy;

θ_r - reprezintă orientarea robotului;

v_r - reprezintă viteza liniară a robotului;

ω_r - reprezintă viteza unghiulară a robotului.

O altă formă pentru sistemul (3.9) este

$$\begin{cases} \dot{x}_r = v_r \cdot \cos\theta_r \\ \dot{y}_r = v_r \cdot \sin\theta_r \\ \dot{\theta}_r = \omega_r \end{cases} . \tag{3.10}$$

3.1.2. Determinarea modelului cinematic al vehiculelor autonome

Modelul cinematic al vehiculului autonom electric SEEKUR Robot Base descrie mișcarea vehiculului fără a lua în considerare masa sau forțele care acționează asupra vehiculului. SEEKUR este un AV care poate fi modelat considerând un WMR cu 4DW/SW. Modelul cinematic determinat va fi folosit la proiectarea conducerii backstepping și sliding-mode.

Modelul cinematic al vehiculului incluzând alunecarea laterală este prezentat in Fig. 3.2. Fiecare roată are un anumit unghi director δi și un unghi al alunecării βi. Unghiul de alunecare defineste cât de mare este alunecarea laterală și se calculează în funție de viteza longitudinală și liniară a roții

$$\beta_i = \tan^{-1}\left(\frac{v_{y_{wi}}}{v_{x_{wi}}}\right). \tag{3.11}$$

Subscriptul i denotă nr. roții așa cum e arătat in Fig. 3.2, (X_{CG}, Y_{CG},θ) defineste pozitia si orientarea centrului de greutate al vehiculului, (x_{wi},y_{wi}) defineste pozitia rotii i, v și

v_{wi} sunt vitezele vehiculului si a fiecărei roti. β reprezinta unghiul de alunecare a vehiculului, lf, lr sunt distanțele de la centrul de greutate la rotile din fata si din spate.

Luând in considerare alunecările, constrângerile nonholonomice se pot exprima astfel:

$$\dot{x}_{CG} \cdot \sin(\beta + \theta) + \dot{y}_{CG} \cdot \cos(\beta + \theta) = 0 \ ; \tag{3.12}$$

$$\dot{x}_{wi} \cdot \sin(\beta_{wi} + \theta_{wi}) + \dot{y}_{wi} \cdot \cos(\beta_{wi} + \theta_{wi}) = 0 . \tag{3.13}$$

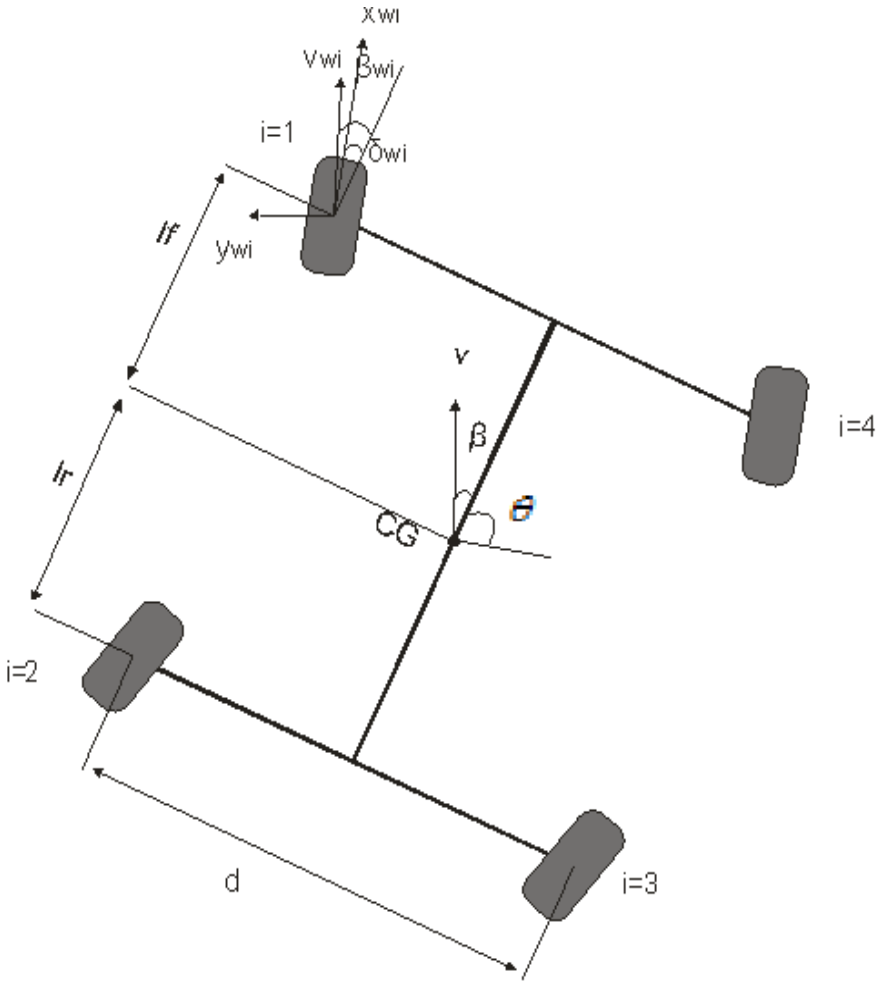

Fig. 3.2 Modelul cinematic al vehiculului autonom SEEKUR

Constrângerile geometrice dintre fiecare roată si centrul de greutate pot fi scrise astfel:

$$\begin{cases} x_{w1} = X_{CG} + l_f \cdot \cos\theta - \dfrac{d}{2} \cdot \sin\theta \\ x_{w2} = X_{CG} - l_f \cdot \cos\theta - \dfrac{d}{2} \cdot \sin\theta \\ x_{w3} = X_{CG} - l_r \cdot \cos\theta + \dfrac{d}{2} \cdot \sin\theta \\ x_{w3} = X_{CG} + l_r \cdot \cos\theta + \dfrac{d}{2} \cdot \sin\theta \end{cases} ; \tag{3.14}$$

$$\begin{cases} y_{w1} = y_{CG} + l_f \cdot \sin\theta - \dfrac{d}{2} \cdot \cos\theta \\ y_{w2} = y_{CG} - l_f \cdot \sin\theta - \dfrac{d}{2} \cdot \cos\theta \\ y_{w3} = y_{CG} + l_r \cdot \sin\theta + \dfrac{d}{2} \cdot \cos\theta \\ y_{w4} = y_{CG} + l_r \cdot \sin\theta + \dfrac{d}{2} \cdot \cos\theta \end{cases}. \tag{3.15}$$

Considerând roțile 1 si 2 si constrângerile obtinem în forma matriceală

$$A_{12} \cdot \dot{q}_0 = 0, \tag{3.16}$$

$$A_{12} = \begin{bmatrix} \sin\phi_{w1} & -\cos\phi_{w1} & -l_f \cdot \cos(\phi_{w1} - \theta) - \dfrac{d}{2} \cdot \sin(\phi_{w1} - \theta) \\ \sin\phi_{w2} & -\cos\phi_{w2} & -l_f \cdot \cos(\phi_{w2} - \theta) - \dfrac{d}{2} \cdot \sin(\phi_{w2} - \theta) \\ \sin\phi_0 & -\cos\phi_0 & 0 \end{bmatrix}, \tag{3.17}$$

$$\dot{q}_0 = \begin{bmatrix} \dot{x}_{CG} \\ \dot{y}_{CG} \\ \dot{\theta} \end{bmatrix}, \tag{3.18}$$

$$\phi_{wi} = \beta_{wi} + \delta_{wi} + \theta \ , \tag{3.19}$$

$$\phi_0 = \beta + \delta . \tag{3.20}$$

Considerând o mișcare planară, corp rigid și fără alunecarea roții, modelul poate fi aproximat de modelul bicicletei(Fig. 3.3)

$$\begin{bmatrix} \sin(\delta_f + \theta) & -\cos(\delta_f + \theta) & -l_f \cdot \cos\delta_f \\ \sin(\delta_r + \theta) & -\cos(\delta_r + \theta) & -l_f \cdot \cos\delta_r \\ \sin(\beta + \theta) & -\cos(\beta + \theta) & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_{CG} \\ \dot{y}_{CG} \\ \dot{\theta} \end{bmatrix} = 0 . \tag{3.21}$$

Se poate obține vectorul $\dot{q}_0$ care satisface relația

$$\begin{bmatrix} \dot{x}_{CG} \\ \dot{y}_{CG} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\beta + \theta) \\ \sin(\beta + \theta \\ \dfrac{\cos\beta(\tan\delta_f - \tan\delta_r)}{l_f + l_r} \end{bmatrix} \cdot v \quad , \tag{3.22}$$

în care:

$$\beta = \arctan \frac{l_f \cdot \tan\delta_r + l_r \cdot \tan\delta_f}{l_f + l_r},$$

v - este viteza liniară a vehiculului.

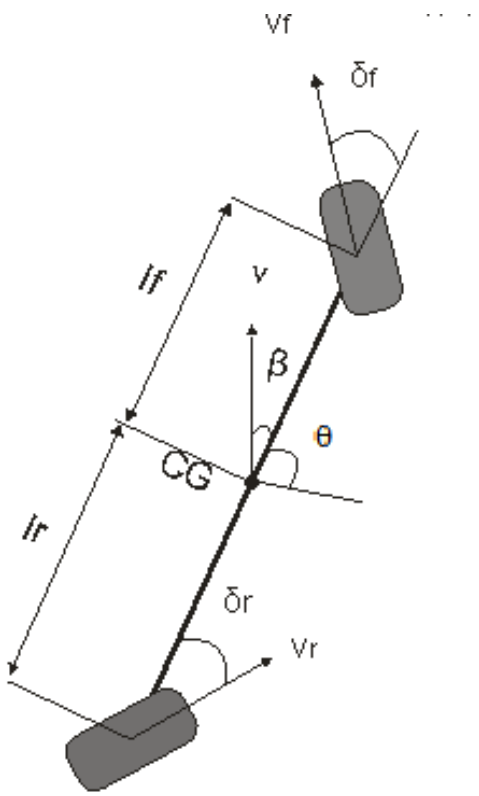

Fig. 3.3 Modelul cinematic al bicicletei

Presupunem că unghiul roților vehiculului este limitat de formula

$$-\frac{\pi}{2} \le \delta_f, \delta_r \le \frac{\pi}{2}. \tag{3.23}$$

Presupunem de asemenea ca vehiculul se deplasează utilizând manevra cu alunecare laterală zero(Zero-side-slip maneuver), propusă de Danwei și Feng[15]. În acest caz unghiul de alunecare este 0 pe întreaga durată a deplasării, iar orientarea vehiculului θ se potrivește cu unghiul tangent la traiectoria dorită,

$$\theta(t) = \theta_d(t), t: 0 \to t_f. \tag{3.24}$$

Unghiurile roților, în acest caz, sunt $\delta_f = -\delta_r$, situație prezentată în Fig. 3.3.

Modelul cinematic al vehiculului (3.22) în cazul manevrei cu alunecare laterală zero devine

$$\begin{bmatrix} \dot{x}_{CG} \\ \dot{y}_{CG} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\beta+\theta) \\ \sin(\beta+\theta \\ \dfrac{(\tan\delta_f - \tan\delta_r)}{2 \cdot l_f} \end{bmatrix} \cdot v. \tag{3.25}$$

Modelul (3.25) poate fi scris și în forma

$$\begin{cases} \dot{x}_{CG} = v \cdot \cos\theta \\ \dot{y}_{CG} = v \cdot \sin\theta \\ \dot{\theta} = v \cdot \tan\delta_f / L \end{cases}, \tag{3.26}$$

unde

v- reprezintă viteza liniară;

δ_f - reprezintă unghiul roții din față;

(x_{CG}, y_{CG}) -reprezintă coordonatele carteziene ale centrului vehiculului;

θ -reprezintă unghiul dintre direcția de mers și axa Ox;

$L = 2 \cdot l_f$ - reprezintă distanța dintre roți.

3.2. Conducerea sliding-mode în timp continuu

Chiar dacă conducerea sliding-mode a fost inventată în anii 60, folosirea ei în exteriorul Rusiei a început abia după anii 70. A evoluat într-o metodă de elaborare a comenzilor care poate fi aplicată unui număr foarte mare de sisteme și în special pentru sistemele neliniare.

Conducerea sliding mode este o conducere bazată pe "modul alunecare", iar o mişcare de alunecare apare când starea sistemului trece prin comutaţii multiple şi este condusă spre starea dorită.

O lege de conducere este definită pentru a conduce traiectoria stării sistemului intr-o stare prescrisă a suprafeţei în spaţiul stărilor şi să menţină traiectoria stării sistemului pe această traiectorie. Această suprafată se numeşte suprafaţa de comutaţie.

Controlerul sliding mode este un controler în buclă închisă discontinuă, care conţine comutaţii la frecvenţe mari. Avantajul major îl reprezintă robusteţea la variaţia parametrilor şi perturbaţii. Principala problemă a controlului sliding mode este apariţia fenomenului de chattering. Fenomenul de chattering poate cauza probleme şi se încearcă reducerea lui.

Alt avantaj major al conducerii sliding-mode îl reprezintă faptul că sistemul se comportă ca un sistem de ordin redus în comparație cu procesul și faptul că alunecarea pe suprafețele de comutație nu este afectată de incertitudini de model sau perturbații.

Conducerea sliding-mode a fost analizată de cercetători ca Utkin în [101], [102], Gao în [31], Khalil în [57]. Mai mulți cercetători din [11], [12], [14], [17], [18], [49], [63], [65], [91], [92], [93], [94], [95], [96] și [103] au ales conducerea sliding-mode pentru conducerea roboților mobili și vehiculelor autonome, caracterizate de sisteme nonholonomice. În [62] a fost tratată problema reducerii efectului de chattering în conducerea sliding-mode.

Se consideră sistemul nelinear affine

$$\dot{x}(t) = f(t,x) + g(t,x) \cdot u(t), \tag{3.26}$$

în care: $x(t) \in R^n, u(t) \in R^m$, $f(t,x)) \in R^{n\times n}$ și $g(t,x) \in R^{n\times m}$. Semnalul de comandă $u(t,x)$ are următoarea formă

$$u_i(t,x) = \begin{cases} u_i^+(t,x), dacă\, s_i > 0 \\ u_i^-(t,x), dacă\, s_i < 0 \end{cases}, i = 1,2,...,m. \tag{3.27}$$

În care: $s_i(0)$ este suprafața de alunecare cu numărul i și

$$s(x) = [s_1(x) \quad s_2(x) \quad .... \quad s_m(x)]^T = 0, \tag{3.28}$$

este colectorul de alunecare. Datorită faptului că $u_i(t,x)$ este discontinuă pe suprafața $s_i(x)$, această suprafață se numește suprafață de comutație.

Definiție 1: Fie $S = \{x|\, s(x) = 0\}$ o suprafață de comutație care include originea $x = 0$. Dacă pentru orice $x_0 \in S$, există $x(t) \in S$ pentru orice $t > t_0$, atunci $x(t)$ este un sistem sliding-mode.

Definiție 2: Dacă există sliding-mode pe $S = \{x|\, s(x) = 0\}$ atunci suprafața de comutație S se numește suprafață de alunecare.

Definiție 3: Existența conducerii sliding-mode necesită stabilitatea traiectoriilor stărilor spre suprafața de alunecare $S = \{x|\, s(x) = 0\}$ cel puțin în vecinătatea lui S. Această condiție de suficiență pentru sliding-mode se numește condiția de atingere.

Cea mai mare vecinătate a lui S pentru care condiția de atingere e satisfăcută se numește zona de atracție.

Definiție 4: Traiectoria stării condiționată de condiția de atingere se numește faza de atingere.

Problema conducerii necesită calcularea funcțiilor continue u_i^+și u_i^-, și suprafața de alunecare $s(x) = 0$ astfel încât sistemul în buclă închisă prezintă sliding-mode pe suprafața de alunecare.

Proiectarea sliding-mode poate fi divizată în 2 faze. Prima fază constă în proiectarea suprafeței de alunecare care asigură evoluția dorită a sistemului. Faza a doua constă în proiectarea legii discontinue de conducere care forțează sistemul pe traiectoria dorită.

Un sistem de ordinul n cu m intrări va avea $2^m - 1$ suprafețe de comutație:

- m suprafețe de dimensiunea (n-1)ș

- Se consideră intersecția a 2 suprafețe S_i, S_j, $i \neq j$. Intersecția lor este o suprafața de comutație de dimensiune (n-2). Numarul total de astfel de intersecții este numărul de combinații de m suprafețe S_i luate câte 2 în același timp

$$\begin{bmatrix} m \\ 2 \end{bmatrix} = \frac{m!}{(m-2)! \cdot 2!} = \frac{m(m-1)}{2} \quad . \tag{3.29}$$

Suprafețele de comutație sunt descrise de

$$S_{ij} = S_i \cap S_j .$$

- Intersecția a 3 suprafețe S_i, S_j și S_k este o suprafață de comutație S_{ijk} de dimensiune (n-3). Există $\binom{m}{3}$ astfel de suprafețe.
- Există o singură suprafață S_e de dimensiune (n-m), care e intersecția suprafețlor $S_i, i = 1, \dots, m$ luate împreună adică

$$S_e = \{x | s(x) = 0\} = S_1 \cap S_2 \cap \dots \cap S_m . \tag{3.30}$$

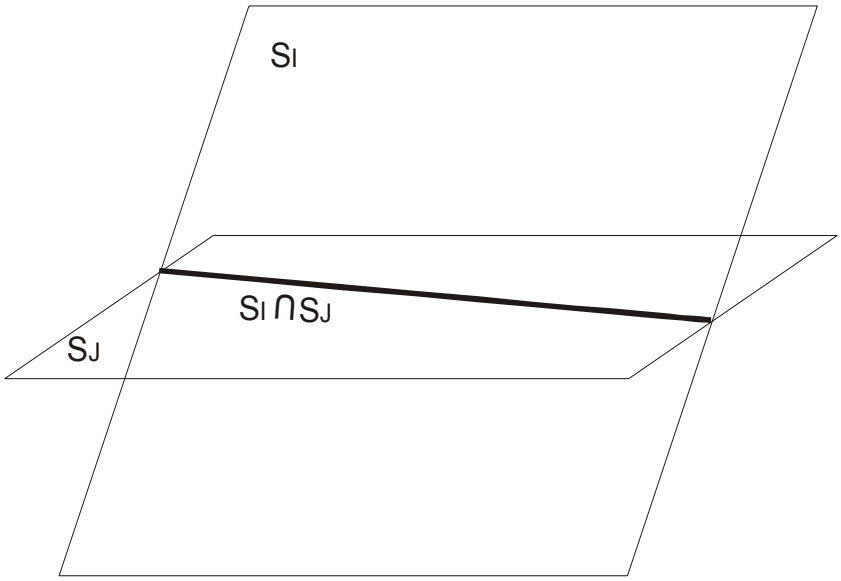

Fig. 3.4 Interpretarea geometrica a suprefețelor de comutație

Metoda lui Filippov este una din metodele posibile pentru determinarea evoluției sistemului în sliding-mode. O altă tehnică aplicabilă sistemelor MIMO este conducerea echivalentă propusă de Utkin, [102].

Suprafețele de alunecare pot fi atât liniare cât și neliniare. Teoria proiectării suprafețelor de alunecare pentru sisteme cu dinamici liniare a fost cercetată amănunțit, dar în cazul sistemelor neliniare este încă o problemă deschisă.

Se consideră sistemul

$$\dot{x} = A \cdot x + B \cdot u \,. \tag{3.31}$$

Se presupune că există o transformată nesingulară Q astfel încât

$$Q \cdot B = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}, \tag{3.32}$$

în care B_2 este o matrice m*m nesingulară. Sistemul se transformă în

$$\begin{cases} \dot{x}_1 = A_{11} \cdot x + A_{12} \cdot x_2 \\ \dot{x}_2 = A_{21} \cdot x + A_{22} \cdot x_2 + B_2 \cdot u \end{cases}, \tag{3.33}$$

în care:

$$x_1 \in R^{n-m};$$

$$x_2 \in R^{m};$$

Suprafața de alunecare poate fi scrisă ca

$$s(x) = \sigma_1 x_1 + \sigma_2 x_2 \,. \tag{3.34}$$

Se presupune că σ_2 e nesingulară și în regimul sliding-mode avem

$$\sigma_1 x_1 + \sigma_2 x_2 = 0 \,. \tag{3.35}$$

Dacă x_2 este dependent liniar de x_1 și sistemul satisface

$$\begin{cases} \dot{x}_1 = A_{11} \cdot x + A_{12} \cdot x_2 \\ \dot{x}_2 = -K \cdot x_1 \end{cases}, \tag{3.36}$$

în care: $K = \sigma_2^{-1}\sigma_1$.

Se obține astfel un sistem de ordin (n-m), iar dinamica sistemului este

$$\dot{x}_1 = (A_{11} \cdot x + A_{12} \cdot K_2) x_1 \,. \tag{3.37}$$

Se observă că s-a obținut o simplificare a problemei de proiectare. După calcularea lui K, suprafața dorită poate fi proiectată astfel

$$s(x) = \sigma_2 [K, I] x \,. \tag{3.38}$$

Proiectarea suprafețelor variabile în timp pentru urmărirea traiectoriilor [90] se realizează pentru un sistem cu o intrare, proiectând suprafața conform lărgimii de bandă dorite

$$s(x,t) = \left(\frac{d}{dt} + \sigma \right)^{n-1} x = 0 \,, \tag{3.39}$$

în care:

x – este eroarea de urmărire

σ - este o constantă pozitivă care determină lungimea de bandă în buclă închisă.

Se vede că s depinde doar de eroarea x.

După calculul suprafețelor sliding-mode este necesară rezolvarea problemei de aducere a sistemului la suprafața dorită. Este necesară calcularea unei comenzi în buclă închisă $u: R^n \rightarrow R^m$, care să conducă starea x spre suprafață și să o mențină acolo.

Legile de aducere sunt: metoda comutației directe, metoda funcției Lyapunov sau legea de aducere a lui Gao[31].

Metoda comutației directe consideră suficient satisfacerea condiției

$$s_i \dot{s}_i < 0, i = 1, \dots, m \ . \tag{3.40}$$

Este o lege globală dar nu garantează aducerea în timp finit.

Metoda Lyapunov presupune alegerea unei funcții candidat cum ar fi

$$V(x,t) = \frac{1}{2} s^T s. \tag{3.41}$$

Legea de aducere este dată de

$$\dot{V}(x,t) < 0. \tag{3.42}$$

Sliding-mode este garantat doar pe intersecția tuturor suprafețelor.

Gao şi Hung[31] au propus urmatoarea lege de conducere

$$s = -Q \cdot \mathrm{sgn}(s) - P \cdot h(s), \tag{3.43}$$

unde:

$$Q=diag[q_1,q_2,\dots,q_m];\ q_i>0;$$

$$sgn(s)=[sgn(s_1),\ sgn(s_2),\dots,sgn(s_m)]^{T;}$$

$$P=diag[p_1,p_2,\dots,p_m];\ p_i>0;$$

$$h(s)=[h_1(s_1),h_2(s_2),\dots,h_m(s_m)]^{T;}$$

$$s_i * h_i(s_i)>0; h_i(0)=0;$$

$$\mathrm{sgn}(s_i) = \begin{cases} -1 & dacă \quad s_i < 0 \\ 1 & dacă \quad s_i > 0 \end{cases}.$$

O formă practică a acestei legi poate fi definită ca

$$s = -Q \cdot \mathrm{sgn}(s) - P \cdot s \ . \tag{3.44}$$

Legea de conducere se poate obține prin metoda legii de aducere, comanda se obține calculând derivata lui $s(x)$:

$$\dot{s} = \frac{\partial s}{\partial x}((A(x) + B(x)u) = -Qsgn(s) - Ph(s); \tag{3.45}$$

$$u = -\left(\frac{\partial s}{\partial x} A(x) + Qsgn(s) + Ph(s)\right)(\frac{\partial s}{\partial x} B(x))^T. \tag{3.46}$$

3.2.1 Conducerea sliding- mode în timp continuu a roboților mobili cu 2 roți motoare și 2 libere (WMR 2DW/2FW)

Arhitectura conducerii sliding-mode în timp continuu a WMR cu 2 DW/2FW este prezentată în Fig. 3.5. Această arhitectură de conducere permite robotului să urmărească o traiectorie dorită cu un profil de viteză impus. Modulul Odometry din Fig. 3.5 primește datele de la encoderele robotului și calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în softul ARIA de la Mobile Robots și datele calculate pot fi obținute apelând în program funcțiile din ARIA.

Trajectory-tracking consideră că robotul urmărește un WMR virtual care se deplasează pe traiectoria dorită cu profilul de viteză impus. Considrând că traiectoria dorită $q_d(t) = [x_d \quad y_d \quad \theta_d]^T$ este generată de un WMR virtual(Fig. 3.6).

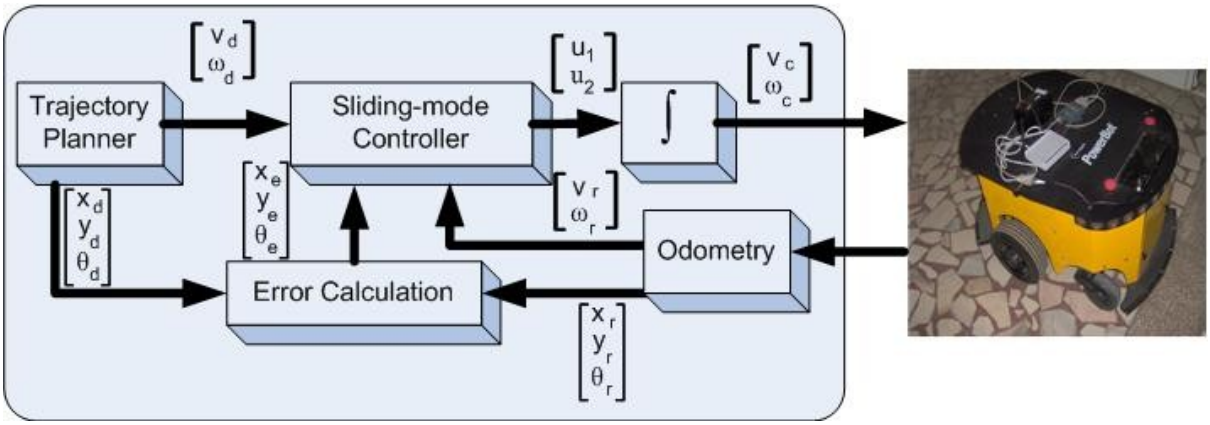

Fig. 3.5 Arhitectura conducerii sliding-mode a WMR 2DW/2FW

Modelul cinematic al robotului virtual este

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \omega_d \end{cases}, \tag{3.47}$$

unde (x_d, y_d) reprezintă coordonatele Carteziene ale centrului geometric, v_d este viteza liniară, θ_d reprezintă orientarea și ω_d reprezintă viteza unghiulară.

Când robotul este condus pentru a urmări o anumită traiectorie se obțin erori de urmărire(Fig. 3.6). Erorile de urmărire[56] sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix}. \tag{3.48}$$

Dinamica erorilor de urmărire este

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \omega_d \cdot x_e \\ \dot{\theta}_e = \omega_r - \omega_d \end{cases}. \tag{3.49}$$

Se presupune că $|\theta_e| < \pi/2$, astfel orientarea robotului nu este perpendiculară pe traiectoria dorită.

Considerând erorile de urmărire (3.48) și derivata lor (3.49) se definesc suprafețele de comutație:

$$\begin{cases} s_1 = \dot{x}_e + k_1 \cdot x_e \\ s_2 = \dot{y}_e + k_2 \cdot y_e + k_0 \cdot \mathrm{sgn}(y_e) \cdot \theta_e \end{cases}, \tag{3.50}$$

unde $k_1, k_2, k_3 \geq 0$ sunt parametri constanți pozitivi, x_e, y_e, θ_e sunt erorile de urmărire din (3.48).

Dacă s_1 converge la zero, atunci x_e converge la zero. Dacă s_2 converge la 0, atunci $\dot{y}_e = -k_2 \cdot y_e + k_0 \cdot \mathrm{sgn}(y_e) \cdot \theta_e$. Dacă $y_e > 0$ atunci $\dot{y}_e < 0$ doar dacă $k_0 < k_2 \cdot |y_e| / |\theta_e|$. Se observă că dacă y_e și $\dot{y}_e$ converg la zero atunci θ_e converge la zero.

Derivatele suprafețelor devin

$$\begin{cases} \dot{s}_1 = \ddot{x}_e + k_1 \cdot \dot{x}_e \\ \dot{s}_2 = \ddot{y}_e + k_2 \cdot \dot{y}_e + k_0 \cdot \mathrm{sgn}(y_e) \cdot \dot{\theta}_e \end{cases}. \tag{3.51}$$

Legea de conducere este folosită în forma

$$\dot{s} = -Q \cdot \mathrm{sgn}(s) - P \cdot s, \tag{3.52}$$

unde $$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}, \quad P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}, \quad Q_1, Q_2, P_1, P_2 \geq 0,$$

$$\mathrm{sgn}(s) = [\mathrm{sgn}(s_1) \quad \mathrm{sgn}(s_2)]^T, \; s = [s_1 \quad s_2]^T.$$

Din (3.48), (3.49), (3.50), (3.51) și (3.52) se obține legea de comandă sliding mode:

$$\dot{v}_c = \frac{-Q_1 \cdot sign(s_1) - P_1 \cdot s_1 - k_1 \cdot \dot{x}_e - \dot{\omega}_d \cdot y_e - \omega_d \cdot \dot{y}_e + v_r \cdot \dot{\theta}_e \cdot \sin\theta_e + \dot{v}_d}{\cos\theta_e}; \tag{3.53}$$

$$\omega_c = \frac{-Q_2 \cdot sign(s_2) - P_2 \cdot s_2 - k_2 \cdot \dot{y}_e - \dot{v}_r \cdot \sin\phi_e + \dot{\omega}_d \cdot x_e + \omega_d \cdot \dot{x}_e}{v_r \cdot \cos\phi_e + k_0 \cdot \mathrm{sgn}(y_e)} + \omega_d. \tag{3.54}$$

Fie $V = \frac{1}{2} \cdot s^T \cdot s$ o funcție Lyapunov. Derivata acestei funcții este

$$\dot{V} = s_1 \cdot \dot{s}_1 + s_2 \cdot \dot{s}_2 = s_1 \cdot \left(-Q_1 \cdot \mathrm{sgn}(s_1) - P_1 \cdot s_1\right) + s_2 \cdot \left(-Q_2 \cdot \mathrm{sgn}(s_2) - P_2 \cdot s_2\right). \tag{3.55}$$

Derivata funcției Lyapunov este negativă semi-definită dacă alegem parametrii $Q_i, P_i \geq 0$.

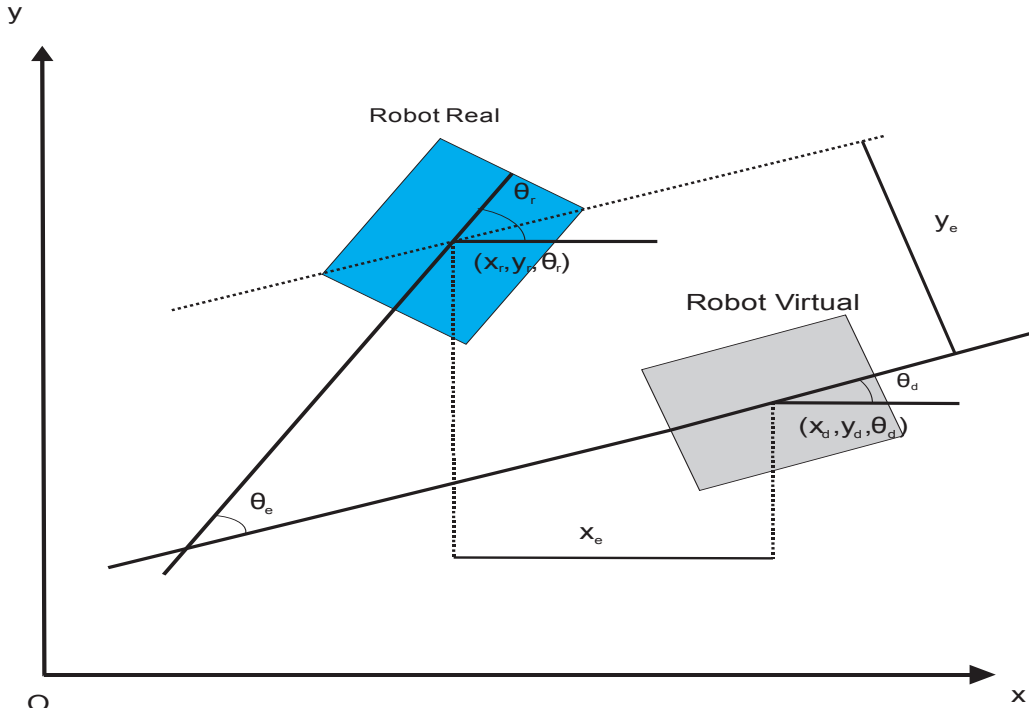

Fig. 3.6 Erorile de urmărire în cazul roboților mobili

3.2.2 Conducerea sliding- mode a vehiculului autonom cu 4 roți motoare și 4 directoare (4DW/SW)

Calculul comenzii se realizează pornind de la modelul cinematic simplificat al vehiculului din ecuația(3.26) și considerând un vehicul virtual care se deplasează pe traiectoria dorită. Problema urmăririi traiectoriei presupune proiectarea unui controler care să calculeze viteza liniară și viteza unghiulară care să permită vehiculului real să urmărească traiectoria vehiculului virtual cu erori de poziționare cât mai mici.

Arhitectura conducerii sliding-mode în timp continuu a AV 4DW/SW este prezentată în Fig. 3.7. Modulul Odometry din Fig. 3.7 primește datele de la encoderele vehiculului și calculează poziția, orientarea, viteza liniară a vehiculului și unghiul roților vehiculului. Acest modul este implementat în softul ARIA de la Mobile Robots și datele calculate pot fi obținute apelând în program funcțiile din ARIA.

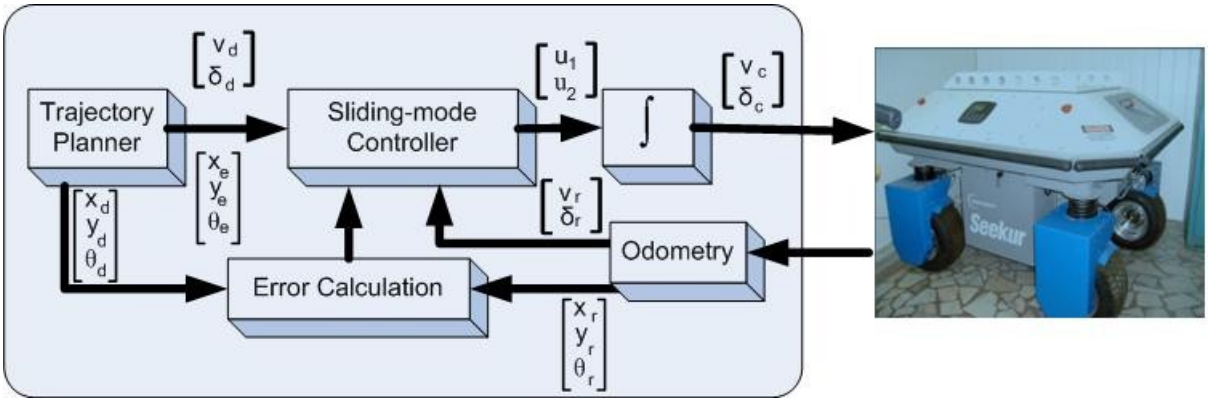

Fig. 3.7 Arhitectura conducerii sliding-mode în timp continuu a AV 4DW/SW

Modelul cinematic al vehiculului virtual este

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \dfrac{v_d \cdot \tan\delta_d}{L} \end{cases}, \tag{3.56}$$

unde (x_d, y_d) reprezintă coordonatele Carteziene ale centrului geometric, v_d este viteza liniară, θ_d reprezintă orientarea și δ_d reprezintă unghiul roților.

Erorile de urmărire[56] sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix}. \tag{3.57}$$

Dinamica erorilor de urmărire este

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \dfrac{v_d}{L} \cdot \tan\delta_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \dfrac{v_d \cdot \tan\delta_d}{L} \cdot x_e \\ \dot{\theta}_e = \dfrac{v_r \cdot \tan\delta_r}{L} - \dfrac{v_d \cdot \tan\delta_d}{L} \end{cases}. \tag{3.58}$$

Erorile de urmărire ale vehiculului autonom electric SEEKUR, considerând modelul cinematic simplificat, reprezentat de modelul cinematic al bicicletei sunt descrise în Fig. 3.8.

Considerând erorile de urmărire (3.57) și derivata lor (3.58) se definesc suprafețele de comutație:

$$s_1 = \dot{x}_e + k_1 \cdot x_e; \tag{3.59}$$

$$s_2 = \dot{y}_e + k_2 \cdot y_e + k_0 \cdot \mathrm{sgn}(y_e) \cdot \theta_e; \tag{3.60}$$

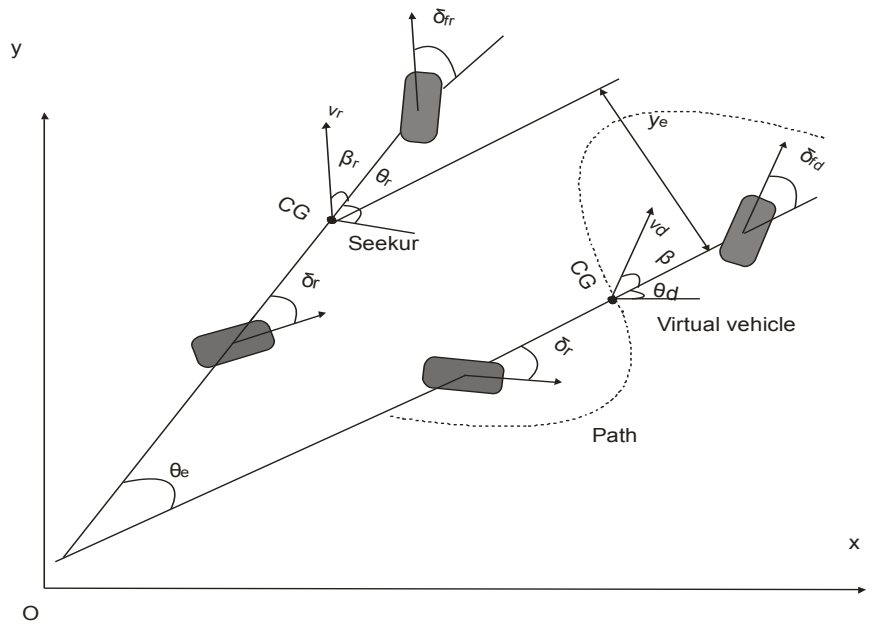

Fig. 3.8 Erorile de urmărire în cazul vehiculului SEEKUR

Legea de conducere este folosită în forma

$$\dot{s} = -Q \cdot \mathrm{sgn}(s) - P \cdot s \ . \tag{3.61}$$

Din (3.56), (3.57), (3.58), (3.60) și (3.61) se obține legea de comandă sliding mode:

$$\dot{v}_c = \frac{-Q_1 \cdot sign(s_1) - P_1 \cdot s_1 - k_1 \cdot \dot{x}_e - \dot{\omega}_d \cdot y_e - \omega_d \cdot \dot{y}_e + v_r \cdot \dot{\theta}_e \cdot \sin\theta_e + \dot{v}_d}{\cos\phi_e} \ ; \tag{3.62}$$

$$\delta_c = \arctan\left(\frac{L \cdot (-Q_2 \cdot sign(s_2) - P_2 \cdot s_2 - k_2 \cdot \dot{y}_e - \dot{v}_r \cdot \sin\theta_e + \dot{\omega}_d \cdot x_e + \omega_{d.} \cdot \dot{x}_e)}{v_r \cdot \cos\theta_e + k_0 \cdot \mathrm{sgn}(y_e)} + \frac{L}{v_r}\omega_d\right). \tag{3.63}$$

Fie $V = \frac{1}{2} \cdot s^T \cdot s$ o funcție Lyapunov. Derivata acestei funcții este

$$\dot{V} = s_1 \cdot \dot{s}_1 + s_2 \cdot \dot{s}_2 = s_1 \cdot \left(-Q_1 \cdot \mathrm{sgn}(s_1) - P_1 \cdot s_1\right) + s_2 \cdot \left(-Q_2 \cdot \mathrm{sgn}(s_2) - P_2 \cdot s_2\right). \tag{3.64}$$

Derivata funcției Lyapunov este negativă semi-definită dacă alegem parametrii $Q_i, P_i \geq 0$.

3.2 Conducerea sliding-mode în timp discret

Conducerea sliding-mode a fost aplicată cu succes la rezolvarea problemei conducerii sistemelor nonholonomice. Conducerea sliding-mode în timp discret a fost aplicată mai rar în cazul sistemelor nonholonomice decât conducerea sliding-mode pentru timp continuu și pentru că algoritmul de conducere este implementat pe sisteme digitale se va folosi conducerea sliding-mode pentru timp discret.

Conducerea sliding-mode în timp discret a fost propusă de [16], [20], [21], [22] [32], [49], [73], [81], [83], [88], [105], [106] și [108]. În cazul conducerii în timp discret accesul la informațiile sistemului este restricționat la anumite perioade de timp, iar comenzile pot fi trimise dor în aceste perioade. Se presupune că semnalele discrete sunt obținute cu metoda zero order hold. Conducerea sliding-mode în timp discret presupune calculul unei suprafețe de alunecare și a unei legi de conducere care să mențină traiectoria în apropierea suprafeței.

Se consideră sistemul discret

$$x[k+1] = Ax[k] + b \cdot u[k], \tag{3.65}$$

unde x este un vector cu n component, u este un scalar, iar A și b au dimensiunile corespunzătoare.

Un controller cu structură variabilă în timp discret a cărui evoluție pornește de la orice stare inițială , conduce traiectoria spre un plan de comutație și ajunge la acest plan în timp finit și după ce sistemul a ajuns în planul de comutație traiectoria descrie o mișcare în zigzag în acest plan, se numește quasi-sliding mode.

O lege de aducere convenabilă se definește astfel

$$s[k+1] - s[k] = -qT_s s[k] - \varepsilon T_s \operatorname{sgn}(s[k]), \tag{3.63}$$

unde $\varepsilon, q > 0$ sunt constant și $1 - qT_s > 0$, T_s este perioada de eșantionare.

Suprafața de comutație în cazul ideal este

$$c^T Ax[k] + c^T bu[k] = s[k] = 0. \tag{3.64}$$

Comanda este dată de formula

$$u = -(c^T b)^{-1} c^T Ax[k]. \tag{3.65}$$

Ecuația dinamică pentru quasi-sliding-mode ideal este

$$\begin{cases} x[k+1] = [I - b(c^T b)^{-1} c^T]Ax[k] \\ c^T x[k] = 0 \end{cases}. \tag{3.66}$$

Legea de conducere este

$$s[k+1] - s[k] = -qT_s s[k] - \varepsilon T_s \operatorname{sgn}(s[k]) = c^T Ax[k] + c^T bu[k] - c^T x[k]. \tag{3.67}$$

Comanda calculată este

$$u = -(c^T b)^{-1}[c^T Ax[k] - c^T x[k] + qT_s c^T x[k] + \varepsilon T_s \operatorname{sgn}(c^T x[k])]. \tag{4.46}$$

3.3.1 Conducerea sliding-mode în timp discret a WMR cu 2 DW/2FW

Controllerul primește de la modulul de planificare a traiectoriei viteza liniară și viteza unghiulară dorite, și erorile de poziționare de la modulul de calcul al erorilor și calculează

comanda pentru viteza liniară și viteza unghiulară. Conducerea sliding-mode în timp discret are structura din Fig. 3.9. Modulul Odometry din Fig. 3.9 primește datele de la encoderele robotului și calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în softul ARIA de la Mobile Robots și datele calculate pot fi obținute apelând în program funcțiile din ARIA.

Conducerea sliding-mode în timp discret a roboților mobili cu 2DW/2FW este realizată pornind de la ecuațiile cinematice ale centrului de greutate.

Considerând periaoada de eșantionare T_s și metoda zero order hold sistemul neliniar (3.10) poate fi scris în timp discret astfel:

$$\begin{cases} x_r[k+1] = x_r[k] + v_r[k] \cdot \cos\theta_r[k] \cdot T_s \\ y_r[k+1] = y_r[k] + v_r[k] \cdot \sin\theta_r[k] \cdot T_s \\ \theta_r[k+1] = \theta_r[k] + \omega_r[k] \cdot T_s \end{cases} \tag{3.68}$$

Conducerea își propune rezolvarea problemei urmăririi traiectoriei. Rezolvarea acestei probleme necesită proiectarea unui controller capabil de urmărirea traiectoriei dorite. În acest scop se consideră un robot virtual având traiectoria dorită $q_d(t) = [x_d(t) \quad y_d(t) \quad \theta_d(t)]^T$ și următorul model cinematic

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\theta}_d \end{bmatrix} = \begin{bmatrix} \cos\theta_d & 0 \\ \sin\theta_d & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_d \\ \omega_d \end{bmatrix}, \tag{3.69}$$

în care:

x_d - reprezintă poziția dorită pe axa Ox;

y_d - reprezintă poziția dorită pe axa Oy;

θ_d - reprezintă orientarea dorită;

v_d - reprezintă viteza liniară dorită;

ω_d - reprezintă viteza unghiulară dorită.

Sistemul (3.69) poate fi scris ca

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \omega_d \end{cases}. \tag{3.70}$$

Sistemul (3.70) poate fi scris în timp discret astfel

$$\begin{cases} x_d[k+1] = x_d[k] + v_d[k] \cdot \cos\theta_d[k] \cdot T_s \\ y_d[k+1] = y_d[k] + v_d[k] \cdot \sin\theta_d[k] \cdot T_s \\ \theta_d[k+1] = \theta_d[k] + \omega_d[k] \cdot T_s \end{cases}. \tag{3.71}$$

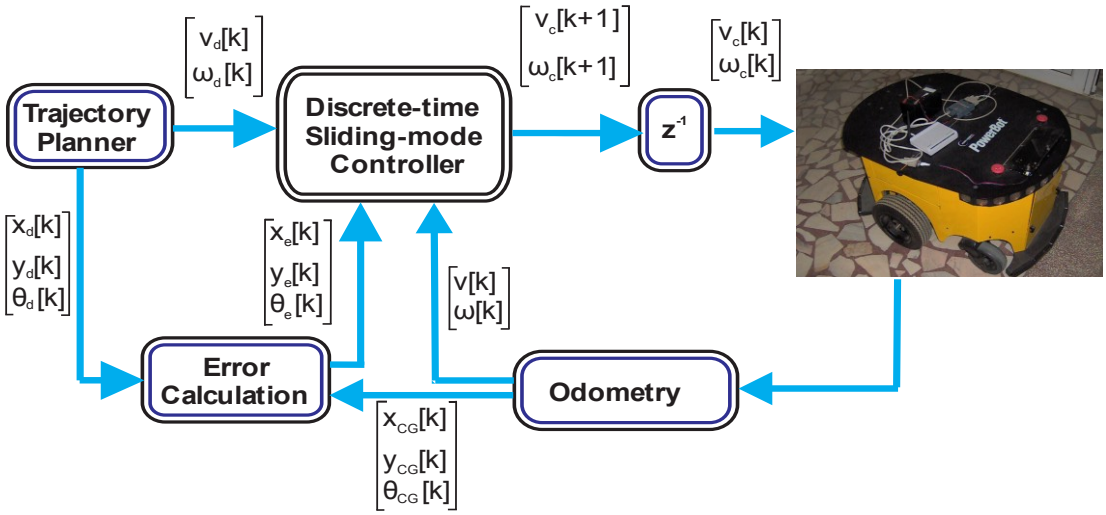

Fig. 3.9 Arhitectura de conducere sliding-mode în timp discret a WMR cu 2DW/2FW

Erorile de urmărire din Fig. 3.6sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \omega_r - \omega_d \end{bmatrix}. \tag{3.72}$$

Aceste erori pot fi scrise ca

$$\begin{cases} x_e[k] = x_r[k] - x_d[k]\cdot\cos\theta_e[k] + y_r[k] - y_d[k]\cdot\sin\theta_e[k] \\ y_e[k] = -x_r[k] - x_d[k]\cdot\sin\theta_e[k] + y_r[k] - y_d[k]\cdot\cos\theta_e[k] \\ \theta_e[k] = \theta_r[k] - \theta_d[k] \end{cases}. \tag{3.73}$$

Dinamica erorilor poate fi exprimată astfel

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \omega_d \cdot x_e \\ \dot{\theta}_e = \dot{\theta}_r - \dot{\theta}_d \end{cases}. \tag{3.74}$$

În timp timp discret dinamica erorilor se exprimă

$$\begin{cases} x_e[k+1] = x_e[k] + (v_r[k]\cdot\cos\theta_e[k] - v_d[k] + y_e[k]\cdot\omega_d[k])\cdot T_s \\ y_e[k+1] = y_e[k] + (v_r[k]\cdot\sin\theta_e[k] - x_e[k]\cdot\omega_d[k])\cdot T_s \\ \theta_e[k+1] = \theta_e[k] + (\omega_r[k] - \omega_d[k])\cdot T_s \end{cases}. \tag{3.78}$$

Controlerul sliding-mode în timp discret este un controler cu structură variabilă, care efectuează măsurători și calculează comenzile la intervale regulate de timp și păstrează semnalul de comandă constant între intervale. O importantă proprietate a conducerii sliding-mode în timp discret este controlul discontinuu.

Gao et al. [32] a propus următoarea condiție pentru existență sliding-mode

$$s[k]\cdot(s[k+1] - s[k]) < 0. \tag{3.79}$$

Legea de conducere propusă de Gao et al. [32] este:

$$s[k+1] = (1 - q \cdot T_s) \cdot s[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s[k]); \tag{3.80}$$

$$0 < 1 - q \cdot T < 1; \tag{3.81}$$

$$0 < \varepsilon \cdot T_s < 1; \tag{3.82}$$

în care:

$T_s > 0$ este perioada de eșantionare;

$\varepsilon > 0$ este viteza de aducere;

$q > 0$ este convergenta exponențială.

Traiectoria sliding-mode evoluează în cvasi-banda sliding-mode

$$|s[k]| < \frac{\varepsilon \cdot T_s}{1 - q \cdot T_s}. \tag{3.83}$$

Se definesc suprafețele de alunecare

$$\begin{cases} s_1[k] = x_e[k+1] + k_1 \cdot x_e[k] \\ s_2[k] = y_e[k+1] + k_2 \cdot y_e[k] + k_0 \cdot sign(y_e[k]) \cdot \theta_e[k] \end{cases}, \tag{3.84}$$

în care:

k_0, k_1, k_2 sunt constante pozitive x_e, y_e și θ_e sunt erorile de urmărire.

Din (3.80) și (3.84) obținem:

$$s_1[k+1] = x_e[k+2] + k_1 \cdot x_e[k+1] = (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]) \tag{3.85}$$

$$\begin{aligned} s_2[k+1] &= y_e[k+2] + k_2 \cdot y_e[k+1] + k_0 \cdot \mathrm{sgn}(y_e[k]) \cdot \theta_e[k] = \\ &(1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]) \end{aligned}. \tag{3.86}$$

Din (3.85) și (3.86) se obțin comenzile pentru viteza liniară și viteza unghiulară astfel:

$$\begin{aligned} v[k+1] = &\frac{1}{\cos\theta_e[k] * T_s} \cdot [-(1 - q_1 \cdot T_s) \cdot s_1[k] + \varepsilon_1 \cdot T_s \cdot \mathrm{sgn}(s_1[k] - x_e[k+1] \cdot (1 + k_1) - \\ &- (v_d[k+1] - v_r[k] \cdot \theta_e[k+1] \cdot \sin\theta_e[k] - w_d[k+1] * y_e[k] - w_d[k] \cdot y_e[k+1]) \cdot T_s] \end{aligned}; \tag{3.87}$$

$$\begin{aligned} \omega[k] = &\frac{1}{v_r[k] \cdot \cos\theta_e + k_0 \cdot \mathrm{sgn}\, y_e[k+1]} \cdot [-(1 - q_2 \cdot T_s) \cdot s_2[k] + \varepsilon_2 \cdot T_s \cdot \mathrm{sgn}(s_2[k]) - \theta_e[k] - \\ &- y_e[k+1] \cdot (k_2 + 1) - (v_r[k+1] \cdot \sin\theta_e[k] + w_d[k+1] \cdot x_e[k] - w_d[k] \cdot x_e[k+1]) \cdot T_s)] + w_d[k] \end{aligned}. \tag{3.88}$$

3.3.2 Conducerea sliding-mode în timp discret a vehiculelor autonome

Conducerea sliding-mode în timp discret a roboților mobili și vehiculelor autonome este realizată pornind de la ecuațiile cinematice ale centrului de greutate. Arhitectura conducerii sliding-mode în timp discret a vehiculului autonom cu 4 DW/SW este prezentată în Fig. 3.10. Modulul Odometry din Fig. 3.10 primește datele de la encoderele robotului și

calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în softul ARIA de la Mobile Robots și datele calculate pot fi obținute apelând în program funcțiile din ARIA.

Considerând periaoada de eșantionare T_s și metoda zero order hold sistemul neliniar (3.26) poate fi scris în timp discret astfel

$$\begin{cases} x_r[k+1] = x_r[k] + v_r[k] \cdot \cos\theta_r[k] \cdot T_s \\ y_r[k+1] = y_r[k] + v_r[k] \cdot \sin\theta_r[k] \cdot T_s \\ \theta_r[k+1] = \theta_r[k] + \dfrac{v_r[k]}{L} \cdot \tan\delta_r[k] \cdot T_s \end{cases} . \tag{3.88}$$

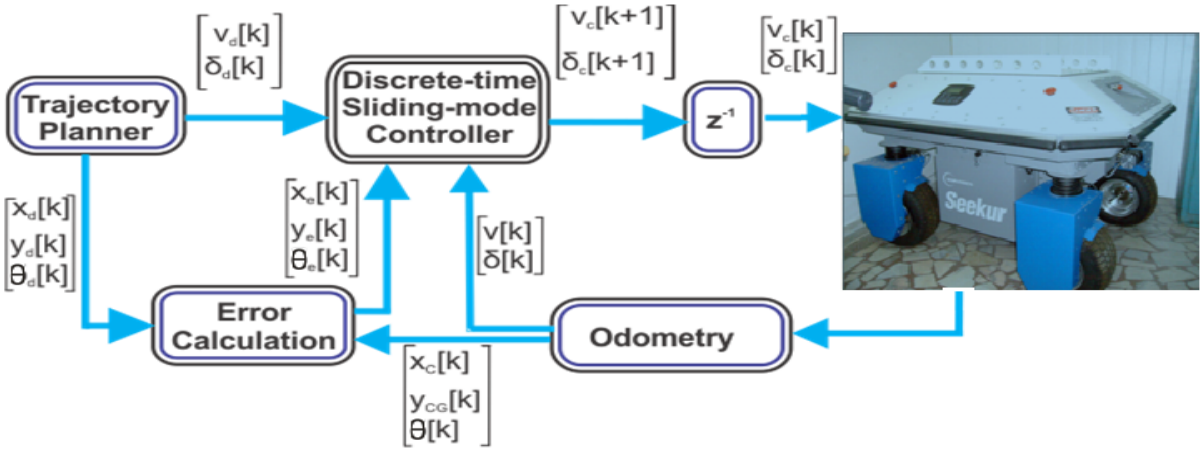

Fig. 3.10 Arhitectura conducerii sliding-mode în timp discret a vehiculului autonom

Conducerea își propune rezolvarea problemei urmăririi traiectoriei. Rezolvarea acestei probleme necesită proiectarea unui controller capabil de urmărirea traiectoriei dorite. În acest scop se consideră un robot virtual având traiectoria dorită $q_d(t) = [x_d(t) \quad y_d(t) \quad \theta_d(t)]^T$ și următorul model cinematic

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \dfrac{v_d}{L} \cdot \tan\delta_d \end{cases} , \tag{3.89}$$

în care:

x_d - reprezintă poziția dorită pe axa Ox;

y_d - reprezintă poziția dorită pe axa Oy;

θ_d - reprezintă orientarea dorită;

v_d - reprezintă viteza liniară dorită;

δ_d - reprezintă unghiul roților.

Sistemul (3.89) poate fi scris în timp discret astfel

$$\begin{cases} x_d[k+1] = x_d[k] + v_d[k] \cdot \cos\theta_d[k] \cdot T_s \\ y_d[k+1] = y_d[k] + v_d[k] \cdot \sin\theta_d[k] \cdot T_s \\ \theta_d[k+1] = \theta_d[k] + \dfrac{v_d[k]}{L} \cdot \tan\delta_d[k] \cdot T_s \end{cases}. \tag{3.90}$$

Erorile de urmărire din Fig. 3.6sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix}. \tag{3.91}$$

Aceste erori pot fi scrise ca

$$\begin{cases} x_e[k] = x_r[k] - x_d[k] \cdot \cos\theta_e[k] + y_r[k] - y_d[k] \cdot \sin\theta_e[k] \\ y_e[k] = -x_r[k] - x_d[k] \cdot \sin\theta_e[k] + y_r[k] - y_d[k] \cdot \cos\theta_e[k] \\ \theta_e[k] = \theta_r[k] - \theta_d[k] \end{cases}. \tag{3.92}$$

Dinamica erorilor poate fi exprimată astfel

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \omega_d \cdot x_e \\ \dot{\theta}_e = \dot{\theta}_r - \dot{\theta}_d \end{cases}. \tag{3.93}$$

În timp timp discret dinamica erorilor se exprimă

$$\begin{cases} x_e[k+1] = x_e[k] + (v_r[k] \cdot \cos\theta_e[k] - v_d[k] + y_e[k] \cdot \dfrac{v_d[k]}{L} \cdot \tan\delta_d[k]) \cdot T_s \\ y_e[k+1] = y_e[k] + (v_r[k] \cdot \sin\theta_e[k] - x_e[k] \cdot \dfrac{v_d[k]}{L} \cdot \tan\delta_d[k] \cdot T) \cdot T_s \\ \theta_e[k+1] = \theta_e[k] + (\dfrac{v_r[k]}{L} \tan\delta_r[k] - \dfrac{v_d[k]}{L} \tan\delta_d[k]) \cdot T_s \end{cases}. \tag{3.94}$$

Controlerul sliding-mode în timp discret este un controler cu structură variabilă, care efectuează măsurători și calculează comenzile la intervale regulate de timp și păstrează semnalul de comandă constant între intervale. O importantă proprietate a conducerii sliding-mode în timp discret este controlul discontinuu.

Gao et al. [32] a propus următoarea condiție pentru existență sliding-mode

$$s[k] \cdot (s[k+1] - s[k]) < 0. \tag{3.95}$$

Legea de conducere propusă de Gao et al. [32] este:

$$s[k+1] = (1 - q \cdot T_s) \cdot s[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s[k]); \tag{3.96}$$

$$0 < 1 - q \cdot T < 1; \tag{3.97}$$

$$0 < \varepsilon \cdot T_s < 1; \tag{3.98}$$

în care:

$T_s > 0$ este perioada de eșantionare;

$\varepsilon > 0$ este viteza de aducere;

$q > 0$ este convergenta exponențială.

Traiectoria sliding-mode evoluează în cvasi-banda sliding-mode

$$|s[k]| < \frac{\varepsilon \cdot T_s}{1 - q \cdot T_s}. \tag{3.99}$$

Se definesc suprafețele de alunecare

$$\begin{cases} s_1[k] = x_e[k+1] + k_1 \cdot x_e[k] \\ s_2[k] = y_e[k+1] + k_2 \cdot y_e[k] + k_0 \cdot sign(y_e[k]) \cdot \theta_e[k] \end{cases}, \tag{3.100}$$

în care: k_0, k_1, k_2 sunt constante pozitive x_e, y_e și θ_e sunt erorile de urmărire.

Din (3.96) și (3.100) obținem:

$$s_1[k+1] = x_e[k+2] + k_1 \cdot x_e[k+1] = (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]); \tag{3.101}$$

$$\begin{aligned} & s_2[k+1] = y_e[k+2] + k_2 \cdot y_e[k+1] + k_0 \cdot \mathrm{sgn}(y_e[k]) \cdot \theta_e[k] = \\ & (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]) \end{aligned} \tag{3.102}$$

Din (3.101) și (3.102) se obțin comenzile pentru viteza liniară și viteza unghiulară astfel:

$$\begin{aligned} v[k+1] = & \frac{1}{\cos\theta_e[k] \cdot T_s} \cdot [-(1 - q_1 \cdot T_s) \cdot s_1[k] + \varepsilon_1 \cdot T_s \cdot \mathrm{sgn}(s_1[k] - x_e[k+1] \cdot (1 + k_1) - \\ & - (v_d[k+1] - v_r[k] \cdot \theta_e[k+1] \cdot \sin\theta_e[k] - w_d[k+1] \cdot y_e[k] - w_d[k] \cdot y_e[k+1]) \cdot T_s] \end{aligned} \tag{3.103}$$

$$\begin{aligned} \delta[k] = & \arctan\Big(\frac{L}{v_r[k] \cdot \cos\theta_e + k_0 \cdot \mathrm{sgn}\, y_e[k+1]} \cdot [-(1 - q_2 \cdot T_s) \cdot s_2[k] + \varepsilon_2 \cdot T_s \cdot \mathrm{sgn}(s_2[k]) - \theta_e[k] - \\ & - y_e[k+1] \cdot (k_2 + 1) - (v_r[k+1] \cdot \sin\theta_e[k] + w_d[k+1] \cdot x_e[k] - w_d[k] \cdot x_e[k+1]) \cdot T_s)] + \frac{L}{v_r} w_d[k]\Big) \end{aligned} \tag{3.104}$$

3.4. Conducerea backstepping

Conducerea backstepping este una dintre cele mai importante tehnici de conducere a proceselor neliniare și poate fi aplicată la numeroase aplicații. Acest tip de conducere este necesar în cazul sistemelor non-holonomice datorită faptului că nu se poate aplica condiția de stabilitate a lui Brockett, [7].

Probelma conducerii backstepping a mai multor tipuri de roboți nonholonomici a fost prezentată într-un număr mare de lucrări [9], [51], [52], [61], [72], [80], [89], [98] și [117].

Backstepping este o procedură recursivă care descompune o problemă de proiectare a întregului sistem în probleme de proiectare pentru sisteme de ordin inferior. Proiectarea backstepping are la bază integratorul backstepping, acesta fiind aplicat recursiv pentru rezolvarea problemei sistemelor complexe, procedură descrisă de Chen, [9], și Khalil, [57].

Integratorul backstepping, metoda backstepping pentru sisteme și conducerea backstepping a vehiculului autonom electric SEEKUR Robot Base sunt prezentate în acest subcapitol.

Integratorul backstepping este un caz special, în proiectarea folosind metoda backstepping. Se consideră sistemul

$$\begin{cases} \dot{\eta} = f(\eta) + g(\eta) \cdot \xi \\ \dot{\xi} = u \end{cases}, \tag{3.105}$$

În care: $[\eta^T \ \xi]^T \in R^{n+1}$ este starea, iar $u \in R$ este comanda, funcțiile $f, g: D \to R^n$ sunt netede într-un domeniu $D \subset R^n$ care conține $f(0) = 0$ și $g(0) = 0$.

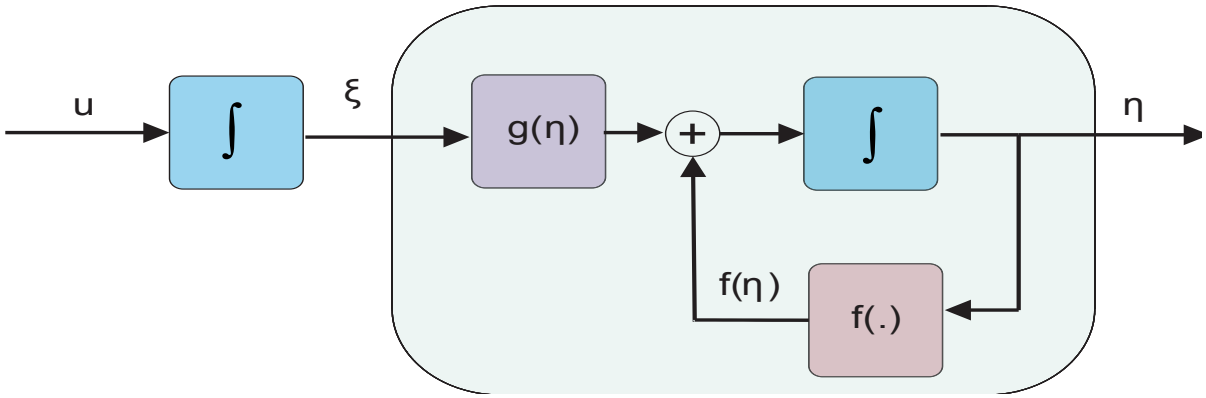

Fig. 3.11 Schema bloc a integratorului backstepping rezultată din sistemului (3.105)

Se dorește proiectarea unui controler în buclă închisă care să stabilizeze originea: $(\eta = 0, \xi = 0)$. Presupunem că funcțiile f și g sunt cunoscute. Acest sistem poate fi privit ca o conexiune în cascadă a 2 componente(Fig. 3.11). Prima componentă a sistemului, $\dot{\eta} = f(\eta) + g(\eta)\xi$, are ca intrare ξ, iar componenta $\dot{\xi} = u$ reprezintă integratorul. Dacă prima componentă se poate stabiliza cu $\xi = \varphi(\eta)$, $\varphi(0) = 0$, atunci originea componentei

$$\dot{\eta} = f(\eta) + g(\eta) \cdot \varphi(\eta) \tag{3.106}$$

este asimptotic stabilă.

Se consideră o funcție Lyapunov netedă și pozitiv definită $V(\eta)$ care satisface inegalitatea

$$\frac{\partial V(\eta)}{\partial \eta}[f(\eta) + g(\eta) \cdot \varphi(\eta)] \leq -W(\eta), \forall \eta \in D, \tag{3.107}$$

$W(\eta)$ este pozitiv definită.

Prin adunarea și scăderea lui $g(\eta)\varphi(\eta)$ în partea dreaptă a primei ecuații din sistemul (3.105) se obține reprezentarea echivalentă

$$\begin{cases} \dot{\eta} = [f(\eta) + g(\eta)\cdot\varphi(\eta)] + g(\eta)\cdot[\xi - \varphi(\eta)] \\ \dot{\xi} = u \end{cases}. \tag{3.108}$$

Această reprezentare este prezentată în Fig. 3.12.

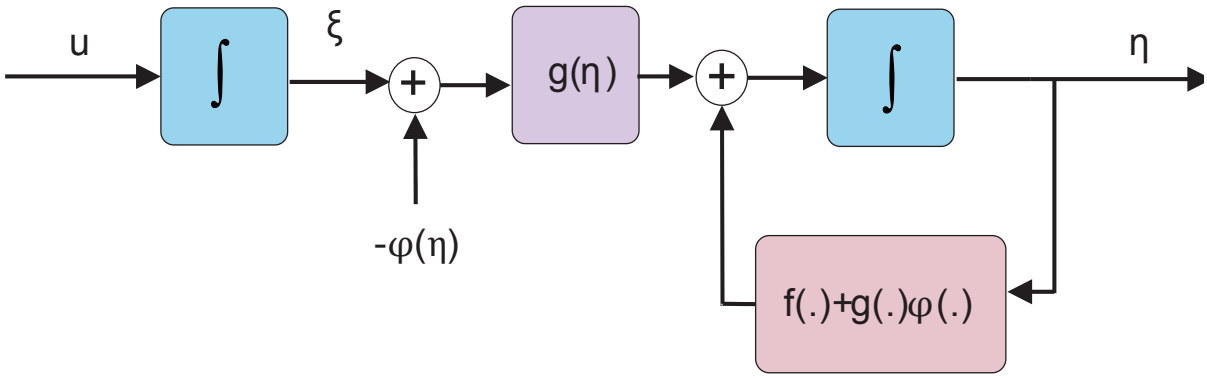

Fig. 3.12 Schema bloc a integratorului backstepping rezultată din sistemul (3.108)

Se efectuează schimbarea de variabilă

$$z = \xi - \varphi(\eta). \tag{3.109}$$

Întroducând (3.109) în (3.108) se obține sistemul

$$\begin{cases} \dot{\eta} = [f(\eta) + g(\eta)\cdot\varphi(\eta)] + g(\eta)z \\ \dot{z} = u \end{cases}. \tag{3.110}$$

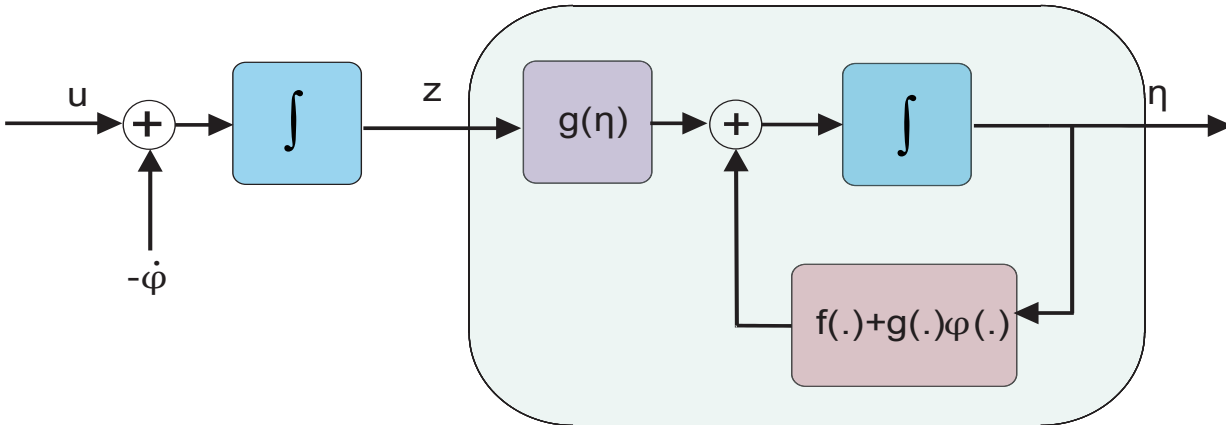

Fig. 3.13 Schema bloc a integratotului backstepping rezultată din sistemul (3.110)

Se calculează $\dot{\varphi}$ folosind formula

$$\dot{\varphi} = \frac{\partial\varphi}{\partial\eta}[f(\eta) + g(\eta)\xi]. \tag{3.112}$$

Se consideră v=u-$\dot{\varphi}$ sistemul se reduce la conexiunea în cascadă

$$\begin{cases} \dot{\eta} = [f(\eta) + g(\eta) \cdot \varphi(\eta)] + g(\eta)z \\ \dot{z} = v \end{cases}. \tag{3.113}$$

Sistemul (3.113) este similar sistemului (3.105), cu excepția faptului că prima componentă a noului sistem are o origine stabilă asimptotic când intrarea este 0. Această caracteristică este utilă în proiectarea lui v pentru a stabiliza întreg sistemul.

Alegem funcția Lyapunov

$$V_a(\eta, \xi) = V(\eta) + \frac{1}{2}z^2, \tag{3.114}$$

$$\dot{V}_a = \frac{\partial V(\eta)}{\partial \eta}[f(\eta) + g(\eta)\varphi(\eta)] + \frac{\partial V(\eta)}{\partial \eta}g(\eta)z + zv \leq W(\eta) + \frac{\partial V(\eta)}{\partial \eta}g(\eta)z + zv. \tag{3.115}$$

Alegând

$$v = -\frac{\partial V(\eta)}{\partial \eta}g(\eta)z + kz, \ k > 0, \tag{3.116}$$

Rezultă

$$\dot{V}_a \leq W(\eta) - kz^2. \tag{3.117}$$

Se observă că originea $(\eta = 0, \xi = 0)$ este asimptotic stabilă.

Legea de comandă se obține înlocuind v, z *și* $\dot{\varphi}$

$$u = \frac{\partial \varphi}{\partial \eta}[f(\eta) + g(\eta)\xi] - \frac{\partial V(\eta)}{\partial \eta}g(\eta) - k[\xi - \varphi(\eta)]. \tag{3.118}$$

Lema 1. Se consideră sistemul (3.113). Fie $\varphi(\eta)$ o stare stabilizatoare a sistemului în buclă închisă cu $\varphi(0) = 0$ și $V(\eta)$ o funcție Lyapunov care satisface (3.117) și o funcție pozitiv definită $W(\eta)$. Atunci comanda după stare în buclă închisă stabilizează originea lui (3.105), cu $V(\eta) + \frac{1}{2}[\xi - \varphi(\eta)]^2$ ca funcție Lyapunov. Mai mult dacă toate presupunerile sunt valide și $V(\eta)$ este nemărginită, originea va fi global asimptotic stabilă.

Se consideră un sistem mai general decât integratorul backstepping

$$\begin{cases} \dot{\eta} = f(\eta) + g(\eta)\xi \\ \dot{\xi} = f_\alpha(x, \xi) + g_\alpha(x, \xi)u \end{cases}. \tag{3.119}$$

f_α și g_α sunt funcții netede. Dacă $g_\alpha(x, \xi) \neq 0$ în domeniul de interes atunci transformata intrării

$$u = \frac{1}{g_\alpha(x, \xi)}[u_\alpha - f_\alpha(x, \xi)], \tag{3.120}$$

reduce $\dot{\xi}$,în (3.119), la forma integratorului $\dot{\xi} = u_\alpha$. Dacă există o comandă $\varphi(\eta)$ stabilizatoare și o funcție Lyapunov care satisface condițiile Lemei 1, atunci lema împreună cu (3.119) determină $u = \varphi_0(\eta, \xi)$ astfel

$$u = \frac{1}{g_\alpha(x, \xi)}\{\frac{\partial \varphi}{\partial \eta}[f(\eta) + g(\eta)\xi] - \frac{\partial V(\eta)}{\partial \eta}g(\eta) - k[\xi - \varphi(\eta)] - f_\alpha(x, \xi), \tag{3.121}$$

pentru k>0 și

$$V_a(\eta,\xi) = V(\eta) + \frac{1}{2}[\xi - \varphi(\eta)]^2 \tag{3.122}$$

ca funcție Lyapunov pentru sistemul (3.119).

Aplicând recursiv tehnica integratorului backstepping se pot stabiliza sisteme de forma

$$\begin{cases} \dot{x} = f(x) + g(x)\xi_1 \\ \dot{\xi}_1 = f_1(x,\xi_1) + g_1(x,\xi_1)\xi_2 \\ \dot{\xi}_2 = f_2(x,\xi_1,\xi_2) + g_1(x,\xi_1,\xi_2)\xi_3 \\ \vdots \\ \dot{\xi}_{k-1} = f_{k-1}(x,\xi_1,\cdots,\xi_{k-1}) + g_1(x,\xi_1,\cdots,\xi_{k-1})\xi_{k-1} \\ \dot{\xi}_k = f_k(x,\xi_1,\cdots,\xi_k) + g_1(x,\xi_1,\cdots,\xi_k)\xi_k \end{cases}, \tag{3.123}$$

în care: $x \in R^n$ și $\xi_1,\cdots,\xi_k \in R$, funcțiile $f, f_1,\cdots,f_k$ dispar la origine.

Procedura recursiva începe cu ecuația

$$\dot{x} = f(x) + g(x)\xi_1. \tag{3.124}$$

Se determină o funcție stabilizatoare $\xi_1 = \alpha(x)$, cu $\alpha(0) = 0$, pentru (3.105), construind o funcție Lyapunov pozitiv definită, nemărginită radial, $V(x)$ astfel încât cu această lege de conducere, derivata în timp

$$\frac{\partial V}{\partial x}[f(x) + g(x)\alpha(x)] \leq -W(x), \tag{3.125}$$

unde $W(x)$ este pozitiv definită.

Pasul 1. Se consideră subsistemul

$$\begin{cases} \dot{x} = f(x) + g(x)\xi_1 \\ \dot{\xi}_1 = f_1(x,\xi_1) + g_1(x,\xi_1)\xi_2 \end{cases}. \tag{3.126}$$

Se presupune că $\xi_1$este o comandă virtuală și legea de comandă $\xi_1 = \alpha(x)$, $\alpha(0) = 0$ stabilizează subsistemul. Considerăm sistemul (3.126) un caz special al sistemului (3.119) în care:

$$\eta = x;$$
$$\xi = \xi_1;$$
$$u = \xi_2;$$
$$f_\alpha = f_1;$$
$$g_\alpha = g_1.$$

Comanda stabilizatoare și funcția Lyapunov se obține astfel

$$\alpha_1(x,\xi_1) = \frac{1}{g_1}[\frac{\partial\alpha}{\partial x}[f + g\xi_1] - \frac{\partial V}{\partial x}g - k_1[\xi_1 - \alpha] - f_1], \tag{3.127}$$

pentru $k_1 > 0$

$$V_1(x,\xi_1) = V(x) + \frac{1}{2}[\xi_1 - \alpha_1(x)]^2. \quad (3.128)$$

Pasul 2. Se consideră sistemul

$$\begin{cases} \dot{x} = f(x) + g(x)\xi_1 \\ \dot{\xi}_1 = f_1(x,\xi_1) + g_1(x,\xi_1)\xi_2 \\ \dot{\xi}_2 = f_2(x,\xi_1,\xi_2) + g_1(x,\xi_1,\xi_2)\xi_3 \end{cases}, \quad (3.129)$$

ca un caz special al sistemului (3.126), în care:

$$\eta = \begin{bmatrix} x \\ \xi_1 \end{bmatrix};$$

$$\xi = \xi_2;$$

$$u = \xi_3;$$

$$f = \begin{bmatrix} f + g\xi_1 \\ f_1 \end{bmatrix};$$

$$g = \begin{bmatrix} 0 \\ g_1 \end{bmatrix};$$

$$f_\alpha = f_2;$$

$$g_\alpha = g_2.$$

Comanda stabilizatoare și funcția Lyapunov se obține astfel

$$\alpha_2(x,\xi_1,\xi_2) = \frac{1}{g_2}[\frac{\partial\alpha_1}{\partial x}(f + g\xi_1) + \frac{\partial\alpha_1}{\partial\xi_1}(f_1 + g_1\xi_2) - \frac{\partial V_1}{\partial\xi_1}g_1 - k_2[\xi_2 - \alpha_1] - f_1], \quad (3.130)$$

pentru $k_2 > 0$

$$V_2(x,\xi_1,\xi_2) = V_1(x,\xi_1) + \frac{1}{2}[\xi_2 - \alpha_2(x)]^2. \quad (3.131)$$

Procesul recursiv se repetă până la pasul k pentru obținerea comenzii stabilizatoare generală $u = \alpha_k(x,\xi_1,\xi_2,\cdots,\xi_k)$ și funcția Lyapunov $V_k(x,\xi_1,\xi_2,\cdots,\xi_k)$.

3.4.1 Conducerea backstepping aWMR cu 2DW/2FW

Arhitectura de conducere backstepping a roboților mobili cu 2 roți motoare și 2 libere este prezentată în Fig. 3.14.

Pentru a calcula comanda în cazul roboților mobili pentru problema urmăririi globale din [52]. considerăm modelul cinematic al robotului (3.10), modelul cinematic al robotului virtual (3.47) erorile de urmărire (3.48) și dinamica erorilor de urmărire(3.49).

Conducerea backstepping presupune calculul legilor de conducere ale vitezelor liniare și unghiulare având forma:

$$v = v(x_e, y_e, \theta_e, v_d, \omega_d, \dot{v}_d, \dot{\omega}_d); \quad (3.132)$$

$$\omega = \omega(x_e, y_e, \theta_e, v_d, \omega_d, \dot{v}_d, \dot{\omega}_d); \quad (3.133)$$

astfel încât traiectoria în buclă închisă definită de de (4.24), (4.102),(4.103) să fie mărginită și să conveargă spre zero.

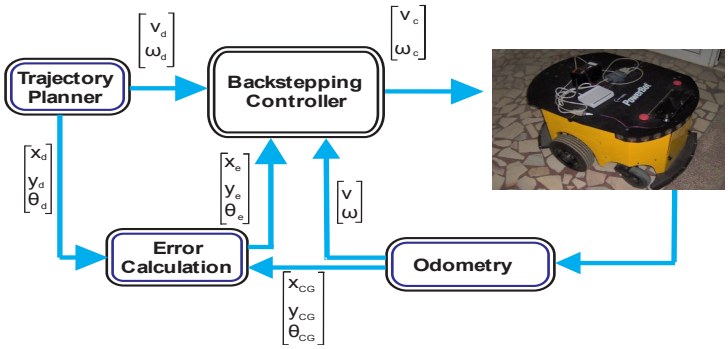

Fig. 3.14 Arhitectura de conducere backstepping a WMR cu 2DW/2FW

Integratorul backstepping se folosește pentru că y_e nu este controlat direct în(3.49). Funcțiile $x_e = k_1 \cdot \omega_d \cdot y_e$ și $\theta_e = 0$ sunt funcții stabilizatoare pentru y_e în (3.49).

Folosim schimbarea de variabilă

$$\bar{x}_e = x_e - k_1 \cdot \omega_d \cdot y_e, \tag{3.134}$$

unde $k_1 > 0$ este o constantă.

Utilizând (3.134) în (3.49) se obține ecuația

$$\dot{\bar{x}}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e - k_1 \cdot \dot{\omega}_d \cdot y_e - k_1 \cdot \omega_d \cdot (-\omega_d \cdot x_e + v_r \cdot \sin\theta_e). \tag{3.135}$$

Folosind schimbarea de variabilă

$$u = v_r \cdot \cos\theta_e + \omega_d \cdot y_e - k_1 \cdot \omega_d \cdot (-\omega_d \cdot x_e + v_r \cdot \sin\theta_e). \tag{3.136}$$

Rezultă comanda backstepping a roboților mobili:

$$v = u - y_e \cdot \omega_d + k_2 \cdot \bar{x}_e; \tag{3.137}$$

$$\omega = \omega_r + \lambda \cdot y_e \cdot v_r \int_0^1 \cos(s \cdot \theta_e) ds + k_3 \cdot \lambda \cdot \theta_e. \tag{3.138}$$

3.4.2 Conducerea backstepping a vehiculului autonom cu 4 DW/SW

Pentru a calcula comanda în cazul roboților mobili pentru problema urmăririi globale din [52]. Considerăm modelul cinematic al robotului (3.26), modelul cinematic al robotului virtual (3.56) erorile de urmărire (3.57) și dinamica erorilor de urmărire(3.58).

Conducerea backstepping presupune calculul legilor de conducere ale vitezelor liniare și unghiulare având forma:

$$v = v(x_e, y_e, \theta_e, v_d, \omega_d, \dot{v}_d, \dot{\omega}_d); \tag{3.139}$$

$$\delta = \delta(x_e, y_e, \theta_e, v_d, \omega_d, \dot{v}_d, \dot{\omega}_d); \tag{3.140}$$

astfel încât traiectoria în buclă închisă definită de (3.58), (3.139) și (3.140) să fie mărginită și să conveargă spre zero.

Integratorul backstepping se folosește pentru că y_e nu este controlat direct în(3.58). Funcțiile $x_e = k_1 \cdot \frac{v_d}{L} \cdot \tan\delta_d \cdot y_e$ și $\theta_e = 0$ sunt funcții stabilizatoare pentru y_e în (3.58).

Folosim schimbarea de variabilă

$$\bar{x}_e = x_e - k_{1.} \frac{v_d}{L} \cdot \tan\delta_d \cdot y_e, \tag{3.141}$$

unde $k_1 > 0$ este o constantă și notăm $\frac{v_d}{L} \cdot \tan\delta_d$ cu ω_d.

Utilizând (3.141) în (3.58) se obține ecuația

$$\dot{\bar{x}}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e - k_1 \cdot \dot{\omega}_d \cdot y_e - k_1 \cdot \omega_d \cdot (-\omega_d \cdot x_e + v_r \cdot \sin\theta_e). \tag{3.142}$$

Folosind schimbarea de variabilă

$$u = v_r \cdot \cos\theta_e + \omega_d \cdot y_e - k_1 \cdot \omega_d \cdot (-\omega_d \cdot x_e + v_r \cdot \sin\theta_e). \tag{3.143}$$

Rezultă comanda backstepping a roboților mobili

$$v = u - y_e \cdot \omega_d + k_2 \cdot \bar{x}_e; \tag{3.144}$$

$$\delta = a\tan(\frac{L}{v_r}(\omega_r + \lambda \cdot y_e \cdot v_r \int_0^1 \cos(s \cdot \theta_e) ds + k_3 \cdot \lambda \cdot \theta_e)). \tag{3.145}$$

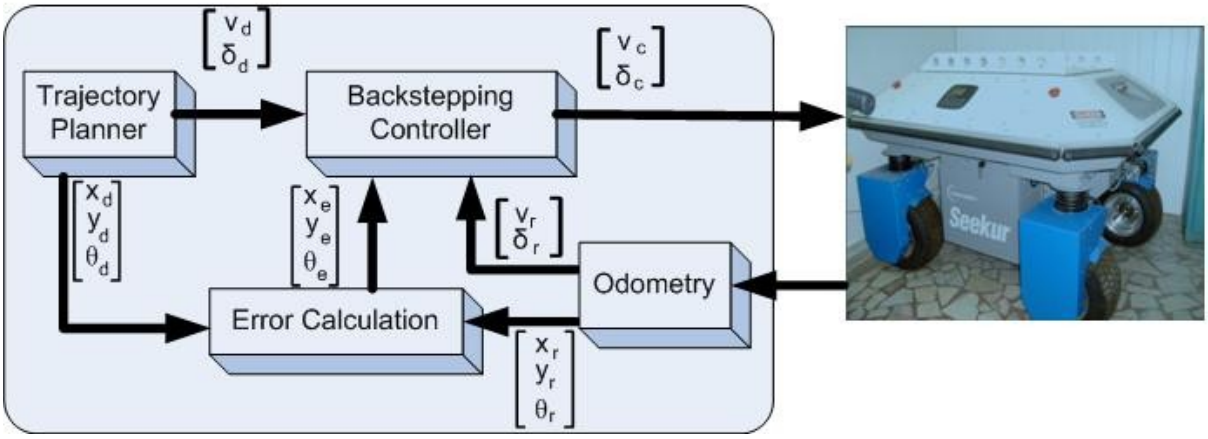

Fig. 3.15 Arhitectura de conducere backstepping a AV cu 4DW/SW

Arhitectura de conducera este prezentată în Fig. 3.15. Modulul Odometry din Fig. 3.15Fig. 3.10 primește datele de la encoderele robotului și calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în softul ARIA de la Mobile Robots și datele calculate pot fi obținute apelând în program funcțiile din ARIA.

3.5 Concluzii

În acest capitol am determinat modelul cinematic al WMR cu 2DW/2FW și modelul cinematic al vehiculului autonom electric SEEKUR (4DW/SW). Modelele cinematice descriu mișcarea robotului sau a vehiculului și nu iau în calcul forțele care acționează asupra lor și sunt folosite la calculul comenzii pentru conducerea WMR și AV.

Pentru determinarea modelului cinematic al WMR am considerat variabilele generalizate ale sistemului, rotirea roților robotului fără alunecare și aplicat constrângerile nonholomice specifice acestui caz. Modelul rezultat astfel are cinci variabile: 2 variabile reprezintă centrul geometric al WMR, o variabilă reprezentând unghiul de direcție al robotului și două variabile reprezentând unghiurile fiecărei roți. Aceste variabile depind de vitezele de rotire ale roților. Am simplificat acest model pentru că am dorit doar calculul coordonatelor carteziene ale centrului geometric și unghiul de direcție, înlocuind vitezele celor două roți cu viteza liniară și viteze unghiulară a robotului.

Pentru a determina constângerile nonholonomice ale vehiculului autonom SEEKUR am pornit de la cazul în care există alunecare laterală. Astfel am obținut constrângeri nonholonomice pentru centrul de greutate al vehiculului și pentru fiecare roată de la AV.

Am considerat doar roțile 1 și 2 modelul cinematic al vehiculului se simplifică la cazul modelului cinematic al bicicletei. Am considerat doar mișcarea Zero-side-slip, am obținut unghiul de alunecare 0 și am obținut modelul cinematic final folosit pentru conducerea vehiculului autonom.

Am propus trei metode pentru rezolvarea problemei conducerii roboților mobili și vehiculelor autonome: conducerea sliding-mode în timp continuu, conducerea sliding-mode în timp discret și conducerea backstepping.

Am prezentat cazul genaral pentru sinteza comenzii sliding-mode în timp continuu. De la cazul general și folosind modelul cinematic al WMR cu 2 DW/2SW , modelul erorilor de urmărire și dinamica erorilor de urmărire am calculat comenzile pentru viteza liniară și viteza unghiulară a WMR. În calcul am folosit suprafețe de comutație și legi de conducere bazate pe modelul propus de Gao pentru timp continuu în [31].

Utilizând cazul general și modelul cinematic simplificat al vehiculului autonom SEEKUR, suprafețe de comutație, legi de conducere similare celor propuse de Gao pentru timp continuu în [31], modelul erorilor de urmărire și dinamica erorilor de urmărire am calculat comenzile pentru viteza liniară și unghiul de direcție.

Am sintetizat modul de calcul al comenzii utilizând discrete-time sliding-mode în cazul general.Utilizând cazul general am trecut la calculul comenzilor pentru WMR și AV.

Pornind de la modelul cinematic al WMR în timp discret, erorile de urmărire în timp discret, dinamica erorilor de urmărire în timp discret, suprafețele sliding pentru timp discret și legea de conducere pentru timp discret am calculat comenzile pentru viteza liniară și viteza unghiulară a WMR.

Utilizând modelul cinematic simplificat al vehiculului autonomon în timp discret, erorile de urmărire în timp discret , dinamica erorilor de urmărire în timp discret, suprafețele sliding pentru timp discret și legea de conducere pentru timp discret am calculat comenzile pentru viteza liniară și unghiul de direcție a vehiculului autonom.

Am prezentat cazul general al conducerii backstepping. În continuare am calculat comanda pentru conducerea backstepping în cazul WMR utilizând modelul cinematic al robotului și dinamica erorilor de urmărire a robotului.

Am calculat comenzile pentru viteza liniară și unghiul de direcție al vehiculului autonom SEEKUR utilizând modelul cinematic și dinamica de urmărire a erorilor și procedura de conducere backstepping.

Capitolul 4.

4. Algoritmi de conducere a vehiculelor autonome și a roboților mobili, destinați evitării obstacolelor

În acest capitol am propus o soluție pentru evitarea obstacolelor de către roboții mobili și vehiculele autonome. Am consideră că roboții mobili sunt echipați cu sonare, iar vehiculele autonome sunt echipate cu senzori laser pentru depistarea obstacolelor și se propun două soluții în funcție de senzorii utilizați.

4.1 Algoritmi de evitare a obstacolelor implementați în conducerea roboților mobili

Am considerat situația în care se dorește urmărirea unei traiectorii dorite cu abateri minore în cazul apariției unui obstacol și revenirea la traiectoria impusă imediat dupa ce obstacolul a fost depășit. În literatură evitarea obstacolelor se face fără a fi impusă o traiectorie strictă, se preferă folosirea unor metode pentru a îndrepta robotul spre destinația dorită. Astfel de algoritmi au fost prezentați în introducere.

Obstacolele sunt detectate cu ajutorul a 14 sonare dispuse frontal și numerotate de la 0 la 14. Sonarul cu numărul 7 nu a fost folosit.Sonarele sunt dispuse circular ca în Fig. 4.1. Sonarele acoperă o zonă de 180 de grade permițând detectarea oricărui obstacol care ar putea bloca navigarea. Pentru detectarea obstacolelor se folosește bula de sensibilitate prezentată în [97]. Această bulă este proporțională cu viteza de deplasare și permite o bună detecție a obstacolelor periculoase. Bula de sensibilitate este prezentată în Fig. 4.2. Metoda constă în stabilirea unor coeficienți pentru fiecare sonar și calcularea graniței de siguranță prin multiplicarea coeficienților cu viteza și timpul de eșantionare. Bula de sensibilitate ($Boundry = [b_1\ b_2 \ldots b_{14}]$) se calculează utilizând formula

$$b_i = k_i \cdot v \cdot T_s, \tag{4.1}$$

unde - b_i- reprezintă componenta bulei de sensibilitate corespunzătoare sonarului i;

k_i- reprezintă coeficientul de siguranță a bulei de sensibilitate;

v- reprezintă viteza robotului;

T_s- reprezintă constanta de eșantionare.

După calcularea bulei de sensibilitate se verifică dacă un obstacol a pătruns în zona sensibilă. Un obstacol este detectat dacă distanța măsurată de sonar este mai mică decât valoarea componentei bulei de sensibilitate corespunzătoare sonarului respectiv. Cât timp nu

sunt detectate obstacole în bula de sensibilitate traiectoria globală calculată off-line este impusă modulului de conducere și robotul urmărește această traiectorie. Traiectoria este calculată off-line cu ajutorul unui genrerator de traiectorii care returnează viteza liniară, viteza unghiulară, accelerația unghiulară și accelerația unghiulară, necesare regulatoarelor din capitolul 3 pentru a conduce robotul.

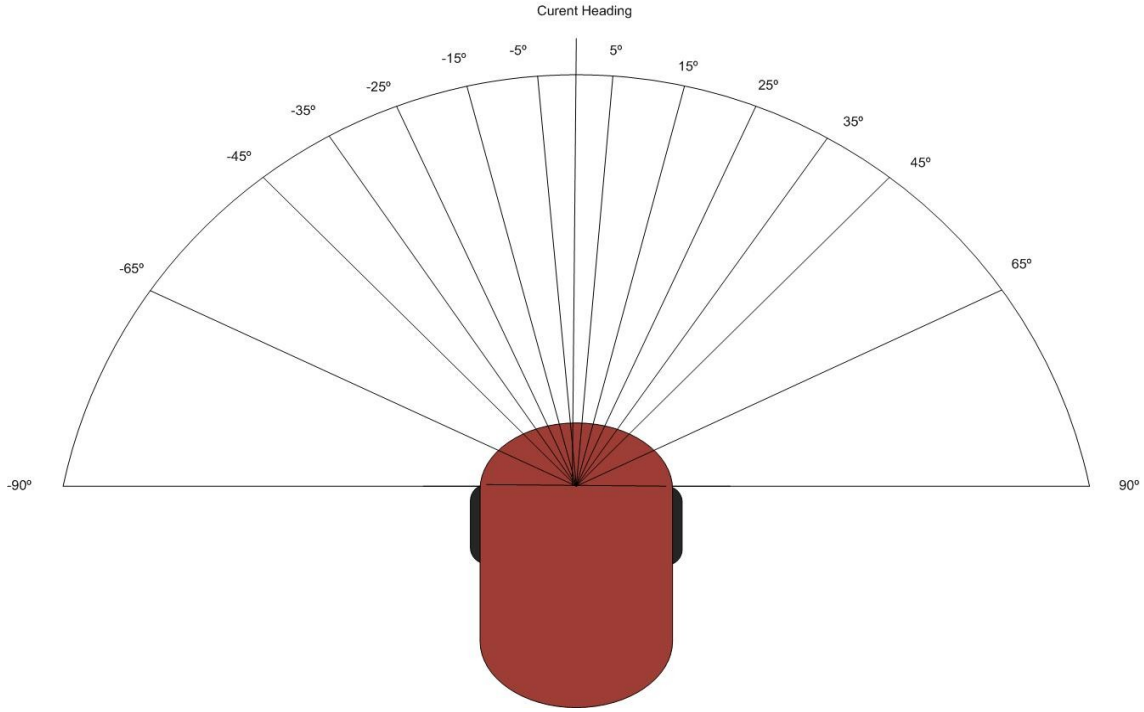

Fig. 4.1 Dispunerea sonarelor la PowerBot

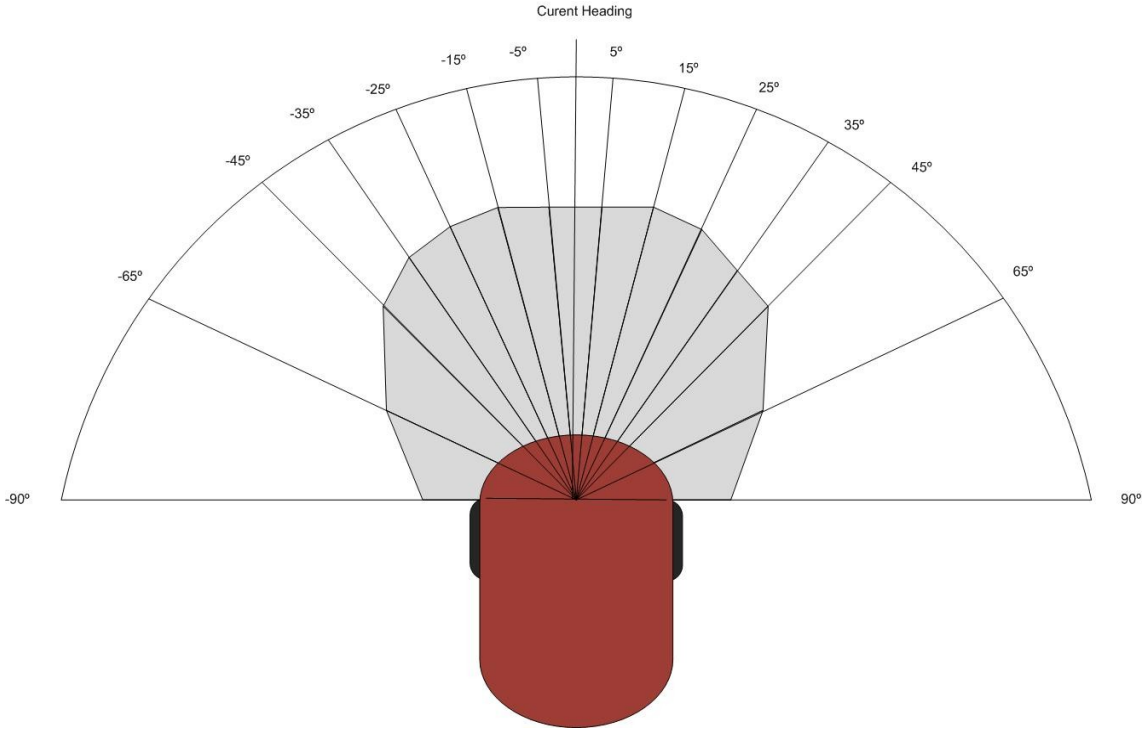

Fig. 4.2 Bula de sensibilitate la PowerBot

Dacă un obstacol a fost detectat în zona de sensibilitate modulului de conducere îi este impusă o traiectorie locală generată pentru evitarea obstacolului. În Fig. 4.3 este prezentat un obstacol care este detectat de sonarul cu numărul 6.

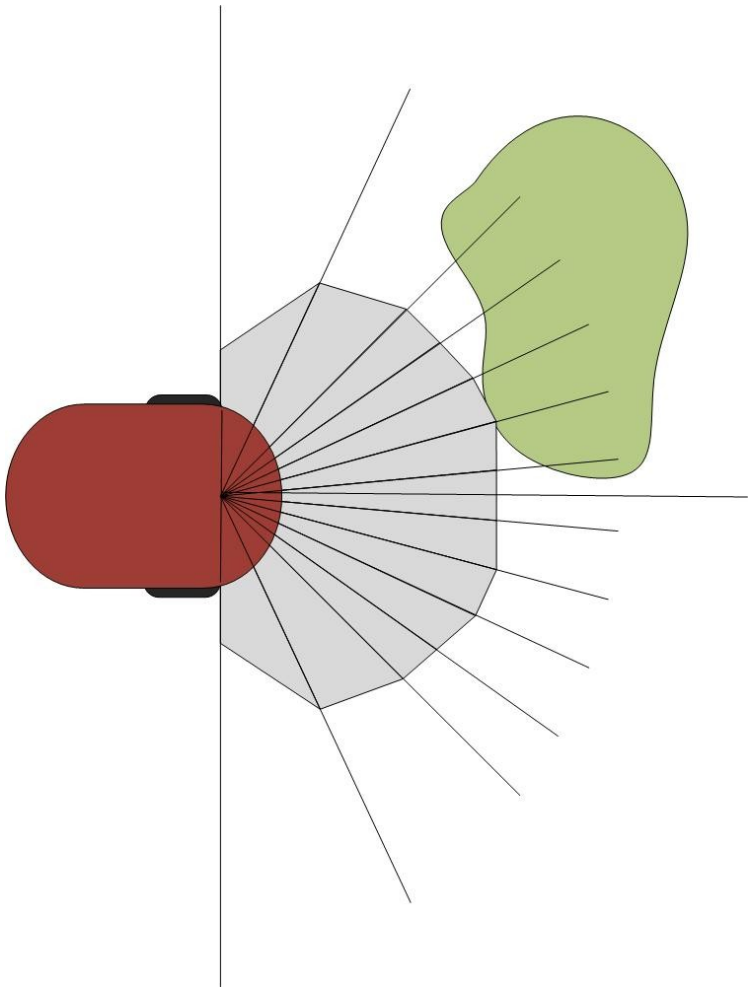

Fig. 4.3 Exemplu de obstacol detectat

Traiectoria locală urmărește deplasarea robotului la un punct aflat perpendiculara la direcția de mers dusă din punctul în care este detectat cel mai pe apropiat obstacol. Pentru determinarea acestui punct se caută o zonă liberă suficient de mare pe perpendiculara la direcția de mers dusă din punctul în care este detectat cel mai pe apropiat obstacol pentru a permite trecerea robotului. Dacă distanța minimă este măsurată de sonarele aflate în partea stângă(0-6) atunci punctul dorit se caută întâi în partea dreaptă și dacă nu se găsește un astfel de punct se caută în partea stângă. Dacă distanța minimă este măsurată de sonarele aflate în partea dreaptă(8-14) atunci punctul dorit se caută întâi în partea stângă și dacă nu se găsește un astfel de punct se caută în partea dreaptă.

Algorimul este următorul: se determină sonarul care a depistat un obstacol în zona de sensibilitate și se stie că sonarele cu numărul mai mic decât 7 sunt în partea stângă, iar cele

mai mari decât 7 sunt în partea dreaptă. În funcție de partea pe care se află obstacolul se încearcă căutarea în partea opusă a unei zone libere suficient de mari pentru a permite trecerea robotului. În Fig. 4.3 se observă că cel mai ușor se poate evita obstacolul cu o ocolire la dreapta.

În Fig. 4.4 se observă că traiectoria dorită(linia neagră) este blocată și se dorește găsirea unui punct cu proprietatea că robotul poate ocoli obstacolul dacă centrul său de greutate se deplasează spre acel punct. Este cunoscută lățimea robotului care este considerată distanța minimă(hmin) pentru ca robotul să poată trece și începânând cu sonarul 8, și incrementând indicele sonarului ales se verifică, care este primul sonar care nu detectează un obstacol apropiat în cazul ocolirii prin partea dreaptă. În cazul ocolirii prin partea stângă se pornește de la sonarul 6 și se scade indicele sonarului până se găsește primul sonar care nu depistează un obstacol. Se consideră zona de la intersecția direcției de înaintare cu perpendiculara dusă din punctul în care s-a detectat cel mai apropiat obstacol până la punctul determinat de această perpendiculară și unghiul sonarului ales o zonă de excludere.

Se consideră că un obstacol nu depistează un obstacol dacă:

$$range[k] \cdot \cos(alfa[k]) > \min\_range + c \tag{4.2}$$

Unde $range[k]$ -reprezintă distanța măsurată de sonarul k,

$alfa[k]$-reprezintă unghiul sonarului k

min_range- reprezintă proiecția distanței până la cel mai apropiat obstacol pe direcția de înaintare,

c- reprezintă o constanta pentru validarea absenței obstacolelor

Se calculează proiecția distanței până la cel mai apropiat obstacol(min_range) și pe baza unghiului sonarului selectat se calculează zona de excludere:

$$Ze = |\min\_range * \tan(angle)| \tag{4.3}$$

Urmează determinarea zonei în care poate naviga robotul astfel: se crește(sau scade în funcție de direcția de căutare) numărul sonarului analizat și se verifică dacă zona delimitată de sonarul ales, min_range și sonarul curent este suficient de mare pentru a trece robotul. În caz afirmativ se alege destinația punctul aflat la jumătatea distanței. Dacă nu se găsește un loc de ocolire se încearcă schimbarea direcției de ocolire.

Zona delimitată pe perpendiculara la direcția de mers aleasă are proprietatea

$$h = \min\_range \cdot \tan(alfa(j)) - Ze > h\min, \tag{4.4}$$

unde min_range- reprezintă proiecția pe direcția de înaintare a distanței până la obstacol, alfa(j)- reprezintă unghiul sonarului j, Ze – reprezintă zona de excludere(4.3).

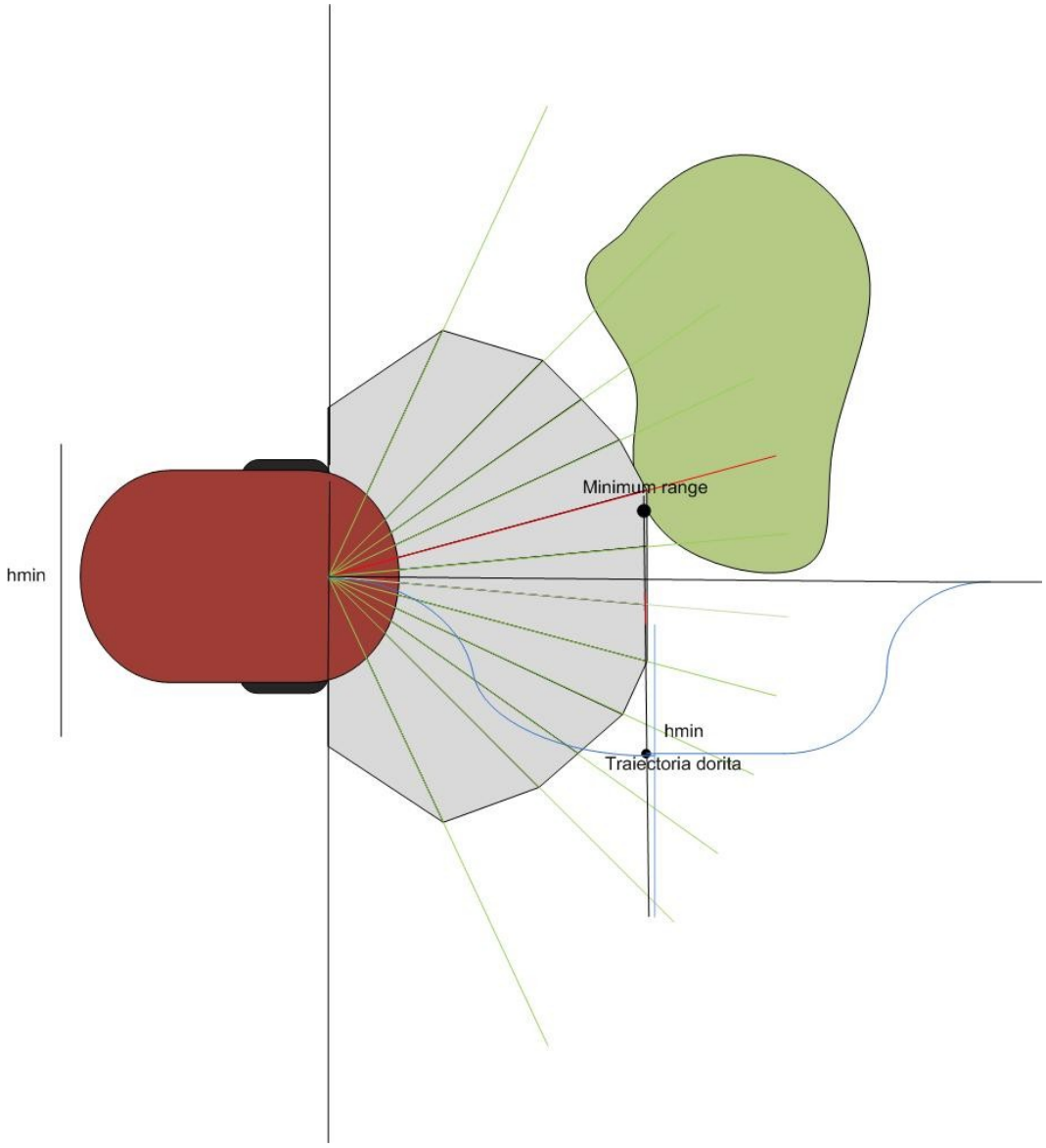

Fig. 4.4 Traiectoria dorita pentru evitarea obstacolului

După determinarea zonei h prin care este posibilă evitarea obstacolului se determină punctul(new_x,new_y) în care se dorește deplasarea centrului de greutate al robotului

$$\begin{cases} new\_x = x + \min\_range \cdot \cos(Th) - H \cdot \sin(Th) \\ new\_y = y - \min\_range \cdot \cos(Th) - H \cdot \sin(Th) \end{cases}, \quad (4.5)$$

unde: (x,y) reprezintă coordonatele curente ale robotului;

H- reprezintă distanța de la direcția de înaintare la mijlocului segmentului delimitat de h;

Th- reprezintă orientarea robotului.

Algoritmul pentru selectarea direcției la dreapta este următorul:

j=8

Dacă $range[j] \cdot \cos(alfa[j]) \geq \min\_range + 0.1 \parallel range[j] = \max\_range$

Atunci

$Ze = |\min\_range * \tan(alfa[j])|$

altfel

Cât timp $range[j] \cdot \cos(alfa[j]) < \min\_range + 0.1 \&\& j < 14$

$j = j + 1;$

$Ze = |\min\_range * \tan(alfa[j])|$

$h = 0$

Cât timp $(h < h\min) \&\&(j < 14) \&\&(range[j] > \min\_range)$

$\{ \; j = j + 1;$

$h = |\min\_range * \tan(alfa[j])| - Ze \}$

Dacă $h >= h\min$

$H = \frac{h}{2} + Ze$

$new\_x = x + \min\_range \cdot \cos(Th) - H \cdot \sin(Th)$

$new\_y = y - \min\_range \cdot \cos(Th) - H \cdot \sin(Th)$

Gasit=1

Altfel Gasit=0

După selecția noii destinații trebuie calculată o nouă traiectorie care să permită deplasarea robotului de la punctul curent la punctul (new_x,new_y). Traiectoria este generată folosind Quintic equations din[96] pentru a obține traseul dorit apoi se obține traiectoria dorită calculând profilul vitezelor pentru că robotul poate urmări noua traiectorie utilizând unul din regulatoarele prezentate în capitolul 3. Se consideră punctul curent(xc,yc) și unghiul curent(θc) și se dorește calcularea punctelor intermediare care le unesc de punctul calculat și $unghi\_final = unghi\_curent + pi/4$ reprezentând orientarea finală a robotului, se determină punctele intermediare care descriu traseul pentru evitarea obstacolului($P(u) = \{p_1(u), p_2(u), \cdots, p_n(u)\}, u \in [0,1]$). Fiecare punct p_i este compus din coordonatele x_i,y_i și θ_i.

O curbă bazată pe quintic equations se calculează astfel[96]:

$$p_{i,i+1}(u) = \begin{bmatrix} x_{i,i+1}(u) \\ y_{i,i+1}(u) \\ \phi_{i,i+1}(u) \end{bmatrix} = \begin{bmatrix} \alpha_{i0} + \alpha_{i1} \cdot u + \alpha_{i2} \cdot u^2 + \alpha_{i3} \cdot u^3 + \alpha_{i4} \cdot u^4 + \alpha_{i5} \cdot u^5 \\ \beta_{i0} + \beta_{i1} \cdot u + \beta_{i2} \cdot u^2 + \beta_{i3} \cdot u^3 + \beta_{i4} \cdot u^4 + \beta_{i5} \cdot u^5 \\ \phi_i(u) \end{bmatrix}, \quad (4.6)$$

unde:

$$\alpha_{i0} = x_i(0);$$

$$\alpha_{i1} = g_1 \cdot \cos(\theta_i(0));$$

$$\alpha_{i2} = \frac{1}{2}\left(g_3 \cdot \cos(\theta_i(0)) - g_1^2 \cdot k_i \cdot \sin(\theta_i(0))\right);$$

$$\alpha_{i3} = 10 \cdot (x_{i+1} - x_i) - \left(6 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\cos(\theta(0)) - \left(4 \cdot g_2 - \frac{1}{2} g_4\right)\cos(\theta_{i+1}(0)) +$$
$$+ \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) - \frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \quad ;$$

$$\alpha_{i4} = -15 \cdot (x_{i+1} - x_i) + \left(8 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\cos(\theta(0)) + (7 \cdot g_2 - g_4)\cos(\theta_{i+1}(0)) -$$
$$- \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) + g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \quad ;$$

$$\alpha_{i5} = 6 \cdot (x_{i+1} - x_i) - \left(3 \cdot g_1 + \frac{1}{2} \cdot g_3\right)\cos(\theta(0)) - \left(3 \cdot g_2 - \frac{1}{2} g_4\right)\cos(\theta_{i+1}(0)) +$$
$$+ \frac{1}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) - \frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \quad ;$$

$$\beta_{i0} = y_i(0);$$

$$\beta_{i2} = \frac{1}{2}\left(g_3 \cdot \sin(\theta_i(0)) - g_1^2 \cdot k_i \cdot \cos(\theta_i(0))\right);$$

$$\beta_{i3} = 10 \cdot (y_{i+1} - y_i) - \left(6 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\sin(\theta(0)) - \left(4 \cdot g_2 - \frac{1}{2} g_4\right)\sin(\theta_{i+1}(0)) +$$
$$+ \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) - \frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \quad ;$$

$$\beta_{i4} = -15 \cdot (y_{i+1} - y_i) + \left(8 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\sin(\theta(0)) + (7 \cdot g_2 - g_4)\sin(\theta_{i+1}(0)) -$$
$$- \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) + g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \quad ;$$

$$\beta_{i5} = 6 \cdot (y_{i+1} - y_i) - \left(3 \cdot g_1 + \frac{1}{2} \cdot g_3\right)\sin(\theta(0)) - \left(3 \cdot g_2 - \frac{1}{2} g_4\right)\sin(\theta_{i+1}(0)) +$$
$$+ \frac{1}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) - \frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \quad ,$$

parametrii k_i, k_{i+1} reprezintă curbatura scalară și pot primi valori aleatoare, în acest caz se consideră valoarea 0.Parametrii q_1, q_2, g_3, g_4 influențează forma curbei. În acest caz se consideră valorile: $g_1 = g_2 = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$, $g_3 = g_4 = 0$.

Controllerul sliding-mode primește la intrare viteza, accelerația , viteza unghiulară și accelerația unghiulară, iar utilizând Quintic equations am obținut punctele noului traseu și nu traiectoria reprezentată de profilul de viteză. Pentru a utiliza același regulator trebuie calculate vitezele, accelerațiile, vitezele unghiulare și accelerațiile unghiulare care permit robotului să urmărească traseul generat mai sus..

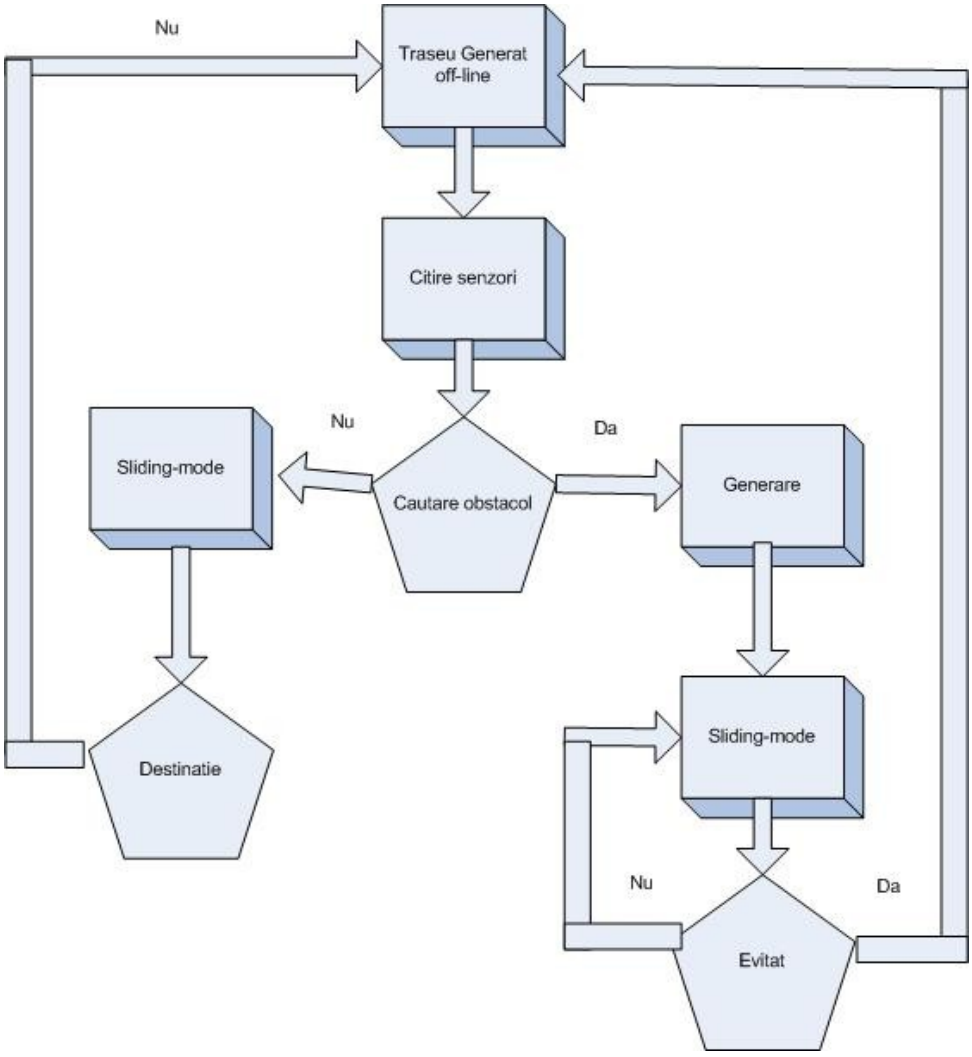

Fig. 4.5 Schema bloc a algoritmului de conducere cu evitare de obstacole

Pentru fiecare punct al traseului calculăm lungimea segmentului până la punctul precedent. Obținem viteza împărțind această lungime la durata de timp în care se dorește parcurgerea segmentului și accelerația împărțind viteza la durata de timp.

Viteza unghiulară se obține astfel

$$\omega[n] = a\tan 2(y[n] - y[n-1], x[n] - x[n-1]) / Ts$$

În final obținem accelerația unghiulară împărțind viteza unghiulară la durata de timp.

Algoritmul pentru calculul acestor parametri este:

De la 1 la 100 repetă {

$l[n] = \sqrt{(x[n]-x[n-1])^2 + (y[n]-y[n-1])^2}$ Calculul lungimei segmentului

$v[n] = l[n]/Ts$ Calculul vitezei

$a[n] = v[n]/Ts$ Calculul accelerației

$\omega[n] = a\tan 2(y[n]-y[n-1], x[n]-x[n-1])/Ts$ Calculul vitezei unghiulare

$a\omega[n] = \omega[n]/Ts$ Calculul accelerației

}.

Se parcurge traiectoria nou generată prin impunerea acestei traiectorii la modulul de conducere. Se continuă deplasarea cu viteză constantă în ultima direcție până când obstacolul este evitat. Se estimează pozițiile în care s-ar fi aflat robotul dacă urma traiectoria inițială și apoi se calculează traiectoria de revenire la traiectoria inițială în locul în cel mai apropiat de poziția curentă a robotului și pozițiile estimate utilizând același algoritm folosit la determinarea traiectoriei locale, și se impune această traiectorie modulului de conducere, apoi se revine la traiectoria impusă începând cu punctul selectat anterior. Fig. 4.5 prezintă schema bloc a algoritmului prezentat pentru evitarea de obstacole.

4.2. Algoritmi de evitare a obstacolelor implementați în conducerea vehiculelor autonome

Se consideră un mediu cunoscut în care se dorește urmărirea unei traiectorii impuse de către un vehicul autonom. În acest mediu este posibilă apariția unor obstacole neașteptate care trebuie evitate. Se dorește evitarea acestor obstacole abandonând traiectoria impusă doar cât timp este blocată de un obstacol și revenirea la traiectoria impusă după ce obstacolul a fost evitat.

Obstacolele sunt detectate cu ajutorul unui laser SICK LMS111. Acest laser permite detectarea obstacolelor aflate într-un arc de la $-90°$ la $90°$ de centrul robotului. Laserul este programat sa ofere date cu o rezoluție de $1°$, rezultând un număr total de 181 distanțe măsurate. Pentru a reduce volumul de date procesate programul folosește o rezoluție de $5°$, obținută din media aritmetică a 5 distanțe consecutive. Această aproximare garantează că nici un obstacol nu va trece neobservat și că influența unor date eronate este limitată. În Fig. 4.6 este prezentată zona de detecție a laserului.

Micșorând rezoluția de la 1 grad la 5 grade obținem 37 distanțe. La aceste 37 distanțe se poate defini o bulă de sensibilitate[97],care acoperă 180 de grade și permite evitarea oricărui obstacol. Bula de sensibilitate definește o zonă de siguranță proporțională cu viteza

vehiculului, în care vehiculul se poate deplasa în siguranță. Calculul bulei de siguranță se realizează multiplicand coeficienții stabiliți pentru fiecare citire cu viteza vehiculului și perioada de eșantionare(4.1).

Procedura de evitare a obstacolelor începe dacă este detectat un obstacol în interiorul bulei de sensibilitate. În lipsa obstacolelor conducerea se realizează utilizând conducerea sliding-mode sau backstepping și traiectoria globală generată off-line.

Dacă un obstacol este detectat se calculează o traiectorie locală de ocolire care este impusă modulului de conducere până când obstacolul este evitat. Traiectoria este generată folosind Quintic equations din [96] pentru a obține traseul dorit apoi se obține traiectoria dorită calculând profilul vitezelor pentru că robotul poate urmări noua traiectorie utilizând unul din regulatoarele prezentate în capitolul 3. În Fig. 4.7 este prezentată traiectoria dorită pentru evitarea unui obstacol.

Traiectoria locală urmărește deplasarea robotului la un punct aflat perpendiculara la direcția de mers dusă din punctul în care este detectat cel mai pe apropiat obstacol. Pentru determinarea acestui punct se caută o zonă liberă suficient de mare pe perpendiculara la direcția de mers dusă din punctul în care este detectat cel mai pe apropiat obstacol pentru a permite trecerea robotului.

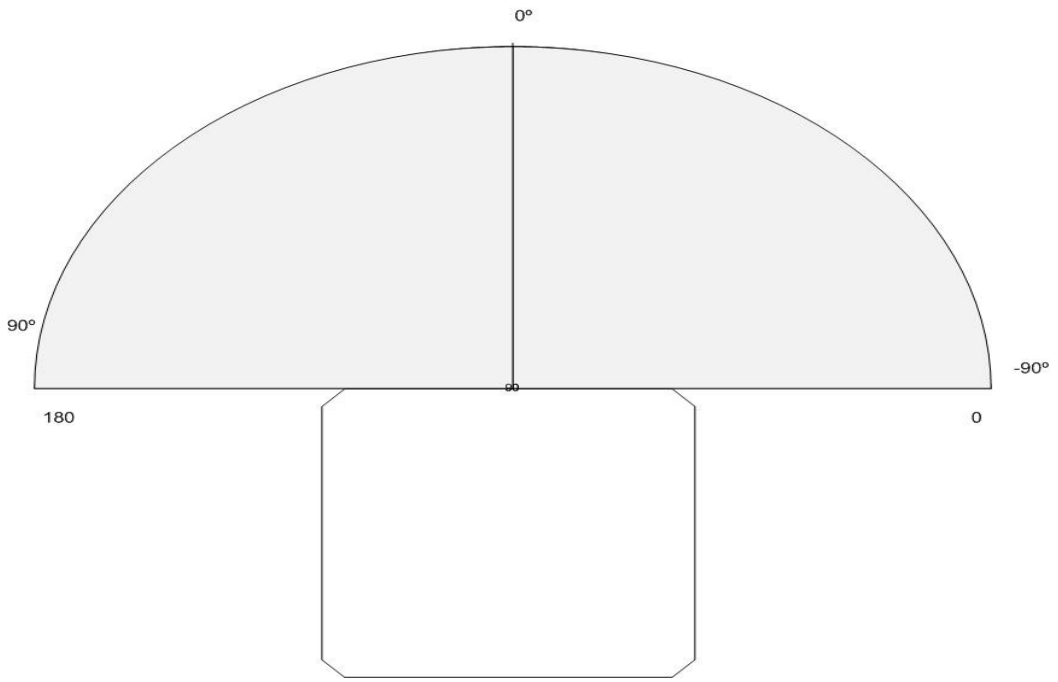

Fig. 4.6 Zona de detecție a laserului

Dacă distanța minimă este se află în partea stângă(i>17) atunci punctul dorit se caută întâi în partea dreaptă și dacă nu se găsește un astfel de punct se caută în partea stângă. Dacă

distanța minimă este măsurată se află în partea dreaptă(i<17) atunci punctul dorit se caută întâi în partea stângă și dacă nu se găsește un astfel de punct se caută în partea dreaptă.

Algorimul este următorul: se determină indexul la care s-a depistat un obstacol în zona de sensibilitate și se stie că indexurile mai mari decât 17 sunt în partea stângă, iar cele mai mici decât 17 sunt în partea dreaptă. În funcție de partea pe care se află obstacolul se încearcă căutarea în partea opusă a unei zone libere suficient de mari pentru a permite trecerea robotului.

Pentru a evita obstacolul trebuie găsită o zonă fără obstacole suficient de mare pentru a permite trecerea vehiculului. Se definesc 2 zone de evitare, aflate la stânga și dreapta direcției de deplasare. Citirile de la 0 la 17 se repartiteză în zona din dreapta, iar citirile de la 19 la 36 în zona din partea stângă. Se determină la care citire s-a detectat cel mai apropiat obstacol în zona de sensibilitate. Dacă obstacolul se află în partea stângă se încearcă ocolirea prin partea dreaptă, dacă nu se obținte o soluție se caută o soluție și în partea stângă. Dacă obstacolul se află în partea dreaptă se încearcă ocolirea prin partea stângă, dacă nu se obținte o soluție se caută o soluție și în partea dreaptă. Evitarea prin ocolire la dreapta este prezentată în Fig. 4.7. Se observă că traiectoria dorită(linia roșie) este blocată și se calculează traiectoria de ocolire(linia albastră). Punctul spre care se deplasează vehiculul pentru evitarea traiectoriei se calculează în funcție de lățimea robotului plus o marjă de siguranță.

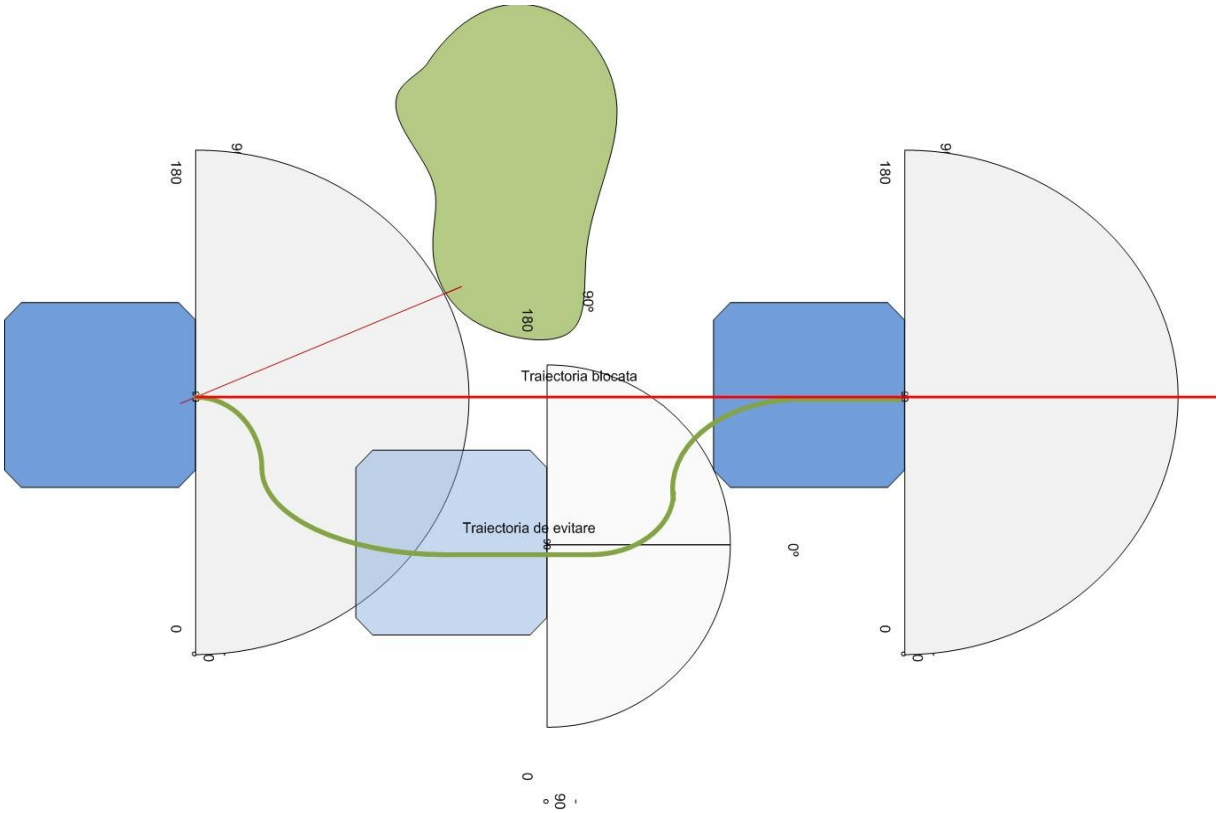

Fig. 4.7 Traiectoria dorită pentru evitarea obstacolului în cazul utilizării detecției cu ajutorul senzorilor laser

Se determină primul index a cărui distanță măsurată este mai mare decât distanța minimă. Se calculează proiecția distanței până la cel mai apropiat obstacol(min_range) și pe baza unghiului selectat se calculează zona de excludere folosind ecuația (4.4).

Condiția pentru selectarea primului index este

$$range[k] \cdot \cos(alfa[k]) > \min\_range + c, \tag{4.7}$$

Unde $range[k]$ este distanța masurată la indexul k, $alfa[k]$ unghiul la care se măsoară distanța.

Urmează determinarea zonei în care poate naviga vehiculul astfel: se crește(sau scade în funcție de direcția de căutare) indexul citirilor analizate și se verifică dacă zona delimitată de unghiul ales, min_range și unghiul curent este suficient de mare pentru a trece vehiculul aplicând condiția (4.4) În caz afirmativ se alege ca destinație punctul aflat la jumătatea distanței. Dacă nu se găsește un loc de ocolire se încearcă schimbarea direcției de ocolire.

Coordonatele acestui punct sunt date de (4.5).

Algoritmul pentru selectarea direcției la stânga este următorul:

j=18

Dacă

$range[j] * \cos(alfa[j]) \geq \min\_range + 0.1 \,||\, range[j] = \max\_range$

Atunci

$Ze = |\min\_range * \tan(alfa[j])|$

altfel

Cât timp $range[j] * \cos(alfa[j]) < \min\_range + 0.1 \,\&\&\, j < 36$

$j = j + 1;$

$Ze = |\min\_range * \tan(alfa[j])|$

$h = 0$

Cât timp $(h < h\min) \,\&\&\, (j < 36) \,\&\&\, (range[j] > \min\_range)$

$\{\ j = j + 1;$

$h = |\min\_range * \tan(alfa[j])| - Ze\ \}$

Dacă $h \geq h\min$

$$H = \frac{h}{2} + Ze$$

$new\_x = x + \min\_range \cdot \cos(Th) - H \cdot \sin(Th)$

$$new\_y = y - \min\_range \cdot \cos(Th) - H \cdot \sin(Th)$$

Gasit=1

Altfel Gasit=0

După selecția noii destinații trebuie calculată o nouă traiectorie care să ducă robotul de la punctul curent la punctul (new_x,new_y). Traiectoria va fi generată folosind Quintic equations (5.6). Se consideră punctul curent și unghiul curent și se dorește calcularea punctelor intermediare care le unesc de punctul calculat și $unghi\_final = unghi\_curent + pi/4$.

Controllerul sliding-mode primește la intrare viteza, accelerația , viteza unghiulară și accelerația unghiulară. Având punctele traseului calculăm vitezele, accelerațiile, vitezele unghiulare și accelerațiile unghiulare care descriu traiectorie necesară modulului de conducere.

Pentru fiecare punct al traseului calculăm lungimea segmentului până la punctul precedent. Obținem viteza împărțind această lungime la durata de timp în care se dorește parcurgerea segmentului și accelerația împărțind viteza la durata de timp.

Viteza unghiulară se obține astfel

$$\omega[n] = a\tan 2(y[n] - y[n-1], x[n] - x[n-1])/Ts.$$

În final obținem accelerația unghiulară împărțind viteza unghiulară la durata de timp.

Algoritmul pentru calculul acestor parametri este:

De la 1 la 100 repetă {

$l[n] = \sqrt{(x[n] - x[n-1])^2 + (y[n] - y[n-1])^2}$ Calculul lungimei segmentului

$v[n] = l[n]/Ts$ Calculul vitezei

$a[n] = v[n]/Ts$ Calculul accelerației

$\omega[n] = a\tan 2(y[n] - y[n-1], x[n] - x[n-1])/Ts$ Calculul vitezei unghiulare

$a\omega[n] = \omega[n]/Ts$ Calculul accelerației

}.

Se parcurge traiectoria nou generată prin impunerea acestei traiectorii la modulul de conducere. Se continuă deplasarea cu viteză constantă în ultima direcție până când obstacolul este evitat. Se estimează pozițiile în care s-ar fi aflat robotul dacă urma traiectoria inițială și

apoi se calculează traiectoria de revenire la traiectoria inițială în locul în cel mai apropiat de poziția curentă a robotului și pozițiile estimate utilizând același algoritm folosit la determinarea traiectoriei locale, și se impune această traiectorie modulului de conducere, apoi se revine la traiectoria impusă începând cu punctul selectat anterior. Fig. 4.5 prezintă schema bloc a algoritmului prezentat pentru evitarea de obstacole.

4.3. Concluzii

Am propus o soluție pentru problema evitării obstacolelor în cazul în care se dorește urmărirea unei traiectorii impuse și abaterea de la această traiectorie doar pentru a evita un obstacol și întoarcerea la traiectoria impusă. Am fost considerat cazul în care robotul este echipat cu sonare și cazul în care vehiculul este echipat cu senzori laser.

Detecția obstacolelor folosește o bulă de sensibilitate pentru a determina dacă un obstacol trebuie evitat. Am folosit aceași metodă de conducere în cazul în care nu există obstacole și în cazul în care un obstacol este prezent. Cât timp nu este detectat nici un obstacol care blochează traiectoria dorită platforma urmărește traiectoria impusă folosind metoda de conducere aleasă din cele 3 prezentate în capitolul anterior. Traiectoria este compusă din vitezele liniare, vitezele unghiulare, acceleratiile liniare și accelerațiile unghiulare.

Dacă un obstacol a fost detectat se caută coordonatele care să permită conducerea în siguranță a platformei și se generează o traiectorie bazată pe Quintic equations. Această traiectorie este impusă modulului de conducere. Se continuă deplasarea pe această traiectorie până când obstacolul a fost evitat. După ce obstacolul a fost evitat se calculează traiectoria pentru a reveni la traiectoria impusă și modul de conducere urmărește această traiectorie apoi se revine la traiectoria impusă inițial.

Capitolul 5.

5. Implementarea conducerii în timp real. Rezultate experimentale

Activitatea de cercetare s-a desfăşurat în Laboratorul de Robotică al Facultăţii de Automatică, Calculatoare, Inginerie Electrică şi Electronică din Universitatea „Dunărea de Jos” din Galaţi. Cercetarea s-a realizat pe roboţii mobili Pioneer P3-DX, PatrolBot, PowerBot şi vehiculul autonom Seekur de la Adept Mobile Robots.

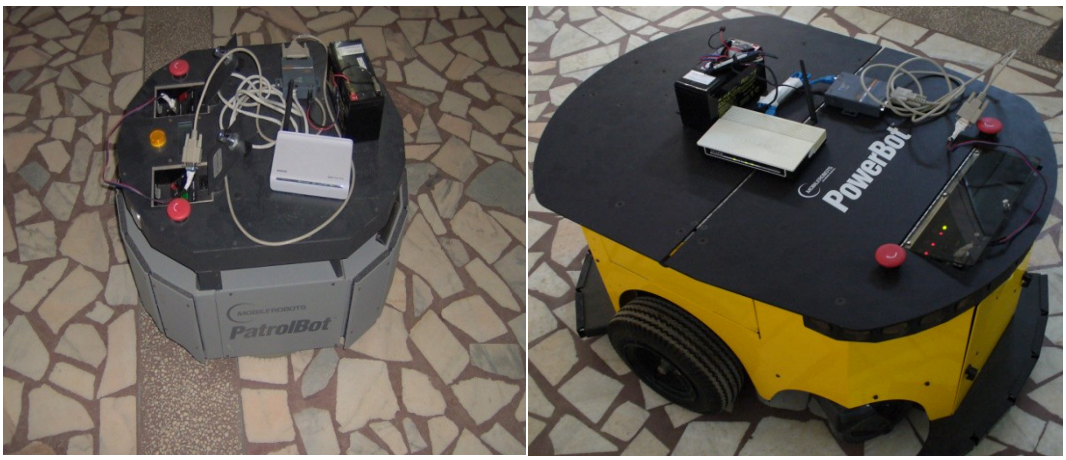

Fig. 5.1 Robotul mobil PatrolBot şi robotul mobil Powerbot

Fig. 5.2 Vehiculul autonom SEEKUR

Roboţii mobili Pioneer 3-DX, PatrolBot şi PowerBot sunt roboţi conduşi diferenţial(2DW/2FW) şi echipaţi cu sonare, în timp ce Seekur este un vehicul cu tracţiune integrală şi cu 4 roţi directoare(4DW/SW).

Robotul mobil Pioneer este echipat cu un braț articulat cu 7 grade de libertate și un gripper. Brațul robotului nu este prevăzut cu senzori, iar comanda brațului poate fi realizată doar în buclă deschisă.

Fiecare platformă robotică vine echipată cu motoare și controlere, toate controlate de un microcontroller încorporat care acționează ca un server și software client pentru roboții mobili. Dezvoltarea de software include Advanced Robotics Interface for Applications(ARIA) și ArNetworking, dezvoltate sub licență publică GNU, și completă cu librării complet documentate pentru C++, Java și Python și cod sursă.

Roboții sunt prevăzuți cu o conexiune serială RS232 pentru comunicația cu exteriorul, iar conectarea la calculator se realizează utilizând o conexiune wireless, utilizând un access point wireless și un universal device server pentru conversia de la protocolul RS232 la protocolul Ethernet.

Programul este scris în C++ și rulat pe un PC cu o frecvență de eșantionare de 100 ms. În continuare sunt prezentate simulări și implementări în timp real pentru a valida metodele propuse. Pentru a trimite comenzile calculate de algoritmii de conducere am folosit funcțiile din ARIA: setVel(Velocity), setVel2(leftVelocity,RightVelocity), setRotVel(Angular Velocity). Am citit datele de la robot cu ajutorul funcțiilor ARIA: getX(), getY(), getTH(), getRotVel(), getVel().

5.1. Simularea conducerii roboților mobili și vehiculelor autonome

Controlerul propus pentru conducerea roboților mobili și vehiculelor autonome a fost testat în simulări realizate cu ajutorul softului MobileSim[39] de la Mobile Robots. MobileSim este un soft pentru simularea platformelor MobileRobots/ActivMedia și mediile lor și experimentare cu ARIA. Înlocuiește SRIsim distribuit anterior cu ARIA.

MobileSim are la bază simulatorul Stage, creat de Richard Vaughan, Andrew Howard, și alții ca parte din proiectul Player/Stage, cu niște modificări făcute de MobileRobots. MobileSim poate simula comportamentul tuturor roboților produși de MobileRobots. Pentru simularea evitării obstacolelor a fost creată o hartă în Mapper3basic în care a fost introdus obstacolul și această hartă a fost încărcată în MobileSim. Programele sunt scrise în C++ și compilate în Visual Studio. Comunicarea cu simulatorul se realizează cu ajutorul funcțiilor ARIA.

Softul Aria realizează conectarea automată la simulatorul MobileSim în cazul în care nu este detectat nici un robot conectat la portul COM1. Simulatorul are implementate modelele cinematice ale roboților, funcții pentru simularea sonarelor și laserelor, care sunt folosite pentru a simula comportamentul unui robot real. Programul scris în C++ apelează funcțiile ARIA în cazul în care se dorește trimiterea unor comenzi către simulator sau citirea datelor simulate.

- Simularea 5.1

În acest test s-a dorit urmărirea unei traiectorii liniare cu viteza de 0.5m/s, urmată de o traiectorie circulară, cu viteza unghiulara 0.2 radiani/s, de către vehiculul autonom 4DW/SW SEEKUR utilizând conducerea sliding-mode în timp continuu. Parametrii constanți folosiți în acest experiment sunt: $Q_1 = 0.05$, $Q_2 = 0.5$, $P_1 = 0.5$, $P_2 = 0.75$, $k_0 = 30$, $k_1 = 1.25$, $k_2 = 100$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor. Arhitectura de conducere utilizând MobileSim este prezentată în Fig. 5.3. Modulul Trajectory Planner constă în acest caz în impunerea unei traiectorii cu viteza liniară de $0.5m/s$ și $\delta_d = 0$ timp de 20 s, după 20 s se impune valoarea $\delta_d = a\tan(0.2 \cdot 0.8/0.5)$, în care 0.2 radiani/s reprezintă viteza unghiulară, 0.8 lungimea vehiculului, 0.5 viteza liniară. Modulul Odometry este implementat de softul ARIA și viteza liniară, unghiul director, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare. Programul folosit pentru implementarea conducerii sliding-mode în timp continuu este prezentat în Anexa 2. Programul folosit la conducerea sliding-mode Comunicarea dintre programul de conducere și simulator este realizată de ARIA.

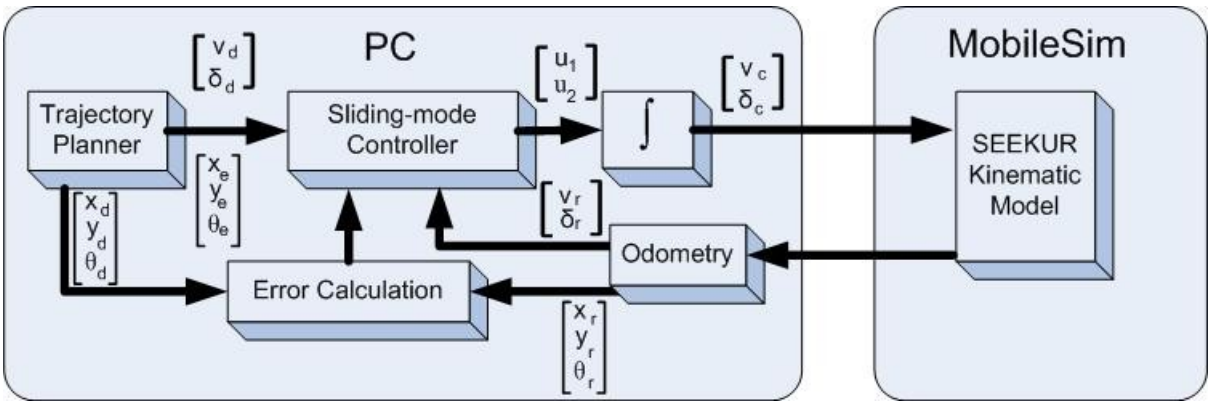

Fig. 5.3. Arhitectura de conducere a AV 4DW/SW SEEKUR în mediul MobileSIM

În continuare sunt prezentate graficele rezultate. În Fig. 5.4 este prezentată simularea în MobileSim a traiectoriei vehiculului autonom SEEKUR utilizând conducerea sliding-mode în

timp continuu. În Fig. 5.5 este prezentată traiectoria reală cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră rezultată în urma simulării conducerii sliding-mode în timp continuu. Fig. 5.6 prezintă eroarea pe axa Ox obținută prin simularea conducerii sliding-mode în timp continuu.

Fig. 5.7 prezintă eroarea pe axa Oy generată de simularea conducerii sliding-mode în timp continuu. Fig. 5.8 prezintă eroarea de orientare a robotului în cazul simulării în conducerii sliding-mode în timp continuu a vehiculului autonom SEEKUR. În Fig. 5.9 este prezentată suprafața de comutație s_1 și în Fig. 5.10 este prezentată suprafața de comutație s_2 calculată în timpul simulării conducerii sliding-mode în timp continuu a vehiculului autonom SEEKUR.

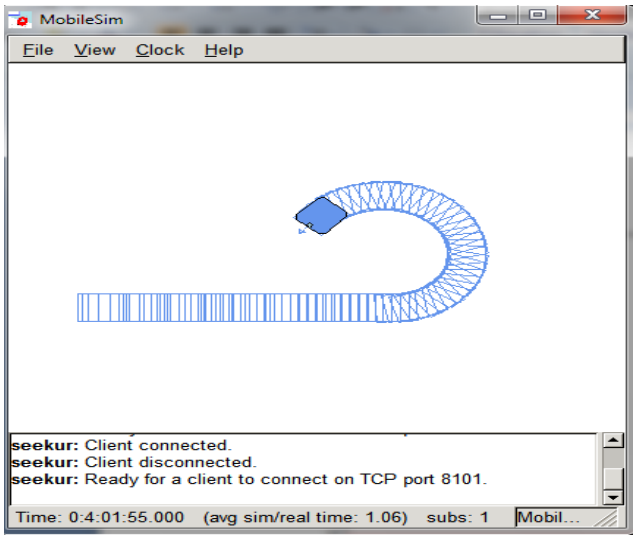

Fig. 5.4 Traiectoria obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

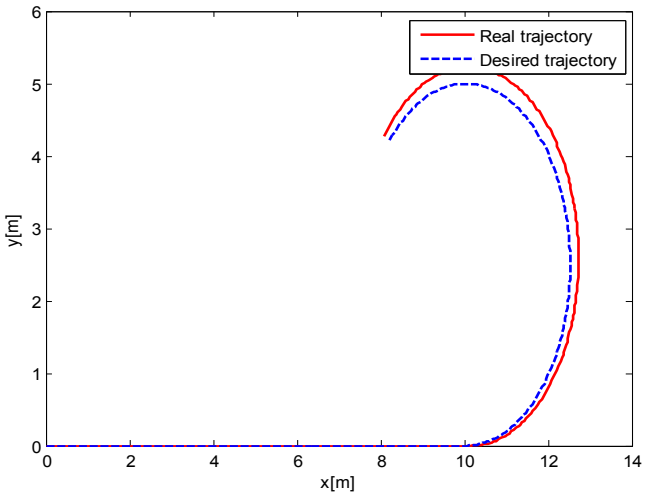

Fig. 5.5 Traiectoria obținută prin simulare în MobileSim și traiectoria impusă la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

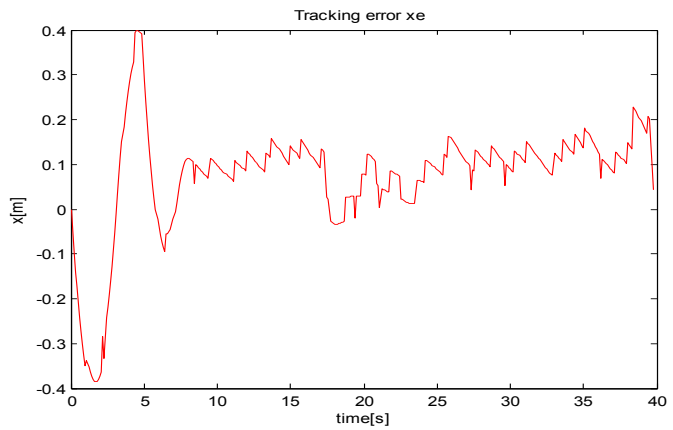

Fig. 5.6 Eroarea de urmărire pe axa X obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

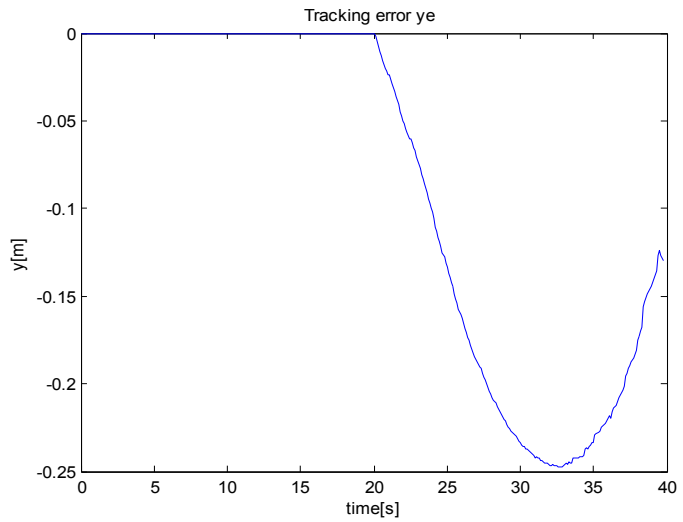

Fig. 5.7 Eroarea de urmărire pe axa Y obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

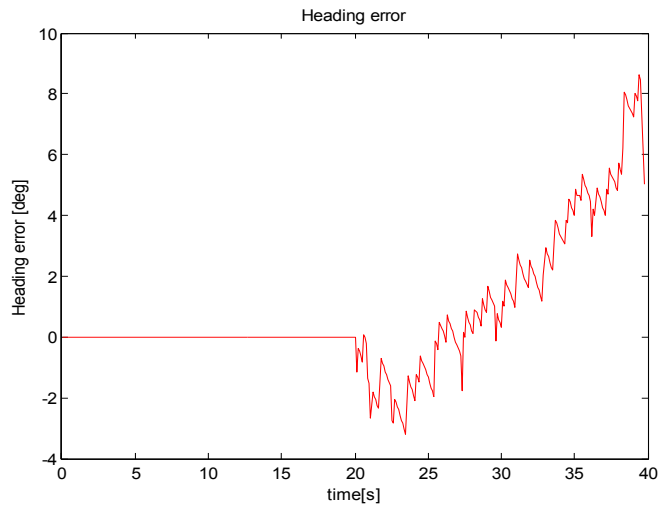

Fig. 5.8 Eroarea de urmărire a direcției obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

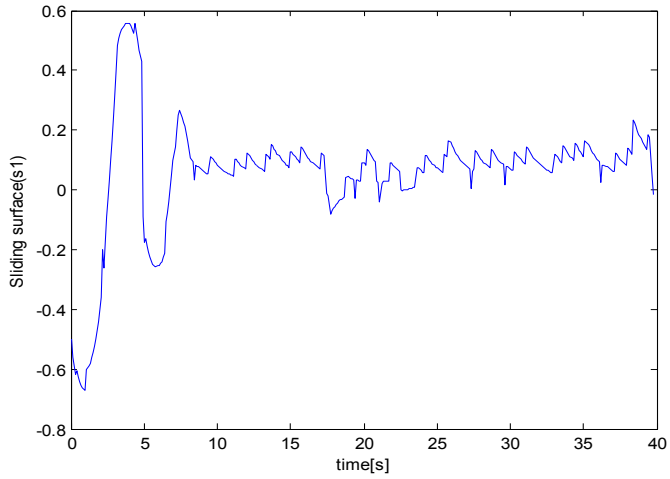

Fig. 5.9 Suprafața de comutație s1 obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

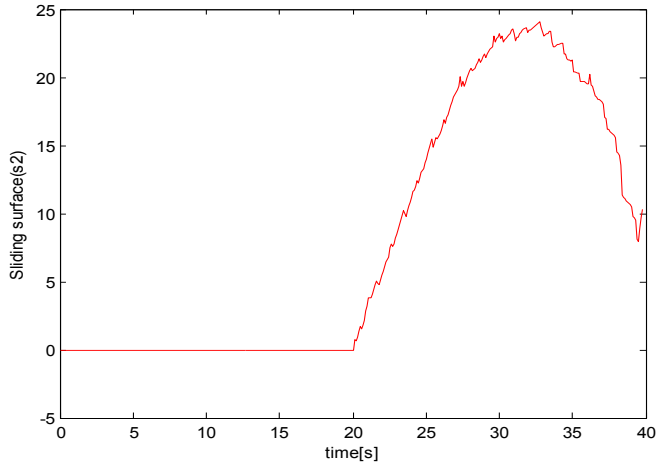

Fig. 5.10 Suprafața de comutație s2 obținută prin simulare în MobileSim la conducerea sliding-mode în timp continuu a vehiculului autonom 4DW/SW SEEKUR.

Se observă că erorile de poziționare sunt mici, performanțele de urmărire a traiectoriei sunt destul de bune în cazul folosirii conducerii sliding-mode în timp continuu a vehiculului autonom SEEKUR.

- Simularea 5.2.

Urmărirea unei traiectorii liniare urmată de o traiectorie în formă de S a fost testată pentru robotul mobil PatrolBot utilizând conducerea sliding-mode în timp discret. Parametrii constanți folosiți în acest experiment sunt: $q_1 = 0.9$, $q_2 = 0.9$, $\varepsilon_1 = 0.01$, $\varepsilon_2 = 0.75$, $k_0 = 9.5$, $k_1 = 0.75$, $k_2 = 15$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor. Arhitectura de conducere folosită este prezentată în Fig. 5.11. Modulul Trajectory Planner constă în acest caz în impunerea vitezeri liniare de 0.4m/s și vitezei unghiulare de 0 radiani/s timp de 10s, apoi impunerea vitezei

unghiulare de 0.2 radiani/s timp de 15s și în final impunerea vitezei unghiulare de -0.2 radiani/s timp de 15s. Modulul Odometry este implementat de softul ARIA și viteza liniară, viteza unghiulară, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare. Pentru implementarea conducerii sliding-mode în timp discret se folosește programul prezentat în Anexa 2. Programul folosit la conducerea sliding-mode, în care se folosesc ecuațiile conducerii în timp discret prezentate în secțiunea 3.3.1 Conducerea sliding-mode în timp discret a WMR cu 2 DW/2FW Comunicarea dintre programul de conducere și simulator este realizată de ARIA.

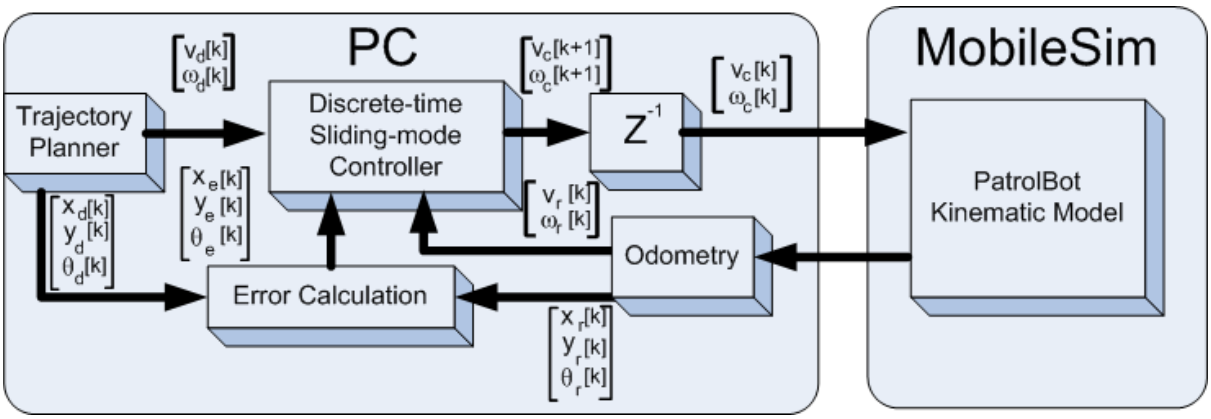

Fig. 5.11. Arhitectura de conducere a WMR 2DW/2FW PatrolBot în mediul MobileSIM

În Fig. 5.12 este prezentată simularea în MobileSim a traiectoriei robotului PatrolBot utilizând conducerea sliding-mode în timp discret. În Fig. 5.13 este prezentată traiectoria simulată în MobileSim a conducerii sliding-mode în timp discret a robotului Patrolbot și cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră.

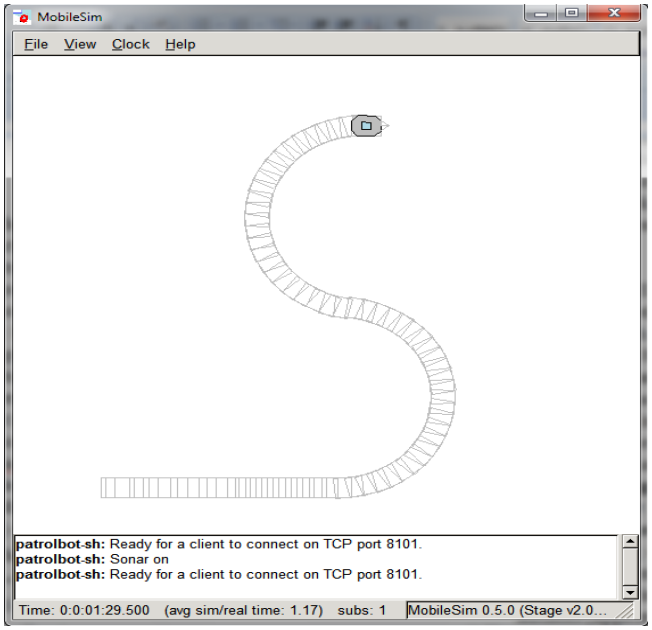

Fig. 5.12. Traiectoria obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot

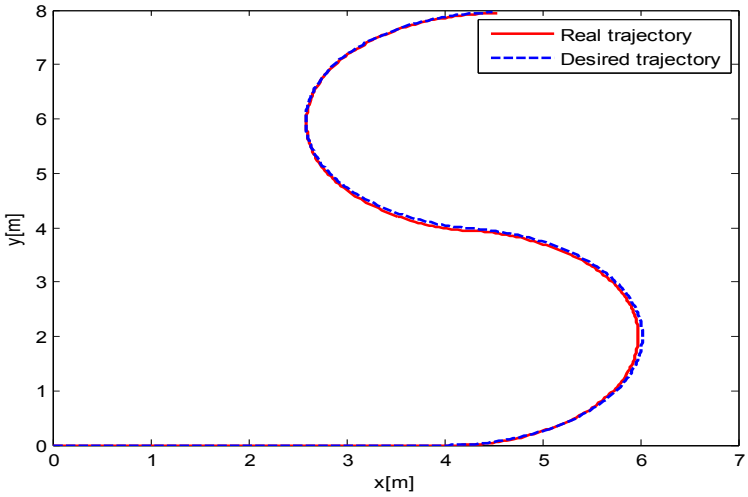

Fig. 5.13 Traiectoria obținută prin simulare în MobileSim și traiectoria impusă la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot

Fig. 5.14 prezintă eroarea pe axa Ox de urmărire traiectoriei a robotului PatrolBot utilizând conducerea sliding-mode în timp discret. Fig. 5.15 prezintă eroarea pe axa Oy de urmărire traiectoriei a robotului PatrolBot utilizând conducerea sliding-mode în timp discret. Fig. 5.16 prezintă eroarea de orientare a robotului de urmărire traiectoriei a robotului PatrolBot utilizând conducerea sliding-mode în timp discret. În Fig. 5.17 este prezentată suprafața de comutație s_1 de urmărire traiectoriei a robotului PatrolBot utilizând conducerea sliding-mode în timp discret. În Fig. 5.18este prezentată suprafața de comutație s_2 de urmărire traiectoriei a robotului PatrolBot utilizând conducerea sliding-mode în timp discret.

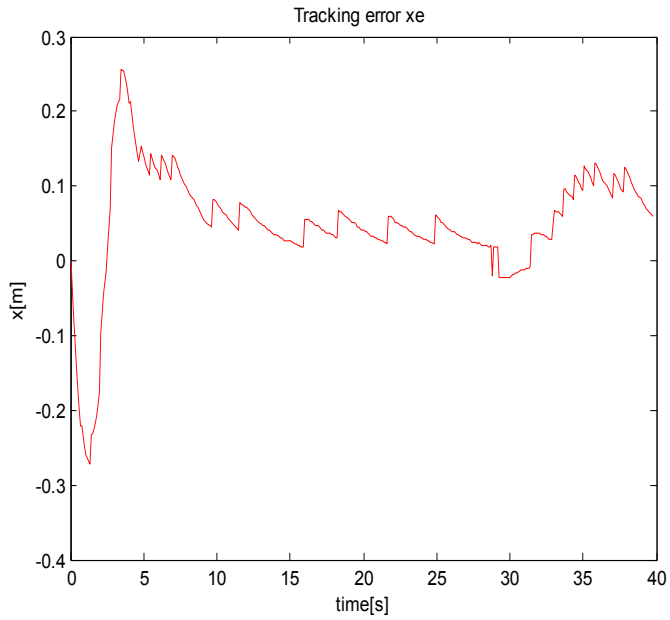

Fig. 5.14 Eroarea de urmărire pe axa X obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot

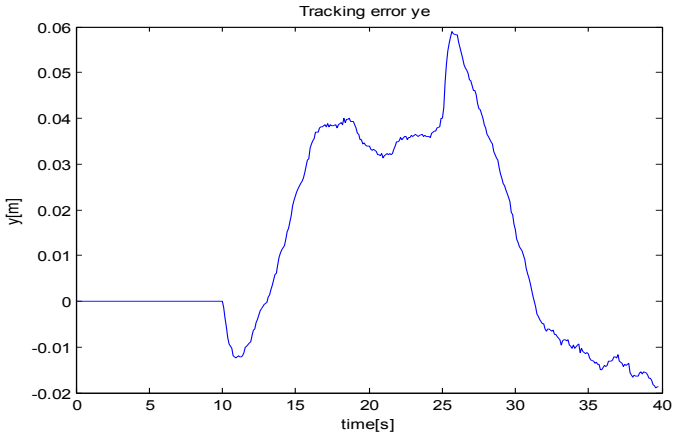

Fig. 5.15 Eroarea de urmărire pe axa Y obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot

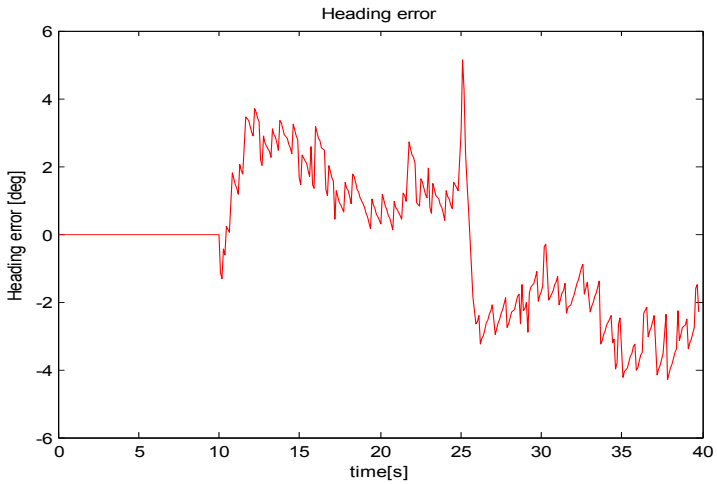

Fig. 5.16 Eroarea de urmărire a direcției obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot.

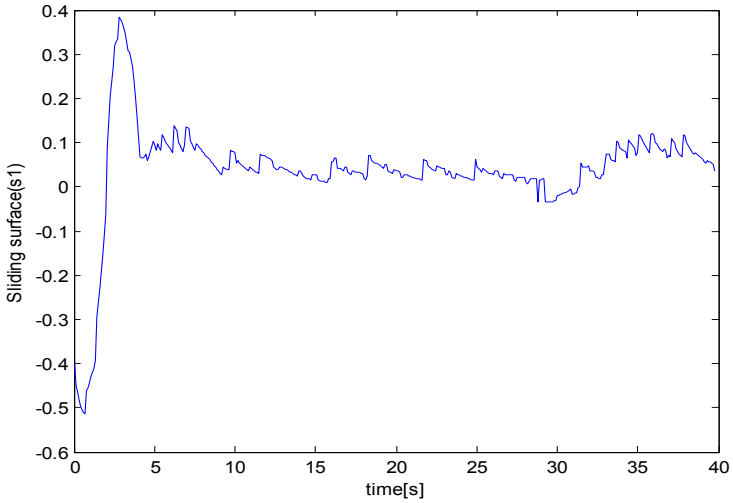

Fig. 5.17 Suprafața de comutație s1 obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot.

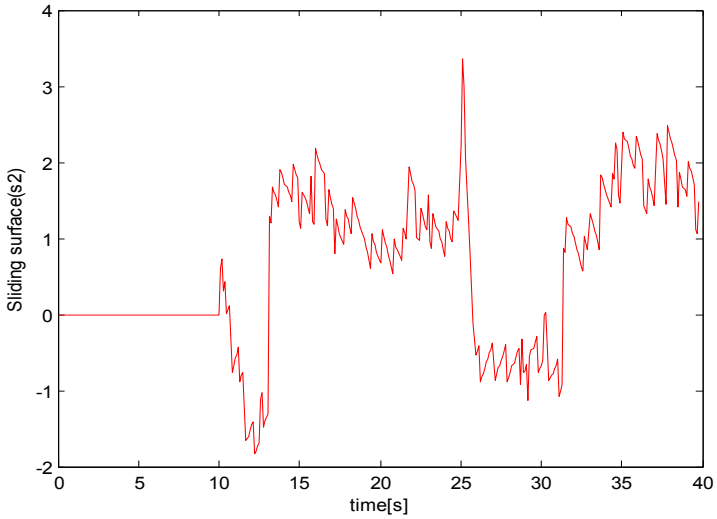

Fig. 5.18 Suprafața de comutație s2 obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a robotului 2DW/2FW PatrolBot.

Din figurile prezentate se observă că robotul urmărește traiectoria dorită cu erori mici.

- Simularea 5.3

Testul își propune urmărirea unei traiectorii în forma din Fig. 5.20 utilizând vehiculul autonom SEEKUR folosind conducerea sliding-mode în timp discret. Parametrii constanți folosiți în acest experiment sunt: $q_1 = 0.9$, $q_2 = 0.9$, $\varepsilon_1 = 0.01$, $\varepsilon_2 = 0.75$, $k_0 = 30$, $k_1 = 0.75$, $k_2 = 15$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor. În Fig. 5.19 este prezentată arhitectura conducerii folosite. În acest caz modulul Trajectory Planner folosește planificatorul de traiectorii din [96] pentru generarea vitezei unghiulare și unghiului director. Modulul Odometry este implementat de softul ARIA și viteza liniară, unghiul director, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare.

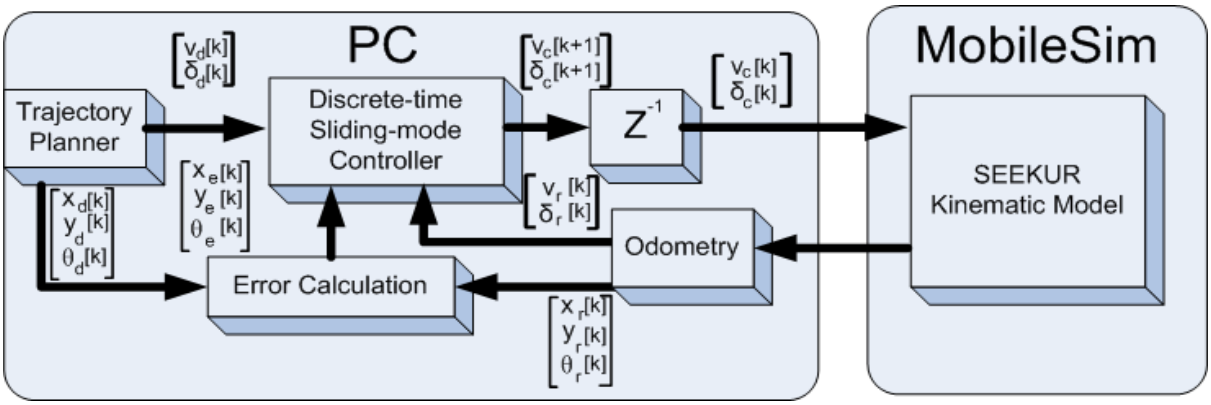

Fig. 5.19. Arhitectura de conducere sliding-mode în timp discret a AV 4DW/SW SEEKUR în MobileSIM.

Pentru implementarea conducerii sliding-mode în timp discret se folosește programul prezentat în Anexa 2. Programul folosit la conducerea sliding-modeîn care se folosesc ecuațiile conducerii în timp discret prezentate în secțiunea 3.3.2 Conducerea sliding-mode în timp discret a vehiculelor autonome3.3.1 Conducerea sliding-mode în timp discret a WMR cu 2 DW/2FW .Comunicarea dintre programul de conducere și simulator este realizată de ARIA.

Fig. 5.20 prezintă simularea în MobileSim a traiectoriei generate de conducerea sliding-mode în timp discret a vehiculului autonom SEEKUR.

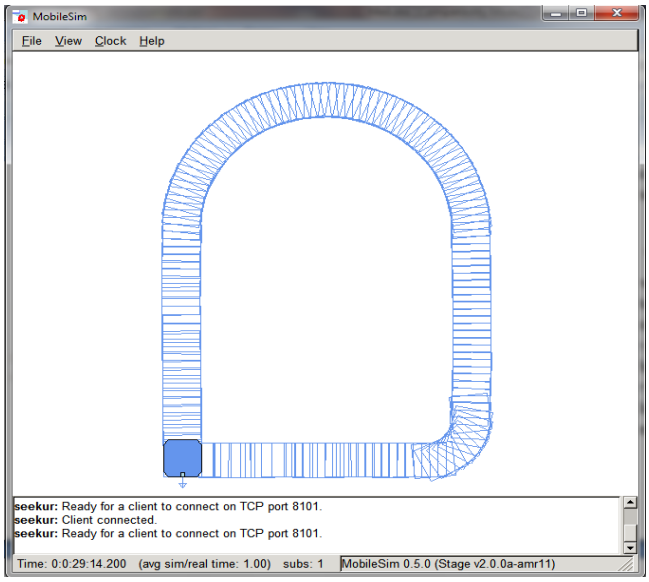

Fig. 5.20 Traiectoria obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

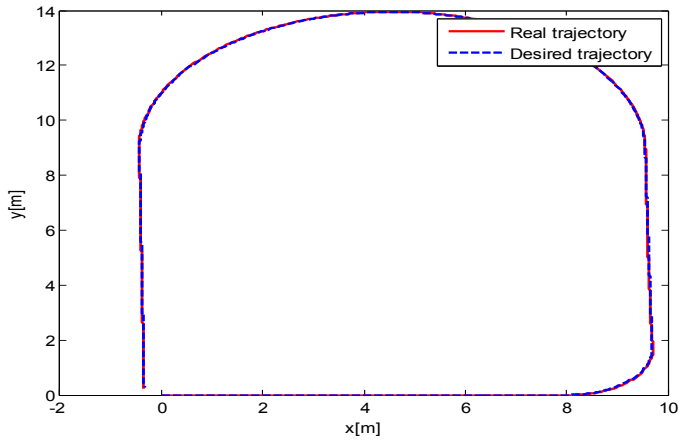

Fig. 5.21 Traiectoria obținută prin simulare în MobileSim și traiectoria impusă la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

În Fig. 5.21 este prezentată traiectoria simulată utilizând conducerea sliding-mode în timp discret cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră a

vehiculului autonom SEEKUR. Fig. 5.22 prezintă eroarea simulată pe axa Ox utilizând conducerea sliding-mode în timp discret a vehiculului autonom SEEKUR. Fig. 5.23 prezintă eroarea simulată pe axa Oy utilizând conducerea sliding-mode în timp discret a vehiculului autonom SEEKUR. Fig. 5.24 prezintă eroarea simulată de orientare a robotului utilizând conducerea sliding-mode în timp discret a vehiculului autonom SEEKUR. În Fig. 5.25 este prezentată simularea suprafeței de comutație s_1. În Fig. 5.26 este prezentată simularea suprafeței de comutație s_2.

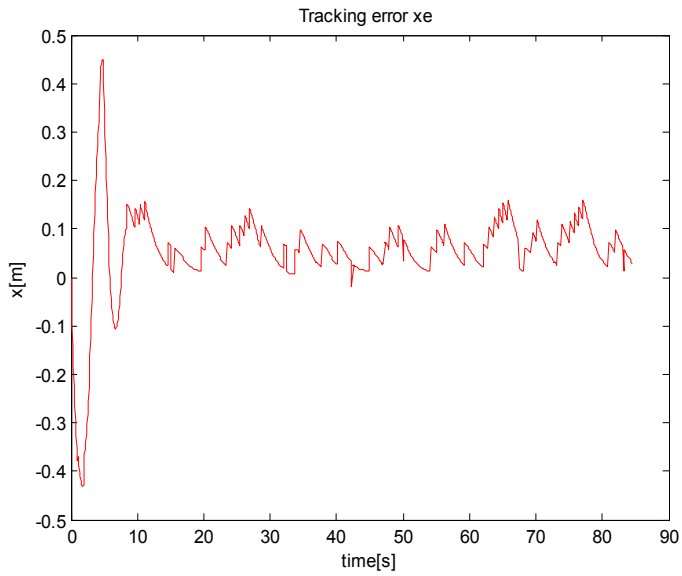

Fig. 5.22 Eroarea de urmărire pe axa X obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

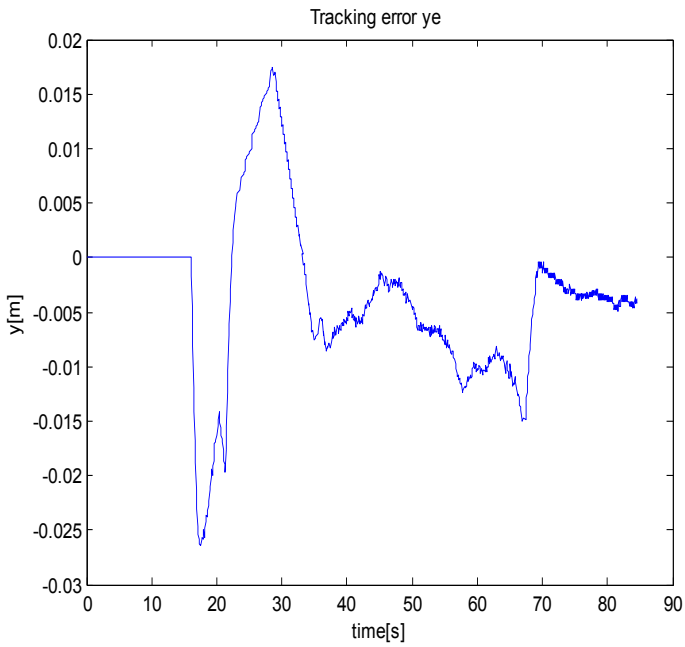

Fig. 5.23 Eroarea de urmărire pe axa Y obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

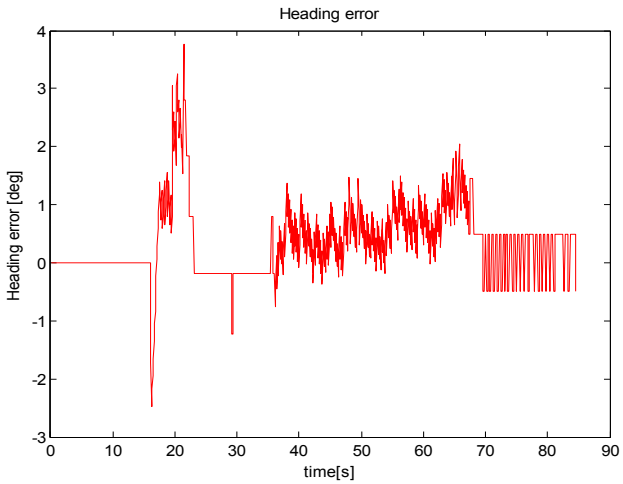

Fig. 5.24 Eroarea de urmărire a direcției obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

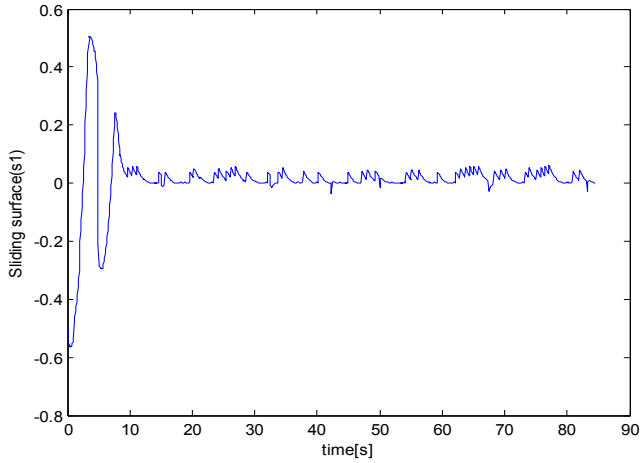

Fig. 5.25 Suprafața de comutație s1 obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

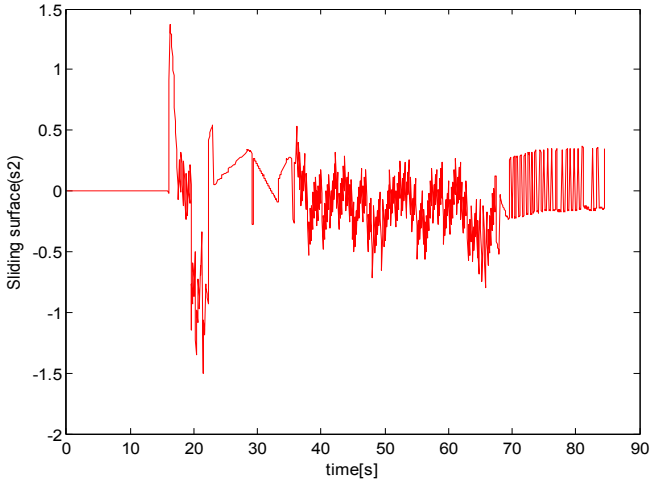

Fig. 5.26 Suprafața de comutație s2 obținută prin simulare în MobileSim la conducerea sliding-mode în timp discret a vehiculului autonom 4DW/SW SEEKUR.

Se observă performanțe ridicate la urmărirea traiectoriei propuse, cu erori de urmărire mici în toate graficele, iar suprafețele de alunecare oscilează în apropierea valorii zero.

- Simularea 5.4

Simularea a dorit urmărirea de către robotul PowerBot a traiectoriei prezentate în Fig. 5.28 utilizând conducerea backstepping, această traiectorie a fost generată de un planificator. Parametrii constanți folosiți în acest experiment sunt:, $k_1 = 0.5$, $k_2 = 1.25$, $k_1 = 0.3$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor.

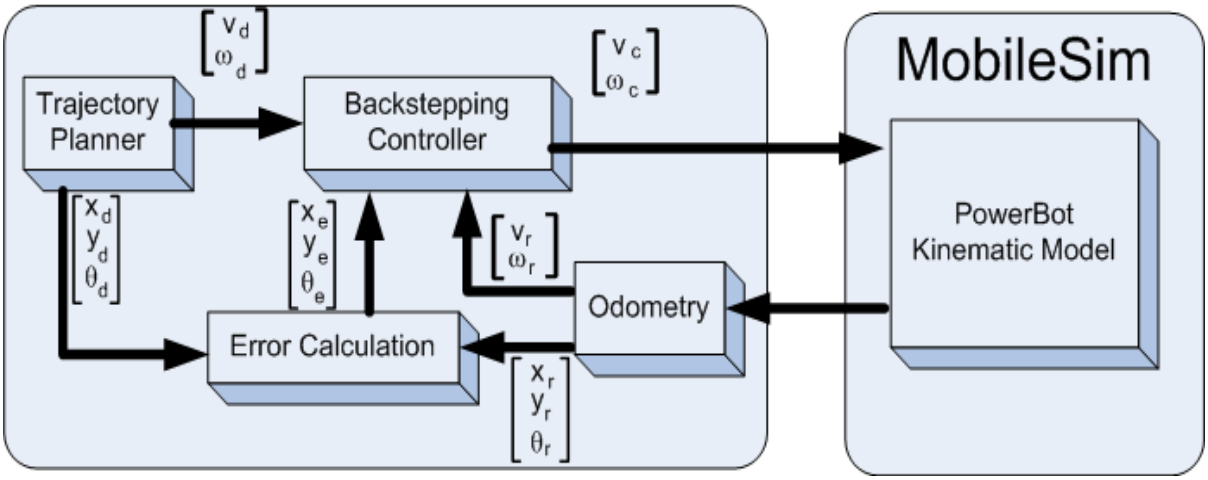

Fig. 5.27 Arhitectura de conducere backstepping a WMR 2DW/2FW PowerBot în MobileSim.

În acest caz modulul Trajectory Planner folosește planificatorul de traiectorii din [96] pentru generarea vitezei unghiulare și vitezei unghiulare. Modulul Odometry este implementat de softul ARIA și viteza liniară, viteza unghiulară, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare.

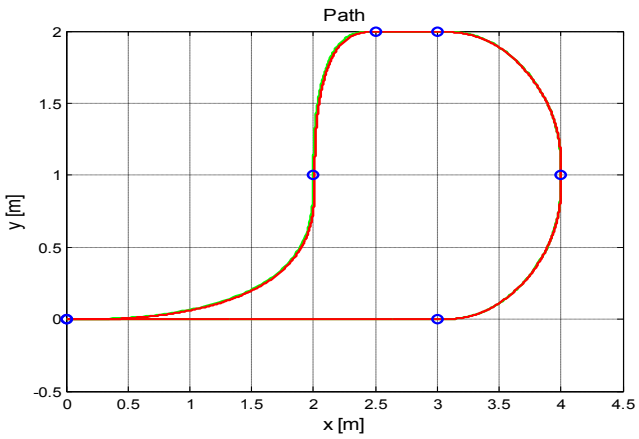

Fig. 5.28. Traiectoria generată de planificator pentru testarea conducerii backstepping a robotului 2DW/2FW Powerbot

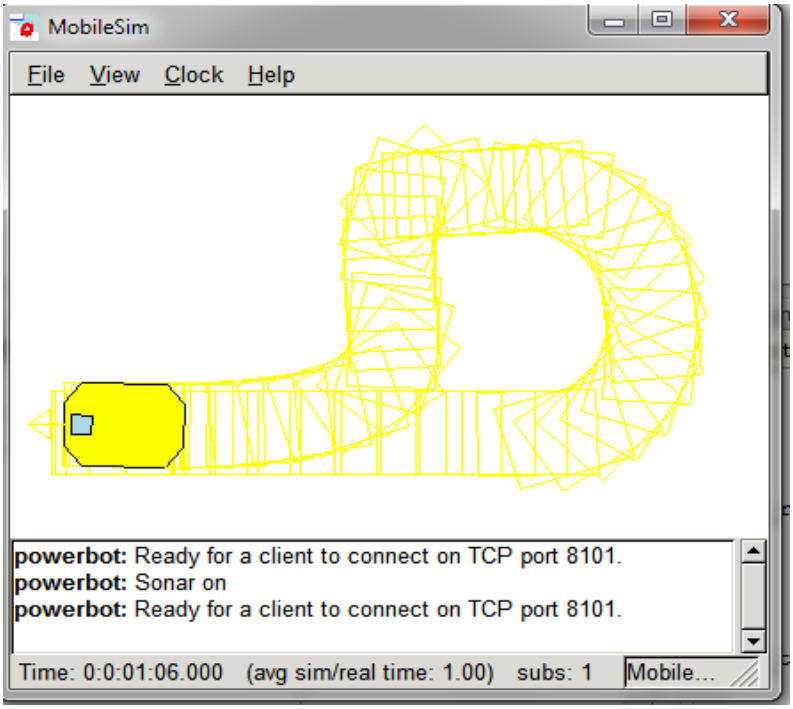

Fig. 5.29 Traiectoria obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot

În Fig. 5.30 este prezentată traiectoria simulată cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră a robotului PowerBot utilizând conducerea backstepping. Fig. 5.31 prezintă eroarea de urmărire a robotului PowerBot simulată pe axa Ox utilizând conducerea backstepping.

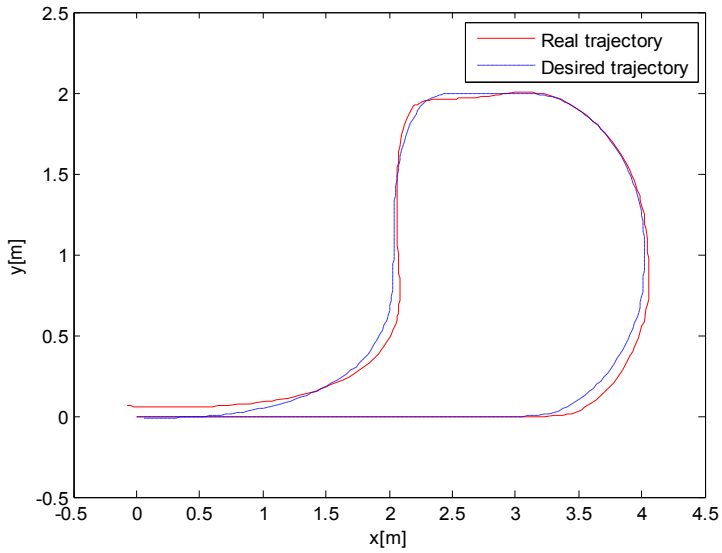

Fig. 5.30. Traiectoria obținută prin simulare în MobileSim și traiectoria impusă la conducerea backstepping a robotului 2DW/2FW Powerbot

Fig. **5.32** prezintă eroarea de urmărire simulată pe axa Oy a robotului PowerBot utilizând conducerea backstepping. Fig. 5.33 prezintă eroarea de orientare a robotului

simulată utilizând conducerea backstepping. În Fig. 5.34 este prezentată viteza simulată, viteza dorită și viteza calculată utilizând conducerea backstepping a robotului PowerBot. Fig. 5.35 prezintă viteza unghiulară dorită, viteza unghiulară reală, viteza unghiulară calculată utilizând conducerea backstepping a robotului PowerBot.

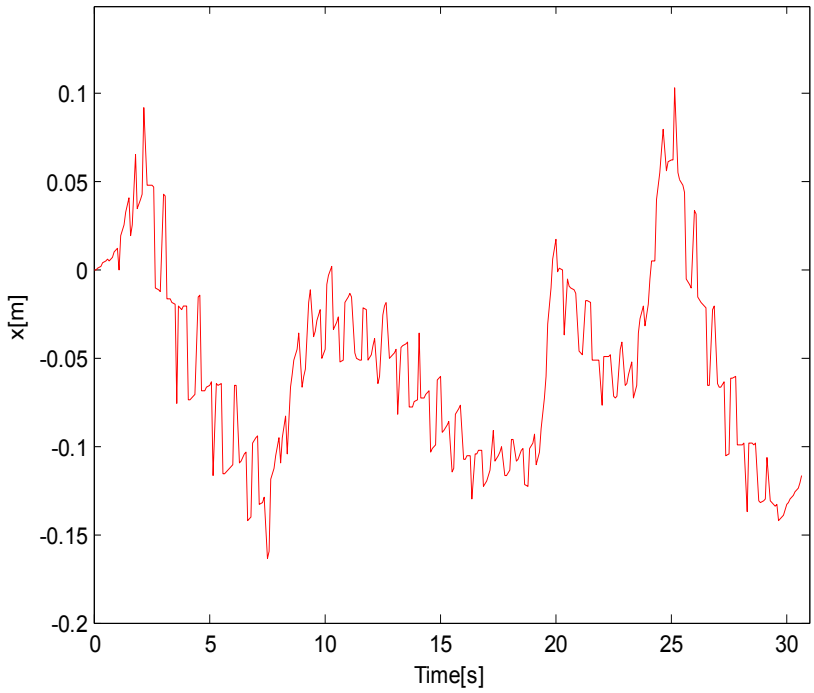

Fig. 5.31. Eroarea de urmărire pe axa X obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot

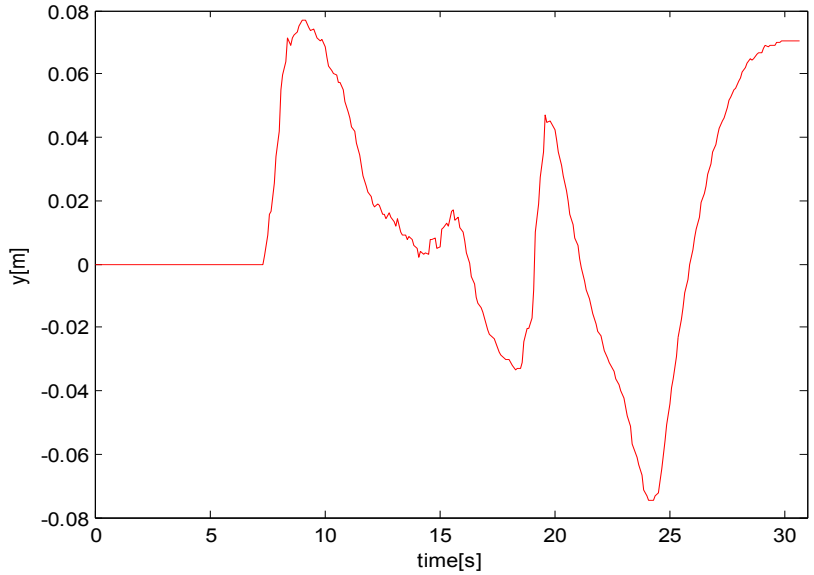

Fig. 5.32. Eroarea de urmărire pe axa Y obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot.

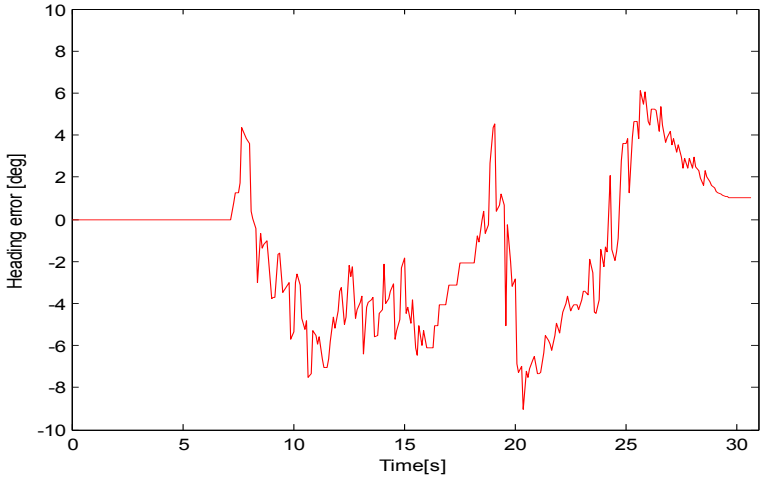

Fig. 5.33. Eroarea de urmărire a direcției obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot

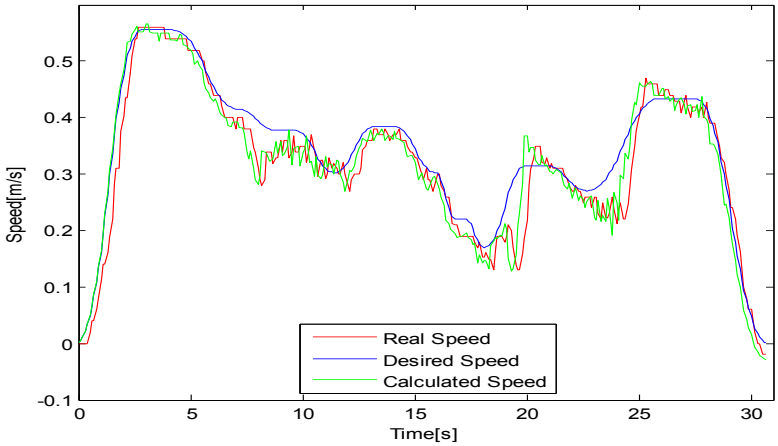

Fig. 5.34 Viteza liniară obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot

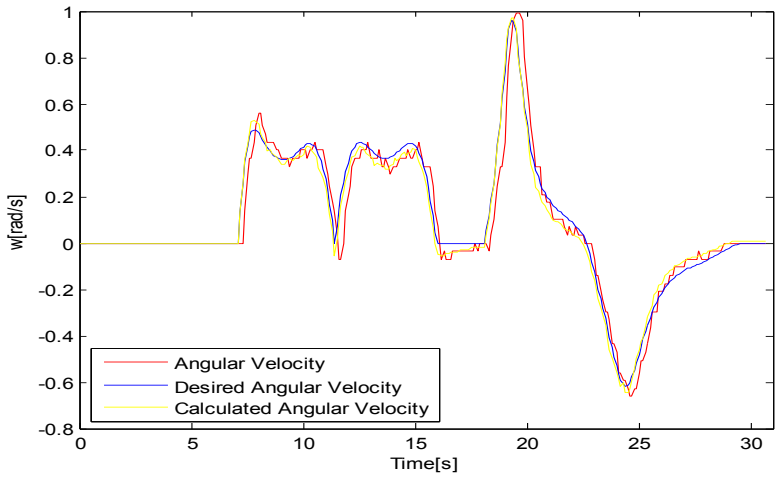

Fig. 5.35. Viteza unghiulară obținută prin simulare în MobileSim la conducerea backstepping a robotului 2DW/2FW Powerbot

Se observă că traiectoria propusă este urmărită și erorile de urmărire sunt mici. Din graficele vitezelor liniare și unghiulare se observă că vitezele dorite, vitezele calculate și vitezele reale au valori apropiate și această metodă de urmărire este eficientă.

Simularea 5.5

Folosind MobileSim a fost simulată funcționarea arhitecturii de evitare a obstacolelor propuse atât în lipsa obstacolelor cât și în cazul în care e prezent unul sau două obstacole. O hartă simulând holul de la intrarea în corpul Y cu și fără obstacole a fost creată utilizând Mapper3basic. Traiectoria care se dorește a fi urmărită este prezentată în exemplul fără obstacole (Fig. 5.37), iar modul în care obstacolele sunt tratate e prezentat în următoarele exemple.

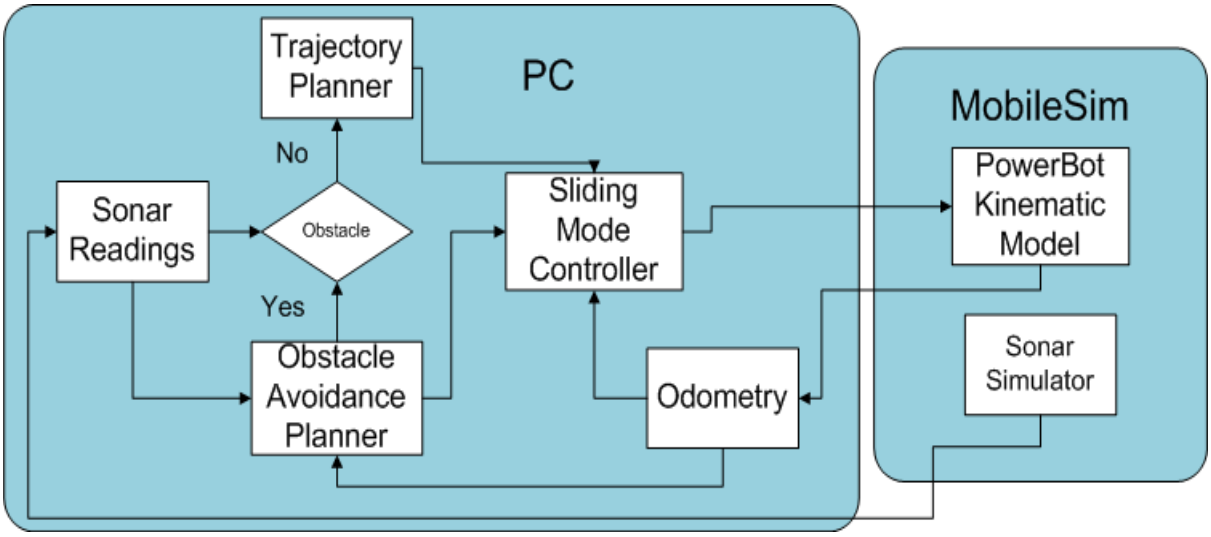

Fig. 5.36 Arhitectura algoritmului pentru evitarea obstacolelor a robotului 2DW/2FW PowerBot în MobileSim

Modulul de conducere folosit pentru testarea arhitecturii de evitare a obstacolelor utilizează conducerea sliding-mode în timp discret. În acest caz modulul Trajectory Planner folosește planificatorul de traiectorii din [96] pentru generarea vitezei liniare și vitezei unghiulare. Modulul Odometry este implementat de softul ARIA și viteza liniară, viteza unghiulară, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare. Conectarea la simulator este realizată de ARIA. Modulul de conducere este prezentat în Anexa 2. Programul folosit la conducerea sliding-mode.

Parametrii constanți sunt $q_1 = 0.9$, $q_2 = 0.9$, $\varepsilon_1 = 0.01$, $\varepsilon_1 = 0.5$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor. Fig. 5.36 prezintă arhitectura folosită pentru evitarea obstacolelor a WMR 2DW/2FW PowerBot.

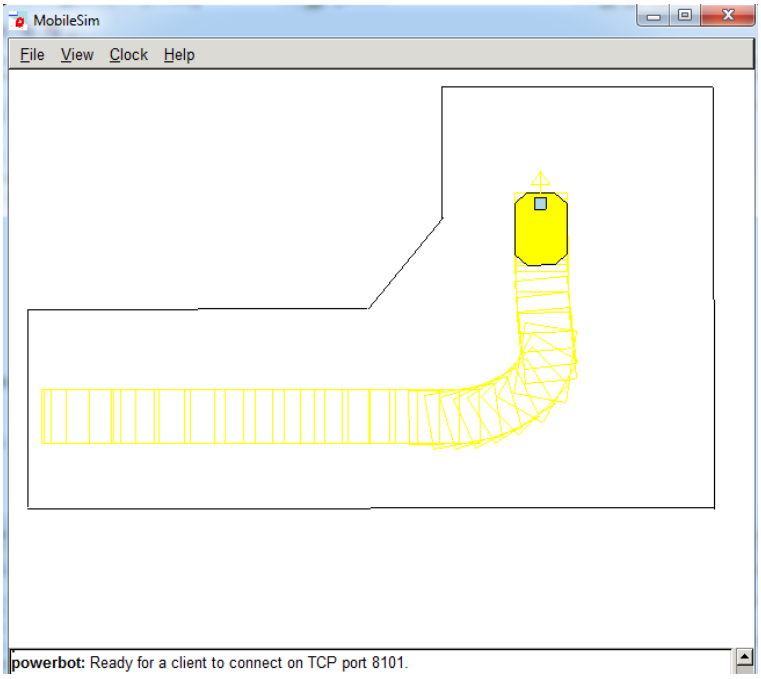

Fig. 5.37. Traiectoria obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot

Fig. 5.37. prezintă traiectoria rezultată în urma conducerii sliding-mode în timp discret de către robotul PowerBot. Se dorește obținerea unei traiectorii cât mai apropiată de aceasta în cazurile în care apar obstacole. În Fig. 5.38 este prezentată traiectoria reală cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră. Fig. 5.39 prezintă eroarea pe axa Ox. Fig. 5.40 prezintă eroarea pe axa Oy. Fig. 5.41 prezintă eroarea de orientare a robotului. În Fig. 5.42 este prezentată suprafața de comutație s_1. În Fig. 5.43 este prezentată suprafața de comutație s_2.

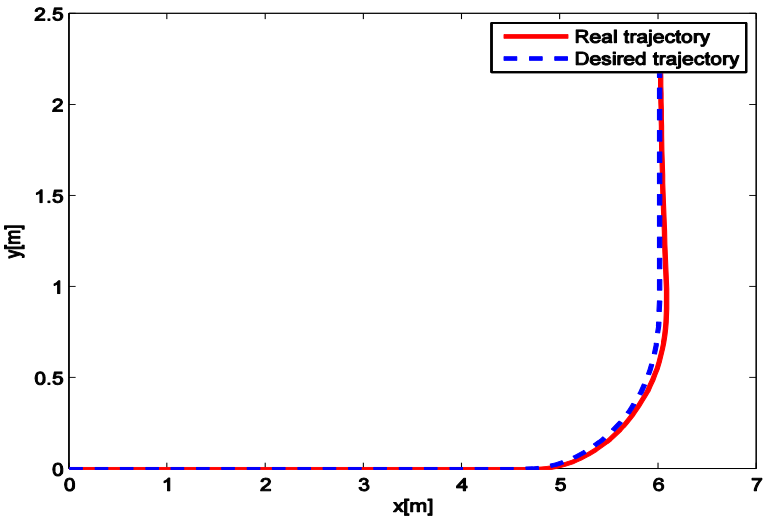

Fig. 5.38. Traiectoria obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot și traiectoria impusă

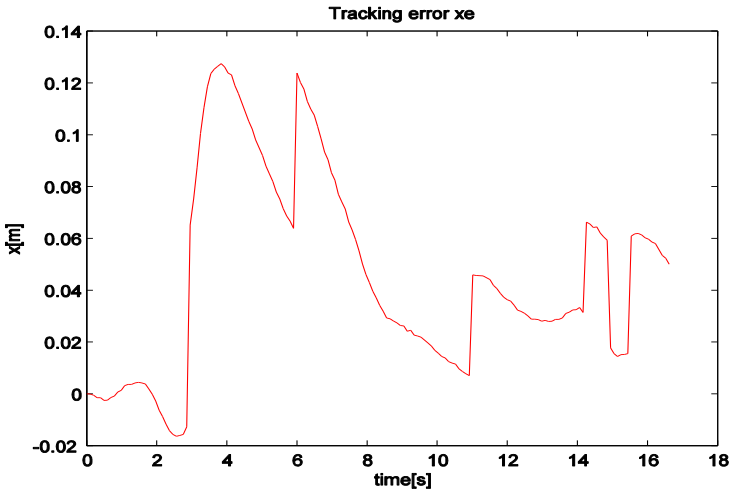

Fig. 5.39. Eroarea de urmărire pe axa X obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot.

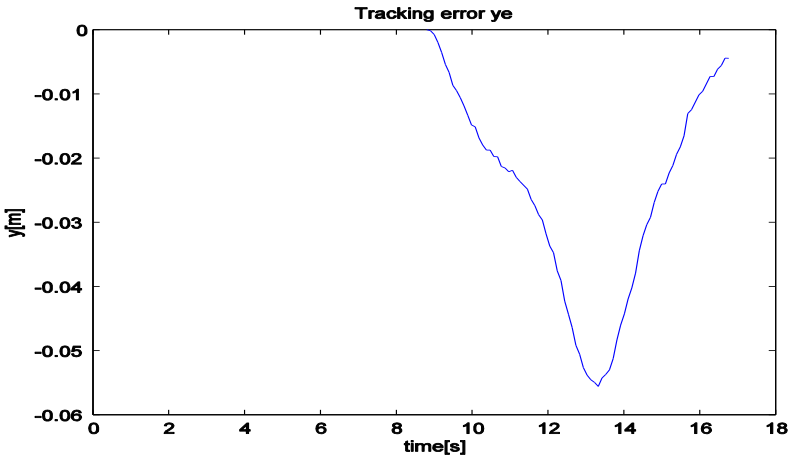

Fig. 5.40. Eroarea de urmărire pe axa Y obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot.

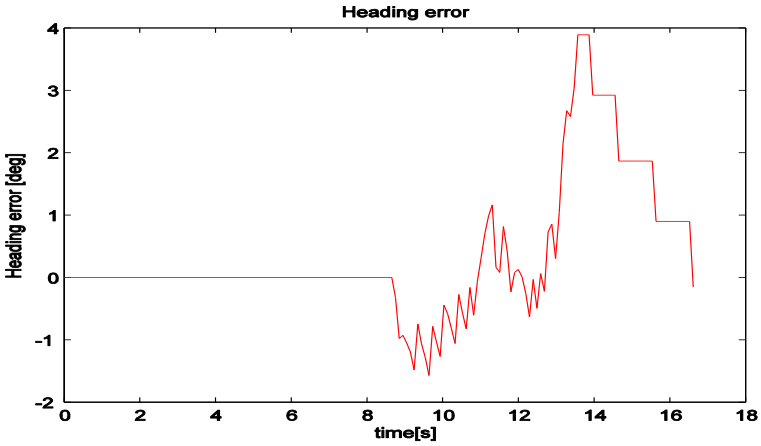

Fig. 5.41. Eroarea de urmărire a direcției obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot.

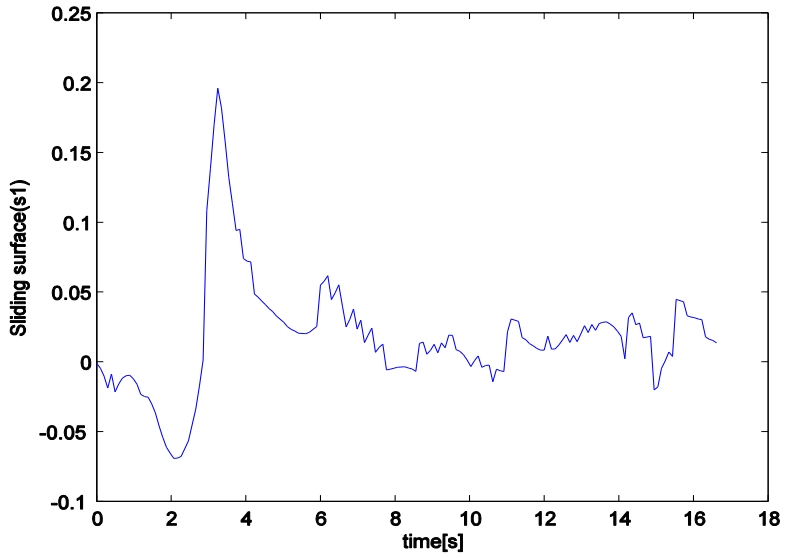

Fig. 5.42 Suprafața de alunecare s1 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor , a robotului 2DW/2FW PowerBot.

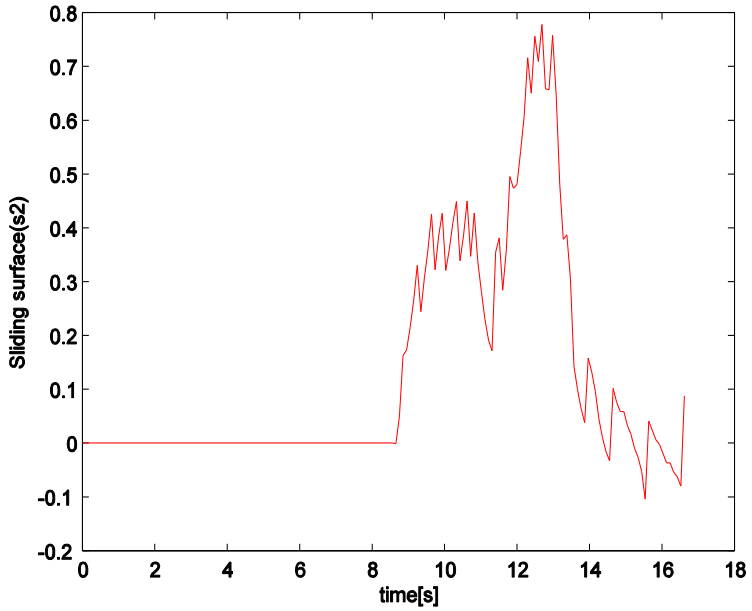

Fig. 5.43. Suprafața de alunecare s2 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în lipsa obstacolelor, a robotului 2DW/2FW PowerBot.

Se observă o bună urmărire a traiectoriei impuse cu erori mici, iar suprafețele de alunecare evoluează în apropierea valorii zero.

Simularea 5.6

Considerăm aceeasi hartă și traiectorie ca la simularea 5.5 dar adăugăm 1 obstacol. În Fig. 5.51 este prezentată traiectoria de evitare a obstacolului calculată.

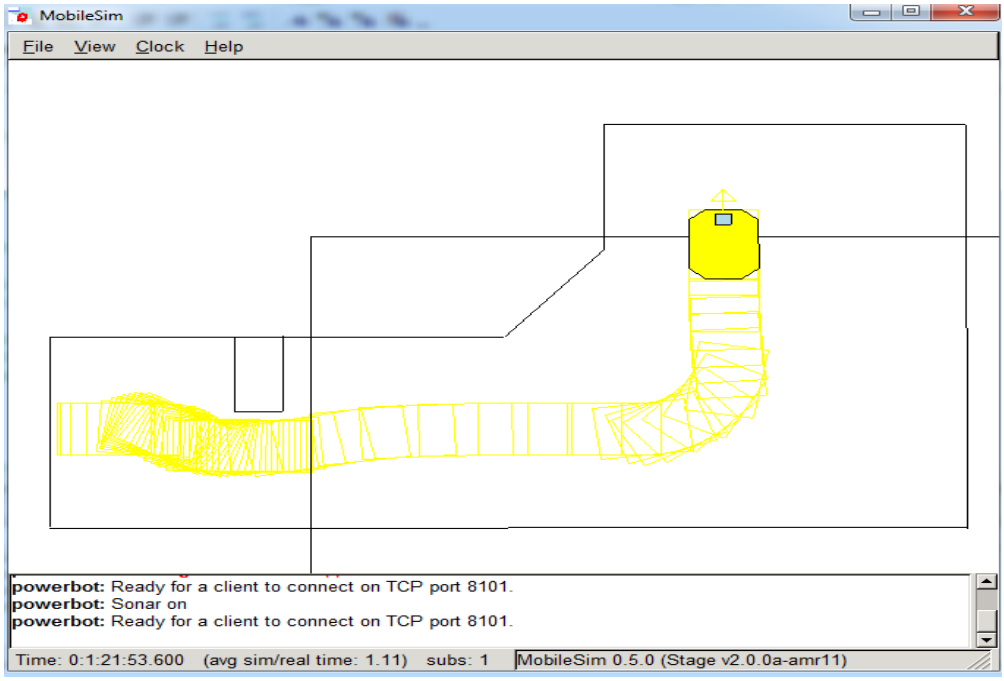

Fig. 5.44. Traiectoria obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot

În Fig. 5.45 este prezentată traiectoria reală cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră.

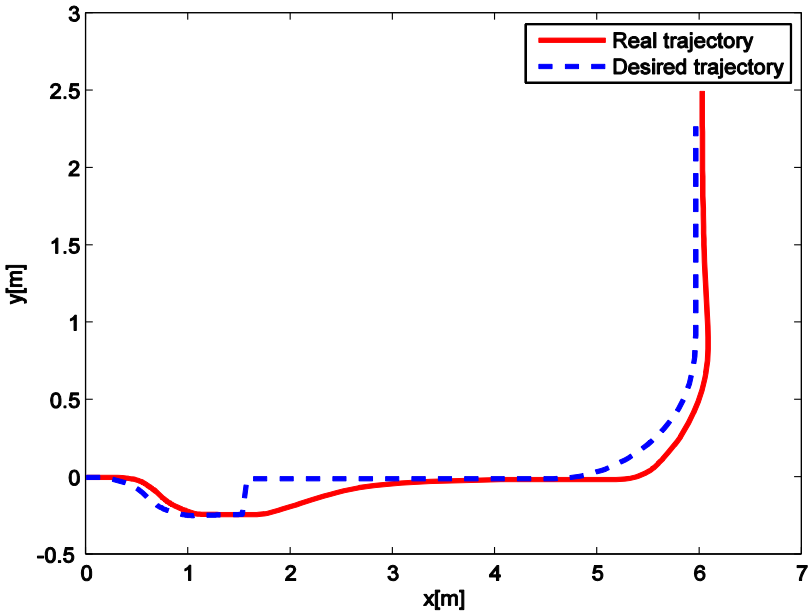

Fig. 5.45. Traiectoria obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot și traiectoria impusă

Fig. 5.46 prezintă eroarea pe axa Ox. Fig. 5.47prezintă eroarea pe axa Oy. Fig. 5.48 prezintă eroarea de orientare a robotului. În Fig. 5.49 este prezentată suprafața de alunecare s_1.

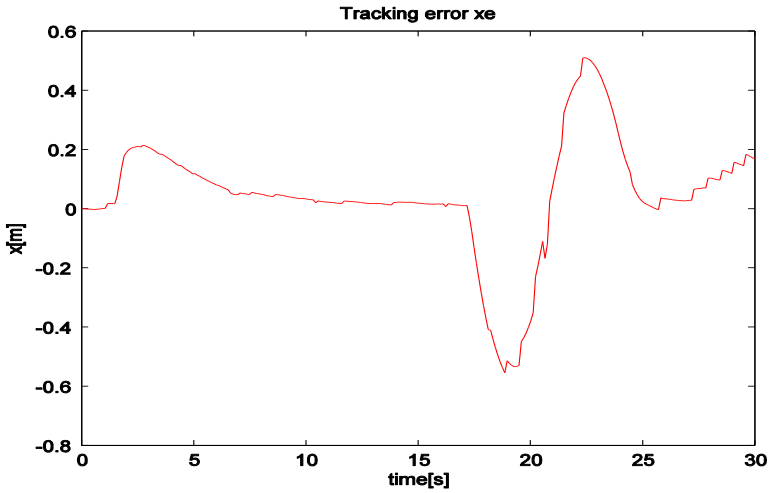

Fig. 5.46. Eroarea de urmărire pe axa X obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot.

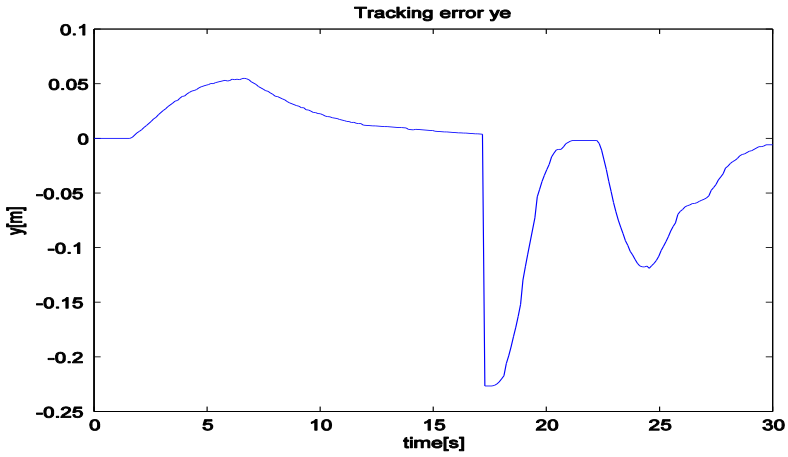

Fig. 5.47. Eroarea de urmărire pe axa Y obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot.

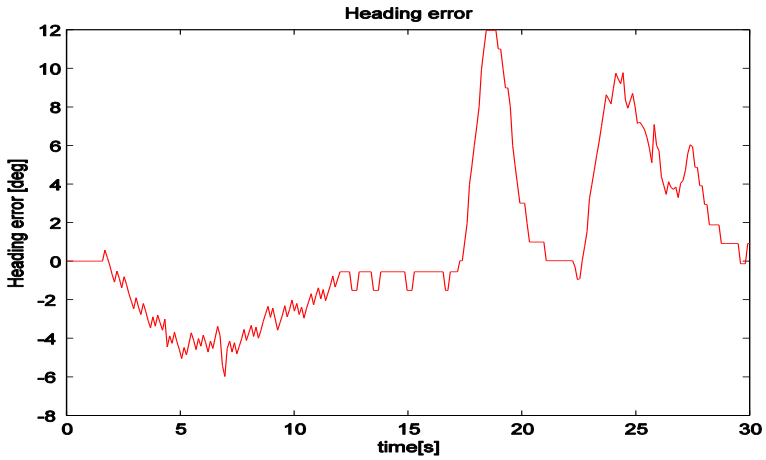

Fig. 5.48. Eroarea de urmărire a direcției obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot.

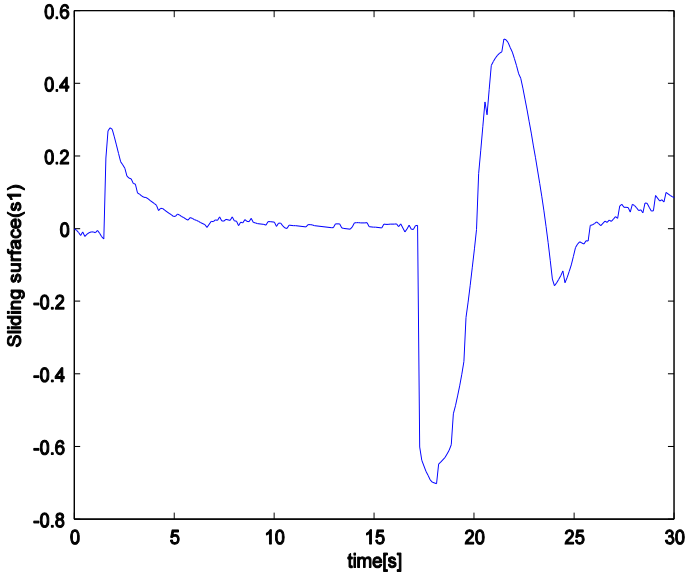

Fig. 5.49. Suprafața de alunecare s1 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot.

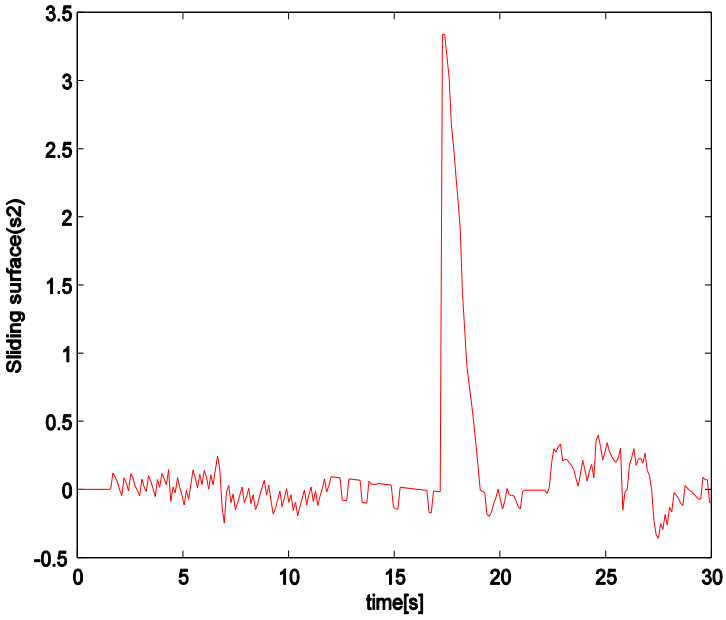

Fig. 5.50. Suprafața de alunecare s2 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența unui obstacol, a robotului 2DW/2FW PowerBot.

Se observă că traiectoria dorită este urmărită chiar și în prezența obstacolului, dar se constată o creștere a erorilor de urmărire cauzată de traiectoria de evitare a obstacolului și revenirea la traiectoria inițială.

Simularea 5.7.

Simularea 5.7 este asemănătoare cu simularea 5.6 dar se mai introduce un obstacol.

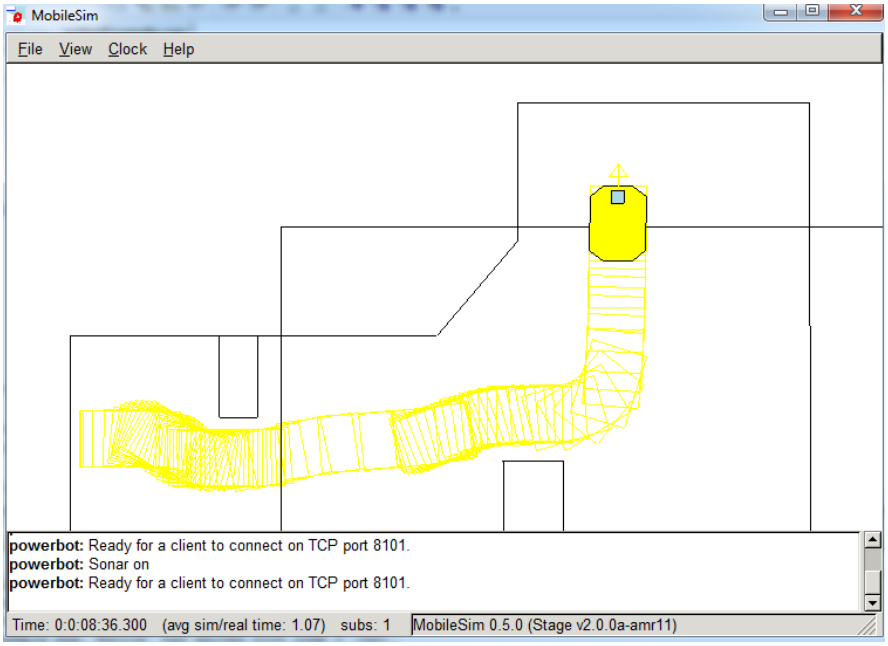

Fig. 5.51. Traiectoria obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot

În Fig. 5.52 este prezentată traiectoria reală cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră. Fig. 5.53 prezintă eroarea pe axa Ox. Fig. 5.54 prezintă eroarea pe axa Oy. Fig. 5.55 prezintă eroarea de orientare a robotului. În Fig. 5.56 este prezentată suprafața de comutație s_1. În Fig. 5.57 este prezentată suprafața de comutație s_2.

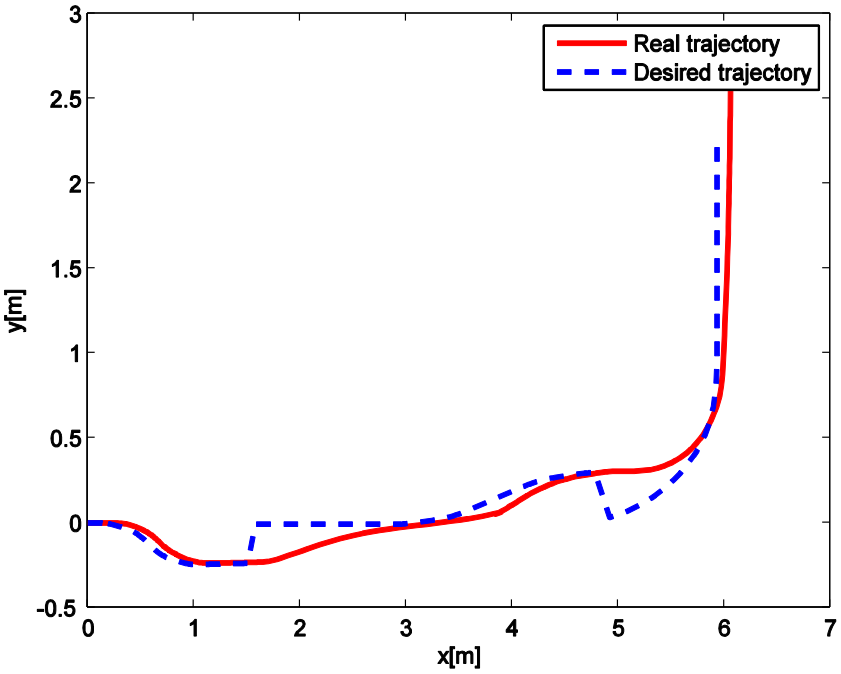

Fig. 5.52. Traiectoria de evitare a obstacolului obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot și traiectoria dorită.

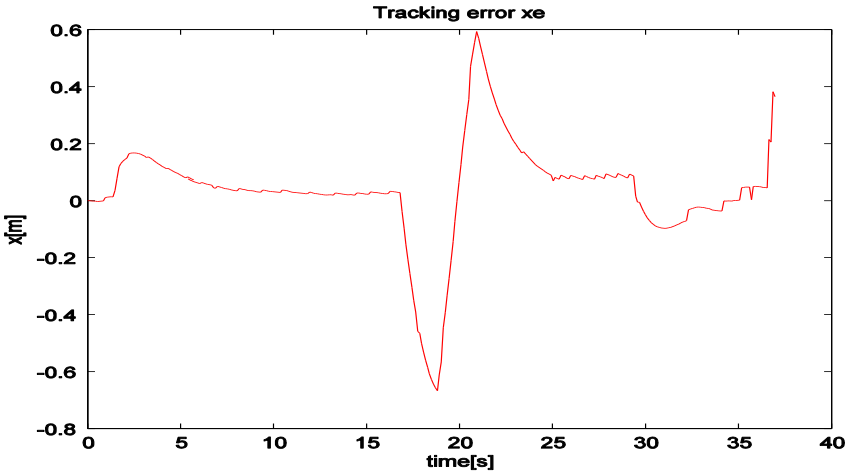

Fig. 5.53. Eroarea de urmărire pe axa X obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot.

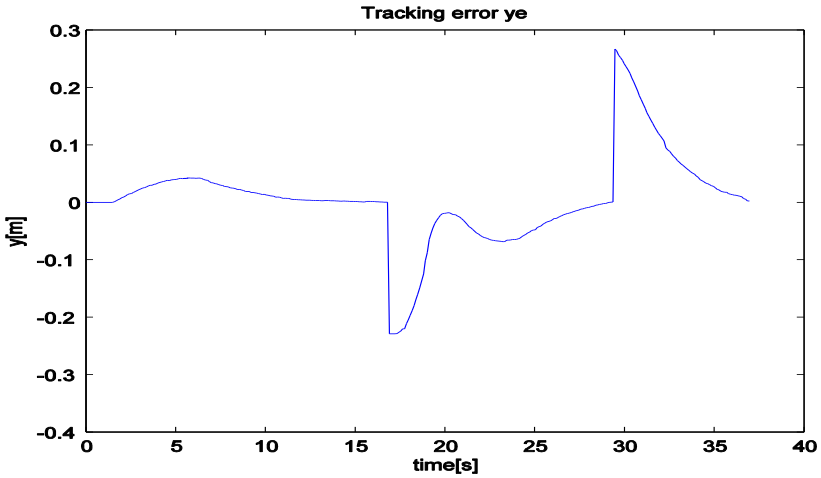

Fig. 5.54. Eroarea de urmărire pe axa Y obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot.

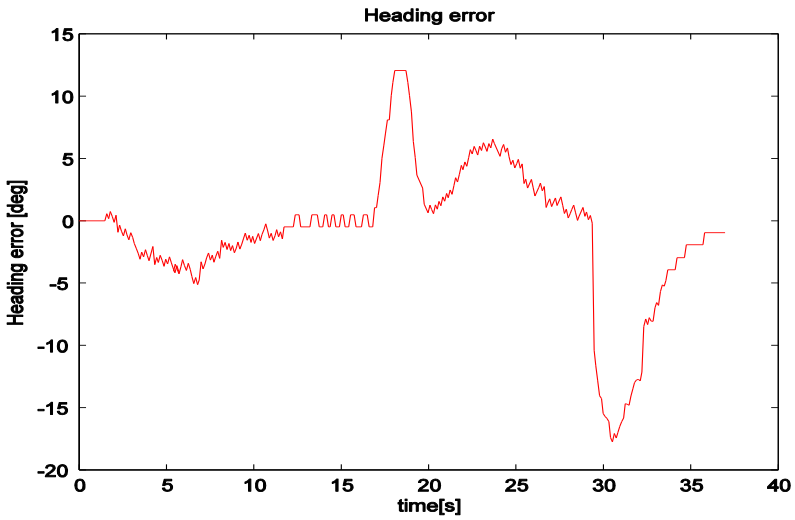

Fig. 5.55. Eroarea de urmărire a direcției obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot.

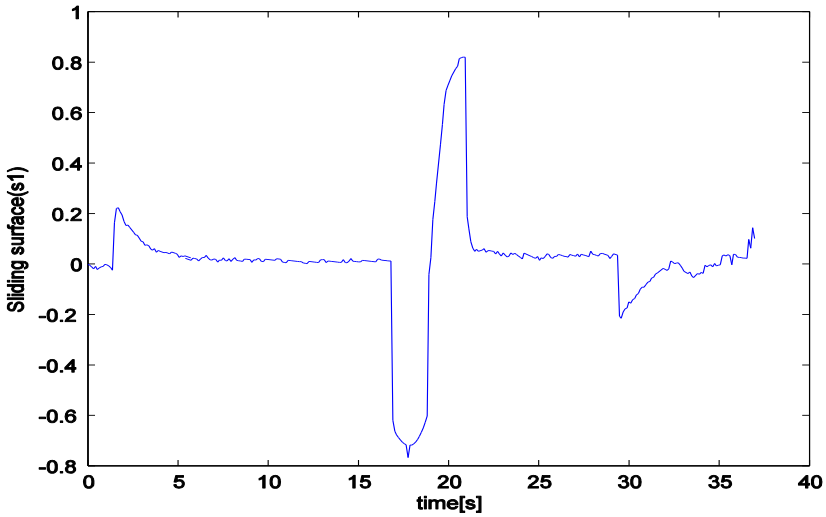

Fig. 5.56 Suprafața de comutație s1 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot.

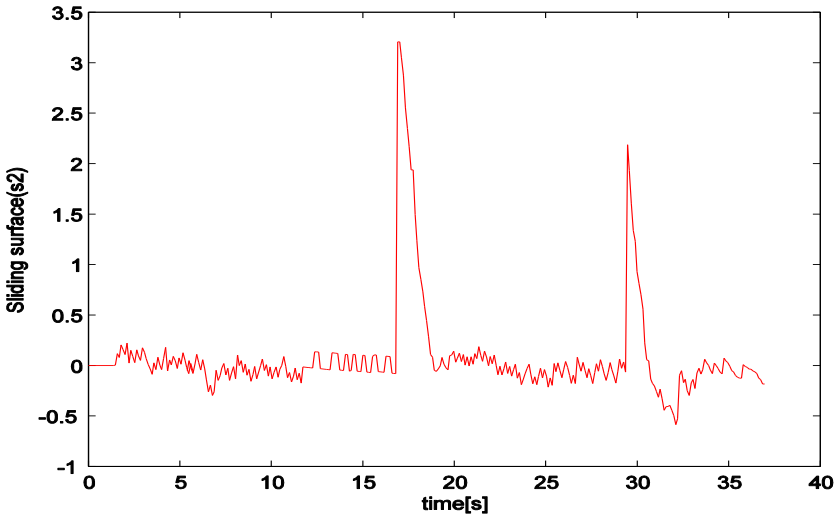

Fig. 5.57. Suprafața de comutație s2 obținută prin simularea în MobileSim a algoritmului de evitare a obstacolelor, în prezența a două obstacole, a robotului 2DW/2FW PowerBot.

Se observă că algoritmul de evitare a obstacolelor propus evită cele două obstacole și revine la traiectoria dorită. Evitarea obstacolelor introduce erori semnificative de urmărire și analizând graficele se poate determina începutul fiecărei proceduri de evitare a obstacolelor.

Simularea 5.8.

Se folosește arhitectura pentru evitarea obstacolelor din Fig. 5.58. Se consideră un mediu extern prezentat în harta din Fig. 5.59. Se dorește deplasarea vehiculului conform traiectoriei din Fig. 5.60. În acest caz se consideră că nu există obstacole în calea vehiculului. Parametrii constanți sunt $q_1 = 0.9$, $q_2 = 0.9$, $\varepsilon_1 = 0.02$, $\varepsilon_1 = 0.4$. Acești parametri au fost

obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor și sunt utilizați și la simularea 5.8. În acest caz modulul Trajectory Planner folosește planificatorul de traiectorii din [96] pentru generarea vitezei liniare și unghiului director. Modulul Odometry este implementat de softul ARIA și viteza liniară, viteza unghiulară, poziția și orientarea vehiculului sunt obținute utilizând funcțiile ARIA corespunzătoare. Conectarea la simulator este realizată de ARIA. Modulul de conducere este prezentat în Anexa 2. Blocul Range Readings este implementat în Anexa 5. Modulul Obstacle este implementat în Anexa 6. Modulul Obstacle Avoidance Planner este implementat în Anexa 8 și utilizează funcțiile din Anexa 7.

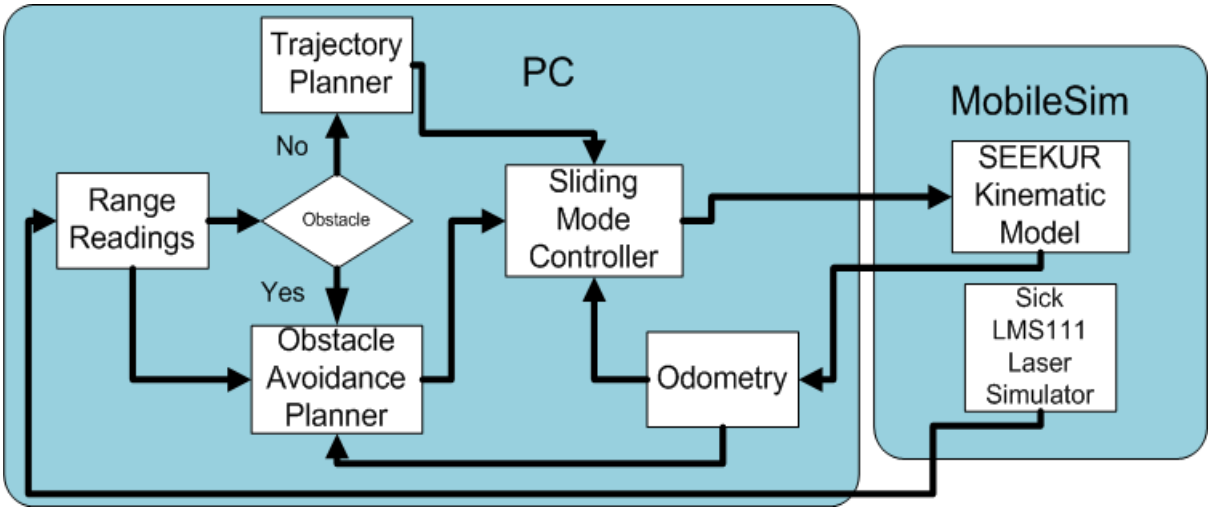

Fig. 5.58. Arhitectura pentru evitarea obstacolelor a AV 4DW/SW SEEKUR

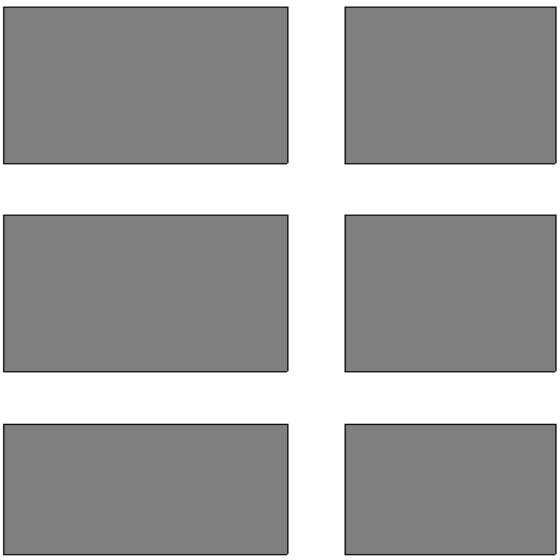

Fig. 5.59. Harta utilizată pentru testarea evitării obstacolelor utilizând vehiculul autonom 4DW/SW SEEKUR

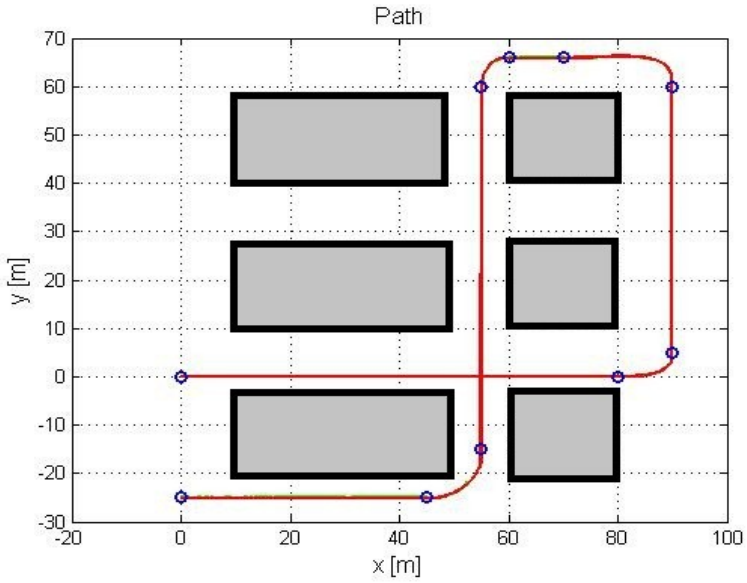

Fig. 5.60. Traiectoria generată de planificator pentru vehiculul autonom 4DW/SW SEEKUR

În Fig. 5.61 este prezentată traiectoria dorită și traiectoria simulată. Fig. 5.62 prezintă simularea în MobileSim. Fig. 5.63 prezintă eroarea de urmărire xe.

Fig. 5.64 prezintă eroarea de urmărire ye. În Fig. 5.65 este prezentată eroarea de urmărire a direcției.

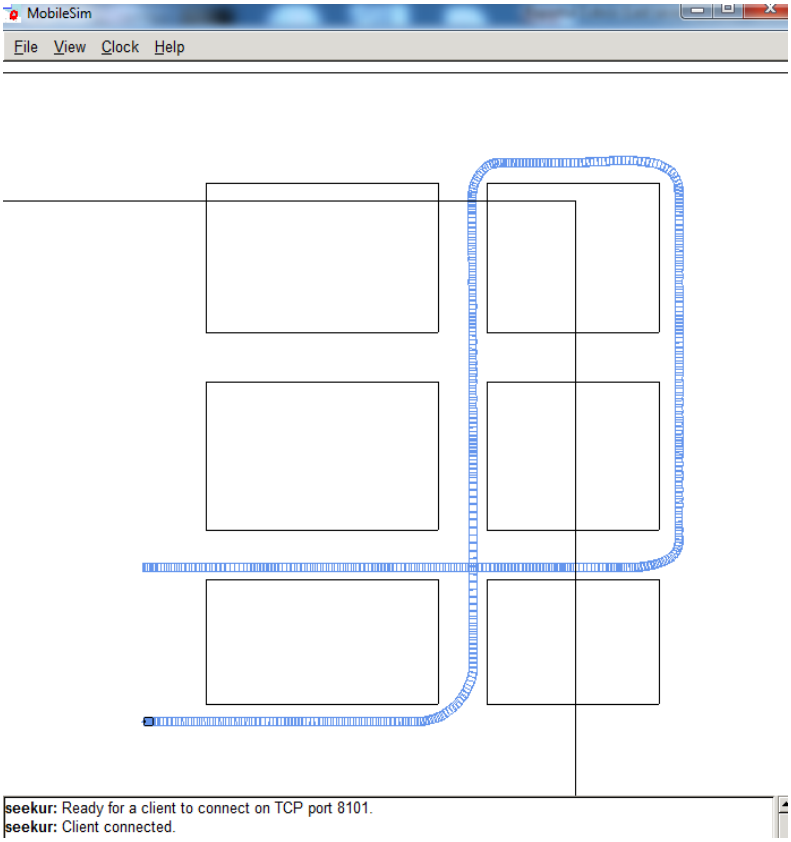

Fig. 5.61. Traiectoria obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor în cazul absenței obstacolelor a vehiculului autonom 4DW/SW SEEKUR

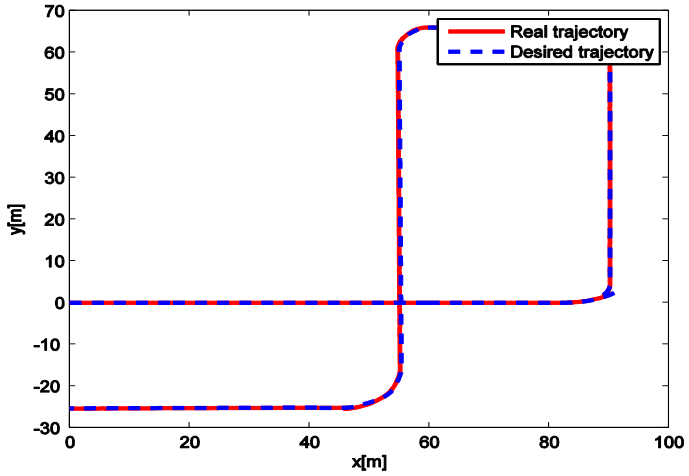

Fig. 5.62. Traiectoria obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul absenței obstacolelor, a vehiculului autonom 4DW/SW SEEKUR și traiectoria impusă

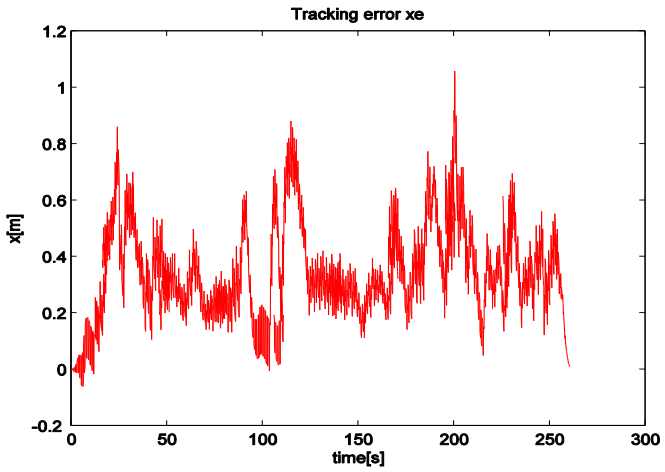

Fig. 5.63. Eroarea de urmărire pe axa X obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul absenței obstacolelor, a vehiculului autonom 4DW/SW SEEKUR.

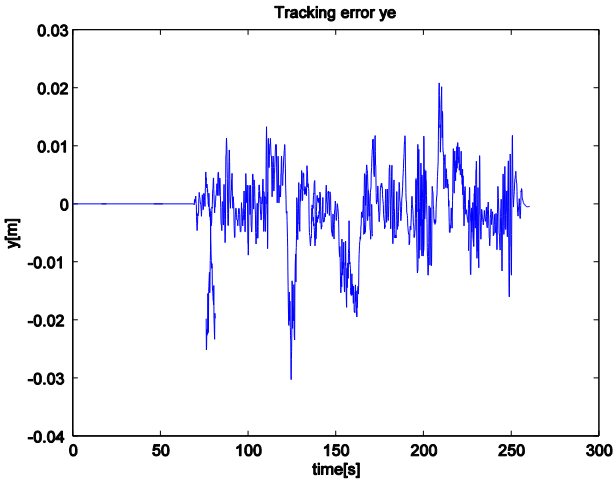

Fig. 5.64. Eroarea de urmărire pe axa Y obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul absenței obstacolelor, a vehiculului autonom 4DW/SW SEEKUR.

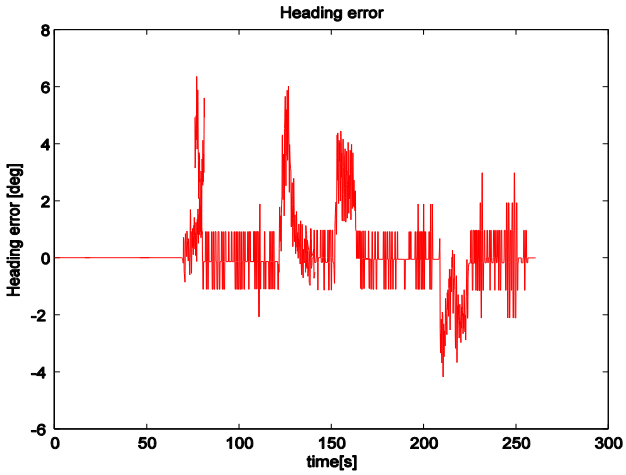

Fig. 5.65. Eroarea de urmărire a direcției obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul absenței obstacolelor, a vehiculului autonom 4DW/SW SEEKUR.

Simularea 5.9.

Se folosește harta din Fig. 5.59, dar în acest caz se introduce un obstacol în prima intersecție. Parametrii folosiți sunt identici cu cei din simularea precedentă. Vehiculul trebuie să ocolească acest obstacol și apoi să revină la traiectoria inițială. Fig. 5.66 prezintă simularea MobileSim a depășirii obstacolului. În Fig. 5.67 este prezentat traseul simulat.

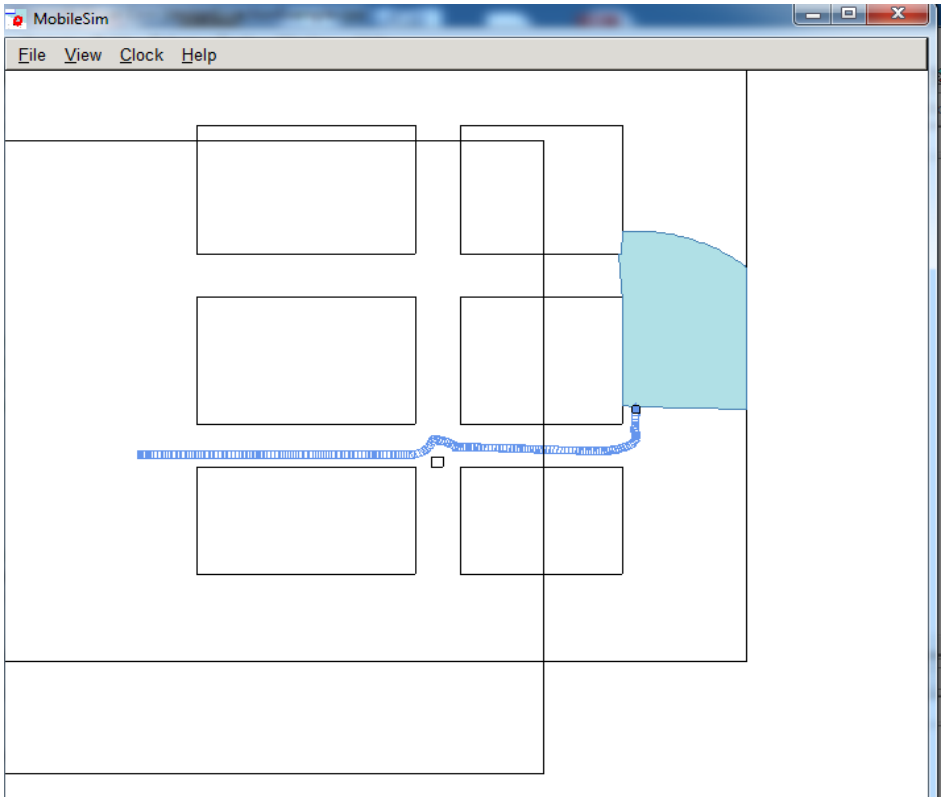

Fig. 5.66. Vehiculul autonom 4DW/SW SEEKUR simulând evitarea obstacolului în MobileSim

În Fig. 5.68este prezentată traiectoria dorită și traiectoria simulată. prezintă simularea în MobileSim. Fig. 5.69 prezintă eroarea de urmărire xe. Fig. 5.70. prezintă eroarea de urmărire ye. În Fig. 5.71 este prezentată eroarea de urmărire a direcției.

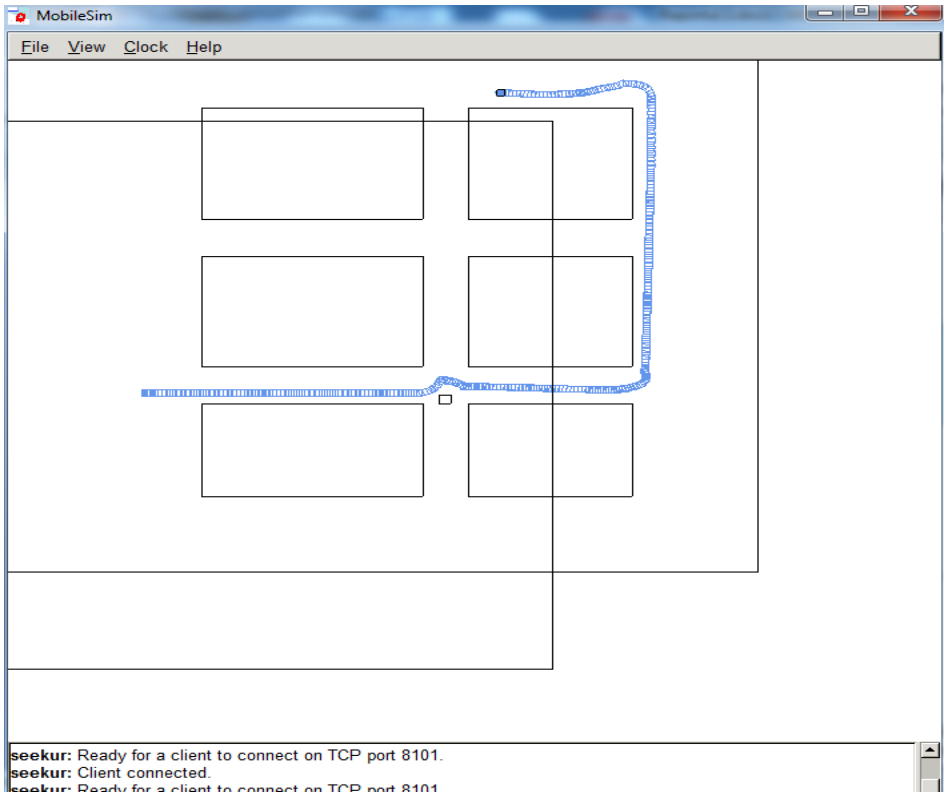

Fig. 5.67. Traiectoria obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul prezenței unui obstacol, a vehiculului autonom 4DW/SW SEEKUR

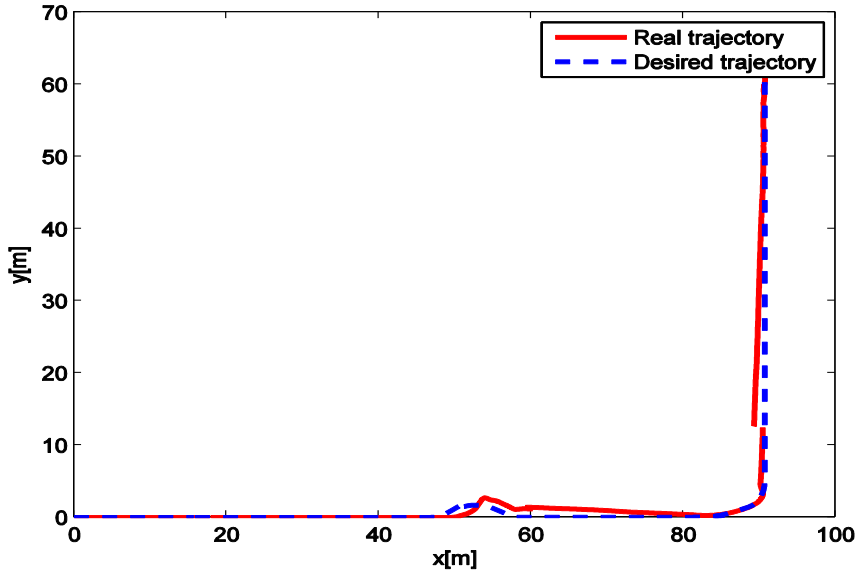

Fig. 5.68. Traiectoria obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul prezenței unui obstacol, a vehiculului autonom 4DW/SW SEEKUR și traiectoria dorită.

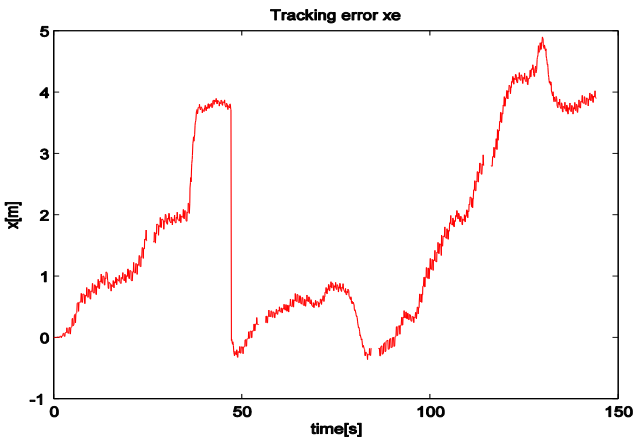

Fig. 5.69. Eroarea de urmărire pe axa X obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul prezenței unui obstacol, a vehiculului autonom 4DW/SW SEEKUR.

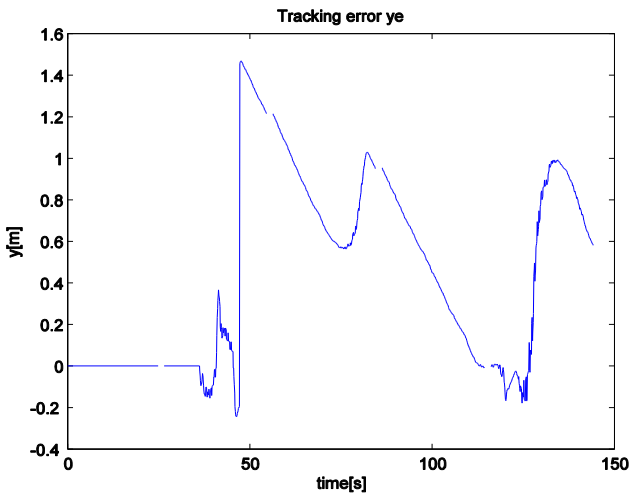

Fig. 5.70. Eroarea de urmărire pe axa Y obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul prezenței unui obstacol, a vehiculului autonom 4DW/SW SEEKUR.

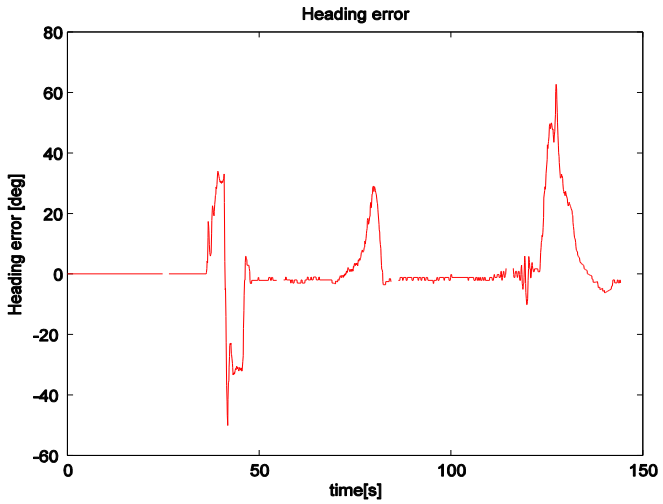

Fig. 5.71. Eroarea de urmărire a direcției obținută prin simularea în MobileSim a algoritmului pentru evitarea obstacolelor, în cazul prezenței unui obstacol, a vehiculului autonom 4DW/SW SEEKUR.

Se observă că vehiculul urmărește traiectoria chiar și în prezența obstacolului, dar prezența obstacolului introduce erori de urmărire semnificative. Se observă o întârziere între momentul în care este trimisă comanda pentru începerea evitaării obstacolului și începerea evitării de către vehicul.

5.2 Rezultatele experimentale ale conducerii în timp real ale roboților mobili și vehiculelor autonome

Experimente în timp real au fost efectuate pe robotul mobil PowerBot pentru a testa eficiența controlerului sliding-mode în timp discret. Programul, scris în C++, pentru conducerea robotului rulează pe un PC și trimite comenzi robotului și primește datele despre odometrie utilizând comenzile ARIA. Comunicarea dintre calculator și robot este gestionată de softul ARIA. Conectarea la calculator se realizează utilizând o rețea wireless. Pentru conectarea la robot se folosește un router wireless pentru conexiunea la rețea și pentru că robotul are o conexiune RS232 la router se conectează un Universal Data Server de la Lantronix pentru conversia din protocolul RS232 la protocolul Ethernet. Programul scris în C++ și compilat în Visual Studio este rulat utilizând CMD. Arhitectura conducerii sliding-mode în timp real a WMR 2DW/2FW PowerBot este prezentată în Fig. 5.72.

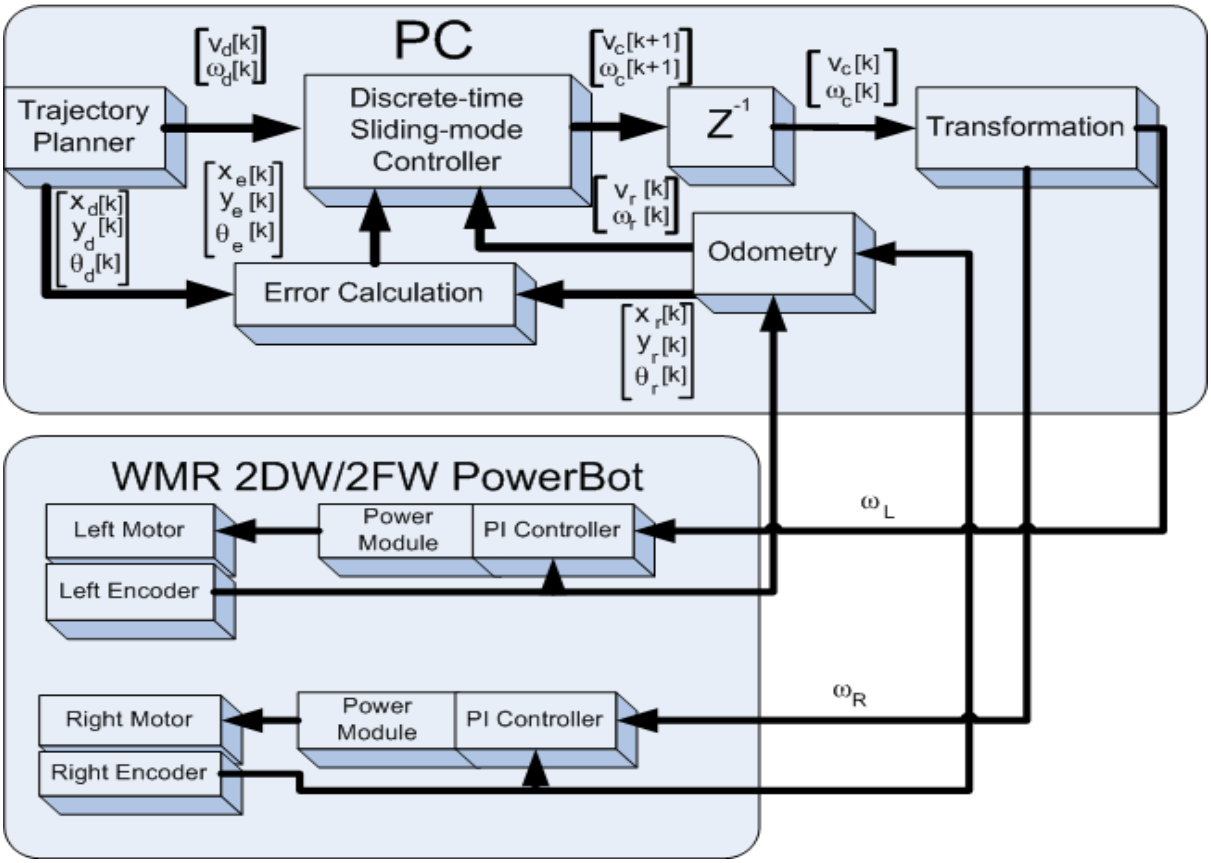

Fig. 5.72. Arhitectura conducerii sliding-mode în timp discret a WMR 2DW/2FW Powerbot în timp real

Modulul Odometry este implementat de softul ARIA, care interpretează datele primite de la robot pentru a calcula poziția, orientarea, viteza liniară, viteza unghiulară pe baza informației primite de la encoderele robotului. Pentru implementarea modulului de conducere se folosește programul din Anexa 2. Programul folosit la conducerea sliding-mode cu ecuațiile modificate conform secțiunii 3.3.1 Conducerea sliding-mode în timp discret a WMR cu 2 DW/2FW.

- Implementarea conducerii în timp real 1

Traiectoria urmărită a fost compusă dintr-o traiectorie liniară și o traiectorie circulară. Parametrii constanți sunt $q_1 = 0.9$, $q_2 = 0.85$, $\varepsilon_1 = 0.02$, $\varepsilon_1 = 0.5$. Acești parametri au fost identificați experimental. Fig. 5.73 este prezentată traiectoria reală cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră. În acest caz modulul Trajectory Planner constă în impunerea vitezei liniare de 0.3 m/s și viteza unghiulară de 0 radiani/s timp de 5s, apoi viteza unghiulară impusă devine 0.2 radiani/s.

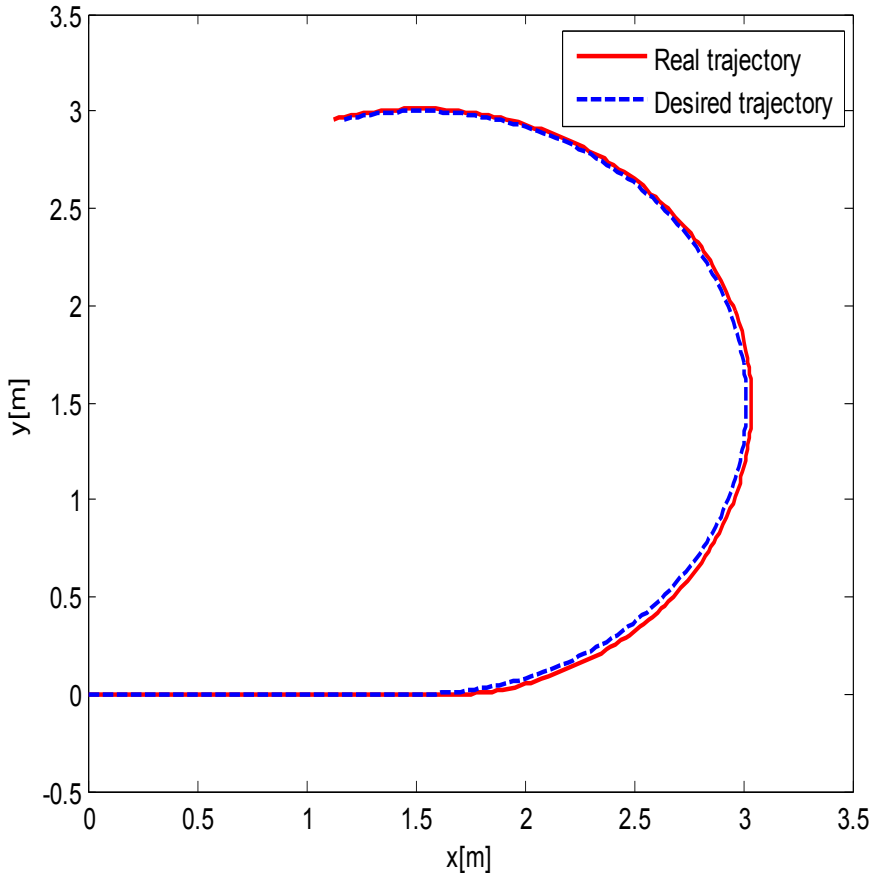

Fig. 5.73. Traiectoria obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot

Fig. 5.74 prezintă eroarea pe axa Ox. Fig. 5.75 prezintă eroarea pe axa Oy. Fig. 5.76 prezintă eroarea de orientare a robotului. În Fig. 5.77 este prezentată suprafața de alunecare s_1. În Fig. 5.78 este prezentată suprafața de alunecare s_2.

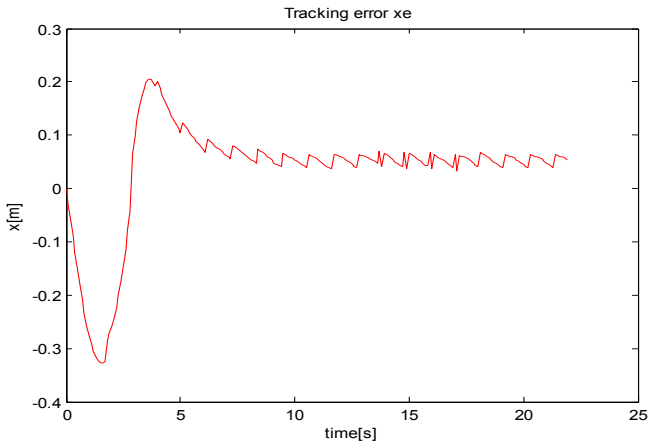

Fig. 5.74. Eroarea de urmărire pe axa X obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

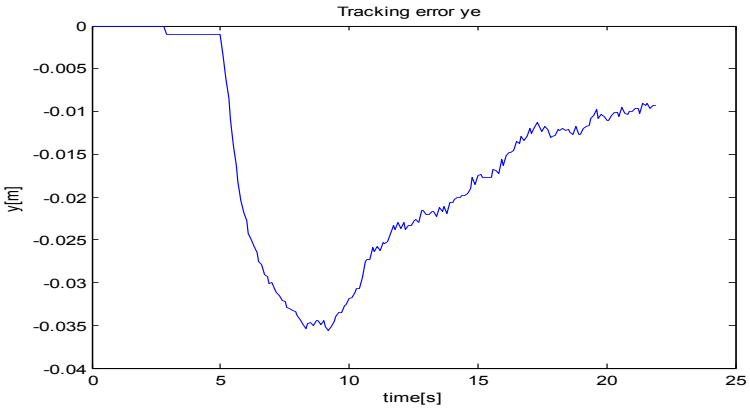

Fig. 5.75. Eroarea de urmărire pe axa Y obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

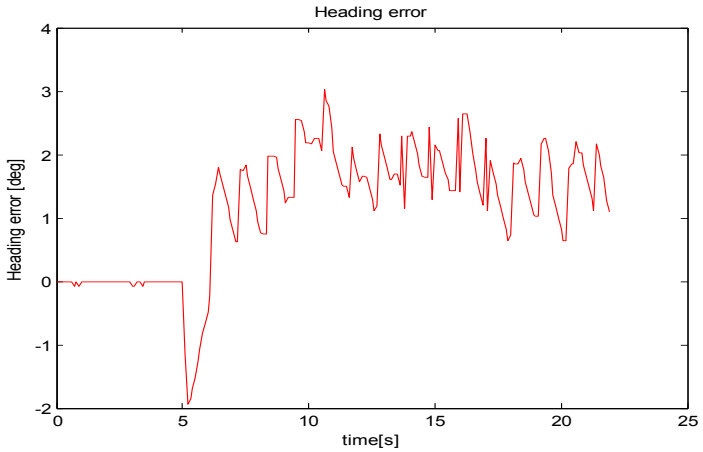

Fig. 5.76. Eroarea de urmărire a direcției obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

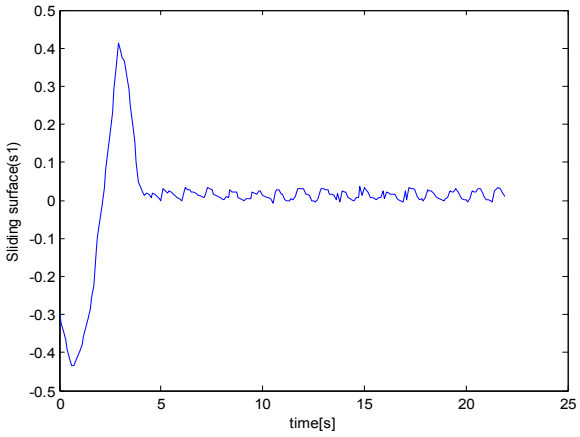

Fig. 5.77. Suprafața de alunecare s1 obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

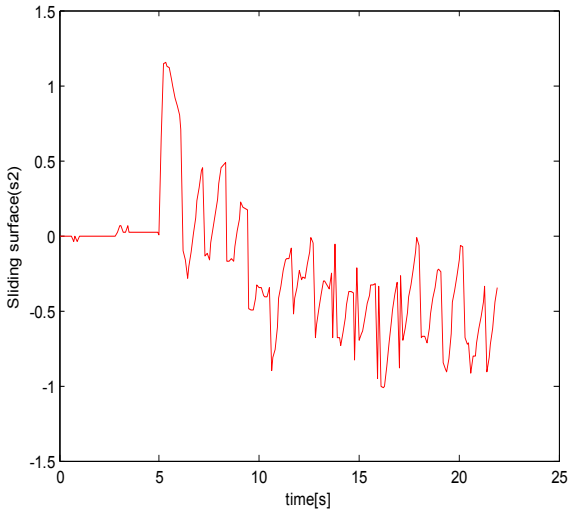

Fig. 5.78. Suprafața de alunecare s2 obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

Se observă o eroare inițială pe axa X la pornirea robotului care este redusă după aproximativ 5s.Erorile de urmărire obținute sunt mici și dovedesc eficiența conducerii propuse.

- Implementarea conducerii în timp real 2.

Robotul PowerBot a urmat o traiectorie în formă de S, prezentată în Fig. 5.79. Parametrii constanți sunt $q_1 = 0.9$, $q_2 = 0.85$, $\varepsilon_1 = 0.02$, $\varepsilon_1 = 0.5$. Acești parametri au fost identificați experimental. Modulul Trajectory planner constă în impunerea unei viteze liniare de 0.3 m/s și a unei viteze unghiulare de 0.2 radiani/s timp de 12.5 s și apoi impunerea vitezei unghiulare de -0.2 radiani/s timp de 12.5 s.

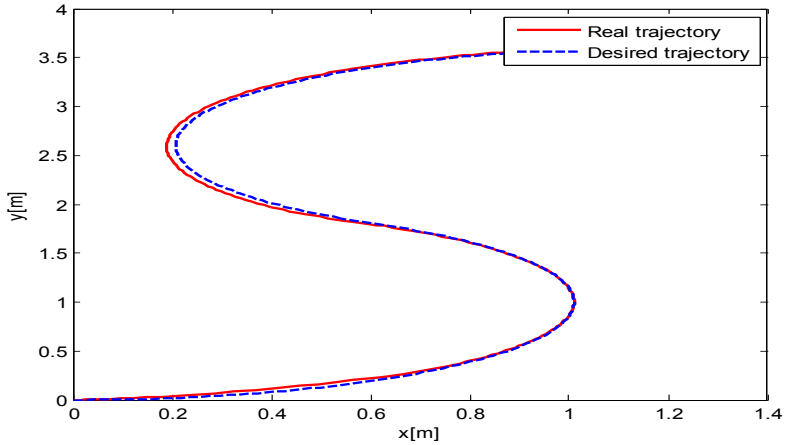

Fig. 5.79. Traiectoria în formă de S obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/g2FW PowerBot.

Fig. 5.79 prezintă traiectoria reală și traiectoria dorită. Fig. 5.80 prezintă eroarea de urmărire pe axa X. Fig. 5.81 prezintă eroarea de urmărire a direcției

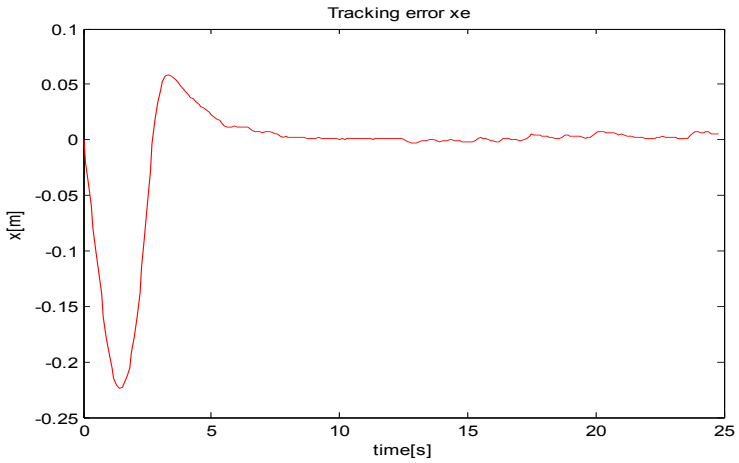

Fig. 5.80. Eroarea de urmărire pe axa X a traiectoriei în formă de S obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

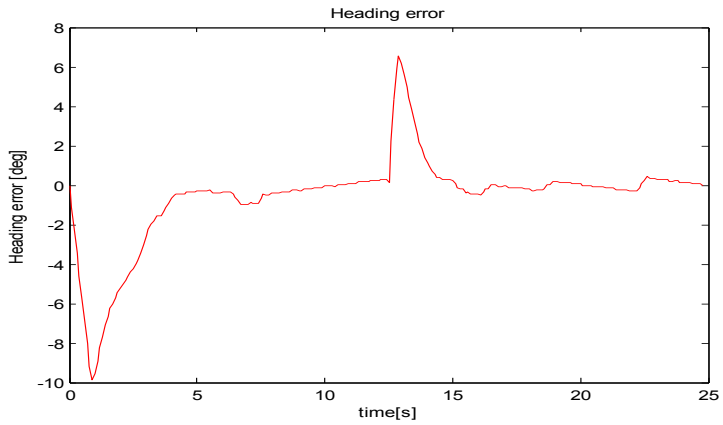

Fig. 5.81. Eroarea de urmărire a orientării în cazul traiectoriei în formă de S obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot.

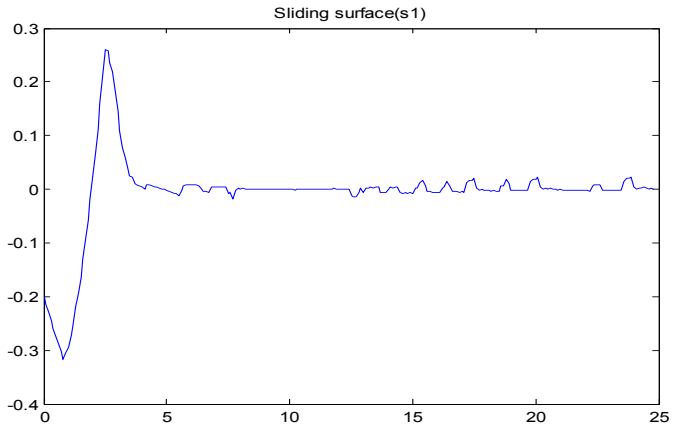

Fig. 5.82. Suprafața de alunecare s1 obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot pentru urmărirea traiectoriei în formă de S.

Fig. 5.82 prezintă suprafața de comutație s1 Fig. 5.83 prezintă suprafața de comutație s2.

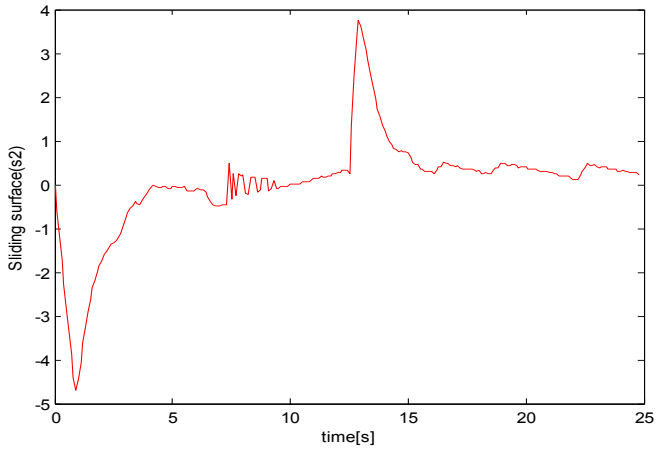

Fig. 5.83. Suprafața de comutație s2 obținută la conducerea în timp real prin metoda sliding-mode în timp discret a robotului 2DW/2FW PowerBot pentru urmărirea traiectoriei în formă de S.

Din grafice reiese o foarte bună urmărire a traiectoriei propuse, chiar dacă eroarea de urmărire pe axa X este mare la începutul testului, după aproximativ 5s este redusă aproape la zero. Se observă că cele două suprafețe de alunecare oscilează în jurul valorii zero.

5.3 Concluzii

Pentru a valida algoritmii propuși am prezentat 9 simulări realizate în MobileSim și 2 implementări în timp real.Am scris programele pentru testarea algoritmilor în C++ și i-am compilat utilizând Visual Studio. Pentru realizarea simulărilor am folosit softul MobileSim de la Mobile Robots. MobileSim simulează comportamentul fiecărui robot produs de Mobile

Robots şi detecţia laser sau utilizând sonarele a obstacolelor. Am creat obstacolele simulate în MobileSim în Mapper3Basic. Mapper3Basic este un utilitar creat pentru a genera hărţi ale mediilor dorite pentru a putea fi simulate în MobileSim.

Pentru simularea 5.1 am impus urmărirea de către robotul PatrolBot a unei traiectorii în formă de S utilizând conducerea sliding-mode în timp discret. La simularea 5.2 am dorit urmărirea unei traiectorii liniare, urmate de una circulară de către vehiculul SEEKUR, utilizând conducerea sliding-mode în timp continuu. Simularea 5.3 a presupus urmărirea unei traiectorii în formă de S de către vehiculul SEEKUR, utilizând conducerea sliding-mode în timp discret. În simularea 5.4 am testat urmărirea unei traiectorii generate de un planificator de traiectorii, folosind metoda backstepping şi robotul PowerBot. În simularea 5.5 am realizat urmărirea unei traiectorii de către PowerBot în mediul definit fără obstacole. Următoarele două simulări au testat dacă conducerea mai este utilizabilă în prezenţa unor obstacole. Simularea 5.8 a testat conducerea vehiculului SEEKUR utilizând sliding-mode în timp discret şi o hartă care simulează o zona a unui oraş.În cadrul simulării 5.9 am introdus un obstacol pentru a testa dacă este posibilă evitarea sa.

La implementarea conducerii în timp real 1 am realizat urmărirea unei traiectorii liniare urmate de una circulară de către PowerBot, folosind conducerea sliding-mode în timp-discret, iar în cadrulimplementării conducerii în timp real 2 am testat urmărirea unei traiectorii în formă de S.

Rezultatele testelor au arătat eficienţa metodelor propuse atât pentru urmărirea unei traiectorii utilizând cele trei metode de conducere: sliding-mode în timp continuu, sliding-mode în timp discret şi backstepping, cât şi pentru algoritmul de evitare a obstacolelor. Am constatat că urmărirea traiectoriei în absenţa obstacolelor are erori mai bune, iar prezenţa unui obstacol introduce erori de urmărire suplimentare când se revine la traiectoria iniţială.

Capitolul 6

6. Concluzii

În această lucrare am tratat problema conducerii și evitarii de obstacole utilizând roboți mobili și vehicule autonome. Pentru o mai buna calculare a odometriei am analizat senzorul IMU Xsens Mti de la Xsens și am proiectat un filtru Kalman pentru a elimina erorile apărute în calculul atitudinii furnizată de IMU datorită acumularii erorilor de integrare a semnalului afectat de zgomot. Cercetarea senzorilor IMU a fost realizată cu suportul Institutului de Sisteme și Robotică , din cadrul Universității din Coimbra, sub supervizarea profesorului Urbano Nunes. Am determinat modelele cinematice ale roboților mobili și vehiculelor autonome. Am folosit aceste modele pentru a sintetiza comenzile necesare pentru urmărirea traiectoriilor.

Am folosit trei metode de conducere diferite pentru a realiza conducerea roboților mobili și a vehiculelor autonome: conducerea sliding-mode în timp continuu, conducerea sliding-mode în timp discret și conducerea backsteping. Pentru fiecare tip de conducere am calculat comenzile specifice pentru roboții mobili și vehiculele autonome. Metodele de conducere obținute pot fi implementate pe toți roboții din laboratorul de robotică.

Am propus o metodă de evitare a obstacolelor care utilizează sonarele în cazul WMR și senzori laser în cazul AV. Am considerat cazul în care se dorește urmărirea traiectoriei impuse și abatera doar pentru a evita un obstacol și întoarcerea la traiectoria inițială. Am folosit un generator de traiectorii online care stabilește o nouă traiectorie de urmărit și care este impusă modulului de conducere, fără am fi nevoie de elaborarea unui nou modul de conducere pentru a evita obstacolul. Metoda propusă permite utilizarea oricărei metode de conducere propuse în această lucrare.

Am realizat simulări și implementări în timp real pentru a valida algoritmii propuși în această lucrare.

6.1. Contribuții privind conducerea și evitarea obstacolelor utilizând roboți mobili și vehicule autonome

Principalele contribuții gravitează în jurul domeniilor:

- Estimarea atitudinii
- Conducerea trajectory-tracking a roboților mobili și vehiculelor autonome
- Evitarea obstacolelor
- Testarea algoritmilor propuși

6.1.1 Estimarea atitudinii

Am pornit de la nevoia de a avea date corecte despre orientarea vehiculului și faptul că IMU poate oferi date despre orientare dacă se filtrează erorile. Pentru că am constatat erori semnificative la integrarea semnalelor de la senzori pentru obținerea atitudinii am analizat senzorul IMU Xsens Mti pentru a determina erorile caracteristice. Am analizat senzorul utilizând metoda Allan. Am ales quaternionii ca mod de reprezentare a atitudinii pentru că nu prezintă singularitatea specifică reprezentării cu ajutorul unghiurilor Euler, dar pentru validarea rezultatelor, am trecut de la reprezentarea atitudinii în quaternioni la reprezentarea Euler pentru o mai bună întelegere a rezultatelor. Utilizând caracteristica rezultată am elaborat un filtru Kalman indirect în două etape pentru a elimina erorile care se acumulează la calculul orientării. Am constatat că este necesară folosirea mai multor senzori pentru o estimare corectă a atitudinii. În acest caz am folosit datele de la accelerometre și senzorii magnetici pentru a obține corecțiile necesare. Filtrul folosește datele de la giroscoape pentru a calcula atitudinea estimată apoi aplică corecții în două etape. În prima etapă sunt aplicate corecții rezultate de la accelerometre, starea eroare rezultată este apoi folosită la corecția atitudinii estimate. Etapa a doua aduce corecțiile magnetice, după actualizarea stării eroare se corectează estimarea atitudinii cu noua valoare obținută.

6.1.2 Conducerea trajectory-tracking a roboților mobili și vehiculelor autonome

Am tratat problema urmăririi traiectoriei în capitolul 3. Am analizat trei metode de conducere pentru conducerea vehiculelor și roboților mobili: conducerea sliding-mode în timp continuu, conducerea sliding-mode în timp discret și conducerea backstepping.

Am propus rezolvarea problemei conducerii roboților mobili și vehiculelor autonome utilizând conducerea sliding-mode în timp continuu, conducerea sliding-mode în timp discret și conducerea backstepping bazate pentru WMR și AV.

De la cazul general și folosind modelul cinematic al WMR cu 2 DW/2SW , modelul erorilor de urmărire și dinamica erorilor de urmărire am calculat comenzile pentru viteza liniară și viteza unghiulară a WMR. În calcul am folosit suprafețe de comutație și legi de conducere bazate pe modelul propus de Gao pentru timp continuu[31].

Utilizând cazul general și modelul cinematic simplificat al vehiculului autonom SEEKUR, suprafețe de comutație, legi de conducere similare celor propuse de Gao pentru

timp continuu[31], modelul erorilor de urmărire și dinamica erorilor de urmărire am calculat comenzile pentru viteza liniară și unghiul de direcție.

Am sintetizat modul de calcul al comenzii utilizând sliding-mode în timp discret pentru cazul general.Utilizând cazul general am trecut la calculul comenzilor pentru WMR și AV.

Pornind de la modelul cinematic al WMR în timp discret, erorile de urmărire în timp discret , dinamica erorilor de urmărire în timp discret, suprafețele sliding pentru timp discret și legea de conducere pentru timp discret am calculat comenzile pentru viteza liniară și viteza unghiulară a WMR. Conducerea în timp discret a WMR și AV afost prezentată în publicațiile: [16], [17], [18], [18], [20], [21], [24] și [93].

Utilizând modelul cinematic simplificat al vehiculului autonomon în timp discret, erorile de urmărire în timp discret , dinamica erorilor de urmărire în timp discret, suprafețele sliding pentru timp discret și legea de conducere pentru timp discret am calculat comenzile pentru viteza liniară și unghiul de direcție a vehiculului autonom.

Am prezenat cazul general al conducerii backstepping. În continuare am calculat comanda pentru conducerea backstepping în cazul WMR utilizând modelul cinematic al robotului și dinamica erorilor de urmărire a robotului. Conducerea backstepping a WMR a fost publicată în [19].

Am calculat comenzile pentru viteza liniară și unghiul de direcție al vehiculului autonom SEEKUR utilizând modelul cinematic și dinamica de urmărire a erorilor și procedura de conducere backstepping.

6.1.3 Evitarea obstacolelor

O soluție pentru problema evitării obstacolelor în cazul în care se dorește urmărirea unei traiectorii impuse și abaterea de la această traiectorie doar pentru a evita un obstacol și întoarcerea la traiectoria impusă este propusă. Am considerat cazul în care robotul este echipat cu sonare și cazul în care vehiculul este echipat cu senzori laser.

Detecția obstacolelor folosește o bulă de sensibilitate pentru a determina dacă un obstacol trebuie evitat. Atât sonarele cât și laserele au permis definirea unei bule de sensibilitate care acoperă $180°$. Este folosită aceași metodă de conducere în cazul în care nu există obstacole și în cazul în care un obstacol este prezent. Cât timp nu este detectat nici un obstacol care blochează traiectoria dorită platforma urmărește traiectoria impusă folosind metoda de conducere aleasă din cele 3 prezentate în capitolul anterior.

Dacă un obstacol a fost detectat se caută coordonatele care să permită conducerea în siguranță a platformei și se generează o traiectorie bazată pe Quintic equations. Nu se dorește identificarea formei obstacolului ci doar o zonă în care acesta poate fi evitat. Această traiectorie este impusă modulului de conducere. Se continuă deplasarea pe această traiectorie până când obstacolul a fost evitat. După ce obstacolul a fost evitat se calculează traiectoria pentru a reveni la traiectoria impusă și modul de conducere urmărește această traiectorie apoi se revine la traiectoria impusă inițială.

6.1.4 Testarea algoritmilor propuși

Problema conducerii prezentată în capitolul 3 și problema evitării obstacolelor prezentată în capitolul 4 sunt testate în capitolul 5 cu ajutorul simularilor și testelor în timp real. Au fost realizate 9 simulări și 2 teste pentru a verifica algoritmii propuși.

Am realizat simulările în MobileSim, un simulator oferit de Mobile Robots care permite simularea comportamentului tuturor roboților comercializați de Mobile Robots și a senzorilor laser sau a sonarelor. MobileSim primește comenzile generate de ARIA, iar utilizatorul trebuie să folosească funcțiile implementate în ARIA pentru conducerea robotului.

Pentru a simula evitarea obstacolelor a fost nevoie să creez hărți în care să fie simulate obstacolele dorite. Toate hărțile folosite au fost create în Mapper3Basic. Acest program este oferit de Mobile Robots pentru a crea hărți ale mediilor în care se doresc să fie simulați roboții. Hărțile generate în Mapper3Basic sunt compatibile cu MobileSim și sunt încărcate în simulator înaintea simulărilor.

În cazul testelor în timp real am rulat programul pe un PC cu frecvența de eșantionare de 100ms și am folosit o rețea wireless pentru a comunica cu robotul. Pentru că robotul nu are o conexiune Ethernet a trebuit să folosesc un Universal Data Server de la Lantronix care a realizat conversia de la comunicația RS232 a robotului și comunicația Ethernet a routerului cu ajutorul căruia am implementat rețeaua wireless.

Testarea conducerii sliding-mode în timp continuu a fost realizată în simularea 5.1 utilizând vehiculului SEEKUR și o traiectorie liniară urmată de una circulară.

Testarea conducerii sliding-mode în timp discret a fost realizată în simularea 5.2, simularea 5.3, implementarea conducerii în timp real 1 și implementarea conducerii în timp real 2. Simularea 1 a realizat urmărirea unei traiectorii în formă de S de către robotul PatrolBot. Simularea 5.3 a realizat urmărirea unei traiectorii în formă de S utilizând vehiculul

SEEKUR. Conducerea robotului PowerBot a fost testată utilizând 2 implementări în timp real. Primul test în timp real a realizat urmărirea unei traiectorii liniare urmate de una circulara, iar al doilea test în timp real a realizat urmărirea unei traiectorii în formă de S.

Simulările și testele efectuate au dovedit o bună urmărire a traiectoriei dorite, iar erorile au fost mici, astfel controllerele fiind validate.

Evitarea obstacolelor a fost testată în simulările 5.5, 5.6, 5.7, 5.8, 5.9. Evitarea obstacolelor utilizând robotul PowerBot a fost testată în simulările 5.5, 5.6, 5.7. Inițial în simularea 5.5 a fost testat algoritmul în lipsa obstacolelor, iar în simularea următoare a fost adăugat un obstacol, iar în simularea 5.7 au fost folosite 2 obstacole. Evitarea obstacolelor folosind detecția laser și vehiculul SEEKUR a fost testată în simulările 5.8 și 5.9. Inițial a fost testat algoritmul fără a exista obstacole, iar în ultima simulare a fost adăugat un obstacol. Simulările în lipsa obstacolelor au dovedit performanțe similare cazului în care nu au fost folosiți senzori pentru detectarea obstacolelor. În prezența obstacolelor erorile de urmărire după evitarea obstacolelor au crescut și se pot observa în grafice erori semnificative în momentul începerii procedurii de evitare a obstacolelor și la teminarea acesteia, dar conducerea realizează traiectoria dorită.

Din testele efectuate reiese că atât metodele de conducere cât și de evitare a obstacolelor oferă rezultate bune.

6.1.5 Direcții de cercetare deschise

O direcție de cercetare poate fi implementarea în timp real a algorimului de evitare a obstacolelor utilizând vehiculul autonom SEEKUR.

O altă direcție de cercetare poate fi conducerea vehiculelor autonome utilizând atitudinea furnizată de IMU, în timp real și dezvoltarea unui filtru Kalman care să furnizeze și odometria vehiculului, corectată cu date de la encodere sau GPS.

6.1.6 Diseminarea rezultatelor

Rezultatele cercetării au fost diseminate în 8 lucrări prezentate la conferințe internaționale și o lucrare publicată în Annals of the University of Coimbra. Din cele 9 lucrari sunt primul autor la 7 lucrări și coautor la 2 lucrări.

[1] Dumitrascu B. and Filipescu A., Discrete-time sliding-mode controller for wheeled mobile robots, Proceedings of *the 18th International Conference on Control Systems and Computer Science, vol*. 1, Bucuresti, pag. 397-403, mai, 2011.

Această lucrare propune un controller sliding-mode în timp discret pentru rezolvarea problemei urmăririi traiectoriei robotului PowerBot. Controllerul se bazează pe modelul cinematic calculat în capitolul 3 și este descris în capitolul 3. Teste pentru dovedirea eficienței conducerii sunt prezentate în capitolul 5.

[2] Dumitrascu B. and Filipescu A., Sliding mode control of lateral motion for four driving-steering wheels autonomous vehicle, Annals of *the University of Craiova, vol*. 7 (34), pag 20-25, 2010.

Această lucrare este bazată pe lucrarea [18] prezintă un controller sliding-mode în timp continuu pentru conducerea laterală a vehiculului autonom SEEKUR, controolerul a fost prezentat în capitolul 3 și folosește modelul cinematic din capitolul 3.

[3] Dumitrascu B. and Filipescu A., Sliding mode controller for steering of four-wheels driving and steering vehicle, Proceedings of *the 14th International Conference on System Theory and Control, pag*. 202-206, Sinaia, 2010.

Această lucrare prezintă un controller sliding-mode în timp continuu pentru conducerea laterală a vehiculului autonom SEEKUR, controolerul a fost prezentat în capitolul 3, și folosește modelul cinematic din capitolul 3.

[4] Dumitrascu B., Filipescu A., Backstepping control of wheeled mobile robots, Proceedings of *the 15th International Conference on System Theory and Control, Sinaia*, pag. 206-211, 2011.

În această lucrare a fost propusă conducerea backstepping a robotului mobil PowerBot, care a fost prezentată în capitolul 3 și testată în capitolul 5. Conducerea este calculată pornind de la modelul cinematic al robotului prezentat în capitolul 3.

[5] Dumitrascu B., Filipescu A., Minzu V., Voda A., Minca E.,Discrete-Time Sliding-Mode Control of Four Driving-Steering Wheels Autonomous Vehicle, Proceedings of *the 30th Chinesse Control Conference, pag.* 3620-3625, 2011.

Lucrarea prezintă conducerea sliding-mode în timp discret a vehiculului autonom SEEUR, din capitolul 3 al acestei teze. Conducerea folosește modelul cinematic simplificat al vehiculelor 4DW/SW și este testat în capitolul 5.

[6]] Dumitrascu B., Filipescu A., Radaschin A., Filipescu A. Jr., Minca E., Discrete-time sliding mode control of wheeled mobile *robots, Proceedings of 2011 8th Asian Control Conference* (ASCC) ,Kaohiung, Taiwan, pag. 771-776, mai , **2011.**

Această lucrare propune un controller sliding-mode în timp discret pentru rezolvarea problemei urmăririi traiectoriei robotului PowerBot. Controllerul se bazează pe modelul

cinematic calculat în capitolul 3 și este descris în capitolul 3. Teste pentru dovedirea eficienței conducerii sunt prezentate în capitolul 5.

[7] Dumitrascu B., Filipescu A., Vasilache C., Minca E., Filipescu A. Jr., Discrete-time sliding-mode control of four driving/steering wheels mobile platform, Proceedings of *the 19th* Mediterranean Conference on Control and Automation, Corfu, Grecia, iunie, 2011, pag. 1076-1081.

Lucrarea propune metoda de conducere sliding-mode în timp discret pentru problema urmăririi traiectoriei vehiculului autonom cu 4 DW/SW, SEEKUR. Conducerea se bazează pe modelul cinematic calculat în capitolul 3 și este descris în capitolul 3. Teste pentru dovedirea eficienței conducerii sunt prezentate în capitolul 5.

[8]]Filipescu A., Minzu V., Dumitrascu B. and Filipescu A., Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots, The 2011 IEEE International Conference on Information and Automation, iunie, **2011.**

Această lucrare propune un metoda de conducere sliding-mode în timp discret pentru rezolvarea problemei urmăririi traiectoriei robotului PowerBot. Conducerea se bazează pe modelul cinematic calculat în capitolul 3 și este descris în capitolul 3. Teste pentru dovedirea eficienței conducerii sunt prezentate în capitolul 5.

[9] Solea R., Filipescu A., Filipescu S, and Dumitrascu B., Sliding-mode controller for four- wheel-steering vehicle: trajectory-tracking problem, Proceedings of *the 8th World Congress on Intelligent Control and Automation, Jinan*, China, pag. 1185-1190, 2010.

Această lucrare propune metoda de conducere sliding-mode în timp continuu pentru rezolvarea problemei urmăririi traiectoriei vehiculului autonom SEEKUR Conducerea se bazează pe modelul cinematic calculat în capitolul 3 și este descris în capitolul 3. Teste pentru dovedirea eficienței conducerii sunt prezentate în capitolul 5.

7. Bibliografie

[1] Andrade Da Silva, Teppa Garrán J. M. and Suárez P.A., Sliding mode fuzzy gain scheduling in sampled data nonlinear systems, *2005 American Control Conference*, Portland, OR, USA, pag. 4965-4970, iunie, 2005.

[2] Arambula Cosio F. and Padilla Castaneda M.A., Autonomous robot navigation using adaptive potential fields, *Mathematical and Computer Modelling*, vol. 40, pag. 1141-1156, 2004.

[3] Arthur Gelb, Joseph F. Kasper, Raymond A. Nash, Charles F. Price, Arthur A. Sutherland, Applied optimal estimation, The M.I.T. Pess 2001.

[4] Bakker T., Kees van Asselt, Jan Bontsema, Müller J., Gerrit van Straten, A path following algorithm for mobile robots, *21 April 2010 This article is published with open access at Springerlink.com.*

[5] Borenstein J., Koren Y., Real-time obstacle avoidance for fast mobile robots, IEEE *Transactions on Systems, Man, and Cybernetics,* vol. 19, nr. 5, pag. 1179-1187, 1989.

[6] Borenstein J., Koren Y., The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Journal of Robotics and Automation, pag. 278-288, vol. 7, 1991.

[7] Brockett R., Control theory and singular Riemannian geometry, *New Directions in Applied Mathematics, Springer-Verlag*, pag. 11-27 ,1981.

[8] Carlo L. Bottasso, Leonello D., and Savini B., Path planning for autonomous vehicles by trajectory smoothing using motion primitives, *Paper presented at the AHS International Specialists' Meeting on Unmanned Rotorcraft,* Phoenix, AZ, USA, January 23–25, 2007.

[9] Chen Chieh, Backstepping control design and its applications to vehicle lateral control in automated highway systems, University of California at Berkeley, 1996.

[10] Choukroun D., Novel methods for attitude determination using vector observations for attitude determination, PhD Thesis, Haifa, Mai, 2003.

[11] Chu-qing C., Lian-zheng G. and Rui-feng L., Mobile robots target tracking using finite-time convergence sliding mode controller, *2010 8th IEEE International Conference on Control and Automation*, Xiamen, China, June 9-11, pag. 460-464, 2010.

[12] Chwa D., Seo J. H., Kim P., and Choi J. Y., Sliding mode tracking control of nonholonomic wheeled mobile robots, *Proceedings of the American Control Conference,* Anchorage, pag 3991-3996, mai, 2002.

[13] Chwa D., Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates, *IEEE Transactions on Control Systems Technology*, vol. 12, nr. 4, iulie, pag. 637-644, 2004.

[14] Corradini M. L., Leo T., Orlando G., Variable structure control of robotic asistance system, *Proceedings of the 5th IEEE Mediterranean Conference on Control and Systems*, http://med.ee.nd.edu/MED5/PAPERS/052/052.PDF.

[15] Danwei W. and Feng Qi, Trajectory planning for a four-wheel steering vehicle, *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, pag. 3320-3325, 2001.

[16] Dumitrascu B. and Filipescu A., Discrete-time sliding-mode controller for wheeled mobile robots, *Proceedings of the 18th International Conference on Control Systems and Computer Science*, vol. 1, Bucuresti, pag. 397-403, mai, 2011.

[17] Dumitrascu B. and Filipescu A., Sliding mode control of lateral motion for four driving-steering wheels autonomous vehicle, *Annals of the University of Craiova*, vol. 7 (34), pag 20-25, 2010.

[18] Dumitrascu B. and Filipescu A., Sliding mode controller for steering of four-wheels driving and steering vehicle, *Proceedings of the 14th International Conference on System Theory and Control*, pag. 202-206, Sinaia, 2010.

[19] Dumitrascu B., Filipescu A., Backstepping control of wheeled mobile robots, *Proceedings of the 15th International Conference on System Theory and Control*, Sinaia, pag. 206-211, 2011.

[20] Dumitrascu B., Filipescu A., Radaschin A., Filipescu A. Jr., Minca E., Discrete-time sliding mode control of wheeled mobile robots, *Proceedings of 2011 8th Asian Control Conference (ASCC)* ,Kaohiung, Taiwan, pag. 771-776, mai , 2011.

[21] Dumitrascu B., Filipescu A., Vasilache C., Minca E., Filipescu A. Jr., Discrete-time sliding-mode control of four driving/steering wheels mobile platform, *Proceedings of the* 19th Mediterranean Conference on Control and Automation, Corfu, Grecia, iunie, 2011, pag. 1076-1081.

[22] Dumitrascu B., Filipescu A., Minzu V., Voda A., Minca E.,Discrete-Time Sliding-Mode Control of Four Driving-Steering Wheels Autonomous Vehicle, *Proceedings of the 30th Chinesse Control Conference,* pag. 3620-3625, 2011.

[23] El-Sheimy N., Hou H., Niu Xiaoji, Analysis and modelling of inertial sensors using Allan variance, *IEEE Transactions on Instrumentation and Measurement*, vol. 57, nr. 1, pag.140-149, ianuarie, 2008.

[24]Filipescu A., Minzu V., Dumitrascu B. and Filipescu A., Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots, *The 2011 IEEE International Conference on Information and Automation*, iunie, 2011.

[25] Flenniken W.S., Wall, J. H., Bevly D.M., Characterization of various IMU error sources and the effect on navigation performance, *http://www.eng.auburn.edu/~dmbevly/ gavlab/pub-pre/Flenniken_ION_GNSS_2005.pdf.*

[26] Foka A., Trahanias P., Real-time hierarchical POMDPs for autonomous robot navigation, *Robotics and Automatonomous Systems,* pag. 561-571, vol. 55, 2007.

[27] Fraichard T. And Scheuer A., From Reeds and Shepp's to continuous-curvature paths, IEEE *Transactions On Robotics and Automation*, vol. 20, nr. 6,pag 1025-1035, 2004.

[28] Frazzoli E., *Robust Hybrid Control for Autonomous Vehicle Motion Planning,Massachusetts Institute of Technology,* iunie, 2001.

[29] Freeman J. A., Skapura D. M., Neural networks algorithms, applications, and programming techniques*, Loral Space Information Systems and Adjunct Faculty, School of Natural and Applied Sciences University of Houston at Clear Lake,* 1991.

[30] Fukao T., Nakagawa H., Adachi N., Adaptive tracking control of a nonholonomic mobile robot, *IEEE Transactions on Robotics and Automation*, vol. 16, nr.5, pag. 609-615, 2000.

[31] Gao W. and. Hung J. C, Variable structure control of nonlinear systems: A new approach, *IEEE Transactions on Industrial Electronics*, 40(1), pag. 45 - 55, 1993.

[32] Gao W., Wang Y. and Homaifa A., "Discrete-time variable structure control systems", *IEEE Transaction on Industrial Electronics*, vol. 42, 117-122, 1995.

[33] Garcia M. A. P., Montiel O., Castillo O., Sepulveda R., Melin P., Path palanning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, *Applied Soft Computing*, vol.9 , pag 1102-1110, 2009.

[34] Ge S. S., Lai X.-C., Mamun A. Al., Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints, *Robotics and Autonomous Systems*, vol. 57, pag. 513-526, 2007.

[35] Grewal M.S., Andrews A. P., Kalman filtering: theory and practice using MATLAB, second edition, John Wiley & Sons, Inc., New York, 2001.

[36] Henderson T. C., Minor M., Drake S., Hetrick A., Quist J., Roberts J., Sani H., Bandaru R., Rasmussen M., Collins A., Sun Y. ,Suraj, Xiuyi Fan, Ed St. Louis, Shigenori Mikuriya, Ken Dean, Robust autonomous vehicles*,University of Utah,* June 2007.

[37]Hou H., Modeling inertial sensors errors using Allan variance. Phd Thesis, University of Calgary, 2004.

[38] http://en.wikipedia.org/wiki/TerraMax_(vehicle)

[39] http://www.mobilerobots.com/software/mobilesim.aspx

[40] http://news.xinhuanet.com/english/2010-01/18/content_12829542_1.htm

[41] http://photojournal.jpl.nasa.gov/catalog/PIA01122

[42] http://store.irobot.com/home/index.jsp

[43] http://viac.vislab.it/

[44] http://www.cs.iastate.edu/~cs577/handouts/quaternion.pdf

[45] http://www.husqvarna.com/us/homeowner/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners/

[46] http://www.tigertek.com/servo-motor-resources/incremental-absolute-encoders.html.

[47] Hung J. Y., Gao W., Hung J. C., Variable structure control: A survey, *IEEE Transactions on Industrial Electronics*, vol. 40(1), pag.2-22,1993.

[48] IEEE standard, Specification format guide and test procedure for single-axis interferometric fiber optic gyros, *IEEE-SA Standards Board*.

[49] Jalili-Kharaajoo M. and Moshiri B., Discrete-time sliding mode control for linear time-varying systems with application to steering control on highway vehicles, University of Tehran, Iran.

[50] Jeffrey Roderick Norman Forbes, *Reinforcement Learning for Autonomous Vehicles,B.S. (Stanford University), 1993.*

[51] Jiang J.P. and Nijmeijer H., Backstepping-based tracking control of nonholonomic chained systems, *European Control Conference*, Brussels, iulie, 1997.

[52] Jiang J.P. and Nijmeijer H., Tracking control of mobile robots: a case study in backstepping, *Automatica*, vol. 33, nr. 7, pag. 1393-1399, 1997.

[53] Joan Sola, Quaternion Kinematics for the error-state Kalman filter, http://www.joansola.eu/ JoanSola/objectes/notes/kinematics.pdf, august, 2012.

[54] Joao Luis Marins, Xiaoping Yun, Eric R. Bachman, Robert B. McGhee, Michael J. Zyda, An extended Kalman filter for quaternion-based orientation estimation using MARG sensors, *IEEE/RSJ International Conference on Intelligent Robots and Systems,* Maui, USA,octombrie, 2001.

[55] Kanarat A., Motion Planning and robust control for nonholonomic mobile robots under uncertainties, PhD Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2004.

[56] Kanayama Y., Kimura Y., Miyazaki F. and Noguchi T., A stable tracking control scheme for an autonomous mobile robot, *Proceedings of the IEEE International Conference on Robotics and Automation*, pag. 384-389,1990.

[57] Khalil H.K., Nonlinear Systems, Second Edition, Prentice Hall, New Jersey, 1996.

[58] Khatib O., Real-time obstacle avoidance for manipulators and mobile robots, *IEEE International Conference on Robotics and Automation*, pag. 500-505, martie, 1985.

[59] Kimberly Tuck, Tilt sensing using linear accelerometers, Freescale Semiconductors Application Note, 2007.

[60] Kong X., Inertial navigation system algorithms for low cost IMU, PhD Thesis, University of Sydney, 2007.

[61] Kumar U. and Sukavanam N., Backstepping based trajectory tracking control of a four wheeled mobile robot, *International Journal of Advanced Robotic Systems*, Vol..5, No.4 , pag 403-410,2008.

[62] Lee H., Chattering suppression in sliding mode control system, PhD Thesis M. S. Ohio State University, 2007.

[63] Lee J. H, Lin C., Lim H. and Jang Myung Lee, Sliding mode control of mobile robot in the RFID sensor space, *International Journal of Control, Automation and Systems,* pag 429-435,2009.

[64] Luca Caracciolo, Alessandro De Luca Stefano Iannitti, Trajectory Tracking Control of a Four-Wheel Differentially Driven Mobile Robot, *Proceedings of the 1999 IEEE International Confererence on Robotics & Automation,* Detroit, Michigan, May 1999.

[65] Lucet E. , Grand C., Salle D., Bidaud P., Dynamic sliding mode control of a four-wheel skid-steering vehicle in presence of sliding, http://www.doc-center.robosoft.com/@api/ deki/ files/3002/=Dynamic_sliding_mode_control_of_a_four-wheel_skid-steering_vehicle_in_presence_of_sliding.pdf.

[66] Lumelsky V., Skewis T., Incorporating range sensing in the robot navigation function, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, nr. 5, pag. 1058-1068, 1990.

[67] Madgwick S. O.H., Harrison A. J.L., Vaidyanathan R., Estimation of IMU and MARG orientation using a gradient descent algorithm, *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, Elvetia, pag. 1-7 ,2011.

[68] Malcolm D. Shuster, The quaternion in Kalman filtering, AAS/AIAA Astrodynamics Conference, British Columbia, august 1993, Advances in the Astronautical Sciences, vol 85, pag. 25-37, 1993.

[69] Manalov O. B., Adaptive steering control for autonomous mobile robots, *Proceedings of the 10th Mediterranean Conference on Control and Automation- Med2002*, Portugalia, 2002.

[70] Maxim Likhachev, Dave Ferguson, Planning long dynamically-feasible maneuvers for autonomous vehicles, *http://www.cs.cmu.edu/~maxim/files/planlongdynfeasmotions _rss08.pdf.*

[71] Menegatti E., Maeda T., Ishiguro H., Image-based memory for a robot navigation using properties of omnidirectional images, *Robotics and Autonomous Systems*, vol. 47 , pag. 251-267, 2004.

[72] Mnif F., Recursive backstepping stabilization of a wheeled mobile robot, *International Journal of Advanced Robotic Systems*, vol. 1 nr. 4 , pag. 287 – 294,2004.

[73] Momcilovic O. I., Discrete-time variable structure controller synthesis for third order objects with finite zero using delta transform, *Journal of Electrical Engineering*, vol. 55, 3-4, pag . 71-76, 2004.

[74] Myungsoo Jun, Atif I. Chaudhry, Raffaello D'Andrea, The navigation of autonomous vehicles in uncertain dynamic environments: a case study, *Sibley School of Mechanical and Aerospace Engineering, Cornell University Ithaca, NY 14853-7501, USA.*

[75] Naveen K. Boggarpu, Richard C. Kavanagh, New learning algorithm for high-quality velocity measurement from low-cost optical encoders, *IMTC 2008- IEEE Instrumentation and Measurement Technology Conference*, Victoria, Canada, mai, 2008.

[76] Nassar S., Improving the inertial navigation system (INS) error model for INS and INS/DGPS applications, PhD Thesis, The University of Calgary, 2003.

[77] Park M., Error analysis and stochastic modeling of MEMS based inertial sensors for land vehicle navigation applications, Master Thesis University of Calgary, 2004.

[78] Petrella R., Tursini M., Peretti L., Zigliotto M., Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative a analysis, *International Aegean Conference on Electrical Machines and Power Electronics*,Bodrum, Turcia, septembrie, 2007.

[79] Philips W. F., Hailey C.E., and Gebert G.A., Review of attitude representations used for aircraft kinematics, *Journal of Aircraft*, 38(4), pag. 718-737, 2001.

[80] Qiang Z., Zengbo L., Yao C., A Back-stepping Based Trajectory Tracking Controller for a Non-chained Nonholonomic Spherical Robot, *Chinese Journal of Aeronautics* 21, pag. 472-480, 2008.

[81] Qu S., Wang Y., Zhu Q., A discrete sliding variable structure control with dynamic disturbance compensator, Control 2004, University of Bath, Uk, 2004.

[82] Ren H. And Kazanzides P., Investigation of attitude tracking using an integrated inertial and magnetic navigation system for hand-held surgical instruments, *IEEE/ASME Transactions on Mechatronics*, vol. 17, nr.2, pag. 210-217, 2012.

[83] Rubio J. De Jesu´s and Yu W., A new discrete-time sliding-mode control with time-varying gain and neural identification, *International Journal of Control*, vol. 79, nr. 4, pag. 338–348, aprilie, 2006.

[84] Salah Sukkarieh, Low cost, high integrity, aided inertial navigation systems for autonomous land vehicles, PhD thesis, 2000.

[85] Schouwenaars T., Safe trajectory planning of Autonomous Vehicles, PhD Thesis, Massachusetts Institute of Technology, 2006.

[86] Sebastian O. H. Madwick, Quaternions, An efficient orientation filter for inertial and inertial/magnetic sensor arrays, aprilie, 2010.

[87] Seekur Quick Start, Mobile Robots Inc., 2007.

[88] Shaocheng Q., Yongji W., Robust sliding mode control for uncertain discrete time systems, *Journal of Chongqing University-Eng*. Ed. vol. 2, nr. 2, pag. 51-54, 2003.

[89] Shi Hongli, A novel scheme for the design of backstepping control for a class of nonlinear systems, *Applied Mathematical Modelling* 35 (2011) 1893–1903.

[90] Slotine J.J.E. and Li W., Applied nonlinear control. *Prentice-Hall*, London, 1991.

[91] Solea R., Filipeascu A., Stamatescu G., Trajectory-tracking sliding-mode control for wheeled mobile robots, Energy, *Transport and Envinronment Control Applications- International Workshop*, pag 121-134, mai, 2009.

[92] Solea R., Filipescu A. and Stamatescu G., Sliding-mode real-time mobile platform control in the presence of uncertainties, *Proceedings of the 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, pag. 7747-7752,2009.

[93] Solea R., Filipescu A., Filipescu S, and Dumitrascu B., Sliding-mode controller for four- wheel-steering vehicle: trajectory-tracking problem, *Proceedings of the 8th World Congress on Intelligent Control and Automation*, Jinan, China, pag. 1185-1190, 2010.

[94] Solea R., Filipescu A., Manzu V., Filipescu S., Sliding-mode trajectory-tracking control of a four wheel steering vehicle, *2010 8th IEEE International Conference on Control and Automation*, Xiamen, China, pag. 382-387,2010.

[95] Solea R., Filipescu A., Nunes U., Sliding-mode control for trajectory-tracking of a wheeled mobile robot in presence of uncertainties, *Proceedings of the 7th Asian Control Conference*, Hong Kong, pag. 1701-1706,2009.

[96] Solea R., Sliding mode control applied in trajectory-tracking of WMRs and autonomous vehicles, PhD Thesis, University of Coimbra, Portugal, 2009.

[97] Susnea I., Filipescu A., Vasiliu G., Coman G., Radaschin A., "The buble rebound obstacle avoidance algorithm for mobile robots", in *Proceedings. 8th IEEE International Conference on Control and Automation*, Xiamen, pag. 540-545, 2010.

[98] Tanner H. G., Kyriakopoulos K. J., Backstepping for nonsmoth systems, *Automatica*, 39, pag 1259-1265, 2003.

[99] Totha C. K., Paska E., Mobile mapping and autonomous vehicle navigation, *Department of Civil and Environmental Engineering and Geodetic Science, Columbus, USA.*

[100] Tsang, Chi Chu, Error reduction techniques for a MEMS accelerometer-based digital input device, Master Thesis, University of Hong Kong, 2008.

[101] Utkin V.I., Guldner J., and Shi J., Sliding mode control in electromechanical systems, *Taylor & Francis*, London, 1999.

[102] Utkin V.I., Sliding modes in optimization and control, *Springer-Verlag*, New York, 1992.

[103] Vecchio C., Sliding mode control: theoretical developments and applications to uncertain mechanical systems, PhD Thesis, Universita Degli Studi Di Pavia.

[104] Velagic J., Lacevic B., Perunicic B., A 3-level autonomous mobile robot navigation system designed by using reasoning/ search approaches, *Robotics and Systems*, vol. 54, 989-1004,2006.

[105] Vivekanandan C., Prabhakar R., A redefined discrete quasi-sliding mode strategy, *International Journal of Recent Trends*, vol. 1, nr. 3, pag 92-96, 2009.

[106] Vivekanandan V., Prabhakar R. and Gnanambigai M., A redefined quasi-sliding mode control strategy, *World Academy of Science, Engineering and Technology*, 39, pag 292-295, 2008.

[107] Woodman O.J., An introduction to inertial navigation, Raport tehnic(696), *Computer Laboratory of Cambridge*, 2007.

[108] Xiao L., Su H., Zhang X., Chu J., A new discrete variable structure control algorithm based on sliding mode prediction, *2005 American Control Conference*, Portland, USA, pag 4643-4648, 2005.

[109] Xiaoping Yun, Bachman E.R., McGhee R. B. and Life Fellow, A simplified quaternion-based algorithm for orientation estimation from Earth gravity and magnetic field measurements, Instrumentation, 57(3), pag. 638-650,2008.

[110] Xiaoping Yun, Mariano Lizarraga, Eric R. Bachman, Robert B. McGhee, An improved quaternion-based Kalman filter for real-life tracking of rigid body orientation, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, octombrie, 2003.

[111] Xin Zhang, Yong Li, Peter Mumford, and Chris Rizos, Allan variance analysis on error characters of MEMS inertial sensors for an FPGA-based GPS/INS system, International *Symposium on GPS/GNSS 2008*, Tokyo, Japonia, 2008.

[112] Yang K. and Sukkarieh S., An analytical continuous-curvature path-smoothing algorithm, *IEEE Transactions on Robotics*, vol. 26, nr. 3, pag. 561-568, 2010.

[113] Young Soo Suh, Orientation estimation using a quaternion-based indirect Kalman filter with adaptive estimation of external acceleration, *IEEE Transactions on Instrumentation and Measurement*, 59(12), pag. 3296-3305, decembrie, 2010.

[114] Yuan J., Tang G.-Y., Finite-time tracking control algorithms based on variable structure for mobile robots, *Proceedings of the 29th Chinese Control Conference*, Beijing, China, pag. 419-423*,* Iunie , 2010.

[115] Yuan Z. P., Wang Z. P. and Chen Q. J., Trajectory tracking control of a nonholonomic mobile robot, *2010 8th IEEE International Conference on Control and Automation* , Xiamen, China, pag. 2207-2211, iunie, 2010.

[116] Zhen Ziyang, Wang Zhisheng, Hu Yong, Multi-sensor information fusion for aircraft attitude determination system, 2009 WRI World Congress on Computer Science and Information Engineering, pag. 474-478, 2009.

[117] Zhou1 B., Han J., Dai X., Backstepping Based Global Exponential Stabilization of a Tracked Mobile Robot with Slipping Perturbation, *Journal of Bionic Engineering,* vol. 8 , pag. 69–76,2011.

8. Anexe

În anexe sunt prezentate programele sau funcțiile folosite pentru testarea metodelor de conducere şi fuziunea datelor.

Anexa 1. Programul folosit la conducerea backsteppping

Este prezentat programul folosit la testarea conducerii backstepping a vehiculului.

```
#include "Aria.h"
#include "Velocity.h"

//  Press escape to shut down Aria and exit.

static double angleLimit(double angle)
{
  if (angle > 3.14)
  {  printf("\t %4.5f  \n", "Angle");
  angle = angle - 2*3.14;}
  else
     if (angle < -3.14){
      angle = angle + 2*3.14;
     printf("\t %4.5f  \n", "Angle");}
  return(angle);
}

/* ------------- SIGN & SATURATION FUNCTION -------------- */

double sign(double value){
  if (value<0)
    return(-1.00);
  else
    if (value>0)
    return(1.00);
   else
 return(1.00);
}

double satur(double value){
  if (value<-1)
    return (-1.00);
  else if (value > 1)
   return (1.00);
  else
   return (value);
}
/* ------------ END (SIGN & SATURATION FUNCTION) ------------- */

int main(int argc, char **argv)
{
  Aria::init();
  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);
  ArRobot robot;
  ArSonarDevice sonar;
```

```
  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);

  // Make a key handler, so that escape will shut down the program
  // cleanly
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("You may press escape to exit\n");

  // Collision avoidance actions at higher priority
  ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
  ArActionLimiterForwards limiterFarAction("speed limiter far", 300, 11000, 400);
  //ArActionLimiterForwards limiterFarAction("speed limiter far", 600, 2200, 800);
  ArActionLimiterTableSensor tableLimiterAction;
  robot.addAction(&tableLimiterAction, 100);
  robot.addAction(&limiterAction, 95);
  robot.addAction(&limiterFarAction, 90);

  // Goto action at lower priority
  ArActionGoto gotoPoseAction("goto");
  robot.addAction(&gotoPoseAction, 50);

  // Parse all command line arguments
  if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
  {
    Aria::logOptions();
    exit(1);
  }

  // Connect to the robot
  if (!simpleConnector.connectRobot(&robot))
  {
    printf("Could not connect to robot... exiting\n");
    Aria::exit(1);
  }
  robot.runAsync(true);

  // turn on the motors, turn off amigobot sounds
  robot.enableMotors();
  //robot.comInt(ArCommands::SOUNDTOG, 0);

  bool first = true;
  int goalNum = 0;
/*
  while (Aria::getRunning())
  {
    robot.lock();
*/

  FILE *date1;
/*  double v, av, w, aw;                                    /* velocity to drive left wheel*/
  double theta_des_next;                  /* theta desired */
  double ref_xr, ref_yr, ref_Th, ref_w, ref_v, x_des_next, y_des_next;      /* desired position */
/
  int i; // n;

  double v,av,w,aw;
  double x,y,theta;
  double xd, yd, theta_d;
```

```
  double v_c, w_c;
double Ts;
double e1, e2, e1_der, e2_der;
double alfa;
double k1, k2, k3;
double timp;
double v_l, v_r;
  //-------------------------------------
   double lungime=240;

  date1=fopen("d:/test/test1.txt","w+");

  timp=0.0;
  Ts=0.1;
k1=0.5;
k2=1.25;
k3=0.3;


  ref_xr = 0.0;
  ref_yr = 0.0;
  ref_Th = 0.0;

  ref_v = 0.0;
  ref_w = 0.0;

  //gama0 = gama_y/(3.14/2);

 // n = lungime-1;
 //    n = 3;
                robot.lock();
                ArUtil::sleep(100);
                robot.unlock();
                ArUtil::sleep(100);


                for (i=1; i<lungime; i++){
                robot.lock();
                                v=velocity[i][0];
                av=velocity[i][1];
                w=velocity[i][2];
                aw=velocity[i][3];



                theta_des_next = angleLimit(Ts*w+ref_Th);
                x_des_next = Ts*v*cos(ref_Th) + ref_xr;
                y_des_next = Ts*v*sin(ref_Th) + ref_yr;



                x=(robot.getX()/1000);
                y=(robot.getY()/1000);
                theta=robot.getTh()*(3.14/180);
                e1=x-ref_xr;
                e2=y-ref_yr;

                e1_der=cos(theta)*(robot.getVel()/1000)-v*cos(ref_Th);
                e2_der=sin(theta)*(robot.getVel()/1000)-v*sin(ref_Th);
```

```
                //v_c_der = (temp1 + (robot.getVel()/1000)*theta_e_der*sin(theta_e))/(cos(theta_e));

                v_c = (-k1*e1+v*cos(ref_Th))/cos(theta);
        // v_c = (-k1*e1+v*cos(ref_Th))/cos(theta)*Ts+ref_v;

                alfa=tan(theta)*(v*cos(ref_Th)-k1*e1)-v*sin(ref_Th)+k2*e2;


        //w_c =(cos(theta)/v_c)*(tan(theta)*(v*cos(ref_Th)-k1*k1*e1)-v*sin(ref_Th)-k2*k2*e2-
k2*alfa+k3*alfa);

        w_c =(cos(theta)/v_c)*(tan(theta)*(av*cos(ref_Th)-v*ref_Th*sin(ref_Th)-k1*k1*e1)-av*sin(ref_Th)-
v*ref_Th*cos(ref_Th)-k2*k2*e2-k2*alfa+k3*alfa);

                // ROBOT
                        v_l = v_c + 0.28*w_c;       //0.19 //0.28 //0.24
                        v_r = v_c - 0.28*w_c;


                //      Simulator
                //      v_r = v_c + 0.19*w_c;
                //      v_l = v_c - 0.19*w_c;

            // setVel2 (double leftVelocity, double rightVelocity)
              robot.setVel2(1000*v_l, 1000*v_r);

                // printf("\t %4.5f \t %4.5f \t %4.5f %4.5f \t %4.5f \n", robot.getX()/1000, robot.getY()/1000,
robot.getTh()*(3.14/180), v_r, v_l);
                // printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", robot.getVel()/100,
robot.getRotVel()*3.14/180, v_c, w_c);

         fprintf(date1," %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t
%4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n",
robot.getX()/1000, robot.getY()/1000, robot.getTh()*(3.14/180), ref_xr, ref_yr, ref_Th, e1, e2,
(robot.getVel()/100), (robot.getRotVel()*3.14/180), v, w, v_c, w_c,  timp);
                printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", e1, e2,
w,w_c,robot.getRotVel()*3.14/180);

        //   printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", ref_xr, ref_yr, ref_Th*(180/3.14),
robot.getRobotDiagonal()/1000, robot.getRobotRadius()/1000);
                ref_xr = x_des_next;
                ref_yr = y_des_next;
                ref_Th = theta_des_next;
                ref_w = w_c;
                ref_v = v_c;
                timp=timp+Ts;

                robot.unlock();
                ArUtil::sleep(100);
          }

          // Robot disconnected, shut down
          Aria::shutdown();
          return 0;
        }
```

Anexa 2. Programul folosit la conducerea sliding-mode

Este prezentat programul folosit la testarea conducerii sliding-mode a vehiculului.

```
#include "Aria.h"
static double angleLimit(double angle)
{
  if (angle > 3.14)
    angle = angle - 2*3.14;
  else
    if (angle < -3.14)
      angle = angle + 2*3.14;
  return(angle);
}

double fi(double value1, double value2){
   if(abs(value1)>value2)
              value1=1;
   else value1=0;
   return (value1);
}
double psi(double value1,double value2){
   if(abs(value1)>value2)
              value1=1;
   else value1=(abs(value1)*abs(value1))/value2;
   return (value1);
}
/* ------------- SIGN & SATURATION FUNCTION -------------- */

double sign(double value){
  if (value<0)
    return(-1.00);
  else
    if (value>0)
    return(1.00);
   else
 return(1.00);
}

double satur(double value){
  if (value<-1)
    return (-1.00);
  else if (value > 1)
   return (1.00);
  else
   return (value);
}
/* ------------ END (SIGN & SATURATION FUNCTION) ------------- */

int main(int argc, char **argv)
{
  Aria::init();
  ArArgumentParser parser(&argc, argv);
  parser.loadDefaultArguments();
  ArSimpleConnector simpleConnector(&parser);
  ArRobot robot;
  ArSonarDevice sonar;
  ArAnalogGyro gyro(&robot);
  robot.addRangeDevice(&sonar);
```

```
  // Make a key handler, so that escape will shut down the program
  // cleanly
  ArKeyHandler keyHandler;
  Aria::setKeyHandler(&keyHandler);
  robot.attachKeyHandler(&keyHandler);
  printf("You may press escape to exit\n");

  // Collision avoidance actions at higher priority
  ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
  ArActionLimiterForwards limiterFarAction("speed limiter far", 300, 11000, 400);
  //ArActionLimiterForwards limiterFarAction("speed limiter far", 600, 2200, 800);
  ArActionLimiterTableSensor tableLimiterAction;
  robot.addAction(&tableLimiterAction, 100);
  robot.addAction(&limiterAction, 95);
  robot.addAction(&limiterFarAction, 90);

  // Goto action at lower priority
  ArActionGoto gotoPoseAction("goto");
  robot.addAction(&gotoPoseAction, 50);

  // Parse all command line arguments
  if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
  {
    Aria::logOptions();
    exit(1);
  }

  // Connect to the robot
  if (!simpleConnector.connectRobot(&robot))
  {
    printf("Could not connect to robot... exiting\n");
    Aria::exit(1);
  }
  robot.runAsync(true);

  // turn on the motors, turn off amigobot sounds
  robot.enableMotors();
  //robot.comInt(ArCommands::SOUNDTOG, 0);

  bool first = true;
  int goalNum = 0;
/*
  while (Aria::getRunning())
  {
    robot.lock();
*/

  FILE *date1;
  double v, av, w, aw;
  double theta_des_next;
  double ref_xr, ref_yr, ref_Th, ref_w, ref_v, x_des_next, y_des_next;
  double x_e, y_e, theta_e, x_e_der, y_e_der, theta_e_der;
  double s_1, s_2, Q1, Q2, P1, P2, gama0, gama_x, gama_y;
  double v_c, v_c_der, w_c, temp1, temp2;
  double Ts, v_l, v_r, timp, alpha;
  int i; // n;

  //-------------------------------------

  date1=fopen("d:/ccc/testcc8.txt","w+");
```

```
timp=0.0;
Ts=0.1;

Q1 = 0.05;
Q2 = 0.5;
P1 = 0.5;
P2 = 0.75;
alpha = 0.5;

 gama0 = 30;        //20
  gama_x = 1.25;   //1.25
 gama_y = 100;                //15

 ref_xr = 0.0;
ref_yr = 0.0;
ref_Th = 0.0;

ref_v = 0.0;
ref_w = 0.0;


            robot.lock();
            ArUtil::sleep(100);
            robot.unlock();
            ArUtil::sleep(1000);


  for (i=1; i<400; i++){
            robot.lock();
            v=0.5;
            av=0;
            w=0;
            aw=0;
            if(i>200){
                        w=0.2;}

 /*         for (i=1; i<400; i++){
            robot.lock();
            v=0.5;
            av=0;
            w=0;
            aw=0;
            if(i>200){
                        w=-0.2;}            */
            theta_des_next = angleLimit(Ts*w+ref_Th);
            x_des_next = Ts*v*cos(ref_Th) + ref_xr;
            y_des_next = Ts*v*sin(ref_Th) + ref_yr;

            x_e = ((robot.getX()/1000)-ref_xr)*cos(ref_Th)+((robot.getY()/1000)-ref_yr)*sin(ref_Th);
            y_e = -((robot.getX()/1000)-ref_xr)*sin(ref_Th)+((robot.getY()/1000)-ref_yr)*cos(ref_Th);
            theta_e = angleLimit((robot.getTh()*(2.14/180))-ref_Th);

                        if (y_e < 0)
         {
            gama_y = -abs(gama_y);
         }
          else if( y_e > 0 )
         {
            gama_y = abs( gama_y );
```

```
            }

                x_e_der = -v + (robot.getVel()/1000)*cos(theta_e)+y_e*w;
                y_e_der = (robot.getVel()/1000)*sin(theta_e)-x_e*w;
                theta_e_der = (robot.getRotVel()*3.14/180)-w;


                /* SLIDING SURFACE */
                 s_1 = x_e_der + gama_x*x_e;
                 s_2 = y_e_der + gama_y*y_e + gama0*sign(y_e)*(theta_e);


            temp1 = -Q1*satur(s_1/0.5) - (P1*s_1) - gama_x*x_e_der - (aw*y_e) - w*y_e_der + av;
                  temp2 = -Q2*satur(s_2/0.5) - (P2*s_2) - gama_y*y_e_der + (aw*x_e) + w*x_e_der;


                v_c_der = (temp1 + (robot.getVel()/1000)*theta_e_der*sin(theta_e))/(cos(theta_e));

                v_c = (Ts*v_c_der) + ref_v;

                w_c                                                    =                                    (temp2-
v_c_der*sin(theta_e))/((robot.getVel()/1000)*cos(theta_e)+gama0*sign(y_e))+w;

                // ROBOT
                        v_l = v_c + 0.4*w_c;        //0.19 //0.28 //0.24
                        v_r = v_c - 0.4*w_c;


                    robot.setVel(1000*v_c);

                robot.setRotVel(w*180/3.14);

        fprintf(date1," %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t
%4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n",
robot.getX()/1000, robot.getY()/1000, robot.getTh()*(2.14/180), ref_xr, ref_yr, ref_Th, x_e, y_e, theta_e,
x_e_der, y_e_der, theta_e_der, (robot.getVel()/1000), (robot.getRotVel()*3.14/180), v, w, v_c, w_c, s_1, s_2,
timp);
                printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", x_e, y_e, theta_e, robot.getVel()/1000,
w);


                ref_xr = x_des_next;
                ref_yr = y_des_next;
                ref_Th = theta_des_next;
                ref_w = w_c;
                ref_v = v_c;
                timp=timp+Ts;

                robot.unlock();
                ArUtil::sleep(100);
        }

        // Robot disconnected, shut down
        Aria::shutdown();
        return 0;
       }
```

Anexa 3. Programul principal folosit la estimarea atitudinii

Este prezentat programul principal scris în MATLAB, folosit la fuziunea datelor de la senzorul IMU Xsens pentru determinarea atitudinii:

```
clc
clear all

%Log=OpenAndProcessLogs4('log-18_9_29_3_2012_front_back_LITcars.txt');
Log=OpenAndProcessLogs4('log-18_0_29_3_2012_firetruck_front.txt');
Acc=[Log.IO.Xsens_acc_X Log.IO.Xsens_acc_Y Log.IO.Xsens_acc_Z]';
Acc(3,:)=Acc(3,:);
Gyro=[Log.IO.Xsens_gyro_X Log.IO.Xsens_gyro_Y Log.IO.Xsens_gyro_Z]';
Mag=[Log.IO.Xsens_compass_X  Log.IO.Xsens_compass_Y Log.IO.Xsens_compass_Z]';
Calibrated=[Log.IO.Xsens_quaternion_X Log.IO.Xsens_quaternion_Y Log.IO.Xsens_quaternion_Z
Log.IO.Xsens_quaternion_W]';
r=eulr2qua([0 0 pi/2]);
for i=1:length(Calibrated)
   %Calibrated(:,i)=quaternionmul(Calibrated(:,i)',r);
   xsens(:,i)=quaternion2euler(Calibrated(:,i));
end

R=[0 1 0; -1 0 0; 0 0 1];
ym=Mag;
%ym(1,:)=Mag(1,:);
%ym(2,:)=Mag(2,:);
%ym(3,:)=Mag(3,:);

yg=Gyro;
%yg(1,:)=Gyro(1,:);
%yg(2,:)=Gyro(2,:);
%yg(3,:)=Gyro(3,:);

ya=Acc;
%ya(1,:)=Acc(1,:);
%ya(2,:)=Acc(2,:);
%ya(3,:)=-Acc(3,:);

tt=0:.1:(length(ya)-1)*0.1;

D2R = pi/180;
R2D = 180 / pi;
Rg = 1e-4 * eye(3);
Ra = 1e-2 * eye(3);
Rm = 1e-3*eye(3);

[q4, eulercom4, bahat, bghat, bmhat] = computeorientation (yg,ya,ym,tt,Rg,Ra,Rm);

%
% estimation result
%

figure(3)
title('Pitch')
plot(tt,xsens(2,:)*R2D,'r--',tt,eulercom4(2,:)*R2D);
ylabel('pitch \phi (deg)');
xlabel('time (sec)');
```

```
legend('true value','estimated value')

figure(4)
plot(tt,xsens(1,:)*R2D,'r--',tt,eulercom4(1,:)*R2D);
title('Roll');
ylabel('roll \theta (deg)');
xlabel('time (sec)');
legend('true value','estimated value')
figure(5)
plot(tt,xsens(3,:)*R2D,'r--',tt,eulercom4(3,:)*R2D);
ylabel('yaw \psi (deg)');
xlabel('time (sec)');
title('Heading');
legend('true value','estimated value')
```

Anexa 4. Funcția pentru calculul atitudinii utilizând filtrul Kalman

Funcția MATLAB computerorientation returnează atitudinea reprezentată în quaternion (q4), atitudinea reprezentată în unghiuri Euler(eulercom4) și biasurile estimate utilizând filtrul Kalman. Funcția primește ca parametri datele de la giroscoape(yg), datele de la accelerometre(ya), datele de la senzorii magnetici(ym), vectorul timp(tt) și matricile de covarianță corespunzătoare senzorilor.

```
function [q4, eulercom4, bahat, bghat, bmhat] = computeorientation (yg,ya,ym,tt,Rg,Ra,Rm)

D2R = pi/180;

% number of data: N (>= 3)
N = max(size(ya));

% gravitation acceleration
g = 9.8;
gtilde = [ 0 ; 0 ; g];


% Magnetic Inclination angle at COIMBRA (degree)
inc = (54+49/60) * D2R;
% Magnetic Declination angle at COIMBRA (degree)
dec=-3 * D2R;
mtilde = [cos(inc)*cos(dec);cos(inc)*sin(dec); -sin(inc) ];
% dip angle at Ulsan (degree)
%alpha = 50 * D2R;
%mtilde = [ cos(alpha) ; 0 ; -sin(alpha) ];

% bias update covariance
Qbg = eye(3) * 1e-8;    %allan plot bias instabilitty for gyro 10^-4
Qba = eye(3) * 9e-8;    %allan plot bias instabilitty for accelerometer 3*10^-4
Qbm = eye(3) * 1e-10;    %allan plot bias instabilitty for magnet 10^-4   Xsense
Qbg0 = eye(3)* 1e-4;    %initial covariance
Qba0 = eye(3)* 1e-4;    %initial covariance
Qbm0 = eye(3)* 1e-5;    %initial covariance

% -------------------------------------------------------
% Filtrul Kalman
% -------------------------------------------------------

% q4 : quaternion
q4 = zeros(4,N);

% eulercom4 : euler angles
eulercom4 = zeros(3,N);

bghat = zeros(3,1);
bahat = zeros(3,1);
bmhat = zeros(3,1);

yabar = ya(:,1) / norm(ya(:,1));
ymbar = ym(:,1) / norm(ym(:,1));

f1 = cross(yabar,ymbar) / norm( cross(yabar,ymbar) );
C = [ -cross(yabar,foo1)  , f1 , yabar ] ;
q4(:,1) = dcm2quaternion(C);
```

```
x = zeros(12,1);
P = zeros(12,12);
Q = zeros(12,12);
Q(1:3,1:3) = 0.025 * Rgş
Q(4:6,4:6) = Qbg;
Q(7:9,7:9) = Qba;
Q(10:12,10:12) = Qbm;

% intial error covariance
P(1:3,1:3) = 0.01*eye(3);
P(4:6,4:6) = Qbg0;
P(7:9,7:9) = Qba0;
P(10:12,10:12) = Qbm0;

% A matrix
A = zeros(12,12);
A(1:3,4:6) = -0.5 * eye(3);

% initializarea quaternionului pt integrare
wx = yg(1,1);
wy = yg(2,1);
wz = yg(3,1);    %
oldomega4 =[  0 ,-wx ,-wy ,-wz ;
            wx ,  0 , wz ,-wy ;
            wy ,-wz ,  0 , wx ;
            wz , wy ,-wx ,  0];


% variabile folosite in algorimul de adaptare
r2count = 100;

% H1 and H2 for measurement update
H1 = zeros(3,12);
H1(1:3,7:9) = eye(3);
H2 =  zeros(3,12);
H2(1:3,10:12) = eye(3);

% parametrii pentru algoritmul adaptiv
M1 = 3;
M2 = 3;
gamma = 0.1;

R = zeros(3,3*N);

% bucla filtrului Kalman
for i = 2:N
   T = tt(i) - tt(i-1);

   Cq = quaternion2dcm(q4(:,i-1));
   A(1:3,1:3) = -vec2product(yg(:,i-1) - bghat);
   dA = eye(12) + A * T + A * A * T^2 / 2;

   x = dA * x;
   Qd = Q*T + 0.5*T*T*A*Q + 0.5*T*T*Q*A';
   Qd = 0.5*(Qd + Qd');
   P = dA * P * dA' + Qd;

   yyg = yg(:,i-1) - bghat;
```

```
wx = yyg(1);
wy = yyg(2);
wz = yyg(3);
omega4 = [ 0 , -wx , -wy , -wz ; ...
           wx , 0 , wz , -wy ; ...
          wy , -wz , 0 , wx ; ...
          wz , wy , -wx , 0];

q4(:,i) = ( eye(4) + 0.75 * omega4 * T - 0.25 * oldomega4 * T - (1/6) * norm(yyg)^2 * T^2 * eye(4) -
(1/24) * omega4 * oldomega4 * T^2  - (1/48)*norm(yyg)^2 *omega4 * T^3) * q4(:,i-1);
q4(:,i) = q4(:,i) / norm(q4(:,i));
oldomega4 = omega4;

Cq = quaternion2dcm(q4(:,i));

% ----------------------------------------------------
% update in 2 pasi
% ----------------------------------------------------
H1(1:3,1:3) = 2 * vec2product(Cq*gtilde);
z1 = [ ya(:,i) - bahat - Cq*gtilde ];

% adaptive algorithm
fR1 = (z1 - H1*x) * (z1 - H1*x)';
R(:,3*(i-1)+1:3*i) = fR1;
uk = fR1;
for j = i-1:min([i-(M1-1),1])
   uk = uk + R(:,3*(j-1)+1:3*j);
end
uk = uk / M1;


fR2 = H1*P*H1' + Ra;

[u,s,v] = svd(uk);
u1 = u(:,1);
u2 = u(:,2);
u3 = u(:,3);

lam = [ s(1) , s(2) , s(3)];
mu =  [ u1' * fR2 * u1 , u2' * fooR2 * u2 , u3' * fR2 * u3];

if ( max(lam - mu) > gamma )
   r2count = 0;
   Qstar = max(lambda(1)-mu(1),0)*u1*u1' + max(lam(2) -mu(2),0)*u2*u2' + max(lam(3)-
mu(3),0)*u3*u3';
else
   r2count = r2count + 1;
   if ( r2count < M2 )
      Qs = max(lam(1)-mu(1),0)*u1*u1' + max(lam(2) -mu(2),0)*u2*u2' + max(lam(3)-
mu(3),0)*u3*u3';
   else
      Qs = zeros(3,3);
   end
end

P1 = P;
K1 = P1 * H1' * inv(H1 * P1 * H1' + Ra + Qs);
x = x + K1 * (z1 - H1 * x);
```

```
    P = (eye(12) - K1*H1)*P*(eye(12)-K1*H1)' + K1*(Ra+Qs)*K1';
    P = 0.5 * (P + P');

    qe = [ 1 ; x(1:3) ];
    q4(:,i) = quaternionmul(q4(:,i),qe);
    q4(:,i)=  q4(:,i) / norm(q4(:,i));
    x(1:3) = zeros(3,1);
    Cq = quaternion2dcm(q4(:,i));

    H2(1:3,1:3) =  2 * vec2product(Cq*mtilde);
    z2 = [ym(:,i) - bmhat - Cq*mtilde];


    P2 = P;
    K2 = P2 * H2' * inv(H2 * P2 * H2' + Rm);
       x = x + K2 * (z2 - H2 * x);
    P = P - K2 * H2 * P - P * H2' * K2' + K2*(H2*P*H2'+Rm)*K2';
    P = 0.5 * (P + P');

    bghat = bghat + x(4:6);
    x(4:6) = zeros(3,1);

    bahat = bahat + x(7:9);
    x(7:9) = zeros(3,1);

    bmhat = bmhat + x(10:12);
    x(10:12) = zeros(3,1);

    qe = [ 1 ; x(1:3) ];
    q4(:,i) = quaternionmul(q4(:,i),qe);
    q4(:,i)=  q4(:,i) / norm(q4(:,i));
    x(1:3) = zeros(3,1);

    eulercom4(:,i) = quaternion2euler(q4(:,i));
end
```

Anexa 5. Funcția folosită pentru citirea datelor de la laser

În această secțiune este prezentată funcția getLaser, care este folosită pentru citirea datelor de la laser. Această funcție primește datele de la laser și apelând funcțiile Aria pentru procesarea datelor de la lasere calculează distanțele măsurate de laser utilizând funcția getRange() și memorează citirile din 5 în 5 grade în vectorul range , pentru fiecare distanță din vectorul range se memorează unghiul la care se efectuează măsurarea apoi se calculează bula de sensibilitate în vectorul boundry.

```
void getLaser(ArSick *thisLaser)
        {

        const std::list<ArSensorReading *> *readingsList;   //
                std::list<ArSensorReading *>::const_iterator it;    //

                int i = -1;                                 //
                readingsList = thisLaser->getRawReadings();        //                    int j=4; int
c=0;
for (it = readingsList->begin(); it != readingsList->end(); it++)
                {
                        i++;    j++;
                        double ra=(*it)->getRange();

                        if (j==5){
                                range[c]=ra/1000;
                                double angle=(*it)->getSensorTh();
                                alfa[c]=angle*(3.14/180);
                                  boundry[c]=k[c]*vrobot*Ts;

                                j=0;c++;  }
                                }
        }
```

Anexa 6. Funcția folosită pentru detecția obstacolelor

Este prezentată funcția folosită pentru a determina dacă a fost depistat un obstacol. Funcția obstacle returnează 1 dacă există un obstacol în zona de sensibilitate sau 0 altfel. Această funcție compară distanțele măsurate cu distanțele din vectorul boundry și dacă există distanțe mai mici decât distanța corespunzătoare din vectorul boundry se salvează poziția la care distanța este minimă în variabila pos.

```
int obstacle(double range[37], double boundry[37])  {
int i,j;
j=0;pos=0;
min_range=15;
printf("test");
for (i=1;i<37;i++) {
if(i!=18) {
                    if (range[i]<boundry[i]){

                        if (min_range>range[i])
                        {       min_range=range[i];
                         pos=i;
                        }
                    j=1;

                    }
            }
}
            return (j);

}
```

Anexa 7. Funcțiile folosite la căutarea unei soluții pentru evitarea obstacolului

Funcțiile searchleft() și searchright() caută o soluție pentru evitarea obstacolului. Aceste funcții caută punctul (new_x,new_y) care permite trecerea vehiculului.

```
void searchright()
{ double hb,h,H;
int j;
j=17;

if (((range[j]*cos(alfa[j]))>=min_range+0.1)||(range[j]==maxrange))
{
hb=abs(min_range*tan(alfa[j]));
}
else {
while (((range[j]*cos(alfa[j]))<min_range+0.1)&&(j>0))//////
{ j=j-1;}

hb=abs(min_range*tan(alfa[j]));
}
h=0;
while ((h<hmin)&&(j>0)&&(range[j]>min_range))
{ j=j-1;

h=abs(min_range*tan(alfa[j]))-hb;

printf("%4.5f \n",h);

}
if (h>=hmin){

H=h/2+hb;
new_x=new_x+min_range*cos(angleb)-H*sin(angleb);
new_y=new_y-min_range*sin(angleb)-H*cos(angleb);
found=1;
printf("Cautarea la dreapta a gasit o solutie=");

}
else {
found=0;
printf("Cautarea la dreapta nu a gasit o solutie pt evitare");
}

fprintf(date3," %4.5f \t %4.5f \t %4.5f \t %4.5f \n",j, hb,  new_x, new_y,
found);
}

void searchleft()
{ double hb,h,H;
int j;
j=19;
if
(((range[j]*cos(alfa[j]))>=min_range+0.1)||(range[j]==maxrange))
{
hb=abs(min_range*tan(alfa[j]));
}
else {
while (((range[j]*cos(alfa[j]))<min_range+0.1)&&(j<36))//////
```

```
                                        { j=j+1;}

        hb=abs(min_range*tan(alfa[j]));
                                }
                        h=0;
                        while ((h<hmin)&&(j<36)&&(range[j]>min_range))
                                { j=j+1;

                         h=abs(min_range*tan(alfa[j]))-hb;

                                }
                        if (h>=hmin){

                                        H=h/2+hb;

        new_x=new_x+min_range*cos(angleb)-H*sin(angleb);

        new_y=new_y+min_range*sin(angleb)-H*cos(angleb);
                                 found=1;
                         printf("Cautarea la stanga a gasit o solutie=");
                                printf("Pozitian n_X, n_Y=");
                        printf("%4.5f \t %4.5f  \n" , new_x, new_y);
                        }
                else {
                         found=0;
        printf("Cautarea la stanga nu a gasit o solutie pt evitare");
                }

                                fprintf(date3," %4.5f \t %4.5f \t %4.5f \t %4.5f \n",j, hb,  new_x, new_y,
found);

        }
```

Anexa 8. Funcția pentru generarea traiectoriei de ocolire a obstacolului

Funcția generare() generează traiectoria de ocolire a obstacolului care este impusă în locul traiectoriei dorite. Funcția calculează o traiectorie pentru evitarea obstacolului folosind Quintic ecuations, de la punctul curent la punctul (new_x, new_y) și asociază un profil de viteză acestei traiectorii. Traiectoria care urmează o fi urmărită de modulul de conducere este conținută în vectorii vel, acc, angvel și accang.

```
void  generare()
{
                    angleb=robot.getTh();
                    new_x=robot.getX()/1000;
                    new_y=robot.getY()/1000;
                    int po=pos;
                    min_range=min_range*cos(alfa[po]);
                         if (pos<18) {

                         printf("Cautare stanga");
                         searchleft();
                         if (found==0)
                         {
                         printf("Cautare dreapta");
                                   searchright();
                                   }
                         }
                         else {
                         printf("Cautare dreapta");
                         searchright();
                         if (found==0)
                         {
                         printf("Nu a fost gasita solutie");
                                   printf("Cautare stanga");
                                   searchleft();

                                   }
                                             }

                    if (found==1){

                    printf("directie selectata");

                    double tetaA=angleb; // unghi initial
                    double tetaB=angleb+3.14/4; // unghi final
                                                  //     robot.lock();
                    double xA=robot.getX()/1000;
                    double yA=robot.getY()/1000;
                    printf("PozitianX, Y=");
                    printf("%4.5f \t %4.5f  \n",xA, yA);
                                   //          robot.unlock();

                    double xB=new_x;
                    double yB=new_y;

                         double kA=0;
                         double kB=0;
```

```
                                        double kA_d=0;
                                        double kB_d=0;

                                        double g1=sqrt(pow((xA-xB),2)+pow(yA-yB,2));
                                        double g2=g1;
                                        double g3=0;
                                        double g4=-g3;

                                        double x0 = xA;
                                        double x1 = g1*cos(tetaA);
                    double x2 = 1/2*(g3*cos(tetaA)-ow((g1),2)*kA*sin(tetaA));
          double x3 = 10*(xB-xA)-(6*g1 + 3/2*g3)*cos(tetaA) - (4*g2 - (1/2)*g4 )*cos(tetaB) +
(3/2)*pow((g1),2)*kA*sin(tetaA) - (1/2)*pow((g2),2)*kB*sin(tetaB);
          double x4 = -15*(xB - xA) + (8*g1 + (3/2)*g3)*cos(tetaA) + (7*g2 - g4)*cos(tetaB) -
(3/2)*pow((g1),2)*kA*sin(tetaA) + pow((g2),2)*kB*sin(tetaB);
                    double x5 = 6*(xB - xA) - (3*g1 + (1/2)*g3)*cos(tetaA) - (3*g2 - (1/2)*g4)*cos(tetaB) +
(1/2)*pow((g1),2)*kA*sin(tetaA)-(1/2)*pow((g2),2)*kB*sin(tetaB);

                                        double y0 = yA;
                                        double y1 = g1*sin(tetaA);
                    double y2 = 1/2*(g3*sin(tetaA)+pow(g1,2)*kA*cos(tetaA));
                    double y3 = 10*(yB-yA)-(6*g1 + (3/2)*g3)*sin(tetaA) - (4*g2 - (1/2)*g4 )*sin(tetaB) -
(3/2)*pow((g1),2)*kA*cos(tetaA) + (1/2)*pow((g2),2)*kB*cos(tetaB);
                    double y4 = -15*(yB - yA) + (8*g1 + (3/2)*g3)*sin(tetaA) + (7*g2 - g4)*sin(tetaB) +
(3/2)*pow((g1),2)*kA*cos(tetaA) - pow((g2),2)*kB*cos(tetaB);
                    double y5 = 6*(yB - yA) - (3*g1 + (1/2)*g3)*sin(tetaA) - (3*g2 - (1/2)*g4)*sin(tetaB) -
(1/2)*pow((g1),2)*kA*cos(tetaA)+(1/2)*pow((g2),2)*kB*cos(tetaB);

                    printf("x1=");
                    printf("%4.5f \n",x3);


                    n=1;
                                                l[0]=0;
                                        double u;
                                        double alfa1[101],beta[101];
                                        double x_d;
                                        double y_d;
          double x_d_d;
          double y_d_d;
          double KK[101];
          double lungime=0;
                                        u=0;
                                        for (n=0;n<=100;n++)
                                        {
                                        u=u+0.01;
                                        printf("u=");
                                        printf("%4.5f \n",x0);

            alfa1[n]=x0+x1*u+x2*pow(u,2)+x3*pow(u,3)+x4*pow(u,4)+x5*pow(u,5);

            beta[n]=y0+y1*u+y2*pow(u,2)+y3*pow(u,3)+y4*pow(u,4)+y5*pow(u,5);
                                printf("alfa1=");
                    printf("%4.5f \t %4.5f \n",alfa1[n], beta[n]);
                                        if (n > 0)
                                //                  l[n] = l[n-1] + sqrt(pow((alfa1[n] - alfa1[n-1]),2) + pow((beta[n] -
beta[n-1]),2));
          l[n] = sqrt(pow((alfa1[n] - alfa1[n-1]),2) + pow((beta[n] - beta[n-1]),2));
```

```
x_d = x1 + 2*x2*u + 3*x3*pow(u,2) + 4*x4*pow(u,3) + 5*x5*pow(u,4);
y_d = y1 + 2*y2*u + 3*y3*pow(u,2) + 4*y4*pow(u,3) + 5*y5*pow(u,4);

x_d_d = 2*x2 + 6*x3*u + 12*x4*pow(u,2) + 20*x5*pow(u,3);
y_d_d = 2*y2 + 6*y3*u + 12*y4*pow(u,2) + 20*y5*pow(u,3);

KK[n]=(x_d*y_d_d - x_d_d*y_d)/((sqrt(pow((x_d),2) + pow((y_d),2)),3));
lungime=lungime+l[n];


}

for (n=1; n<100;n++)
{
vel[n]=l[n]/Ts;
acc[n]=vel[n]/Ts;
angvel[n]=vel[n]*KK[n]*l[n];
accang[n]=angvel[n]*vel[n];
 }
}

}
```

Anexa 9. Realizări roboți mobili

Fig. 8.1 Aspirator robotic autonom Roomba[41]

Fig. 8.2 Robotul autonom pentru spălat podele Scooba[41]

Fig. 8.3 Robotul autonom de tuns iarba produs de Husqvarna[44]

Fig. 8.4 Robotul autonom Robovie-II

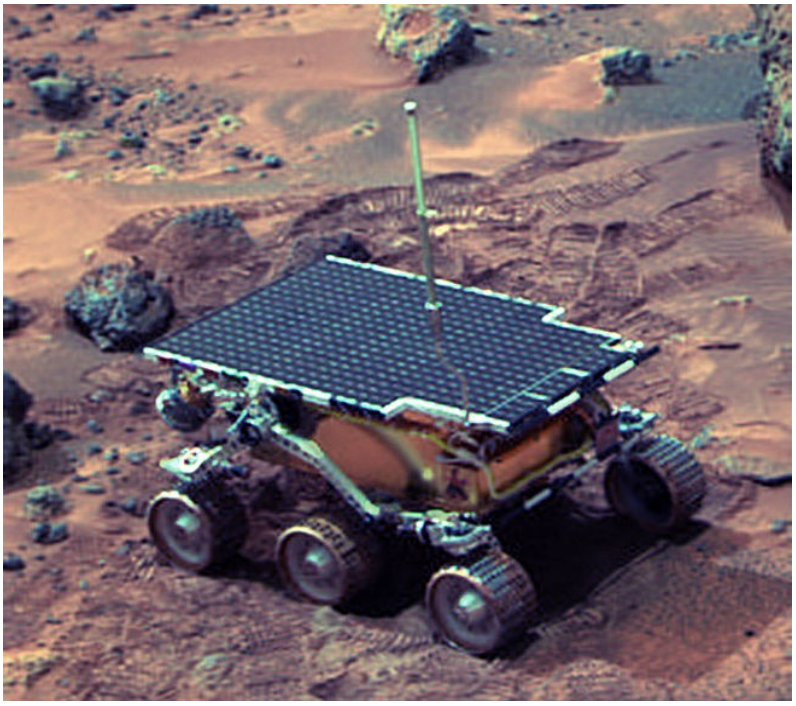

Fig. 8.5 Vehiculul autonom Sojourner

Fig. 8.6 Vehiculul autonom TerraMax

Fig. 8.7 Vehiculul autonom VIAC [43]