CONTRIBUŢII LA NAVIGAŢIA ROBOŢILOR MOBILI AUTONOMI UTILIZÂND REŢELE NEURONALE CELULARE

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul INGINERIE ELECTRONICĂ ŞI TELECOMUNICAŢII
de către

Ing. Gavriluţ Ioan

Conducător ştiinţific:	Prof.dr.ing. Virgil Tiponuţ
Referenţi ştiinţifici:	Prof.dr.ing. Mircea Ivănescu
	Prof.dr.ing. Gavril Toderean
	Prof.dr.ing. Toma Leonida Dragomir

Ziua susţinerii tezei: 26.01.2007

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica" din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Roboţii mobili, denumiţi şi vehicole autonome sau semi-autonome, sunt în continuă expansiune, lucru ce a determinat ca un mare număr de cercetători să fie interesaţi de studiul acestora. Acest domeniu este unul în care se întâlnesc şi îşi aduc aportul diverse domenii de cercetare cum ar fi: manipulatorii robotici convenţionali, inteligenţa artificială, prelucrarea imaginilor, dar mai nou şi domenii cum ar fi biologia şi chiar filozofia.

Un robot mobil autonom este un sistem complex dotat cu o serie de capabilităţi care acţionează în comun. Astfel, sarcinile de navigare într-un spaţiu, perceperea acestuia şi raţionamentul cu privire la mediul înconjurător sunt problemele fundamentale în cadrul studierii roboţilor mobili autonomi.

Cu toate că realizările de până acum, în scopul perfecţionării roboţilor mobili, au reuşit să le imprime acestora anumite nivele de inteligenţă, roboţii sunt încă departe de a fi capabili de o autonomie completă. Cercetătorii sunt în continuare preocupaţi să rezolve problema navigării roboţilor mobili în medii dinamice şi dezordonate fără intervenţia operatorului uman.

În ultimul timp, pe baza comparaţiei cu sistemele biologice, a crescut interesul pentru colectivităţile de roboţi mobili. Fiecare robot este un agent independent şi semi-autonom, care comunică cu toţi ceilalţi roboţi şi are acces în egală măsură la informaţii. Aceste colonii de roboţi pot realiza diverse sarcini colective pe baza comunicării şi cooperării dintre roboţi. S-a observat că utilizarea mai multor roboţi, pentru diverse sarcini, este mai eficientă decât utilizarea unui singur robot sofisticat.

Noile tendinţe din robotică sunt deseori inspirate din natură (comportamentul insectelor, animalelor şi al oamenilor), însă doar progresele tehnicii vor permite transferarea lor în inginerie, într-un mediu funcţional, respectiv implementare practică. Astfel, odată cu progresele făcute în domeniul senzorilor, creşterea puterii de procesare a semnalelor, miniaturizarea părţilor mecanice şi biotehnologia, sisteme uimitoare vor fi posibil de realizat.

Aplicaţiile roboţilor mobili sunt extrem variate, aceştia pătrunzând în ultimii ani în foarte multe domenii. Astfel, roboţii *indoor* (care operează în medii structurate) sunt folosiţi în: industrie (operaţii de manipulare a materialelor), sau pentru diverse servicii (ghizi în muzee şi magazine, operaţii de curăţare şi spălare, supravegherea unor spaţii). Roboţii *outdoor* (care navighează în exteriorul clădirilor) au şi ei aplicaţii dintre cele mai diverse cum ar fi în: construcţii, agricultură, exploatări miniere, inspectarea şi curăţarea reţelelor de canalizare, stingerea incendiilor, operaţii de deminare, exploatări forestiere, cercetări spaţiale.

Reţelele neuronale celulare (CNN), sunt caracterizate de faptul că celulele componente (neuronii) au doar interacţiuni locale. Acest fapt a făcut posibilă implementarea acestora în tehnologia VLSI pe un singur chip. În acest fel, au apărut o serie de procesoare CNN care prezintă avantajul procesării paralele a semnalelor.

Navigarea roboţilor mobili autonomi pe baza imaginilor furnizate de o cameră video a fost studiată în diverse lucrări de specialitate. Utilizarea unei camere video presupune procesarea unei cantităţi mari de informaţie, care de regulă, necesită timp şi poate duce în final la o viteză de navigare scăzută.

Reţelele neuronale celulare au un timp foarte mic de procesare a semnalelor şi din acest motiv sunt considerate o soluţie promiţătoare pentru ghidarea roboţilor mobili autonomi, în medii cu obstacole, pe baza imaginilor.

Autorul

Gavriluţ, Ioan

Contribuţii la navigaţia roboţilor mobili autonomi utilizând reţele neuronale celulare

Teze de doctorat ale UPT, Seria 7 , Nr. 2 , Editura Politehnica, 2007, 196 pagini, 96 figuri, 2 tabele.

ISSN: 1842-7014

ISBN: 978-973-625-417-8

Cuvinte cheie:
roboţi mobili, reţele neuronale celulare, planificarea traiectoriei, algoritmi CNN, mediu integrat

Rezumat,

Utilizarea unei camere video pentru navigaţia roboţilor mobili autonomi presupune procesarea unei cantităţi mari de informaţie care, de regulă, necesită timp şi poate duce în final la o viteză de navigare scăzută. Reţelele neuronale celulare (CNN) au un timp foarte mic de procesare a imaginilor şi din acest motiv sunt considerate o soluţie promiţătoare pentru ghidarea roboţilor mobili autonomi, în medii cu obstacole, pe baza imaginilor.
În prezenta lucrare este abordată problema navigării roboţilor mobili şi a colectivităţilor de roboţi mobili în medii cu obstacole. Este examinat, în special, modul de planificare a traiectoriei pe baza imaginilor mediului furnizate de o cameră video. Aceste imagini sunt procesate cu ajutorul reţelelor neuronale celulare care prezintă avantajul procesării paralele şi pot fi implementate hard.
Pe parcursul a şase capitole se încearcă prezentarea unei soluţii complete, începând cu prezentarea roboţilor mobili, navigaţia roboţilor mobili, prezentarea reţelelor neuronale celulare, elaborarea de algoritmi CNN pentru planificarea traiectoriei roboţilor mobili şi pentru coordonarea unei colectivităţi de roboţi mobili şi terminând cu prezentarea unui mediu integrat pentru navigaţia unui robot mobil prin procesarea cu CNN a imaginilor achiziţionate de o cameră video.

CUPRINS

1. ROBOŢII MOBILI AUTONOMI

1.1. Noţiuni introductive

Roboţii mobili reprezintă un domeniu de cercetare relativ nou, cunoscând o dezvoltare rapidă în ultimii ani. În principiu, se studiază şi perfecţionează comanda vehicolelor autonome şi semi-autonome [1]. Acest domeniu preia din alte domenii de cercetare unele concepte cum ar fi: manipulatorii robotici convenţionali, inteligenţa artificială, prelucrarea imaginilor, etc., care sunt folosite de către roboţii mobili în special pentru perceperea şi modelarea mediului înconjurător. Comportamentul unui robot mobil autonom nu implică doar achiziţia incrementală a poziţiei, estimarea erorilor de poziţie, abilitatea de a recunoaşte obiectele sau locurile familiare şi răspunsul în timp real, ce necesită ca toate aceste funcţionalităţi să acţioneze în comun. Sarcinile de navigare într-un spaţiu, perceperea acestuia şi raţionamentul cu privire la mediul înconjurător sunt problemele fundamentale în cadrul studierii roboţilor mobili autonomi.

Roboţii mobili nu sunt şi nu trebuie să fie doar o colecţie de algoritmi, ci şi întruchipări fizice care operează în medii reale. Acestea trebuie să perceapă lumea reală pe baza senzorilor şi prin intermediul lor sunt testate conceptele şi algoritmii teoretici. Prin urmare, domeniul roboţilor mobili este acela în care literalmente robotul sau vehicolul se deplasează pe drum pe baza unor algoritmi de planificare a traiectoriei, reprezentare a cunoaşterii, percepţie şi raţionament.

Schema bloc de principiu a unui robot mobil autonom este prezentată în Figura 1.1. Se poate observa că un robot mobil este compus dintr-o serie de sub-sisteme care toate împreună participă la elaborarea şi efectuarea comenzilor de deplasare a acestuia în mediul de lucru.

Pe baza informaţiilor despre structura mediului de lucru, obţinute de la sistemul senzorial, şi ţinând cont de informaţiile apriori memorate într-o bază de date, se poate construi o hartă a mediului. Pe baza acesteia sunt determinate poziţiile obstacolelor în raport cu poziţia robotului. Odată cunoscută poziţia şi orientarea robotului în raport cu mediul, se poate planifica în continuare traiectoria acestuia, care va fi comunicată sistemului de comandă a deplasării. Acesta la rândul său va comanda sistemul mecanic de locomoţie al robotului pentru efectuarea deplasării în mediu. Practic, aceste sub-sisteme funcţionează permanent într-o buclă închisă, astfel că dacă unul dintre ele este afectat de anumiţi factori, atunci sistemul de comandă al robotului, luat în ansamblu, nu va funcţiona normal şi operaţia de navigare va fi compromisă.

O mare parte a roboţilor mobili sunt încă în stadiul de experimentare şi testare dar au apărut totuşi aplicaţii importante în domeniul industrial şi comercial în care aceştia înlocuiesc cu succes munca umană. Sarcinile în care roboţii mobili sunt folosiţi cu succes, sunt caracterizate de una sau mai multe din următoarele caracteristici:

- un mediu "ne-ospitalier" în care realizarea operaţiilor de către o fiinţă umană poate fi foarte costisitoare sau foarte periculoasă;

- un mediu îndepărtat pentru care trimiterea unui operator uman poate fi prea dificilă sau poate lua mult timp;
- sarcini care necesită foarte multe operaţii ciclice sau au un factor de oboseală ridicat;
- sarcini care sunt foarte dezagreabile pentru om.

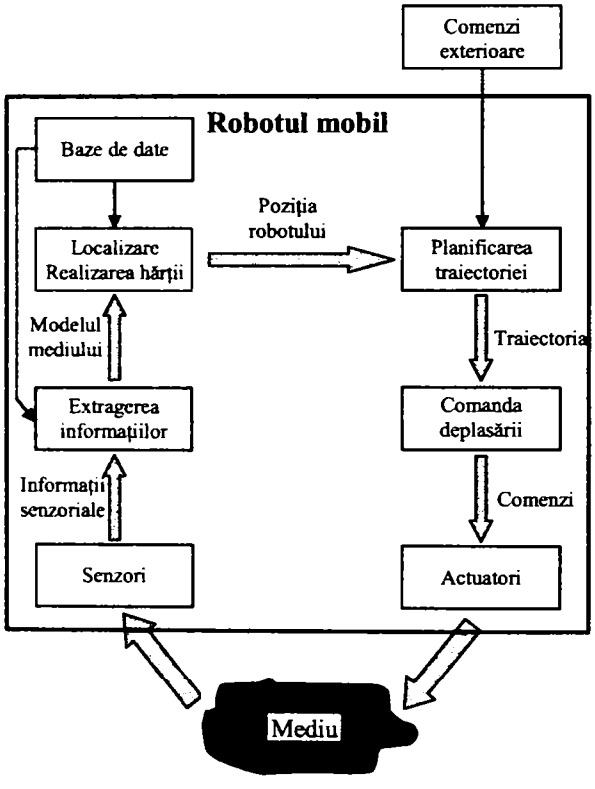


Figura 1.1. Schema bloc de principiu a unui robot mobil autonom şi circulaţia informaţiilor şi comenzilor între sub-sistemele acestuia.

Roboţii mobili sunt, de fapt, o "ispravă" a ingineriei. Actuatorii, procesoarele, interfeţele, senzorii, mecanismele de comunicare şi de locomoţie care permit unui robot mobil să funcţioneze, trebuie să fie integrate astfel încât să permită sistemului să funcţioneze ca un tot unitar. După cum se observă şi din Figura 1.1, robotul mobil are o structură internă complexă, care necesită o investiţie considerabilă de resurse umane şi financiare pentru asigurarea funcţionării sale.

La nivel fizic sau hard, un robot mobil poate fi descompus în următoarele părţi [1]:

- un mecanism cu ajutorul căruia robotul se deplasează prin mediu. Acesta include organizarea fizică a motorului, curele de transmisie şi roţi dinţate necesare pentru punerea în mişcare a robotului;
- un calculator sau o reţea de calculatoare pentru comanda robotului;
- o reţea de senzori prin care robotul culege informaţii referitoare la mediu;

- un sistem hard de comunicare prin care se stabileşte comunicarea robotului cu operatorul de comandă sau cu alte calculatoare exterioare robotului.

Dintr-un alt punct de vedere, detaliile hard pot fi abstractizate şi un robot poate fi considerat astfel:

- o abstracţie la nivel soft a motoarelor, traductoarelor, actuatoarelor, etc., care permit mişcarea robotului. Cei mai mulţi producători de accesorii hard pentru roboţi furnizează doar suport pentru exploatarea acestora şi nu neapărat detalii tehnice despre modul lor de funcţionare;
- mecanisme soft sau biblioteci pentru accesul la modul de funcţionare a senzorilor robotului, spre exemplu, pentru indicaţia curentă a senzorilor ultrasonici despre distanţă;
- un mecanism de comunicaţie standard cum ar fi o interfaţă pentru o reţea distribuită în exteriorul robotului.

Dintr-o perspectivă mai abstractă, roboţii mobili pot fi consideraţi doar la nivel soft, astfel încât: senzorii, sistemele de comunicare şi sistemele de locomoţie sunt văzute simplu, ca module soft care stabilesc interacţiunea robotului cu mediul său. Componentele tipice într-o astfel de arhitectură sunt:

- sub-sistem de comandă şi procesare senzorială;
- sub-sistem de interpretare a informaţiilor senzoriale care decide acţiunile viitoare pe care le va face robotul;
- sub-sistem de comandă a mişcării.

1.2. Clasificarea roboţilor mobili

Roboţii mobili pot fi clasificaţi, în sens general, după zona specifică de activitate [2]. Apare astfel în literatura de specialitate, clasificarea în roboţi destinaţi navigării interioare (*indoor)*, sau mai bine zis, în interiorul clădirilor sau halelor industriale şi roboţi pentru navigarea exterioară, pe teren deschis sau pe străzi (*outdoor*).

Roboţii *indoor* prezintă, faţă de roboţii *outdoor*, un număr însemnat de avantaje, ce simplifică mult munca programatorului precum şi sistemul de comandă necesar:

- mediul de lucru este bine structurat, având de cele mai multe ori forme regulate, cum ar fi, spre exemplu: muchiile şi colţurile drepte;
- în mediul de lucru pot fi adăugate foarte uşor, în mod intenţionat, diverse puncte de reper (*beacons*) care să ajute la ghidarea robotului;
- mediul *indoor* are un grad mare de predictibilitate, astfel sunt situaţii, hale de producţie spre exemplu, în care evoluţia mediului poate fi prezisă cu un grad de acurateţe destul de ridicat;
- condiţiile de lucru sunt mult mai acceptabile. În mediul din interiorul clădirilor temperatura şi umiditatea pot fi controlate şi menţinute între limite rezonabile. De asemenea şi iluminarea poate fi controlată, având un caracter uniform şi constant.

Roboţii *outdoor* trebuie proiectaţi astfel încât să-şi poată desfăşura activitatea într-un mediu complex. Ei trebuie să se deplaseze într-un teren extrem de variat, ale cărui trăsături, de cele mai multe ori, nu se repetă.

În plus, mediul *outdoor* este, în cele mai multe cazuri şi impredictibil. Din aceste motive, sistemul de comandă al robotului trebuie să dispună de o capacitate mare de procesare a informaţiilor senzoriale, care de cele mai multe ori, conduce la o viteză de navigare scăzută.

Pe de altă parte, condiţiile de lucru, condiţii care în unele cazuri pot fi chiar ostile, influenţează destul de mult procesul de navigare. Un robot ce navighează într-un mediu *outdoor* poate fi supus unei plaje de temperaturi extrem de largi, în condiţiile unei umidităţi variabile. La toate acestea se adaugă variaţiile gradului de iluminare care pot fi prezente nu numai în cadrul ciclului zi-noapte, ci chiar şi pe parcursul zilei odată cu evoluţia condiţiilor atmosferice.

Pe de altă parte, priviţi din punctul de vedere al domeniului de aplicabilitate, literatura face distincţie între două mari clase:

- roboţi pentru aplicaţii industriale;
- roboţi pentru servicii.

În ceea ce priveşte roboţii pentru servicii aceştia sunt definiţi ca: "dispozitive cinematice programabile care îndeplinesc servicii semiautomat sau automat" [3]. Pentru termenul servicii, literatura oferă definiţia: "serviciile sunt acţiuni care nu contribuie în mod direct la fabricarea de bunuri, dar reprezintă acţiuni utile pentru oameni sau echipamente".

Pe lângă noţiunile de roboţi industriali şi roboţi pentru servicii a apărut în ultimii ani noţiunea de roboţi personali. Gradul de nestructurare al mediului precum şi gradul de autonomie, creşte de la roboţii industriali spre cei pentru servicii şi roboţii personali. Se consideră că roboţii personali sunt clasa cea mai avansată, capabilă de o comunicare naturală cu mediul. Roboţii personali vor putea, de asemenea, să-şi modeleze singuri mediul în care evoluează şi să manipuleze aceste modele în raţionamente.

O altă clasificare a roboţilor mobili poate fi făcută după sistemul lor de locomoţie, respectiv modul de deplasare a acestora:

- roboţi cu roţi - sunt cei mai utilizaţi, roţile sunt angrenate de cele mai multe ori de motoare de curent continuu;
- roboţi cu şenile - se deplasează pe două şenile paralele care au avantajul aderenţei ridicate comparativ cu roţile;
- roboţi păşitori - sistemul lor de locomoţie îl imită pe cel uman;
- roboţi târâtori - reprezintă o clasă specială a roboţilor mobili care încearcă să copieze modul de deplasare al reptilelor;
- roboţi submersibili - sunt roboţii ce lucrează în mediul subacvatic;
- roboţi zburători - sunt dedicaţi automatizării sistemului de pilotaj al elicopterelor;
- roboţi spaţiali - pot fi incluşi cei geostaţionari a căror mijloace de deplasare sunt bazate pe propulsoare cu jet.

1.3. Evoluţia roboţilor mobili

Roboţii mobili au evoluat continuu, în decursul anilor, de la mecanisme ce realizau operaţii simple şi de multe ori repetitive la mecanisme complexe ce pot naviga autonom în medii nestructurate (Figura 1.2). Pentru început au apărut manipulatoarele robotice iar apoi roboţii au progresat gradual în roboţi programabili,

roboţi comandaţi de calculator, roboţi "îndemânatici" având un grad mare de libertate şi roboţi cu mai multe braţe care cooperează [4].

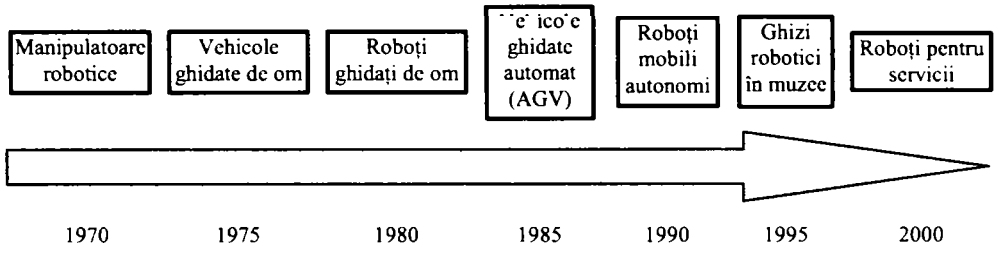


Figura 1.2. Evoluţia roboţilor mobili.

În funcţie de comanda mişcării roboţilor, aceştia au avansat, de asemenea, de la tipuri de roboţi ghidaţi de om, la roboţi ghidaţi automat, roboţi ce se deplasează singuri, până la platforme mobile multiple care pot coopera reciproc pentru execuţia sarcinilor. Primele manipulatoare mobile erau comandate de către oameni, care trebuiau să aibă o anumită îndemânare. Apoi au apărut vehicolele ghidate automat (AGV - ***A**utomated **G**uided **V**ehicles*), care aveau unele capabilităţi de manipulare a materialelor într-o hală industrială. Aceste vehicole se deplasau pe traiectorii bine stabilite şi realizau în general operaţii de încărcare şi descărcare. Cei mai avansaţi roboţi sunt cei care se pot deplasa singuri şi au capabilităţi de manipulare a materialelor şi pe lângă acestea ei pot singuri să ia unele decizii cu privire la operaţiile ce trebuie executate.

În ultimul timp, pe baza comparaţiei cu sistemele biologice, a crescut interesul pentru colectivităţile de roboţi mobili. Fiecare robot din grup poate fi proiectat ca un agent independent şi semi-autonom, care comunică cu toţi ceilalţi roboţi şi are acces în egală măsură la informaţii. Deciziile cu privire la planificarea traiectoriilor sunt luate de regulă de către un supervizor (calculator).

Realizările de până acum, în scopul perfecţionării roboţilor mobili, au reuşit să le imprime acestora anumite nivele de inteligenţă. Totuşi, nivelul de inteligenţă atins de roboţii mobili este încă departe de a fi suficient pentru obţinerea unei autonomii complete care să nu necesite intervenţia umană în cazul realizării unei sarcini date. Cercetătorii în domeniu sunt în continuare preocupaţi să rezolve problema operării roboţilor în medii dinamice şi dezordonate fără intervenţie umană.

Unul din primele proiecte care au introdus inteligenţa artificială pentru comanda roboţilor mobili a fost robotul Shakey construit la Institutul de Cercetare din Stanford în 1967. Robotul avea legătură radio cu un computer şi era programat în FORTRAN. Deşi era o realizare de excepţie, la acea dată, robotul era departe de ceea ce am putea numi complet autonom.

Imediat după aceasta, au fost demarate mai multe proiecte având ca bază de studiu roboţii mobili. Următorul ţel propus de către cercetători a fost realizarea unui robot care să imite mersul uman. „Hexapod"-ul construit la Universitatea statului Ohaio este considerat ca fiind primul sistem de acest gen din lume. Robotul era echipat cu un motor electric care să poată genera toate tipurile de mişcări de care robotul era capabil şi opera în mod *maste-slave*, fiind legat de calculator cu un "cordon ombilical". Provocarea pe care acest robot a trebuit să o înfrunte era menţinerea unei mişcări uniforme a corpului în timp ce picioarele execută mişcări complexe [5].

Unul din roboţii mobili, care au devenit referinţă, a fost construit de către H. R. Everett [6]. A fost unul din primii roboţi dotaţi cu senzori multipli şi care era pe deplin controlat de computerul din dotare. Scopul principal al acestui robot denumit Robart I era de a servi ca mobil de test pentru cercetări în domeniul Inteligenţei Artificiale şi al altor domenii adiacente. În cadrul roboţilor de acest tip din a doua generaţie, numiţi Robart II, au fost integrate multe din caracteristicile atât de comune acum roboţilor mobili.

La realizarea roboţilor mobili sau a colectivităţilor de roboţi mobili trebuie realizat un compromis între gradul de autonomie al acestora şi complexitatea sarcinii pe care aceştia o îndeplinesc. De regulă, tendinţa firească în domeniu este obţinerea unei autonomii cât mai mari a roboţilor sau a colectivităţii de roboţi luată în ansamblu şi în acelaşi timp aceştia să realizeze sarcini cât mai complexe.

Desigur, obiectivele urmărite diferă în funcţie de modul de ghidare a robotului sau a grupului de roboţi precum şi de destinaţia acestora (aplicaţii industriale sau alte aplicaţii care necesită un grad de autonomie mai ridicat) şi nu în ultimul rând în funcţie de complexitatea sarcinii. Stadiul actual şi tendinţa cercetării respectiv evoluţiei în domeniul roboţilor mobili în funcţie de aplicaţiile acestora sunt prezentate în Figura 1.3.

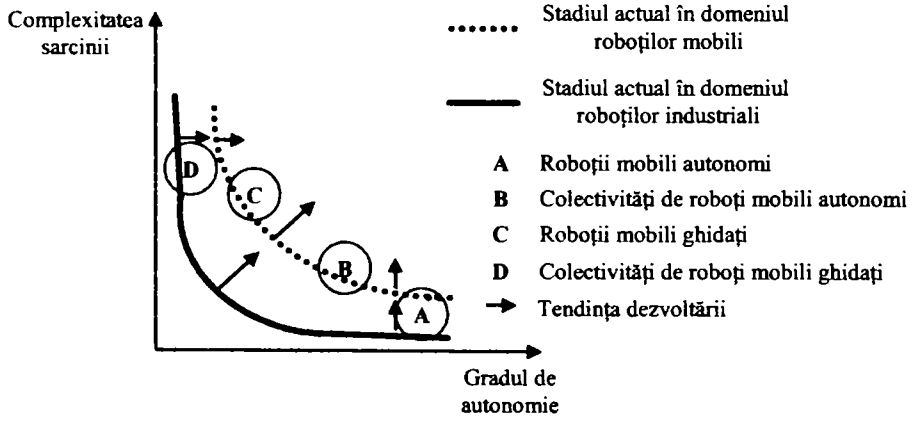


Figura 1.3. Stadiul actual şi tendinţele dezvoltării roboţilor mobili în funcţie de complexitatea sarcinii, gradul de autonomie şi aplicaţiile acestora.

Una din tendinţele de perfecţionare a roboţilor mobili o constituie utilizarea limbajului natural, respectiv vorbirii, pentru comanda acestora, sau chiar pentru comunicarea lor cu un supervizor central.

Utilizarea limbajului natural reprezintă cel mai confortabil mod de a comunica cu roboţii. Astfel, încă din 1999 a fost demarat proiectul CARL (***C**ommunication, **A**ction, **R**easoning and **L**earning in Robotics*) [7], în cadrul căruia este propusă o interfaţă om-robot prin intermediul căreia operatorul uman poate să comunice cu un robot mobil prin vorbire. Se propunea realizarea unui dialog cu robotul prin care operatorul uman să-i comunice acestuia ce sarcină să îndeplinească, iar robotul la rândul său să se poată informa despre modul de realizare a unei acţiuni particulare atunci când este nevoie. De asemenea, se urmărea ca operatorul uman să îl poată atenţiona verbal pe robot despre anumite pericole ce pot apărea în timpul realizării unor operaţii.

O altă tendinţă, care a început să se concretizeze, o constituie realizarea de roboţi în miniatură [8]. Având dimensiuni de doar câţiva cm^3 ei pot genera uneori o forţă comparabilă cu forţa aplicată de un operator uman. Partea electronică a acestor roboţi este realizată cu ajutorul dispozitivelor electronice montate pe suprafaţa cablajului (SMD - ***S**urface **M**ounted **D**evices*), care permit o creştere importantă a densităţii componentelor pe cablajele imprimate. De asemenea, sunt utilizaţi senzori compacţi ce includ unele preprocesări ale semnalelor sau aşa zişi senzori inteligenţi (*smart sensors*) şi astfel este redus numărul de conductoare şi de module.

1.3.1. Colectivităţi de roboţi mobili

În ultimii ani, eforturile majore de cercetare au fost focalizate pe îmbunătăţirea performanţelor unui robot mobil individual prin utilizarea senzorilor şi actuatorilor avansaţi, şi a algoritmilor de comandă inteligenţi. Acest fapt se datorează în principal necesităţii de a realiza sarcini tot mai complexe cerute de practică. Prin urmare, robotul mobil individual a devenit un dispozitiv foarte sofisticat.

Multe dintre sistemele robotice curente folosesc de regulă, pentru aplicaţii la distanţă platforme sofisticate şi scumpe, tipic roboţi mari şi mijlocii desfăşuraţi ca o singură unitate, fiind înzestraţi adesea cu instrumentaţie complexă şi/sau echipe de ingineri pentru diverse operaţii sau service. Aceşti roboţi au costuri mari de realizare, sunt greu de transportat şi nu oferă un răspuns rapid în cazul sarcinilor la distanţă şi pe lângă acestea, adesea, aceşti roboţi nu pot fi recuperaţi datorită expunerii lor la diverse pericole.

Pornind de la aceste premise, în ultimii ani, colectivităţile de roboţi denumite şi colonii de roboţi sau sisteme multi-robot, au fost mult studiate şi chiar implementate în unele domenii cum ar fi: transportul materialelor (în special a materialelor sau obiectelor grele), operaţii de explorare, căutare şi supraveghere, aplicaţii industriale de curăţare şi spălare, etc.

Avantajele utilizării echipelor de roboţi a atras atenţia cercetătorilor mai ales în ultimii ani, fapt observat prin numeroase lucrări ce au apărut în domeniu [9],[10],[11],[12],[13],[14]. Ideea de bază este realizarea unor grupuri de roboţi mobili care să lucreze în cooperare pentru execuţia diverselor tipuri de sarcini. Deşi având capacităţi limitate, astfel de roboţi realizaţi în număr mare pot reprezenta o forţă cumulativă puternică precum o colonie de furnici sau un roi de albine. De asemenea, utilizarea roboţilor multipli va face să crească siguranţa în funcţionarea sistemului în timp ce scade complexitatea sarcinilor pentru un robot precum şi timpul de execuţie total al unei sarcini complexe.

În multe situaţii, utilizarea sistemelor multi-robot prezintă o serie de avantaje faţă de utilizarea unui singur robot, cum ar fi:

- sunt realizate unele sarcini complexe ce nu pot fi realizate de către un singur robot;
- creşte productivitatea prin lucrul în echipă, respectiv prin efectuarea mai multor operaţii complementare în paralel;
- e mai economic uneori, să se realizeze câţiva roboţi de mici dimensiuni decât un singur robot complex care să execute aceleaşi operaţii;
- sistemul multi-robot poate fi mai robust şi mai sigur, datorită redundanţei de care acesta poate dispune;

- se poate obţine o mai bună distribuţie spaţială, deoarece mai mulţi roboţi pot acoperi concomitent o zonă mai mare, în cazul operaţiilor de explorare, căutare sau supraveghere.

Implementarea şi coordonarea coloniilor de roboţi mobili este desigur mult mai complexă. Pentru realizarea unei sarcini colective sistemul de comandă al echipei de roboţi trebuie să fie proiectat astfel încât să rezolve cel puţin două probleme importante:

- descompunerea şi alocarea sarcinilor;
- interacţiunea şi colaborarea dintre roboţi.

1.4. Aplicaţii ale roboţilor mobili

Aplicaţiile roboţilor mobili sunt extrem variate, aceştia pătrunzând în ultimii ani în foarte multe domenii. Astfel, roboţii *indoor* care operează în medii structurate sunt folosiţi în: industrie (operaţii de manipulare a materialelor), sau pentru diverse servicii (ghizi în muzee şi magazine, operaţii de curăţare şi spălare, supravegherea unor spaţii). Roboţii *outdoor* au şi ei aplicaţii dintre cele mai diverse cum ar fi în: construcţii, agricultură, exploatări miniere, inspectarea şi curăţarea reţelelor de canalizare, stingerea incendiilor, operaţii de deminare, exploatări forestiere, cercetări spaţiale.

Sunt prezentate în continuare câteva din aplicaţiile reprezentative ale roboţilor mobili, aplicaţii grupate pe domenii economice, subliniind şi unele exemplificări în acest sens.

1.4.1. Aplicaţii în domeniul serviciilor

În 1990 firma Putzmeister în cooperare cu Stuttgart Fraunhofer Institute au realizat cu succes cel mai mare robot mobil la acea dată, Skywash SW 33, destinat spălării aeronavelor [3]. Sistemul era, de fapt, un manipulator gigant cu 9 axe. Caracteristicile sale definitorii sunt: greutatea maximă admisă a sarcinii 1500 kg, o întindere maximă a braţului de 22 m, o precizie de poziţionare de 15 mm şi o viteza maximă de deplasare a efectorului final de 1,5 m/s. Arhitectura sistemului se bazează pe un concept hibrid în care acţiunea este iniţiată de către un operator uman, ce realizează poziţionarea pe suprafaţa avionului, urmând ca apoi în regim automat să fie executată operaţiunea de spălare pe aproximativ 80% din suprafaţa aeronavei. Skywash are avantajul scurtării timpului necesar spălării unei aeronave de la aproximativ 10-12 ore câte erau necesare pentru 10 oameni, la 2-3 ore, timp în care robotul efectua singur aceeaşi operaţie.

Una din aplicaţiile testate a fost aceea de curăţire a bazinelor de apă, în special a celor comerciale care trebuie curăţate zilnic. Pe fundul bazinului şi pe pereţii acestuia se depun alge şi diverse sedimente care trebuie înlăturate iar curăţarea manuală pe lângă faptul că ia mult timp este şi neplăcută. Astfel a fost realizat robotul mobil WEDA B400 [15], care echipat cu o pompă de apă şi perii rotative poate să cureţe suprafaţa bazinului în mod automat. Algele şi sedimentele sunt colectate într-un sac prin care apa poate să treacă. Pentru localizarea robotului sunt folosiţi senzori ultrasonici şi filtrare Kalman a datelor senzoriale [16].

Altă aplicaţie utilă a roboţilor mobili este aceea de ghizi în locuri publice cum ar fi muzeele. În 1997 a fost instalat în Muzeul German din Bonn, primul robot ghid Rhino, care era echipat cu senzori laser şi se ghida după o hartă obţinută manual,

deoarece nu avea capabilitatea de a ridica *on-line* harta mediului. Mai apoi în anul 1998 a apărut robotul mobil interactiv Minerva [17], care a fost instalat în Muzeul Naţional de Istorie Americană, echipat cu un sistem de navigare mult mai sofisticat. Pe lângă senzorii laser, ultrasonici şi vizuali, robotul deţinea şi o unitate de comunicare prin voce, panou de comandă prin atingere, server de internet, etc. Vizitatorii puteau selecta zona de vizitare prin atingerea panoului situat în spatele robotului. Viteza maximă de deplasare era de 163 cm/sec dar în prezenţa turiştilor era de maxim 70 cm/sec. Poziţia sa era actualizată de câteva ori pe secundă cu ajutorul unui program de simulare scris în Java şi pe baza comunicării prin internet cu robotul mobil.

Roboţii mobili sunt folosiţi şi pentru asistenţa persoanelor cu handicap [3]. Un astfel de sistem aduce beneficii importante acestor categorii de persoane. Au fost concepute cărucioare mobile cu un braţ, utilizate în general pentru acţiuni de prindere şi apropiere a obiectelor, spre exemplu la ridicarea unui pahar, sau pentru a deschide o uşă. Problema majoră a acestor tipuri de roboţi este modul de interfaţare cu persoana handicapată. Pentru iniţierea comenzilor au fost gândite sisteme cu joystick, cu tastatură, prin voce sau bilă spaţială. Iar pentru semnalizare s-au conceput sisteme de avertizare prin voce, afişare text sau semne grafice.

Pentru instituţii, problema supravegherii pe timp de noapte poate fi lăsată pe seama unor roboţi de patrulare [18]. Un număr oarecare de roboţi colindă pe culoarele instituţiei, avertizând un operator uman în situaţiile critice. În multe cazuri sunt folosiţi senzori olfactivi, înlăturându-se astfel necesitatea unui program care să specifice ordinea de patrulare. Roboţii de supraveghere lasă în urma lor o dâră volatilă astfel că, aflat într-o intersecţie de coridoare, un robot va alege culoarul pe care simte cel mai puţin substanţa volatilă. Se asigură astfel o supraveghere uniformă şi cvasi-aleatoare a spaţiului.

Conducerea autovehiculelor de către roboţi, a beneficiat de o atenţie sporită în ultima perioadă. Principalele sarcini ale acestor sisteme *outdoor* sunt: recunoaşterea traiectoriei ce trebuie urmată şi în acelaşi timp evitarea oricărui obstacol apărut în cale. Este posibilă o clasificare a sistemelor de conducere automată a autovehiculelor după locul unde se face navigaţia. Astfel, sistemele dedicate deplasării pe şosele sunt denumite *on-road*, iar cele dedicate navigării în teren deschis *off-road* [19]. Sistemele *on-road* sunt, de fapt, vehicule ghidate performante, ce se orientează după liniile de demarcaţie ale carosabilului. Acestea sunt capabile să se descurce (pentru un interval scurt de timp) şi în situaţiile în care aceste marcaje dispar cum ar fi în cazul intersecţiilor. Există la ora actuală, sisteme capabile să piloteze un autovehicul, în mediu natural, cu viteze de până la 100 km/oră [20]. Tot legat de conducerea automată a autovehiculelor, apare problema parcării automate. O serie de mari fabricanţi de automobile studiază această problemă sperând că într-un viitor nu prea îndepărtat o vor putea oferi ca şi facilitate.

Mercedes a introdus conducerea automată a vehicolelor pe autostrăzi pe baza procesării imaginilor. Un astfel de sistem poate intra pe piaţă, în primă fază, ca un mecanism de atenţionare pentru şoferii cu un grad de oboseală ridicat. S-a reuşit conducerea pe o distanţă de 2500 km a unui camion cu 80% autonomie. Totuşi siguranţa rutieră şi unele aspecte legale rămân încă nerezolvate.

Explorarea spaţiilor greu accesibile sau periculoase este unul din domeniile în care roboţii mobili îşi dovedesc din plin utilitatea. Roboţii cu această aplicabilitate fac parte în general din clasa roboţilor telecomandaţi. Normele în vigoare în statele industrializate prevăd anumite moduri stricte de stocare şi supraveghere a substanţelor toxice reziduale. Aceste depozite trebuiesc inspectate periodic pentru a

preîntâmpina degradarea containerelor. De regulă, inspecţia se face vizual, expunând totuşi operatorul uman. Există dispozitive dotate cu senzori de gaz, radianţi sau chimici, care pot sesiza o creştere a factorului nociv. La detectarea unei astfel de situaţii operatorul uman trebuia să pătrundă în perimetrul afectat şi să inspecteze. Au fost construite aşadar, sisteme de inspecţie automată a depozitelor pentru substanţe toxice. Se elimină astfel prezenţa operatorului uman în interiorul zonei periculoase, beneficiind în plus de posibilitatea unei inspecţii pe tot parcursul celor 24 de ore. Rămâne însă necesară prezenţa operatorului uman în fazele de decizie.

Au fost concepute, de asemenea, sisteme de inspecţie şi de executare a unor lucrări de întreţinere în interiorul sistemelor de canalizare [3].

Inspecţia tancurilor petroliere poate fi făcută folosind serviciile unui robot mobil teleghidat. Inspecţia se face vizual de către operator, urmărind imaginile transmise de robot. Principalul avantaj este cel al economiei de timp. Pentru o inspecţie clasică sunt necesare circa 2 săptămâni, timp în care tancul trebuie golit, curăţat şi apoi cercetat. În cazul sistemului de inspecţie Neptun [21], aceste operaţii nu mai sunt necesare, robotul lucrând în imersie, inspecţia se poate face fără a scoate tancul din circuitul economic.

1.4.2. Aplicaţii industriale

Dintre aplicaţiile industriale ale roboţilor mobili, aprovizionarea şi transportul de materiale sunt cel mai des întâlnite. De cele mai multe ori, aprovizionarea fluxului de producţie într-o fabrică modernizată, se face cu ajutorul roboţilor mobili. În general aceşti roboţi sunt vehicule ghidate automat. Pentru cazul unui depozit obişnuit, ce nu prezintă facilităţi de ghidare, problema conducerii roboţilor de transport se poate rezolva utilizând senzori olfactivi. Operatorul trasează o dâră volatilă între locul în care este depozitată marfa şi rampa la care aceasta urmează să fie distribuită. Dacă necesităţile impun schimbarea traseului înaintea volatilizării complete a substanţei existente, aceasta se poate realiza prin spălarea pardoselei.

Alte aplicaţii în acest domeniu sunt ghidarea automată a tractoarelor industriale şi agricole precum şi aplicaţii în cazul mineritului şi a instalaţiilor de foraj. Pe lângă acestea sunt şi aplicaţii în care procesul este supervizat de om, în cazul în care operaţiile complet autonome sunt considerate riscante, cum ar fi manipularea containerelor de marfă sau teleoperarea în cazul lucrărilor de exploatare forestieră [22].

Roboţii mobili sau denumiţi uneori vehicule inteligente, au apărut şi vor apărea în aplicaţii în care siguranţa în funcţionare poate fi asigurată. Spaţiile pentru astfel de aplicaţii sunt acelea în care este interzisă intrarea personalului uman în timpul "lucrului" şi astfel este eliminat riscul apariţiei unor accidente.

Pentru transportul obiectelor grele firma LIFTEC a realizat un camion cu ghidare automată care poate încărca, transporta şi descărca automat o sarcină maximă de 90 tone, având viteza maximă de deplasare 1,33 m/s [22].

În exploatările miniere, vehicolele autonome pot fi utilizate cu succes deoarece în galeriile de mină nu intră personal uman, condiţiile de iluminare sunt constante iar pereţii sunt în majoritatea timpului aproape de vehicol şi pot constitui repere pentru acesta.

1.4.3. Aplicaţii în agricultură

Cu toate că în industrie şi servicii există un număr impresionant de aplicaţii ale roboţilor mobili, numărul acestora în sectorul agricol este mult mai mic. Teoretic

este posibilă programarea unui robot mobil pentru o serie de activităţi din agricultură cum ar fi: transplantarea, udarea, săparea şi chiar recoltarea selectivă, dar practic complexitatea lor face acest lucru extrem de dificil. Contrar aplicaţiilor din industrie care sunt simple, repetitive, bine definite şi apriori cunoscute, în agricultură robotul trebuie să negocieze cu un mediu nestructurat, incert şi impredictibil.

Dintre problemele cu care se confruntă un sistem autonom dedicat lucrărilor agricole, se evidenţiază următoarele:

- trebuie să opereze într-un spaţiu tridimensional în continuă schimbare;
- localizare aleatoare a ţintelor (fructelor, legumelor, etc.);
- varietate mare a dimensiunilor, formei, culorii şi texturii fructelor sau legumelor;
- produsele manipulate sunt delicate;
- condiţii ambientale schimbătoare, (lumina, căldura, etc.);
- condiţii ambientale ostile: praf, noroi, temperaturi extreme şi umiditate.

1.4.4. Aplicaţii în medicină

Apariţia mecatronicii, care reprezintă de fapt, o îmbinare a mecanicii cu electronica, la scara micro-universului, a făcut posibilă dezvoltarea unor instrumente medicale deosebite. O realizare a acestui domeniu o reprezintă micro-roboţii pentru explorarea colonului.

Un sistem pentru manipularea unei camere video, a fost conceput pentru a înregistra acţiunile unei operaţii [23]. A fost realizată o structură de manipulator cu şase grade de libertate, la care s-a ataşat o cameră video. Numărul mare de grade de libertate permite poziţionarea arbitrară a camerei, poziţionare care se face prin telecomandă. Calculatorul ataşat sistemului permite implementarea unor funcţii specifice, cum ar fi de pildă mişcarea camerei pe o sferă în jurul obiectului studiat cu păstrarea focalizării într-un punct dorit sau memorarea unor poziţii. Tehnic problema este simplă, dar s-a urmărit un grad mare de flexibilitate a sistemului în funcţie de cerinţelor beneficiarilor.

În [24] şi [25] se prezintă un mediu integrat pentru asistarea deplasării persoanelor cu deficienţe de vedere, care utilizează diferiţi senzori pentru planificarea locală a traiectoriei şi sisteme GPS (***G**lobal **P**ositioning **S**ystem*) şi GIS (***G**eographical **I**nformation **S**ystem*) pentru localizare.

1.4.5. Aplicaţii în cercetarea spaţială

Prima clasă a acestui domeniu o constituie roboţii spaţiali ce evoluează pe o orbită circumterestră. Cercetările s-au îndreptat cu precădere asupra manipulatoarelor (conduse prin teleoperare) montate pe platforme spaţiale [26]. Nu au fost neglijaţi nici roboţii mobili liberi de tipul *free-flying*. În viitor se speră că multe din acţiunile de pe sateliţi vor fi executate de către roboţi.

Cea dea doua clasă a roboţilor spaţiali este cea consacrată explorărilor planetare. În acest scop se urmăreşte construirea unor roboţi care să satisfacă condiţiile particulare ale viitorului mediu de lucru. Rolul lor este acela de a studia suprafaţa planetelor, de a instala instrumente şi de a recolta mostre pentru analiză locală sau pentru o eventuală expediere pe pământ. Se vizează explorarea planetei Marte [18].

Caracteristica principală pentru acest tip de roboţi este gradul ridicat de autonomie. Trebuie să fie capabili să identifice zonele de interes ştiinţific, să poată să-şi programeze acţiunile şi să posede o mare abilitate de navigare. Toate aceste acţiuni trebuiesc făcute în condiţiile unui control riguros al resurselor. Aşadar, problema majoră a roboţilor destinaţi explorărilor planetare o reprezintă sursa de energie.

Cercetările în acest sens au evidenţiat două căi pentru soluţionarea problemei consumului de energie:

- utilizarea roboţilor mobili de mici dimensiuni (roboţi Rover);
- utilizarea roboţilor păşitori, de dimensiuni normale.

Pentru prima variantă, cea a roboţilor Rover, calea principală pe care s-a încercat micşorarea consumului energetic a fost cea a găsirii unor noi metode de fabricaţie care să ducă la porţi logice CMOS mai eficiente. Cu acelaşi obiectiv în vizor se încearcă, de asemenea, scăderea tactului de lucru al procesoarelor. În paralel se încearcă dezvoltarea unui nou limbaj capabil să ruleze pe procesoare lente.

În contrast, cea de-a doua soluţie porneşte de la ideea că orice contact cu mediul duce la un consum suplimentar de energie şi încearcă să reducă consumul energetic prin minimizarea numărului lor. Având ca suport un robot cu şase picioare, algoritmii implementaţi caută soluţia optimă a deplasării, atât din punct de vedere al distanţei cît şi din punct de vedere al numărului de contacte cu solul.

1.4.6. Aplicaţii în domeniul cercetărilor subacvatice

Suprafaţa pământului fiind acoperită în proporţie de 70% cu apă, apare în mod firesc ideea conceperii unor roboţi mobili pentru studiul spaţiului subacvatic. Denumirea întâlnită în literatura de specialitate în legătură cu roboţii mobili submersibili este *underwater robotic vehicles* (URV) [20].

Câteva dintre posibilele aplicaţii ale acestor roboţi submarini sunt: cercetări oceanice, recuperări de mine, construirea şi întreţinerea structurilor subacvatice. Numărul acestor aplicaţii este restrâns din cauza costurilor ridicate care se datorează necesităţii menţinerii în zonă a unei nave care să permită telecomandarea robotului.

Pentru a înlătura prezenţa teleoperatorului, soluţia este dată de construirea unor vehicule subacvatice autonome (AUV - ***A**utonomous **U**nderwater **V**ehicles*) [20]. Cercetări pe această direcţie, în ţări ca Rusia, Japonia şi SUA au condus la sisteme subacvatice autonome capabile să lucreze la adâncimi de până la 6.000 m.

1.5. Perspective în dezvoltarea roboţilor mobili

În prezent se constată o înclinaţie clară către roboţii care se adaptează, învaţă şi interacţionează în mod constant cu mediul înconjurător, cu persoana care îi foloseşte şi cu alţi roboţi. Această varietate de interacţiuni va genera un comportament superior roboţilor industriali.

Se preconizează că în viitor vor apărea, în mod sigur, un număr important de aplicaţii revoluţionare care nu par acum realizabile cu ajutorul roboţilor mobili. Astfel, colectivităţile de roboţi vor ajunge să îndeplinească misiuni intergalactice sau în adâncurile oceanelor, şi poate chiar în interiorul corpului uman. Roboţi mobili de dimensiuni minuscule se presupune că vor putea fi realizaţi pentru deplasarea în interiorul venelor sanguine pentru căutarea şi îndepărtarea unor tumori.

În toate aceste genuri de misiuni redundanţa şi adaptabilitatea sistemului este mult mai importantă decât inteligenţa pe care fiecare membru în parte o posedă. Precum în natură, inteligenţa sistemului va lua naştere prin multiple interacţiuni cu mediul.

O altă specie de roboţi sunt cei care în curând îi vom găsi în mijlocul nostru ajutându-ne, servindu-ne şi al căror rol va creşte în importanţă cu fiecare zi. Pentru a se ajunge la acest stadiu, nivelul de inteligenţă al acestor roboţi va trebui să se ridice mult deasupra celui al actualei generaţii.

Metodele convenţionale din robotică se leagă de noi idei sau tendinţe deseori inspirate din natură (comportamentul animalelor şi al oamenilor), însă doar progresele tehnicii vor permite transferarea lor în inginerie, într-un mediu funcţional, respectiv implementare practică. În mod sigur numărul aplicaţiilor sistemelor robotice distribuite va creşte rapid odată cu evoluţiile din domeniul tehnicii. Dacă se adaugă la progresele făcute în domeniul senzorilor sau al puterii de procesare, miniaturizarea părţilor mecanice şi biotehnologia, sisteme uimitoare vor fi posibil de realizat în viitor. Astfel, în câţiva ani, diverse societăţi de roboţi vor ieşi din laboratoarele de cercetare şi vor pătrunde în viaţa de zi cu zi. Acestea vor fi folosite în diferite aplicaţii ce includ sarcini de curăţare, monitorizare, transport de materiale etc.

1.6. Concluzii

Roboţii mobili autonomi au cunoscut o dezvoltare rapidă în ultimii ani. Au apărut astfel o serie de implementări soft a acestora, prin diverse programe de simulare dar mai ales realizări fizice în domeniu şi anume roboţi mobili care sunt capabili să navigheze autonom în medii nestructurate. Pe lângă procesul de navigare unii dintre aceştia sunt înzestraţi cu capabilităţi de comunicare cu alţi roboţi sau cu operatorul uman, în cele mai variate moduri.

Multe din noile tendinţe şi idei, apărute în robotică, sunt inspirate din natură respectiv din modul de percepţie şi acţiune al animalelor şi chiar din comportamentul uman. Cele mai mari eforturi în domeniul roboticii mobile au drept scop înzestrarea roboţilor cu o inteligenţă cât mai apropiată de cea umană.

Pe măsură ce va creşte nivelul de inteligenţă al roboţilor mobili, o altă specie de roboţi (roboţii pentru servicii) vor apărea treptat în mijlocul nostru pentru a ne ajuta în viaţa de zi cu zi la diverse operaţii şi vor ajunge să fie chiar indispensabili vieţii cotidiene.

Un interes deosebit a fost acordat în ultima vreme coloniilor de roboţi care pot realiza diverse sarcini colective prin comunicarea şi cooperarea dintre roboţi. S-a observat că utilizarea mai multor roboţi, relativ ieftini, poate fi mai eficientă decât utilizarea unui singur robot sofisticat în cazul realizării unor sarcini complexe.

Deşi colectivităţile de roboţi mobili prezintă o anumită redundanţă, acest fapt constituie, de multe ori, un avantaj prin faptul că un robot devenit nefuncţional poate fi înlocuit într-un timp scurt şi astfel misiunea echipei de roboţi poate continua.

2. NAVIGAŢIA ROBOŢILOR MOBILI AUTONOMI

2.1. Noţiuni introductive

Navigarea este sarcina primordială care trebuie realizată de către roboţii mobili autonomi. Aceasta presupune o succesiune de operaţii care trebuiesc realizate de către sistemul de comandă al robotului începând de la "culegerea" informaţiilor despre mediu şi recepţionarea sarcinii de navigare ce trebuie realizată, până la comanda efectivă a sistemului locomotor cu care este echipat robotul.

În general, ţinta ce trebuie atinsă de către robot nu este direct accesibilă sau uneori chiar trebuie găsită, aşa că navigarea trebuie să fie un proces incremental de determinare şi atingere a ţintelor intermediare potrivite, cu scopul de a se obţine traiectoria optimă spre destinaţia finală.

Pornind de la aceste observaţii, navigaţia unui robot mobil poate fi descompusă în cinci etape principale [27]:

- **Percepţia mediului şi a sarcinii** – în funcţie de obiectivele fixate şi de dificultăţile prezente, sistemul de comandă al robotului trebuie să determine ce informaţii îi sunt necesare pentru îndeplinirea sarcinii;
- **Modelarea mediului** – datele preluate de la senzori şi de la sistemul ierarhic superior sunt utilizate apoi pentru construirea modelului mediului;
- **Localizarea** – pentru a şti unde să se deplaseze, robotul trebuie mai întâi să-şi localizeze poziţia şi orientarea în raport cu mediul şi cu destinaţia finală, dacă este posibil;
- **Planificarea traiectoriei** – în această etapă trebuie determinată, prin diverse metode (locale sau globale), traiectoria optimă spre destinaţie, evitându-se obstacolele întâlnite în cale;
- **Executarea deplasării** – robotul se va deplasa efectiv pe traiectoria impusă, dar dacă senzorii cu care este echipat acesta detectează un obstacol neaşteptat, robotul îl va ocoli sau se va opri şi va solicita o nouă planificare a traiectoriei. Pentru evitarea acestor situaţii care de regulă consumă timp, planificarea traiectoriei trebuie să prezinte o anumită flexibilitate iar execuţia mişcării trebuie făcută astfel încât să se încerce pe cât posibil adaptarea la schimbările locale survenite în mediu.

Ultimele două etape, prezentate mai sus, constituie problema generării mişcării cu evitarea obstacolelor. Această problemă a fost studiată pe două direcţii distincte: planificarea traiectoriei şi deplasarea "reactivă". În cazul metodei planificării traiectoriei (planificare globală), traiectoria este planificată şi fixată, în întregime, înainte de execuţia deplasării, iar metoda "reactivă" (planificare locală) determină o deplasare "sigură", pe baza datelor senzoriale obţinute în timpul execuţiei (*online*).

2.2. Metode şi strategii de navigare

În literatura de specialitate sunt descrise o mulţime de metode şi strategii de navigare pentru roboţii mobili autonomi. În funcţie de sarcina ce trebuie îndeplinită şi condiţiile existente va fi aleasă una din aceste metode respectiv strategii.

2.2.1. Metode de navigare

Metodele de navigare a roboţilor mobili pot fi împărţite în trei categorii mari [28]:

- metode globale;
- metode locale;
- metode hibride.

În cazul **metodelor globale** se cunosc informaţiile referitoare la structura mediului de lucru al roboţilor, care sunt prezentate sub forma unei hărţi a mediului. Pe baza acestei reprezentări se poate planifica traiectoria optimă pentru robot, pornind de la o poziţie de start până la poziţia ţintă.

Această metodă este asociată cu nivelul ierarhic superior din sistemul de conducere al unui robot. Cu această metodă se obţine o traiectorie optimă, ocolindu-se obstacolele statice sau dinamice cunoscute din mediul de lucru. De cele mai multe ori metodele globale constă în crearea de hărţi sau grafuri şi abordează problema doar din punct de vedere geometric. Un exemplu de hartă globală este cea a unui oraş pe care sunt reprezentate toate datele necesare unei navigări, cum ar fi: străzile, intersecţiile, clădirile importante, etc.

Metodele globale au avantajul că pot prescrie o traiectorie optimă între două locaţii, chiar în medii extrem de complexe, însă determină limitarea capabilităţilor robotului de a acţiona în timp real în mediu dezordonat, în special când sunt multe informaţii noi despre schimbările din mediu. Aşadar nu au soluţii pentru evitarea obstacolelor ce pot apărea pe neaşteptate în faţa robotului.

Această planificare se bazează pe modelul mediului şi al robotului. Acest mijloc de "predicţie" bazat pe modele (care sunt doar aproximări) şi bazat pe comportamentul robotului pot prezenta de multe ori incertitudini şi imprecizii. Aşadar, o planificare independentă de execuţie nu poate să fie robustă.

Dintre cele mai folosite metode globale de navigare a roboţilor mobili pot fi amintite: metoda spaţiului de configurare [29], metoda câmpului de potenţial [30],[31] şi diagrama Voronoi generalizată [32],[33].

Metodele locale sunt aplicabile atunci când robotul nu are informaţii cu privire la structura mediului său de lucru. În aceste situaţii este pusă în valoare autonomia cu care este înzestrat robotul. Acesta trebuie ca pe baza senzorilor cu care este echipat, să obţină informaţii despre mediu şi să reducă cât mai mult din incertitudinea cu privire la structura mediului de lucru. De regulă, robotul are montaţi pe el diferiţi senzori şi poate să determine structura mediului pe direcţia sa de deplasare. Informaţiile obţinute de la senzori sunt reprezentate sub forma unei hărţi locale a mediului din jurul robotului, în care apar obstacolele din imediata sa apropiere.

Senzorii cei mai utilizaţi pentru navigarea roboţilor mobili prin metodele locale sunt senzorii ultrasonici, senzorii laser şi/sau senzorii vizuali. Aceşti senzori sunt montaţi pe robotul mobil în diverse configuraţii, astfel încât să poată detecta

obstacolele din faţa robotului (direcţia de navigare), din lateral sau chiar din spatele acestuia.

Metodele locale de navigare sunt asociate, de regulă, cu nivelul ierarhic inferior din sistemul de conducere al unui robot şi de cele mai multe ori realizează conducerea robotului pe o traiectorie prescrisă la nivelul sistemului de planificare globală. Prin metoda locală de navigare se pot evita obstacolele necunoscute indiferent dacă acestea sunt statice sau se află în mişcare, compensând incertitudinea datelor furnizate de sistemul de conducere de la nivelul global.

Planificarea sau navigarea locală poate lua în considerare atât cinematica cât şi dinamica robotului, deoarece se bazează pe informaţii obţinute din semnale care se pot prelucra în timp real. Însă, folosirea doar a informaţiilor locale primite de la senzori garantează evitarea coliziunilor dar nu garantează obţinerea unei traiectorii optime între două poziţii din mediu chiar dacă aceasta există şi nu semnalează dacă poziţia ţintei nu este accesibilă.

Aşadar, ambele metode prezentate mai sus au unele dezavantaje şi astfel apare implicit necesitatea integrării celor două metode pentru navigarea roboţilor mobili autonomi. Acest lucru nu înseamnă doar o simplă utilizare concomitentă a acestora, fapt ce poate conduce la multe erori de navigare, ci trebuie realizate numeroase iteraţii planificare-execuţie. De asemenea, trebuie o re-analizare completă a situaţiei, implicând percepţia, modelarea şi localizarea pentru actualizarea contextului şi evitarea apariţiei erorilor în faza de execuţie.

În acest sens, au apărut **metodele hibride** pentru navigarea roboţilor mobili. Metodele globale ghidează robotul pe o traiectorie optimă care duce spre ţintă, iar metodele locale îl ajută pe acesta să evite obstacolele ce pot apărea în calea sa.

O implementare a metodelor hibride o constituie împărţirea sistemului de conducere al robotului pe nivele ierarhice. Astfel, nivelul ierarhic superior al robotului va coordona deplasarea acestuia pe traiectoria optimă ce duce spre ţintă, iar nivelul ierarhic inferior are sarcina de evitare a obstacolelor şi de ieşire din situaţiile de blocaj.

2.2.2. Strategii de navigare

În general, înainte de realizarea planificării traiectoriei trebuie stabilită o strategie de navigare. Acest lucru presupune stabilirea modului de deplasare a robotului corelat cu sarcina ce trebuie îndeplinită. Principalele strategii sau modalităţi de navigare a roboţilor mobili sunt următoarele:

- **Deplasarea dintr-un punct al spaţiului de lucru (configuraţia iniţială) într-un alt punct (configuraţia finală)**, ambele puncte având configuraţii bine determinate. Strategia constă în generarea traiectoriei, respectiv comenzilor necesare pentru atingerea destinaţiei.
- **Baleierea întregului mediu** - este strategia de navigare impusă roboţilor ce trebuie să cerceteze întregul domeniu al unui spaţiu dat. Este cazul roboţilor de cercetare, a căror scop este ridicarea unor hărţi, sau a celor ce trebuie să execute servicii de curăţare a unei suprafeţe. Baleierea mediului se poate face în două maniere distincte. Prima este aceea în care robotul se deplasează pe traiectorii spirale începând din zonele periferice ale mediului şi înaintând spre zonele interioare. Cea de-a doua manieră este deplasarea în zig-zag, din perete în perete, asemănătoare modului de citire a unei cărţi.

- **Navigarea prin metoda *wall-following*** [34],[35] - metodă prin care robotul se deplasează având ca punct de reper unul din pereţii coridorului sau încăperii. Această metodă este folosită de regulă în cazul navigării în interiorul clădirilor.

Datorită faptului că, de cele mai multe ori, mediul de lucru are o structură complexă, cuprinzând multe obstacole, se încearcă divizarea lui în sub-zone în care să se poată practica una dintre strategiile de mai sus. Problema constă în a minimiza numărul de zone pe care robotul trebuie să le parcurgă de mai multe ori.

2.3. Modelarea mediului

Pentru a putea stabili o traiectorie între configuraţia iniţială şi cea finală este necesară o hartă pe baza căreia să se facă această planificare. Harta reprezintă, de fapt, modelul mediului şi este cunoscută de către robot fie apriori, fie este ridicată pe parcursul deplasării acestuia spre ţintă.

Pe baza acestei hărţi este reprezentat mediul în memoria internă a robotului mobil. Datorită formelor complexe ale obstacolelor se procedează, în general, la înscrierea acestora în forme convexe mai mari.

De asemenea, o procedură larg răspândită este aceea în care robotul este privit ca o particulă punctiformă, iar obstacolele sunt dilatate cu o valoare constantă, egală cu raza cercului ce se circumscrie valorilor de gabarit reale ale robotului. Este un mod de simplificare a problemei dar din păcate cu această metodă nu se obţine de fiecare dată traiectoria optimă. Astfel, există situaţii când datorită expandării obstacolelor se pierde o traiectorie mai scurtă, traiectorie ce ar putea fi urmată dacă robotul ar fi privit la dimensiunile sale reale.

Indiferent de tipul modelului adoptat pentru reprezentarea mediului, tendinţa generală este aceea de a împărţi mediul în zone libere (accesibile robotului) şi zone interzise (obstacole), încercând să se construiască, totodată, o listă care să reflecte tangenţiabilitatea zonelor libere.

2.3.1. Metoda grilei

Reprezintă metoda cea mai simplă de modelare a spaţiului, constând în suprapunerea unei grile cu celule de formă pătrată sau dreptunghiulară peste imaginea mediului [36],[37]. Astfel, celulele ocupate de obstacol vor fi celule "interzise" prin care nu va putea trece traiectoria robotului. În schimb, celulele care se suprapun peste spaţiile dintre obstacole sunt denumite celule "libere" prin care poate să treacă traiectoria robotului. Apartenenţa celulelor la una din zonele libere sau cu obstacole este booleană, acele celule care sunt parţial ocupate de obstacol vor fi considerate ca fiind complet ocupate. Precizia modelării mediului este dependentă de dimensiunile alese pentru celule. O discretizare foarte fină a spaţiului, în celule cu dimensiuni mici, conduce la dezavantajul că necesită un spaţiu mare de memorare, precum şi capacitate mare de calcul.

Dacă imaginea mediului nu este cunoscută şi aceasta se formează în timpul deplasării robotului, cu ajutorul senzorilor, atunci reprezentarea binară prezentată mai sus poate fi combinată cu o filtrare bayesiană binară care estimează probabilitatea ca o celulă să aibă una din cele două stări [38]. Se utilizează o ecuaţie incrementală (relaţia 2.1) prin care harta mediului poate fi actualizată pe baza informaţiilor senzoriale recepţionate.

$$\log\frac{p(m_{x,y}\mid z^t,s^t)}{1-p(m_{x,y}\mid z^t,s^t)}=\log\frac{p(m_{x,y}\mid z^t,s^t)}{1-p(m_{x,y}\mid z^t,s^t)}+\log\frac{1-p(m_{x,y})}{p(m_{x,y})}+ \\ +\log\frac{p(m_{x,y}\mid z^{t-1},s^{t-1})}{1-p(m_{x,y}\mid z^{t-1},s^{t-1})}, \tag{2.1}$$

unde z reprezintă măsurătorile de la senzori, s poziţia robotului, iar $m_{x,y}$ valoarea de ocupare a celulei din locaţia (x, y).

Harta obţinută *online* (în timpul deplasării robotului) poate să conţină unele erori datorită faptului că utilizarea senzorilor laser sau a celor ultrasonici poate da greş în cazurile în care apar suprafeţe înclinate, astfel unele obiecte fie nu pot fi detectate fie poziţia lor este estimată greşit.

2.3.2. Metoda arborelui

Reprezintă o variantă îmbunătăţită a metodei grilei. Prin această metodă s-a încercat minimizarea unor dezavantaje ale metodei precedente şi anume cele legate de volumul mare de memorie necesar şi cele referitoare la ineficienţa algoritmului în zone libere mari.

Îmbunătăţirea adusă se referă la dimensiunile celulelor în care se divide spaţiul. Astfel, celulele gridului nu mai au aceleaşi dimensiuni şi valori, ci se adaptează la mediu. Varianta lui David Zhu din [39] procedează la o descompunere în celule pe care apoi le etichetează în: **goale**, **pline** şi **mixte**. Algoritmul se ocupă apoi de celulele mixte, recurgând la divizări succesive care să ducă în final doar la celule goale sau pline. Rezultă astfel o structură ierarhizată de reprezentare a mediului, gen arbore.

2.3.3. Metoda grilei neomogene

În cadrul acestei metode, obstacolele sunt reprezentate cu forme convexe cuprinzătoare. Metoda consideră toate obstacolele din mediu ca fiind dreptunghiulare. Algoritmul metodei grilei neomogene procedează la o împărţire a mediului în zone delimitate de prelungirea contururilor obstacolelor. În acest fel, fiecare dimensiune a spaţiului va fi partiţionată într-un număr de $2n+1$ zone (unde n este numărul obstacolelor distincte pe direcţia considerată). Figura 2.1 exemplifică această metodă pentru cazul unui mediu cu două obstacole.

În urma descompunerii în zonele primare, se alcătuieşte lista ariilor de intersecţie, listă care arată, pentru cazul prezentat în Figura 2.1, astfel:

A-C	0011110000	**B-F**	0010000001
A-E	1000010000	**B-G**	0000100001
A-F	0010010000	**C-F**	0010011000
A-G	0000110000	**C-G**	0000111000
B-D	1110000001	**D-E**	1000000011
B-E	1000000001	**D-F**	0010000011

Cele două şiruri de valori binare reprezintă intersecţiile între zonele menţionate în prima coloană. Astfel, existenţa unei intersecţii este codată cu simbolul "1", iar absenţa ei cu "0". În listă nu au mai fost trecute perechile de zone care nu prezintă intersecţii (spre exemplu A-D sau E-C).

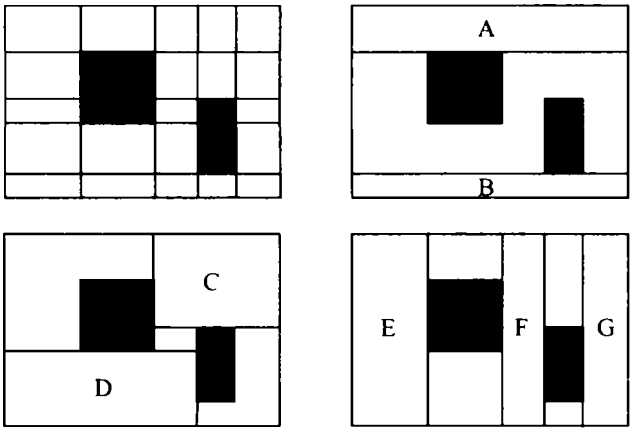


Figura 2.1. Exemplu de descompunere cu metoda grilei neomogene a unui spaţiu cu două obstacole; (A), (B), (C), (D) reprezintă zonele primare ale descompunerii.

Ţinând cont de această listă se poate construi graful aferent spaţiilor libere ale mediului considerat mai sus. Fiecare nod al grafului reprezintă una din celulele considerate (A, B, ..., G), iar arcele oferă posibilitatea de transfer dintr-o celulă în alta. Graful rezultat este cel din Figura 2.2. S-a trasat un arc între două noduri pe acest graf dacă în lista intersecţiilor există cel puţin un simbol de intersecţie (simbolul "1") atât pentru codarea orizontală cât şi pentru cea verticală.

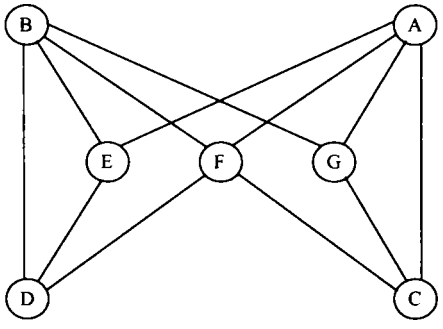


Figura 2.2. Graful rezultat pentru descompunerea neomogenă din Figura 2.1.

2.3.4. Metoda poligoanelor convexe

Această metodă constă în descompunerea mediului în celule poligonale convexe cât mai simple posibil. Două laturi alăturate ale aceleaşi celule nu trebuie să formeze un unghi mai mare de 180°. Trecerea de la o celulă la alta se face prin mijlocul celulei comune, ceea ce duce la minimizarea riscului apariţiei unei coliziuni.

Celulele rezultate vor fi apoi asamblate într-un graf, a cărui noduri reprezintă celulele, iar arcele trecerile posibile între ele. În Figura 2.3 se prezintă un exemplu al acestei metode împreună cu graful aferent. După cum se vede din figură, se pot grupa celulele pe zone şi astfel se poate obţine un graf redus.

Avantajul care derivă este dat de micşorarea volumului de calcul, căutându-se o traiectorie de legătură în primă fază între zonele mari, iar apoi doar în interiorul celor selectate.

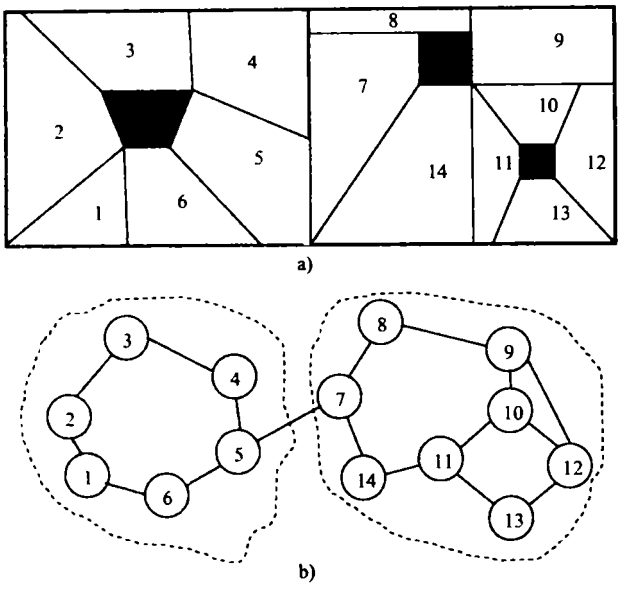


Figura 2.3. Împărţirea mediului utilizând metoda poligoanelor convexe;
a) forma poligoanelor, b) graful rezultat.

2.4. Localizarea robotului

Traiectoria obţinută, poate fi privită ca o secvenţă de configuraţii ale robotului, fiecare din acestea fiind definite de locaţia robotului (x, y) şi orientarea sa (θ) faţă de un sistem de referinţă fix XOY (Figura 2.4) [40]. În figură este prezentată şi poziţia centrului instantaneu de rotaţie (CIR) faţă de care punctul central al robotului descrie o mişcare circulară, în cazul în care acesta execută un viraj. În cazul unui sistem robotic aproximativ omogen, punctul central al acestuia coincide cu centrul de greutate al sistemului.

Robotul mobil trebuie să-şi determine periodic, poziţia şi orientarea în raport cu un sistem de referinţă fix pe baza informaţiilor recepţionate de către senzori.

Fiind cunoscute poziţia şi orientarea robotului la un anumit moment t_0, notate cu p, la un moment ulterior t_1 poziţia şi orientarea robotului notate cu p' sunt exprimate de relaţia 2.2.

$$p(t_0)=\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad p'(t_1)=p(t_0)+\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} \tag{2.2}$$

unde Δx, Δy şi $\Delta\theta$ reprezintă variaţiile variabilelor ce definesc poziţia şi orientarea robotului în intervalul de timp $t_1 - t_0$.

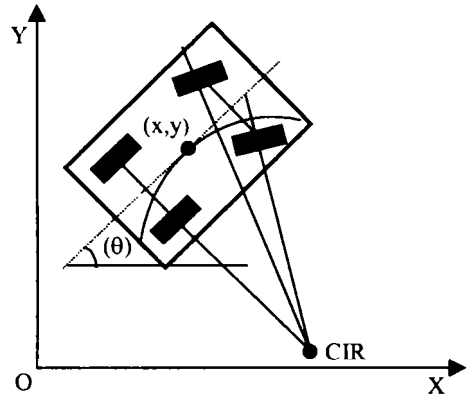


Figura 2.4. Coordonatele folosite pentru localizarea robotului într-un mediu de lucru plan.

Metoda de localizare folosită depinde de tipul de senzori cu care este înzestrat robotul precum şi de sarcina ce trebuie îndeplinită de acesta. Cei mai utilizaţi senzori în acest sens sunt: senzorii optici (camerele CCD) şi senzorii de proximitate (senzorii laser, în infraroşu, ultrasonici, etc.). Cu toate că cercetările recente cu privire la utilizarea camerei video pentru localizare, au demonstrat că aceasta poate fi eficientă, totuşi uneori acestea nu sunt de preferat datorită volumului mare de date ce trebuie procesat. Comparativ cu senzorii vizuali, un avantaj al senzorilor de proximitate este faptul că sunt mai puţin afectaţi de modificări ale condiţiilor de iluminare cum ar fi condiţii de întuneric sau fum.

La alegerea senzorilor trebuie ţinut cont de câmpul de vizibilitate al acestora şi anume dacă acesta se pretează pentru aplicaţia dorită.

Pe timpul deplasării robotului se acumulează în mod inerent erori în ceea ce priveşte estimarea poziţiei sale. În acest sens apare noţiunea de zonă de incertitudine ce reprezintă erorile curente posibile, sau mai bine zis, toate poziţiile posibile ale robotului la un moment dat.

Localizarea este procesul de reducere a zonei de incertitudine a robotului utilizând, de regulă, măsurătorile realizate cu ajutorul senzorilor. Există la ora actuală o serie de modalităţi de localizare ce pot fi clasificate în: localizare *on-line* sau *off-line*, modele de localizare statistice sau geometrice, localizări pe baza reprezentării mediului, cu ajutorul hărţii sau topologice, localizare dacă harta mediului este cunoscută, parţial cunoscută sau necunoscută, etc.

Astfel, sunt cunoscute sistemele de localizare cu senzori ultrasonici utilizând filtre Kalman şi filtre Kalman extinse [41], precum şi metodele de localizare Monte Carlo [42]. De asemenea, sunt utilizate şi metodele de localizare a roboţilor mobili prin triangulaţie.

Estimarea poziţiei unui robot mobil s-a realizat la început folosind sisteme hodometrice [16], care însă au dezavantajul că erorile se acumulează pe timpul deplasării.

S-a constatat că plasarea unor repere vizuale în mediul de lucru, la locaţii cunoscute, cresc precizia de estimare a poziţiei.

Ulterior, în cazul navigării exterioare (*outdoor*), s-au folosit repere naturale şi sisteme de poziţionare globale GPS (***G**lobal **P**ositioning **S**ystems*) [43]. Sistemele GPS au precizie de ordinul centimetrilor, dar din păcate valorile furnizate de acestea nu sunt întotdeauna corecte. Schimbările survenite în constelaţia sateliţilor, clădirile

mari şi chiar pădurile dese pot genera situaţii în care valorile date de GPS fie nu pot fi obţinute, fie sunt eronate.

2.5. Planificarea traiectoriei

În această etapă, din cadrul procesului de navigare a roboţilor mobili, este tratată problematica mai largă a conducerii robotului, în care conceptul de "autonom" începe să se contureze mai pregnant [44]. Planificarea este o etapă caracterizată prin prezenţa acţiunilor de decizie, chiar dacă de cele mai multe ori *off-line*. Robotului îi este comunicată destinaţia şi eventual unele date suplimentare cu privire la mediu, urmând ca el să stabilească traiectoria optimă. Gradul de autonomie al unui robot depinde de capacitatea sa de planificare (globală şi/sau locală) a traiectoriei, care trebuie făcută astfel încât să fie evitate obstacolele din mediu.

Cea mai simplă abordare a problemei planificării consideră robotul ca fiind singurul obiect mobil din mediu, care nu posedă proprietăţi dinamice şi care nu intră în contact cu obiectele înconjurătoare. Consideraţiile presupuse mai sus, transformă problema planificării într-o problemă geometrică, ce tratează robotul ca pe un solid rigid a cărui mişcare este limitată la spaţiul liber.

Diferite alte abordări cresc complexitatea modelului de studiu al planificării traiectoriei, ajungând chiar la considerarea mai multor roboţi în acelaşi spaţiu de lucru. Astfel, în prezenţa mai multor obiecte mobile, problema planificării nu are ca soluţie o simplă traiectorie geometrică. Este necesară introducerea unei noi dimensiuni (timpul) şi generarea unei traiectorii în funcţie de timp, traiectorie pe care să se cunoască poziţia robotului la fiecare moment. Datorită ireversibilităţii timpului, este necesar ca traiectoria propusă de către planificator să nu conţină porţiuni în care axa timpului să fie străbătută înapoi. Dacă în mediu există şi alte elemente mobile în afara roboţilor (spre exemplu persoane), asigurarea unei traiectorii lipsită de coliziuni necesită utilizarea unor algoritmi de evitare a obstacolelor destul de sofisticaţi.

Tendinţa generală a metodelor de planificare, cu excepţia celor bazate pe modelele câmpurilor potenţiale, este aceea de reducere a spaţiului liber la o structură de tip graf, urmând ca în continuare, pentru găsirea unui drum optim să se apeleze la algoritmi deja consacraţi de rezolvare a grafurilor. Un avantaj imediat al acestui mod de planificare este cel legat de posibilitatea considerării unor costuri multiple. Pe lângă lungimea traseului se mai pot lua în considerare factori de genul: timpul necesar pentru străbaterea porţiunii, energie consumată sau siguranţa acestuia, rezultând astfel o analiză calitativ superioară.

2.5.1. Metoda hărţii drumurilor

Această metodă (*roadmaps*) are la bază ideea construirii unui graf care să reflecte conexitatea spaţiului liber al mediului de lucru. Reţeaua de curbe care alcătuieşte graful este denumită harta drumurilor. O astfel de reţea poate fi folosită apoi pentru alegerea traiectoriei optime care leagă configuraţia iniţială de cea finală, utilizând algoritmi specifici grafurilor. În funcţie de principiile pe baza cărora se construieşte harta drumurilor se obţin diferite tipuri de hărţi, cum ar fi:

- graful vizibilităţii;
- diagrama Voronoi;
- reţeaua drumurilor libere.

2.5.1.1. Metoda grafului vizibilităţii

Această metodă se aplică, de regulă, în situaţiile în care obstacolele din mediu au forme poligonale. Principiul metodei constă în construirea unei traiectorii semi-libere cu o linie poligonală ce trece prin vârfurile obstacolelor.

Se defineşte graful vizibilităţii, respectiv graful neorientat astfel:

- nodurile sale sunt: configuraţia iniţială, cea finală şi toate vârfurile obstacolelor;
- două noduri ale grafului sunt conectate printr-o legătură dacă şi numai dacă, segmentul de dreaptă ce le uneşte este muchie a unui obstacol sau se află în întregime în spaţiul liber.

Un exemplu de graf al vizibilităţii în cazul unui mediu bidimensional cu trei obstacole este redat în Figura 2.5.

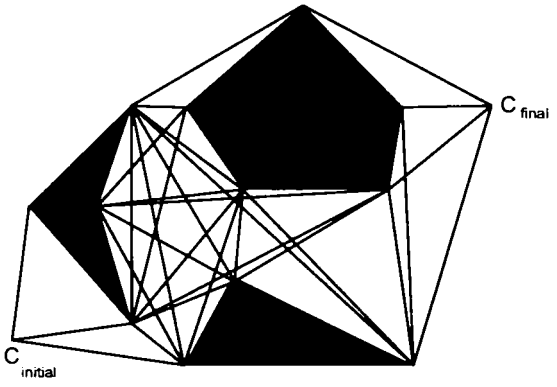


Figura 2.5. Graful vizibilităţii în cazul unui mediu de lucru cu trei obstacole având forme poligonale.

Se poate observa că graful rezultat în urma aplicării metodei vizibilităţii este destul de stufos, fapt ce necesită în continuare un cost computaţional ridicat. Randamentul metodei poate fi însă crescut dacă se ţine cont că unele trasee pot fi optimizate. Dacă unghiul ascuţit al segmentelor ce se sprijină pe un vârf al unui obstacol este cuprins în spaţiul liber, traiectoria poate fi scurtată prin unirea celor două segmente cu un al treilea, cu condiţia ca acesta din urmă să fie conţinut şi el în întregime în spaţiul liber. Dacă unghiul ascuţit format de cele două segmente cuprinde la rândul lui un colţ al unui obstacol, optimizarea nu poate fi aplicată.

Pe de altă parte, din mulţimea de legături al unui graf al vizibilităţii, o parte nu sunt necesare. E suficient dacă se păstrează legăturile grafului care sunt fie segmente suport fie segmente tangente. Pentru aceasta, se defineşte ca segment suport legătura ce uneşte două vârfuri a două obstacole diferite astfel încât cele două obstacole să se găsească amândouă în întregime în acelaşi semiplan. Segmentele tangente leagă şi ele două vârfuri a două obstacole diferite, dar de această dată obstacolele sunt conţinute în semiplanuri diferite. Cu alte cuvinte, atât segmentele suport cât şi cele tangente nu "înţeapă" niciodată obstacolele.

Prin această simplificare se elimină din mulţimea legăturilor grafului, legăturile aferente vârfurilor concave. Graful rezultat în urma simplificării descrise se numeşte graful vizibilităţii redus. Figura 2.6 redă acest graf, provenit din

simplificarea grafului vizibilităţii din Figura 2.5. A fost redată aici cu linie îngroşată traiectoria optimă obţinută ce leagă configuraţia iniţială de cea finală.

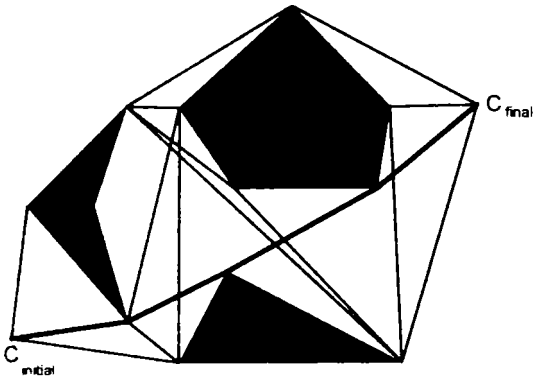


Figura 2.6. Graful vizibilităţii redus şi traiectoria optimă rezultată.

Metoda grafului vizibilităţii se aplică, de obicei, împreună cu o expandare a dimensiunilor obstacolelor, pentru a preîntâmpina coliziunile posibile în zona vârfurilor. Această metodă poate fi folosită şi în cazul în care obstacolele vor fi modelate prin poligoane generalizate, adică poligoane definite atât prin segmente de dreaptă cât şi prin arce de cerc. De asemenea, metoda poate fi aplicată şi în cazul în care spaţiul în care evoluează robotul este tridimensional. Obstacolele sunt modelate, în acest caz, cu ajutorul poliedrelor, iar robotul va putea fi considerat o sferă circumscrisă formei reale. Însă, în cazul spaţiului tridimensional, metoda nu oferă soluţia optimă, adică traiectoria cea mai scurtă. De cele mai multe ori traiectoria optimă se obţine parcurgând o linie poligonală ce se sprijină pe muchiile poliedrelor şi nu pe vârfurile lor. Pentru ameliorarea performanţelor metodei, în cazul tridimensional, se practică introducerea unor vârfuri fictive, respectiv a unor puncte ce aparţin muchiilor.

2.5.1.2. Metoda retractării spaţiului liber

Această metodă constă în construirea unei hărţi a drumurilor prin definirea unei aplicaţii continue a spaţiului liber pe harta drumurilor numită retractare. În literatură [45], se dă definiţia retractării astfel: "dacă X este un spaţiu topologic şi v o submulţime a sa, aplicaţia surjectivă $X \rightarrow v$ se numeşte retractarea lui X la v, dacă şi numai dacă este continuă, iar retracţia sa la v este aplicaţia identitate". Dacă ρ este retracţia lui X la v ea păstrează conexitatea lui X dacă şi numai dacă pentru orice $x \in X$, x şi ρ(x) aparţin aceleiaşi componente conexe a lui X.

Între două configuraţii, iniţială şi finală, există o traiectorie liberă dacă şi numai dacă există o curbă în harta drumurilor între $\rho(config_{iniţial})$ şi $\rho(config_{final})$. În aceste condiţii traiectoria ce uneşte cele două configuraţii se compune din trei segmente [2]:

- o traiectorie liberă de la $config_{iniţial}$ la $\rho(config_{iniţial})$;
- o traiectorie de la $\rho(config_{iniţial})$ la $\rho(config_{final})$ în harta drumurilor;
- o traiectorie liberă de la $\rho(config_{final})$ la $config_{final}$.

În cazul spaţiului bidimensional, metoda constă în retractarea spaţiului liber din mediu la diagrama sa Voronoi. Această diagramă este compusă din mulţimea punctelor din mediu care sunt egal depărtate de muchiile obstacolelor. Un exemplu de retractare la diagrama Voronoi, pentru un caz simplu cu două obstacole într-un mediu mărginit, este prezentat în Figura 2.7.

Segmentele ce alcătuiesc diagrama Voronoi sunt fie segmente de dreaptă, fie arce de parabolă. În schimb, există un arc de parabolă, în zona centrală a spaţiului liber, mărginit de un vârf şi de o muchie.

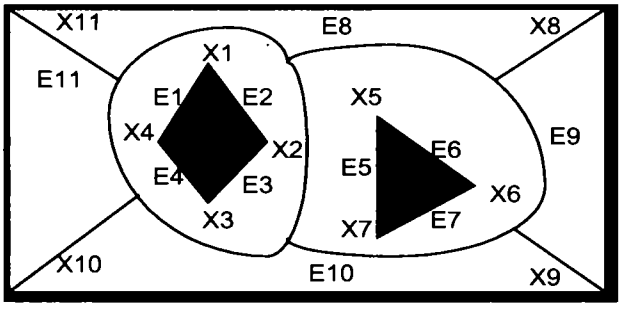


Figura 2.7. Retractarea unui spaţiu bidimensional la diagrama Voronoi.

În ceea ce priveşte localizarea configuraţiilor iniţiale şi finale există trei situaţii distincte:

- acestea sunt situate chiar pe arcele diagramei Voronoi - în acest caz este suficient să se caute ramurile diagramei ce unesc cele două puncte;
- unul din punctele de capăt nu aparţine diagramei Voronoi ci spaţiului liber, dar se găseşte pe o prelungire a unei muchii a unui obstacol - pentru acest caz deplasarea până la diagramă se va face pe prelungirea liniei de contur a obstacolului (Figura 2.8).
- nici una din configuraţii nu se află pe diagramă - în această situaţie legătura cu diagrama se va face pe segmentul de dreaptă coborât perpendicular din cel mai apropiat punct al diagramei Voronoi pe latura obstacolului.

În modul de abordare a ultimelor două cazuri se observă o tendinţă de depărtare faţă de obstacole spre drumul sigur al diagramei Voronoi.

Etapele algoritmului metodei retractării sunt:

- calculul diagramei Voronoi;
- calculul punctelor $\rho(config_{iniţial})$ şi $\rho(config_{final})$ şi identificarea arcelor de diagramă care le conţin;
- căutarea unui traseu pe diagrama Voronoi care să unească $\rho(config_{iniţial})$ şi $\rho(config_{final})$;
- dacă acest traseu nu există, algoritmul se încheie. Dacă există un astfel de traseu se procedează la concatenarea traseului găsit cu segmente de dreaptă ce unesc $\rho(config)$ de configuraţiile iniţiale şi finale.

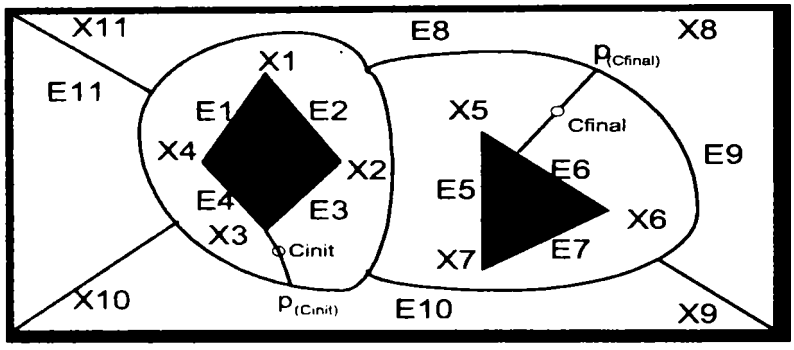


Figura 2.8. Exemplu de racordare la diagrama Voronoi.

2.5.1.3. Metoda drumului liber

Metoda drumului liber se bazează pe extragerea unor figuri geometrice din spaţiul liber, figuri ce poartă denumirea de drumuri libere [45]. Este dată următoarea definiţie: "un drum liber este un cilindru generalizat, a cărui axă este completată de descrierea tuturor orientărilor libere ale robotului, când originea sistemului de referinţă ataşat robotului se deplasează de-a lungul axei sale".

Considerând mediul W şi reuniunea B a tuturor obstacolelor, drumurile libere ale mediului vor fi extrase din regiunea W\B, luând în considerare toate muchiile obstacolelor. Pentru a produce un cilindru generalizat, o pereche de muchii a oricăror două obstacole trebuie să îndeplinească condiţiile:

- oricare din cele două muchii considerate, să nu intersecteze cealaltă muchie;
- produsul scalar al vectorilor normali la cele două muchii să fie negativ.

Cele două condiţii de mai sus impun, de fapt, ca muchiile considerate să fie poziţionate faţă în faţă. În Figura 2.9 se redă un cilindru generalizat construit pe baza a două muchii E1 şi E2.

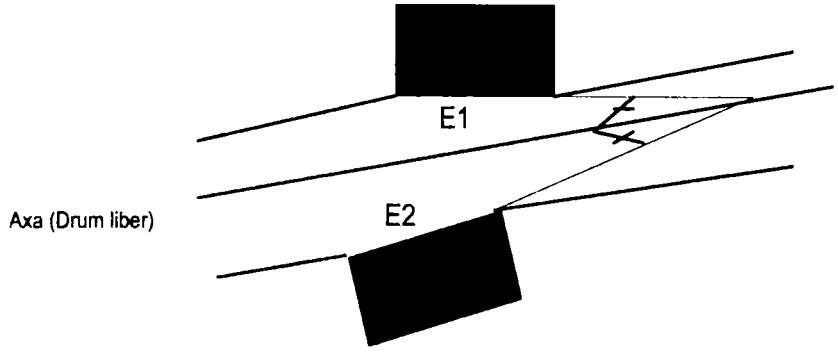


Figura 2.9. Cilindrul generalizat determinat de muchiile E1 şi E2.

Axa cilindrului generalizat este bisectoarea unghiului format de prelungirile muchiilor obstacolelor. Dacă muchiile E1 şi E2 sunt paralele, axa cilindrului este o

dreaptă paralelă şi egal distanţată de muchii. Laturile cilindrului din Figura 2.9 sunt constituite din prelungirile muchiilor obstacolelor cu segmente de dreaptă paralele cu bisectoarea cilindrului.

Drumul liber este ales ca fiind chiar axa cilindrului generalizat. Această axă este egal depărtată de muchiile obstacolelor şi din acest punct de vedere metoda drumurilor libere se aseamănă cu diagrama Voronoi. Chiar o parte a acestei diagrame este conţinută în drumul liber.

Se procedează în mod asemănător cu tot spaţiul W\B, pentru toate perechile de muchii ale reuniunii B, obţinându-se în acest fel o mulţime finită de cilindri generalizaţi ce conţin doar spaţiu liber.

Pentru găsirea hărţii drumurilor se realizează concatenarea axelor cilindrilor într-o serie de linii poligonale frânte, la care este necesar să se adauge pentru fiecare punct al traseului intervalul de orientări admise pentru robot. Se obţine un graf, denumit reţeaua drumurilor libere, graf care descrie complet mediul. În continuare, se vor aplica algoritmi destinaţi rezolvării grafurilor.

2.5.2. Metoda descompunerii celulare

Planificarea traiectoriei prin metoda descompunerii celulare [45] se bazează pe modelarea mediului cu metoda poligoanelor convexe prezentată anterior.

Principial, metoda poate fi rezumată la următorii paşi:

- se descompune spaţiul liber al mediului într-o mulţime de regiuni distincte numite celule, astfel încât în cadrul fiecărei celule să se poată realiza o navigaţie fără coliziune;
- celulele sunt apoi conectate într-un graf al conexităţii, în aşa fel încât două celule sunt conectate dacă există o traiectorie fără coliziune între ele;
- se va extrage traiectoria considerată optimă pe baza grafului conexităţii între celula conţinând poziţia de start şi celula în care este situată poziţia ţintei.

Deplasarea robotului mobil va fi uşor de realizat în cadrul unei celule iar între acestea este specificată pe baza grafului conexităţii. Pentru găsirea unei traiectorii optime, modul de descompunere în celule este foarte important. Celulele rezultate în urma descompunerii trebuie să îndeplinească câteva condiţii şi anume:

- topologia fiecărei celule trebuie să fie îndeajuns de simplă pentru a face posibilă calcularea unei traiectorii între oricare două puncte din celulă;
- pentru oricare două celule adiacente să fie posibilă calcularea unei traiectorii de traversare a frontierei ce desparte celulele.

Figura 2.10 exemplifică descompunerea în poligoane convexe a spaţiului liber pentru un mediu bidimensional, prezentând şi graful rezultat, graf ce se bazează pe relaţia de conexitate a celulelor.

Rezultatul algoritmului de căutare în graful conexităţii este un canal format dintr-o serie de celule adiacente, canal ce leagă configuraţia iniţială de cea finală. Pentru exemplul din Figura 2.10, un posibil canal între poziţia iniţială (celula 1) şi cea finală (celula 11) ar fi succesiunea 1, 7, 6, 9, 10, 11.

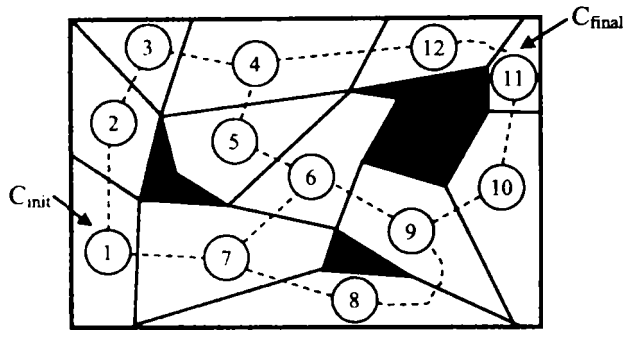


Figura 2.10. Descompunerea convex poligonală în cazul unui mediu bidimensional şi graful conexităţii rezultat.

Pentru o evitare cât mai sigură a obstacolelor, metoda cea mai uzuală de determinare a unei traiectorii este cea a punctelor mediane. Considerând linia de frontieră dintre două celule învecinate ca fiind mulţimea de puncte ce aparţin ambelor celule, respectiv segmentul de dreaptă ce le desparte, traiectoria va trece prin punctul median al segmentului. Trasând o linie frântă, care să unească poziţia iniţială cu poziţia finală, şi care să treacă prin punctele mediane de pe segmentele de frontieră ale celulelor ce formează canalul, se obţine traiectoria căutată (Figura 2.11).

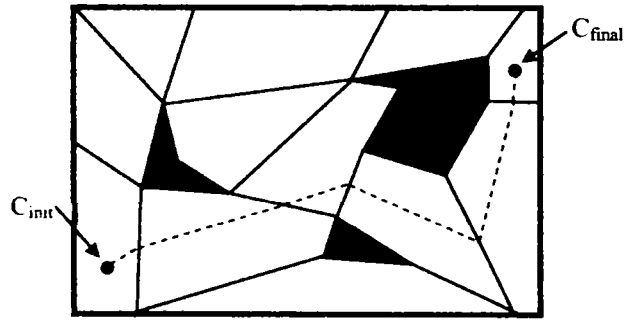


Figura 2.11. Traiectoria rezultată în urma aplicării metodei punctelor mediane.

Există şi o metodă mai simplă care constă în descompunerea spaţiului liber în celule disjuncte de formă trapezoidală. Considerând acelaşi mediu bidimensional prezentat în Figura 2.10, prin aplicarea metodei trapezoidale se obţine descompunerea din Figura 2.12. Fiecare vârf al obstacolelor mediului se prelungeşte cu o semidreaptă paralelă cu direcţia OY a sistemului de referinţă considerat.

Două celule sunt adiacente dacă şi numai dacă frontiera lor este o linie verticală. Acest considerent a făcut ca metoda să mai poarte şi numele de descompunere verticală.

Metoda prezintă avantajul de a putea opera şi cu medii de dimensiuni infinite. Într-un asemenea caz, baleierea mediului începe de la configuraţia iniţială şi se încheie la atingerea configuraţiei finale, generând o serie de celule ce se întind la infinit pe o direcţie OY aleasă.

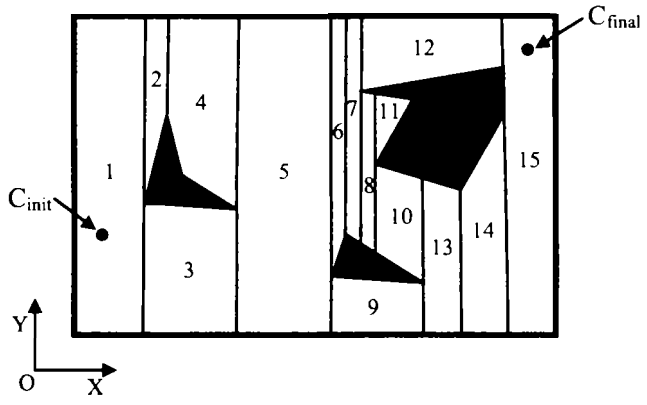


Figura 2.12. Descompunerea trapezoidală a unui mediu bidimensional.

Având descompunerea făcută, aşa ca în Figura 2.12, se poate alcătui graful conexităţii, graf ce este redat în Figura 2.13.

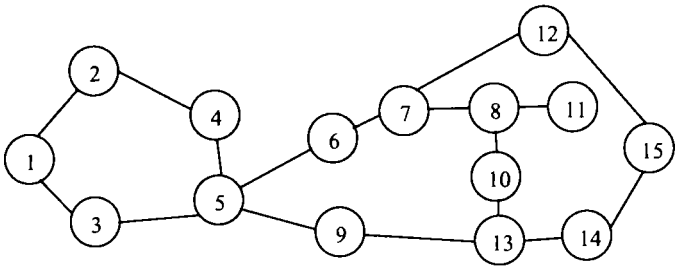


Figura 2.13. Graful conexităţii celulelor obţinut pe baza descompunerii din Figura 2.12.

Pe baza acestui graf se poate alege traiectoria optimă de deplasare între configuraţia iniţială şi cea finală (Figura 2.14). În final, pentru determinarea traseului se procedează analog cu cazul descompunerii convexe la unirea mijloacelor graniţelor celulelor care fac legătura între configuraţia iniţială şi finală.

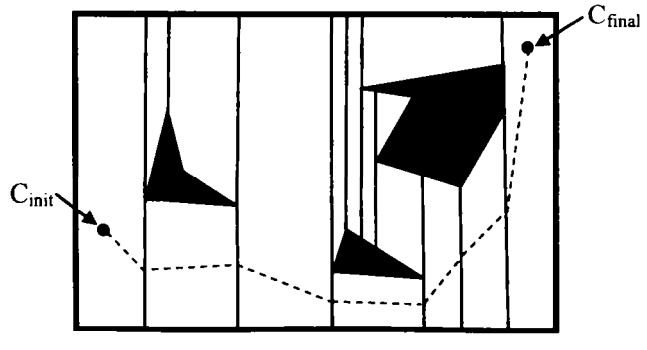


Figura 2.14. Traiectoria rezultată în urma aplicării metodei punctelor mediane pentru cazul descompunerii trapezoidale.

Metoda descompunerii în celule oferă o soluţie de compromis, un optim între cerinţa de a obţine o traiectorie cât mai scurtă şi tendinţa de a menţine robotul pe cât posibil mai departe de obstacole. În plus această metodă poate fi folosită şi în cazul unui mediu tridimensional, caz în care baleierea mediului se face cu un plan şi nu cu o dreaptă.

2.5.3. Metoda câmpului potenţial artificial

În cazul metodei câmpului potenţial artificial [30],[31], robotul mobil este tratat, în spaţiul său de configurare, ca un punct, respectiv ca o particulă aflată sub influenţa unui câmp potenţial artificial U, a cărui variaţie reflectă structura spaţiului liber. Funcţia potenţială se defineşte pe spaţiul de configurare al robotului, ca fiind suma dintre un potenţial atractiv care "atrage" robotul către configuraţia ţintei şi un potenţial repulsiv care "împinge" robotul dinspre obstacole. Planificarea mişcării se realizează într-un mod iterativ. La fiecare iteraţie direcţia forţei artificiale indusă de către funcţia potenţială, în configuraţia curentă, este privită ca fiind cea mai potrivită direcţie ce trebuie urmată de către robot în deplasarea sa către ţintă.

Ideea de bază este că robotul, respectiv punctul caracteristic al robotului este atras către ţintă în timp ce obstacolele îl resping. Această idee este ilustrată prin definirea unei funcţii potenţiale în cazul în care robotul se poate deplasa liber în $W = R^N$, cu N = 2 sau 3, $C = R^N$, unde W reprezintă spaţiul de lucru al robotului, R reprezintă un set de numere reale, iar C spaţiul de configurare al robotului. Un element din C este notat cu q.

Câmpul forţelor artificiale ce acţionează asupra robotului situat în punctul *q* din C este produs de o funcţie potenţială derivabilă $U : C_{liber} \to R$, şi este dată de relaţia 2.3:

$$\vec{F}(q) = -\nabla U(q), \tag{2.3}$$

unde $\nabla U(q)$ reprezintă vectorul gradient al câmpului potenţial artificial U în punctul q. În $C=R^N$ cu (N=2 sau 3), se poate scrie q(x,y) respectiv q(x,y,z), iar ∇U este dat de relaţiile 2.4.

$$\nabla U = \begin{bmatrix} \partial U/\partial x \\ \partial U/\partial y \end{bmatrix} \text{ sau } \nabla U = \begin{bmatrix} \partial U/\partial x \\ \partial U/\partial y \\ \partial U/\partial z \end{bmatrix} \tag{2.4}$$

Pentru atragerea robotului, pe de o parte către ţintă şi în acelaşi timp respingerea lui de către obstacole, câmpul potenţial U rezultant este dat de suma a două funcţii potenţiale (relaţia 2.5).

$$U(q) = U_{atr}(q) + U_{rep}(q), \tag{2.5}$$

unde U_{att} reprezintă potenţialul atractiv asociat punctului în care este situată ţinta $q_{ţintă}$, iar U_{rep} este potenţialul repulsiv asociat zonelor din C în care sunt situate obstacolele.

Având aceste notaţii se pot scrie forţele exercitate de fiecare potenţial în parte (relaţia 2.6).

$$\vec{F}_{atr} = -\nabla U_{atr}\ ,\quad \vec{F}_{rep} = -\nabla U_{rep}\ , \tag{2.6}$$

unde $\vec{F}_{atr}$ reprezintă vectorul forţelor atractive, şi $\vec{F}_{rep}$ este vectorul forţelor repulsive ce acţionează asupra robotului.

2.5.3.1. Potenţialul atractiv

Câmpul de potenţial atractiv U_{atr} poate fi definit sub forma unei funcţii parabolice (relaţia 2.7).

$$U_{atr} = \frac{1}{2} k \cdot p_{ţintă}(q), \tag{2.7}$$

unde k este un factor de scară pozitiv şi $p_{ţintă}(q)$ reprezintă distanţa euclidiană dintre punctul în care este situat robotul şi punctul în care e situată ţinta $||q-q_{ţintă}||$.

Distanţa euclidiană dintre două puncte dintr-un plan $s(x_1, y_1)$ şi $t(x_2, y_2)$, este dată de relaţia 2.8.

$$\|s - t\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.8}$$

Funcţia U_{atr} este pozitivă sau nulă şi atinge minimul în punctul $q_{ţintă}$ unde $U_{atr}(q_{ţintă}) = 0$.

Funcţia $p_{ţintă}$ este diferenţiabilă oriunde în C. În oricare configuraţie a lui q, forţa atractivă F_{atr} ce derivă din U_{atr} este dată de relaţia 2.9.

$$\vec{F}_{atr}(q) = -\nabla U_{atr}(q) = -\sigma \cdot p_{ţintă}(q) \nabla p_{ţintă}(q) = -\sigma(q - q_{ţintă}) \tag{2.9}$$

S-a ales o formă parabolică a funcţiei ce defineşte potenţialul de atracţie deoarece cu aceasta s-a obţinut o stabilitate bună mai ales atunci când se utilizează metoda câmpului potenţial pentru evitarea *on-line* a obstacolelor. În acest fel, forţa rezultantă F_{atr} converge liniar spre 0 atunci când robotul se apropie de ţintă. Pe de altă parte, F_{atr} creşte cu distanţa faţă de ţintă şi în final tinde către infinit atunci când $p_{ţintă}(q) \to \infty$.

În Figura 2.15 se arată un exemplu de potenţial atractiv generat de către o ţintă situată în punctul de coordonate x=12, y=13 în cadrul unui spaţiu de configurare de dimensiune 16×16 elemente.

2.5.3.2. Potenţialul repulsiv

Ideea principală este de a crea o barieră de potenţial în jurul obstacolelor situate în spaţiul de configurare *C* al robotului, care nu poate fi traversată de către acesta. Pe de altă parte, acest potenţial repulsiv nu trebuie să afecteze mişcarea robotului atunci când acesta este suficient de departe de obstacole. Aceste condiţii pot fi îndeplinite prin definirea unei funcţii potenţiale repulsive de forma celei prezentate în relaţia 2.10:

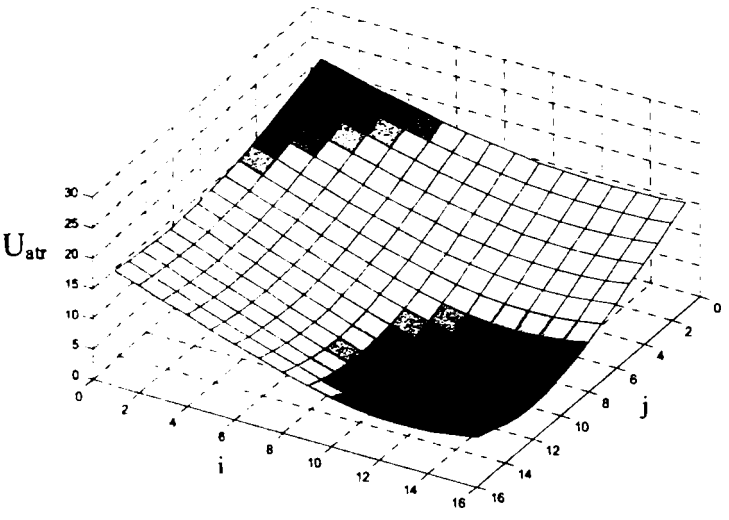


Figura 2.15. Forma potenţialului atractiv creat de către ţintă.

$$U_{rep}(q)=\begin{cases}\frac{1}{2}n\left(\frac{1}{p(q)}-\frac{1}{p_0}\right)^2 & \text{dacă} \quad p(q)<p_0\\ 0 \quad \text{dacă} & p(q)>p_0\end{cases}, \tag{2.10}$$

unde n este un factor de scară pozitiv şi $p(q)$ reprezintă distanţa dintre poziţia robotului şi obstacole.

Funcţia U_{rep} este pozitivă sau nulă, tinde la infinit atunci când robotul se apropie mult de zona obstacolelor şi este nulă când distanţa dintre robot şi obstacole este mai mare decât p_0.

Forţa repulsivă obţinută pe baza potenţialului U_{rep} este definită de relaţia 2.11 astfel:

$$\vec{F}(q)=-\nabla U_{rep}(q)=\begin{cases}n\left(\frac{1}{p(q)}-\frac{1}{p_0}\right)\frac{1}{p^2(q)}\nabla p(q) & \text{dacă} \quad p(q)\le p_0\\ 0 \quad \text{dacă} & p(q)>p_0\end{cases} \tag{2.11}$$

Orice obstacol detectat de unul din senzorii robotului, indiferent dacă acesta este un obstacol sau o parte dintr-un obstacol mai mare, produce o forţă repulsivă. Forţa repulsivă rezultantă este suma forţelor repulsive create de fiecare obstacol în parte şi este dată de relaţia 2.12:

$$\vec{F}_{rep(total)}(q)=\sum_i F_{rep(i)}(q), \tag{2.12}$$

unde prin i s-a notat numărul de senzori ai robotului.

În Figura 2.16 se prezintă un exemplu de potenţial repulsiv creat de trei obstacole situate într-un spaţiu de configurare discretizat de dimensiune 16×16 elemente.

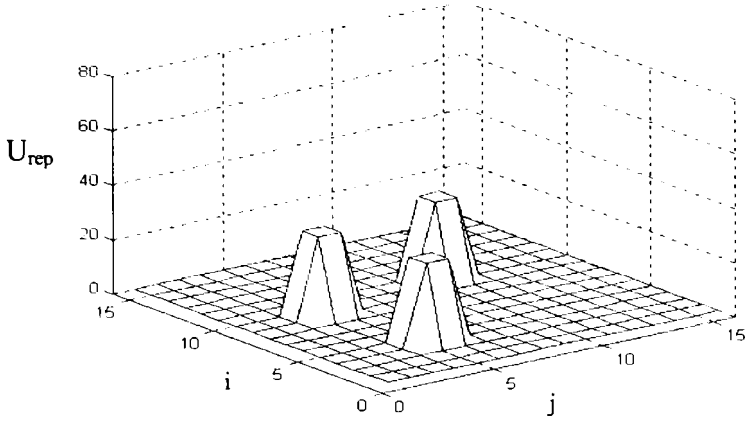


Figura 2.16. Un exemplu de potenţial repulsiv creat de trei obstacole.

În final, asupra robotului situat într-un punct q al spaţiului de configurare C acţionează atât forţa de atracţie dată de către configuraţia ţintei cât şi forţele de respingere determinate de către obstacolele situate în mediul de lucru al robotului. Astfel forţa totală artificială ce acţionează asupra robotului în punctul q este dată de relaţia 2.13.

$$\vec{F}_{total}(q) = \vec{F}_{atr}(q) + \vec{F}_{rep(total)}(q) \tag{2.13}$$

Forma câmpului potenţial artificial total, ce acţionează asupra robotului, produce în final $\vec{F}_{total}(q)$, iar în cazul exemplului prezentat mai sus are forma din Figura 2.17.

Pe baza acestui câmp potenţial poate fi determinată pas cu pas traiectoria unui robot, situat în mediul de lucru, care trebuie să se deplaseze până la ţintă, ocolind obstacolele [46].

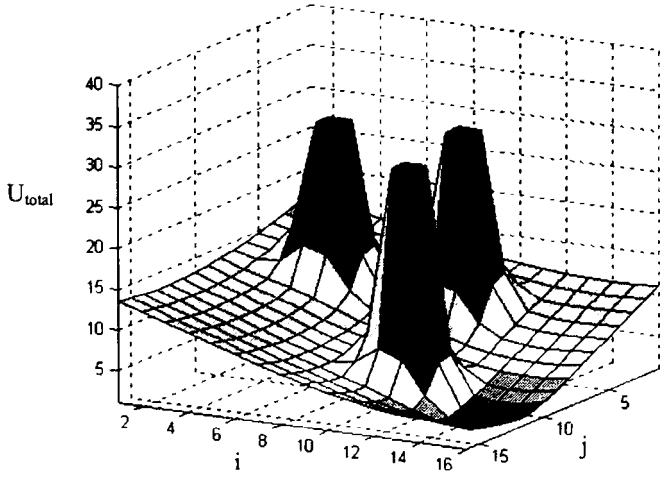


Figura 2.17. Forma câmpului potenţial artificial total.

2.6. Sisteme de conducere a roboţilor mobili

În decursul evoluţiei roboţilor mobili s-au materializat trei forme de organizare a sistemelor de conducere a acestora. Astfel, au apărut sisteme de conducere respectiv de comandă:

- **ierarhice**;
- **reactive** (comportamentale);
- **hibride** (deliberativ / reactive).

Procesele determinate de către sistemul de comandă al roboţilor mobili pot fi împărţite, în general, în trei clase: PERCEPŢIE, PLANIFICARE şi ACŢIUNE [45],[47].

În categoria PERCEPŢIE sunt incluse acele funcţii care preiau informaţia senzorială şi o transformă într-o formă utilă pentru celelalte funcţii. Funcţiile care preiau rezultatele unei funcţii din clasa PERCEPŢIE sau direct informaţia senzorială şi oferă la ieşire un plan ce trebuie executat, aparţin categoriei PLANIFICARE. Funcţiile din categoria ACŢIUNE produc comenzi pentru actuatori, pe baza informaţiilor obţinute de la funcţii din cele două categorii menţionate anterior.

2.6.1. Sistemele de conducere ierarhice

Din punct de vedere cronologic, **sistemele ierarhice** au fost cele dintâi elaborate, dominând arhitectura sistemelor de conducere a roboţilor mobili în perioada 1967 - 1990 [45]. Acest model de conducere are o descompunere "de sus în jos", încercând să mimeze modelul uman de gândire. Orice deplasare a robotului, efectuată în această manieră, presupune existenţa a trei paşi distincţi: în primul pas este perceput mediul înconjurător, apoi în următorul pas se face planificarea mişcării şi în final, în ultimul pas, se efectuează deplasarea propriu-zisă a robotului.

Metodele de conducere ierarhice sunt caracterizate de faptul că informaţia senzorială este concentrată într-un singur model complex al mediului, model ce este apoi folosit de către funcţiile din clasa PLANIFICARE.

Sistemele de conducere ierarhice au avantajul că realizează o ordonare a relaţiei dintre cele trei blocuri principale. Principalul dezavantaj al acestora este legat de modul greoi în care se face planificarea. La fiecare ciclu, robotul actualizează un model global complex al mediului şi re-execută partea de planificare. Atât algoritmii cât şi puterea de calcul disponibilă, mai ales la acea dată, nu puteau face faţă volumului mare de informaţie ce trebuia procesat. Efectul imediat a fost viteza de deplasare extrem de mică (cca. 4m/h).

Definitoriu pentru sistemele ierarhice este despărţirea PERCEPŢIEI de ACŢIUNE, aspect care elimină orice acţiune declanşată de vreun stimul senzorial. De asemenea, dependenţa de un model al mediului este o altă caracteristică definitorie. Această dependenţă a creat mari probleme, deoarece robotul făcea faţă cu greu oricărei modificări a mediului sau incertitudini (zgomot senzorial, semantică bogată, erori ale actuatoarelor) în modelarea acestuia.

În cadrul teoriei conducerii ierarhice nu se face practic, analogia cu sistemele biologice în care informaţia senzorială este conectată direct la actuatoare (exemplul organismelor biologice). Pe de altă parte, datorită conjuncturii în care această paradigmă s-a născut şi maturizat (procesoarele perioadei respective nu ofereau o putere de calcul mare) au fost căutate noi forme de organizare a "inteligenţei" robotice.

2.6.1.1. Arhitecturi de conducere ierarhice

Sistemele de conducere ierarhice au o structură de procesare secvenţială a semnalelor (Figura 2.18), [45]. Robotul percepe lumea înconjurătoare, construieşte o hartă globală a acesteia şi apoi cu "ochii închişi", planifică succesiunea de comenzi necesare îndeplinirii sarcinii curente. În faza finală, robotul execută prima comandă din setul de comenzi stabilit anterior. Acest ciclu se repetă continuu, astfel că după încheierea sa, robotul poate constata consecinţele acţiunii anterioare.

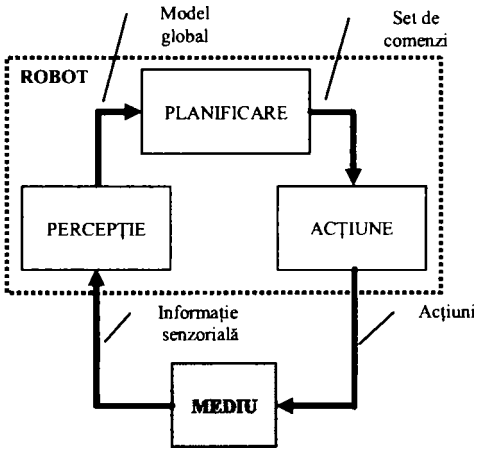


Figura 2.18. Schema bloc a unui sistem de conducere ierarhizat.

În cadrul arhitecturilor de conducere ierarhizată percepţia mediului este monolitică în sensul că toată informaţia senzorială este fuzionată într-un model unic, global, utilizat mai apoi de către blocul de planificare. Acesta este un model al mediului (*world model*) şi reprezintă un concept larg, deoarece robotul include aici întreaga lume înconjurătoare lui.

Informaţiile pe baza cărora se iau deciziile, cu privire la deplasarea robotului, într-un asemenea model, pot fi:

- o reprezentare *a priori* a mediului în care operează robotul (spre exemplu, o hartă a clădirii);
- informaţia senzorială captată;
- cunoştinţe adiţionale utile pentru îndeplinirea sarcinilor.

Crearea unei astfel de reprezentări globale, care să cuprindă toate aceste categorii de informaţii nu e o sarcină uşor de rezolvat mai ales într-un timp scurt, ceea ce influenţează negativ viteza de deplasare a robotului mobil.

Una din structurile de conducere ierarhizate reprezentative, propusă de Meyster în 1982 sub denumirea de NHC (***N**ested **H**ierarchical **C**ontroller*), (prezentată în Figura 2.19), este o structură de conducere specializată în planificarea şi controlul traiectoriei unui robot mobil.

Informaţia senzorială este transmisă la intrarea blocului PERCEPŢIE. Pe baza acestora şi pe baza informaţiilor *a priori*, aflate în baza de date a robotului, se va elabora modelul global al mediului înconjurător.

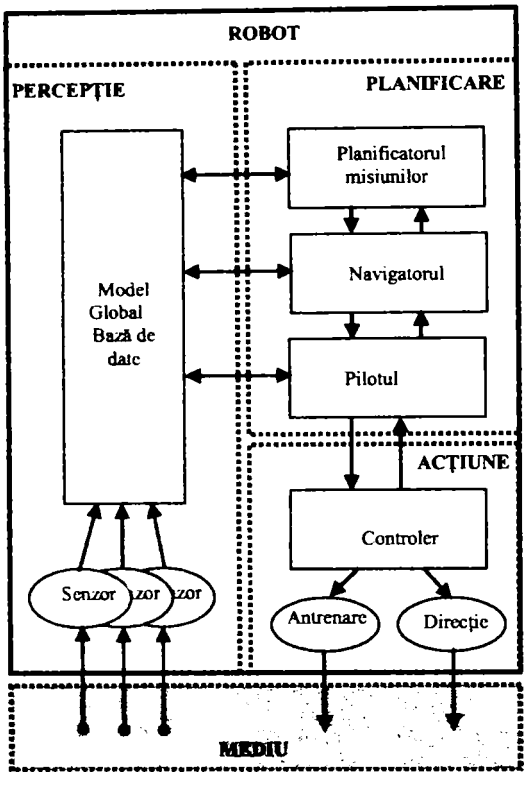


Figura 2.19. Structura de conducere ierarhizată NHC (Nested Hierarchical Controller).

Elementul caracteristic al acestei structuri de conducere îl reprezintă divizarea blocului de PLANIFICARE în trei componente: **Planificatorul misiunilor**, **Navigatorul** şi **Pilotul**. Pentru a-şi putea îndeplini rolul, fiecare din aceste trei componente au acces la modelul global.

Planificatorul Misiunilor poate primi o sarcină de la un operator uman sau poate creea una, cum ar fi spre exemplu: "culege cutia aruncată şi depoziteaz-o în coşul de gunoi". În acelaşi timp acesta are şi sarcina de a traduce noţiuni de genul: "cutie" sau "coş de gunoi" în termeni care să poată fi înţeleşi de către nivelul inferior al **Navigatorului**, definind astfel implicit poziţia actuală şi cea finală a robotului.

Navigatorul preia toate aceste informaţii şi generează o traiectorie între locaţia curentă şi ţintă. În principiu, se generează un şir de puncte de trecere, care unite prin segmente de dreaptă reprezintă traiectoria ce trebuie urmată. Primul segment al acesteia este transmis apoi **Pilotului** care determină ce acţiuni sunt necesare pentru parcurgerea lui.

Pilotul oferă la ieşirea sa vectori viteză de tipul: "întoarce la stânga", "mergi înainte cu 0.6m/s", etc., care sunt transmişi controlerului sistemului de acţionare al robotului. La nivelul acestuia, vectorii viteză sunt convertiţi în profiluri de viteză pentru o deplasare lină.

Controler-ul motoarelor împreună cu actuatoarele robotului constituie cel de-al treilea bloc de bază al unei structuri de conducere ierarhice: ACŢIUNE.

Pentru cazul unui mediu static şi cunoscut, când nu apar obstacole dinamice în mediu, ciclul se completează după cum urmează: la încheierea parcurgerii

primului segment, **Pilotul** raportează acest fapt **Navigatorului**, care procesează următorul segment din listă. Operaţiile se desfăşoară în continuare în mod similar, până când după parcurgerea ultimului segment **Navigatorul** confirmă **Planificatorului misiunilor** încheierea deplasării. La rândul său **Planificatorul misiunilor** desemnează eventual o nouă sarcină.

Dacă pe parcursul traseului robotul întâlneşte vreun obstacol, **Pilotul** invocă **Navigatorul**. Acesta din urmă va defini un nou traseu în funcţie de noua înfăţişare a modelului mediului. Când acest proces este încheiat, mişcarea este reluată în maniera descrisă anterior.

Dintre avantajele structurii NHC trebuie remarcată calitatea acesteia de a îmbina partea de planificare cu cea de deplasare. Robotul construieşte un traseu, începe să-l străbată şi dacă mediul se modifică, atunci schimbă şi traseul. De notat că descompunerea ierarhică este făcută atât ca nivel al inteligenţei, cât şi ca scop. **Planificatorul misiunilor** este mai "inteligent" decât **Navigatorul**, care la rândul său este mai "inteligent" decât **Pilotul**.

Din păcate, deşi aveau un grad de specializare ridicat în rezolvarea sarcinilor tipice de deplasare, capacitatea de adaptare a acestor tip de roboţi, construiţi până în 1990, era mică sau chiar inexistentă.

2.6.2. Sistemele de conducere reactive (comportamentale)

Teoria conducerii reactive apare în a doua jumătate a anilor '80, în parte datorită in-succeselor paradigmei ierarhice, dar şi datorită afluxului de idei din ştiinţe ca: psihologia cognitivă, biologie sau etnologie. Cel care a declanşat acest entuziasm a fost Rodney A. Brooks (1986), el fiind considerat şi principalul promotor al acestei paradigme. În mod paradoxal însă, iniţial paradigma reactivă a fost prezentată ca o extensie a celei ierarhice, deşi ulterior s-a demonstrat o totală discordanţă între cele două proceduri [45].

În cadrul teoriei reactive este eliminată partea de planificare, bazându-se pe sloganul: "Cel mai bun model al mediului este el însuşi!" [45]. Beneficiul imediat al eliminării modelului global este viteza mare de reacţie şi până la urmă, de deplasare. Robotul deţine mai multe dublete PERCEPŢIE - ACŢIUNE (denumite *comportamente*) care pot rula şi în paralel, fiecare fiind particulară unei anumite situaţii.

Prin conducerea roboţilor pe baza comportamentelor s-au obţinut imediat succese notabile. Roboţii conduşi reactiv erau capabili de viteze uimitoare, mişcându-se dintr-o parte în alta cu o dexteritate mare, spre deosebire de cei conduşi ierarhic care erau recunoscuţi pentru viteza lor mică de deplasare. Datorită acestor succese, paradigma reactivă a dominat scena roboţilor mobili din 1988 până în 1992.

Încă din anii '80 sau căutat soluţii pentru ca roboţii mobili să poată fi mai versatili, mai robuşti la modificările mediului. Atenţia a fost îndreptată spre agenţii naturali. Astfel, o serie întreagă de cercetători, dintre care se disting Moravec şi mai ales Arbib, au început să investigheze modele ale comportamentului agenţilor animali - studiate până atunci doar în biologie şi ştiinţe cognitive - în speranţa de a găsi un model de conducere mai viabil [45].

Tot atunci, Valentino Braitenberg prezintă o serie de idei şi experimente care sunt prezentate în cartea *Vehicles: Experiments in Synthetic Psychology*. Experimentele pornesc de la sisteme de comandă simple, bazate pe perechea PERCEPŢIE - ACŢIUNE. Ideile sale au fost intuitive şi intenţionau să copieze principiile de evoluţie a primatelor.

Aceste incursiuni şi multe altele în studiul ştiinţelor biologice au deschis o nouă cale în evoluţia roboţilor mobili, şi anume cea a **conducerii reactive**. Apar astfel, noi metode de organizare şi un nou mod de dobândire a inteligenţei, care au condus în cele din urmă la naşterea, la mijlocul anilor '80, a sistemelor de **conducere comportamentală** (*Reactive Paradigm*).

2.6.2.1. Comportamentele agenţilor biologici

Studiul ştiinţelor cum ar fi: biologia, etnologia sau psihologia cognitivă, pentru dezvoltarea inteligenţei roboţilor mobili autonomi, se bazează pe faptul că evoluţia vieţii a condus la adaptarea remarcabilă şi uneori chiar surprinzătoare a agenţilor biologici la mediul lor natural [45].

Studiul agenţilor biologici poate oferi un punct de plecare şi poate uneori să ofere confirmări directe asupra teoriilor de conducere propuse de către cercetători. Astfel, doar prin faptul că un agent natural procedează într-un anumit fel sau efectuează o anumită acţiune se poate constitui prin ea însăşi într-o validare. Capabilitatea de fuzionare a informaţiei senzoriale a agenţilor naturali este încă un deziderat abia întrezărit în cazul unui robot. De asemenea, în timp ce agenţii actuali fac uz de calităţi moştenite, roboţii mobili se bazează încă pe programe compilate. Mai mult decât atât, agenţii biologici evoluează într-un mediu deschis, complet nestructurat, în timp ce roboţii mobili ai prezentului sunt încă "închişi" în medii puternic structurate.

Conceptul de agent aparţine inteligenţei artificiale şi permite studiul proprietăţilor inteligenţei fără a detalia modul în care agentul a câştigat această experienţă. Astfel, un agent este o entitate independentă, de sine stătătoare. Aceasta deţine propriul creier (sistem de conducere) şi este capabil să interacţioneze cu mediul înconjurător. Este, în plus, capabil să "înţeleagă" ceea ce se întâmplă în jurul lui şi chiar mai mult, are conştiinţă de sine. Orice agent este deci capabil să dezvolte o serie de acţiuni caracteristice (comportamente). Noţiunea de **comportament** (*behavior*) este cel de-al doilea pas pe calea abstractizării propuse. Comportamentul este, conform teoriei cognitive, celula elementară a inteligenţei artificiale. Un comportament este în cele din urmă un mod particular de a transforma informaţia senzorială în acţiuni ale actuatoarelor.

Conform teoriei inteligenţei artificiale comportamentele se pot diviza în trei mari categorii (Figura 2.20) [45]:

- **comportamente reflexive**, răspund direct stimulului din mediu şi sunt dobândite prin moştenire. Un exemplu al acestui tip de comportament este mişcarea involuntară a piciorului atunci când se aplică o lovitură uşoară sub genunchi. Comportamentele reflexive sunt generate în principal de excitaţia unui neuron din zona de percepţie şi transmitere a excitaţiei la efector, obţinându-se în acest mod, cel mai scurt timp de răspuns posibil.
- **comportamentele reactive**, sunt cele care se deprind prin învăţare şi care se execută apoi fără conştientizarea acţiunilor, dar pot fi controlate şi prin conştientizarea lor. Această categorie de comportamente mai este cunoscută şi sub denumirea de "memorie a muşchilor". Exemple de comportamente reactive sunt: mersul, înotul şi mersul cu bicicleta.
- **comportamentele conştiente**, constituie cea de-a treia clasă şi sunt, de fapt, acţiuni complexe, compuse prin "concatenarea" unor comportamente dobândite.

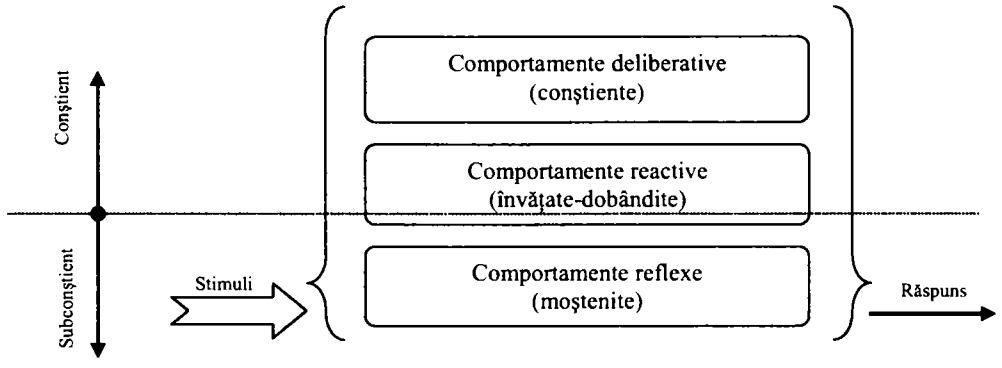


Figura 2.20. Împărţirea comportamentelor.

În cazul roboţilor mobili este interesantă implementarea comportamentelor reflexive deoarece acestea nu implică nici un fel de model al mediului dar se bazează în schimb pe o filozofie de genul: dacă percepi stimulul, acţionează [45].

Modelarea comportamentelor, în teoria inteligenţei artificiale, adaugă fiecărui comportament un mecanism de declanşare (Figura 2.21).

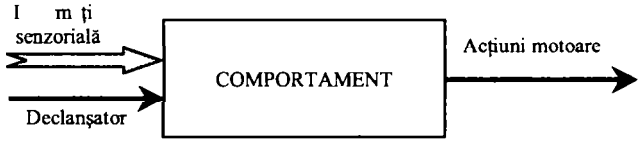


Figura 2.21. Comportament cu declanşare.

Această modelare pare a fi rezonabilă, pentru ca un anumit comportament să poată fi declanşat doar în anumite condiţii. Spre exemplu, animalul de pradă nu va recurge la acţiuni de vânătoare până când nu apare senzaţia de foame care în acest caz este declanşatorul comportamentului de vânător. Astfel, percepţia mediului (incluzând starea internă a robotului) în sistemele reactive are două roluri: fie ghidează comportamentul, prin furnizarea continuă a datelor din mediu, fie declanşează comportamentul.

Comportamentele pot fi dobândite prin moştenire sau învăţare. Cel mai potrivit exemplu este aici reflexul de hrănire moştenit al tuturor agenţilor naturali. Există chiar exemple de secvenţe de comportamente moştenite, cum ar fi spre exemplu, ciclul de împerechere al păsărilor. O altă formă de moştenire a comportamentelor este cea de moştenire cu memorare. Cazul albinelor este un exemplu potrivit în acest sens. Fiind născute în cuiburi, albinele trec printr-o perioadă de învăţare a felului în care cuibul lor arată din exterior. În acest proces albina efectuează câteva zboruri scurte de ordinul metrilor, îndepărtându-se şi apoi revenind la cuib. Începe cu un zbor perpendicular pe urdinişul cuibului şi continuă cu zboruri oblice din ce în ce mai lungi. Interesant este că acest procedeu de memorare a locaţiei este efectuat de fiecare dată după ce cuibul (stupul) a fost deplasat [45]. Prin învăţare pot fi dobândite comportamente dintre cele mai complexe. Exemplul primatelor, dar nu numai, este grăitor prin el însăşi. Puii acestora petrec o bună perioadă din viaţa lor învăţând.

Modurile prin care pot fi dobândite comportamentele agenţilor naturali sugerează felul în care ar putea fi instruit un robot. La una din extreme se situează varianta în care întregul bagaj de cunoştinţe şi aptitudini este moştenit, sau altfel spus robotul este preprogramat şi astfel cade în sarcina "programatorului" să prevadă toate situaţiile posibile pe care robotul le-ar putea întâlni. Experienţele în acest sens, au arătat că în cazul unui mediu dinamic şi nestructurat această modalitate de programare a roboţilor nu este viabilă.

La cealaltă extremă se găseşte varianta în care robotul învaţă pe parcursul efectuării misiunii toate cunoştinţele necesare. Această variantă este tentantă din punct de vedere al constructorului robotului, care scapă astfel de un efort considerabil, dar nici această variantă nu s-a dovedit a fi viabilă pentru utilizator deoarece timpul necesar dobândirii unor cunoştinţe utile ar fi, în multe cazuri, prea mare.

O variantă de mijloc ar putea fi obţinută prin moştenirea a cât mai multor comportamente, în scopul de a accelera faza de învăţare şi pentru a face robotul cât mai repede util. Este esenţial însă, ca acesta să-şi păstreze interesul pentru dobândirea a noi cunoştinţe şi aptitudini pentru întreg ciclul lui de viaţă.

Principiile care ar putea fi preluate din "experienţa" agenţilor naturali sunt următoarele [45]:

- descompunerea sarcinilor complexe în componente simple, realizabile cu comportamente independente, care să conecteze rapid informaţia senzorială la acţiuni;
- procesarea informaţiei senzoriale ar trebui canalizată doar spre acea parte care este specifică comportamentului (percepere orientată spre scop);
- chiar dacă comportamentele nu sunt interdependente, ieşirile lor pot fi combinate pentru obţinerea unui comportament general, aparent inteligent;
- implementarea la nivelul comportamentelor a unor mecanisme de declanşare.

2.6.2.2. Caracteristici ale structurilor de conducere comportamentală

În contrast cu structura orizontală a sistemelor ierarhice, studiul etnologiei arată că sistemele inteligente din natură au o structură verticală. Într-o astfel de structură, un agent natural debutează cu comportamente elementare pentru supravieţuire şi învaţă pe parcurs o serie de comportamente complexe. Se creează în acest mod căi paralele pentru obţinerea unor manifestări mai sofisticate şi care pot fi privite ca şi niveluri de competenţă suprapuse unul peste altul. Fiecare din niveluri are acces independent la informaţia senzorială şi de asemenea la actuatoare (Figura 2.22).

Paralelismul acestei structuri adaugă un plus de robusteţe, astfel dacă unul din nivelurile superioare se defectează, rămân active cel puţin cele inferioare. În final va rezulta un comportament mai puţin inteligent, dar cel puţin se menţine un grad oarecare de funcţionalitate.

Principala caracteristică a sistemelor comportamentale constă în obţinerea unei manifestări globale prin combinarea comportamentelor elementare. Comportamentele sunt în cele din urmă conectări ale intrărilor senzoriale la actuatoare. Matematic privite, ele pot fi asemănate cu funcţii de transfer.

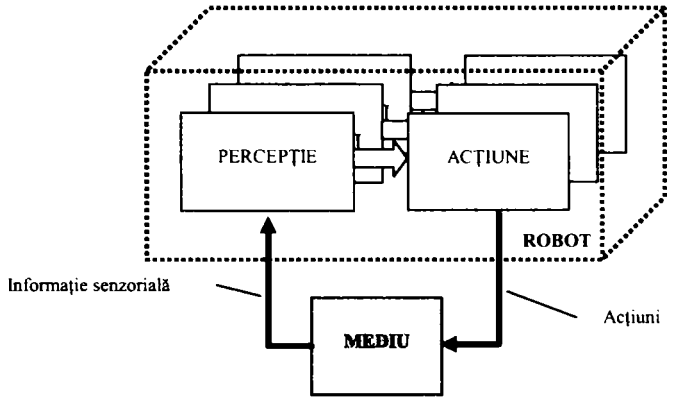


Figura 2.22. Structura unui sistem de conducere comportamental.

Dezavantajul conducerii strict comportamentale este faptul că nu poate fi garantat care din comportamente va guverna conducerea robotului la un moment dat. Nu este posibilă, deci, prezicerea exactă a modului de manifestare a robotului, aşa cum este posibilă în cazul unui sistem de conducere precompilat unde succesiunea instrucţiunilor determină în mod univoc un anumit comportament. În cazul sistemelor de conducere comportamentală, trebuie deci rezolvată problema acţiunilor greşite impusă mai ales de unele domenii cum ar fi operaţiunile militare sau utilizarea roboţilor în uzine nuclearo-electrice. De asemenea, trebuie luat în calcul că o structură reactivă pură nu permite decât o percepere locală a mediului.

Totuşi, datorită vitezei mari de reacţie, care a condus în final la obţinerea unor viteze mari de îndeplinire a misiunilor, conducerea comportamentală a fost larg acceptat în cele din urmă.

Primele sisteme comportamentale au promovat ideea: un senzor - un comportament. Ulterior, pentru a obţine componente mai complexe, s-a acceptat varianta în care acelaşi comportament obţine informaţie de la mai multe tipuri de senzori, sau acelaşi senzor furnizează date mai multor comportamente, dar fiecare comportament are propria sa schemă de procesare. În oricare din variantele amintite comportamentele lucrează în paralel, fără a avea cunoştinţă unul de existenţa altuia.

Informaţia senzorială este disponibilă imediat schemelor perceptuale ale comportamentelor care efectuează doar atâta procesare cât este necesară. Dacă se utilizează, spre exemplu, un atribut de tipul "evident" (uşor de extras) atunci partea de percepere se execută practic instantaneu şi acţiunea asupra actuatoarelor survine rapid.

Pot fi enumerate două avantaje importante ale conducerii comportamentale [45]:

- **răspunsul rapid** (în timp real) al roboţilor la stimuli, deoarece percepţia cu acţiunea sunt strâns legate. Pentru aceasta, comportamentele se implementează hardware sau cu un minim de cod, conform unui algoritm simplu.
- **lipsa memoriei** la roboţii reactivi, ceea ce conduce la un răspuns pur reactiv. Robotul răspunde la modificările din mediu, prin declanşarea comportamentelor corespunzătoare stimulilor şi nu se bazează pe o informaţie stocată.

2.6.3. Sisteme de conducere hibride (deliberativ/reactive)

Sistemele hibride denumite şi deliberativ/reactive au apărut la începutul anilor '90, deoarece s-a ajuns la concluzia că eliminarea etapei deliberative (PLANIFICARE) prezintă şi unele dezavantaje şi că soluţia optimă trebuie căutată undeva între metodele reactive şi ierarhice.

Metodele de conducere hibride constituie actualmente principala direcţie de cercetare în domeniu [45]. Teoria hibridă readuce pe scenă componenta de planificare, dar o face într-o manieră care să nu deranjeze prea mult eficienţa perechii PERCEPŢIE - ACŢIUNE. Astfel, robotul descompune mai întâi sarcina în obiective mai mici ce pot fi rezolvate la nivel reactiv. După încheierea ciclului deliberativ, planificarea şi execuţia decurg simultan. Descrisă în termenii blocurilor elementare, paradigma hibridă are structura: PLANIFICARE, PERCEPŢIE - ACŢIUNE.

Sistemele de conducere reactive se bazează pe faptul că, prin intermediul mediului înconjurător, vor comunica între ele diverse comportamente. Apar însă probleme la implementarea acestora, legate de faptul că stări asemănătoare ale mediului pot să difere în funcţie de context, caz în care robotul poate face confuzii grave. Pe de altă parte, apare problema sesizării incorecte a mediului din cauza impreciziei senzorilor sau din cauza modificărilor ambientale. Este de dorit, spre exemplu, ca un robot să-şi aducă aminte de existenţa unui perete care a fost detectat anterior, dar care datorită reflexiilor speculare dispare brusc, deoarece dispariţia subită a peretelui din informaţia senzorială nu reprezintă şi dispariţia lui fizică.

Utilizând un sistem de conducere reactiv se poate obţine un sistem ce lucrează în timp real, chiar cu procesoare simple, comerciale. Această abordare plăteşte însă ca tribut eliminarea completă a oricărui model, a posibilităţii de a memora stările trecute ale robotului şi în plus se mai pierde şi capacitatea de planificare a acestuia. Roboţii reactivi nu sunt capabili să planifice traiectorii optime, să construiască o hartă a mediului, să-şi monitorizeze performanţele şi nici măcar să decidă care este comportamentul cel mai potrivit la un moment dat pentru îndeplinirea unei sarcini.

Datorită acestor observaţii, începutul anilor '90 a găsit comunitatea ştiinţifică din domeniu în situaţia de a încerca să readucă în sistemul de conducere a roboţilor partea de deliberare şi planificare, fără a afecta însă succesul obţinut de conducerea reactivă. Consensul general a acceptat conducerea reactivă pentru nivelul cel mai de jos de conducere al roboţilor mobili, datorită în primul rând succeselor obţinute pe această cale dar şi datorită eleganţei modelului şi a asemănării sale cu sistemele biologice.

Unii cercetători au afirmat că dacă robotul va trebui să evolueze în lumea reală atunci soluţia este conducerea reactivă, iar dacă mediul robotului este complet cunoscut atunci modelele ierarhice sunt soluţia căutată. Sistemele hibride au fost privite iniţial ca fiind o combinaţie nereuşită a celor două lumi. Denumirea lor iniţială a fost de **planificare deliberativă** pentru a le putea deosebi de sistemele ierarhice.

Actualmente, însă, sistemele de conducere hibride sunt apreciate din mai multe motive ca fiind soluţia generală cea mai bună. În primul rând, utilizarea unui nucleu multiproces asincron permite procesarea funcţiilor deliberative separat de conducerea reactivă. Un proces deliberativ care să decidă următorul "scop" poate evolua lent, în timp ce, robotul se îndreaptă spre ţinta curentă fiind condus reactiv cu o rată de actualizare mare. În al doilea rând, aplicând tehnica programării

orientate pe obiect, se pot obţine sisteme uşor acordabile la diferite medii de lucru şi diverse sarcini.

Organizarea sistemelor hibride deliberative pot fi descrise prin: **planifică**, apoi **percepe-acţionează** (Figura 2.23). În componenta PLANIFICARE sunt incluse toate capabilităţile de deliberare, planificare şi chiar cele de modelare. Robotul va delibera şi planifica întâi cum să îndeplinească o anumită sarcină (folosind modelul global disponibil) după care trece la activarea unui set de comportamente care să conducă la executarea misiunii. La îndeplinirea sarcinii curente, setul de comportamente active se va actualiza pentru următoarea sarcină, şi aşa mai departe. Se decuplează astfel partea de planificare (lentă) de cea de execuţie (rapidă).

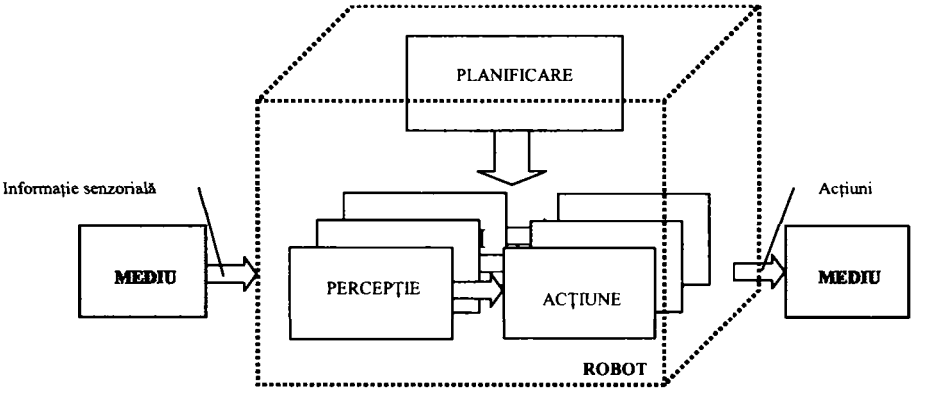


Figura 2.23. Structura unui sistem de conducere hibrid.

Organizarea părţii de percepţie în sistemele hibride este mult mai complexă decât în cazul sistemelor reactive, fiind şi ea hibridă. Astfel, nu doar comportamentele au acces la informaţia senzorială ci şi componenta deliberativă (PLANIFICĂ). În plus, chiar modelul global poate servi ca şi sursă de informaţii pentru comportamente.

Componenta deliberativă a sistemelor hibride conţine module şi funcţii pentru aspecte greu de reprezentat reactiv. Unele din aceste funcţii au nevoie de un model global al mediului (cum ar fi spre exemplu: planificarea traiectoriei sau estimarea poziţiei curente). Dar mai sunt necesare şi cunoştinţe globale de alt gen, cum ar fi spre exemplu modul în care repertoriul disponibil de aptitudini poate fi îmbinat pentru a obţine un anumit mod de comportare. La toate acestea se mai adaugă şi funcţii specifice monitorizării evoluţiei robotului (înaintarea spre ţintă, monitorizarea parametrilor interni ca: temperatura sau starea de încărcare a bateriilor).

Diferenţele dintre diferite sisteme hibride sunt date de modul în care acestea răspund la trei întrebări:

- Care este distincţia între reactiv şi deliberativ?
- Cum sunt organizate responsabilităţile în cadrul porţiunii deliberative?
- Cum se formează comportamentul global?

Principala contribuţie a arhitecturilor hibride este aceea că oferă un model pentru combinarea componentei deliberative cu cea reactivă. Dar îmbinarea celor două componente (în timp real) nu este uşor de realizat, fiind dependentă şi de misiunea curentă. În navigare spre exemplu, trebuiesc cuplate planificarea

traiectoriei cu deplasarea reactivă, dar trebuie monitorizată totodată şi evoluţia robotului, dacă acesta se deplasează spre ţintă sau nu.

Datorită avansului tehnologiei procesoarelor, algoritmii de planificare a traiectoriei care odinioară necesitau timpi de procesare de ordinul a 10-15 minute, actualmente sunt procesaţi în circa o secundă. Scurtarea dramatică a timpului necesar planificării, face ca distincţia între deliberativ şi reactiv să-şi piardă contrastul. Graniţa de separare se păstrează totuşi datorită nivelului noţiunilor cu care se lucrează. Componenta deliberativă va opera cu simboluri şi modele globale, iar cea reactivă procesează direct informaţia senzorială.

Un alt beneficiu al puterii sporite de procesare este oportunitatea de a utiliza modelul global al mediului ca sursă de informaţie pentru aşa numiţii senzori virtuali. Comportamentele pot procesa nu doar informaţia senzorială, pentru a decide reacţia potrivită, ci şi informaţii globale din baza de date a robotului.

Dintre arhitecturile proeminente ale acestor sisteme de conducere hibridă se evidenţiază: AuRA, Sensor Fusion Effect (SFX), Saphira, TCA şi în special arhitectura 3T care este organizată pe trei nivele [48].

2.7. Metodă de planificare a traiectoriei cu metoda câmpului potenţial artificial folosind mediul de simulare Matlab

În aplicaţia de faţă se consideră că un robot mobil trebuie să ajungă dintr-un punct de plecare la un punct ţintă, ocolind obstacolele care îi sunt în cale (Figura 2.24). Se consideră, de asemenea, că mediul conţine doar obstacole statice. Imaginea sa, captată cu o cameră video, este discretizată şi va avea dimensiunea de 32×32 pixeli. Sunt specificate poziţia de plecare pentru robot şi poziţia ţintei. Aceste două poziţii, precum şi robotul sunt reprezentate fiecare printr-un pixel.

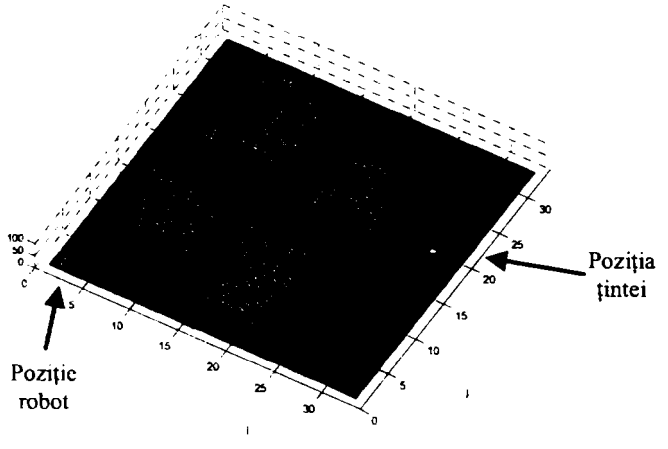


Figura 2.24. Mediul de lucru al robotului.

Deplasarea robotului se va face pas un pas prin spaţiul liber până va atinge ţinta, verificând de fiecare dată poziţiile din jurul său. În poziţiile ocupate de obstacole, robotul nu poate să se deplaseze sau poate chiar să păstreze o anumită distanţă de siguranţă faţă de acestea.

După obţinerea imaginii bidimensionale a mediului cu obstacole, aceasta va fi prelucrată şi se va obţine o imagine tridimensională, cea de-a treia dimensiune fiind dată de către înălţimea obstacolelor (Figura 2.24).

Ştiindu-se poziţia ţintei în plan, se poate crea un câmp potenţial de atracţie, pe baza imaginii mediului de lucru, ce are valorile în fiecare punct, proporţionale cu distanţa din acel punct al mediului şi punctul ţintă unde trebuie să ajungă robotul.

Dacă se notează punctul unde este situată ţinta cu T şi coordonatele sale sunt (x_t, y_t) atunci distanţa din fiecare punct de coordonate (i, j) al imaginii mediului, de dimensiune 32×32 pixeli, până la ţintă este dată de relaţia 2.14.

$$d(i,j)=\sqrt{(x_t-i)^2+(y_t-j)^2} \quad \text{pentru} \quad i=1\ldots32,\ j=1\ldots32. \tag{2.14}$$

Câmpul potenţial va fi creat pe baza unei funcţii potenţiale, care pentru fiecare punct de coordonate (i, j) din mediul de lucru are valoarea dată de relaţia:

$$U(i,j)=\frac{1}{2}\cdot k\cdot d(i,j) \quad \text{pentru} \quad i=1\ldots32,\ j=1\ldots32, \tag{2.15}$$

unde k este un factor de scară pozitiv.

Astfel, valoarea funcţiei potenţiale denumită uneori şi potenţial atractiv va avea minimul în punctul ţintă şi în rest valorile sale în fiecare alt punct din mediu, vor fi proporţionale cu distanţa din acel punct până în punctul ţintă. Peste acest potenţial va fi suprapus un potenţial de repulsie dat de către obstacole şi astfel se obţine câmpul potenţial total prezentat în Figura 2.25.

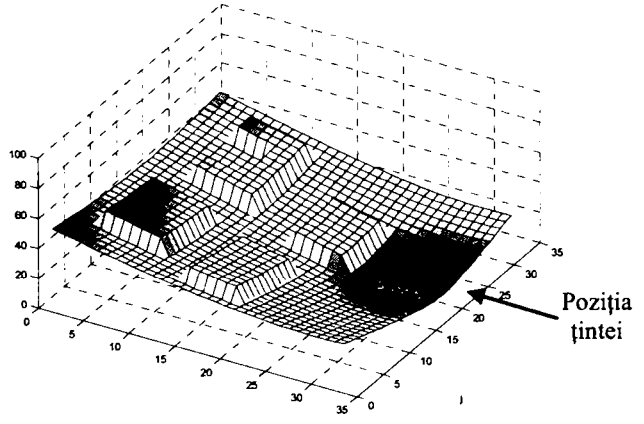


Figura 2.25. Forma câmpului potenţial total.

Distanţa euclidiană are însă, aceeaşi valoare pentru puncte situate simetric faţă de ţintă, ceea ce va determina ca şi potenţialul de atracţie să aibă simetrii. Astfel şi robotul poate fi nevoit în unele situaţii să aleagă între două puncte având acelaşi potenţial. În metoda de planificare a traiectoriei prezentată s-a luat o distanţă euclidiană ponderată în funcţie de valorile i şi j (relaţia 2.16).

$$d_p(i,j)=\sqrt{(x_f-i)^2+p\cdot i+(y_f-j)^2+qj} \quad \text{unde} \quad p\neq q, \tag{2.16}$$

pentru $i=1\ldots32$, $j=1\ldots32$, p şi q fiind coeficienţii de ponderare.

Practic în acest fel se dă o mică înclinare potenţialului de atracţie care nu influenţează semnificativ traiectoria, dar nu vor exista puncte în vecinătatea robotului cu acelaşi potenţial. Înclinarea potenţialului poate fi spre stânga sau spre dreapta robotului în funcţie de care parametru dintre p şi q este mai mare. Metoda a fost simulată în ambele cazuri şi au fost comparate traiectoriile obţinute, acestea fiind egale ca lungime, respectiv ca număr de puncte (pixeli), însă în unele zone acestea prezentau simetrii.

Traiectoria robotului va fi determinată pas cu pas pe baza acestui câmp potenţial, prin identificarea la fiecare pas a pixelului cu valoarea cea mai mică din jurul pixelului ce reprezintă poziţia curentă a robotului. Alegerea direcţiei optime de deplasare se face practic prin compararea valorii potenţialului a celor opt direcţii de deplasare date de punctele cardinale. Direcţiile posibile de deplasare pentru o situaţie concretă se prezintă în Figura 2.26.

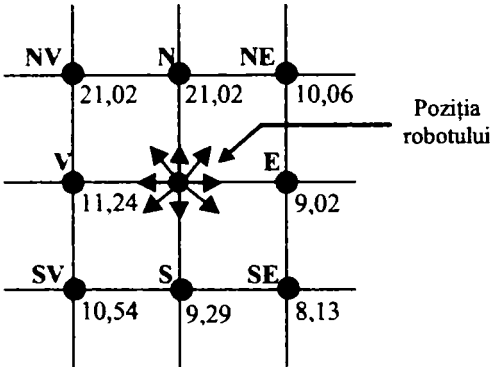


Figura 2.26. Direcţiile posibile de deplasare pentru robot.

Din exemplul de configuraţie prezentat, se observă că direcţia SE are potenţialul minim, prin urmare va fi aleasă această direcţie. Poziţiile N şi NV au potenţial mult mai mare decât toate celelalte poziţii învecinate ale robotului, ele indicând faptul că în direcţia respectivă se află situat un obstacol.

Pentru evitarea blocării robotului în minime locale, care apar în situaţii în care robotul întâlneşte obstacole concave, algoritmul a fost conceput astfel încât robotul să nu poată să facă paşi înapoi. Odată ce robotul a ajuns într-o anumită poziţie, valoarea câmpului potenţial a poziţiei respective va fi actualizată la o valoare egală cu cea dată de un obstacol. La următorul pas al robotului când vor fi comparate valorile potenţialelor punctelor vecine, poziţia de unde a venit nu poate fi luată în calcul pentru o eventuală alegere, şi robotul nu va rămâne pe loc chiar dacă în situaţiile de minime locale potenţialul punctului ocupat de către robot este mai mic decât potenţialul punctelor vecine. Aşadar, având aceste restricţii robotul va reuşi să iasă din aceste minime locale şi va continua drumul către ţintă.

În Figura 2.27 se prezintă rezultatele simulării pentru două poziţii de start din care pleacă robotul mobil, o situaţie în care robotul găseşte drumul spre ţintă fără probleme deosebite (Figura 2.27a) şi una în care reuşeşte să iasă dintr-un minim local (Figura 2.27b).

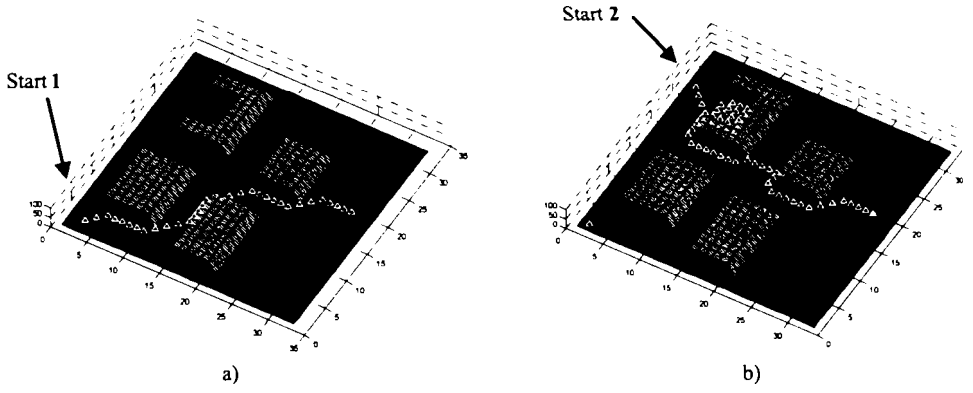


Figura 2.27. Rezultatele simulării în cazul a două poziţii de start diferite pentru robotul mobil: a) traiectoria robotului când acesta nu întâlneşte minime locale; b) traiectoria robotului, când acesta întâlneşte un minim local.

2.8. Concluzii

În general, ţinta ce trebuie atinsă de către robotul mobil nu este direct accesibilă sau uneori chiar trebuie găsită, aşa că navigarea trebuie, în consecinţă, să fie un proces incremental de determinare şi atingere a unor ţinte intermediare, cu scopul de a se obţine în final traiectoria optimă spre destinaţia dorită. În acest sens, navigaţia poate fi descompusă în cinci etape principale: percepţia mediului şi a sarcinii, modelarea mediului, localizarea robotului, planificarea traiectoriei şi executarea deplasării.

Metodele de navigare locale şi globale au fiecare unele dezavantaje şi deci apare implicit necesitatea integrării celor două metode la navigarea roboţilor mobili autonomi. Acest lucru nu înseamnă doar o simplă utilizare concomitentă a acestora, fapt ce poate conduce la multe erori de navigare, ci trebuie realizate numeroase iteraţii planificare / execuţie. De asemenea, trebuie o re-analizare completă a situaţiei, implicând percepţia, modelarea şi localizarea pentru actualizarea contextului şi deci evitarea apariţiei erorilor în faza de execuţie. În acest sens au apărut metodele hibride pentru navigarea roboţilor mobili. Metodele globale ghidează robotul pe o traiectorie optimă care duce spre ţintă, iar metodele locale îl ajută pe acesta să evite obstacolele ce pot apărea în calea sa.

O implementare a metodelor hibride o constituie împărţirea sistemului de conducere al robotului pe nivele ierarhice. Astfel, nivelul ierarhic superior al robotului va coordona deplasarea acestuia pe traiectoria optimă ce duce spre ţintă, iar nivelul ierarhic inferior are sarcina de evitare a obstacolelor şi de ieşire din situaţiile de blocaj.

Prin conducerea roboţilor pe baza comportamentelor, respectiv pe baza conducerii reactive s-au obţinut succese notabile în sensul că aceşti roboţi erau capabili de viteze uimitoare, mişcându-se în mediu o dexteritate mare, spre deosebire de cei conduşi ierarhic care erau recunoscuţi pentru viteza lor mică de deplasare.

Conducerea strict comportamentală are dezavantajul că nu se poate garanta care din comportamente va guverna conducerea robotului la un moment dat. De

asemenea, trebuie luat în calcul că o structură reactivă pură nu permite decât o percepere locală a mediului.

Studiul agenţilor biologici poate oferi un punct de plecare şi poate uneori să ofere confirmări directe asupra teoriilor de conducere propuse de către cercetători. Astfel, doar prin faptul că un agent natural procedează într-un anumit fel sau efectuează o anumită acţiune se poate constitui prin ea însăşi într-o validare.

Utilizarea câmpului potenţial artificial pentru planificarea traiectoriei are dezavantajul minimelor locale în care se poate "împotmoli" robotul. Prin rezolvarea acestor probleme, prin actualizarea *on-line* a câmpului pe baza senzorilor şi prin creşterea capacităţii de calcul a roboţilor, această metodă poate fi aplicată cu succes pentru navigaţia roboţilor mobili.

3. REŢELELE NEURONALE CELULARE

3.1. Noţiuni introductive

O reţea neuronală celulară (CNN - ***C**ellular **N**eural **N**etwork* [49],[50]) este un circuit analogic, neliniar, dinamic şi multidimensional, având o topologie local-recurentă. Astfel, unităţile de circuit numite celule sau neuroni artificiali au fiecare interconexiuni locale. Reţeaua rezultată poate avea diverse arhitecturi cum ar fi: rectangulară, hexagonală, toroidală sau sferică.

Spre deosebire de reţeaua neuronală artificială de tip Hopfield [51], aceste reţele au numai interconexiuni locale şi astfel aria câmpului ocupat de conductoarele de conexiune este mult mai mică. Din acest motiv acestea pot fi implementate în tehnologia VLSI actuală [52]. În consecinţă, fiecare celulă poate interacţiona în mod direct doar cu celulele adiacente sau cu celulele aflate într-o anumită vecinătate, iar celulele neconecate în mod direct se pot influenţa una pe alta datorită efectului de propagare a interacţiunilor locale, respectiv dinamicii continue a reţelelor neuronale celulare.

În cele mai multe aplicaţii, toate celulele precum şi interconexiunile lor sunt presupuse a fi identice, deşi conceptul de reţea neuronală celulară introdus în [49], include posibilitatea de interconexiuni şi parametri de circuit variabile.

Datorită puterii lor mari de calcul, reţelele neuronale celulare sunt ideale pentru realizarea de sarcini mari consumatoare de timp cum ar fi: procesarea imaginilor [53],[54],[55], procesarea în timp real a informaţiilor sau rezolvarea ecuaţiilor diferenţiale parţiale [57],[58].

3.2. Topologia reţelei neuronale celulare de bază

Reţeaua neuronală celulară de bază [49],[50],[52] este o structură bidimensională rectangulară, fiind formată din circuite analogice identice, neliniare, dispuse regulat, numite celule.

În Figura 3.1 se prezintă structura unei reţele neuronale celulare de bază, monostrat, de dimensiune **M×N**, (M linii şi N coloane), precum şi modul de interacţiune a celulelor. S-a notat cu perechea (i, j) coordonatele spaţiale carteziene ale celulei C(i,j) din reţeaua neuronală celulară, unde $1 \le i \le M; 1 \le j \le N$.

3.2.1. Sfera de influenţă a unei celule

Fiecare celulă, respectiv unitate analogică de procesare, comunică în mod direct cu celulele situate în sfera ei de influenţă. Prin sfera de influenţă de rangul r a unei celule C(i,j) se înţelege mulţimea definită de relaţia:

$$S_r(i,j) = \{C(k,l) \mid \max\{|k-i|, |l-j|\} \le r\}, \tag{3.1}$$

unde r este un număr întreg pozitiv.

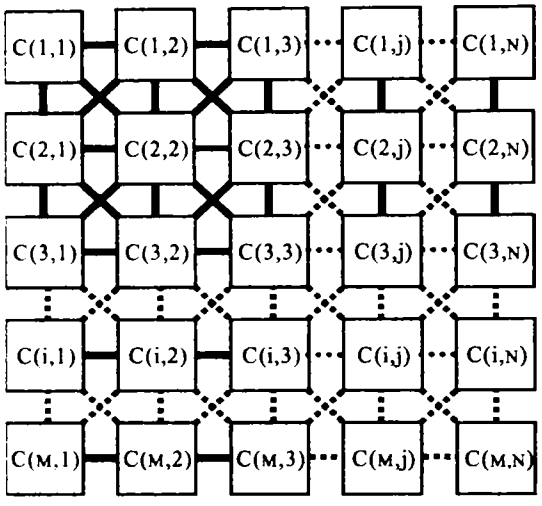


Figura 3.1. Structura de bază a unei reţele neuronale celulare monostrat cu M linii şi N coloane.

Raza sferei de influenţă este, de regulă, mult mai mică decât dimensiunile reţelei. Această regiune este denumită uneori şi câmpul receptiv al celulei şi în cele mai multe cazuri este de rază $r = 1$ (vecinătate 3×3). Pentru o celulă C(i,j) din reţea, această vecinătate este exemplificată în Figura 3.2.

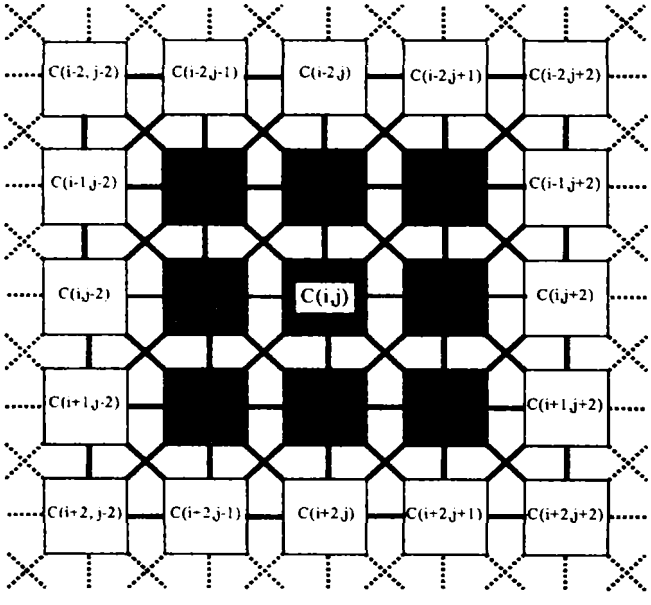


Figura 3.2. Sfera de influenţă a celulei C(i,j) de rază $r = 1$ (vecinătate 3×3).

Astfel, o celulă are conexiuni doar cu acele celule din reţea care se află în sfera ei de influenţă. Celulele între care nu există conexiune directă, pot comunica datorită efectului de propagare a interacţiunilor locale dintre celule, pe durata regimului tranzitoriu din reţea. Astfel, cu toate că interconectările dintre celule sunt

numai locale, se constată că utilizând reţelele CNN, pot fi extrase şi proprietăţi globale ale semnalelor procesate.

3.3. Circuitul electric de bază al unei celule CNN standard

Circuitul electric corespunzător fiecărei celule C(i,j), din cadrul unei reţele neuronale celulare, este prezentat în Figura 3.3.

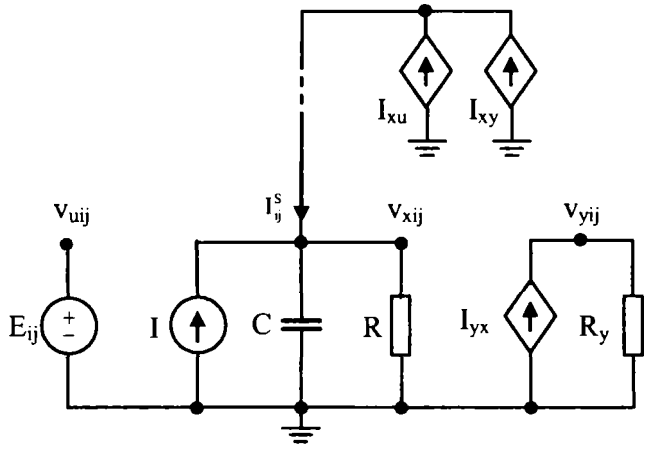


Figura 3.3. Structura circuitului electric pentru o celulă C(i,j).

După cum se observă şi din figură, elementele de circuit aferente unei celule sunt:

- sursa de tensiune independentă $E_{ij} = V_{uij}$, ce constituie mărimea de intrare pentru celulă;
- sursa de curent continuu independentă I, reprezentând polarizarea;
- sursele liniare de curent comandate în tensiune: I_{xu}, I_{xy} şi I_{yx};
- rezistenţele R, R_y şi condensatorul C, identice pentru fiecare celulă.

Cele trei tensiuni care caracterizează starea celulei C(i,j) sunt:

- tensiunea de intrare V_{uij};
- tensiunea de stare V_{xij};
- tensiunea de ieşire V_{yij}.

Curentul I_{ij}^{s} este furnizat de către sursele de curent I_{xy} şi I_{xu}, care sunt comandate cu tensiunile de ieşire V_{ykl}, respectiv cu tensiunile de intrare V_{ukl} ale celulelor C(k,l) aflate în sfera de influenţă $S_r(i,j)$ a celulei C(i,j). Astfel pot fi scrise relaţiile 3.2, 3.3 şi 3.4.

$$I_{xy} = \sum_{\substack{k,l \in S_r(i,j) \\ k,l \neq 0,0}} a_{kl} V_{ykl} \qquad (3.2)$$

$$I_{xu} = \sum_{\substack{k,l \in S_r(i,j) \\ k,l \neq 0,0}} b_{kl} V_{ukl} \tag{3.3}$$

$$I_{ij}^{s} = I_{xy} + I_{xu} = \sum_{\substack{k,l \in S_r(i,j) \\ k,l \neq 0,0}} a_{kl} V_{ykl} + \sum_{\substack{k,l \in S_r(i,j) \\ k,l \neq 0,0}} b_{kl} V_{ukl} = A(i,j;k,l)V_{ukl} + B(i,j;k,l)V_{ykl} \tag{3.4}$$

unde a_{kl} şi b_{kl} sunt operatorii sinaptici ai celulei C(i,j).

A(i,j;k,l) poartă denumirea de operator de reacţie iar B(i,j;k,l) operator de comandă.

Sursa de curent de la ieşirea celulei C(i,j), I_{yx} este comandată de tensiunea de stare a celulei V_{xij} şi are valoarea curentului dată de relaţia 3.5.

$$I_{yx} = \frac{1}{R_y} f(V_{xij}), \tag{3.5}$$

unde funcţia f defineşte caracteristica de ieşire a celulei şi este dată de funcţia sigmoid (relaţia 3.6) aproximată liniar pe porţiuni.

$$f(v) = \frac{1}{2}\left[|v+1| - |v-1|\right]. \tag{3.6}$$

Reprezentarea grafică a funcţiei f(v) este arătată în Figura 3.4.

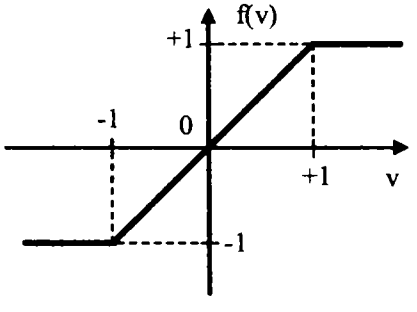


Figura 3.4. Caracteristica de transfer f(v) ce caracterizează ieşirea unei celule.

Se observă că operaţiile de procesare, pe care le realizează o reţea neuronală celulară, sunt determinate de operatorii A(i,j;k,l), B(i,j;k,l) şi curentul I, care împreună formează un aşa zis template CNN, având semnificaţia unor matrice de pondere sau de conexiune.

3.4. Reţeaua neuronală celulară standard

Starea fiecărei celule C(i,j) din reţea, a cărei structură internă a fost prezentată în Figura 3.3, evoluează conform următoarelor relaţii [59]:

- ecuaţia de stare (ecuaţie diferenţială ordinară neliniară)

$$C\frac{dv_{xij}(t)}{dt} = -\frac{1}{R_x}v_{xij}(t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)v_{ukl}(t) + I \quad (3.7)$$

- ecuaţia de intrare

$$v_{uij} = E_{ij} \quad (3.8)$$

- ecuaţia de ieşire

$$v_{yij}(t) = \frac{1}{2}\left[\left|v_{xij}(t)+1\right| - \left|v_{xij}(t)-1\right|\right] \quad (3.9)$$

- condiţii iniţiale

$$\left|v_{xij}(0)\right| \le 1 \quad \text{şi} \quad \left|v_{uij}(t)\right| \le 1 \quad (3.10)$$

$A(i,j;k,l) = A(k,l;i,j)$; C>0 şi R>0,

pentru $1 \le i,k \le M$ şi $1 \le j,l \le N$ cu C(k,l) ∈ $S_r(i,j)$.

Având în vedere relaţiile de mai sus, se pot face următoarele observaţii:

- există posibilitatea implementării VLSI a unor astfel de reţele neuronale celulare prin alegerea unor valori convenabile, din punct de vedere tehnologic, pentru elementele de circuit R, R_y şi C;
- circuitul electric al unei celule are constanta de timp a regimului tranzitoriu: τ_{CNN}=C×R.
- tensiunea de stare pentru orice celulă şi în orice moment pe durata regimului tranzitoriu este limitată astfel [49]:

$$\left|v_{ij}(t)\right|_{max} = V_{max} = 1 + R_x I + R_x \max\left[\sum_{C(k,l)\in S_r(i,j)} \left\{\left|A(i,j;k,l)\right| + \left|B(i,j;k,l)\right|\right\}\right]; \quad (3.11)$$

- reţeaua neuronală celulară având o structură bidimensională poate fi asociată cu o imagine, fiecare celulă corespunzând unui pixel din imagine. De asemenea, pot fi asociate imagini corespunzătoare stării reţelei (*STATE*), intrării (*INPUT*) şi respectiv ieşirii (*OUTPUT*).

Ţinând cont de relaţiile 3.10, valorile elementelor de imagine sunt asociate astfel: în cazul unei imagini binare, pentru nivelul de alb valoarea −1 şi valoarea +1 pentru nivelul de negru al unui pixel, iar la o imagine cu niveluri de gri (*gray-scale*) pentru nivelul pixelilor de la alb spre negru domeniul de valori [-1, +1] (domeniul valorilor CNN standard).

Pentru simplificarea descrierii reţelelor neuronale celulare se utilizează de multe ori forma normalizată în care R, R_y şi C au valori unitare. Astfel, forma simplificată a expresiei matematice prezentată în relaţia 3.7, ce descrie evoluţia în timp a stării reţelei neuronale celulare, este dată de relaţia 3.12.

$$\dot{x} = \frac{dx_{ij}}{dt} = -x_{ij} + \sum_{C_{kl} \in S_r} A_{ij,kl} y_{kl} + \sum_{C_{kl} \in S_r} B_{ij,kl} u_{kl} + z_{ij} . \qquad (3.12)$$

În acelaşi timp, ecuaţia care exprimă mărimea de ieşire a unei celule va avea forma din relaţia 3.13, iar condiţiile iniţiale sunt exprimate de relaţia 3.14.

$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left[\left|x_{ij} + 1\right| - \left|x_{ij} - 1\right|\right], \qquad (3.13)$$

$$\left|x_{ij}(0)\right| \le 1 \quad \text{şi} \quad \left|u_{ij}(t)\right| \le 1, \qquad (3.14)$$

unde u_{ij} reprezintă intrarea, x_{ij} reprezintă starea, iar y_{ij} reprezintă ieşirea celulei C_{ij}.

Operatorul de reacţie (*feedback template*) s-a notat cu $A_{ij,kl}$, $B_{ij,kl}$ reprezintă operatorul de comandă (*feedforward template*), iar z_{ij} semnifică polarizarea (desemnată prin termenii din limba engleză *threshold*, *bias* sau *current*). Celulele C_{kl} sunt din sfera de influenţă de rază r a celulei C_{ij}, adică $C_{kl} \in S_r$.

În cursul unei operaţii de procesare CNN, starea şi ieşirea variază în timp, pe când intrarea rămâne nemodificată.

3.4.1. Reţea neuronală celulară invariantă în spaţiu

O reţea neuronală celulară este invariantă în spaţiu sau izotropică dacă operatorii $A_{ij,kl}$, $B_{ij,kl}$, z_{ij} nu depind de indicii (i,j).

În acest caz:

$$\sum_{C_{kl} \in S_r} A_{ij,kl} y_{kl} = \sum_{|k-i| \le r} \sum_{|l-j| \le r} A_{k-i,l-j} y_{kl} , \qquad (3.15)$$

$$\sum_{C_{kl} \in S_r} B_{ij,kl} u_{kl} = \sum_{|k-i| \le r} \sum_{|l-j| \le r} B_{k-i,l-j} u_{kl} , \qquad (3.16)$$

$$z_{ij} = z. \qquad (3.17)$$

Mărimile $A_{ij,kl}$, $B_{ij,kl}$ şi z_{ij} vor fi considerate invariante în spaţiu şi timp în toate cazurile, dacă nu se va specifica altfel.

Template-urile frecvent utilizate în reţelele neuronale celulare, respectiv operatorii din cadrul unui template pot fi de mai multe tipuri, după cum urmează [60]:

- **operator liniar de reacţie** $A_{ij,kl}$ cu elemente constante din mulţimea numerelor reale, $a_{kl} \in \mathbf{R}$.
- **operator liniar de comandă** $B_{ij,kl}$ cu elemente constante din mulţimea numerelor reale, $b_{kl} \in \mathbf{R}$.
- operator neliniar de reacţie $A_{ij,kl}$ pentru care:

$A_{ij,kl}y_{kl} = a(y_{ij})$, - funcţie de valoarea de la ieşire a celulelor, y_{ij},
$A_{ij,kl}y_{kl} = a(y_{kl})$, - funcţie de valoarea de la ieşirea celulelor, y_{kl},
$A_{ij,kl}y_{kl} = a(y_{ij} \oplus y_{kl})$, - funcţie de combinaţiile valorilor de ieşire a celulelor, y_{ij}, y_{kl}, unde "$\oplus$" poate reprezenta operaţia de adunare, scădere sau produs.

- operator neliniar de comandă $B_{ij,kl}$ pentru care:

$B_{ij,kl}u_{kl} = b(u_{ij})$, - funcţie de valoarea de la intrarea celulelor, u_{ij},
$B_{ij,kl}u_{kl} = b(u_{kl})$, - funcţie de valoarea de la intrarea celulelor, u_{kl},
$B_{ij,kl}u_{kl} = b(u_{ij} \oplus u_{kl})$, - funcţie de combinaţiile valorilor de intrare a celulelor, u_{ij}, u_{kl}, unde "$\oplus$" poate reprezenta operaţia de adunare, scădere, sau produs.

3.5. Reţeaua neuronală celulară standard cu operatori de dimensiune 3×3

În cadrul unei reţele neuronale celulare cu operatori de dimensiune 3×3, o celulă C(i,j) are stabilite conexiuni cu celulele din sfera ei de influenţă de rangul $r = 1$, $S_1(i,j)$. În Figura 3.5 sunt prezentate interconexiunile cu aceste celule.

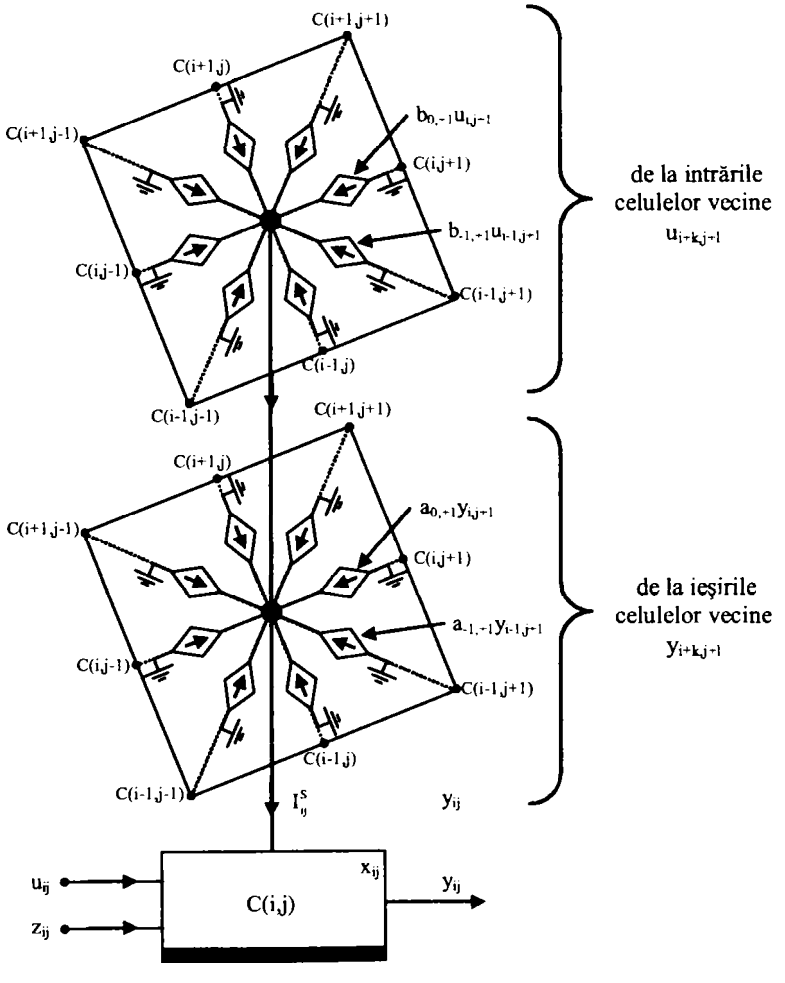


Figura 3.5. Conexiunile celulei $C(i,j)$, în cadrul reţelei neuronale celulare standard cu operatori de dimensiune 3×3.

O celulă CNN de bază sau standard C(i,j), este caracterizată de vectorul de stare x_{ij}, intrarea u_{ij}, pragul z_{ij}, ieşirea y_{ij} şi curentul de intrare de la celulele vecine I_{ij}^{s}. Acest curent depinde de intrarea $u_{i+k,j+l}(t)$ şi starea $x_{i+k,j+l}(t)$ tuturor celulelor aflate în sfera de influenţă $S_1(i,j)$ a celulei C(i,j). Contribuţia de la intrarea $u_{i+k,j+l}(t)$ a fiecărei celule $C(i+k,j+l) \in S_1(i,j)$ este modelată printr-o sursă liniară de curent comandată $b_{kl}u_{i+k,j+l}(t)$ iar contribuţia de la ieşirea $y_{i+k,j+l}(t)$ a fiecărei celule C(i+k,j+l) este modelată tot printr-o sursă liniară de curent comandată $a_{kl}y_{i+k,j+l}(t)$. Aşadar, legăturile sinaptice cu cele opt celule vecine sunt caracterizate prin curentul $I_{ij}^{s}(t)$, respectiv prin 16 coeficienţi a_{kl} şi b_{kl}.

Aceşti coeficienţi sunt reprezentaţi de regulă, sub forma a două matrici de dimensiune 3×3. Astfel operatorul de reacţie A(ij;kl) va avea forma din relaţia 3.12, iar operatorul de comandă B(ij;kl) este dat de relaţia 3.13.

$$A = \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,+1} \\ a_{0,-1} & a_{0,0} & a_{0,+1} \\ a_{+1,-1} & a_{+1,0} & a_{+1,+1} \end{bmatrix} \tag{3.12}$$

$$B = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,+1} \\ b_{0,-1} & b_{0,0} & b_{0,+1} \\ b_{+1,-1} & b_{+1,0} & b_{+1,+1} \end{bmatrix} \tag{3.13}$$

Coeficienţii centrali a_{00} în cazul template-ului A şi b_{00} din template-ul B aparţin celulei C(i,j) corespunzând reacţiei inverse şi directe din cadrul celulei.

După înlocuirea, în ecuaţia de stare, a termenilor corespunzători operatorului de reacţie şi operatorului de comandă se obţin relaţiile 3.14 şi 3.15.

$$\begin{aligned} &\sum_{C_{kl} \in S_r} A_{ij,kl} y_{kl} = \sum_{|k-i| \le 1} \sum_{|l-j| \le 1} A_{k-i,l-j} y_{kl} = \sum_{k=-1}^{1} \sum_{l=-1}^{1} a_{kl} y_{i+k,j+l} \\ &= a_{-1,-1} y_{i-1,j-1} + a_{-1,+1} y_{i-1,j+1} + a_{+1,+1} y_{i+1,j+1} + a_{+1,-1} y_{i+1,j-1} + \\ &+ a_{-1,0} y_{i-1,j} + a_{+1,0} y_{i+1,j} + a_{0,-1} y_{i,j-1} + a_{0,-1} y_{i,j+1} + a_{0,0} y_{i,j} \overset{def}{=} \mathbf{A} \otimes \mathbf{Y_{ij}} \end{aligned} \tag{3.14}$$

$$\begin{aligned} &\sum_{C_{kl} \in S_r} B_{ij,kl} u_{kl} = \sum_{|k-i| \le 1} \sum_{|l-j| \le 1} B_{k-i,l-j} u_{kl} = \sum_{k=-1}^{1} \sum_{l=-1}^{1} b_{kl} u_{i+k,j+l} \\ &= b_{-1,-1} u_{i-1,j-1} + b_{-1,+1} u_{i-1,j+1} + b_{+1,+1} u_{i+1,j+1} + b_{+1,-1} u_{i+1,j-1} + \\ &+ b_{-1,0} u_{i-1,j} + b_{+1,0} u_{i+1,j} + b_{0,-1} u_{i,j-1} + b_{0,-1} u_{i,j+1} + b_{0,0} u_{i,j} \overset{def}{=} \mathbf{B} \otimes \mathbf{U_{ij}} \end{aligned} \tag{3.15}$$

În relaţiile de mai sus s-a notat cu "⊗" operaţia de convoluţie spaţială.

Prin înlocuirea sumelor de corelaţie în expresia ecuaţiei diferenţiale de stare, corespunzătoare unei celule, rezultă relaţia 3.16.

$$\dot{x} = -x_{ij} + \mathbf{A} \otimes \mathbf{Y_{ij}} + \mathbf{B} \otimes \mathbf{U_{ij}} + z_{ij} \tag{3.16}$$

Operaţia pe care o realizează o reţea neuronală celulară asupra unei imagini de intrare $U(t_0)$ pentru obţinerea unei imagini de ieşire stabilă Y, este complet definită de operatorii A, B, z, precizaţi la rândul lor prin cei 19 parametri. Aceste elemente alcătuiesc o instrucţiune elementară CNN.

În Figura 3.6 se prezintă modul în care se face procesarea semnalelor bidimensionale cu o reţea neuronală celulară standard cu operatori de dimensiune 3×3. Astfel, aplicând imaginea U pe intrarea reţelei şi având imaginea X pe starea reţelei, prin procesarea cu operatorii A, B, z se obţine la echilibru imaginea de ieşire a reţelei neuronale celulare, notată cu Y.

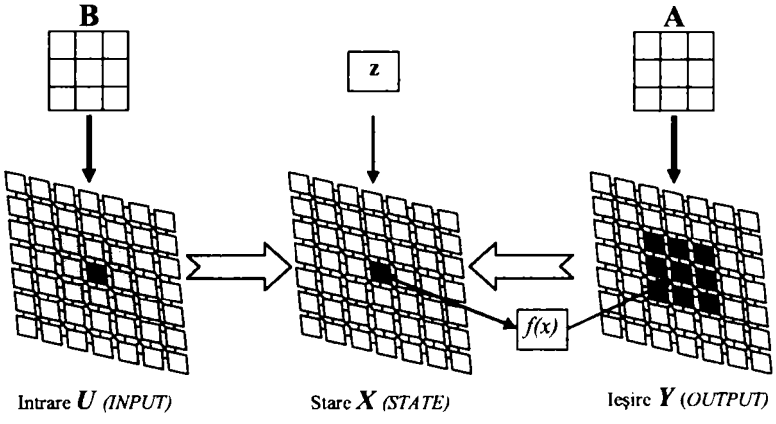


Figura 3.6. Procesarea semnalelor bidimensionale pentru o reţea neuronală celulară standard cu operatori de dimensiune 3x3.

3.6. Reţelele neuronale celulare multistrat

Cei mai mulţi cercetători în domeniul CNN s-au concentrat asupra reţelelor neuronale celulare cu un singur strat, deşi reţele neuronale multistrat au fost introduse odată cu cele monostrat în 1988 [49]. În cazul CNN monostrat, starea fiecărei celule evoluează conform ecuaţiei diferenţiale prezentate în relaţia 3.7 care poate fi denumită reţea neuronală celulară de ordinul întâi. Dacă însă, celulele CNN sunt sisteme dinamice de ordinul n, atunci structura reţelei monostrat poate fi generalizată şi rescrisă ca în relaţia 3.17, [61].

$$F\left(\frac{d^n x_{ij}}{dt^n}, \left\{\frac{d^{n-1} x_{kl}}{dt^{n-1}}\right\}_{C_{kl} \in S_{ij}}, \left\{\frac{d^{n-2} x_{kl}}{dt^{n-2}}\right\}_{C_{kl} \in S_{ij}}, \ldots, \left\{\frac{dx_{kl}}{dt}\right\}_{C_{kl} \in S_{ij}}, x_{kl}\}_{C_{kl} \in S_{ij}}, \{u_{kl}\}_{C_{kl} \in S_{ij}}, z_{ij} = 0\right), \tag{3.17}$$

$$1 \le i \le M; 1 \le j \le N,$$

unde $u_{kl} \in R^m$ este un vector de intrare m-dimensional pentru celula C_{kl}. Această reţea poate fi denumită de ordinul n deoarece fiecare celulă din componenţa sa este un sistem dinamic de ordinul n. Ca şi în cazul reţelei neuronale de ordinul întâi, fiecare celulă din cadrul reţelei neuronale de ordinul n este cuplată doar cu celulele situate în sfera ei de influenţă.

O formă larg răspândită, a reţelei de ordinul n descrisă de ecuaţia 3.17, poartă denumirea de reţea neuronală celulară multistrat [49]. Într-o reţea CNN având k straturi se notează celula situată pe linia i şi coloana j a stratului p cu $C_{ij}^{[p]}$. Ecuaţia de stare a unei celule CNN standard într-o reţea CNN cu k straturi şi fiecare strat având M×N celule este:

$$\dot{x}_{ij}^{[p]} = -x_{ij}^{[p]} + \sum_{q=1}^{k} \sum_{C_{kl} \in S_{ij}} A_{pq}(i,j;k,l) y_{kl}^{[q]} + \sum_{q=1}^{k} \sum_{C_{kl} \in S_{ij}} B_{pq}(i,j;k,l) u_{kl}^{[q]} + z_{ij}^{p},$$

$$1 \le p \le k;\ 1 \le q \le k;\quad 1 \le i \le M;\ 1 \le j \le N, \tag{3.19}$$

unde $x_{ij}^{[p]}$ şi $u_{ij}^{[p]}$ sunt variabilele de stare respectiv de intrare ale celulei $C_{ij}^{[p]}$, $z_{ij}^{[p]}$.este pragul celulei $C_{ij}^{[p]}$, p reprezintă numărul stratului iar q reprezintă numărul de ordine al stratului pe care sunt situate celule din sfera de influenţă a celulei $C_{ij}^{[p]}$. $A_{pq}(i,j;k,l)$ şi $B_{pq}(i,j;k,l)$ sunt operatorii de reacţie respectiv de comandă. Ieşirea unei celule $C_{ij}^{[p]}$, $y_{ij}^{[p]}$ este dată în general de ecuaţia de ieşire standard cunoscută (relaţia 3.20).

$$y_{ij}^{[p]} = f\left(x_{ij}^{[p]}\right) = \frac{1}{2}\left(\left|x_{ij}^{[p]} + 1\right| - \left|x_{ij}^{[p]} - 1\right|\right) \tag{3.20}$$

Ca şi în cazul reţelelor bidimensionale se presupune că intrarea fiecărei celule $u_{ij}^{[p]}$satisface condiţia:

$$u_{ij}^{[p]} \le 1, 1 \le p \le k; 1 \le i \le M; 1 \le j \le N. \tag{3.21}$$

Reţeaua CNN având k straturi, prezentată în relaţia 3.19, este un caz special al reţelei CNN de ordinul k şi are multe avantaje în proiectarea CNN de ordin superior deoarece în multe cazuri este mai uşor de înţeles funcţionarea reţelelor multistrat CNN cuplate de ordinul întâi decât cea a reţelei CNN monostrat de ordin superior.

În Figura 3.7 se prezintă conexiunile pe care le are celula $C_{2,2}^{[2]}$ în cazul unei reţele neuronale celulare cu trei straturi, total conectată, fiecare strat având 3×3 celule.

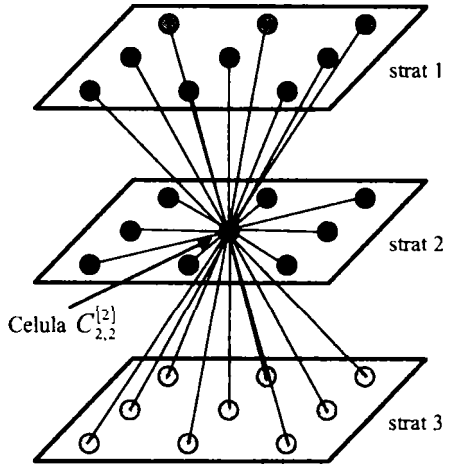


Figura 3.7. Conexiunile unei celule CNN într-o reţea neuronală celulară cu trei straturi.

3.7. Variante de reţele neuronale celulare

Modelul original de reţea neuronală celulară propus în [49], a fost generalizat în mai multe variante, pentru a îmbunătăţi capabilităţile şi eficienţa acestora. Astfel, au fost studiate:

- **CNN cu template-uri neliniare** – comportamentul neliniar este introdus prin template-uri;
- **CNN cu întârziere** – dinamica celulei depinde de valorile anterioare ale variabilelor de intrare şi/sau ieşire ale celulelor vecine;
- **CNN neuniforme sau cu vecinătăţi variabile spaţial** – utile pentru reflectarea unor caracteristici întâlnite la sistemele biologice sau vizuale;
- **CNN discrete în timp (DTCNN)** – pentru fiecare secvenţă de timp se obţine o stare stabilă la ieşire, ceea ce simplifică simularea pe calculator a comportării reţelei.

Ecuaţia dinamică de bază a reţelelor neuronale celulare discrete în timp [62], este dată de relaţia:

$$x_{ij}[t+1] = \sum_{k,l \in S,(i,j)} \left(a_{k-i,l-j} f(x_{kl}[t]) + b_{k-i,l-j} u_{kl}[t] \right) + I, \qquad (3.22)$$

unde $x_{ij}[t+1]$ reprezintă starea celulei C_{ij} la momentul t+1.

3.7.1. Reţele neuronale celulare cu intrare optică

Principalul motiv, pentru găsirea de soluţii în procesarea cantităţilor mari de date, îl constituie necesitatea depăşirii problemelor de transmitere a datelor între sistemele situate în lanţul de prelucrare. Astfel, incorporarea unor circuite electronice de preprocesare la nivelul senzorilor, în sistemele de procesare a imaginilor cu CNN, [63],[64] determină reducerea volumului de date furnizat de aria senzorială şi prin urmare creşte rata de transmisie a datelor. În acest sens, cip-urile CNN cu intrare optică directă au o celulă de calcul analogic la nivelul fiecărui punct

senzorial şi astfel prin combinarea funcţiilor senzoriale şi de procesare se obţine o viteză mare de operare şi în acelaşi timp o arie mică ocupată de circuit.

Fiecare unitate (*smart-pixel*) reprezintă un senzor pentru un pixel din imaginea de intrare şi interacţionează în acelaşi timp cu celelalte unităţi din vecinătatea sa, realizând deci sarcini de procesare paralelă asupra imaginii (Figura 3.8), [65]. În acest fel, calea dintre aria senzorială şi aria de procesare este eliminată, iar în acele aplicaţii ce necesită o procesare ulterioară, cantitatea de date transmisă către procesorul extern este redusă semnificativ. Cipurile CNN de acest tip sunt foarte utile în aplicaţiile de recunoaştere a tiparelor, în care recunoaşterea diverselor trăsături din imagini este esenţială.

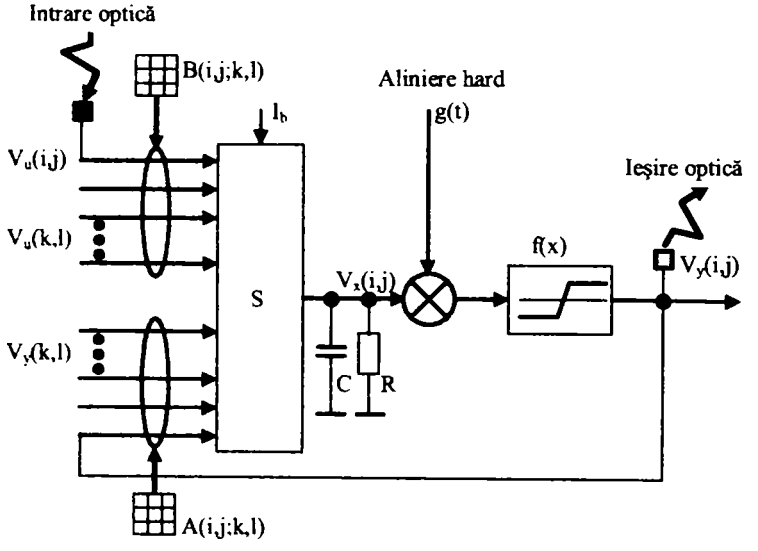


Figura 3.8. Structura unei celule CNN cu intrare optică (smart pixel).

Deoarece semnalele luminoase sunt în mod intrinsec pozitive, ecuaţia dată de relaţia 3.12 trebuie modificată astfel încât să fie implicate doar semnale unipolare. Astfel, se poate scrie forma simplificată pentru ecuaţia de stare a unei celule *c* dintr-o reţea neuronală celulară cu intrare optică denumită şi PSFR-CNN (***P**ositive-**S**ignal **F**ull-**R**ange*):

$$\tau \frac{dx^c}{dt} = g(t) + I_b + \sum_{d \in S_r(c)} \left\{ A_d^c x^d + B_d^c u^d \right\}, \tag{3.23}$$

unde x^c reprezintă starea celulei c iar x^d şi u^d semnifică starea respectiv intrarea corespunzătoare celulelor d situate în sfera de influenţă a celulei c.

Ecuaţia de ieşire a celulei c la un moment de timp m + 1 este dată de relaţia 3.24.

$$y^c(m+1) = f_o\left(x^c(m+1)\right), \tag{3.24}$$

unde funcţia f_o, este funcţia sigmoid unipolară.

Graficul funcţiei $f_o(x)$ este prezentat în Figura 3.9.

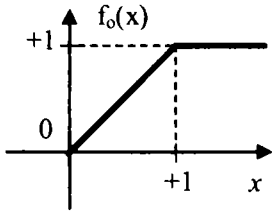


Figura 3.9. Caracteristica de transfer a funcţiei $f_o(x)$.

O aplicaţie tipică pentru care au fost realizate astfel de cipuri CNN este detecţia componentei verticale şi orizontale a unui obiect dintr-o imagine [64]. Astfel, în imaginea de ieşire se obţine o linie verticală sau orizontală având lungimea obţinută prin scanarea obiectului pe direcţia respectivă.

În Figura 3.10 se prezintă modul de captare şi procesare a unei imagini utilizând un cip CNN cu intrare optică:

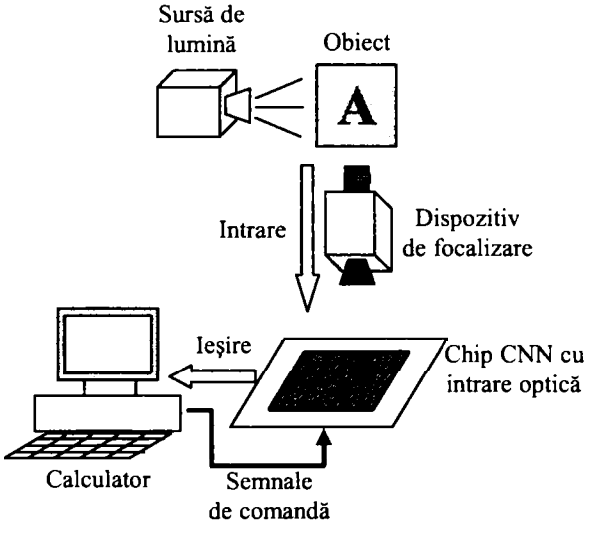


Figura 3.10. Captarea şi procesarea unei imagini cu un cip CNN cu intrare optică.

În Tabelul 3.1 se prezintă comparativ caracteristicile tehnice pentru două cipuri CNN cu intrare optică, având dimensiuni 16×16 celule şi respectiv 32×32 de celule, utilizate pentru detecţia componentelor verticale şi orizontale ale obiectelor din imagini.

3.8. Implementarea şi simularea reţelelor neuronale celulare

Pe plan mondial se manifestă un interes deosebit cu privire la cercetarea în domeniul reţelelor neuronale celulare, mai ales datorită faptului că acestea pot fi implementate în tehnologie VLSI [66],[67]. Astfel a apărut şi calculatorul universal CNN (**CNN-UM** - ***CNN Universal Machine*** [68]), pentru care au fost realizate medii de dezvoltare şi simulare.

Dimensiune cip / Caracteristici	16x16 celule	32x32 celule
Dimensiuni	2,7 x 2,7 mm	5 x 5,8 mm
Nr. tranzistori (aprox.)	2×10^4	8×10^4
Tensiune de alimentare	5 V	5 V
Putere disipată	100 mW	380 mW
Frecvenţa de tact	4 Mhz	10 Mhz
Timp total de procesare	8 μs	28,8 μs
Frecvenţa de tact	4 Mhz	10 Mhz

Tabelul 3.1. Caracteristicile tehnice pentru două cip-uri CNN cu intrare optică.

3.8.1. Calculatorul universal CNN

La baza calculatorului universal CNN stă procesorul analogic sau chipul CNN. Procesorul realizează operaţii analogice, pe baza unui program format din instrucţiuni analogice şi logice, adică prin calcul dual. Semnalele cu care lucrează procesorul analogic pot fi analogice şi numerice şi la fel şi template-urile utilizate pot realiza funcţii analogice sau operaţii logice. Un program CNN analogic conţine algoritmi care se realizează atât secvenţial cât şi paralel. CNN-UM este primul calculator analogic programabil, cu limbaj de programare şi sistem de operare, care, pe un singur chip, are puterea de calcul a unui supercomputer. În Figura 3.11 este prezentată structura unui calculator universal CNN.

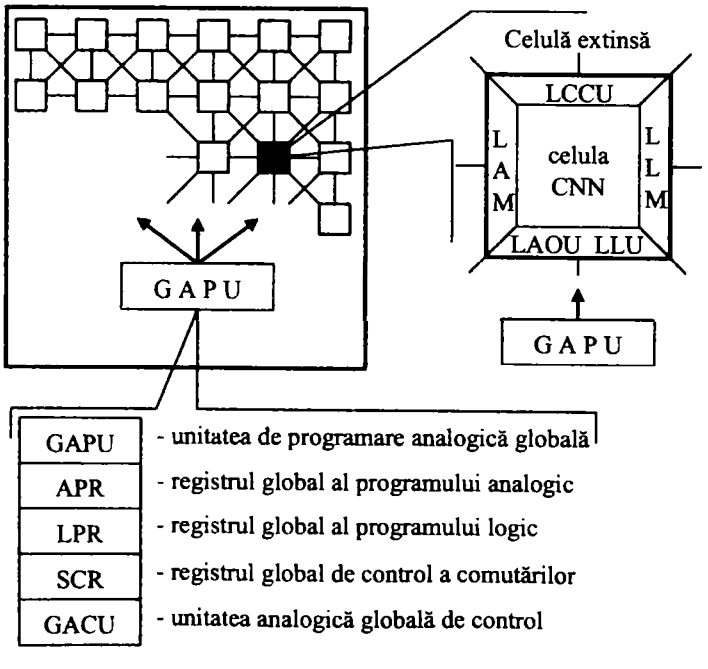


Figura 3.11. Structura calculatorului universal CNN-UM.

Comanda, în ansamblu, a procesorului este realizată de către unitatea de programare analogică globală (**GAPU** - ***G**lobal **A**nalogic **P**rogramming **U**nit*). Această unitate conţine registrul global al programului analogic cu template-uri (**APR** - *Global **A**nalog **P**rogram **R**egister*) şi registrul global cu instrucţiuni logice (**LPR** - *Global **L**ogic **P**rogram **R**egister*). Realizarea corectă a interacţiunilor între celulele reţelei se obţine cu ajutorul unui registru de configurare, care comandă conexiunile locale (**SCR** - ***S**witch **C**onfiguration **R**egister*). Unitatea globală de control analogic (**GACU** - ***G**lobal **A**nalogic **C**ontrol **U**nit*) conţine programul de comandă a instrucţiunilor analogice în cod maşină. Acest program este rezultatul unei compilări a unui program analogic de nivel înalt sau a unui interpretor. O celulă extinsă conţine, în afară de elementele specifice celulei propriu-zise şi alte elemente cum sunt: unitatea logică locală (**LLU** – ***L**ocal **L**ogic **U**nit*), unitatea locală analogică de ieşire (**LAOU** - ***L**ocal **A**nalog **O**utput **U**nit*), unitate locală de memorie analogică cu mai multe elemente de memorie analogică locală (**LAM** - ***L**ocal **A**nalogic **M**emory*), memorie logică locală (**LLM** - ***L**ocal **L**ogic **M**emory*) şi unitate locală de comandă şi comunicare a celulei (**LCCU** - ***L**ocal **C**ommunication and **C**ontrol **U**nit*).

CNN-UM execută programe bazate pe algoritmi analogici, prin instrucţiuni cu template-uri analogice şi operaţii logice (în cazul în care sunt procesate imagini binare). Pentru execuţia unor astfel de programe este nevoie de imaginea originală, de imagini cu rezultate parţiale, de memorarea unor template-uri logice şi analogice.

Memoriile analogice locale (LAM) servesc la memorarea imaginilor cu niveluri de gri cum ar fi imagini de intrare, de stare sau de ieşire. Aceste memorii sunt foarte importante, deoarece existenţa lor asigură memorarea pe chip a imaginilor intermediare, care vor fi supuse unor prelucrări ulterioare cu alte template-uri, conform programului analogic general. Unitatea logică, cu două intrări şi o ieşire, este programabilă şi poate realiza o funcţie logică pentru pixelii corespunzători aceleaşi poziţii din două imagini binare. În acest caz, nu se realizează nici o conexiune logică cu celulele vecine.

Pentru memorarea imaginilor binare se folosesc memoriile logice locale (LLM). Scrierea şi citirea din memoriile LAM şi LLM se poate face pentru o linie întreagă, utilizând valori analogice sau binare, astfel încât transferul pe o magistrală M dimensională a unei imagini complete M×N se poate face în N paşi; astfel, se micşorează substanţial timpul necesar pentru încărcarea şi extragerea imaginilor de pe chip. Reducerea efectivă, a timpului de lucru, se realizează ca urmare a procesării paralele, specifică reţelelor neuronale celulare.

3.8.2. Medii de simulare a reţelelor neuronale celulare

Pentru realizarea proiectărilor, testărilor şi optimizărilor în domeniul CNN au fost concepute diverse medii de dezvoltare şi de simulare. În cadrul prezentei lucrări, au fost utilizate mediul de dezvoltare **CadetWin** (***C**NN **a**pplication **d**evelopment **e**nvironment and **t**oolkit under **Win**dows* [69]) şi simulatorul analogic **MatCNN** [70],[71] din mediul de programare Matlab [72].

3.8.2.1. Mediul de dezvoltare CadetWin

Chipurile CNN, pentru a putea fi utilizate practic, trebuie să fie însoţite de un mediu de dezvoltare de aplicaţii. Mediul de dezvoltare **CadetWin** poate realiza simularea soft precum şi emularea hard a reţelelor neuronale celulare.

Componentele principale ale mediului sunt:

- **Platforma VisMouse** (*CNN **Vis**ual **Mouse** software platform for Windows*, [73]) bazată pe concepţia *visual mouse*. Această componentă este un mediu software integrat pentru proiectarea de aplicaţii CNN. Cu ajutorul acestei platforme se pot achiziţiona, memora şi vizualiza imaginile, înainte şi după prelucrarea cu CNN. De pe această platformă se pot accesa şi alte mijloace software ale mediului de dezvoltare **CadetWin**.
- **TemMaster** (***Tem**plate design and optimization tool for binary input-output CNN's* [74]) este un software dedicat pentru proiectarea şi optimizarea template-urilor.
- **SimCNN** (*Multi-layer **CNN Sim**ulator for visual mouse platform* [75]) este un program de simulare; cu ajutorul lui se pot testa template-uri respectiv se pot proiecta, testa şi optimiza algoritmi ori subrutine spaţiio-temporale, care includ operaţii aritmetice şi logice şi care utilizează reţele neuronale celulare monostrat sau multistrat.
- **Editorul şi compilatorul Alpha** (*CNN **Alpha** language and compiler* [74]) este o componentă care permite editarea şi compilarea programelor într-un limbaj de nivel înalt, specific CNN.
- **Editorul şi interpretorul în limbaj de asamblare AMC** (*Extended **A**nalogic **M**acro **C**ode and interpreter* [77]).
- **Editor şi interpretor de programe pseudocod Script** [76].
- **Interfaţa CPS** (*CNN **C**hip **P**rototyping **S**ystem* [78]), dintre calculator şi platforma cu chip-CNN. Cu ajutorul acestei interfeţe se poate conecta şi aborda în mod unitar oricare platformă cu chip-CNN nou, în vederea testării şi perfecţionării ei. Utilizând platforma CPS (adică o implementare CNN hard) timpul de prelucrare a semnalelor, pe baza programelor AMC, este mult mai scurt decât în cazul simulării soft. Acest lucru se datorează pe de o parte faptului că instrucţiunile AMC, care utilizează template-uri, sunt efectuate chiar pe chip-ul CNN, iar instrucţiunile logice se efectuează direct pe platforma CPS, cu ajutorul procesoarelor numerice pe care aceasta le conţine.
- **Biblioteci software de template-uri şi algoritmi** (*CNN templates and algorithms software library*), facilitează programarea în mediul CNN şi se dezvoltă continuu pe baza rezultatelor cercetărilor în domeniul CNN din întreaga lume [79].

Prin utilizarea mediului de dezvoltare **CadetWin** se pot realiza programe şi algoritmi CNN în limbaj de nivel înalt specific **Alpha** sau în limbaj de asamblare **AMC**. Aceste programe sunt unitare şi compatibile sub aspectul rulării în mediul **CadetWin**, atât pe mijloacele de simulare software sau emulare hardware cât şi pe cipul CNN analogic.

În Figura 3.12 se prezintă etapele de realizare ale unui algoritm analogic, de la proiectare până la simulare soft, emulare hardware respectiv testare direct pe un chip CNN [80].

3.8.2.2. Simulatorul analogic MatCNN

Pentru simularea reţelelor neuronale celulare se poate utiliza şi mediul MATLAB [72], pentru care a fost elaborat un *toolbox* specific (Matcnn), ce poate fi utilizat atât pentru reţele neuronale celulare analogice cât şi pentru reţele neuronale celulare discrete în timp (DTCNN) [62].

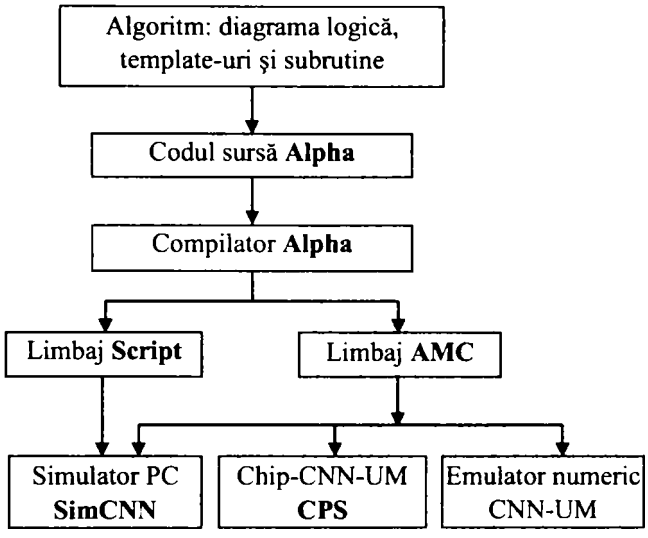


Figura 3.12. Etapele de elaborare şi testare ale unui algoritm analogic CNN.

MatCNN [70],[71] este, de fapt, o colecţie de funcţii şi scripturi care permit simularea şi testarea reţelelor neuronale celulare cu un singur strat. El a fost conceput pe suportul matematic şi de vizualizare oferit de mediul Matlab şi poate fi utilizat ca orice alt *toolbox* din acest mediu complex.

În biblioteca **MatCNN** sunt incluşi o serie de template-uri liniare şi neliniare, utili pentru prelucrarea imaginilor. Aceştia pot fi modificaţi în funcţie de cerinţele aplicaţiei sau pot fi adăugaţi noi template-uri.

Intrarea în mediul de simulare **MatCNN** se realizează cu ajutorul instrucţiunii **SetEnv**. Prelucrarea diverselor tipuri de imagini cu **MatCNN** presupune întâi conversia acestora astfel încât valorile pixelilor să fie în domeniul standard CNN. Astfel, pentru conversia imaginilor în /şi din standardul CNN pot fi utilizate funcţiile [81]:

- **cnn2gray** - converteşte o imagine CNN într-o imagine cu nivele de gri;
- **gray2cnn** - converteşte o imagine cu nivele de gri într-o imagine CNN;
- **lbmp2cnn** - încarcă o imagine de tip BMP de pe disc şi o converteşte într-o imagine CNN;
- **scnn2bmp** - salvează pe disc, în format BMP o imagine CNN;
- **lavi2cnn** - încarcă o imagine dintr-un fişier de tip AVI şi o converteşte într-o imagine CNN;
- **scnn2avi** - salvează pe disc, sub forma unui cadru de imagine de tip AVI o imagine de tip CNN.

Principalele variabile corespunzătoare imaginilor asociate modelului de reţea neuronală celulară utilizat în **MatCNN** [82], sunt:

- **INPUT1** - **U** sau $\mathbf{U_1}$ (imaginea de intrare a reţelei)
- **INPUT2** - $\mathbf{U_2}$ (imagine de intrare secundară)
- **STATE** - **X** (imaginea de stare)
- **OUTPUT** - **Y** (imaginea de ieşire a reţelei)
- **BIAS** - **B** (imaginea de polarizare (*bias map*))
- **MASK** - **M** (imaginea mască).

O altă categorie de variabile globale sunt cele care se impun a fi iniţializate de către utilizator (înainte de efectuarea simulării propriu-zise) [82]. Acestea sunt:

- **UseMask** - validează (UseMask = 1) sau invalidează (UseMask = 0) utilizarea imaginii mască **M;** valoarea implicită este UseMask = 0;
- **UseBiasMap** - validează (UseBiasMap = 1) sau invalidează (UseBiasMap = 0) utilizarea imaginii de polarizare **B;** valoarea implicită este UseBiasMap = 0;
- **Boundary** - variabila prin care se specifică condiţiile de frontieră; poate fi o constantă în intervalul $-1 \leq$ Boundary ≤ 1, zeroflux (Boundary = 2), torus (Boundary = 3); valoarea implicită este Boundary = 2;
- **TemGroup** - permite precizarea bibliotecii din care vor fi extraşi template-urile folosite în program; valoarea implicită este TemLib;
- **TimeStep** - specifică intervalul dintre două momente discrete de timp la care au loc două iteraţii succesive; are valoarea implicită TimeStep = 2;
- **IterNum** - precizează numărul de iteraţii care au loc în cazul unei simulări; are valoarea implicită IterNum = 25.

Pentru afişarea variabilelor implicate în program sunt disponibile instrucţiunile:

- **showenv** - afişează toate variabilele globale ale mediului CNN;
- **showtem** - afişează elementele template-ul actual "încărcat" în mediul CNN;
- **cnnshow** - afişează imaginile CNN (ex. pentru afişarea imaginii de ieşire a reţelei se utilizează instrucţiunea: cnnshow(OUTPUT)).

Simularea unei aplicaţii, cu simulatorul **MatCNN**, se realizează prin parcurgerea principalelor etape prezentate în organigrama din Figura 3.13.

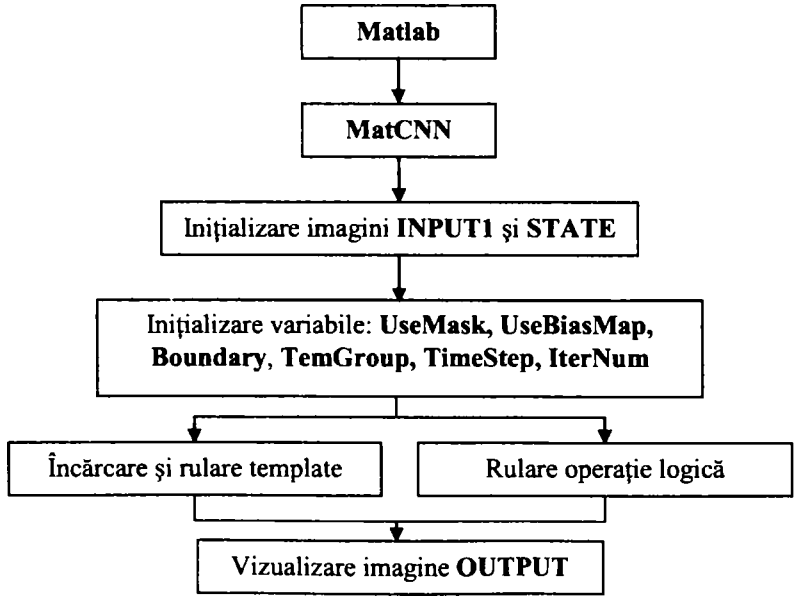


Figura 3.13. Principalele etape de realizare a unei simulări cu MatCNN.

3.8.3. Circuite integrate CNN

În scurta perioadă de timp de când au fost propuse pentru prima oară reţelele neuronale celulare, în laboratoarele de cercetare din întreaga lume s-au proiectat şi realizat o serie prototipuri de cipuri CNN analogice [66],[67],[83],[84], şi circuite de emulare numerice [85].

Circuitul CNN **cP 400** [66] este primul procesor analogic care prezintă caracteristicile specifice unui calculator CNN-UM. Cu ajutorul lui putea fi executat un program analogic, care include operaţii analogice şi logice asupra unor imagini binare din memoria locală. Acest circuit are următoarele caracteristici:

- reţeaua neuronală este formată din 20×22 de celule;
- procesează imagini binare cu valori standard CNN, adică valoarea -1 pentru nivelul de alb şi +1 pentru nivelul de negru;
- ecuaţia de stare care caracterizează funcţionarea unei celule este dată de relaţia 3.25:

$$\frac{dx^c}{dt} = -g(x^c) + \sum A_d x^d + \sum B_d u^d + D_A + D_B \,, \tag{3.25}$$

unde x^c şi u^c reprezintă starea respectiv intrarea celulei c, iar x^d şi u^d semnifică starea respectiv intrarea corespunzătoare unei celule d din reţea, situată în vecinătatea sa. Variabilele D_A şi D_B precum şi polarizările, sunt codate pe 8 biţi, dintre care unul este pentru semn;

- caracteristica funcţiei de transfer neliniare g(x) este prezentată în Figura 3.14;
- la o procesare elementară, în condiţii optime, constanta de timp caracteristică sau timpul de stabilizare pentru o celulă este τ_{CNN} = 250 ns;
- se pot memora local cel mult 4 imagini binare, care pot fi imagini de intrare de la senzorul optic sau de pe platformă, ori imagini de ieşire rezultate în urma unor prelucrări anterioare. La achiziţia imaginilor pragul de binarizare este reglabil;
- pot fi memorate local cel mult 8 template-uri (A, B, z), cu cel mult 20 de elemente reprezentate fiecare, pe câte 8 biţi. Dimensiunea unui operator este de 3×3. Valorile elementelor nu pot depăşi în modul valoarea 3, în cazul operatorilor de tip A şi B, iar în cazul curentului această valoare limită este 6.

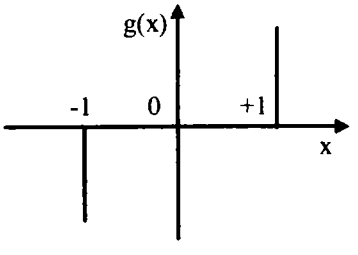


Figura 3.14. Caracteristica de transfer g(x).

Mai târziu a apărut circuitul analogic CNN **cP 4000** [83]. Cele mai semnificative proprietăţi ale acestui circuit sunt următoarele:

- reţeaua este formată din 64×64 de celule;
- utilizează imagini cu niveluri de gri şi imagini binare cu valori standard CNN, (adică este atribuit domeniul de valori [-1,1] pentru nivelurile situate între alb şi negru);
- pot fi memorate pe chip 32 de template-uri (A, B, z);
- poate memora local în LAM cel mult 4 imagini cu niveluri de gri precum şi alte 4 imagini binare în LLM;
- prezintă facilitatea utilizării imaginii mască binară.

Procesorul analogic CNN cu intrare optică, denumit şi microprocesor vizual, **ACE16k** este descris în [67]. Acesta este compus din 128×128 de celule şi incorporează toate circuitele necesare funcţionării unui microprocesor (circuite de comandă, generator de tact - 32 MHz, memorie internă, etc.) precum şi o interfaţă digitală completă. Circuitul poate lucra în două moduri şi anume în aplicaţii în care imaginile care trebuiesc procesate sunt achiziţionate direct de către modulul optic de intrare al cip-ului şi ca coprocesor de imagini, lucrând în paralel cu un sistem digital care furnizează şi recepţionează imaginile sub formă de semnale electrice. Pentru comunicaţie cu alte dispozitive, integratul foloseşte o magistrală de date bidirecţională de 32 de biţi şi mai multe magistrale de adrese.

Avantajul major al procesării paralele prin utilizarea reţelelor neuronale celulare este prezent însă numai în cazul implementării acestora pe un chip. De aceea, proiectarea şi realizarea de chipuri CNN cât mai performante este o direcţie de cercetare pentru care se depun în continuare eforturi umane şi materiale substanţiale.

3.9. Concluzii

Diferenţa esenţială care există între o reţea neuronală celulară şi alte reţele neuronale este că interconexiunile dintre celule sunt în principal locale şi astfel este minimizată aria câmpului ocupat de firele de conexiune. Din acest motiv, reţelele neuronale celulare pot fi uşor implementate folosind tehnologia VLSI actuală.

Fiecare celulă poate interacţiona în mod direct doar cu celulele adiacente sau cu celulele aflate într-o anumită vecinătate, iar celulele neconecate în mod direct comunică pe baza efectului de propagare a interacţiunilor locale respectiv dinamicii continue a reţelelor neuronale celulare. Astfel, cu toate că interconectările dintre celule sunt numai locale, se constată că utilizând reţele neuronale celulare pot fi extrase şi proprietăţi globale ale semnalelor procesate.

Datorită puterii lor de calcul mari, reţelele CNN sunt ideale pentru realizarea de sarcini mari consumatoare de timp cum ar fi: procesarea imaginilor, procesarea în timp real a informaţiilor sau rezolvarea ecuaţiilor diferenţiale parţiale.

Deoarece reţeaua neuronală celulară are o structură bidimensională, poate fi asociată cu o imagine, fiecare celulă corespunzând unui pixel din imagine. De asemenea, pot fi asociate imagini corespunzătoare stării reţelei (*STATE*), intrării (*INPUT*) şi respectiv ieşirii (*OUTPUT*). În cursul unei operaţii de procesare CNN starea şi ieşirea reţelei variază în timp, pe când intrarea rămâne nemodificată.

Operaţia pe care o realizează o reţea neuronală celulară asupra unei imagini de intrare $U(t_0)$ pentru obţinerea unei imagini de ieşire stabilă Y, este complet definită de operatorii A, B, z, precizaţi la rândul lor prin cei 19 parametri (un

template complet). Aceste elemente (template-ul şi imaginile) alcătuiesc o instrucţiune elementară CNN.

Prin incorporarea unor circuite electronice de preprocesare la nivelul senzorilor, în sistemele de procesare a imaginilor cu CNN, se obţine reducerea volumului de date furnizat de aria senzorială şi prin urmare creşte rata de transmisie a datelor. În acest sens, cip-urile CNN cu intrare optică directă au o celulă de calcul analogic la nivelul fiecărui punct senzorial (*smart-pixel*) şi astfel prin combinarea funcţiilor de senzor şi de procesare se obţine o viteză mare de operare şi în acelaşi timp o arie mică ocupată de circuit. Cipurile CNN de acest tip sunt foarte utile în aplicaţiile de recunoaştere a tiparelor, în care recunoaşterea diverselor trăsături din imagini este esenţială.

4. PLANIFICAREA TRAIECTORIEI ROBOŢILOR MOBILI PRIN PROCESĂRI CNN

4.1. Noţiuni introductive

O temă importantă de cercetare în domeniul roboţilor mobili, o constituie proiectarea sistemului de planificare a traiectoriei. Un astfel de sistem trebuie să-i furnizeze robotului traiectoria optimă posibilă pentru deplasarea lui de la o poziţie iniţială la o poziţie ţintă, evitând obstacolele care sunt localizate între aceste două poziţii. Dacă se ia în calcul că obstacolele precum şi ţinta pot fi în mişcare [86], şi că obstacolele pot avea diverse forme, este evident că nu e o problemă uşor de rezolvat.

Planificarea traiectoriei unui robot mobil presupune întâi de toate cunoaşterea mediului de lucru cu obstacole. Perceperea mediului se poate face *off-line* pe baza unor hărţi cunoscute apriori sau *on-line* (în timpul deplasării robotului) cu ajutorul senzorilor. Senzorii cei mai utilizaţi pentru navigaţia roboţilor mobili sunt: senzorii optici (camerele video) şi senzorii de proximitate (senzorii laser, în infraroşu şi ultrasonici). Imaginile mediului, captate cu o cameră video, constituie de departe, cea mai bogată sursă de informaţii cu privire la structura mediului de lucru, comparativ cu senzorii de proximitate. Utilizarea camerei video la navigaţia roboţilor mobili presupune procesarea unei cantităţi mari de informaţie care consumă timp şi poate duce în final la o viteză de navigare scăzută.

Prin utilizarea reţelelor neuronale celulare (CNN) [49],[50],[68], a căror timp de procesare a semnalelor este foarte mic, se poate obţine o creştere a vitezei de deplasare pentru roboţii mobili şi se pot elimina unele dezavantaje mai ales în cazul metodei globale. Din acest motiv, reţele neuronale celulare sunt considerate o soluţie promiţătoare pentru procesarea imaginilor în scopul ghidării roboţilor mobili autonomi [87],[88],[89],[90],[91],[92]. Alegerea acestor reţele se bazează şi pe posibilitatea implementării lor pe un singur cip, în tehnologia VLSI [67].

Pentru planificarea traiectoriei roboţilor mobili cu ajutorul reţelelor neuronale celulare, în literatura de specialitate există o multitudine de soluţii [59],[87],[93],[94],[95],[96] care în general, au la bază prelucrări de imagini şi de asemenea una sau ambele metode cunoscute pentru planificarea traiectoriei: metoda locală şi /sau globală.

În [93] se face o corespondenţă între mediul de lucru al robotului şi o hartă discretizată în celule pătratice. Starea fiecărei celule: ocupată, liberă sau necunoscută este obţinută pe baza informaţiilor furnizate de senzorii laser ai robotului. Actualizarea stării celulelor se face prin metoda probabilistică bayesiană. Modulul de planificare a traiectoriei este realizat sub forma unei reţele neuronale celulare cu două straturi pe care este implementat un algoritm de difuzie (generarea unei unde a cărui front avansează în reţea). Primul strat recepţionează semnalele exterioare pe baza cărora este realizată o hartă a ocupării celulelor, respectiv neuronilor componenţi. De asemenea, neuronul ce corespunde locaţiei ţintei este setat la o valoare distinctivă, iar neuronul ataşat poziţiei robotului este activat şi o

undă se va propaga prin reţea pornind din acest punct până când este atins neuronul ţintă. Nivelele de activare ale celulelor sunt apoi propagate spre al doilea strat al reţelei, care realizează căutarea celei mai scurte traiectorii ce leagă poziţia ţintei de poziţia robotului. Traiectoria planificată este apoi reprezentată ca o secvenţă de celule adiacente între cele două poziţii.

În lucrarea [97] se prezintă un algoritm CNN bazat pe imagini, de urmărire a unui obiect în mişcare cu ajutorul unei camere video montate pe braţul unui robot ce are două grade de libertate. Între două imagini de intrare achiziţionate consecutiv, prin prelucrarea complet paralelă cu reţele neuronale celulare a semnalelor bidimensionale, se asigură efectuarea tuturor procesărilor necesare în vederea funcţionării în timp real a sistemului.

În acest capitol sunt propuşi o serie de algoritmi de planificare pe baza imaginilor a traiectoriei roboţilor mobili, folosind reţele neuronale celulare. Sunt prezentate mai multe modalităţi de planificare, concepute de către autor, toate acestea fiind testate cu ajutorul mediului de dezvoltare **CadetWin** (***C****NN* ***a****pplication* ***d****evelopment* ***e****nvironment and* ***t****oolkit under* ***Win****dows*) [69], şi s-au făcut comparaţii între ele, mai ales în ceea ce priveşte timpul necesar pentru ca algoritmii să furnizeze traiectoria sau traiectoriile dorite.

4.2. Algoritm CNN de planificare a traiectoriei unui robot mobil

În aplicaţia de faţă, se consideră un robot mobil care se găseşte într-un mediu cu obstacole statice (Figura 4.1). Acesta trebuie, ca pe baza imaginilor achiziţionate cu ajutorul camerei video, să se deplaseze spre ţintă pe drumul cel mai scurt ocolind obstacolele.

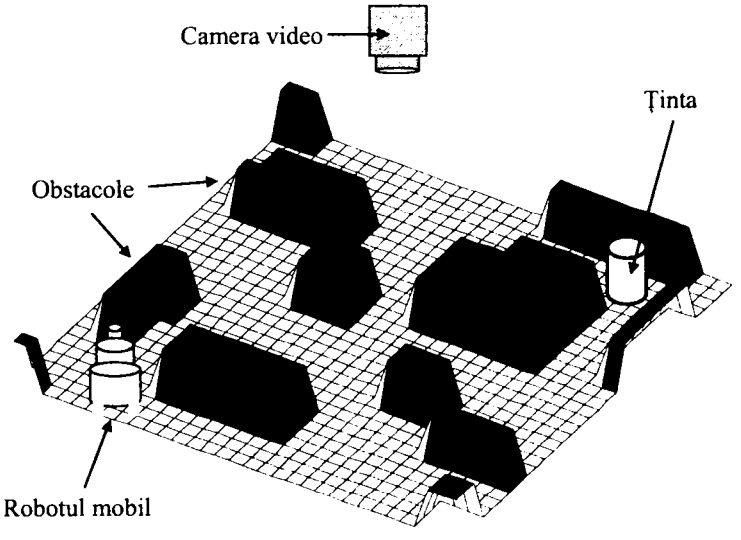


Figura 4.1. Mediul de lucru al robotului.

Pentru a putea fi prelucrate cu reţele neuronale celulare, imaginile mediului cu obstacole sunt divizate în imagini discrete având $M \times N$ pixeli, fiind posibil în acest

fel, reprezentarea imaginii mediului printr-o reţea neuronală standard având $M \times N$ celule.

Imaginile reale cu nivele de gri (*gray-scale*), ale mediului, sunt transferate reţelei neuronale celulare respectiv cipului CNN. În urma unor procesări elementare CNN se obţine imaginea binară a mediului (Figura 4.2), necesară algoritmului. În urma discretizării spaţiale a imaginii, discretizare care corespunde cu rezoluţia CNN, se presupune că fiecare obstacol este reprezentat de cel puţin un pixel negru iar robotul şi ţinta sunt identificaţi fiecare prin câte un pixel negru.

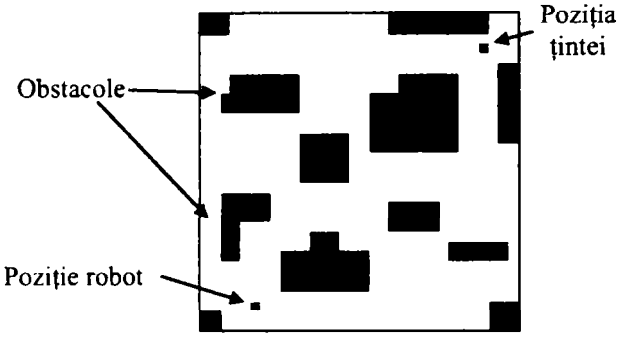


Figura 4.2. Imaginea binară a mediului de lucru.

În imaginile cu niveluri de gri valorile pixelilor sunt în domeniul standard CNN, adică [-1, +1], de la alb către negru, iar pentru imaginile binare valorile pixelilor sunt +1 pentru negru şi −1 pentru alb. În imaginea binară din Figura 4.2, pixelilor cu valori +1 li se asociază poziţiile interzise din mediu, care nu sunt accesibile robotului, iar pixelii cu valoarea −1 semnifică poziţiile libere, accesibile pentru robot.

Organigrama algoritmului de planificare a traiectoriei robotului mobil, pe baza imaginilor, este prezentată în Figura 4.3. Algoritmul detectează şi semnalează chiar de la început situaţiile posibile în care ţinta nu este accesibilă robotului, adică faptul că ţinta este "înconjurată" de obstacole.

4.2.1. Evaluarea distanţelor dintre punctele din spaţiul liber al mediului de lucru şi punctul ţintă

Pentru stabilirea unei traiectorii optime între poziţia de start şi ţintă trebuiesc evaluate poziţiile punctelor (pixelilor) din spaţiul liber al mediului de lucru faţă de punctul ţintă. Pentru aceasta se generează o undă în planul imaginii, având centrul sursei situat chiar în punctul ţintă. Pentru obţinerea acestei imagini se utilizează template-ul EXPLORE [79], definit de relaţiile 4.1.

În domeniul aplicaţiilor CNN, generarea controlată a unei unde este o metodă eficientă, care se poate utiliza la detectarea contururilor sau la clasificarea binară.

$$A = \begin{pmatrix} 0 & a & 0 \\ a & 1 & a \\ 0 & a & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad z = 0 \tag{4.1}$$

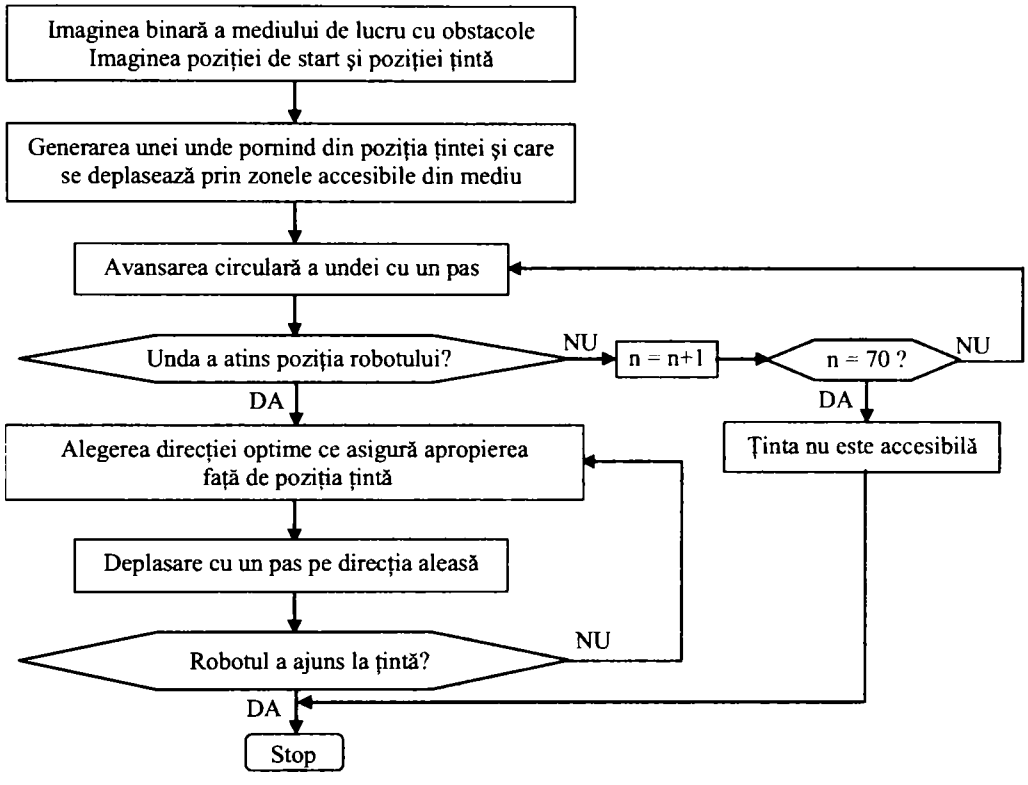


Figura 4.3. Organigrama algoritmului de planificare a traiectoriei robotului mobil.

Template-ul de mai sus este neliniar deoarece parametrul *a* reprezintă o funcţie neliniară şi depinde de diferenţa dintre valoarea tensiunii de la ieşirea celulei C_{ij} şi valoarea tensiunii de la ieşirea celulei C_{kl} situată în vecinătatea sa ($y_{ij} - y_{kl}$). Reprezentarea grafică a funcţiei *a* este arătată în Figura 4.4. Parametrul β are semnificaţia de unitate de măsură a distanţei.

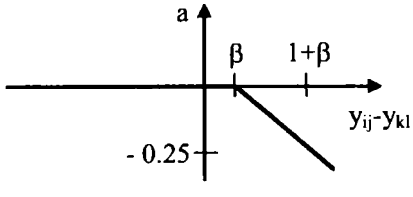


Figura 4.4. Caracteristica de transfer a funcţiei neliniare a.

Prin propagarea ei, unda explorează toate căile accesibile din mediul de lucru, începând din punctul ţintă Figura 4.5. Pe starea iniţială a reţelei $x(t_0)$ se aplică o imagine în care toate elementele de imagine au valoarea +1, cu excepţia pixelului corespunzător punctului ţintă, care are valoarea –1. Această imagine este, de fapt, inversa imaginii prin care se indică poziţia ţintei. Imaginea binară a mediului se va folosi ca imagine mască. Pixelii de la marginea imaginii mediului de lucru, respectiv frontierei reţelei neuronale celulare, vor fi consideraţi poziţii interzise.

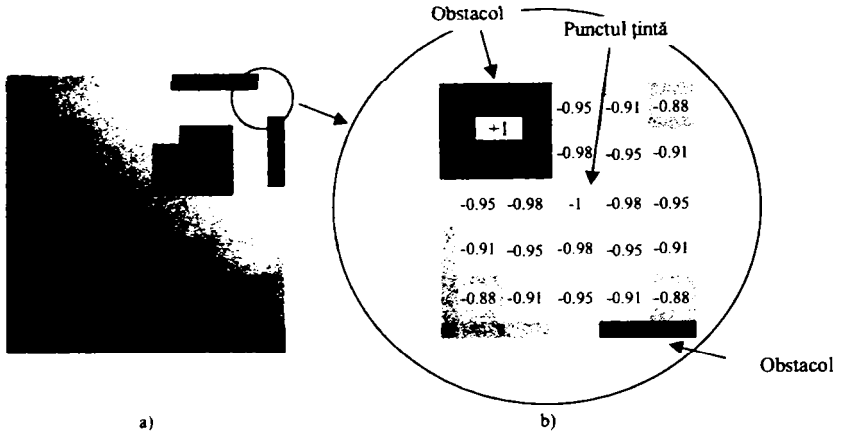


Figura 4.5. Principiul propagării circulare a unei unde din punctul ţintă:
a) imaginea undei; b) valorile pixelilor din jurul poziţiei ţintei.

În urma acestei prelucrări, pixelul corespunzător poziţiei ţintă din imaginea de ieşire a reţelei va rămâne la valoarea iniţială –1, iar elementele de imagine care vor avea valoarea +1 vor constitui poziţii interzise sau inaccesibile robotului. Toţi ceilalţi pixeli vor avea valori proporţionale cu distanţele dintre poziţiile lor şi poziţia ţintei. Astfel, în imaginea rezultată, pornind din centrul sursei undei, valorile pixelilor cresc aproximativ cu câte o unitate de măsură a distanţei β, la creşterea cu o unitate a razei undei cu propagare circulară.

Obţinerea unei unde care să atingă poziţia de start se face prin aplicarea repetată a template-ului EXPLORE. Numărul de aplicări a template-ului va creşte treptat de la valoarea 1 până la o valoare care va determina ca în imaginea de ieşire să se obţină o modificare a pixelului ce indică poziţia de start. Dacă acest număr atinge o valoare maximă N_{max}, dat de relaţia 4.2, şi valoarea luminanţei pixelului ce indică poziţia de start nu se modifică, înseamnă că ţinta nu este accesibilă. Aceasta se poate întâmpla fie din cauza poziţiei obstacolelor, fie din cauză că ţinta este situată la o distanţă mult prea mare faţă de robot.

$$N_{max} = \frac{+1-(-1)}{p} \tag{4.2}$$

Valoarea lui p reprezintă pasul minim de discretizare în nivele de gri, tipic p = 0.03, astfel că N_{max} = 67. După cum se vede şi în organigrama din Figura 4.3, în astfel de situaţii, algoritmul se va opri după ce numărul de aplicări a template-ului EXPLORE va atinge valoarea n = 70.

4.2.2. Alegerea direcţiei optime de deplasare

Alegerea direcţiei optime, care să asigure lungimea minimă pentru traiectoria robotului, precum şi un număr minim de viraje, se realizează prin extragerea valorii unui pixel cu niveluri de gri. Poziţia robotului este reprezentată de o imagine având toţi pixelii de valoare –1 cu excepţia unui singur pixel de valoare +1. Această imagine care indică poziţia curentă a robotului, va fi suprapusă la fiecare pas al algoritmului cu imaginea reprezentând evaluarea distanţelor dintre punctele din mediul de lucru accesibile robotului şi punctul ţintă (Figura 4.5a).

Pentru obţinerea traiectoriei optime, trebuie să fie ales, la fiecare pas, pixelul cu cea mai mică valoare din vecinătatea de rază r = 1 a pixelului ce indică poziţia curentă a robotului. Acest pixel va indica direcţia optimă de deplasare şi va reprezenta apoi poziţia actuală a robotului.

Determinarea acestui pixel s-a făcută în două moduri:

- explorarea tuturor pixelilor vecini şi apoi determinarea sa prin comparaţii succesive;
- prin aplicarea template-ului PATH [98].

4.2.2.1. Alegerea direcţiei optime prin comparaţii succesive

Prin inversarea imaginii, care indică iniţial poziţia punctului de start a robotului, apoi poziţia actualizată în care se va alege o nouă direcţie de deplasare, se obţine o imagine binară cu aceleaşi dimensiuni care va fi folosită în continuare ca imagine mască.

În această etapă de prelucrare sunt necesare: imaginea mască menţionată mai sus şi imaginea cu niveluri de gri care conţine estimarea distanţelor poziţiilor libere din mediu faţă de poziţia ţintei (Figura 4.5a).

Dintre cei opt pixeli vecini care corespund direcţiilor date de punctele cardinale (N, S, E, V, SE, NE, NV, SV), după care se poate efectua deplasarea, se evaluează şi se alege cu o metodă locală celula vecină cu valoarea cea mai mică. Practic sunt comparate valorile pixelilor din jurul pixelului ce indică poziţia robotului în imaginea cu niveluri de gri din Figura 4.5a şi este ales pixelul cu luminanţa cea mai mare. Pentru aceasta, se utilizează procesări elementare AMC asupra imaginilor, având la bază familia de template-uri SHIFT [79], care corespund celor opt direcţii posibile de deplasare (relaţia 4.3).

$$A=\begin{pmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{pmatrix}\quad B=\begin{pmatrix}se & s & sv\\ e & 0 & v\\ ne & n & nv\end{pmatrix}\quad z=0 \tag{4.3}$$

În cazul unuia din template-urile din familia SHIFT doar unul din elementele operatorului B sunt egale cu 1 restul fiind 0, astfel: SHIFTE (e = 1), SHIFTSE (se = 1), SHIFTS (s = 1), SHIFTSV (sv = 1), SHIFTV (v = 1), SHIFTNV (nv = 1), SHIFTN (n = 1), SHIFTNE (ne = 1).

4.2.2.2. Alegerea direcţiei optime prin aplicarea template-ului PATH

Având indicată poziţia robotului printr-un pixel negru, delimitarea vecinătăţii robotului de rază r = 1, poate fi obţinută cu template-ul DILATION [79], dat de relaţia 4.4.

$$A=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\quad B=\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}\quad z=8 \tag{4.4}$$

Dacă pe starea şi pe intrarea reţelei se aplică imaginea ce reprezintă poziţia curentă a robotului, după aplicarea template-ului de mai sus, pe ieşire se obţine imaginea vecinătăţii robotului (Figura 4.6).

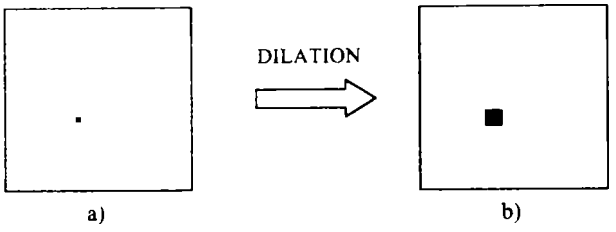


Figura 4.6. Delimitarea vecinătăţii de rază r=1 a robotului; a) imagine reprezentând poziţia robotului, b) imagine reprezentând vecinătatea robotului.

După realizarea unor operaţii logice asupra imaginii vecinătăţii robotului, se va obţine o imagine mască (Figura 4.7a), care suprapusă peste imaginea undei din Figura 4.5a sau zona selectată din Figura 4.7b, va rezulta o imagine care reprezintă unda obţinută prin aplicarea template-ului EXPLORE asupra mediului cu obstacole, doar în zona ce corespunde vecinătăţii robotului, restul pixelilor din imagine având valoarea +1 (Figura 4.7c).

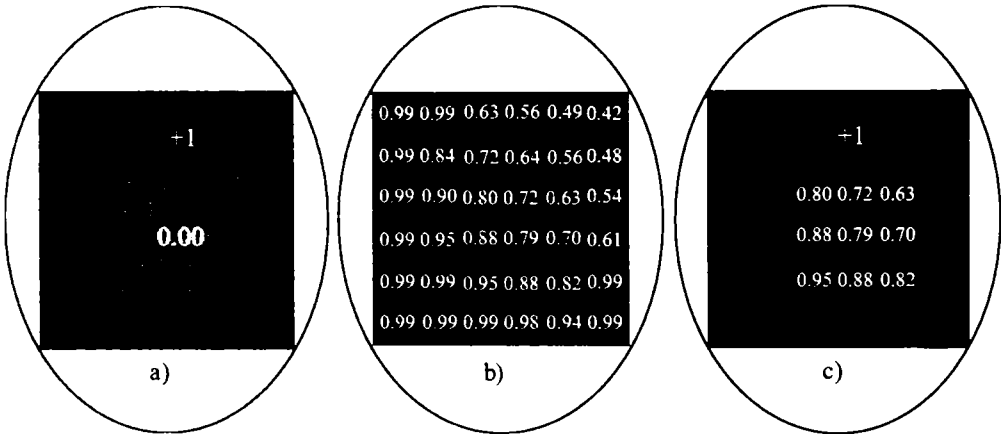


Figura 4.7. Delimitarea zonei corespunzătoare vecinătăţii robotului cu valorile pixelilor obţinute după propagarea undei; a) imagine mască obţinută din imaginea vecinătăţii robotului, b) imagine reprezentând valorile pixelilor în jurul poziţiei actuale după aplicarea template-ului EXPLORE, c) imagine cu valorile pixelilor din vecinătatea de rază r=1 a robotului, după propagarea undei.

Dintre toţi pixelii vecini va fi ales pixelul având valoarea cea mai mică, astfel pixelul ce trebuie ales din imaginea prezentată în Figura 4.7c va fi cel cu valoarea 0,63. Pentru ca algoritmul să determine acest pixel, se poate proiecta un template care aplicat asupra acestei imagini, dată ca exemplu, să rezulte o imagine în care toţi pixelii au valoarea -1 cu excepţia pixelului ce indică poziţia viitoare a robotului, care va avea valoarea +1. Din păcate, aplicarea acestui template nu va indica poziţia viitoare pentru robot în alte situaţii, adică atunci când va fi o altă configuraţie a valorilor pixelilor din vecinătatea robotului. Ar trebui, spre exemplu, ca valoarea

de prag *z* a template-ului să se modifice la fiecare altă configuraţie în care se află robotul.

Din acest motiv, în algoritmul propus se vor efectua procesări asupra imaginilor de forma celei din Figura 4.7c, astfel încât prin aplicarea aceluiaşi template, asupra acestor tipuri de imagini, să se obţină de fiecare dată o imagine ce indică poziţia viitoare ce trebuie ocupată de robot. Astfel, se va memora valoarea minimă din imagine şi apoi se va crea o imagine având toţi pixelii cu această valoare (Figura 4.8a). Dacă această imagine va fi scăzută din imaginea de tipul celei din Figura 4.7b, se obţine o imagine care are valorile pixelilor din vecinătatea robotului prezentate în Figura 4.8b. Imaginea obţinută în final (Figura 4.8c) va constitui imaginea asupra căreia se va opera cu un template pentru determinarea poziţiei viitoare a robotului.

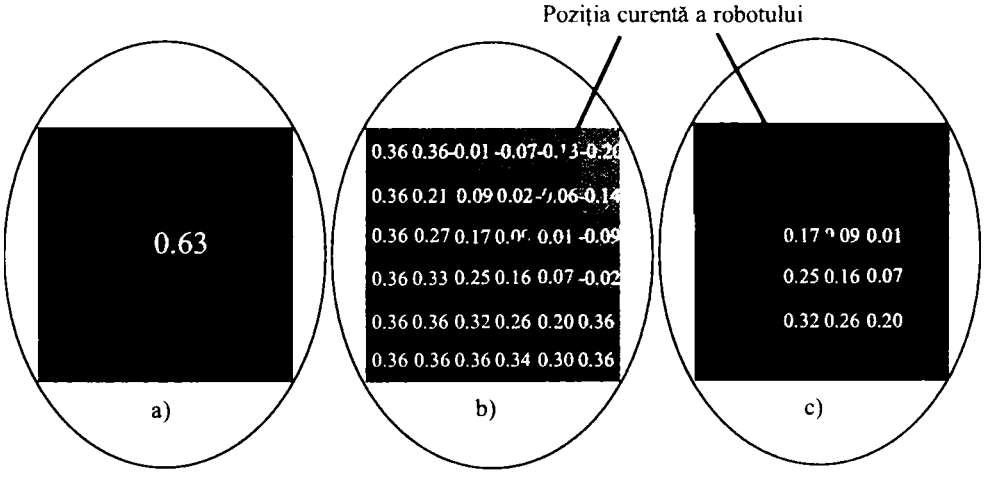


Figura 4.8. Exemplu de actualizare a pixelilor în funcţie de poziţia curentă a robotului; a) imagine având valoarea tuturor pixelilor egală cu valoarea minimă a undei în vecinătatea poziţiei robotului (0.63), b) valoarea actualizată a pixelilor în jurul poziţiei curente a robotului, c) valoarea actualizată a pixelilor în vecinătatea poziţiei robotului.

Imaginea astfel rezultată, are aproximativ aceleaşi valori ale pixelilor în jurul poziţiei curente a robotului, indiferent de poziţia în care este situat robotul în drumul lui către ţintă. Având un template potrivit ce se va aplica, prin intermediul unei imagini mască, asupra imaginii de tipul celei din Figura 4.8c se va obţine, de fiecare dată, o imagine ce va indica, printr-un pixel negru, poziţia viitoare ce trebuie ocupată de către robot.

Template-ul căutat poate fi de forma dată de relaţia 4.5.

$$A=\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}\quad B=\begin{pmatrix}b_{11}&b_{12}&b_{13}\\b_{21}&b_{22}&b_{23}\\b_{31}&b_{32}&b_{33}\end{pmatrix}\quad z=c \tag{4.5}$$

Prin aplicarea acestui template, denumit PATH, asupra imaginii din Figura 4.8c, valoarea viitoare a fiecărui pixel va depinde de valoarea curentă a pixelului precum şi de valoarea pixelilor din vecinătatea sa. Aplicarea acestui template se va face de fiecare dată cu o imagine mască care în afara vecinătăţii robotului va avea

toţi pixelii la valoarea +1. Imaginea mască se obţine prin inversarea imaginii reprezentând vecinătatea robotul în poziţia curentă (Figura 4.7a).

Presupunând că template-ul căutat va fi simetric, între elementele acestuia există relaţiile 4.6.

$$b_{11} = b_{12} = b_{13} = b_{23} = b_{33} = b_{32} = b_{31} = b_{21} = b_1 \quad b_{22} = b_2, \tag{4.6}$$

Pentru reducerea volumului de calcul se va considera:

$$b_{11} = b_{13} = b_{33} = b_{31} = 0 \tag{4.7}$$

Ţinând cont de relaţiile 4.6 şi 4.7, operatorul **B** va fi de forma:

$$B = \begin{pmatrix} 0 & b_1 & 0 \\ b_1 & b_2 & b_1 \\ 0 & b_1 & 0 \end{pmatrix} \tag{4.8}$$

Pe intrarea (*INPUT*) a reţelei neuronale celulare se aplică imaginea mediului cu obstacole cu valorile actualizate ale undei (Figura 4.9a), iar pe starea iniţială (*STATE*) a reţelei $x(t_0)$ o imagine având toţi pixelii la valoarea 0 (Figura 4.9b). Ecuaţia de stare a reţelei corespunzătoare template-ului PATH are forma:

$$\dot{x}_{ij}(t) = \sum_{C_{kl} \in N_1} b_{ij,kl} u_{kl} + c \cdot \tag{4.9}$$

Din relaţiile 4.8 şi 4.9 se obţine ecuaţia de stare particularizată:

$$\dot{x}_{ij}(t) = b_1 \cdot u_{i-1,j} + b_1 \cdot u_{i+1,j} + b_1 \cdot u_{i,j-1} + b_1 \cdot u_{i,j+1} + b_2 \cdot u_{ij} + c \tag{4.10}$$

Imaginea binară, reprezentând vecinătatea de rază r = 1 a poziţiei curente a robotului, se va utiliza ca imagine mască (*MASK*) pentru această etapă de prelucrare (Figura 4.9c). În acest fel, doar valorile pixelilor din jurul poziţiei curente a robotului se vor modifica. Imaginea de ieşire trebuie să conţină doar un pixel negru care va indica, de fapt, poziţia viitoare ce trebuie ocupată de robot (Figura 4.9d).

Aplicând asupra tuturor pixelilor din vecinătatea robotului, relaţia 4.10, elementele template-ului PATH trebuie să satisfacă relaţiile 4.11.

$$\begin{aligned}
&(-0.06 - 0.09 + 0.07 + 0.09) \cdot b_1 + 0.01 \cdot b_2 + c > 0 \\
&(0.09 + 0.07 + 0.26 + 0.25) \cdot b_1 + 0.16 \cdot b_2 + c < 0 \\
&(0.02 + 0.01 + 0.16 + 0.17) \cdot b_1 + 0.09 \cdot b_2 + c < 0 \\
&(0.01 - 0.02 + 0.20 + 0.16) \cdot b_1 + 0.07 \cdot b_2 + c < 0 \\
&(0.07 + 0.36 + 0.30 + 0.26) \cdot b_1 + 0.20 \cdot b_2 + c < 0 \\
&(0.16 + 0.20 + 0.34 + 0.32) \cdot b_1 + 0.26 \cdot b_2 + c < 0 \\
&(0.25 + 0.26 + 0.36 + 0.36) \cdot b_1 + 0.32 \cdot b_2 + c < 0 \\
&(0.17 + 0.16 + 0.32 + 0.33) \cdot b_1 + 0.25 \cdot b_2 + c < 0 \\
&(0.09 + 0.09 + 0.25 + 0.27) \cdot b_1 + 0.17 \cdot b_2 + c < 0
\end{aligned} \tag{4.11}$$

Dacă aceste inegalităţi sunt îndeplinite, atunci integratorul ce intră în componenţa fiecărei celule va determina la ieşirea celulei nivelul $y_{ij}(\infty) = -1$, pentru toţi pixelii vecini şi $y_{ij}(\infty) = +1$, pentru pixelul din poziţia viitoare ce trebuie ocupată de către robot. Desigur, acest pixel are de fiecare dată valoarea cea mai scăzută, în imaginea undei, dintre toţi pixelii din vecinătatea pixelului ce reprezintă poziţia curentă a robotului.

După rezolvarea iterativă a primelor două inegalităţi din relaţiile 4.11 ce corespund valorilor pixelilor din poziţia curentă şi viitoare şi ajustarea parametrilor b_1, b_2 şi c astfel încât să fie îndeplinite şi restul inegalităţilor, se obţine pentru template-ul PATH forma:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & -0.1 & 0 \\ -0.1 & -8 & -0.1 \\ 0 & -0.1 & 0 \end{pmatrix} \quad z = 0.3. \tag{4.12}$$

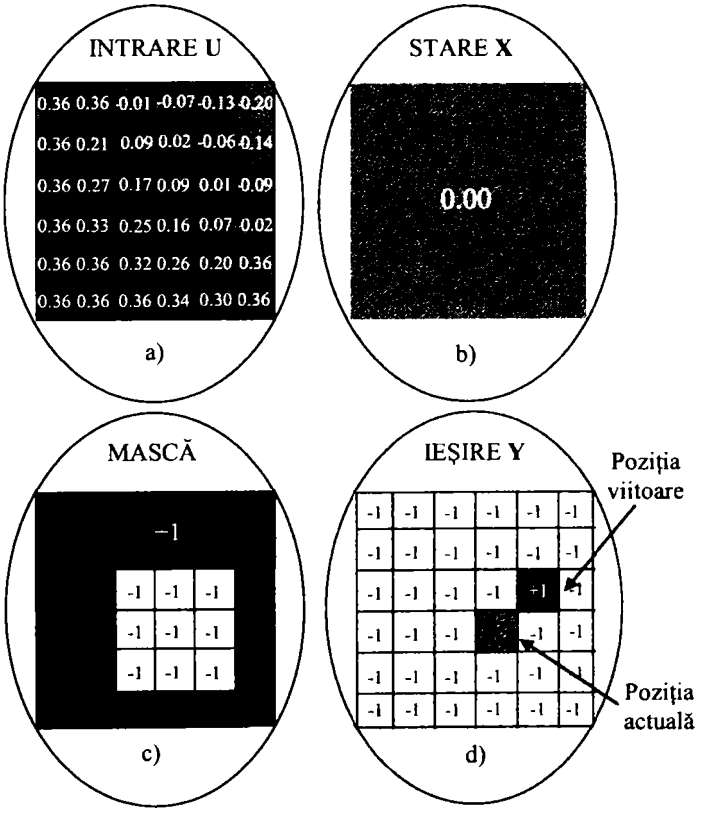


Figura 4.9. Modul de determinare a poziţiei viitoare pentru robotul mobil; a) imaginea mediului cu obstacole peste care e suprapusă unda actualizată, b) imagine având toţi pixelii de valoare 0, c)imaginea mască; d) imagine în care toţi pixelii au valoarea –1 cu excepţia pixelului ce indică poziţia viitoare ce urmează a fi ocupată de către robot.

Trebuie remarcat faptul că sunt şi unele limitări în ceea ce priveşte deplasarea robotului mobil pe baza acestei metodei de alegere a direcţiei optime de deplasare. Una din aceste limitări, observată experimental, este aceea că dacă robotul trebuie să se deplaseze pe un "coridor" cu lăţimea de un singur pixel situat în poziţie orizontală sau verticală, în cadrul imaginii procesate, acesta va efectua câţiva paşi apoi poate să se blocheze, deoarece nu va putea detecta poziţia viitoare pe baza metodei prezentate anterior. Nu sunt probleme de acest gen atunci când robotul se află la o distanţă relativ mare de ţintă, aceste probleme apărând atunci când "coridoarele" sunt situate în apropierea ţintei.

În Figura 4.10 sunt exemplificate două posibile situaţii în care robotul mobil se poate bloca şi nu va continua drumul către ţintă.

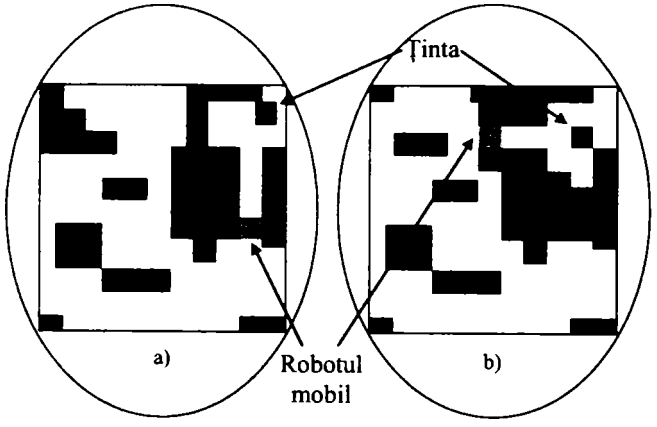


Figura 4.10. Situaţii posibile de blocaj, pentru robot, în apropierea ţintei;
a) în cazul unui coridor vertical, b) în cazul unui coridor orizontal.

Explicaţia acestui fenomen este dată de faptul că valorile obstacolelor au iniţial valoarea +1, şi deoarece pe parcursul algoritmului trebuie actualizată imaginea cu valorile undei, imagine care conţine şi obstacolele, se va scădea din aceasta valoarea minimă din jurul poziţiei actuale a robotului. În apropierea ţintei valorile pixelilor sunt negative şi practic prin scăderea unei valori negative din pixelii de valoare +1 aceştia vor rămâne tot la aceeaşi valoare (nu pot depăşi valoarea +1) pentru că aceasta este valoarea maximă în domeniul standard CNN. Dacă însă aceşti pixeli încadrează pe aceeaşi linie sau coloană pixelul ce ar trebui să fie poziţia viitoare care trebuie ocupată de robot atunci aceştia vor influenţa puternic valoarea viitoare a acestui pixel atunci când se aplică template-ul PATH asupra pixelilor din jurul poziţiei curente a robotului, şi mai ales asupra acestuia. Influenţa asupra valorii viitoare a unui pixel se manifestă din partea pixelilor vecini de pe aceeaşi linie sau coloană datorită structurii alese pentru template şi din cauză că valorile elementelor template-ului au valori negative. Aşadar, în apropierea ţintei, dacă obstacolele sunt apropiate, pot exista situaţii când prin procesarea imaginilor pe baza metodei prezentate, valoarea pixelului care ar trebui să indice poziţia viitoare nu va trece la valoarea +1. Practic, nici unul din pixelii din vecinătatea poziţiei curente nu vor trece la această valoare şi în acest fel algoritmul se va opri.

Soluţia pentru ca robotul să iasă din impas ar fi: fie proiectarea unui template care să fie aplicat doar în astfel de situaţii, fie să se aleagă pentru aceste cazuri o altă metodă de deplasare (spre exemplu, *wall-folowing*), apoi după ieşirea robotului din "coridor" poate fi aplicată din nou metoda menţionată mai sus.

4.2.3. Deplasarea robotului spre ţintă

După determinarea pixelului care indică poziţia viitoare, ce trebuie ocupată de către robot, acest pixel va deveni poziţie curentă. În această poziţie poate fi aleasă o nouă direcţie de deplasare, sau se poate ca robotul să se deplaseze în continuare pe direcţia aleasă anterior atâta timp cât se apropie de ţintă. În primul caz lungimea traiectoriei măsurată în număr de pixeli va fi minimă iar în cazul al doilea numărul de viraje va fi minim.

În continuare sunt prezentate cele două modalităţi de obţinere a traiectoriei pentru robot şi anume:

- deplasarea robotului pe direcţia pixelului cu valoare minimă atâta timp cât se asigură apropierea de ţintă;
- deplasarea pixel cu pixel a robotului alegând de fiecare dată pixelul cu valoare minimă.

4.2.3.1. Deplasarea continuă a robotului pe direcţia pixelului cu valoare minimă atâta timp cât se asigură apropierea de ţintă

În acest caz, direcţia de deplasarea rămâne neschimbată atâta timp cât această direcţie asigură apropierea robotului de poziţia ţintei. Aceasta înseamnă că, de pe poziţia curentă se va păşi pe următoarea poziţie de pe aceeaşi direcţie numai dacă valoarea pixelului corespunzător pentru poziţia respectivă este mai mică decât valoarea pixelului din poziţia curentă. Spre exemplu, în Figura 4.11, din punctul A robotul se va deplasa succesiv în punctele B, C, D, E, şi F unde se va opri deoarece pe aceeaşi direcţie urmează un pixel ce are valoarea mai mare decât cel al poziţiei curente. În punctul F se va alege o nouă direcţie de deplasare şi anume direcţia dată de punctul G respectiv H.

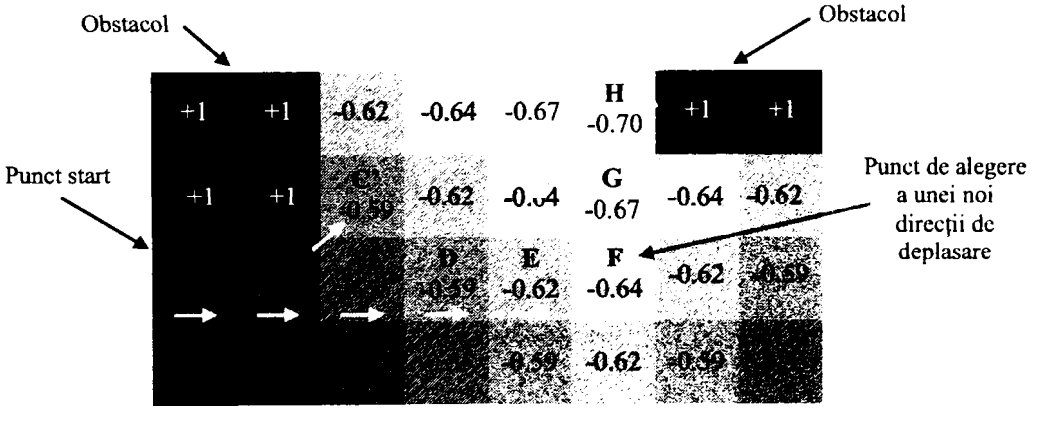


Figura 4.11. Alegerea direcţiei optime prin menţinerea direcţiei atâta timp cât se asigură apropierea de ţintă.

După alegerea direcţiei optime, într-un punct de viraj, obţinerea imaginii traiectoriei pe direcţia respectivă care să indice în acel moment drumul spre ţintă, se obţine prin utilizarea familiei de template-uri SELECT, extinsă pe una din cele opt direcţii [79], definită de relaţiile 4.13:

$$A=\begin{pmatrix} se & s & sv \\ e & 4 & v \\ ne & n & nv \end{pmatrix} \quad B=\begin{pmatrix} se & s & sv \\ e & 0 & v \\ ne & n & nv \end{pmatrix} \quad z=0 \tag{4.13}$$

În cazul unuia din template-urile din familia SELECT, elementele operatorilor A şi B din relaţia 4.13 vor avea valorile astfel:

SELECTE: $A(e=1, se=s=sv=v=nv=n=ne=0);$
$B(e=b, se=s=sv=v=nv=n=ne=0);$
SELECTSE: $A(se=1, e=s=sv=v=nv=n=ne=0);$
$B(se=b, e=s=sv=v=nv=n=ne=0);$
SELECTS: $A(s=1, e=se=sv=v=nv=n=ne=0);$
$B(s=b, e=se=sv=v=nv=n=ne=0);$
SELECTSV: $A(sv=1, e=se=s=v=nv=n=ne=0);$
$B(sv=b, e=se=s=v=nv=n=ne=0);$
SELECTV: $A(v=1, e=se=s=sv=nv=n=ne=0);$
$B(v=b, e=se=s=sv=nv=n=ne=0);$
SELECTNV: $A(nv=1, e=se=s=sv=v=n=ne=0);$
$B(nv=b, e=se=s=sv=v=n=ne=0);$
SELECTN: $A(n=1, e=se=s=sv=v=nv=ne=0);$
$B(n=b, e=se=s=sv=v=nv=ne=0);$
SELECTNE: $A(ne=1, e=se=s=sv=v=nv=n=0);$
$B(ne=b, e=se=s=sv=v=nv=n=0).$

Funcţia neliniară *b* este prezentată în Figura 4.12, în care β are semnificaţia de unitate de măsură a distanţei în imagine.

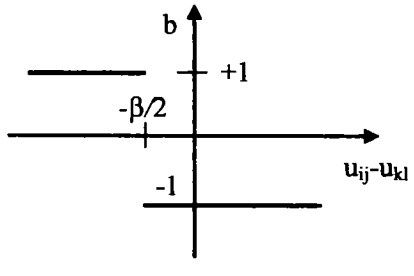


Figura 4.12. Caracteristica de transfer a funcţiei b.

Pentru obţinerea traiectoriei robotului între două puncte de viraj, se aplică pe intrarea reţelei neuronale celulare imaginea cu estimarea distanţelor punctelor din spaţiul liber al mediului faţă de poziţia ţintei iar pe starea reţelei se aplică o imagine având un sigur pixel de valoare +1 ce indică poziţia de start pentru porţiunea curentă din traiectoria planificată, restul pixelilor având valoarea −1. În

acelaşi timp se aplică unul din template-urile din familia SELECT ce corespunde direcţiei ce a fost aleasă în prealabil.

Traiectoria totală între poziţia iniţială de start şi poziţia ţintei se obţine prin însumarea traiectoriilor dintre două puncte de viraj. Aşadar, după "selectarea" traiectoriei pe direcţia respectivă aceasta va fi însumată cu traiectoriile precedente şi tot aşa până la ţintă. Fiecare punct de viraj, în care trebuie aleasă o nouă direcţie, trebuie să fie reprezentat printr-o imagine ce are activ doar acel pixel, astfel că, asupra imaginii ce conţine ultima porţiune de traiectorie trebuie aplicat de fiecare dată unul din template-urile din familia DEL [79], extinsă pe cele opt direcţii de deplasare. Template-urile DEL sunt date de relaţiile 4.14:

$$A=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\quad B=\begin{pmatrix}se & s & sv\\ e & 0 & v\\ ne & n & nv\end{pmatrix}\quad z=0 \tag{4.14}$$

În cazul unuia din template-urile din familia DEL, elementele operatorului ***B*** din relaţia 4.14 vor avea valorile date de relaţiile 4.15, astfel:

$$\begin{aligned}
&\text{DELE } e=1, v=-1, se=s=sv=nv=n=ne=0;\\
&\text{DELSE } se=1, nv=-1, e=s=sv=v=n=ne=0;\\
&\text{DELS } s=1, n=-1, e=se=sv=v=nv=ne=0;\\
&\text{DELSV } sv=1, ne=-1, e=se=s=v=nv=n=0;\\
&\text{DELV } v=1, e=-1, se=s=sv=nv=n=ne=0;\\
&\text{DELNV } nv=1, se=-1, e=s=sv=v=n=ne=0;\\
&\text{DELN } n=1, s=-1, e=se=sv=v=nv=ne=0;\\
&\text{DELNE } ne=1, sv=-1, e=se=s=v=nv=n=0.
\end{aligned} \tag{4.15}$$

Prin aplicarea acestor operatori asupra unei imagini binare ce conţine porţiunea de traiectorie reprezentată de pixeli de valoare +1, se va realiza ştergerea după direcţia specificată, a unui pixel. Ştergerea mai multor pixeli pe aceeaşi direcţie se realizează prin aplicarea repetată a operatorului respectiv.

Aplicând algoritmul CNN de planificare a traiectoriei, prin această metodă de deplasare a robotului, în cazul imaginii binare din Figura 4.2 se obţine în final traiectoria prezentată în Figura 4.13c.

După cum s-a menţionat şi anterior, avantajul acestei metode constă în faptul că numărul de viraje realizat de către robot este minimizat.

4.2.3.2. Deplasarea robotului pixel cu pixel cu alegerea direcţiei optime la fiecare pas

Dezavantajul metodei prezentate mai sus este dat de faptul că lungimea traiectoriei măsurată în număr de pixeli, nu este cea minimă. În exemplul prezentat în Figura 4.11 între punctele de schimbare a direcţiei A şi F există punctul B care dacă ar fi punct de alegere a unei noi direcţii, deplasarea ar fi spre punctul C' şi nu spre C. Dacă robotul s-ar deplasa pixel cu pixel alegând după fiecare pas direcţia optimă atunci lungimea traiectoriei măsurată în număr de pixeli ar fi minimă în schimb timpul total necesar algoritmului de planificare a traiectoriei este mai mare.

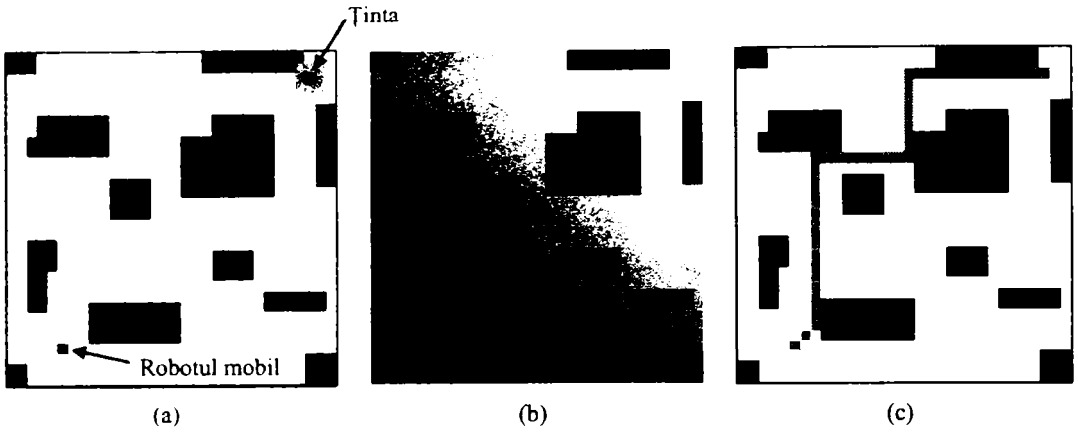


Figura 4.13. Obţinerea traiectoriei robotului mobil la deplasarea acestuia într-un mediu cu obstacole statice prin metoda deplasării pe direcţia pixelului cu valoare minimă atâta timp cât se asigură apropierea de ţintă: a) imaginea iniţială binară a mediului cu obstacole; b) imaginea undei ce se propagă prin mediul de lucru din punctul în care este situată ţinta; c) imaginea traiectoriei rezultate.

În Figura 4.14 se prezintă forma traiectoriei în cazul deplasării pixel cu pixel a robotului mobil cu determinarea direcţiei optime la fiecare pas (pixel).

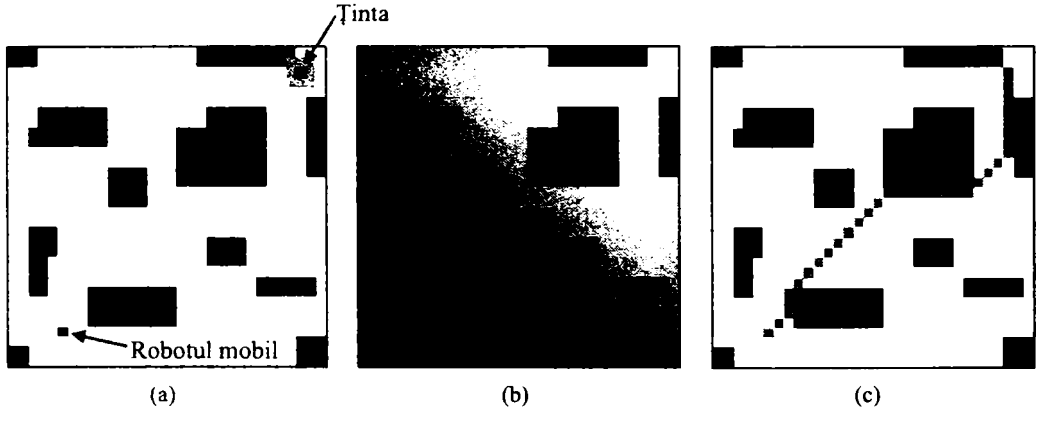


Figura 4.14. Determinarea traiectoriei robotului mobil la deplasarea pixel cu pixel şi alegerea direcţiei optime la fiecare pas: a) imaginea iniţială binară a mediului cu obstacole; b) imaginea undei ce se propagă prin mediul de lucru din punctul în care este situată ţinta; c) imaginea traiectoriei obţinute pentru deplasarea robotului mobil.

4.2.4. Atingerea ţintei

Poziţia viitoare pe care o va ocupa robotul mobil va deveni poziţie curentă şi tot aşa până când va fi atinsă ţinta. Traiectoria planificată, între poziţia de start şi poziţia ţintă, se compune din porţiunile de traiectorii dintre două puncte consecutive în care este schimbată direcţia de deplasare. Astfel, la traiectoria parcursă până într-un anumit punct (poziţie curentă) se va adăuga de fiecare dată poziţia viitoare (la

deplasarea pixel cu pixel) sau porţiuni din traiectorie (la deplasarea pe o direcţie atata timp cât se asigură apropierea de ţintă). Această operaţie se poate realiza în domeniul CNN prin aplicarea operaţiei logice AND între două imagini.

Algoritmul de planificare a traiectoriei se va încheia atunci când robotul reprezentat printr-un pixel se va suprapune cu pixelul corespunzător poziţiei în care este situată ţinta. După fiecare deplasare a robotului fie că aceasta se face continuu pe o direcţie descrescătoare a valorilor pixelilor sau se face pixel cu pixel se va verifica dacă robotul a atins ţinta.

Algoritmul de planificare a traiectoriei unui robot mobil într-un mediu de lucru cu obstacole prezentat mai sus a fost testat utilizând platforma **VisMouse - SimCNN** (***Vis****ual* ***Mouse*** *platform and multi-layer* ***CNN Sim****ulator* [73],[75]). Rezultatele experimentale au fost obţinute prin simulare cu ajutorul unor programe scrise în limbaj de asamblare **AMC** (*Extended* ***A****nalogic* ***M****acro* ***C****ode and interpreter* [77]), (vezi Anexele A1 şi A2).

4.3. Algoritm CNN de planificare simultană a traiectoriilor pentru doi roboţi mobili

Se prezintă în continuare un algoritm de planificare simultană a traiectoriilor pentru doi roboţi mobili, care trebuie să ajungă la o ţintă comună, într-un mediu cu obstacole, pe baza imaginilor captate cu o cameră video. Pentru prelucrarea imaginilor şi planificarea traiectoriei se utilizează tot reţele neuronale celulare [49],[50],[68], invariante în spaţiu, cu operatori de dimensiune 3×3. În acest fel metoda poate fi implementată pe un cip CNN, care este capabil de o procesare în timp real a imaginilor. Algoritmul propus furnizează roboţilor traiectoria optimă din punct de vedere al lungimii şi a numărului de viraje, între poziţiile de start ale roboţilor şi poziţia ţintei. Pe baza acestor traiectorii, fiecare robot ocoleşte obstacolele întâlnite în cale precum şi pe celălalt robot. Algoritmul este conceput astfel încât să semnaleze încă de la început situaţiile posibile în care ţinta nu este accesibilă din cauza obstacolelor.

În aplicaţia de faţă se consideră doi roboţi mobili care se găsesc într-un mediu plan cu obstacole statice (Figura 4.15), şi care trebuie să ajungă la aceeaşi ţintă ocolind obstacolele şi de asemenea unul pe celălalt. În acest fel, fiecare robot va deveni un obstacol dinamic pentru celălalt. Supravegherea este realizată cu o cameră video, iar imaginile captate de aceasta sunt prelucrate conform algoritmului global din Figura 4.16

Deplasarea roboţilor se va face alternativ spre ţintă, pe baza traiectoriei furnizate de algoritm, care ţine cont în cazul fiecărui robot despre poziţia celuilalt şi totodată de poziţia obstacolelor din mediu. În acest fel se vor obţine traiectorii fără coliziune pentru ambii roboţi. Deplasarea unui robot, în cadrul unui pas al algoritmului, se va realiza atâta timp cât acesta se apropie de ţintă fără a schimba direcţia de mers, urmând apoi deplasarea în acelaşi mod a celuilalt robot. Înainte de începerea deplasării, fiecare robot va schimba mai întâi direcţia de deplasare avută anterior. Doar în cazul în care unul din roboţi ajunge în vecinătatea ţintei algoritmul va furniza rând pe rând traiectoriile necesare doar pentru celălalt robot pentru ca şi acesta să ajungă în vecinătatea ţintei.

Etapele principale ale algoritmului sunt prezentate în Figura 4.16.

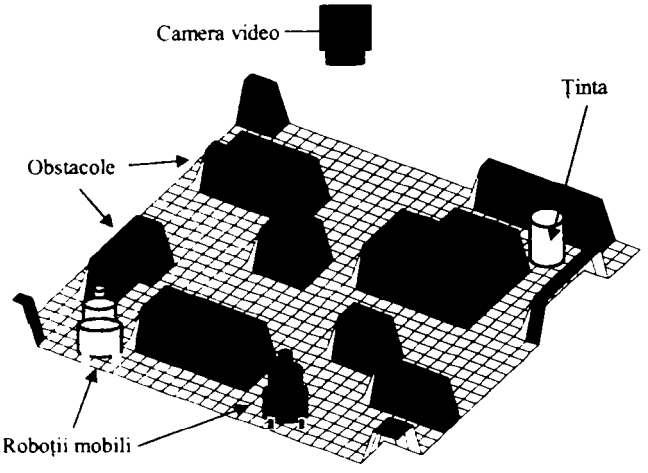


Figura 4.15. Mediul de lucru cu obstacole al celor doi roboţi mobili.

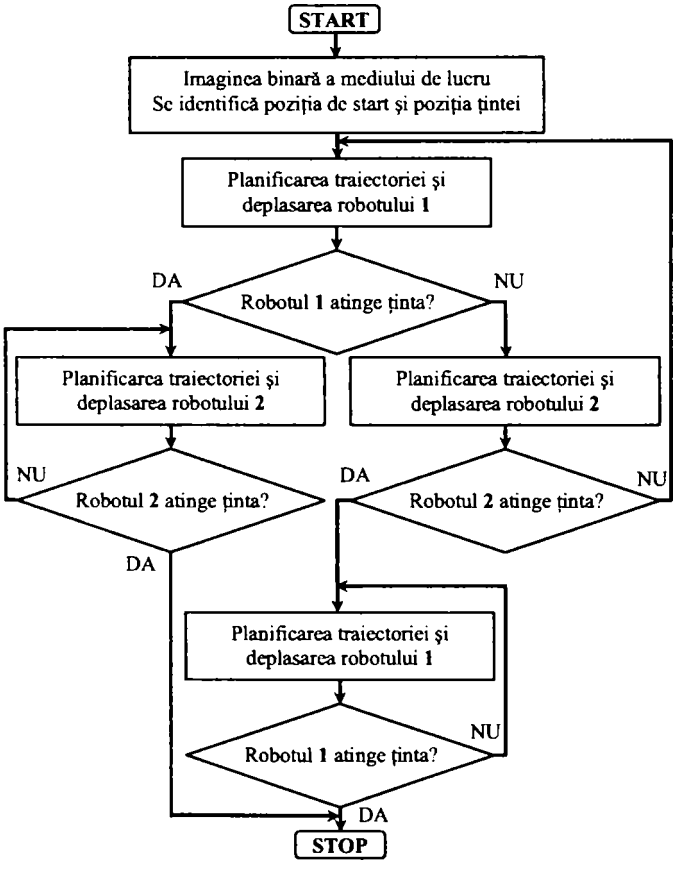


Figura 4.16. Organigrama algoritmului global de planificare a traiectoriilor pentru doi roboţi mobili, pentru deplasarea lor spre o ţintă comună într-un mediu de lucru cu obstacole.

Înainte de a fi procesate, imaginile mediului sunt discretizate într-un număr de pixeli care corespunde cu rezoluţia CNN. În imaginile cu niveluri de gri valorile pixelilor sunt în domeniul standard CNN, adică [-1, +1], de la alb către negru, iar pentru imaginile binare, valorile pixelilor sunt +1 pentru negru şi −1 pentru alb. Imaginea binară a mediului de lucru cu obstacole se obţine în urma unor procesări elementare CNN asupra imaginii reale, cu niveluri de gri, achiziţionate de cameră. Pentru obţinerea acestei imagini s-a utilizat template-ul TRESHOLD prezentat în relaţia următoare:

$$A=\begin{pmatrix}0&0&0\\0&2&0\\0&0&0\end{pmatrix}\quad B=\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}\quad z=-z^{*}, \tag{4.16}$$

unde z^{*} reprezintă pragul de binarizare, $-1<z^{*}<+1$.

Fiecare obstacol din mediu este reprezentat de cel puţin un pixel negru iar roboţii sunt identificaţi fiecare prin câte un pixel negru. Astfel, în imaginea binară rezultată în urma procesării, pixelilor cu valori +1 li se asociază poziţiile interzise, care trebuie evitate de către roboţi, iar pixelii cu valoarea −1 semnifică poziţii libere, accesibile roboţilor.

După identificarea poziţiilor de start ale roboţilor în imaginea binară, se va trece la pasul următor care constă în planificarea traiectoriei şi deplasarea cu un pas a robotului **1** spre ţintă. Se verifică apoi dacă acesta a ajuns la ţintă şi dacă da, algoritmul va continua doar pentru robotul **2**. Dacă nu, pasul următor va consta în planificarea traiectoriei şi deplasarea cu un pas a robotului **2**. Se verifică dacă acesta din urmă a ajuns la ţintă, apoi se va reveni la robotul **1** şi tot aşa până când ambii roboţi vor atinge ţinta.

În situaţia în care unul din roboţi nu are acces la ţintă, algoritmul poate furniza pas cu pas traiectoriile pentru celălalt robot, iar dacă nici unul din roboţi nu are acces spre ţintă, acest fapt va fi semnalat chiar înainte de începerea planificării traiectoriei pentru robotul **1**.

4.3.1. Alegerea direcţiei optime de deplasare pentru roboţi

Alegerea direcţiei optime de deplasare a roboţilor se face prin generarea unei unde în planul imaginii [87], având centrul sursei situat chiar în punctul ţintă cu ajutorul template-ului EXPLORE [79] prezentat la subcapitolul **4.2.1**. Pe baza imaginii obţinute, respectiv a valorilor pixelilor, pot fi evaluate distanţele dintre poziţiile accesibile din mediul de lucru şi punctul ţintă. Elementele de imagine care reprezintă obstacolele rămân la valoarea +1 iar pixelul care indică poziţia ţintei la valoarea -1. Toţi ceilalţi pixeli vor avea valori proporţionale cu distanţa dintre poziţia lor şi poziţia ţintei (Figura 4.17). Dacă există cel puţin o posibilitate de a se ajunge de la poziţiile de start ale celor doi roboţi la poziţia ţintei, atunci în imaginile de ieşire valoarea pixelului din poziţiile corespunzătoare acestor poziţii (poziţiile de start) se vor modifica de la +1 la o valoare care este proporţională cu distanţa dintre aceste poziţii şi poziţia ţintei.

Acest principiu de estimare a distanţelor faţă de ţintă, se va aplica de două ori la începutul algoritmului, o dată pentru fiecare robot. Dacă valoarea unuia din pixelii ce reprezintă poziţiile de start, rămâne la valoarea +1, înseamnă că robotul ce se află în această poziţie nu are acces spre ţintă şi va sta pe loc, în timp ce robotul celălalt se va deplasa singur spre ţintă. În situaţia cea mai defavorabilă,

când după etapa de procesare pentru evaluarea distanţelor, ambii pixeli ce reprezintă poziţiile de start rămân la valoarea +1, algoritmul va indica de la început că nici unul dintre roboţi nu are acces la ţintă.

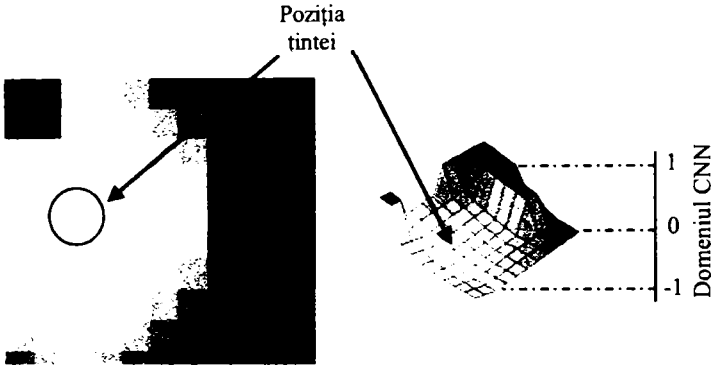


Figura 4.17. Principiul determinării distanţelor dintre poziţiile libere şi poziţia ţintei, prin propagarea unei unde din punctul ţintă. Valorile pixelilor cresc cvasiproporţional cu distanţa faţă de punctul de origine al sursei.

Alegerea direcţiei optime, care să asigure o lungime minimă pentru traiectoria roboţilor, precum şi un număr minim de viraje are la bază posibilitatea extragerii valorii unui pixel cu niveluri de gri prin procesări CNN. Pentru extragerea valorilor pixelilor corespunzători direcţiilor posibile de deplasare (N, S, E, V, SE, NE, NV, SV), se utilizează procesări elementare cu instrucţiuni AMC asupra imaginilor obţinute anterior, având la bază familia de template-uri SHIFT [69], care corespund celor opt direcţii de deplasare. Acest template precum şi exemplificarea modului de alegere a direcţiei de deplasare prin comparaţii succesive a valorilor pixelilor din jurul pixelului ce indică poziţia robotului a fost prezentat în acest capitol la subcapitolul **4.2.2.1**.

Deplasarea pe direcţia aleasă se va face continuu atâta timp cât această direcţie asigură apropierea robotului de ţintă. Asta înseamnă că, de pe poziţia curentă se va păşi pe următoarea poziţie de pe aceeaşi direcţie numai dacă valoarea pixelului corespunzător pentru poziţia respectivă este mai mică decât valoarea pixelului din poziţia curentă. Practic, înainte ca roboţii să se deplaseze pe direcţia aleasă, conform celor arătate mai sus, se vor opri şi vor schimba direcţia de mers avută anterior.

4.3.2. Delimitarea vecinătăţii roboţilor

În decursul deplasării unuia dintre roboţi printre obstacole, cu scopul de a ajunge la punctul ţintă, acesta va trebui să-l ocolească şi pe celălalt robot şi nu va putea ajunge într-o anumită vecinătate a acestuia, cu alte cuvinte va păstra o distanţă faţă de el. În situaţia în care ambii roboţi se întâlnesc şi ar urma să continue deplasarea pe acelaşi drum, primul care ajunge la punctul de întâlnire o va lua înainte iar celălalt se va deplasa în urma sa. Reprezentarea robotului în cadrul algoritmului se face pe un pixel, iar vecinătatea lui în care nu poate să ajungă celălalt robot poate fi în acest caz de rază r =1 (vecinătate 3×3) cum este prezentată în Figura 4.18 sau de rază şi vecinătate mai mare în cazul în care pentru o aplicaţie particulară se cere acest lucru.

Pentru obţinerea unei imagini ce reprezintă vecinătatea robotului s-a folosit template-ul DILATION [79], prezentat în relaţiile 4.4. Prin aplicarea acestuia asupra imaginii ce arată, printr-un pixel negru, poziţia curentă a robotului, se va obţine o imagine care reprezintă vecinătatea acestuia.

În Figura 4.18 se arată două situaţii posibile: situaţia când robotul **1** stă iar robotul **2** se deplasează spre acesta (Figura 4.18a) şi situaţia inversă când robotul **2** staţionează şi robotul **1** vine spre el. În ambele situaţii, după cum se observă şi din figură, când unul din roboţi ajunge la marginea vecinătăţii celuilalt robot se va opri.

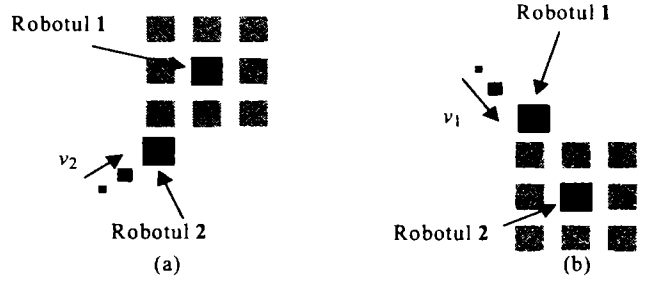


Figura 4.18. Vecinătatea roboţilor mobili de rază r=1 (vecinătate 3×3): a) vecinătatea robotului **1** la deplasarea robotului **2**; b) vecinătatea robotului **2** la deplasarea robotului **1**; v_1 – direcţia de deplasare pentru robotul **1**; v_2 – direcţia de deplasare pentru robotul **2**.

În oricare din situaţiile posibile, asemănătoare celor din Figura 4.18, robotul care a stat pe loc se va deplasa primul spre ţintă şi celălalt se va deplasa în urma sa, pe acelaşi drum.

O vecinătate mai mare pentru roboţi se poate obţine prin aplicarea repetată a template-ului DILATION [79]. Dacă se defineşte o vecinătate mai mare atunci şi ţinta trebuie să fie dimensionată corespunzător, sau trebuie mărită vecinătatea ţintei unde trebuie să ajungă roboţii mobili.

4.3.3. Atingerea ţintei de către roboţi

Roboţii mobili vor face alternativ câte un "pas" până când amândoi vor ajunge în vecinătatea ţintei. Dacă unul din roboţi ajunge la ţintă (Figura 4.19b), algoritmul va continua pentru celălalt robot până când şi acesta din urmă va atinge ţinta (Figura 4.19c). Dacă vecinătatea ţintei în care trebuie să intre roboţii este aleasă de rază r =1 (Figura 4.19a), iar roboţii au fiecare vecinătate de rază r =1, ca spaţiu de gardă pentru evitarea ciocnirii cu celălalt robot, atunci roboţii pot atinge ţinta evitându-se coliziunea dintre aceştia (Figura 4.19c). De menţionat că vecinătatea de rază r =1 este definită doar pentru robotul care staţionează, astfel că, de fiecare dată când roboţii se întâlnesc, se va păstra distanţa de un pixel între aceştia.

Algoritmul semnalează, dacă este cazul, situaţia în care roboţii ajung în vecinătatea ţintei venind din aceeaşi direcţie, desigur unul în urma celuilalt. În acest caz, primul care ajunge în vecinătatea ţintei se va deplasa în jurul ţintei cu doi pixeli sau chiar în direcţia opusă (Figura 4.19d), cedând locul său celuilalt robot.

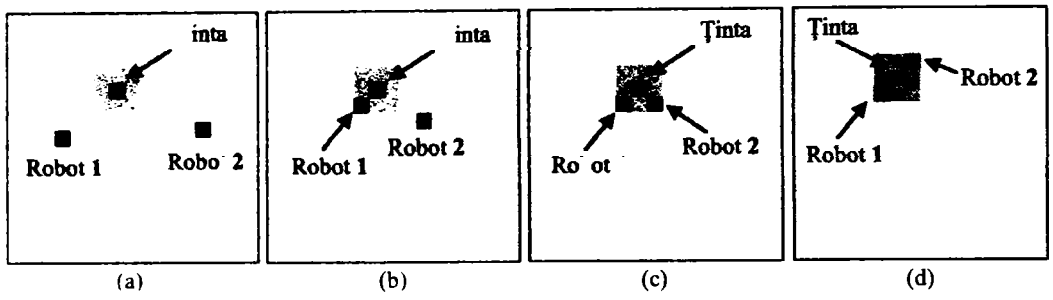


Figura 4.19. Exemplu de atingere a ţintei: a) cazul în care nici un robot nu ajunge la ţintă; b) cazul când unul din roboţi ajunge la ţintă; c) situaţie finală când ambii roboţi ajung la ţintă; d) situaţie finală în cazul în care roboţii au ajuns la ţintă din aceeaşi direcţie.

4.3.4. Rezultatele simulării algoritmului

Pentru testarea algoritmului de planificare a traiectoriilor celor doi roboţi, într-un mediu de lucru cu obstacole, s-a utilizat mediul de simulare **VisMouse – SimCNN** [73],[76]. Rezultatele au fost obţinute pe baza unor programe scrise în limbaj de asamblare **AMC** [77], (vezi Anexa A3). Vecinătatea robotului şi a ţintei au fost alese de rază r = 1, iar imaginile folosite au dimensiunea de 32×32 pixeli.

Sunt prezentate în Figura 4.20 rezultatele simulării pentru o situaţie concretă. Pentru o mai bună înţelegere a algoritmului, sunt prezentate imaginile obţinute după fiecare pas (momentele de timp t_0 ... t_{11}). Se observă că la momentul t_6 robotul **1** (R1) întâlneşte robotul **2** (R2) şi conform algoritmului se va opri la o distanţă de un pixel de acesta. Deoarece robotul **2** a ajuns primul la "punctul de întâlnire", o va lua înainte, iar robotul **1** se va deplasa în urma sa. Când robotul **2** ajunge în vecinătatea ţintei se va opri pentru moment, urmând ca în pasul următor să treacă de partea cealaltă a ţintei pentru ai face loc celuilalt robot.

Algoritmul a fost testat şi cu imagini de dimensiune 64×64 pixeli, dar în acest caz, dacă poziţiile de start sunt la distanţă mare faţă de ţintă şi sunt multe obstacole în mediul de lucru, e posibil ca unda generată din punctul ţintă să nu ajungă până la poziţiile de start. În astfel de situaţii va trebui introdusă o ţintă intermediară, respectiv un punct via prin care vor trece roboţii.

De asemenea, algoritmul CNN de planificare a traiectoriilor pentru doi roboţi poate fi uşor extins pentru situaţia în care la o ţintă trebuie să ajungă trei sau chiar mai mulţi roboţi. Dacă vecinătatea ţintei este definită ca fiind de rază r = 1, atunci numărul maxim de roboţi este patru, iar dacă vecinătatea ţintei este mai mare atunci şi numărul de roboţi ce pot fi folosiţi în algoritmul descris, va fi mai mare.

Timpul total de procesare necesar pentru derularea algoritmului variază puţin în funcţie de lungimea traiectoriilor celor doi roboţi, dar s-a observat că depinde aproape liniar de numărul de viraje cumulat al roboţilor. În Figura 4.21 se prezintă un grafic obţinut în urma măsurătorilor şi care arată o dependenţă aproape liniară între timpul total necesar pentru procesări şi numărul de viraje cumulat efectuate de către roboţi în drumul lor către ţintă. Timpii au fost măsuraţi în cazul simulării pe un calculator cu procesor Pentium III la 800 MHz.

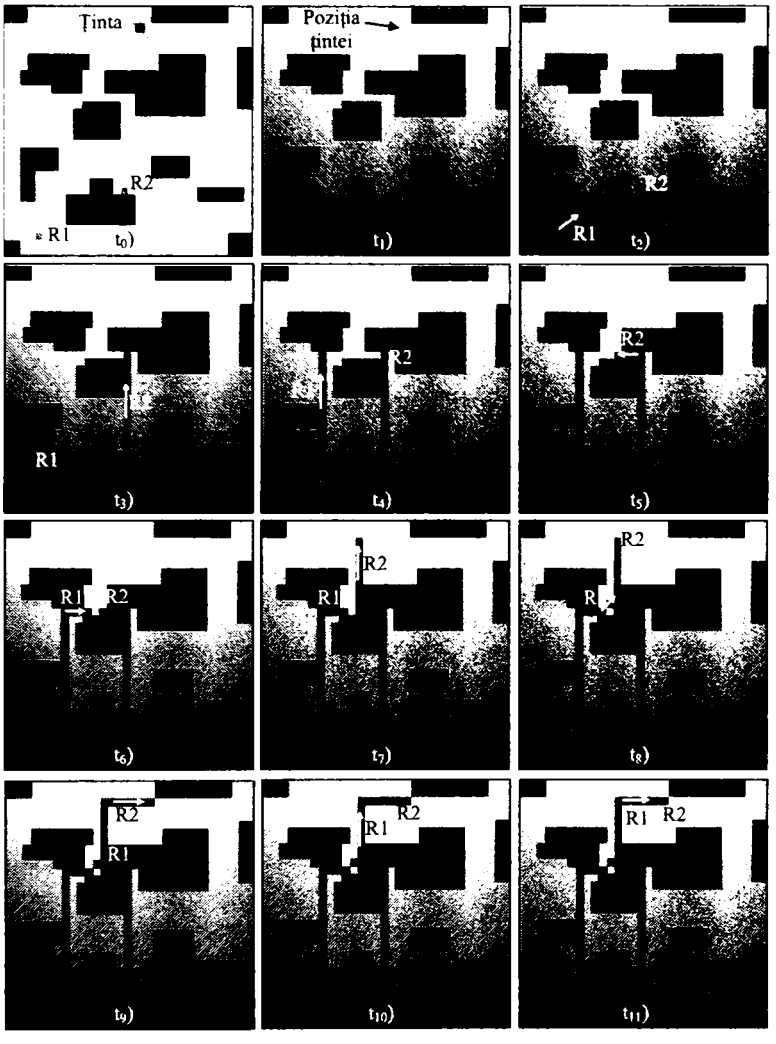


Figura 4.20. Rezultatele simulării algoritmului CNN de planificare a traiectoriilor pentru doi roboţi mobili la deplasarea lor spre o ţintă comună dintr-un mediu cu obstacole

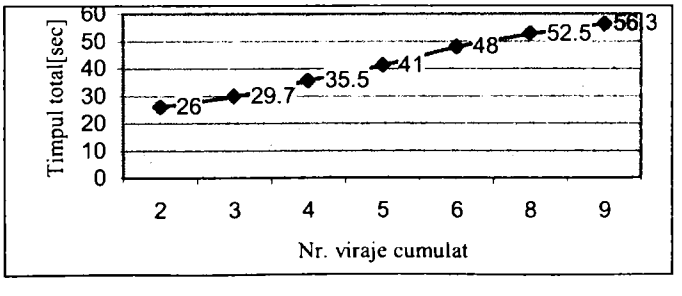


Figura 4.21. Dependenţa dintre timpul total de procesare necesar algoritmului şi numărul de viraje cumulat al celor doi roboţi.

4.4. Algoritm CNN pentru deplasarea unui robot mobil prin metoda *wall-following*

Navigarea autonomă a unui robot mobil într-un labirint, nu e deloc o sarcină uşor de rezolvat. Chiar şi în situaţia când acesta cunoaşte forma labirintului şi locul unde trebuie să ajungă, planificarea traiectoriei acestuia, astfel încât să ajungă la ţintă pe drumul cel mai scurt, necesită elaborarea unor algoritmi sofisticaţi. Dacă poziţia ţintei nu e cunoscută apriori de către robot, atunci acesta nu poate decât să se deplaseze aleator prin labirint fără să se ciocnească de pereţi, şi eventual să nu caute prin zonele pe unde a mai trecut, până ce va găsi ţinta [99].

În continuare se prezintă un algoritm, (conceput, realizat şi testat de către autor) prin care un robot mobil, ce se află în interiorul unui labirint, se deplasează spre un perete şi apoi paralel cu acesta până când va atinge ţinta. Robotul poate să aibă peretele în dreapta sau în stânga sa, în funcţie de sensul de deplasare. Traiectoria sa nu va fi una optimă, dar printr-o simulare preliminară în ambele sensuri de deplasare, din poziţia de start până la ţintă şi apoi prin evaluarea şi optimizarea celor două traiectorii obţinute, se poate determina în final ce traiectorie trebuie urmată de către robot astfel încât aceasta să fie cât mai apropiată de traiectoria optimă.

În general, operaţia de urmărire a peretelui (*wall-following*) se bazează pe senzori ultrasonici. Datele preluate de la aceştia sunt folosite pentru obţinerea unei reprezentări locale a mediului pe baza căreia este comandat robotul mobil [35]. Reprezentarea mediului poate fi făcută printr-o discretizare spaţială [30],[100], caz în care mediul este divizat într-un anumit număr de celule care pot fi "ocupate" sau "libere" într-un anumit grad sau pe baza unor trăsături [101],[102], spre exemplu mediul poate fi modelat printr-un set de puncte, linii şi plane.

O altă metodă de deplasare pe lângă un perete este prezentată în [34], unde robotul mobil este echipat cu o antenă formată din mai multe tronsoane îmbinate cu articulaţii. Când antena atinge peretele, aceasta se va îndoi din articulaţii şi în funcţie de unghiurile făcute de fiecare articulaţie se poate determina poziţia robotului faţă de perete. În [103] se utilizează procesări de imagini pentru detecţia liniei imaginare ce corespunde centrului coridorului şi în final se realizează ghidarea robotului mobil de-a lungul acestei linii.

Algoritmul reprezintă o concepţie originală şi se bazează pe informaţiile vizuale date de o cameră video, iar imaginile obţinute sunt prelucrate cu reţele neuronale celulare. Imaginea labirintului este binarizată şi apoi discretizată, astfel încât rezoluţia ei să corespundă cu rezoluţia CNN. În acest fel imaginea poate fi reprezentată de o reţea neuronală celulară având M×N celule.

4.4.1. Prezentarea algoritmului

În aplicaţia de faţă, se consideră un exemplu de labirint având forma din Figura 4.22. În această figură sunt indicate poziţia de start a robotului precum şi locul unde este situată ţinta. Robotul mobil, arătat în figură, trebuie să ajungă la poziţia ţintei deplasându-se pe culoarele labirintului.

Se presupune că robotul mobil nu cunoaşte poziţia ţintei, astfel că el va trebui să folosească doar informaţii locale pentru deplasare. Robotul va avea totuşi ca punct de reper, pereţii labirintului, astfel că la fiecare pas vor fi verificate poziţiile din jurul său (dacă sunt libere sau dacă sunt "ocupate" de pereţi) şi în funcţie de acestea el va alege să se deplaseze de-a lungul peretelui până când "va găsi" ţinta.

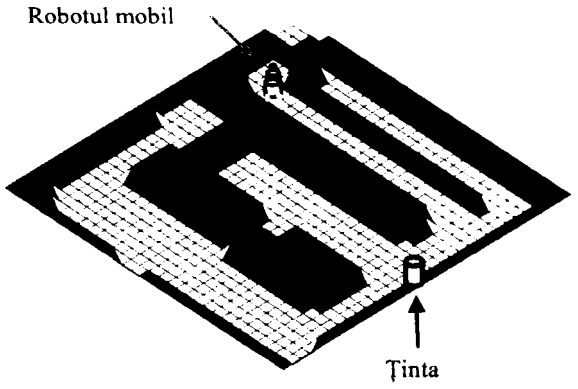


Figura 4.22. Forma labirintului, poziţia de start a robotului şi poziţia ţintei.

Se presupune în acest algoritm, că spaţiile libere au lăţimea de cel puţin doi pixeli iar ţinta se află situată lângă perete sau la o distanţă de cel mult un pixel faţă de acesta. În acelaşi timp, pereţii labirintului, se presupun uniţi între ei, sau dacă există pereţi izolaţi, ţinta nu se va afla lângă aceştia. În exemplul prezentat în Figura 4.22, labirintul nu are pereţi izolaţi iar ţinta se află situată lângă perete.

Algoritmul de deplasare pe lângă perete a robotului mobil a fost testat folosind mediul **VisMouse – SimCNN** [73],[75]. Programul utilizat pentru simularea deplasării robotului din punctul iniţial până la ţintă, în ambele direcţii de deplasare (cu peretele în dreapta, apoi cu peretele în stânga sa) a fost scris în limbaj de asamblare **AMC** [77], (vezi Anexa A4), pe baza organigramei prezentată în Figura 4.23.

4.4.1.1. Alegerea direcţiei optime de deplasare

După etapa de discretizare şi preprocesare cu reţele neuronale celulare, imaginea labirintului va fi în domeniul standard CNN, iar pixelii din imagine ce au valoarea +1 reprezintă pereţii labirintului respectiv poziţiile interzise pentru robot, în timp ce pixelii de valoare –1 sunt, de fapt, coridoarele labirintului respectiv zonele libere. Poziţia robotului mobil este identificată printr-un singur pixel de valoare +1.

În continuare, sunt verificaţi pixelii (dacă au valoarea +1 sau -1) din jurul pixelului ce reprezintă poziţia robotului şi va fi determinată poziţia viitoare. Această poziţie va fi aleasă astfel încât robotul să se deplaseze pe lângă peretele coridorului lângă care este situat. Poziţia viitoare (N, S, E sau V) va fi o poziţie liberă şi depinde de configuraţia poziţiilor (liberă sau ocupată), din jurul robotului (N, NE, E, SE, S, SV, V şi NV), (Figura 4.24).

După ce robotul mobil a ajuns lângă perete, poate să se deplaseze avându-l pe acesta în dreapta sau în stânga sa, prin simpla schimbare a sensului de deplasare. Schimbarea sensului de deplasare presupune schimbarea logicii de deplasare adoptate de către robot. Dacă robotul alege sau trebuie să se deplaseze de-a lungul unui perete, avându-l pe acesta în dreapta sa, va explora poziţiile vecine şi apoi va decide poziţia viitoare conform organigramei prezentată în Figura 4.25a.

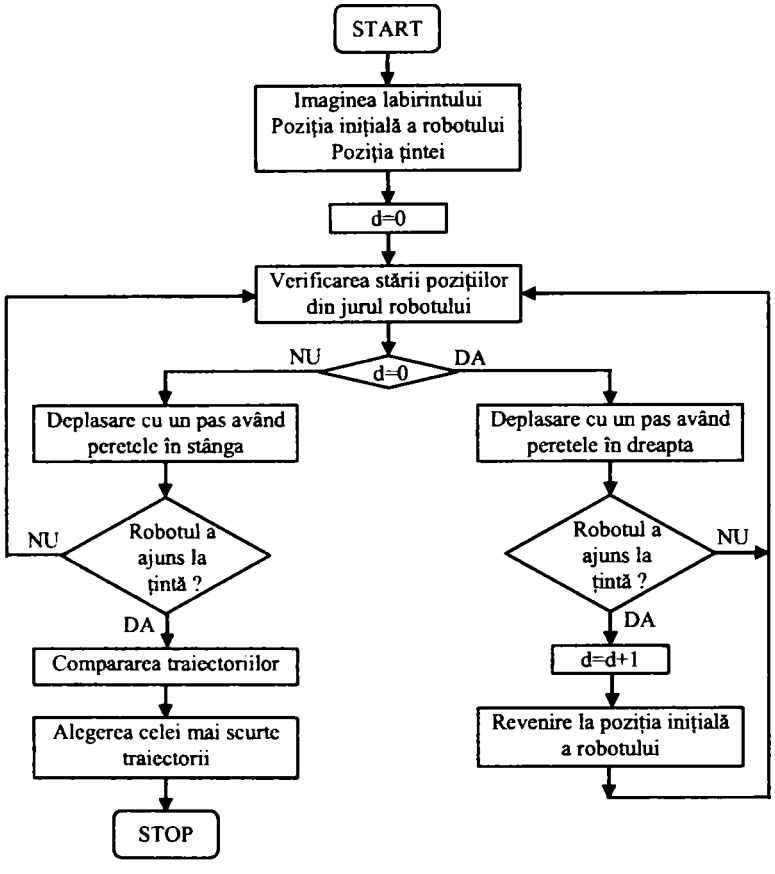


Figura 4.23. Organigrama algoritmului utilizat pentru deplasarea robotului mobil în cele două sensuri de deplasare.

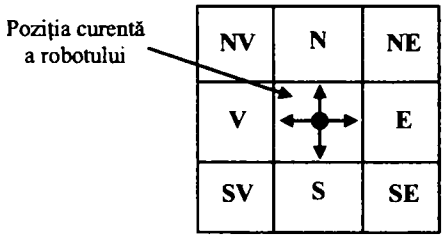


Figura 4.24. Pixelii din jurul pixelului ce indică poziţia robotului şi direcţiile de deplasare posibile.

Practic, pentru reducerea timpului de explorare a poziţiilor vecine robotului, vor trebui iniţial verificate doar poziţiile E, V, N, S şi dacă cel puţin una din aceste poziţii are valoarea +1 adică "ocupată", se va putea decide poziţia următoare fără a se mai verifica poziţiile din direcţiile SE, SV, NV, NE. În situaţia în care toate poziţiile din direcţiile principale E, V, N şi S sunt "libere" atunci robotul va fi nevoit să exploreze şi poziţiile din direcţiile SE, SV, NV, NE şi după ce doar una din aceste poziţii a fost găsită "ocupată" robotul va putea decide poziţia viitoare fără a mai verifica restul poziţiilor.

Dacă robotul trebuie să se deplaseze pe lângă perete, astfel încât acesta să se afle în permanenţă în stânga sa, atunci va adopta logica de deplasare prezentată în Figura 4.25b. Şi în această ipostază robotul va simplifica procesul de explorare a poziţiilor vecine, într-o manieră similară cu cea prezentată mai sus.

Robotul se va deplasa pas cu pas (pixel cu pixel) pe lângă perete, şi după fiecare pas se verifică pe baza algoritmului dacă este atinsă poziţia ţintei, iar dacă acest lucru se întâmplă se va opri. În ambele sensuri de deplasare lăţimea traiectoriei obţinute va fi de un pixel.

Pentru alegerea direcţiei de deplasare se utilizează procesări elementare AMC [77] asupra imaginii robotului (imagine cu un singur pixel negru), având la bază familia de template-uri SHIFT, care corespund celor patru direcţii principale de deplasare (relaţiile 4.16). În cazul unuia din template-urile din familia SHIFT, doar unul din elementele operatorului B sunt egale cu 1 restul fiind 0, astfel: SHIFTE (e = 1), SHIFTS (s = 1), SHIFTV (v = 1), SHIFTN (n = 1).

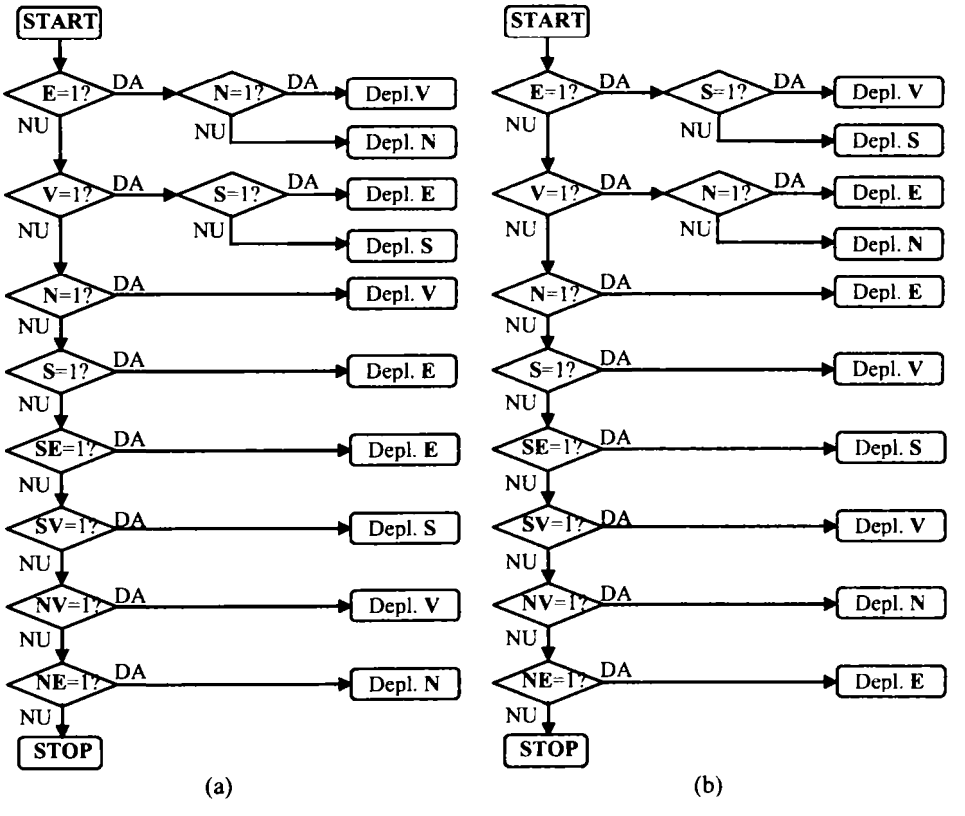


Figura 4.25. Logica de deplasare a robotului: a) având peretele în dreapta; b) având peretele în stânga.

$$A=\begin{pmatrix}0&0&0\\0&2&0\\0&0&0\end{pmatrix}\quad B=\begin{pmatrix}0&s&0\\e&0&v\\0&n&0\end{pmatrix}\quad z=0 \tag{4.16}$$

În Figura 4.26 se prezintă un exemplu concret de alegere a poziţiilor viitoare, în cazul în care robotul se deplasează având peretele în dreapta sa. Logica aleasă în acest caz va fi cea dată în Figura 4.25a.

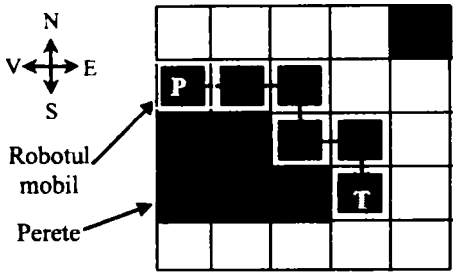


Figura 4.26. Exemplu de alegere a poziţiilor viitoare de către robotul mobil.

Robotul se va deplasa din punctul P spre punctul T, pe lângă peretele din dreapta sa, alegând succesiv direcţiile: E,E,S,E,S.

4.4.1.2. Determinarea traiectoriei optime

Pentru obţinerea unei traiectorii optime, ce uneşte poziţia iniţială de start a robotului şi poziţia ţintă, prin simulare "robotul" se va deplasa din punctul start către ţintă într-o direcţie apoi în cealaltă. Astfel, pe baza algoritmului prezentat mai sus, se vor obţine două traiectorii pentru robot, una când robotul se deplasează având peretele în dreapta şi una când acesta are peretele în stânga. Prin procesări CNN, din imaginile celor două traiectorii posibile pentru robot se alege imaginea cu traiectoria cea mai scurtă, iar aceasta la rândul ei poate fi optimizată astfel că în final se va obţine o traiectorie foarte apropiată de cea optimă, în sensul celui mai scurt traseu.

În Figura 4.27 sunt prezentate cele două traiectorii prin care robotul, pe baza algoritmului, poate să ajungă la punctul ţintă.

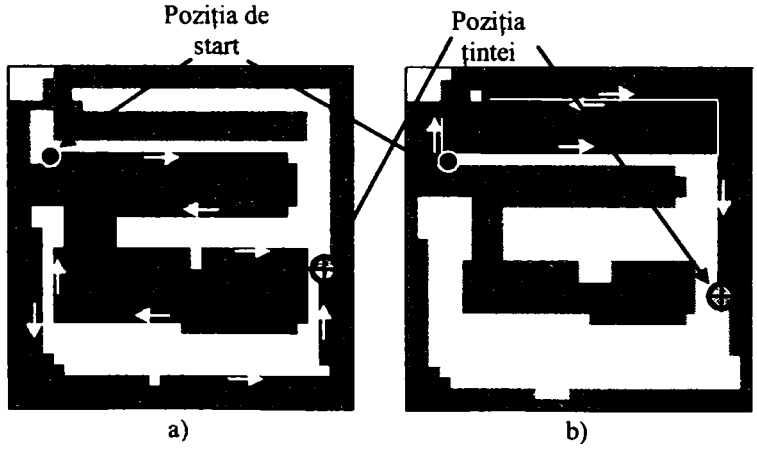


Figura 4.27. Traiectoriile robotului în cele două sensuri de deplasare: a) robotul are peretele în dreapta; b) peretele se află în stânga robotului.

Se observă că traiectoria din Figura 4.27b are lungimea mult mai mică în comparaţie cu lungimea traiectoriei din Figura 4.27a, astfel că aceasta va fi prelucrată în continuare pentru a fi optimizată.

O metodă simplă pentru determinarea traiectoriei optime poate fi aplicată dacă în imaginea labirintului coridoarele au lăţimea de doi pixeli. Prin procesarea imaginii labirintului cu reţele neuronale celulare se poate obţine o astfel de imagine.

După obţinerea celor două traiectorii la deplasarea robotului într-un sens şi în sens opus, imaginea traiectoriei cu lungimea cea mai mică va fi suprapusă cu imaginea labirintului (Figura 4.28a) şi se va aplica încă o dată algoritmul de deplasare prezentat mai sus tot în sensul în care s-a obţinut traiectoria cea mai scurtă. Astfel, pe baza acestei traiectorii, robotul nu va mai pătrunde inutil în anumite spaţii şi va descrie o traiectorie mult mai scurtă spre ţintă, (Figura 4.28b).

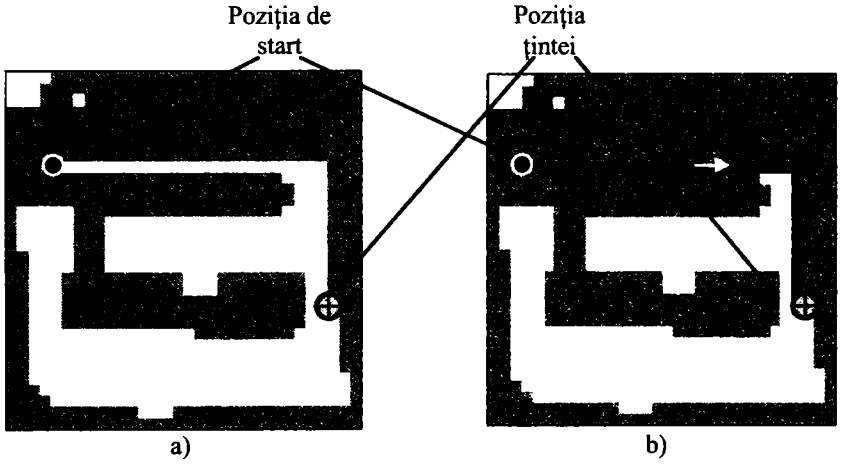


Figura 4.28. Modalitate de obţinere a traiectoriei optime în situaţia când în imaginea labirintului coridoarele au lăţimea de doi pixeli; a) imaginea procesată a labirintului; b) traiectoria optimă obţinută prin aplicarea algoritmului asupra imaginii labirintului prezentată la a).

4.5. Concluzii

Algoritmul CNN de planificare a traiectoriei unui robot mobil a fost realizat în ambele variante de alegere a direcţiei de deplasare a robotului (subcapitolele **4.2.2.1** şi **4.2.2.2**) şi simulat în aceleaşi condiţii, urmărind diferenţele dintre rezultatele obţinute, mai ales în ceea ce priveşte timpul total de procesare. Timpii au fost măsuraţi în cazul simulării pe un calculator cu procesor Pentium III la 800 Mhz.În Figura 4.29 se prezintă un grafic care arată dependenţa dintre timpul necesar pentru simularea algoritmului şi lungimea traiectoriei obţinute, exprimată în număr de pixeli, în cele două variante de alegere a direcţiei de deplasare.

Se observă că în cazul metodei de determinare a pixelului cu valoare minimă prin aplicarea template-ului PATH, timpul de procesare este foarte mic în situaţiile în care robotul se află aproape de ţintă şi este mai mare decât în cazul celeilalte metode doar în cazul în care robotul s-ar afla la o distanţă "foarte" mare de ţintă.

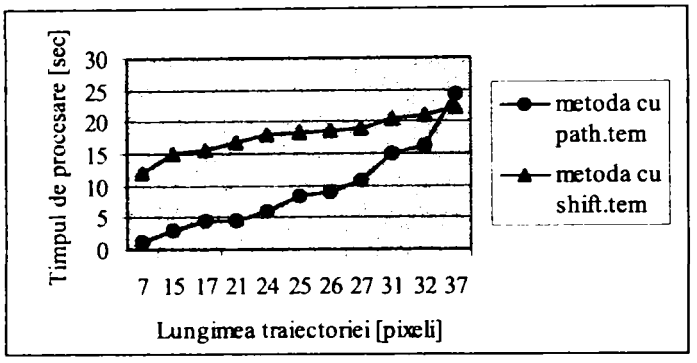


Figura 4.29. Dependenţa dintre lungimea traiectoriei parcurse de robot şi timpul de procesare necesar în cazul celor două metode pentru determinarea direcţiei optime.

Timpul total minim de prelucrare necesar algoritmului se obţine însumând timpii minimi necesari de prelucrare pentru fiecare etapă în parte. Din acest timp, timpul necesar pentru evaluarea distanţelor dintre punctele libere ale spaţiului de lucru şi poziţia ţintei are o pondere importantă. Acesta depinde de distanţa dintre poziţia de start şi poziţia ţintei, deoarece este absolut necesar ca unda, cu originea în ţintă, să se propage până la pixelul ce indică poziţia de start.

Unul din avantajele metodei propuse constă în faptul că unda ce se propagă din punctul ţintă se deplasează doar până când întâlneşte pixelul ce reprezintă poziţia robotului şi astfel timpul necesar propagării undei respectiv aplicării template-ului EXPLORE scade, mai ales în situaţia în care robotul se află aproape de ţintă. Pe de altă parte poziţia viitoare respectiv direcţia ce asigură apropierea faţă de ţintă se obţine prin aplicarea unui singur template (PATH) asupra imaginii actualizate a undei ce se propagă prin spaţiile libere din mediu.

Timpul necesar rulării algoritmului CNN de planificare simultană a traiectoriei pentru doi roboţi poate fi redus prin predicţia poziţiei viitoare [94], ce urmează să fie ocupată de către roboţii mobili la fiecare pas al algoritmului, astfel roboţii nu vor trebui "să aştepte" ca celălalt robot să parcurgă o porţiune dreaptă din traiectorie pentru a decide care va fi traiectoria viitoare a sa.

Referitor la metoda *wall-following* de deplasare a robotului mobil, s-a avut în vedere reducerea timpului de explorare a poziţiilor vecine robotului, astfel acesta va trebui iniţial să verifice doar poziţiile E, V, N, S şi dacă cel puţin una din aceste poziţii are valoarea +1 adică "ocupată", robotul va putea decide poziţia următoare fără a mai verifica poziţiile din direcţiile SE, SV, NV, NE. Mai mult decât atât, dacă în timpul verificării poziţiilor principale s-a determinat o poziţie ca fiind "ocupată" şi dacă a mai fost verificată cel puţin încă o poziţie principală, robotul va putea deja decide poziţia viitoare, fără a mai verifica toate poziţiile principale. În situaţia în care toate poziţiile din direcţiile principale E, V, N şi S sunt libere atunci robotul va fi nevoit să exploreze şi poziţiile din direcţiile SE, SV, NV, NE şi imediat ce una din aceste poziţii a fost găsită "ocupată" robotul va putea decide imediat poziţia viitoare fără a mai verifica restul poziţiilor.

Timpul total de prelucrare necesar algoritmului creşte aproape liniar cu lungimea traiectoriei măsurată în pixeli. În Figura 4.30 s-a reprezentat dependenţa acestui timp de lungimea traiectoriei în două cazuri distincte de aplicare a algoritmului şi anume cazul în care sunt verificate toate poziţiile din jurul robotului şi apoi se va lua decizia cu privire la poziţia viitoare (Metoda 1) şi cazul în care se

aplică algoritmul CNN sub forma prezentată în lucrare (Metoda 2). Timpii au fost măsuraţi în cazul simulării pe un calculator cu procesor Pentium III la 800 MHz.

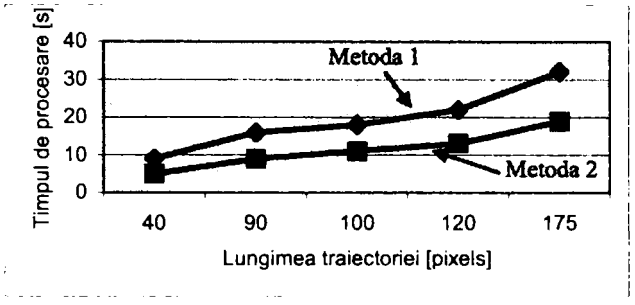


Figura 4.30. Dependenţa timpului de procesare al algoritmului de lungimea traiectoriei obţinute în cazul celor două metode de determinare a poziţiei viitoare a robotului.

Unul din dezavantajele algoritmului prezentat îl constituie faptul că robotul trebuie să verifice de fiecare dată, în cel mai bun caz, cel puţin două din poziţiile vecine din direcţiile principale de deplasare. În situaţia cea mai defavorabilă trebuiesc verificate toate cele opt poziţii situate în jurul robotului, ceea ce implică un timp relativ mare de procesare.

Timpul necesar algoritmului poate fi redus considerabil dacă prin aplicarea unui singur template CNN s-ar putea obţine poziţia viitoare a robotului la fiecare pas al algoritmului. De fapt, ar fi nevoie de două template-uri, unul când robotul se deplasează având peretele în dreapta sa şi un template atunci când acesta se deplasează având peretele în stânga. Prin aplicarea acestor template-uri cu ajutorul unei imagini mască, ce reprezintă vecinătatea robotului, s-ar putea indica mult mai rapid poziţia viitoare ce trebuie ocupată de către robot.

O condiţie necesară pentru buna funcţionare a algoritmilor prezentaţi în acest capitol este ca în imaginile achiziţionate să se poată identifica atât punctul de start cât şi punctul ţintă. Din aceste considerente rezultă dimensiunile imaginii şi pasul de eşantionare minim pentru discretizarea spaţială a imaginii achiziţionate.

5. COLECTIVITĂŢI DE ROBOŢI MOBILI

5.1. Noţiuni introductive

Necesitatea realizării unor sarcini tot mai complexe a făcut ca performanţele roboţilor mobili să crească continuu, prin utilizarea senzorilor şi actuatorilor avansaţi şi a algoritmilor de comandă inteligenţi. În acest fel, robotul mobil individual a devenit cu timpul, un dispozitiv foarte sofisticat deoarece trebuia şi trebuie să fie capabil să se adapteze, să înveţe, şi în special să colaboreze.

Cu toate acestea, unele operaţiuni fie nu puteau fi realizate de către un singur robot fie ar fi durat un timp inacceptabil de mare. Astfel, odată cu avansarea tehnicii în domeniul procesării informaţiei, senzorilor şi al structurilor de locomoţie, au luat fiinţă grupurile sau colectivităţile de roboţi. În cadrul acestora, roboţii componenţi iau în considerare pentru realizarea operaţiunilor, atât percepţia proprie dar şi informaţiile obţinute prin cooperarea cu ceilalţi roboţi din sistem. Ca urmare, complexitatea operaţiunilor realizabile de către roboţii mobili creşte, deschizând calea către un întreg spectru de aplicaţii. Ideea de bază este realizarea unor grupuri de roboţi care să lucreze în cooperare. Deşi având capacităţi limitate, astfel de roboţi, realizaţi în număr mare, pot reprezenta o forţă cumulativă puternică precum o colonie de furnici sau roi de albine.

Roboţii mobili dintr-un grup, interacţionează în mod constant, nu numai cu un mediu înconjurător dinamic, uneori ostil, dar şi cu alţi roboţi şi nu în cele din urmă, cu operatorul uman. Un astfel de robot va fi construit să fie parte integrantă a unui sistem complex, unde gradul de cooperare dintre structurile componente caracterizează eficienţa şi utilitatea sistemului. Inteligenţa colectivă ce se dezvoltă ca urmare a acestor interacţiuni, justifică denumirea de "societăţi de roboţi", pe baza analogiei cu structuri din natură. O societate robotică este un grup de roboţi (membri ai societăţii) care au abilitatea de a comunica şi de a realiza sarcini în comun. O societate este definită prin informaţiile de care dispune, precum şi de structura sa de comandă care face posibilă planificarea şi execuţia sarcinilor.

Se poate constata, în ultimii ani, un interes crescând privind teoria şi implementarea colectivităţilor (coloniilor) de roboţi, alcătuite din indivizi care cooperează între ei în vederea realizării unor sarcini specifice. Aproape fără excepţie, cercetările vizează coloniile de roboţi mobili, probabil datorită faptului că modelul biologic care a inspirat aceste cercetări - insectele sociale - se caracterizează, în general, prin mobilitate.

Există cel puţin două motive pentru care societăţile de roboţi prezintă interes potenţial pentru aplicaţiile tehnice. Un prim motiv îl constituie toleranţa la defecte. Conceptul de societate presupune implicit o înaltă redundanţă, în sensul că atribuţiile unui individ, devenit nefuncţional, pot fi preluate cu uşurinţă de către alţi membri ai societăţii, fără a fi necesară o reconfigurare a întregii societăţi. Cel de-al doilea motiv îl constituie structura de comunicaţie într-o societate, de la individ la individ, impusă de numărul mare de membri ai acesteia şi mai ales de situarea acestora mai mult sau mai puţin dispersaţi. Această caracteristică permite mărirea sau micşorarea cu uşurinţă a numărului de indivizi dintr-o societate, fără nici o redefinire a structurii de comunicaţie.

Este evident, că aplicarea conceptelor de mai sus este benefică într-o serie de aplicaţii tehnice, mai cu seama când se impune o funcţionare autonomă, în condiţii de mediu variabile şi cu o fiabilitate sporită.

Ca şi în cazul roboţilor mobili, studiul colectivităţilor de roboţi este în mod sigur un domeniu de cercetare inter-disciplinar. Această arie de cercetare combină metode din domenii dintre cele mai diverse cum ar fi: mecanică, electronică, ştiinţa calculatoarelor dar şi din biologie sau chiar filozofie.

5.1.1. Avantajele utilizării colectivităţilor de roboţi

Avantajele utilizării echipelor de roboţi a atras atenţia cercetătorilor mai ales în ultimii ani [9]. În viitorul apropiat, mediul în care vor opera cea mai mare parte a roboţilor mobili autonomi nu va fi nici stabil nici structurat. Acesta este unul din principalele motive pentru care o tendinţă majoră în robotică este către colectivităţile de roboţi denumite uneori şi sisteme multi-robot.

Folosirea unui sistem multi-robot cu scopul de a realiza în locul oamenilor diverse probleme se justifică, în primul rând, din punct de vedere economic. Descompunerea unei sarcini în mai multe sub-sarcini ce pot fi executate în paralel, în cele mai multe cazuri, măreşte viteza de execuţie. Pe de altă parte, mediul de operare al roboţilor poate impune restricţii cu privire la mărime, greutate, etc., ceea ce conduce la soluţia "divizării" unui robot complex în mai mulţi roboţi, fiecare din aceştia fiind specializat pentru o anumită operaţie. Spre exemplu, poate fi considerat mediul în care roboţii trebuie să treacă prin unele locuri etanşate cum ar fi în centralele nucleare în care o sarcină de curăţare trebuie să fie realizată într-un perimetru puternic radioactiv.

Sistemele robotice curente folosesc, de regulă, pentru aplicaţii la distanţă platforme sofisticate şi scumpe, tipic roboţi mari sau mijlocii desfăşuraţi ca o singură unitate, fiind înzestraţi adesea cu instrumentaţie complexă şi echipe de ingineri pentru diverse operaţii sau pentru service. Aceşti roboţi au costuri mari de realizare, sunt greu de transportat, şi pe lângă acestea, în multe cazuri, nu pot fi recuperaţi datorită expunerii lor la diverse pericole.

Utilizarea roboţilor multipli va face să crească siguranţa în funcţionarea sistemului, în timp ce scade complexitatea sarcinilor pentru un robot şi de asemenea timpul total de execuţie al unei sarcini complexe. În acelaşi timp, roboţii de dimensiuni mici pot reduce costul operaţiilor, pot înlocui muncitorii în locuri periculoase cum ar fi: zone radioactive, toxice, etc., şi pot creşte productivitatea.

Aşadar, se justifică utilizarea sistemele multi-robot în loc de un singur robot pentru anumite sarcini, din cel puţin următoarele motive:

- acestea pot realiza unele sarcini complexe ce nu pot fi realizate de către un singur robot;
- e mai economic să se construiască câţiva roboţi mici şi ieftini decât unul mare şi scump;
- se obţine o mai bună distribuţie spaţială - o echipă de roboţi poate acoperi o arie mult mai mare decât un singur robot. Acest avantaj e scos în evidenţă atunci când se fac operaţii de explorare sau căutare şi salvare.
- se pot efectua mai multe operaţii în paralel - un robot poate să facă doar puţine lucruri în acelaşi timp. În cadrul unei echipe de roboţi fiecare din ei poate să facă altceva.
- sistemul este mai robust şi mai sigur - dacă o parte din roboţi se defectează sau sunt scoşi din funcţiune, restul pot să continue operaţiile sau chiar să-i înlocuiască pe ceilalţi.

Au fost implementate şi metode hibride, spre exemplu, a fost realizat un sistem care constă într-un robot mai mare care explorează o clădire desfăşurând un robot mai mic în camerele şi zonele unde el nu poate să ajungă sau să intre din cauza gabaritului. Acest mod de explorare are unele avantaje faţă de modul de explorare cu singur robot, dar nu beneficiază totuşi de avantajele utilizării sistemelor de roboţi complet distribuite.

De asemenea, există anumite sarcini numite "strâns cuplate" (*tightly coupled tasks)* care, chiar nu pot fi executate de către un singur robot necesitând utilizarea mai multor roboţi care lucrează împreună [104]. Pentru execuţia acestor sarcini complexe, robotul trebuie să acţioneze într-un mod extrem de coordonat şi acest mod de lucru necesită cunoştinţe despre stările şi acţiunile echipei fie prin comunicare implicită (senzorială) fie prin comunicare explicită. De regulă, în cele mai multe cazuri acţiunea fiecărui robot este hotărâtoare în ceea ce priveşte realizarea cu succes a sarcinii, în sensul că o nereuşită individuală poate compromite funcţionarea echipei şi implicit nerealizarea sarcinii propuse.

Manipularea prin cooperare este un exemplu tipic de sarcină "strâns cuplată". Pentru transportul unui obiect prin cooperare roboţii trebuie să se coordoneze ei înşişi pentru a ridica obiectul şi a-l transporta sau împinge între două locaţii diferite.

Există însă şi tipuri de sarcini care pot fi executate de către un singur robot numite "slab cuplate" *(loosely coupled tasks)* dar se obţin performanţe mai bune când este folosită o echipă de roboţi.

5.1.2. Aplicaţii ale colectivităţilor de roboţi

În ultimul timp au apărut multe aplicaţii în care colectivităţile de roboţi mobili sunt utilizate cu succes. Cele mai cunoscute dintre acestea sunt:

- operaţii de căutare, salvare şi recuperare;
- îndepărtarea materialelor radioactive;
- explorarea unor domenii necunoscute (cercetări subacvatice, explorări planetare, etc.);
- operaţii de supraveghere;
- în minerit;
- manipularea unor obiecte grele, sau voluminoase.

Realizarea unor roboţi mobili păşitori compacţi pot avea aplicaţii, spre exemplu, pentru explorări planetare cum ar fi în cazul misiunilor de pe planeta Marte [105]. În general, NASA studiază posibilitatea trimiterii unor echipe de roboţi ca alternativă la trimiterea de fiinţe umane. Pentru auto-susţinerea acestor colonii de roboţi, trebuie proiectate sisteme de comandă flexibile pentru a putea fi stabilite interacţiuni multiple şi pentru obţinerea unor proprietăţi adaptive a roboţilor. De asemenea, roboţii trebuie să dispună de un sistem mecanic integrat astfel încât să fie capabili să-şi prelungească timpul de viaţă prin auto-depanare şi prin înlocuirea instrumentelor uzate sau rupte cu care aceştia operează.

În domeniul explorărilor planetare, trimiterea unui singur robot nu pare a fi o soluţie bună, deoarece dacă acesta se defectează atunci misiunea poate fi compromisă. Dacă sunt mai mulţi roboţi, atunci un altul poate să îi ia locul sau roboţii se pot ajuta între ei pentru a realiza unele sarcini.

În [106] sunt prezentate câteva posibile aplicaţii ale colectivităţilor de roboţi mobili. Spre exemplu, mai mulţi roboţi mobili pot fi dispersaţi în clădiri avariate pentru a căuta eventualii supravieţuitori prin detecţia sunetelor sau chiar a

respiraţiei umane. De asemenea, aceştia pot detecta o ţintă inamică într-o clădire. Un soldat poate, spre exemplu, să desfăşoare o echipă de roboţei care să "colinde" prin clădiri. Dacă unul din aceştia detectează, prin intermediul senzorilor săi, o mişcare umană sau sunet atunci va transmite un semnal spre o staţie de bază.

O echipă de roboţi mobili care se poate organiza şi deplasa în diverse forme ordonate, poate fi folosită, spre exemplu, ca o antenă distribuită într-un anumit spaţiu.

Securitatea clădirilor poate fi asigurată de echipe de roboţi care patrulează şi sunt echipaţi fiecare cu un sistem de intercomunicaţie.

Se prefigurează că în viitor roboţii mobili de dimensiuni minuscule vor putea fi injectaţi în corpul uman pentru a realiza unele "misiuni" cum ar fi "atacarea" celulelor cancerigene.

5.2. Modele de societăţi întâlnite în natură

Termenul "societate" duce cu gândul în primul rând la societatea umană, cu structura, legile şi comportamentul său extrem de complex. Şi în societatea animală sunt structuri ce posedă un nivel ridicat de inteligenţă, cum sunt mamiferele, dar există şi alte organisme complexe care au evoluat într-o direcţie diferită. În locul unui singur individ înzestrat cu inteligenţă superioară, natura a creat sisteme în care inteligenţa este distribuită între mai multe unităţi.

Un prim exemplu de organisme care au tendinţa să formeze structuri care au caracteristici apropiate de cele ale unei societăţi, sunt bacteriile.

Asemănătoare, dar mult mai rafinată, este organizarea întâlnită la anumite specii de insecte: furnici, albine sau termite. Pentru a avea o cooperare eficientă aceste sisteme au nevoie de un set de reguli de bază pentru a reduce pe cât posibil toate interferenţele şi concurenţa între membrii săi.

5.2.1. Bacteria

Până de curând bacteriile erau considerate a fi doar simpli microbi unicelulari. Deşi studiile făcute încă la începutul secolului trecut, demonstrau contrariul, au fost necesari zeci de ani ca bacteria să fie considerată organism multicelular cu abilitatea de a crea comunităţi, vâna în grup şi chiar să comunice [107].

Unele bacterii au capacitatea de a forma structuri stabile şi de o regularitate surprinzătoare. Organizarea coloniei este facilitată de deplasarea bacteriilor de-a lungul gradientului unei substanţe chimice pe care de altfel o şi răspândesc.

Această bacterie vânează prin secreţia unei enzime care dizolvă stratul protector al prăzii. Pentru a avea rezultatul scontat, enzima trebuie să aibă timpul necesar să-şi facă efectul. Strategia prin care bacteriile au rezolvat problema este următoarea: înconjoară prada şi apoi emit enzimele astfel încât aceasta nu are cale de scăpare. Aşteptând ca enzimele să acţioneze, bacteriile folosesc hrana prăzii, economisind astfel energie.

În [108] se precizează că bacteria a reuşit să folosească faptul că este un organism multicelular în favoarea sa: putere prin număr, potenţialul de a se specializa şi diviziunea muncii.

În viitor, când un sistem de roboţi va număra mii de membri, observaţiile făcute cu privire la aceste organisme simple pot constitui inspiraţii pentru modul de a controla aceste sisteme artificiale distribuite.

5.2.2. Insectele sociale

Inspiraţiile din domeniul sistemelor multi-robot au venit, în principiu, de la insectele sociale. Furnicile, în ciuda nivelului redus de inteligenţă, au supravieţuit schimbărilor petrecute în mediul înconjurător. Aceste animale cu structuri ce par la prima vedere haotice, dezvăluie la o cercetare mai atentă, un nivel ridicat de inteligenţă distribuită. Abilitatea de a supravieţui vine din structura şi redundanţa societăţii. Furnicile au unele caracteristici uimitoare cum ar fi:

- comunicare pe bază de substanţe chimice;
- auto-organizare;
- comportament coordonat;
- auto-activare.

Aceste vieţuitoare, care au reguli simple de convieţuire, pot demonstra un comportament complex, pe baza interacţiunilor între membri societăţii şi alţi membri sau mediu. Dacă aceste reguli pot fi identificate şi formulate într-un mod care să permită aplicarea lor în robotică, pot fi create sisteme artificiale care să prezinte robusteţe şi flexibilitate asemănătoare.

Se consideră că, comportamentul insectelor sociale se realizează pe baza unui program predefinit, care este executat în momentul în care senzorul detectează stimulii necesari. Analogia cu robotica este evidentă.

În [109] se citează un exemplu de comportament invocat de un anumit stimul, în acest caz o substanţă chimică. Într-un muşuroi de furnici, corpurile furnicilor moarte sunt transportate la grămada de cadavre. Prin extragerea unor substanţe din corpul furnicilor moarte şi atingerea unor furnici vii, acestea din urmă au fost cărate în aceiaşi grămadă.

Pentru cercetători, una din cele mai mari probleme este modul de obţinere a informaţiilor cu privire la comportamentul colectiv prin studierea comportamentului individual. În [110] se afirmă: "comportamentul colectiv este mai mult decât suma participanţilor, din moment ce la nivelul sistemului apar alte tipuri de comportament".

Se pune deci întrebarea: Cum reuşesc furnicile să dezvolte societăţi cu o structură atât de dezvoltată, în timp ce considerate individual, par să fie atât de ineficiente? În [111] se enumeră câteva posibile soluţii pentru a răspunde la această întrebare. În primul rând, comportamentul individual al furnicilor este posibil să nu fie atât de aleatoriu pe cât îl percepem. Furnicile nu sunt considerate particule aleatorii ci animale care comunică şi au numeroase moduri de a diviza sarcinile. O altă idee ar fi acceptarea comportamentului aleatoriu al indivizilor, dar să se admită faptul că organizarea lor compensează ineficienţa individuală.

Mecanismul care stă la baza îndeplinirii unei sarcini poate fi descris prin faptul că fiecare individ face ce face vecinul lui [109]. Cuplând acest mecanism cu reguli simple şi care sunt invocate de anumiţi stimuli, se poate obţine un sistem a cărui performanţă să fie superioară sumei performanţei tuturor indivizilor.

În [112] se foloseşte termenul *stigmercy* (termen inventat de biologul francez P. Grasse în 1995) pentru a descrie un astfel de comportament colectiv. Definiţia dată acestui termen este: „invocarea unui anumit comportament ca urmare a efectului produs în mediul înconjurător de comportamentul anterior".

În [113] se afirmă că, comportamentele complexe la nivelul societăţii (în acest caz procesul de auto-organizare în cazul formării fagurilor de albine sau selectarea colectivă a sursei de nectar) se dezvoltă de la sine, ca urmare a interacţiunilor concurente între sub-unităţi.

În afară de aceste exemple, multe alte lucrări studiază comportamentul insectelor şi îl modelează pentru al transfera apoi în simulatoare. Toate acestea dovedesc că insectele sociale pot rezolva probleme globale prin interacţiunea între membrii societăţii precum şi între membri şi mediul înconjurător.

5.2.3. Societăţi animale

Animalele dovedesc faptul că pot fi realizate soluţii viabile fără a fi nevoie de structuri de control şi comunicare complexe. Societăţile animale existente în natură pot fi divizate în două mari categorii: cele care diferenţiază şi cele care integrează [114].

Insectele sociale sunt un bun exemplu de societate care diferenţiază. Membrii societăţii sunt divizaţi în caste şi individul există pentru societate şi depinde total de aceasta. Aceste grupuri pot îndeplini sarcini care sunt imposibile oricăruia dintre indivizi.

Celălalt tip de societate are la bază atracţia indivizilor unul pentru celălalt. Membrii societăţii nu sunt înrudiţi. Fiecare individ este condus de motivaţii egoiste, ceea ce îi îndeamnă să formeze grupuri şi să profite de pe urma acestora. Un bun exemplu sunt haitele de lupi (comportament sincronizat) sau bancurile de peşti (coordonare redusă). În acest caz societatea există doar pentru binele individului şi nu invers.

Motivele pentru care cele două tipuri de societăţi au evoluat în natură sunt similare: protecţie împotriva prădătorilor, vânătoarea în grup sau coordonarea unor diverse sarcini comune (căutarea hranei, formarea şi deplasarea în grupuri).

5.3. Studii, implementări şi rezultate privind colectivităţile de roboţi mobili

Colectivităţile de roboţi mobili sunt proiectate de cele mai multe ori în manieră distribuită în ceea ce priveşte partea senzorială şi cea de execuţie, cu scopul de a realiza unele sarcini cu sau fără unele forme de cooperare [115]. Exemple unde se necesită o execuţie distribuită sunt: spălarea duşumelei, tunderea unui gazon, lucrări de recoltare, curăţarea pereţilor, supravegherea unui domeniu, etc. Toate aceste operaţii pot fi realizate de către un singur robot, fără cooperare cu alţi roboţi, într-un timp suficient, uneori prea mare. Pe de altă parte sunt sarcini de cooperare care necesită mai mult de un singur robot cum ar fi transportul unui obiect greu sau stingerea unui incendiu.

Proiectarea sistemelor, pentru aceste sarcini, necesită realizarea de diverse arhitecturi care folosesc fie un set omogen de roboţi fie unul neomogen. Comunicarea explicită dintre roboţi variază de la una inexistentă la o topologie total conectată în care fiecare robot are acces la cunoştinţe globale complete. De asemenea, mărimea sistemului poate varia de la câţiva roboţi la peste 100 de roboţi. Cele mai multe sisteme care au fost experimentate prin implementare fizică au avut mai puţin de 10 roboţi.

Aşadar, punând mai mulţi roboţi să realizeze o sarcină nu se obţin doar avantaje, apar probleme noi ce trebuie luate în considerare [10] cum ar fi:

- cum trebuie formulate, descrise şi alocate sarcinile în cadrul colectivităţii?
- cum trebuie folosite resursele primite de către roboţi?
- cum trebuie stabilite comunicarea şi interacţiunile dintre roboţi?
- cum se asigură ca roboţii să acţioneze coerent?

- cum se permite roboţilor să rezolve conflictele apărute între ei?

Răspunsul la prima problemă depinde de compoziţia setului de roboţi folosit. Dacă setul de roboţi este omogen, înseamnă că roboţii au capacităţi identice. Dacă însă aceştia sunt heterogeni vor avea capacităţi diferite, care sunt de dorit în unele aplicaţii. Distribuirea sarcinilor într-un set heterogen de roboţi este o problemă mult mai complexă, deoarece unele sarcini pot fi potrivite doar pentru unii roboţi din echipă.

O resursă, în ceea ce priveşte a doua problemă, poate spre exemplu, să fie: o unealtă (pe care roboţii o folosesc pentru a-şi îndeplini sarcina), mijlocul de comunicaţie sau spaţiul folosit. Practic, prin împărţirea spaţiului se urmăreşte evitarea coliziunii dintre roboţi. Prin împărţirea resurselor poate apărea problema blocării procesului. Roboţii care sunt "politicoşi" pot să aştepte la infinit ca ceilalţi roboţi să le elibereze resursele.

Cercetările privind societăţile de roboti îşi propun ca obiectiv elucidarea mecanismelor prin care poate fi controlată funcţionarea acestor structuri. În acest scop, fie prin simulare fie prin experimentare directă, în condiţii de laborator sau în mediii reale, sunt imitate o serie de comportamente inspirate cu precădere din comportamentul specific insectelor sociale (albine, furnici, termite) [11],[116],[117]. Câteva experimente, devenite de notorietate în literatura de specialitate, vor fi prezentate în continuare.

În [118] se prezintă o abordare a societăţilor de roboţi în care membrii acesteia trebuie să cerceteze un mediu iniţial necunoscut. În timpul testelor efectuate cu simulatorul construit pentru această societate, a devenit evident că un sistem dinamic (societate de roboţi) operând într-un mediu înconjurător dinamic, rareori are o stare ce poate fi caracterizată statică. Controlul parametrilor operaţionali cu ajutorul tehnicilor tradiţionale a fost imposibil. Aceşti parametri operaţionali sunt cei care controlează eficienţa membrilor societăţii (raza de comunicare, definiţia obstacolelor, parametrii pentru realimentarea cu energie, etc.). Pentru a rezolva această problemă, au fost aplicaţi Algoritmii Genetici pentru implementarea sistemului de control al societăţii. Parametrii au fost codaţi în genomi şi a fost creată prima generaţie. După un anumit timp de simulare, o parte din membrii societăţii au fost scoşi din joc, pe baza unui sistem de clasificare în funcţie de cât de adaptaţi sunt indivizii la cerinţele mediului. Apoi simularea a fost reluată. Experimentul a arătat că deşi algoritmul ales nu a fost cea mai bună alternativă datorită numărului redus al populaţiei, sistemul a funcţionat acceptabil.

5.3.1. Comunicarea dintre roboţi

Majoritatea metodelor de comandă a sistemelor multi-robot subliniază importanţa pe care o are modul de comunicaţie robot-robot, comunicaţie folosită în general pentru a schimba între aceştia mesaje de comandă şi date senzoriale locale. În [14] se precizează că, comunicarea dintre roboţi poate fi caracterizată de:

- **distanţa de interacţiune** – distanţa maximă dintre emiţător şi receptor în timpul transmiterii informaţiilor;
- **explicitatea transmiterii de informaţii** – dacă robotul emiţător intenţionează într-adevăr să transmită informaţii;
- **simultaneitatea interacţiunii** – timpul dintre momentul transmiterii informaţiei şi momentul recepţionării acesteia de către receptor.

Cooperarea dintre roboţii, care formează o echipă, poate fi realizată pe baza comunicării dintre roboţi, şi această comunicarea poate fi **implicită**, **explicită** sau poate fi realizată prin **comunicarea stării** robotului.

5.3.1.1. Comunicarea implicită

Comunicarea implicită reprezintă transmiterea informaţiei prin intermediul mediului de lucru [12]. Se consideră doi roboţi mobili R1 şi R2 care tund un gazon, şi că amândoi sunt capabili să detecteze iarba tunsă faţă de cea netunsă [14]. Robotul R2 nu trebuie să-l informeze pe R1 care parte din iarbă a tuns el. De fapt, nici R1 şi nici R2 nu trebuie să urmărească astfel de informaţii, acestea sunt memorate în mediu. Roboţii comunică în procesul de cosire presupus doar prin intermediul mediului, realizând astfel o comunicare implicită.

Un fapt interesant despre comunicarea implicită e că aceasta este adesea inevitabilă. Robotul R1 nu poate din senin să oprească transmiterea informaţiei despre ce parte din iarbă a fost tunsă. Din acest motiv, comunicarea implicită este extrem de robustă, nu poate fi oprită.

Distanţa pe care se poate realiza comunicarea implicită este dependentă de capacităţile senzoriale ale roboţilor. Un robot poate recepţiona informaţii lăsate de un alt robot dacă poate vedea, atinge, mirosi, auzi sau utilizând alte tipuri de senzori pentru detecţia acestora. Atingerea necesită o distanţă de interacţiune aproape zero, în timp ce vederea necesită doar o distanţă minimă.

Este interesant faptul că această comunicare este un produs secundar al unui comportament robotic oarecare, robotul nu intenţionează să comunice în această manieră (cu toate că se poate baza pe o astfel de comunicare). De asemenea, timpul dintre începerea transmisiei şi recepţionarea mesajelor este nelimitat. Informaţia poate rămâne în mediu un timp nedefinit. Totuşi în cele mai multe cazuri informaţia se risipeşte în timp. De exemplu, informaţia memorată în mediu despre tipul de iarbă cosită sau necosită se va "risipi" odată ce iarba a căzut. Robotul R2 nu va fi capabil "să spună" după un an, ce parte din iarbă a fost tunsă.

Comunicarea implicită are avantajul că este o comunicare simplă care, de regulă, apare în mod automat. Are însă ca şi dezavantaj faptul că este limitată la ceea ce pot distinge senzorii robotului şi la informaţia ce poate fi transmisă prin mediu. Spre exemplu, avansarea în realizarea sarcinii cum ar fi tunderea unei pajişti poate fi transmisă uşor, dar pe de altă parte, descoperirea unui mod de lucru eficient de către unul din roboţi nu poate fi transmis prin mediu.

În lumea animală un bun exemplu de comunicare implicită îl constituie furnicile, care comunică prin eliberarea unei substanţe chimice.

5.3.1.2. Comunicarea explicită

Comunicarea explicită reprezintă transmisia şi recepţia intenţionată a informaţiei. Acest lucru se obţine de obicei cu ajutorul unui mecanism de comunicaţie fundamental cum ar fi: comunicare serială, în infraroşu sau comunicare prin unde radio.

În sistemele multi-robot comunicarea explicită se realizează de regulă prin unde radio. Astfel, un robot poate să anunţe întreg sistemul despre poziţia sa, sau poate folosi un emiţător radio cu rază de acoperire mai mică pentru a comunica doar cu robotul din faţa lui. Topologia utilizată pentru comunicarea în sistemele multi-robot se întinde de la un graf complet la o structură (arbore) ierarhică de bază. Un exemplu în acest sens îl reprezintă un sistem constituit din muncitori şi lideri, unde

fiecare lider are în responsabilitate câţiva muncitori, care pot comunica doar cu acesta.

Una dintre cele mai interesante idei cu privire la comunicarea explicită în sistemele multi-robot este comunicarea abstractă vizavi de comunicarea situaţională. În cadrul comunicării abstracte, se presupune că, conţinutul unui mesaj are mai multe înţelesuri. Spre exemplu, un robot R1 poate trimite un mesaj altui robot R2 de tipul "mergi în locaţia *x*". Robotul R2 trebuie să fie capabil să se deplaseze spre locaţia *x*. La comunicarea situaţională, mesajul însăşi precum şi conţinutul mesajului are înţeles. De exemplu, dacă R1 trimite mesajul "vino spre mine 5 unităţi" R2 va fi capabil să-şi dea seama ce trebuie să facă din conţinutul mesajului. Tot din mesaj el poate să afle poziţia lui R1.

Acest concept este deosebit de puternic în situaţiile în care un lider (sau un alt reper) dirijează o echipă de roboţi. Spre exemplu, o comandă "toţi să vină in apropiere" este posibilă fără ca liderul să cunoască poziţia fiecăruia.

Distanţa de interacţiune, în cazul comunicării explicite, depinde de tehnologia de comunicaţie folosită şi poate fi de la câţiva centimetri la zeci de metri în cazul echipelor de roboţi distribuite. Transmiterea informaţiilor între doi roboţi se poate face instantaneu în sistem semiduplex când unul din roboţi este pe recepţie, celălalt va trimite informaţiile sau în sistem duplex când ambii roboţi pot trimite şi recepţiona simultan informaţiile.

Pentru realizarea comunicării implicite trebuie să se monteze pe roboţi dispozitive de comunicare, apoi aceştia sunt gata pentru trimitere şi recepţionare de informaţii. Siguranţa şi robusteţea comunicării depind de siguranţa şi robusteţea sistemului de comunicaţie folosit precum şi de structura mediului de lucru.

5.3.1.3. Comunicarea stării robotului

Comunicarea stării robotului reprezintă, de fapt, transmiterea de informaţii obţinute prin observarea comportamentelor robotului. Dacă se consideră sarcina de căutare a surselor de lumină, comportamentul implicat poate fi de hoinărire la întâmplare până când este detectată o sursă de lumină, apoi drept înainte spre aceasta. Dacă un robot oarecare hoinăreşte la întâmplare şi detectează un al doilea robot R2 făcând acelaşi lucru, R1 poate trage concluzia că R2 nu a găsit nici el o sursă de lumină. Pe de altă parte, dacă R1 detectează un R2 care se mişcă rapid în linie dreaptă, R1 poate să-şi dea seama că R2 a găsit o sursă de lumină şi poate să-l urmeze pe acesta cu scopul de a găsi şi el sursa de lumină.

Comunicarea stării are două premise. Un robot trebuie să fie capabil să distingă roboţii faţă de alte obiecte din mediu, şi trebuie să fie capabili să sesizeze şi să interpreteze acţiunile lor. Acest lucru este destul de greu de obţinut, uneori chiar proiectanţii unui robot au probleme în a spune ce face un robot.

Acest tip de comunicare are aplicaţii interesante la jocurile de competiţie, unde abilitatea unui robot de a ghici acţiunile altuia, este foarte importantă. Un exemplu este acela în care o echipă de roboţi caută obiecte pentru a le aduce înapoi la o bază. Un robot R1 poate să-şi dea seama că dacă un membru din echipa adversă R2 se deplasează în linie dreaptă într-o anumită direcţie acesta a găsit obiecte. R1 poate decide să urmărească robotul R2 pentru a găsi el însuşi obiectele. R2 pe de altă parte poate să-l inducă în eroare pe R1, aceasta fiind deci o strategie de apărare.

Ca şi în cazul comunicării implicite distanţa de interacţiune este dependentă de capacităţile senzoriale ale robotului. De regulă, senzorii vizuali sunt favoriţi datorită flexibilităţii acestora şi cantităţii mari de informaţii pe care o pot furniza.

De asemenea, comunicarea stării poate fi: implicită sau explicită. Un robot R1 poate fi capabil "să spună" ce face un oarecare robot R2 fără ca R2 să transmită intenţionat aceste informaţii. Aceasta este comunicarea implicită. Pe de altă parte, R2 poate să-l inducă în eroare pe R1, caz în care comunicarea este explicită.

Timpul dintre începerea transmisiei şi recepţionarea mesajului în cazul comunicării stării este instantaneu. Deoarece prin comunicarea stării sunt transmise informaţii despre ce face un anumit robot în acel moment, informaţia trebuie să fie recepţionată imediat.

Este interesant faptul că acest mod de comunicare este oarecum robust deoarece comunicarea între doi roboţi poate avea loc fără a se baza pe alţi factori. Astfel, dacă o mare parte din sistemul multi-robot funcţionează prost, partea rămasă poate continua să comunice fără probleme.

Comunicarea stării este un mod de comunicare care a apărut o dată cu colectivităţile de roboţi în care roboţii componenţi lucrează împreună şi este limitată de capacităţile senzoriale ale roboţilor. De exemplu, sarcina curentă pe care o realizează un robot poate fi comunicată în această manieră dar pe de altă parte comunicarea locaţiei unui obiect şi distincţia dintre un robot care se plimbă la întâmplare şi unul ce se mişcă rapid în linie dreaptă sunt operaţii dificil de realizat.

5.3.1.4. Analiza tehnicilor de comunicare

Fiind date cele trei tehnici menţionate mai sus, care din ele ar fi mai utilă? Adesea se presupune că, comunicarea explicită oferă cele mai multe avantaje. Pentru a alege un anumit tip de comunicare trebuie ţinut cont de câţiva factori cum ar fi: costul, siguranţa în funcţionare, timpul de execuţie al sarcinii, etc. În cadrul experimentelor există, de obicei, un set de sarcini ce trebuie realizat şi se urmăreşte timpul în care se realizează acest set cu diverse forme de comunicare între roboţi.

În lucrarea [141] se stabileşte că în unele situaţii comunicarea explicită nu aduce îmbunătăţiri în realizarea unei sarcini. Autorul a demonstrat, prin experimente, că echipele de roboţi se descurcă bine fără a folosi o comunicare explicită. În studiile sale trage concluzia că o comunicare explicită nu este esenţială la sarcini care includ o comunicare implicită, dar că o comunicare explicită îmbunătăţeşte semnificativ performanţele sistemului, în cazul sarcinilor ce se desfăşoară într-un mediu ce nu favorizează comunicarea implicită. În continuare, autorul găseşte că dacă comunicarea stării este prezentă, atunci comunicarea explicită furnizează puţine informaţii noi.

În [142] autorul a făcut un experiment în care o echipă de roboţi trebuie să mute un număr de pucuri la o destinaţie pe care ei trebuie să o descopere în decursul experimentului. Toţi roboţii comunică între ei şi există o arie dată unde se poate găsi destinaţia. Roboţii trebuie, de asemenea, să raporteze evoluţia lor la intervale date de timp. Autorul a descoperit că sunt trei lucruri care influenţează cooperarea roboţilor: gradul în care efectele acţiunilor pot fi sesizate, cantitatea de muncă care poate fi realizată de un robot şi costul acţiunilor redundante.

Apare aici întrebarea dacă comunicarea implicită este posibilă. Autorul concluzionează că sarcina de mutare a pucurilor se poate baza pe o comunicare implicită. Un robot poate sesiza uşor dacă un puc a fost mutat prin faptul că el nu se mai află în locaţia iniţială. Pe de altă parte, într-o echipă de roboţi este suficient dacă unul din roboţi află destinaţia deoarece acesta va informa restul echipei. Fără o comunicare implicită, toţi membrii echipei trebuie să găsească individual destinaţia. Dacă se renunţă la comunicarea explicită se introduce oarecum o redundanţă dar cu

toate acestea costurile ar putea fi mai mici. Sunt însă şi situaţii când fără o comunicare explicită, performanţele obţinute de grupul de roboţi sunt slabe.

Aşadar, din cele două lucrări prezentate, se poate trage concluzia că dacă la realizarea unei sarcini sunt posibilităţi de comunicare implicită, atunci comunicarea explicită nu mai este neapărat necesară, dar că ea este totuşi folositoare. Dacă costurile de realizare a acţiunilor redundante în prezenţa doar a comunicării implicite sunt mari, atunci comunicarea explicită este benefică.

5.3.2. Cooperarea roboţilor

Inteligenţa colectivă imprimată roboţilor mobili, nu implică în mod automat şi cooperare. Dacă se consideră mai mulţi roboţi care adună cuburi într-o arie predefinită, comportamentul roboţilor demonstrează un anumit tip de inteligenţă, având criteriile de bază: supravieţuirea şi îndeplinirea sarcinii. Totuşi, acest comportament este departe de a fi cooperativ. Poate deveni cooperativ prin introducerea în acest sistem a unei comunicări active. Spre exemplu, transmiterea de mesaje între roboţi, sau comunicare pasivă cum ar fi lăsarea unei substanţe mirositoare care poate fi urmărită de către roboţi.

În [11] a fost definit comportamentul colectiv ca fiind orice comportament a unui set de agenţi ce conţine cel puţin doi agenţi. Autorii consideră comportamentul cooperativ ca fiind o subclasă a comportamentului colectiv şi au propus următoarea definiţie: "Dată fiind o anumită sarcină specificată de proiectant, se poate spune că un sistem multi-robot cooperează dacă, datorită unui mecanism cu care sistemul este înzestrat (numit mecanism de cooperare), apare o creştere în randamentul sistemului".

Această definiţie prezintă o problemă conceptuală: randamentul sistemului trebuie evaluat. În cadrul experimentelor desfăşurate în laborator este relativ simplu de evaluat, dar în lumea reală, randamentul sistemului se evaluează în funcţie de o serie de factori, care trebuie ponderaţi, astfel încât estimarea să fie cât mai corectă. Printre altele, trebuie luate în considerare timpul necesar pentru îndeplinirea sarcinii precum şi numărul de roboţi rămaşi "în viaţă" la sfârşitul experimentului. Cu toate acestea, în lumea reală, randamentul unui sistem este greu de definit şi măsurat cu acurateţe. O soluţie o reprezintă executarea repetată a experimentului şi adunarea datelor în scopul analizării lor din punct de vedere statistic.

O altă definiţie poate fi găsită în [143]. Autorii afirmă că diferenţa între un comportament colectiv ne-cooperant respectiv unul cooperant este legată de modul în care o sarcină poate fi dusă la îndeplinire: "poate un singur robot să ducă la îndeplinire sarcina sau este necesară participarea mai multor roboţi?".

Pentru a fi prezent în diversele acţiuni ale grupului, mecanismul de cooperare trebuie implementat în arhitectura de comandă a colectivităţii de roboţi.

5.3.3. Comanda colectivităţilor de roboţi

Arhitectura de comandă folosită pentru colectivităţile de roboţi poate fi, în linii mari, de două tipuri:

- centralizată;
- descentralizată.

În cazul metodei de comandă centralizate, sistemul dispune de un controler central care supervizează şi coordonează întreaga echipă de roboţi sau poate exista un robot lider, care are acelaşi rol [12]. Comanda descentralizată presupune că

fiecare robot se auto-coordonează într-un mod similar pentru toţi roboţii, sau în mod inegal atunci când există o anumită structură ierarhică în cadrul sistemului [13].

În [10], se face o paralelă între caracteristicile principale ale acestor două tipuri de arhitecturi (Tabelul 5.1).

Centralizată	Descentralizată
- Este posibilă optimizarea mişcării tuturor roboţilor, deoarece supervizorul îi poate lua pe toţi în considerare în acelaşi timp. - Capacitatea de calcul a supervizorului trebuie să crească odată cu creşterea numărului de roboţi. - Faptul că doar un singur agent este implicat în planificarea mişcării, face sistemul multi-robot vulnerabil. Dacă supervizorul dintr-un oarecare motiv nu mai funcţionează corect atunci este compromisă funcţionarea întregului sistem.	- Este inerent necoordonată, deoarece robotul nu are o percepţie globală a acţiunilor. - O creştere a numărului de roboţi nu implică o creştere a capacităţii de calcul a roboţilor. - Problemele apărute la un robot nu afectează planificarea mişcării la nivelul celorlalţi roboţi.

Tabelul 5.1. Comparaţie între caracteristicile arhitecturilor de comandă centralizată şi descentralizată.

Cele mai folosite arhitecturi de control sunt cele descentralizate. Acestea au adesea unele avantaje în plus faţă de arhitecturile centralizate cum ar fi: toleranţă la defecte şi o mai bună siguranţă în funcţionare.

În lucrarea [137] sunt sugerate două metode pentru rezolvarea planificării acţiunilor în cadrul sistemelor multi-robot:

- Metoda tratării tuturor roboţilor, în spaţiul de configurare, ca un singur robot (de exemplu considerarea setului de roboţi ca un singur robot cu mai multe grade de libertate). Aplicarea acestei metode presupune însă, cunoaşterea configuraţiei iniţiale şi finale a întregului sistem.
- Metoda planificării "decuplate", în care planificarea pentru fiecare robot este obţinută în doi paşi: în prima fază este pregătită o planificare aproximativă (mai mult sau mai puţin independentă de ceilalţi roboţi), care va fi corectată în faza a doua în timpul deplasării acestuia.

În [138] este prezentat un planificator cu două nivele pentru evitarea coliziunii în sistemele multi-robot. Prin nivelul înalt al planificatorului se caută o traiectorie fără coliziune (dacă aceasta există), pentru fiecare robot şi nivelul de jos coordonează mişcarea robotului în intersecţiile traiectoriilor planificate. Algoritmul construieşte o reţea bazată pe o descompunere în patru a spaţiului de lucru, care este folosită de roboţi pentru evitarea locală a coliziunilor, cum ar fi mişcarea faţă/spate sau chiar mişcarea de-a lungul unei traiectorii pre-planificate.

Lucrarea [139] descrie o arhitectură flexibilă pentru a modela simultan mii de agenţi mobili autonomi. Comportamentul agenţilor se bazează pe o presupusă arhitectură în care comportamentele individuale sunt prioritare faţă de toate celelalte. Comportamentul de grup se bazează pe câmpurile potenţiale sociale, ale căror modele sunt extinse prin introducerea unei zone neutre, în cadrul căreia celelalte comportamente se pot desfăşura liber. Pe lângă acestea este evaluat efectul câmpurilor potenţiale sociale şi în prezenţa unor agenţi care se defectează,

precum şi în situaţia existenţei unor informaţii senzoriale incomplete, respectiv insuficiente.

În [140] sunt studiaţi un set de roboţi "politicoşi" în care toţi roboţii cedează în favoarea celorlalţi, sistemul având o structură descentralizată. Autorii pretind că obţinerea unei situaţii de blocaj este o problemă dificilă. În experimentul lor, autorii proiectează un mediu cu puţine restricţii pentru a fi mai uşor de detectat situaţiile de blocaj, şi anume:

- spaţiul liber constituit din drumuri drepte;
- lăţimea fiecărui drum este fixată astfel încât doi roboţi să nu poată merge unul cu altul pe acelaşi drum;
- doar trei ramificaţii sunt permise la o intersecţie a reţelei de drumuri.

Fiecare robot transmite informaţii despre starea sa la toţi ceilalţi roboţi. Astfel că, atâta vreme cât comunicaţia dintre roboţi funcţionează toţi roboţii au acces la cunoştinţe globale despre sistem. Roboţii utilizează o înţelegere de manipulare a resurselor pe care autorii o numesc comunicare "cinstită", care înseamnă că un robot nu va încerca să ia o resursă care este în "proprietatea" altui robot. Acest lucru este util mai ales când resursele în cauză sunt spaţiul, deoarece încercarea de a lua o astfel de resursă poate conduce la o coliziune. Dacă un robot vrea să intre pe o parte a drumului, care este deja ocupată, acesta va trebui să se oprească. Un robot care opreşte la o intersecţie, va verifica cu ajutorul cunoştinţelor sale despre poziţia celorlalţi roboţi, câţi roboţi trebuie să se oprească la acea intersecţie şi direcţia fiecărui robot. Prin acest mod poate fi detectată o posibilă situaţie de blocaj. Pentru eliminarea unor astfel de situaţii, unul dintre roboţi va primi statutul de lider. Starea roboţilor se va schimba temporar în cazul situaţiilor de posibil blocaj, de la complet autonomă la robot cu prioritate sau fără prioritate. Este o tehnică simplă de a rezolva situaţiile de blocaj din intersecţii deoarece liderul îi dirijează pe ceilalţi roboţi. După ce blocajul "s-a rezolvat", toţi roboţii implicaţi vor fi din nou autonomi.

Concluziile acestui experiment sunt utile pentru situaţii reale cum ar fi traficul maşinilor sau în aplicaţiile de acest gen dintr-un supermarket. În sistemele unde apar situaţii de blocaj cea mai uzuală soluţie este punerea temporară a unui agent care să preia sarcina de planificare a mişcării cu toate că sistemul în condiţii normale este descentralizat.

5.3.3.1. Coordonarea a doi roboţi mobili printr-o arhitectură de tip *leader-follower*

În [12] se prezintă o modalitate de concepţie şi implementare a sarcinii de transport a unei cutii cu ajutorul a doi roboţi mobili (Figura 5.1). Caracteristica principală a arhitecturii propuse (*leader-follower*), este că robotul conducător este responsabil pentru ghidarea ansamblului prin mediu. Schimbarea liderului adică preluarea conducerii de către celălalt robot este înţeleasă de roboţi prin comunicaţie implicită, şi în acest fel sunt folosite avantajele controlerelor total dependente de informaţiile senzoriale locale.

Prin experimente cu roboţi simulaţi şi chiar roboţi reali având capabilităţi restrânse de comunicare, s-a observat că utilizând strategii simple de comunicaţie implicită, se obţin nivele de performanţă similare sau comparabile cu cele obţinute de echipele de roboţi ce operează cu sisteme de comunicaţie complexe.

Metodologia prezentată se bazează, pe o arhitectură ierarhică compusă dintr-un lider care comandă echipa către o ţintă predefinită şi ceilalţi roboţi care se

străduiesc să menţină o prindere potrivită a obiectului ce trebuie transportat. În situaţia particulară cu doi roboţi transportând o cutie, se presupune că doar liderul cunoaşte poziţia ţintei şi astfel acesta va fi conducătorul pe toată durata operaţiunii. Celălalt robot va conduce grupul pe perioade scurte de timp, atunci când ansamblul trebuie să se deplaseze în spate, şi asta se întâmplă când liderul întâlneşte un obstacol.

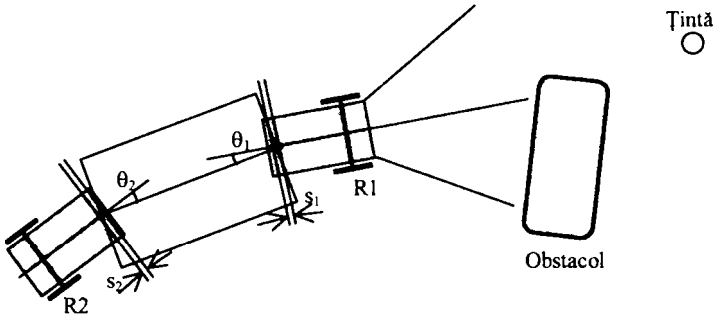


Figura 5.1. Doi roboţi transportând o cutie într-un mediu cu obstacole. Roboţii sunt capabili să sesizeze deplasarea liniară (s_i) şi unghiulară (θ_i) a cutiei.

Se poate observa un concept de schimbare a conducerii: liderul împrumută autoritatea celuilalt robot, care trebuie să o dea înapoi după o perioadă dată de timp sau atunci când cere liderul. Arhitectura de comandă propusă este arătată în Figura 5.2. Fiecare robot are două moduri de comandă principale: **conduce** (în stânga) şi **urmăreşte** (în dreapta). Când controlerul de la lider este activ, robotul respectiv ghidează ansamblul, timp în care controlerul de la celălalt robot este ocupat cu apucarea cutiei.

De obicei, în aplicaţiile de transport a unei cutii de către doi roboţi, controlerele robotului secundar sunt puternic dependente de poziţia şi viteza liderului, date care sunt transmise prin legătură radio.

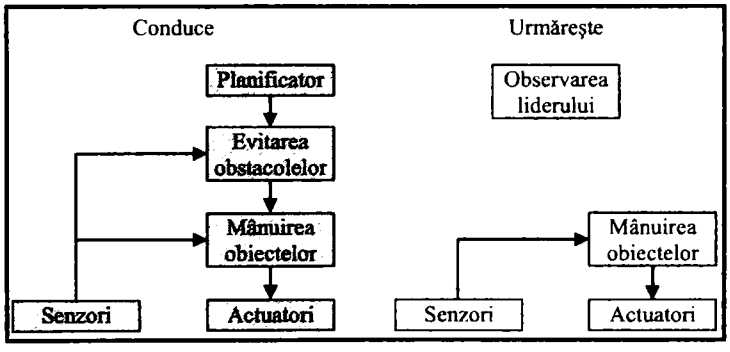


Figura 5.2. Arhitectura de comandă a roboţilor; suprafaţa mai închisă reprezintă controlerul activ al robotului lider.

În lucrare, comunicarea este considerată ca fiind limitată sau uneori chiar inexistentă, astfel că, controlerele vor fi puternic dependente de informaţiile senzoriale locale. Mişcarea robotului poate fi controlată în funcţie de interacţiunile sale cu obiectul. În acest fel, robotul conducător conduce într-o direcţie determinată,

pe când celălalt robot se străduieşte să menţină constante forţele şi momentele aplicate cutiei.

Iniţial, obiectul este în echilibru deoarece suma forţelor aplicate este egală cu zero. Când liderul începe să se mişte, vectorul rezultant este orientat spre direcţia de mişcare. În acelaşi timp controlerul robotului secundar îl coordonează pe acesta astfel încât suma forţelor să tindă spre zero. Acest lucru se poate vedea şi din Figura 5.2, unde arhitectura propusă nu ia în considerare nici un fel de schimb de informaţii explicite dintre roboţi, astfel că, controlerele principale nu depind unul de altul.

Deoarece poziţia ţintei este necunoscută de către robotul secundar, când acesta preia conducerea nu ştie în ce direcţie să se deplaseze. Totuşi, prin observarea traiectoriei anterioare, parcurse de lider, poate să aleagă o traiectorie rezonabilă. O intuiţie simplă care s-a dovedit experimental ca fiind eficientă este deplasarea pe traiectoria liderului văzută în oglindă, fapt care îmbunătăţeşte considerabil procedura de evitare a obstacolelor. O aproximare bună a traiectoriei stabilite de către lider este traiectoria descrisă de robotul secundar, deoarece ambii roboţi se deplasează strâns uniţi prin intermediul cutiei. Această traiectorie poate fi uşor estimată pe baza senzorilor odometrici. Dacă acest lucru nu este posibil datorită anumitor restricţii (lipsa memoriei sau a elementelor de procesare), pentru memorarea traiectoriei liderului, robotul secundar poate, mai simplu, să se deplaseze înapoi menţinând aceeaşi orientare absolută. Deşi acest fapt poate să nu conducă spre o direcţie optimă, ansamblul poate în cele din urmă să evite obstacolul.

Schimbarea conducerii este cea mai dificilă sarcină ce trebuie realizată fără comunicare explicită. De regulă, cei mai utilizaţi algoritmi de evitare a obstacolelor constrâng robotul să se deplaseze înapoi când e imposibil de evitat un obstacol. Deoarece acesta e un caz la îndemână, robotul secundar detectează acest tipar de mişcare, presupunând că liderul a renunţat pentru un timp la conducere. O situaţie similară apare atunci când robotul secundar trebuie să abandoneze comanda grupului. Acest mecanism reprezintă, de fapt, o formă de comunicare implicită. Cu toate acestea, acest fapt poate fi văzut ca un simplu protocol explicit de non-confirmare, deoarece roboţii pot intenţionat să preia conducerea prin deplasarea înapoi. Oricum acest protocol de comunicaţie este limitat, deoarece alte mesaje cum ar fi cerere de conducere sunt dificil de transmis.

5.3.4. Sarcini pentru studiul colectivităţilor de roboţi

Sarcinile cele mai utilizate pentru studiul colectivităţilor de roboţi mobili sunt: căutarea şi recuperarea anumitor lucruri dintr-o arie dată (*collective foraging*), transportul unui obiect sau împingerea unei cutii între două locaţii (*multi-robot box-pushing*), şi deplasare grupului de roboţi într-o anumită formă respectiv formaţie (*formation-marching*) [104],[115],[119]. Aceste sarcini colective au fost studiate atât cu roboţi fizici reali cât şi prin simulare.

5.3.4.1. Explorarea distribuită a unor spaţii

În această categorie de aplicaţii pot fi incluse operaţii cum ar fi: tunderea unui gazon, lucrări de recoltare, curăţarea sau spălarea unei suprafeţe, supravegherea unui domeniu, etc. În literatura de specialitate sunt multe exemple de astfel de aplicaţii, câteva dintre acestea fiind prezentate în continuare.

Un prim exemplu [120], abordat prin simulare, are în vedere o colectivitate de roboţi mobili capabili să se deplaseze autonom într-un mediu apriori necunoscut evitând obstacolele întâlnite. O parte din roboţi au drept scop colecţionarea de obiecte şi transportul acestora la o staţie de bază, în timp ce o a doua categorie de roboţi îi alimentează pe primii cu energie, prelevată de la aceeaşi staţie de bază. Comunicaţia între membrii colectivităţii se poate realiza numai local, adică un individ poate face schimb de mesaje doar cu indivizi situaţi într-o anumită vecinătate. În acelaşi timp, operatorul poate şi el comunica cu roboţii, prin intermediul staţiei de bază (cu rol de interfaţă). Studiul scoate în evidenţă o serie de concluzii interesante privind performanţele societăţii (număr de obiecte colecţionate / energie consumată) în funcţie de o serie de factori (numărul de membri ai colectivităţii, distanţa maximă până la care este posibilă comunicaţia, posibilitatea unui schimb de energie între roboţi, etc.).

Un al doilea exemplu, prezentat în [121], abordează problema cooperării dintre doi roboţi mobili care efectuează curăţenie într-o încăpere. Roboţii sunt dotaţi cu un sistem senzorial (senzori ultrasonici şi palpatoare) care le permit evitarea obstacolelor şi urmărirea conturului pereţilor încăperii. Unul din roboţi este echipat, în plus, cu un senzor vizual capabil să achiziţioneze scene din mediul în care se deplasează. În fine, ambii roboţi sunt înzestraţi cu uneltele necesare realizării sarcinilor impuse: perie şi aspirator pentru unul dintre ei (robotul PA) şi respectiv numai aspirator (robotul A), în cazul robotului dotat cu senzor vizual.

Autorii şi-au imaginat şi au experimentat 4 scenarii, cu complexitate crescândă.

Conform primului scenariu, robotul PA efectuează curăţenie şi depozitează reziduurile colectate în grămezi, în locuri deja curăţate. Cel de-al doilea robot (A), localizează poziţia acestor grămezi, utilizând senzorul vizual şi le colectează prin aspirare. Autorii au denumit acest gen de activitate "cooperare accidentală", având în vedere că dependent de acţiunile celor doi roboţi, cooperarea poate fi prezentă sau dimpotrivă ea poate lipsi.

Cel de-al doilea scenariu experimentat, aşa-numita cooperare prin observare, reprezintă o variantă optimizată a cooperării accidentale. De această dată cooperarea este prezentă explicit, robotul A urmărind activitatea robotului PA. Ori de câte ori robotul PA depozitează reziduuri, robotul A localizează poziţia acestora şi le colectează. O asemenea manieră de cooperare creşte eficienţa echipei.

Scenariul al treilea optimizează activitatea celor doi roboţi ca urmare a faptului că aceştia pot schimba mesaje simple între ei; robotul PA îl poate informa, spre exemplu, pe robotul A, de intenţia de a depozita reziduurile colectate precum şi de poziţia viitoarei grămezi. În această situaţie robotul A îşi poate optimiza deplasarea, rezultând de aici o creştere a eficienţei activităţii echipei. Schimbul de mesaje elimină, totodată, eventualele ambiguităţi în activitatea robotului A, în situaţiile în care grămezile realizate de robotul PA nu pot fi cu certitudine localizate pe cale vizuală.

Ultimul experiment prezintă o îmbunătăţire considerabilă a experimentelor precedente, prin faptul că cei doi roboţi mobili au hărţi ale încăperii în care ei navighează autonom. În aceste hărţi, punctele de reper sunt reprezentate dependent de natura informaţiilor senzoriale achiziţionate: semnale de la senzorii ultrasonici şi palpatori, în cazul robotului PA şi respectiv imagini în cazul robotului A. În aceste condiţii pot fi imaginate scenarii foarte diverse şi complexe. Spre exemplu, robotul A localizează vizual poziţii ale robotului PA, pe care le etichetează şi transmite aceste denumiri robotului PA. Acesta le marchează pe harta proprie, pentru a le utiliza în viitor ca puncte de reper faţă de care să îi precizeze robotului A,

poziţia viitoarelor grămezi ce trebuie colectate. Situarea acestora este dată relativ la cel mai apropiat punct etichetat faţă de grămada respectivă. Activitatea robotului A este în această situaţie considerabil uşurată şi eficientizată, deplasarea făcându-se acum atât pe baza informaţiilor furnizate de hartă cât şi a imaginilor achiziţionate de senzorul vizual.

5.3.4.2. Transportul colectiv

Transportul unui obiect masiv cu ajutorul roboţilor mobili presupune o cooperare strânsă între aceştia. Realizarea cu succes a unei astfel de sarcini este departe de a fi o banalitate. Aceasta presupune stabilirea unor interacţiuni între roboţi în funcţie de tipul de comunicare dintre ei, care poate fi explicită sau implicită. Un tip de comunicare implicită este realizată prin sesizarea forţelor aplicate obiectului transportat de către celălalt robot sau ceilalţi roboţi.

Acest tip de sarcină, precum şi multe altele, folosite pentru studiul colectivităţilor de roboţi fac analogie cu sistemele biologice. Spre exemplu, un grup de furnici au rezolvat problema transportului unui obiect masiv prin sesizarea forţelor şi cuplurilor aplicate obiectului respectiv [122]. Pe baza acestor informaţii ele îşi schimbă direcţia forţei aplicate în concordanţă cu necesitatea unor furnici de a-şi schimba poziţia de prindere. Multe alte exemple similare pot fi întâlnite în natură în jurul nostru. Testate prin evoluţia a milioane de ani, aceste structuri sociale au dovedit că se descurcă în medii dinamice şi ostile dovedind astfel valabilitatea informaţiilor şi inspiraţiei venite din natură pentru a realiza structuri de roboţi similare.

În [123] este introdus un sistem de transport a materialelor compus din roboţi de dimensiuni reduse (în principal prin simulare), fără un control centralizat şi cu comunicare inter-roboţi redusă la minim. În acest sistem, roboţii se organizează în echipe, se adună în jurul încărcăturii ce trebuie transportată şi care este stocată pe paleţi, o ridică şi o deplasează la ţintă, unde o lasă jos. Avantajele unui astfel de sistem de transport sunt, printre altele, faptul că roboţii de dimensiuni mici pot ajunge în locuri în care unităţi mai mari nu ar avea acces, iar echipele se pot organiza în funcţie de mărimea şi greutatea încărcăturii. Misiunea este îndeplinită cu ajutorul unui lider dinamic. Spre exemplificare, unitatea cea mai aproape de încărcătură a fost aleasă ca lider şi restul echipei trebuia să o urmeze. În cazul în care în timpul executării misiunii se întâmplă ceva cu liderul, un nou lider este ales şi misiunea continuă.

Următorul exemplu este inspirat de modul în care unele insecte îşi transportă hrana. Aplicaţia practică a unei asemenea forme de cooperare ar putea-o constitui manipularea de materiale. Pentru studiu se recurge la experimentul denumit *box-pushing*, în care doi sau mai mulţi roboţi mobili cooperează pentru a deplasa prin împingere o cutie de-a lungul unei traiectorii prestabilite. Soluţii foarte diverse, date acestei probleme, sunt menţionate în [109].

În [124] sunt prezentate rezultatele studiilor orientate spre obţinerea cooperării în cadrul unui sistem de roboţi care nu beneficiază de o comunicare explicită. Inspirat de insectele sociale, experimentul s-a concentrat asupra deplasării colective a unui bloc. Atât în mediul simulat pe calculator cât şi în cazul roboţilor din laborator, membrii sistemului au căutat blocul, s-au aliniat corespunzător şi au deplasat blocul la destinaţie. Un fapt demn de remarcat este că sistemul era compus din unsprezece roboţi.

Un alt exemplu interesant de transport colectiv este prezentat în [125]. În experiment sunt implicaţi doi roboţi care au sarcina de a transporta un obiect spre

un punct ţintă, evitând obstacolele. Elementul de noutate al experimentului îl constituie faptul că fiecare robot este prevăzut cu un dispozitiv (cap de captare), la care se poate ataşa cu uşurinţă obiectul ce urmează a fi transportat. În această situaţie, cei doi roboţi mobili sunt rigid cuplaţi între ei, singurul grad de liberate permis fiind rotaţia în raport cu obiectul "prehensat". Este evident, că urmare a acestui fapt, numai printr-o bună cooperare între cei doi roboţi este posibilă realizarea sarcinii impuse. Cooperarea înseamnă, în acest caz, o deplasare a celor doi roboţi mobili astfel încât componentele forţei ce apare în legătura rigidă să se încadreze în limite acceptabile. Dispozitivul de captare este prevăzut cu senzori pentru măsurarea valorilor acestor componente, astfel încât pe baza lor să se poată lua decizii în consecinţă.

5.3.4.3. Deplasarea în formaţie

Păsările, peştii şi multe alte animale se deplasează de multe ori în stoluri, bancuri, haite, etc. În aceste grupuri, animalele trebuie să se deplaseze într-o anumită vecinătate impusă de grup şi să evite coliziunile cu vecinii şi cu obstacolele din mediu.

Pe baza acestor observaţii, mersul în formaţie al roboţilor mobili (*formation-marching*) este abordat în numeroase lucrări cum ar fi [104],[109] şi reprezintă un exemplu tipic de cooperare explicită. Sarcina roboţilor din grup este deplasarea lor astfel încât să fie păstrat un anumit aranjament fizic. În robotică, acest comportament îşi găseşte aplicabilitatea în aplicaţii cum ar fi relevarea terenului, depistarea minelor, etc., în general aplicaţii în care se necesită cercetarea unei suprafeţe.

În [126] sunt considerate patru tipuri de formaţii în care se pot deplasa roboţii mobili (Figura 5.3), şi anume: în linie, în coloană, în formă de romb, în forma de V.

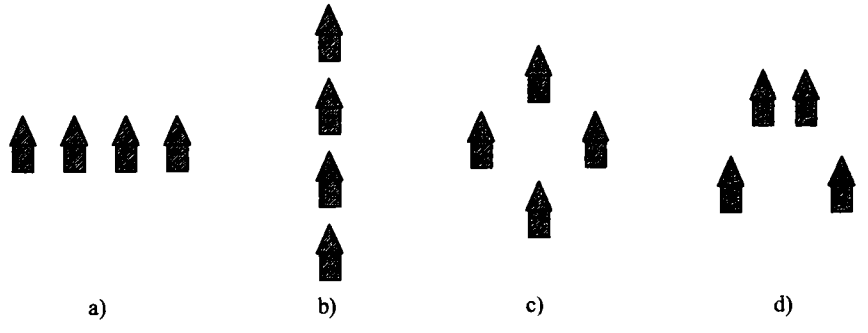


Figura 5.3. Exemple de formaţii pentru deplasarea unei colectivităţi de roboţi mobili; a) linie, b) coloană, c) în formă de romb, d) în formă de V.

Sarcina pentru fiecare robot este deplasarea spre ţintă, evitând obstacolele apărute în cale şi în acelaşi timp să evite coliziunea cu ceilalţi roboţi. Pe lângă toate acestea, roboţii trebuie să menţină o anumită poziţie în cadrul formaţiei. Pentru a realiza aceste cerinţe, fiecare robot îşi va determina poziţia proprie din formaţie pe baza poziţiilor celorlalţi roboţi. Sunt identificate trei tehnici pentru determinarea poziţiei în formaţie:

- în funcţie de centrul de simetrie al formaţiei – acest punct este calculat independent de către fiecare robot prin medierea poziţiilor *x* şi *y* ale tuturor roboţilor din formaţie.
- în funcţie de poziţia liderului - fiecare robot îşi determină poziţia faţă de poziţia robotului considerat lider. În acest caz, liderul nu este responsabil de menţinerea formaţiei, această sarcină le revine celorlalţi roboţi.
- în funcţie de poziţia unui robot vecin - fiecare robot menţine o poziţie prestabilită faţă de un robot vecin.

Dinamica şi stabilitatea formaţiilor de roboţi au atras atenţia în lucrarea [127]. Se prezintă o strategie pentru deplasarea formaţiei de roboţi, în care roboţii individuali se ghidează după un lider sau după roboţii vecini. În acest fel, informaţiile senzoriale pot fi minimizate, deoarece fiecare robot trebuie să cunoască doar poziţia liderului sau a unu ori doi roboţi vecini. Cercetările se bazează pe menţinerea structurii formaţiei printr-o comandă bazată pe o reacţie negativă şi pe stabilitatea sistemului rezultat.

În [117] este descris un algoritm pentru comanda mişcării unor agenţi care se deplasează în grup şi sunt evaluate, prin simulare, performanţele sistemului în cazul a trei tipuri de agenţi: roboţi păşitori (*legged robots*), biciclişti (*bicycle riders*) şi sisteme punctiforme (*point masses*). Algoritmul pune în evidenţă comportamentul de grup al agenţilor şi calculează poziţia dorită pentru fiecare membru al grupului pe baza poziţiei şi vitezei membrilor vecini (vizibili), ţinând cont, în acelaşi timp, de poziţia obstacolelor vizibile precum şi de viteza impusă pentru deplasarea grupului. Au fost comparate performanţele algoritmului în cazul simulării a trei situaţii diferite: deplasarea cu viteză constantă a grupului, întoarcerea şi evitarea obstacolelor. În toate aceste simulări, sistemele punctiforme au realizat cele mai bune performanţe deoarece controlul exercitat prin comportamentele de grup determină schimbări rapide şi predictibile ale poziţiei şi vitezei. Aceste schimbări predictibile permit acestor tipuri de agenţi să navigheze fără coliziuni chiar dacă obstacolele mari au deplasări lente. Predicţia mişcării roboţilor păşitori este mai dificil de realizat deoarece viteza lor variază în cadrul unui ciclu de rulare. Oricum, dacă a fost luată în considerare întârzierea inerentă în controlul vitezei, aceşti roboţi sunt capabili să treacă testele de evitare a obstacolelor şi de întoarcere. Agenţii de tip biciclişti sunt cei mai puţin robuşti la schimbările ce pot surveni în mediu, deoarece sistemul lor propriu este concentrat mai mult pe menţinerea echilibrului. Dificultatea testului de întoarcere a trebuit redus mult pentru ca aceştia să nu cadă sau să se ciocnească de vecini.

În [128] se arată că mai mulţi roboţi pot realiza o formaţie stabilă, în cazul simulării, dacă fiecare robot execută un algoritm identic pentru determinarea poziţiei în cadrul grupului. Fiecare robot poate percepe poziţia relativă a celorlalţi roboţi şi are abilitatea de a se mişca o poziţie de grid în timpul unei unităţi de timp. Prin ajustarea poziţiei fiecărui robot, relativ la cel mai îndepărtat sau la cel mai apropiat vecin, poate fi realizată prin simulare, o formă geometrică regulată pentru dispunerea roboţilor cum ar fi cercul.

Prin proiectarea algoritmului astfel încât fiecare robot să-şi determine poziţia în raport cu roboţii vecini, rezultă o formaţie stabilă, chiar fără o cunoaştere prealabilă cu privire la numărul total al roboţilor. Desemnarea liderilor permite reguli simple în cadrul grupului pentru a creea algoritmi de tipul *leader-follower* şi de a realiza divizarea formaţiei în grupuri mai mici.

În [129] este investigată, prin simulare, stabilitatea asimptotică a mai multor roboţi dintr-o formaţie. Fiecare robot este simulat ca o masă punctiformă şi îi percepe pe ceilalţi roboţi într-o regiune sub formă de con ce are vârful în centrul de

simetrie al robotului şi extins pe direcţia de mişcare. Formaţiile sunt reprezentate ca un set de deviaţii de la robotul de referinţă predefinit. În acest mod, o formaţie poate fi definită direct pentru fiecare robot, ca fiind un set de poziţii, faţă de: lider, cel mai apropiat vecin sau un set compus din cei mai apropiaţi vecini. S-a dovedit că eroarea de poziţionare, relativ la poziţia actuală, scade spre zero pentru fiecare robot independent din formaţie şi astfel formaţia dorită este asimptotic stabilă.

5.3.5. Exemple de implementări

Mare parte din studiile prezentate mai sus precum şi multe alte studii au fost implementate practic, luând fiinţă o serie de sisteme multi-robot care sunt compuse în cele mai multe cazuri din roboţi omogeni sau marea majoritate a acestora fiind omogeni. Câteva sisteme reprezentative în acest sens sunt prezentate mai jos.

Sistemul **CEBOT** (***CE****llular ro****BOT****ics system*) [130] este unul dintre primele sisteme apărute în domeniu şi cel mai cunoscut. Arhitectura descentralizată şi ierarhică a fost inspirată din organizarea entităţilor biologice. Sistemul poate fi reconfigurabil dinamic. Roboţii pot fi conectaţi fizic unul cu celălalt şi pot fi configuraţi în funcţie de necesităţi (mediul de lucru, tipul misiunii, etc.)

Sistemul **ACTRESS** (***ACT****or-based* ***R****obot and* ***E****quipments* ***S****ynthetic* ***S****ystem*) [131] constă dintr-o combinaţie heterogenă între roboţi şi staţii de lucru. Acest sistem poate realiza diverse sarcini. În [132],[133],[134] comportamentul grupului este văzut ca o colecţie de comportamente primitive pe baza cărora sunt construite interacţiunile locale şi care se combină pentru a genera acţiuni complexe. Sunt prezentate cinci tipuri de interacţiuni locale: plimbare, urmărire, dispersie, agregare şi *homing*. Acestea servesc ca blocuri de bază pentru construcţia acţiunilor complexe, cum ar fi deplasarea în formaţie a grupului de roboţi fără *leader* sau urmăritori predefiniţi.

Conceptul **BEROSH** (***Be****havior-based Multiple* ***Ro****bot* ***S****ystem with* ***H****ost for Object Manipulation*) prezintă un sistem multi-robot având sistemul central de coordonare ca *leader* iar membrii sunt omogeni. *Leader*-ul generează *target*-uri pentru roboţi şi oferă asistenţă limitată pe parcursul îndeplinirii misiunii, dar nu calculează dinamica ţintei sau distribuţia forţelor pentru cooperarea dinamică. Rezultatele experimentului arată că deşi roboţii sunt dotaţi cu sisteme de manipulare cu un singur grad de libertate aceştia au reuşit să transporte, împreună, obiectul la ţintă, chiar dacă au fost întâlnite pe traseu obstacole neaşteptate.

Un alt exemplu de implementare a unei colectivităţi de roboţi este prezentat în [135]. Se prezintă aici un concept robotic denumit **SWARM-BOT**, bazat pe un grup de roboţi mobili autonomi denumiţi *s-bots* având capabilităţi de auto-asamblare şi auto-configurare. Echipa realizată este compusă dintr-un număr de 30-35 de roboţi care au fiecare diametrul de 116 mm. Fiecare dintre aceştia este un robot mobil complet autonom, capabil să realizeze sarcini cum ar fi navigare autonomă, percepţia mediului şi apucarea unor obiecte. Pe lângă acestea un *s-bot* este capabil să comunice cu ceilalţi, astfel că, împreună ei pot să transporte obiecte grele sau voluminoase. Roboţii sunt echipaţi cu toţi senzorii necesari pentru o bună navigare, printre care şi o cameră video omnidirecţională. Sistemul de comandă este bazat pe un procesor ARM care poate rula la 400 Mhz în LINUX şi are o memorie RAM de 64 MB.

Tot pentru studiul colectivităţilor de roboţi mobili, au fost realizaţi roboţii în miniatură **Khepera** [136]. Aceştia au diametrul de doar 55 mm, înălţime 30 mm şi o greutate de 70 grame. Sunt echipaţi cu procesor pe 32 biţi la 16 Mhz, senzori odometrici şi de proximitate. Versiunea de bază are 8 senzori de proximitate în

infraroşu plasaţi în jurul său (6 în partea din faţă şi 2 în spate). Acumulatorii îi pot furniza energie pentru asigurarea unei autonomii de o jumătate de oră.

Robotul **Khepera** are o construcţie modulară care permite realizarea unei organizări flexibile. Astfel, pot fi echipaţi cu gripper, modul radio, modul de comunicaţie locală în infraroşu, sistem vizual stereo, etc. Arhitectura de comunicaţie dintre roboţi şi unitatea supervizoare este proiectată astfel încât transferul informaţiilor să nu fie afectat odată cu creşterea numărului de roboţi

În cazul comunicării prin unde radio sunt posibile două moduri: **standard** (masterul este unitatea de bază) şi **robot-based** (masterul este robotul). La modul standard, ce este curent folosit pentru colectivele bio-inspirate, toate tranzacţiile sunt pornite şi dirijate de către unitatea radio de bază, pentru un robot sau pentru toţi roboţii implicaţi în experiment. La modul **robot-based** nu este definit un protocol iniţial, fiecare robot poate începe o tranzacţie cu un alt robot sau cu toţi roboţii din experiment. Aria acoperită de reţeaua radio este aproximativ 100 m^2.

Senzorii de proximitate, pentru **Khepera,** permit detecţia obstacolelor la o distanţă maximă de 6 cm în toate direcţiile, astfel că, informaţiile despre forma şi mărimea obiectelor întâlnite sunt vagi şi sunt greu deosebite de obstacole.

O soluţie găsită pentru îmbunătăţirea capabilităţii de discriminare a roboţilor, fără creşterea complexităţii hard, a fost dezvoltarea de "seminţe active" care pot fi uşor recunoscute de către senzorii de proximitate. Când senzorii robotului sunt activaţi de către un obiect, robotul începe procedura de discriminare, dacă acesta se află în faţa unui obstacol mare (perete, alt robot ori un grup de "seminţe") obiectul este considerat ca fiind obstacol şi este evitat, dacă nu, este identificat ca fiind "sămânţă".

5.4. Algoritm CNN pentru coordonarea deplasării unei colectivităţi de roboţi mobili

Sunt multe situaţii în practică când roboţii mobili, aparţinând unei colectivităţi, trebuie să se adune la un loc pentru a putea realiza împreună o anumită operaţie sau trebuie să se deplaseze într-o anumită formaţie.

Mare parte din literatura de specialitate cu privire la algoritmii de comandă distribuită pentru roboţii mobili autonomi a fost concentrată pe două sarcini de bază şi anume adunarea acestora (*gathering*) şi convergenţa (*convergence*) [144]. Prima sarcină se referă la adunarea roboţilor mobili într-un singur punct, în cadrul unui timp finit, iar convergenţa este sarcina prin care roboţii trebuie doar să conveargă spre un singur punct şi nu neapărat să-l atingă. Mai precis, pentru orice $\varepsilon > 0$ există un timp t_ε în care toţi roboţii trebuie să se situeze la o distanţă de cel mult ε unul de altul.

Tot în [144] se prezintă un algoritm de convergenţă a roboţilor mobili spre un punct care reprezintă centrul de simetrie al tuturor poziţiilor ocupate de aceştia. Astfel, dacă se notează cu $\bar{r}_i(t)$ locaţia robotului i la momentul t, centrul de simetrie este dat de relaţia:

$$\bar{c}(t) = \frac{1}{N}\sum_{i=1}^{N}\bar{r}_i(t) \qquad (5.1)$$

Se presupune că centrul de simetrie calculat la momentul $t' < t$, $\bar{c}(t')$ este $\bar{c}(t') = \bar{c}_i(t)$, astfel că robotul i va calcula această locaţie în propriul său sistem de coordonate.

Se consideră în acest capitol, situaţia în care un set de roboţi mobili omogeni sunt dispersaţi pe o anumită arie şi se doreşte ca roboţii care au poziţii extreme, în raport cu poziţiile celorlalţi roboţi, să se deplaseze spre roboţii cu poziţii centrale astfel încât această operaţie să se facă printr-o deplasare cumulată minimă a roboţilor. Astfel, dacă există un robot cu o poziţie centrală, acesta va sta pe loc iar ceilalţi roboţi se vor deplasa spre acesta. Pot exista şi situaţii când doi sau chiar trei roboţi vor rămâne pe loc şi restul roboţilor se vor aduna în jurul lor. După deplasarea roboţilor spre cei cu poziţii centrale, vor fi reevaluate poziţiile tuturor roboţilor şi în final toţi roboţii vor fi adunaţi în jurul a unu sau maxim doi roboţi.

De asemenea, algoritmul descris în continuare poate fi aplicat şi în cazul în care roboţii trebuie să se deplaseze într-o formaţie, astfel încât aceştia să marcheze vârfurile unei figuri geometrice regulate plane (triunghi, pătrat, hexagon, etc.). În acest caz, pot fi identificaţi roboţii care au tendinţa de a se îndepărta sau apropia de centrul de simetrie al formaţiei, putând fi corectate traiectoriile acestora.

Algoritmul are o concepţie originală şi se bazează pe procesarea cu reţele neuronale celulare [49],[50], a imaginilor obţinute cu o cameră video, în care pot fi identificate poziţiile roboţilor. Aceste reţele, au avantajul că prezintă un timp de procesare a imaginilor relativ mic şi, faţă de alte reţele neuronale, pot fi implementate hard, în tehnologie VLSI, pe un singur cip [68].

Utilizarea reţelelor neuronale artificiale şi a reţelelor neuronale celulare pentru ghidarea roboţilor mobili, aparţinând unei colectivităţi, a fost studiată şi în [145],[146],[147],[148].

În cazul unei colectivităţi compusă din *n* roboţi mobili (R1, R2, ..., Rn), dispersaţi într-un mediu de lucru, se doreşte identificarea acelor roboţi care au poziţii extreme în raport cu ceilalţi şi apoi să fie stabilite pas cu pas traiectoriile pentru aceştia astfel încât să se deplaseze spre roboţii cu poziţii centrale.

După achiziţia imaginii mediului, vor fi identificate poziţiile roboţilor şi după unele procesări simple se vor obţine o serie de imagini, fiecare din acestea indicând, printr-un singur pixel de culoare neagră, poziţia unuia dintre roboţi (Figura 5.4).

În acest algoritm se consideră rezoluţia imaginii mediului ca fiind egală cu dimensiunea reţelei neuronale celulare folosită pentru procesare. De asemenea, camera video, se presupune că poate achiziţiona imaginea întregului mediu de lucru al roboţilor şi are aceeaşi poziţie, orientare şi reglaje în cazul captării imaginilor de tipul celor prezentate în Figura 5.4.

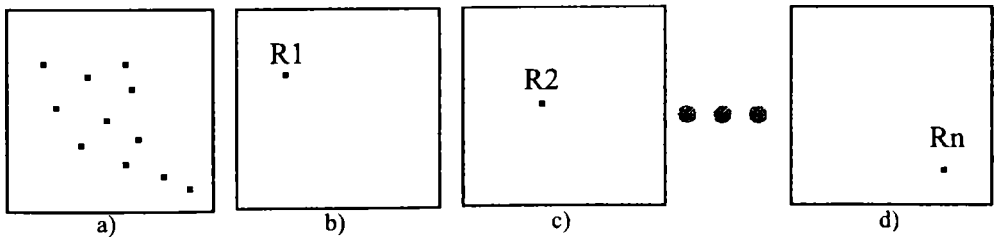


Figura 5.4. Imagini reprezentând poziţiile roboţilor; a) imaginile suprapuse ale poziţiilor roboţilor, b) poziţia robotului R1, c) poziţia robotului R2, d) poziţia robotului Rn.

5.4.1. Estimarea distanţei dintre roboţi

Stabilirea poziţiei roboţilor, în cadrul colectivităţii, se poate face prin determinarea distanţelor dintre fiecare robot şi toţi ceilalţi roboţi ce aparţin grupului. În acest fel, va fi cunoscută distribuţia spaţială a colectivităţii de roboţi şi pot fi identificaţi acei roboţi ce au poziţii centrale sau roboţii care din diverse motive au rămas în afara formaţiei şi care trebuie dirijaţi astfel încât să se deplaseze spre ceilalţi roboţi din grup.

Estimarea distanţei dintre doi roboţi s-a făcut prin determinarea vecinătăţii cu rază minimă, a pixelului ce indică poziţia unui robot, care include pixelul corespunzător poziţiei celuilalt robot.

Dacă se presupune situaţia generalizată cu o colectivitate de n roboţi mobili, având fiecare coordonatele poziţiei în imagine date de perechea (i, j), (R1(i_1, j_1), R2(i_2, j_2) ... Rn(i_n, j_n)), vecinătăţile de rază r pentru fiecare din ei sunt date de relaţiile 5.2:

$$V_{R1,r}(i_1,j_1)=\left\{C(k_1,l_1)\middle|\max\left\{|k_1-i_1|,|l_1-j_1|\right\}\le r\right\}$$

$$V_{R2,r}(i_2,j_2)=\left\{C(k_2,l_2)\middle|\max\left\{|k_2-i_2|,|l_2-j_2|\right\}\le r\right\} \qquad (5.2)$$

$$V_{Rn,r}(i_n,j_n)=\left\{C(k_n,l_n)\middle|\max\left\{|k_n-i_n|,|l_n-j_n|\right\}\le r\right\}$$

Dacă se presupune că vor fi determinate mai întâi distanţele dintre robotul R1 şi ceilalţi roboţi, va fi generată vecinătatea de rază $r=1$, a pixelului reprezentând poziţia lui R1, apoi aceasta va fi incrementată ($r=r+1$) până când vor fi acoperiţi rând pe rând toţi pixelii prin care sunt reprezentaţi ceilalţi roboţi ($r=r_1$). Distanţa măsurată, în număr de pixeli, r_{1m} dintre robotul R1 şi un robot Rm este dedusă astfel:

$$\text{dacă } (i_m,j_m)\subset V_{R1,r}(i_1,j_1) \text{ şi } (i_m,j_m)\notin V_{R1,r-1}(i_1,j_1) \text{ atunci } r_{1m}=r$$
$$\forall \text{ar fi } m\in[1,n],\ m\neq 1 \text{ şi } r\in[1,r_1] \qquad (5.3)$$

În mod similar vor fi determinate distanţele, măsurate în număr de pixeli, dintre robotul R1 şi toţi ceilalţi roboţi din grup: $r_{1m}\ \forall \text{ar fi } m\in[1,n],\ m\neq 1$.

În continuare se va trece la generarea vecinătăţilor de rază r pentru pixelul reprezentând poziţia lui R2, începând cu $r=1$. După generarea fiecărei vecinătăţi de rază r, $r\in[1,r_2]$ a robotului R2 se va verifica dacă acestea includ pixelii care reprezintă poziţiile celorlalţi roboţi pe baza relaţiilor 5.4:

$$\text{dacă } (i_m,j_m)\subset V_{R2,r}(i_2,j_2) \text{ şi } (i_m,j_m)\notin V_{R2,r-1}(i_2,j_2) \text{ atunci } r_{2m}=r$$
$$\forall \text{ar fi } m\in[1,n],\ m\neq 2 \text{ şi } r\in[1,r_2] \qquad (5.4)$$

După determinarea tuturor parametrilor r_{2m} se va trece la generarea vecinătăţii robotului R3 ş.a.m.d. Spre exemplu, pentru un robot oarecare Rp, din cadrul grupului, determinarea distanţelor faţă de un robot Rm din grup se va face conform relaţiilor 5.5:

$$\text{dacă } (i_m, j_m) \subset V_{Rp,r}(i_p, j_p) \text{ şi } (i_m, j_m) \notin V_{Rp,r-1}(i_p, j_p) \text{ atunci } r_{pm} = r$$
$$\forall \text{ ar fi } m \in [1, n],\ p \in [1, n],\ m \neq p,\ r \in [1, r_p] \tag{5.5}$$

Pentru simplificarea algoritmului, se va lua în considerare proprietatea de reciprocitate şi anume:

$$r_{pm} = r_{mp} \quad \forall \text{ ar fi } \quad p \in [1, n] \text{ şi } m \in [1, n],\ p \neq m. \tag{5.6}$$

Imaginea vecinătăţii unui robot se poate obţine cu ajutorul reţelelor neuronale celulare prin aplicarea template-ului DILATION [69]. Astfel, aplicând acest template, prezentat în relaţia 5.7, asupra imaginii cu un singur pixel negru, ce reprezintă poziţia robotului, se va obţine imaginea vecinătăţii de rază r =1, iar prin aplicarea repetată a acestuia asupra imaginii rezultate se vor obţine vecinătăţi cu raze din ce în ce mai mari.

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad z = 8 \tag{5.7}$$

Pe intrarea şi pe starea reţelei neuronale celulare se va aplica imaginea cu un singur pixel negru, iar după aplicarea template-ului DILATION, pe ieşirea reţelei neuronale se va obţine imaginea corespunzătoare vecinătăţii robotului.

Pentru o situaţie concretă, cu patru roboţi mobili, se prezintă în Figura 5.5 modalitatea de obţinere succesivă a vecinătăţilor robotului R1, până când acestea acoperă treptat pixelii corespunzători poziţiei roboţilor R2, R3 şi R4.

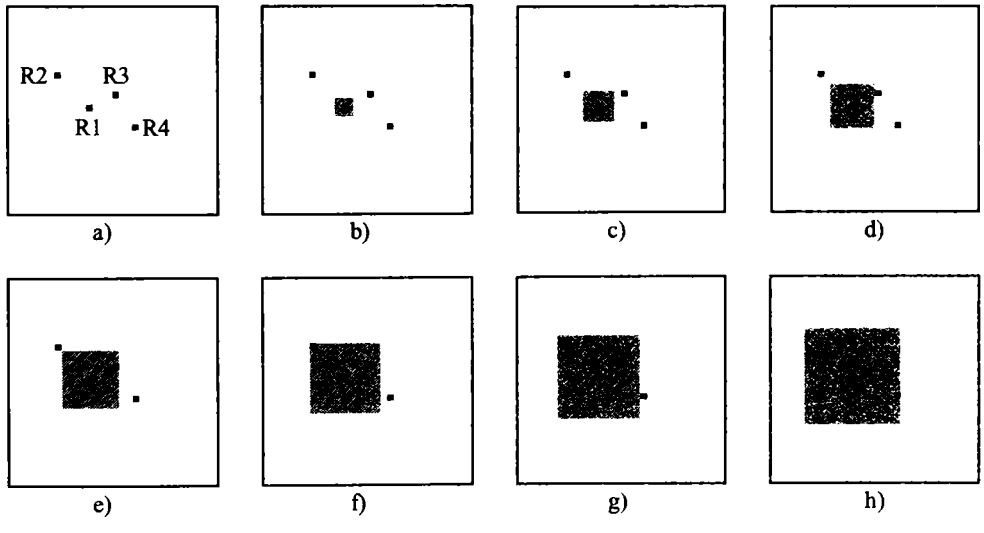


Figura 5.5. Generarea vecinătăţilor corespunzătoare robotului R1:
a) poziţiile roboţilor R1..R4; b) vecinătatea de rază r =1; c) r =2;
d) r =3; e) r =4 (acoperirea pixelului corespunzător robotului R3); f) r =5
(acoperirea şi a pixelului corespunzător robotului R2); g) r =6; h) r =7
(acoperirea şi a pixelului corespunzător robotului R4).

În mod similar vor fi generate vecinătăţile pentru ceilalţi roboţi din grup: R2, R3 şi R4.

Ordinea estimării distanţelor dintre unul din roboţi şi ceilalţi roboţi s-a făcut în două moduri şi anume:

- într-o ordine prestabilită;
- în ordinea crescătoare a distanţelor.

5.4.1.1. Estimarea distanţelor într-o ordine prestabilită

Organigrama pentru estimarea distanţelor dintre fiecare din cei patru roboţi şi ceilalţi roboţi, într-o ordine prestabilită, se prezintă în Figura 5.6.

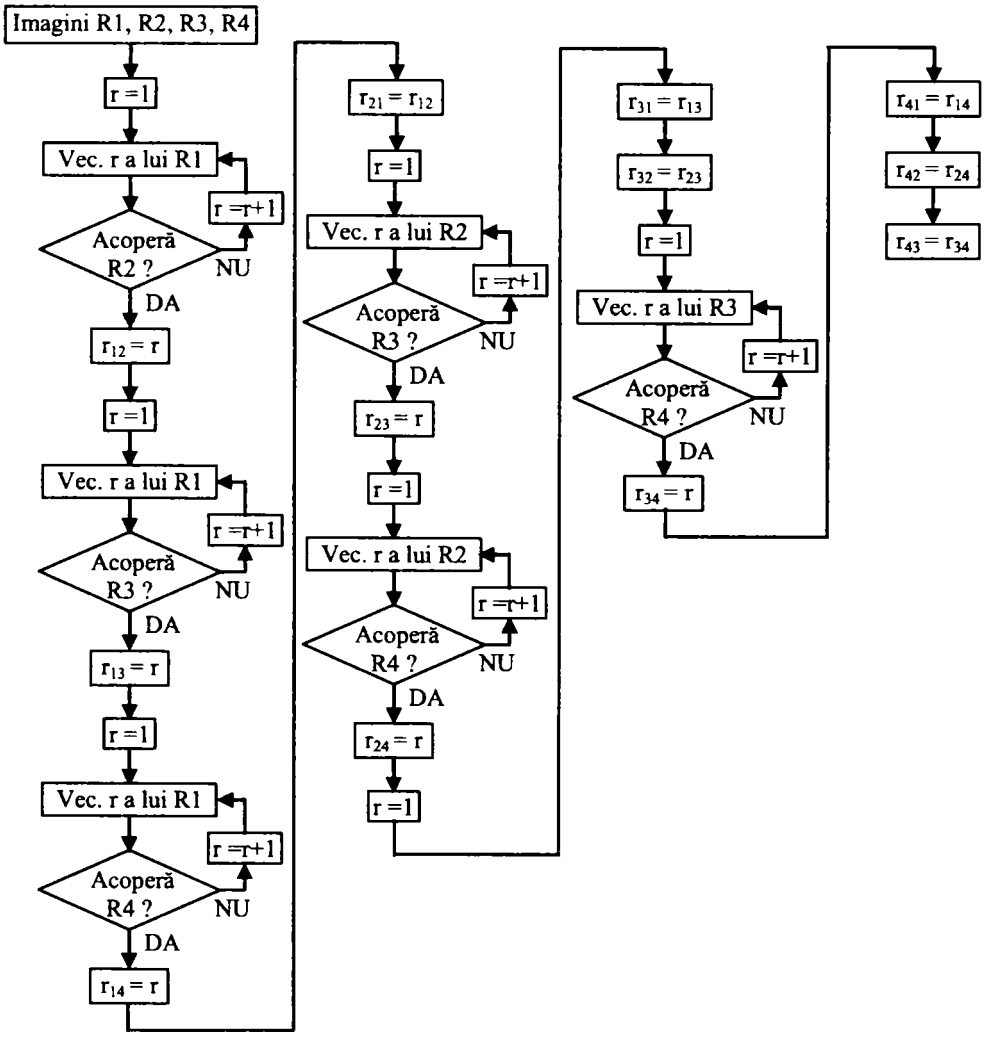


Figura 5.6. Organigrama pentru estimarea distanţelor dintre roboţi într-o ordine prestabilită.

Se generează, prin procesări CNN vecinătatea de rază r =1 a lui R1. Se verifică dacă aceasta include pixelul ce reprezintă poziţia pe care e situat robotul R2. Pot exista două situaţii:

- dacă $(i_2, j_2) \subset V_{R1,1}(i_1, j_1)$ atunci $r_{12} = 1$;
- dacă $(i_2, j_2) \not\subset V_{R1,1}(i_1, j_1)$ atunci se va incrementa valoarea lui r ($r = r + 1$), şi se va genera vecinătatea lui R1 cu noua valoare a razei r.

Dacă se întâmplă a doua situaţie, procesul se va repeta până când noua vecinătate va include pixelul corespunzător robotului R2. În final parametrul r_{12} va avea valoarea atribuită ultima dată lui r.

În continuare se va proceda la fel pentru estimarea distanţelor dintre R1 şi R3 (r_{13}) respectiv R1 şi R4 (r_{14}).

Se vor utiliza apoi imaginile roboţilor R2, R3 şi R4, pentru generarea vecinătăţilor şi ţinând cont de relaţia 5.5 vor fi determinate:

- r_{23} şi r_{24} în cazul robotului R2, ($r_{21} = r_{12}$);
- r_{34} în cazul robotului R3, ($r_{31} = r_{13}$; $r_{32} = r_{23}$).

În cazul robotului R4, toate distanţele până la ceilalţi roboţi vor fi deduse pe baza relaţiei 5.6, astfel: $r_{41} = r_{14}$, $r_{42} = r_{24}$ şi $r_{43} = r_{34}$.

5.4.1.2. Estimarea distanţelor în ordinea lor crescătoare

În cazul acestei metode de estimare a distanţelor, va fi determinată mai întâi distanţa până la robotul cel mai apropiat apoi până la următorul robot situat cel mai aproape şi tot aşa până când sunt determinate distanţele până la toţi roboţii din colectivitate.

În Figura 5.7 se prezintă modul de estimare a distanţelor dintre robotul R1 şi ceilalţi roboţi, în ordinea lor crescătoare.

Se verifică dacă vecinătatea de rază r =1 a lui R1 include unul sau mai mulţi din pixelii care indică poziţiile pe care sunt situaţi ceilalţi roboţi (R2, R3 şi R4). În funcţie de situaţiile posibile pot fi determinate valorile pentru r_{12}, r_{13} sau r_{14} astfel:

- dacă $(i_2, j_2) \subset V_{R1,1}(i_1, j_1)$ atunci $r_{12} = 1$;
- dacă $(i_3, j_3) \subset V_{R1,1}(i_1, j_1)$ atunci $r_{13} = 1$;
- dacă $(i_4, j_4) \subset V_{R1,1}(i_1, j_1)$ atunci $r_{14} = 1$.

Dacă au fost determinate toate valorile (r_{12}, r_{13} şi r_{14}), atunci se va trece la robotul R2 şi se va proceda similar. Dacă nu, va fi incrementată valoarea lui r ($r = r + 1$) şi va fi generată imaginea vecinătăţii de rază r =2 a robotului R1. Astfel pot exista situaţiile:

- dacă $(i_2, j_2) \not\subset V_{R1,1}(i_1, j_1)$ dar $(i_2, j_2) \subset V_{R1,2}(i_1, j_1)$ atunci $r_{12} = 2$;
- dacă $(i_3, j_3) \not\subset V_{R1,1}(i_1, j_1)$ dar $(i_3, j_3) \subset V_{R1,2}(i_1, j_1)$ atunci $r_{13} = 2$;
- dacă $(i_4, j_4) \not\subset V_{R1,1}(i_1, j_1)$ dar $(i_4, j_4) \subset V_{R1,2}(i_1, j_1)$ atunci $r_{14} = 2$.

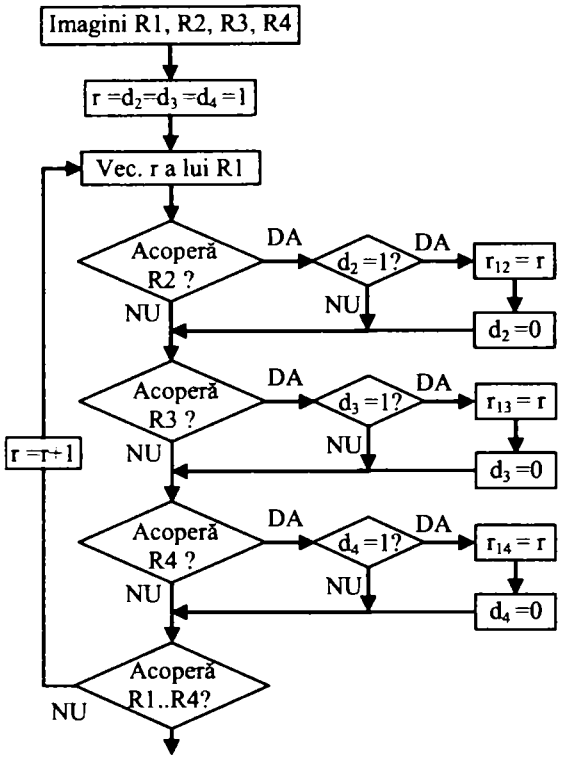


Figura 5.7. Organigrama pentru estimarea distanţelor în ordine crescătoare dintre robotul R1 şi ceilalţi roboţi (R2, R3 şi R4).

Procesul de incrementare a razei r va continua până se ajunge în situaţia în care au fost determinaţi toţi parametrii r_{12}, r_{13} şi r_{14}.

În continuare, se va proceda similar pentru roboţii R2, R3 şi R4 dar se va ţine cont, şi aici, de relaţia 5.6.

5.4.2. Determinarea robotului sau roboţilor cu poziţii centrale şi extreme

Pentru fiecare robot, valorile obţinute pentru estimarea distanţelor faţă de ceilalţi roboţi vor fi cumulate şi astfel vor fi obţinute valorile N_1, N_2, N_3 şi N_4. Pe baza acestor valori se poate aprecia cât de aproape este situat robotul faţă de ceilalţi roboţi. Valorile acestor indicatori sunt date de relaţiile 5.8:

$$\begin{aligned} N_1 &= r_{12} + r_{13} + r_{14}; \\ N_2 &= r_{21} + r_{23} + r_{24}; \\ N_3 &= r_{31} + r_{32} + r_{34}; \\ N_4 &= r_{41} + r_{42} + r_{43}. \end{aligned} \qquad (5.8)$$

Prin compararea acestor valori se poate determina care robot sau roboţi au poziţii centrale comparativ cu poziţiile celorlalţi roboţi. Modalitatea de determinare a robotului sau roboţilor centrali, în cadrul algoritmului, se prezintă în organigrama din Figura 5.8.

Pot astfel exista mai multe situaţii şi anume:

- nu există nici un robot cu poziţie centrală ($N_1 = N_2 = N_3 = N_4$);
- un singur robot (ex. R1) are poziţie centrală ($N_1 < \min(N_2, N_3, N_4)$);
- doi roboţi (ex. R1, R2) au poziţii centrale ($N_1 = N_2 < \min(N_3, N_4)$);
- trei roboţi (ex. R1, R2, R3) au poziţii centrale ($N_1 = N_2 = N_3 < N_4$).

Algoritmul a fost realizat şi în varianta în care prin compararea valorilor lui N_1, N_2, N_3 şi N_4, să fie determinat acel robot sau roboţi cu poziţii extreme în raport cu ceilalţi roboţi din grup. Aceştia trebuie grupaţi în jurul celorlalţi roboţi, adică vor trebui să se deplaseze spre robotul sau roboţii centrali sau în primă fază spre robotul cel mai apropiat din grup care nu are atributul de robot extremă. Modul concret de determinare a robotului sau roboţilor cu poziţii extreme în cazul unei colectivităţi de patru roboţi se prezintă în organigrama din Figura 5.9.

În funcţie de distribuţia la un moment dat a colectivităţii de roboţi pot exista mai multe situaţii şi anume:

- nu există nici un robot cu poziţie extremă ($N_1 = N_2 = N_3 = N_4$);
- un singur robot (ex. R1) are poziţie extremă ($N_1 > \max(N_2, N_3, N_4)$);
- doi roboţi (ex. R1, R2) au poziţii extreme ($N_1 = N_2 > \max(N_3, N_4)$);
- trei roboţi (ex. R1, R2, R3) au poziţii extreme ($N_1 = N_2 = N_3 > N_4$).

5.4.3. Adunarea roboţilor în jurul celor cu poziţii centrale

Adunarea roboţilor în jurul roboţilor cu poziţii centrale, astfel încât să fie minimizată lungimea deplasărilor cumulate ale roboţilor mobili $tr_{totală}$ dată de relaţia 5.9, se poate obţine prin deplasarea roboţilor extremă spre roboţii mobili având poziţii centrale.

$$tr_{totală} = tr_{R1} + tr_{R2} + tr_{R3} + tr_{R4} \tag{5.9}$$

Dintre situaţiile posibile, în cazul unei colectivităţi de patru roboţi, sunt prezentate trei situaţii reprezentative şi anume:

- un robot central şi un robot extremă;
- doi roboţi centrali şi un robot extremă;
- un robot central şi doi roboţi extremă.

În situaţia cu un robot central şi un robot extremă, robotul din poziţia extremă se va deplasa pas cu pas până va intra în vecinătatea de rază r =1 a robotului situat în poziţia centrală. Dacă se presupune că robotul R1 este robotul cu poziţie centrală în cadrul grupului iar R2 este robotul având o poziţie extremă, organigrama algoritmului de grupare pentru această situaţie este prezentată în Figura 5.10.

Se atribuie lui r valoarea $r = r_{12}$, valoare obţinută prin aplicarea metodei descrise mai sus, care va fi apoi decrementată cu o unitate. Dacă $r \neq 0$ atunci va fi generată vecinătatea de rază r a lui R1.

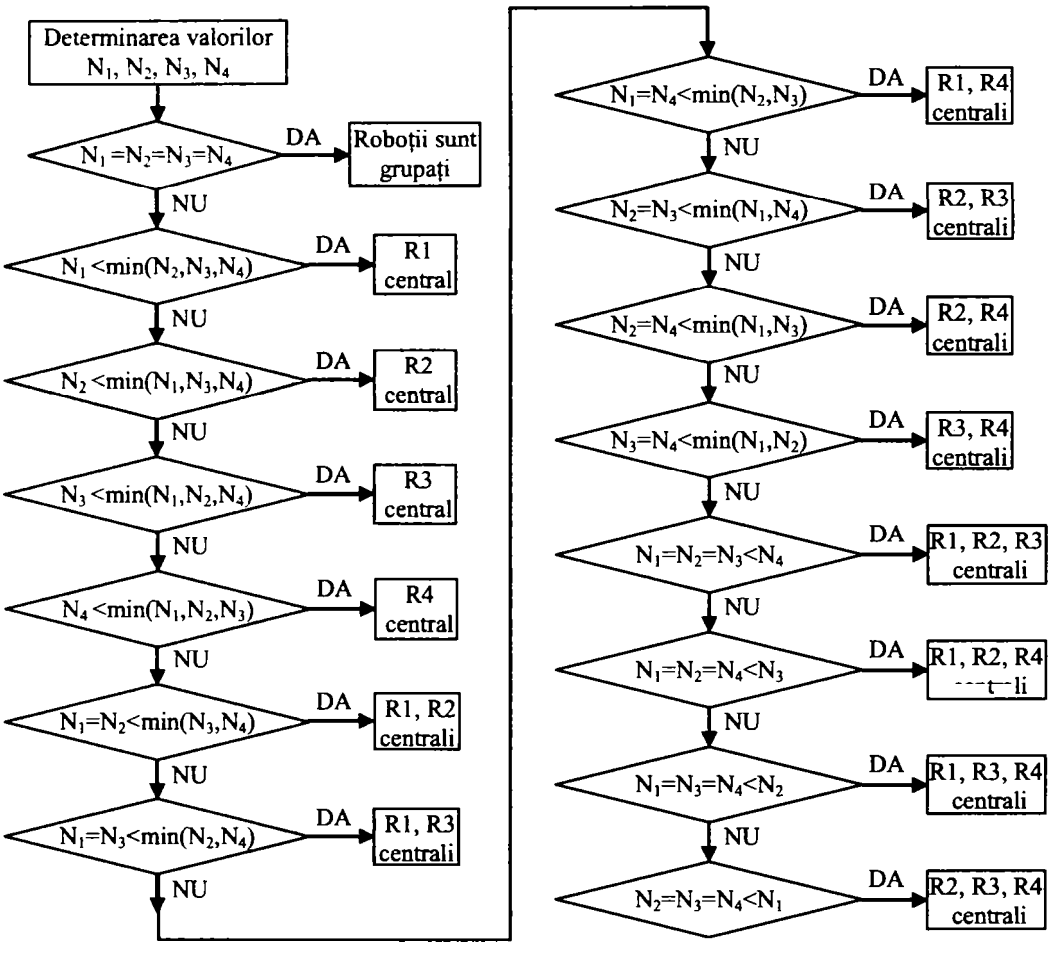


Figura 5.8. Modul de determinarea a robotului sau roboţilor cu poziţii centrale.

Robotul R2 se va deplasa de fiecare dată cu un pas (pixel), astfel că după verificarea poziţiilor din jurul său va intra în vecinătatea lui R1. În continuare, va fi din nou decrementată cu o unitate valoarea lui r şi procesul va continua până când robotul R2 se va situa în vecinătatea de rază r =1 a lui R1. În acelaşi timp r va fi r = 0 iar algoritmul de grupare se încheie.

După deplasarea robotului R2 cu un anumit număr de paşi se va aplica din nou algoritmul pentru estimarea distanţelor dintre roboţi deoarece roboţii R3 şi/sau R4 pot deveni la rândul lor roboţi extremă. Timpul necesar grupării se reduce mult, dacă doi sau mai mulţi roboţi extremă se deplasează simultan spre robotul central.

Dacă doi roboţi au poziţii centrale şi un singur robot are poziţie extremă, acesta din urmă va trebui să se deplaseze spre robotul central cel mai apropiat de el. Astfel, dacă se presupune că roboţii R1 şi R2 sunt roboţi centrali iar R3 este robotul extremă care trebuie grupat, acesta va alege să se deplaseze spre unul din aceştia prin compararea valorilor r_{31} şi r_{32}.

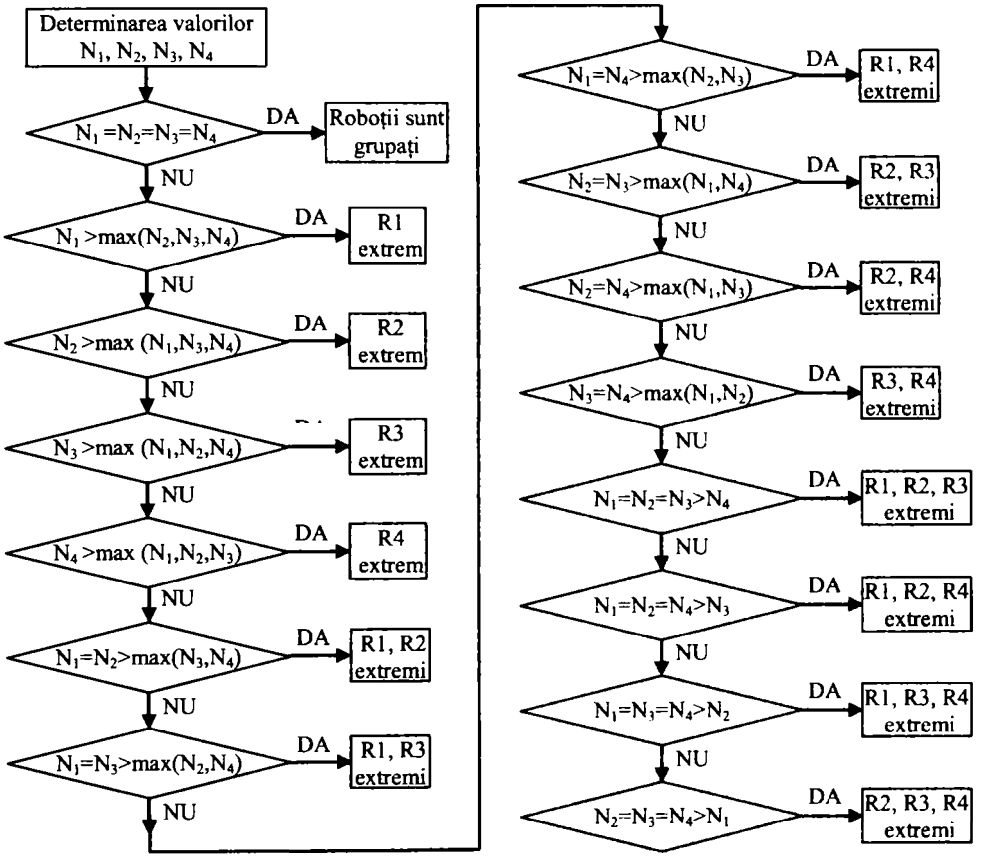


Figura 5.9. Situaţiile posibile în cazul determinării robotului sau roboţilor cu poziţii extreme.

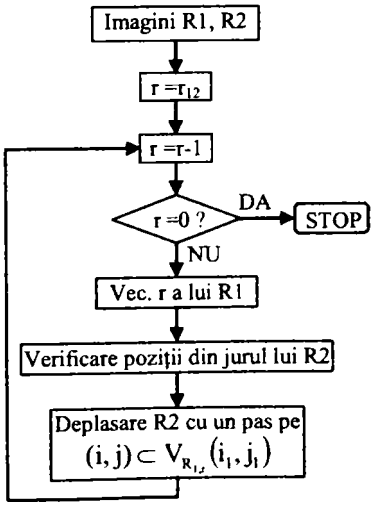


Figura 5.10. Organigrama pentru gruparea lui R2 în jurul lui R1.

Astfel, pot exista situaţiile:

- dacă $r_{31} < r_{32}$ robotul R3 se va deplasa spre R1;
- dacă $r_{31} > r_{32}$ robotul R3 se va deplasa spre R2;
- dacă $r_{31} = r_{32}$ robotul R3 se va deplasa spre oricare din roboţii R1 sau R2.

Odată ales robotul spre care trebuie să se deplaseze R3, algoritmul va continua conform organigramei din Figura 5.10.

Dacă se aplică din nou algoritmul de estimare a distanţelor, atunci robotul mobil care se află iniţial la distanţă mai mare de R3, respectiv robotul care nu a fost ales ca ţintă pentru deplasarea acestuia, poate deveni la rândul său robot extremă şi se va deplasa şi el spre robotul ales ca şi robot ţintă pentru R3, tot pe baza unei organigrame ca şi cea prezentată în Figura 5.10.

Dacă un singur robot are poziţie centrală şi sunt doi roboţi cu poziţii extreme, aceştia doi se vor deplasa spre robotul cu poziţie centrală. Deplasarea acestora se poate face rând pe rând conform organigramei din Figura 5.10, situaţie în care unul din roboţi va aştepta până când celălalt va ajunge în vecinătatea de rază r =1 a robotului central, urmând apoi să facă şi el acelaşi lucru.

Se poate însă reduce timpul necesar grupării, dacă cei doi roboţi extremă se vor deplasa simultan sau cvasimultan pas cu pas spre robotul central. Dacă se consideră, spre exemplu, situaţia concretă cu R1 robot central şi R2, R3 roboţi extremă atunci deplasarea acestora din urmă spre robotul central se va face conform organigramei din Figura 5.11.

În principiu, va fi generată vecinătatea lui R1 având raza cea mai mare dintre r_{12} şi r_{13} astfel încât vor fi acoperiţi ambii pixeli care indică poziţiile roboţilor R2 şi R3. În continuare, va fi decrementată valoarea lui r şi noua vecinătate, astfel generată, va lăsa afară unul din pixeli, urmând ca acest robot să se deplaseze cu un pas în interiorul vecinătăţii. Procesul va continua în acest fel, până când şi celălalt pixel ce reprezintă poziţia celuilalt robot va rămâne în afara vecinătăţii lui R1, iar din acel moment deplasarea se va face alternativ, pas cu pas, până când R2 şi R3 vor ajunge amândoi în vecinătatea de rază r =1 a lui R1.

5.4.4 Rezultatele simulărilor

Algoritmii de grupare a roboţilor mobili prezentaţi în lucrare au fost testaţi utilizând mediul de simulare **Cadetwin** [69], prezentat în **Capitolul 3**. Rezultatele simulărilor au fost obţinute pe baza unor programe scrise în limbaj de asamblare **AMC** [77] (vezi Anexele A5 şi A6). Imaginile utilizate pentru testare au dimensiunea 32×32 pixeli.

În Figura 5.12 se prezintă un exemplu de determinare a roboţilor centrali şi a roboţilor extremă utilizând algoritmul prezentat în lucrare. Se presupune că o colectivitate de patru roboţi mobili R1...R4 are dispunerea geometrică ca în Figura 5.12a. În Figura 5.12b se arată vecinătăţile pixelului care indică poziţia robotului R1, astfel încât acestea să acopere pixelii ce indică poziţia roboţilor R2, R3 şi R4. În Figura 5.12c se arată vecinătăţile pixelului care indică poziţia robotului R2, astfel încât să acopere pixelii ce indică poziţia roboţilor R3 şi R4. În Figura 6d se arată vecinătăţile pixelului care indică poziţia robotului R3, astfel încât să acopere pixelul ce indică poziţia robotului R4.

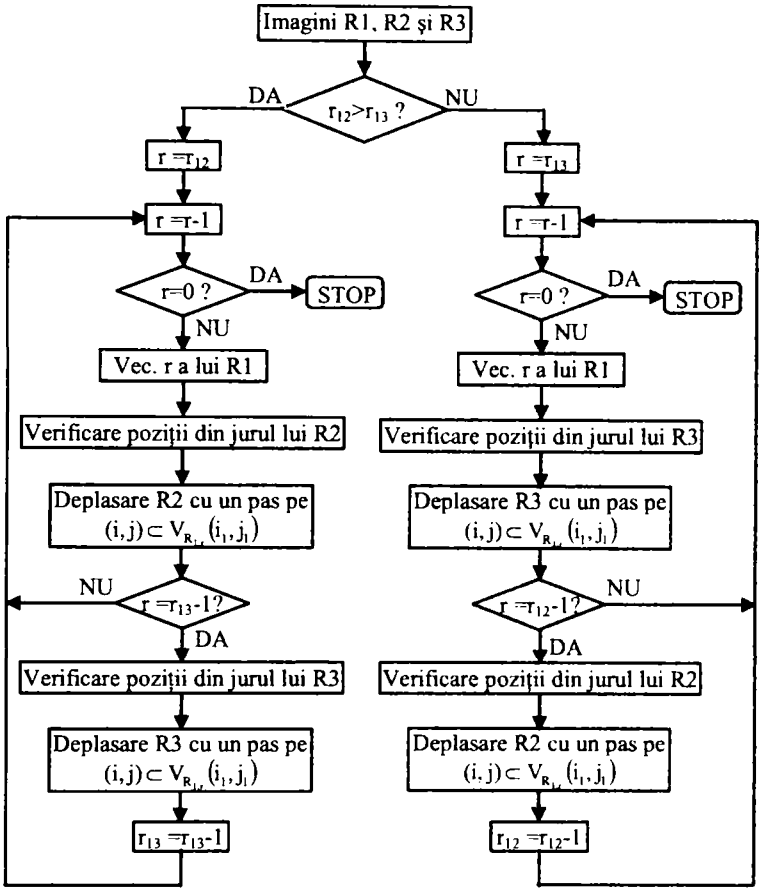


Figura 5.11. Organigrama algoritmului pentru gruparea roboţilor în cazul existenţei unui robot central (R1) şi a doi roboţi extremă (R2) şi (R3).

După determinarea razelor pentru vecinătăţile prezentate mai sus se obţine în final imaginea pixelilor ce reprezintă roboţii cu poziţii centrale (Figura 5.12e) şi respectiv cu poziţii extreme (Figura 5.12f).

În Figura 5.13 se prezintă un exemplu de grupare a doi roboţi extremă în jurul unui robot cu poziţie centrală. În Figura 5.13a sunt reprezentate prin câte un pixel negru poziţiile iniţiale ale celor trei roboţi, R1 fiind robotul cu poziţie centrală iar roboţii R2 şi R3 cu poziţii extreme. Într-o primă fază se va deplasa robotul R2 spre robotul R1 atâta timp cât el este cel mai îndepărtat de R1 (Figura 5.13b). Deoarece robotul R3 va deveni robotul cu poziţia cea mai îndepărtată de robotul central se va deplasa în mod similar ca şi robotul R2. În continuare roboţii R2 şi R3 vor face alternativ câte un pas (pixel) până când ambii vor ajunge în vecinătatea de rază $r = 1$ a robotului central R1 (Figura 5.13c).

Timpul total necesar pentru rularea algoritmului de grupare a roboţilor depinde de varianta de estimare a distanţelor dintre roboţi. Cel mai bun timp s-a obţinut în cazul în care estimarea distanţelor dintre roboţi s-a făcut în ordinea crescătoare a lor. Astfel, în această variantă, acest timp a fost redus cu aproximativ 25%, faţă de varianta cu estimarea distanţelor într-o ordine prestabilită.

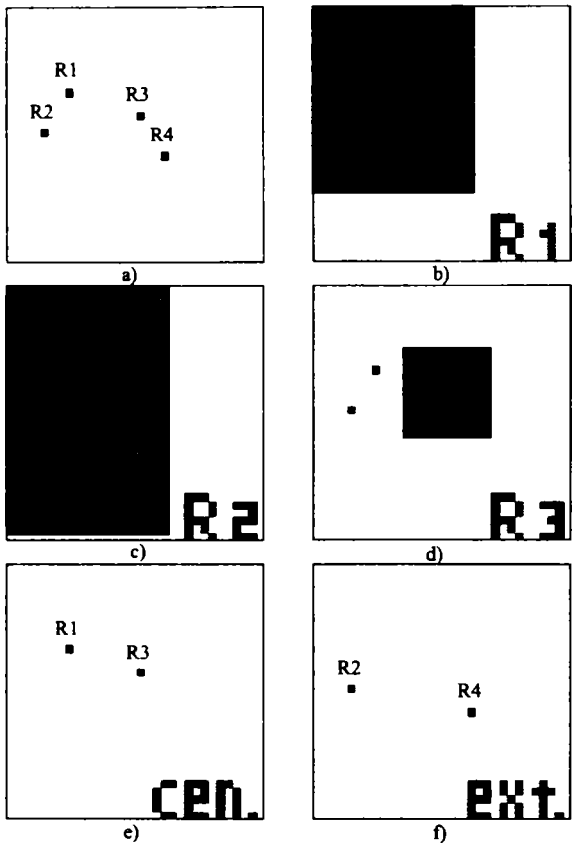


Figura 5.12. Exemplu de determinare a roboţilor cu poziţie centrală şi poziţie extremă dintr-un grup; a) poziţiile roboţilor, b) vecinătăţile lui R1, c) vecinătăţile lui R2, d) vecinătăţile lui R3, e) roboţii cu poziţie centrală, f) roboţii cu poziţie extremă.

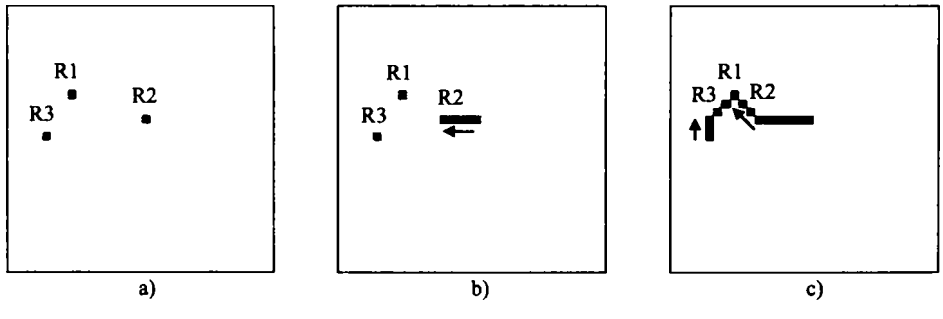


Figura 5.13. Exemplu de grupare a doi roboţi mobili în jurul unui robot cu poziţie centrală; a) poziţiile iniţiale ale roboţilor, b) deplasarea lui R2, c) deplasarea alternativă a lui R2 şi R3.

De asemenea, ţinându-se cont de relaţia 5.6, s-a obţinut o reducere a timpului de procesare cu 40%.

5.5. Concluzii

În anii de început ai roboţilor mobili, funcţionarea în mod corect şi constant a unui robot era considerată la acea dată, o sarcină suficient de grea, ceea ce a condus la concluzia că ideea folosirii mai multor roboţi care să colaboreze, este nerealistă.

Odată cu avansarea tehnicii, în special în domeniul senzorilor, al procesării informaţiei şi al structurilor de locomoţie, sistemele de roboţi încep să constituie o realitate şi chiar o necesitate. Apar treptat sisteme de roboţi în care roboţii iau în considerare, pentru realizarea operaţiunilor, atât percepţia proprie dar şi informaţiile obţinute prin interacţiunea cu ceilalţi roboţi din sistem.

Ca şi în cazul roboţilor mobili, studiul colectivităţilor de roboţi este un domeniu de cercetare inter-disciplinar care combină metode din domeniul tehnic dar şi din biologie sau chiar filozofie.

S-a observat că multe operaţii pot fi executate mult mai eficient cu ajutorul mai multor roboţi mai ales dacă aceştia cooperează. În consecinţă, o schimbare majoră în domeniul roboţilor mobili, ce a luat amploare în ultimi 10-15 ani, a fost folosirea mai multor roboţi în acelaşi timp pentru execuţia unor sarcini complexe.

Se preconizează că utilizarea roboţilor multipli va face să crească siguranţa în funcţionarea sistemului în timp ce scade complexitatea sarcinilor pentru un robot şi de asemenea, timpul de execuţie total al unei sarcini complexe.

Principalele avantaje ce derivă din utilizarea sistemele multi-robot, faţă de folosirea unui singur robot, sunt: posibilitatea realizării unor sarcini complexe ce nu pot fi executate de către un singur robot, de multe ori e mai economic să se construiască câţiva roboţi simpli şi ieftini decât unul sofisticat şi scump, obţinerea unei mai bune distribuţii spaţiale, posibilitatea efectuării mai multor operaţii în paralel şi dacă un robot se defectează atunci altul sau alţii pot să-i ia locul fără a fi afectată funcţionalitatea sistemului.

Utilizarea unei echipe de roboţi presupune şi apariţia unor probleme noi, ce trebuie luate în considerare, cum ar fi: modul de formulare, descriere şi alocare a sarcinilor, modul de folosire a resursele primite de către roboţi, tipul de comunicare folosit şi cum se rezolvă conflictele apărute între roboţi.

Cele mai multe metode de comandă a sistemelor multi-robot subliniază importanţa pe care o are modul de comunicaţie robot-robot, comunicaţie folosită pentru a schimba între aceştia mesaje de comandă şi date senzoriale locale. Adesea se presupune că, comunicarea explicită oferă cele mai multe avantaje. Pentru a alege un anumit tip de comunicare trebuie însă ţinut cont de câţiva factori cum ar fi: costul, siguranţa în funcţionare, timpul de execuţie al sarcinii, etc. În cadrul experimentelor există, de obicei, un set de sarcini ce trebuie realizat şi se urmăreşte timpul în care se realizează acest set cu diverse forme de comunicare între roboţi.

S-a stabilit că în unele situaţii comunicarea explicită poate lipsi deoarece echipele de roboţi se pot descurca bine fără a folosi o astfel de comunicare. Totuşi, aceasta îmbunătăţeşte semnificativ performanţele în cazul sarcinilor ce se desfăşoară într-un mediu ce nu favorizează comunicarea implicită. Cu alte cuvinte, dacă costurile de realizare a acţiunilor redundante în prezenţa doar a comunicării implicite este mare, atunci comunicarea explicită este benefică.

O definiţie interesantă a cooperării dintre roboţi este legată de diferenţele ce există între un comportament colectiv ne-cooperant şi unul cooperant, respectiv de modul în care o sarcină poate fi dusă la îndeplinire: poate un singur robot să ducă la îndeplinire sarcina sau este necesară participarea mai multor roboţi? De asemenea,

se poate spune că un sistem multi-robot cooperează dacă, datorită unui mecanism cu care sistemul este înzestrat (numit mecanism de cooperare), apare o creştere în randamentul sistemului.

Algoritmii CNN prezentaţi în acest capitol pot fi testaţi practic, doar dacă în imaginile achiziţionate de către camera video se pot identifica poziţiile roboţilor mobili. Din aceste considerente rezultă dimensiunile imaginii şi pasul de eşantionare minim pentru discretizarea spaţială a imaginii achiziţionate.

Timpul total necesar pentru rularea algoritmului de grupare a roboţilor depinde de varianta de estimare a distanţelor dintre roboţi. Cel mai bun timp s-a obţinut în cazul în care estimarea distanţelor dintre roboţi s-a făcut în ordinea crescătoare a lor.

De remarcat faptul că algoritmul nu ţine cont neapărat de poziţiile geometrice ale roboţilor în grup, adică se determină că un robot este mai aproape sau mai departe de un altul dar nu este important în ce direcţie se află acesta. Totuşi, în faza de grupare a roboţilor, acel robot considerat extremă se va deplasa către un robot cu poziţie centrală în grup verificând la fiecare pas poziţiile din jurul său, şi se pot memora direcţiile sau poziţiile consecutive pe care s-a deplasat, astfel că ulterior se va putea găsi care a fost poziţia sa geometrică în raport cu robotul central.

Algoritmul pentru determinarea roboţilor cu poziţii centrale sau extreme din grup poate fi util şi în cazul menţinerii deplasării roboţilor mobili într-o anumită formaţie, dacă aceasta are o structură regulată, adică roboţii mobili componenţi se deplasează formând o figură geometrică regulată. Astfel, dacă unul din roboţi are la un moment dat o poziţie centrală în raport cu ceilalţi înseamnă că acesta s-a deplasat prea mult spre centrul formaţiei iar dacă are o poziţie extremă, atunci deplasarea sa, s-a făcut prea mult spre exteriorul formaţiei. În acest fel, poate fi corectată poziţia robotului respectiv, realizându-se menţinerea compactă a formaţiei.

Algoritmul de adunare, respectiv de grupare a roboţilor extremă în jurul robotului sau roboţilor cu poziţii centrale, descris la subcapitolul **5.4.3,** poate fi utilizat şi pentru planificarea traiectoriilor roboţilor mobili în medii fără obstacole (unul sau mai mulţi roboţi), caz în care poziţia robotului central poate fi considerată ca fiind poziţia ţintei.

6. MEDIU INTEGRAT PENTRU NAVIGAŢIA UNUI ROBOT MOBIL

6.1. Introducere

Realizarea practică a unui robot mobil, capabil să se deplaseze autonom în medii nestructurate, este una dintre cele mai dificile sarcini în domeniul roboticii. Deşi au apărut de-a lungul timpului o serie de algoritmi şi concepte teoretice cu privire la conducerea roboţilor mobili, multe din acestea au avut de aşteptat până când progresele în domeniul ştiinţei şi tehnicii au făcut posibilă implementarea lor.

Navigaţia autonomă a unui robot mobil într-un mediu cu obstacole, presupune existenţa unui sistem complex care, în primul rând, trebuie să perceapă mediul de lucru prin detecţia poziţiilor în care sunt situate obstacolele. Pe baza acestor informaţii şi ţinând cont de poziţia robotului şi de poziţia ţintei sistemul va determina o traiectorie optimă care să unească aceste două poziţii, astfel încât să fie evitate obstacolele. Pe baza acestei traiectorii, trebuie elaborate semnale de comandă pentru sistemul locomotor al robotului pentru ca acesta să urmărească traiectoria prescrisă. Sunt multe situaţii în care robotul nu poate menţine exact traiectoria indicată şi astfel sunt necesare unele corecţii ale acesteia în funcţie de informaţiile senzoriale locale obţinute în decursul navigării.

Perceperea structurii mediului de lucru cu obstacole se poate face pe baza unor hărţi cunoscute apriori sau realizate în timpul deplasării robotului pe baza informaţiilor senzoriale. Cei mai utilizaţi senzori în acest sens sunt: senzorii vizuali (camerele video) şi senzorii de proximitate (senzorii laser, în infraroşu şi ultrasonici).

Camera video constituie, în acest sens, o sursă bogată de informaţii, dar utilizarea ei pentru navigarea roboţilor mobili presupune procesarea unei cantităţi mari de informaţie care consumă timp şi poate duce în final la o viteză de navigare scăzută.

Prin utilizarea reţelelor neuronale celulare [49],[50],[68], care au un timp de procesare a imaginilor foarte mic, poate fi atenuat acest inconvenient. Sunt multe articole care propun utilizarea lor pentru procesarea imaginilor în scopul ghidării roboţilor mobili autonomi [87],[88],[95],[149], bazându-se şi pe faptul că reţele neuronale celulare pot fi implementate hard pe un singur chip VLSI [67].

În acest capitol se prezintă o concepţie originală a unui mediu integrat pentru comanda deplasării unui robot mobil pe baza imaginilor achiziţionate cu o cameră video. Procesarea imaginilor *gray-scale* ale mediului cu obstacole se realizează cu reţele neuronale celulare, prin utilizarea simulatorului analogic **MatCNN** [70],[71] din mediul de simulare Matlab [72]. Comenzile de deplasare sunt transmise spre robot prin intermediul portului paralel al calculatorului utilizând comenzile din Matlab dedicate accesării acestui port.

6.2. Structura mediului integrat

În Figura 6.1 se prezintă schema bloc de principiu a mediului integrat pentru comanda navigării unui robot mobil pe baza imaginilor reale ale mediului de lucru captate cu o cameră video.

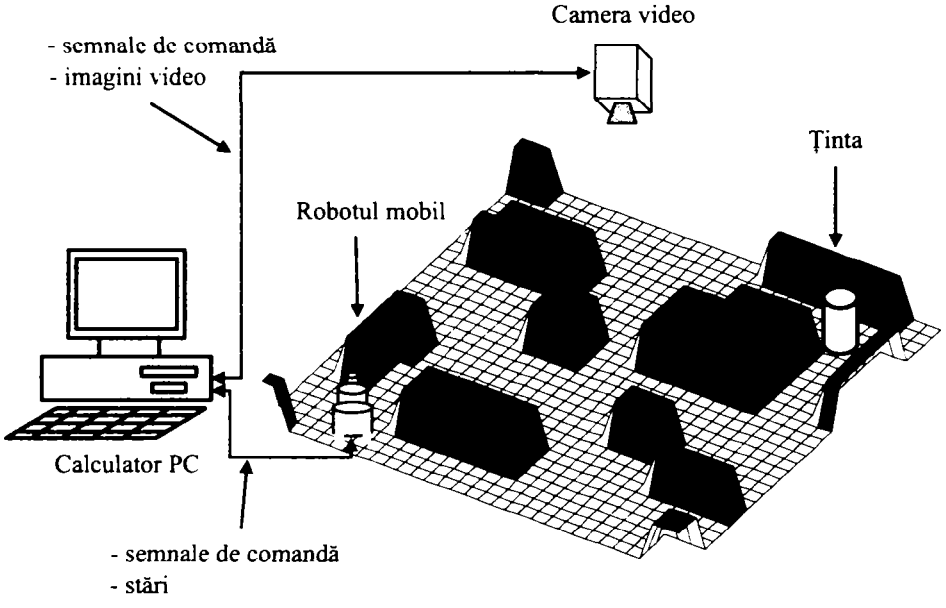


Figura 6.1. Structura sistemului integrat pentru navigaţia unui robot mobil într-un mediu cu obstacole.

Robotul trebuie să se deplaseze spre ţintă pe o traiectorie cât mai scurtă, ocolind în acelaşi timp, obstacolele pe care le întâlneşte în calea sa. Camera video supraveghează întreg mediul de lucru şi captează imagini la momente discrete de timp. După procesarea acestora va fi planificată, de fiecare dată, traiectoria ce trebuie urmată în continuare de către robot şi vor fi elaborate şi transmise comenzi către sistemul de locomoţie al robotului pentru realizarea deplasării pe direcţia specificată.

Organigrama pe baza căreia se derulează întreg procesul de navigare, începând cu achiziţia imaginilor, apoi procesarea acestora, planificarea traiectoriei şi în final comanda şi realizarea deplasării robotului se prezintă în Figura 6.2.

6.2.1. Achiziţia imaginilor

Într-o primă variantă, captarea imaginii mediului s-a făcut cu o cameră Web de tipul *USB PC Camera 305* montată pe portul USB al unui calculator Pentium IV. Comanda camerei video, respectiv stabilirea momentelor de achiziţie a imaginilor şi setarea parametrilor este realizată din mediul Matlab pe baza unui program denumit VFM (***Vision For Matlab***) [150],[151]. Practic, acest program transferă imaginile în mediul Matlab sub forma unor matrici, fiecare reprezentând ponderea uneia din culorile primare (roşu, verde şi albastru) în fiecare pixel din imagine.

Camera poate fi setată să achiziţioneze şi imaginea *gray-scale*, situaţie în care cele trei matrici sunt identice.

Rezoluţia imaginilor poate fi modificată, din setările soft ale camerei, în cinci trepte de la 160×120 pixeli până la valoarea maximă 640×480 pixeli. De asemenea, înainte de captarea imaginii pot fi realizate unele reglaje asupra camerei cum ar fi reglajul de luminozitate, contrast, saturaţie şi balansul de alb.

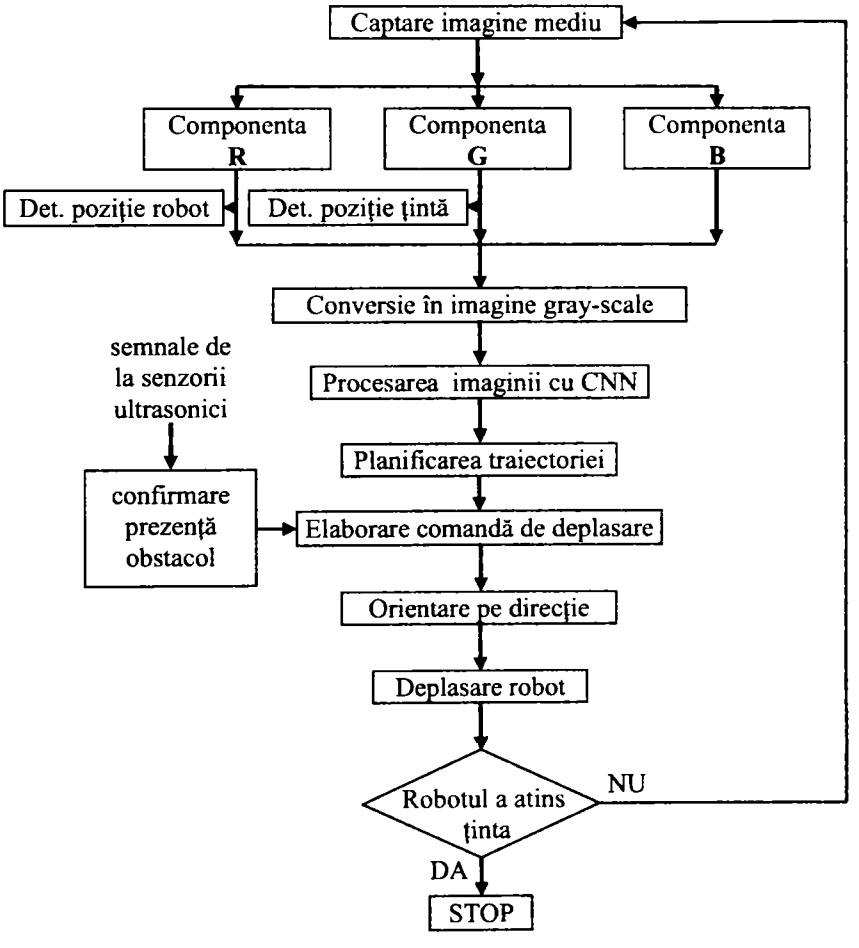


Figura 6.2. Organigrama procesului de planificare şi comandă a navigării robotului mobil.

În aceste imagini vor fi identificate de fiecare dată, pe baza matricelor rezultate, poziţia robotului mobil precum şi poziţia ţintei. Poziţia acestora va fi identificată prin câte un pixel chiar dacă dimensiunea reală a acestora este mai mare, se va reprezenta practic centrul de simetrie al imaginii celor două elemente.

Pentru a se face deosebire între acestea şi obiectele din mediu, poate fi ataşată în partea superioară a acestora, o sursă de lumină monocromă (ex. LED). Dacă culoarea acestor surse este aleasă una din culorile primare (roşu, verde sau albastru), atunci identificarea se poate face uşor, direct din imaginile color furnizate de cameră. Practic, prin identificare se înţelege aici, determinarea liniei şi coloanei

pe care se află pixelul din imagine, corespunzător poziţiei robotului şi ţintei. În acest scop discretizarea în spaţiu a imaginii mediului de lucru este aleasă corespunzător.

Poziţia robotului şi poziţia ţintei pot fi determinate şi pe baza imaginii alb-negru a mediului prin utilizarea unor algoritmi de recunoaştere a formelor şi apoi de detecţie a punctului central al acestor forme. În cazul robotului, pe lângă poziţia acestuia trebuie determinată şi orientarea sa în raport cu ţinta sau cu un sistem de referinţă fix.

Imaginea reală a mediului poate fi achiziţionată o singură dată şi apoi prin procesarea acestei imagini să fie planificată în întregime traiectoria robotului sau pot fi achiziţionate periodic mai multe imagini ale mediului la momente discrete de timp $(t_0 + kT)$, începând de la momentul iniţial t_0, cu pasul de eşantionare T, k fiind un număr natural, $k \in N$. Fiecare imagine achiziţionată va fi procesată, rezultând comanda (direcţia şi timpul de deplasare) ce trebuie transmisă robotului. În acest fel, procesul **achiziţie imagine – planificare traiectorie – comandă deplasare**, se va repeta până când robotul atinge ţinta. Această din urmă metodă are avantajul că ţine cont de eventualele modificări ale structurii mediului cu obstacole survenite în momentul deplasării robotului.

6.2.2. Procesarea imaginilor cu CNN

Pe baza celor trei matrici de culoare se obţine imaginea *gray-scale* a mediului care va fi utilizată pentru identificarea poziţiilor din mediu în care sunt situate obstacolele. Pentru a putea fi procesată cu ajutorul unei reţele neuronale celulare, imaginea trebuie transferată în domeniul standard CNN adică [-1, +1], de la alb către negru. Pentru prelucrarea acestor imagini s-a utilizat *toolbox*-ul MatCNN [81], din mediul de simulare Matlab.

Se consideră că obstacolele din mediul de lucru au luminanţa mai mică decât cea a spaţiilor libere, astfel că, pentru identificarea acestora se poate folosi template-ul TRESHOLD dat de relaţia 6.1:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad z = 0 \cdot \tag{6.1}$$

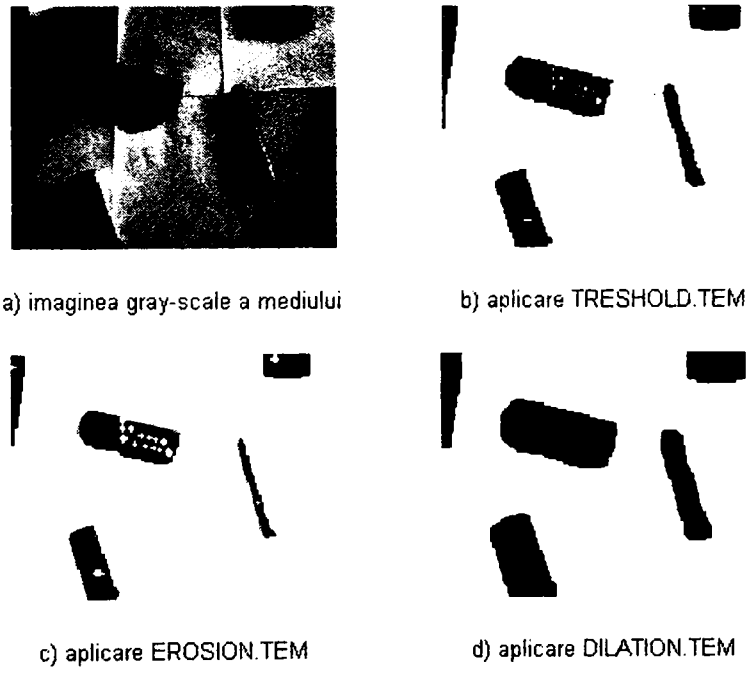
Pe intrarea şi pe starea iniţială a reţelei neuronale celulare se aplică imaginea cu niveluri de gri a mediului cu obstacole. După aplicarea template-ului prezentat mai sus, pe ieşirea reţelei se obţine imaginea binară a mediului, în care obstacolele sunt reprezentate de către pixelii negri (valoarea +1) iar spaţiile libere respectiv spaţiile prin care poate să se deplaseze robotul, de pixelii albi (valoarea -1).

În funcţie de condiţiile de iluminare existente în mediul de lucru precum şi de variaţia luminanţei zonelor libere, în imaginea achiziţionată pot apare unele "zgomote", astfel că, unele porţiuni mici din spaţiul liber pot fi interpretate ca fiind obstacole. Apare astfel necesitatea eliminării acestora din imagine, lucru ce poate fi realizat prin aplicarea template-ului EROSION [70], dat de relaţia 6.2.

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad z = -4 \cdot \tag{6.2}$$

Datorită faptului că prin această acţiune sunt afectate şi dimensiunile obstacolelor din imagine, se aplică în continuare template-ul DILATION [70], prezentat în capitolele precedente. În acest fel, este posibilă refacerea dimensiunilor formelor care semnifică obstacolele.

În Figura 6.3 se prezintă un exemplu de aplicare a template-urilor prezentate mai sus asupra unei imagini *gray-scale* a mediului cu obstacole achiziţionate cu o cameră video (Figura 6.3a). În aceste imagini, cu negru sunt reprezentate obstacolele iar cu alb spaţiile libere ale mediului. Prin aplicarea template-ului TRESHOLD, valorile pixelilor care au luminanţa peste un anumit prag devin albi iar ceilalţi pixeli vor fi de culoare neagră, obţinându-se astfel o imagine binară a mediului (Figura 6.3b). Eliminarea unor eventuale "zgomote" sub forma unor dungi subţiri de culoare neagră, ce pot fi interpretate ca fiind obstacole se realizează prin aplicarea template-ului EROSION (Figura 6.3c), iar refacerea dimensiunilor obstacolelor este posibilă prin aplicarea template-ului DILATION (Figura 6.3d).



a) imaginea gray-scale a mediului

b) aplicare TRESHOLD.TEM

c) aplicare EROSION.TEM

d) aplicare DILATION.TEM

Figura 6.3. Exemplu de procesări CNN a imaginii reale ale unui mediu cu obstacole.

6.3. Planificarea traiectoriei

Deplasarea robotului se va face pas un pas prin spaţiul liber al mediului, ocolind obstacolele întâlnite în cale, până va atinge ţinta. Traiectoria acestuia trebuie planificată astfel încât robotul să nu atingă obstacolele şi chiar să păstreze o anumită distanţă minimă faţă de acestea.

6.3.1. Realizarea câmpului potenţial artificial

Pentru estimarea poziţiei robotului faţă de ţintă şi planificarea apoi a traiectoriei acestuia s-a utilizat metoda câmpului potenţial artificial [30],[31] care presupune realizarea, pe baza imaginii mediului cu obstacole, a unui câmp potenţial de atracţie şi a unui câmp potenţial de repulsie.

6.3.1.1. Câmpul potenţial de atracţie

Acest câmp se realizează cu ajutorul unei funcţii potenţiale de atracţie care alocă fiecărui punct (pixel) al imaginii discretizate a mediului o valoare proporţională cu distanţa (măsurată în număr de pixeli) din punctul respectiv până la pixelul ce indică poziţia ţintei.

Dacă se notează pixelul unde este situată ţinta cu T şi coordonatele sale în imagine cu (x_t, y_t) atunci distanţa de la un pixel de coordonate (i, j) al imaginii mediului de dimensiune $m \times n$, până la pixelul care reprezintă poziţia ţintei, se poate calcula cu formula de calcul a distanţei euclidiene dată de relaţia 6.2.

$$d(i,j) = \sqrt{(x_t - i)^2 + (y_t - j)^2} \quad \text{pentru} \quad \forall \text{ar fi } i = 1 \ldots m, j = 1 \ldots n. \tag{6.2}$$

Funcţia potenţială, denumită şi potenţial atractiv, pentru fiecare punct din mediul de lucru este dată de relaţia 6.3.

$$U_{atr}(i,j) = \frac{1}{2} \cdot k \cdot d(i,j) \quad \text{pentru} \quad \forall \text{ar fi } i = 1 \ldots m, j = 1 \ldots n, \tag{6.3}$$

unde k reprezintă un număr întreg pozitiv.

Valoarea câmpului potenţial artificial astfel creat, va avea minimul în punctul ţintă şi în rest valorile sale în fiecare alt punct din mediu, vor fi proporţionale cu distanţa din acel punct până în punctul ţintă.

6.3.1.2. Câmpul potenţial de repulsie

Prin crearea acestui câmp, robotul va fi "respins" de către obstacole şi va menţine o anumită distanţă faţă de ele. Câmpul potenţial de repulsie are valori maxime în punctele în care sunt situate obstacolele şi valori descrescătoare în imediata vecinătate a acestora.

Dacă se notează cu (z, c) perechile de coordonate ce reprezintă poziţiile ocupate de către obstacole, atunci valorile funcţiei potenţiale de repulsie pe baza căreia se realizează câmpul potenţial de repulsie sunt date de relaţia 6.4.

$$U_{rep}(i,j) = \begin{cases} U_{max} & \text{dacă} \quad i = z \text{ şi } j = c \\ n \cdot \left(\dfrac{1}{d((z,c),(i,j))} \right)^2 & \text{dacă } i \in [z-q, z+q] \text{ sau } j \in [c-q, c+q], \end{cases} \tag{6.4}$$

unde n reprezintă un număr întreg pozitiv, q reprezintă raza de acţiune, măsurată în număr de pixeli, a câmpului în jurul obstacolelor şi $d((z,c),(i,j))$ reprezintă distanţa euclideană dintre pixelii pe care sunt situate obstacolele şi pixelii din vecinătatea obstacolelor care sunt situaţi în raza de acţiune a câmpului.

Prin însumarea celor două câmpuri potenţiale, pe întreg mediul de lucru al robotului, se obţine câmpul potenţial artificial total dat de relaţia 6.5:

$$U_{total}(i,j) = U_{atr}(i,j) + U_{rep}(i,j), \text{ pentru } \forall \text{ ar fi } i = 1 \ldots m, j = 1 \ldots n. \quad (6.5)$$

În Figura 6.4 se prezintă un exemplu de realizare a unui câmp potenţial artificial pentru un mediu cu obstacole având imaginea cu rezoluţia 160×120 pixeli (Figura 6.4a). Dacă ţinta este situată în acest mediu în poziţia $i = 130$ şi $j = 40$, atunci câmpul potenţial de atracţie are forma prezentată în Figura 6.4b. Având cunoscute poziţiile pe care sunt situate obstacolele, se realizează câmpul potenţial de repulsie Figura 6.4c. Prin însumarea celor două câmpuri potenţiale se obţine câmpul potenţial artificial total (Figura 6.4d).

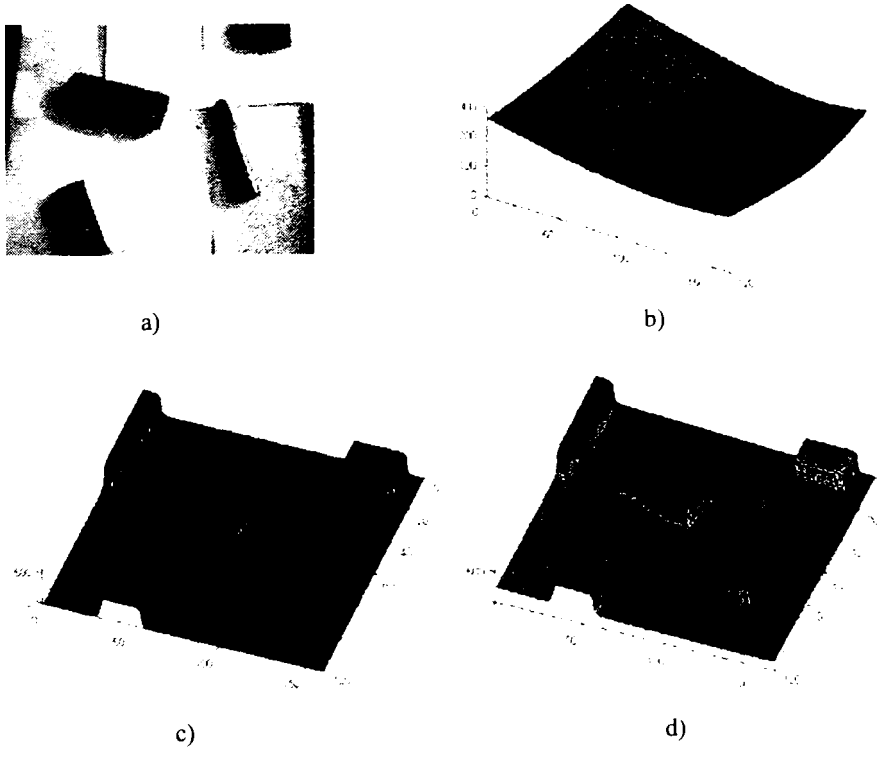


Figura 6.4. Exemplu de realizare a câmpului potenţial artificial; a) imaginea gray-scale a unui mediu cu obstacole, b) potenţialul de atracţie, c) potenţialul de repulsie, d) potenţialul total.

Pe baza acestui câmp, în funcţie de poziţia potenţialului minim din jurul poziţiei actuale a robotului, va fi stabilită direcţia optimă de deplasare pentru robot. Deplasarea acestuia se va face atâta timp cât valoarea câmpului potenţial pe direcţia deplasării este în scădere.

6.3.2. Calculul traiectoriei

Planificarea traiectoriei este realizată pixel cu pixel, începând de la pixelul care reprezintă poziţia iniţială a robotului (iR, jR), prin determinarea de fiecare dată a acelui pixel ce are valoarea cea mai mică dintre cei opt pixeli vecini, situaţi în jurul pixelului care reprezintă poziţia curentă a robotului.

Dacă se presupune că pixelul având coordonatele (i, j) este pixelul care coincide cu poziţia robotului, atunci direcţiile posibile de deplasare sunt date de cei opt pixeli vecini şi sunt reprezentate în Figura 6.5.

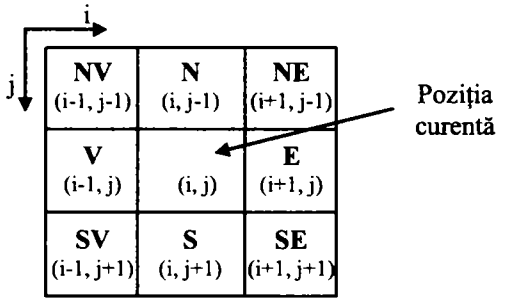


Figura 6.5. Direcţiile posibile de deplasare a robotului.

Valorile pixelilor din jurul poziţiei curente sunt exprimate în algoritmul de planificare a traiectoriei printr-o matrice linie X, iar valoarea minimă este dată de parametrul d (relaţia 6.5).

$$X = [N \quad S \quad E \quad V \quad NE \quad NV \quad SE \quad SV], \quad d = \min(X). \tag{6.5}$$

Prin compararea valorii lui d cu valorile elementelor matricei X, va fi identificată direcţia optimă de deplasare, respectiv poziţia viitoare (pixelul) prin care va trece traiectoria. Acest pixel va deveni pixel actual şi procesul se va repeta până va fi identificat pixelul care corespunde poziţiei ţintei (iT, jT). Acesta are valoarea cea mai mică dintre toţi pixelii ce reprezintă câmpul potenţial artificial al mediului de lucru, valoare atribuită prin potenţialul de atracţie.

În Figura 6.6 se arată forma traiectoriei planificate în cazul imaginilor prezentate mai sus, în care poziţia ţintei este dată de pixelul având coordonatele $i = 130$ şi $j = 40$, iar poziţia robotului este dată de pixelul de coordonate $i = 15$ şi $j = 100$.

6.4. Deplasarea robotului spre ţintă

Deplasarea robotului de-a lungul traiectoriei planificate presupune parcurgerea următoarelor etape:

- elaborarea comenzilor;
- transmiterea acestora către sistemul locomotor;
- orientarea robotului pe direcţia specificată;
- deplasarea robotului.

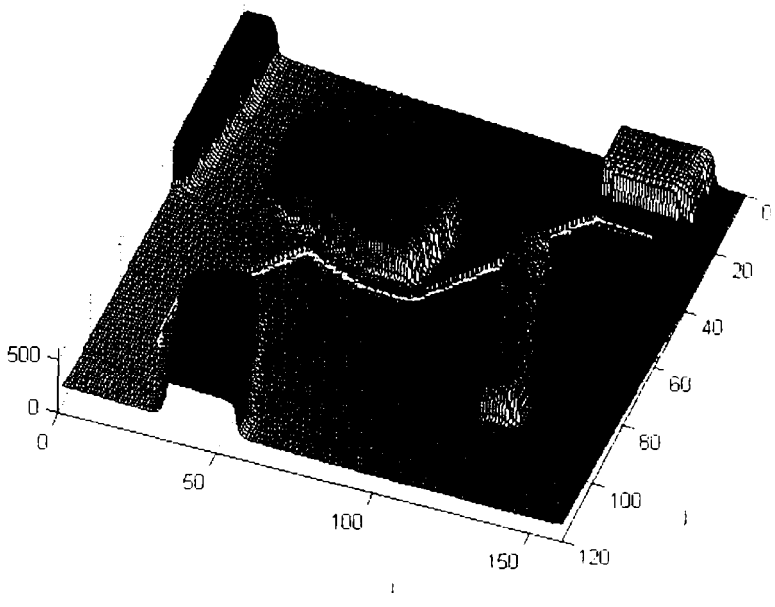


Figura 6.6. Forma traiectoriei pentru imaginile prezentate în Figura 6.4.

Aceste etape sunt parcurse o singură dată în cazul în care este achiziţionată o singură imagine a mediului şi pe baza ei se planifică întreaga traiectorie sau sunt parcurse ciclic atunci când imaginile sunt achiziţionate periodic.

6.4.1. Elaborarea comenzilor

Sistemul de locomoţie al robotului (Figura 6.7) este realizat cu două motoraşe de curent continuu MD şi MS care sunt comandate cu nivele de tensiune TTL (Ud şi Us). MD antrenează o roată motoare situată în partea laterală dreaptă iar MS o roată motoare situată pe partea laterală stângă. În partea din faţă a şasiului este montată o roată omnidirecţională pentru menţinerea echilibrului. Viteza maximă de deplasare a robotului este de 0.92 m/s.

Mişcările ce pot fi realizate de către robot şi semnalele de comandă aferente acestora sunt:

- deplasare în faţă (Ud = 1, Us = 1);
- viraj spre dreapta (Ud = 0, Us = 1);
- viraj spre stânga (Ud = 1, Us = 0).

Unghiul de viraj depinde de intervalul de timp t în care este menţinută comanda cu nivelul logic 1 asupra circuitului de comandă a motoraşului. Astfel, au fost stabilite tipurile de viraje:

- viraj la 45 de grade - t = 0.2 secunde;
- viraj la 90 de grade - t = 0.4 secunde;
- viraj la 135 de grade - t = 0.6 secunde;
- viraj la 180 de grade - t = 0.8 secunde.

În concluzie, semnalele de comandă pentru navigaţia robotului mobil sunt date de tripletul (Ud, Us, t).

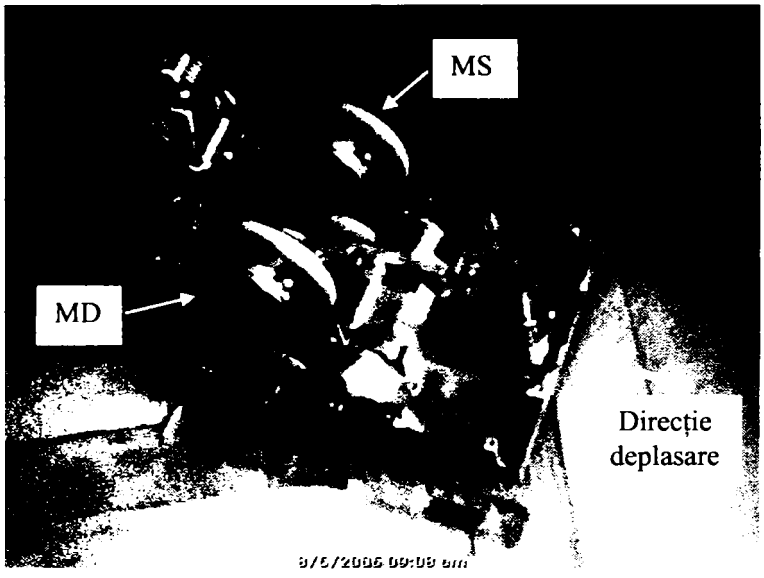


Figura 6.7. Robotul mobil utilizat pentru testarea practică a algoritmului de planificare a traiectoriei pe baza imaginilor.

6.4.2. Transmiterea comenzilor spre robot

Pentru transmiterea comenzilor de la calculator spre robotul mobil s-a utilizat portul paralel. Astfel, pentru comanda deplasării robotului s-au folosit pinii Data0 (D0) şi Data1 (D1) ai portului paralel (pinii 2 şi 3 ai conectorului de tip D). Aceştia au fost configuraţi din mediul Matlab ca pini de ieşire, cu rol de comandă pentru sistemul locomotor al robotului. Nivele de tensiune au valori TTL standard, adică +5V la nivelul logic 1 şi 0V la nivelul logic 0.

Comenzile transmise prin intermediul acestor pini sunt:

- **deplasare în faţă** – D0 = 1 şi D1 = 1;
- **viraj spre dreapta** – D0 = 0 şi D1 = 1;
- **viraj spre stânga** – D0 = 1 şi D1 = 0;
- **oprire** – D0 = 0 şi D1 = 0.

Legătura dintre portul paralel al calculatorului şi robot s-a realizat într-o primă etapă prin trei fire de conexiune, dintre care două fire pentru transmiterea semnalelor de comandă şi un fir pentru legătura de masă.

Schema electronică a montajului care realizează adaptarea semnalelor date de către port, astfel încât acestea să poată comanda controlerul motoarelor cu care este echipat robotul, se prezintă în Figura 6.8.

Acest montaj realizează şi separarea galvanică între tensiunile date de portul calculatorului şi tensiunile de comandă, respectiv de alimentare a motoraşelor cu care este echipat robotul.

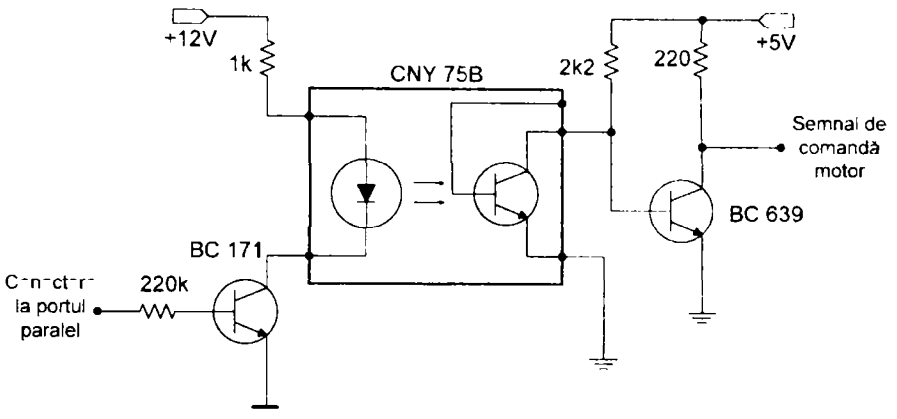


Figura 6.8. Schema electronică a montajului pentru adaptarea semnalelor de comandă şi pentru separare galvanică.

6.4.3. Orientarea robotului pe direcţia specificată

Dacă robotul primeşte o comandă de deplasare pe o anumită direcţie, şi dacă aceasta diferă de direcţia pe care s-a deplasat anterior atunci într-o primă fază, robotul va trebui să facă o rotaţie în jurul axei sale spre stânga sau spre dreapta pentru a fi pe direcţia specificată în comandă.

În cadrul algoritmului a fost stabilită o orientare iniţială a robotului, urmând apoi ca după fiecare schimbare de direcţie să fie actualizată această orientare. Spre exemplu, dacă se stabileşte orientarea iniţială ca fiind **SE** atunci vor fi iniţializate valorile $\mathrm{on}=0$, $\mathrm{os}=0$, $\mathrm{oe}=0$, $\mathrm{ov}=0$, $\mathrm{one}=0$, $\mathrm{onv}=0$, $\mathrm{ose}=1$, $\mathrm{osv}=0$. Dacă robotul are această orientare şi trebuie să schimbe direcţia de deplasare, atunci va trebui să facă unul din virajele prezentate în Figura 6.9.

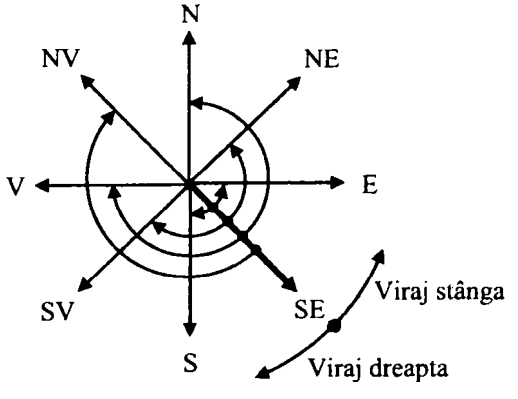


Figura 6.9. Modul de realizare a orientării robotului faţă de direcţia **SE**.

Pentru ca robotul să nu facă viraje mai mari de 180 de grade, manevrele pentru realizarea orientării au fost împărţite în viraje spre dreapta (de la 0 la 180 de grade) şi viraje spre stânga (de la 0 la 135 de grade).

Pentru situaţia în care orientarea robotului este pe direcţia **SE,** orientările posibile sunt:

- orientare spre **S** – viraj dreapta cu 45 de grade;
- orientare spre **SV** – viraj dreapta cu 90 de grade;
- orientare spre **V** – viraj dreapta cu 135 de grade;
- orientare spre **NV** – viraj dreapta cu 180 de grade;
- orientare spre **E** – viraj stânga cu 45 de grade;
- orientare spre **NE** – viraj stânga cu 90 de grade;
- orientare spre **N** – viraj stânga cu 135 de grade.

După ce robotul se află pe direcţia specificată, urmează deplasarea efectivă a acestuia. Pentru a avea o navigare continuă pe poţiunile drepte de pe traiectorie, comanda deplasării robotului va fi dată la sfârşitul planificării porţiunii respective. Timpul de menţinere a comenzii de deplasare în faţă (D0 = 1, D1 = 1) va fi proporţional cu numărul de pixeli ai segmentului de traiectorie planificat.

6.5. Concluzii

Pentru procesarea semnalelor s-au folosit atât funcţii din toolbox-ul **MatCNN** [81], cât şi instrucţiuni din mediul Matlab (vezi Anexa A7). Reducerea timpului de prelucrare necesar întregului algoritm ar fi posibilă dacă toate procesările şi chiar comenzile ar fi realizate în întregime cu ajutorul reţelelor neuronale celulare, respectiv utilizând un cip CNN.

Recunoaşterea robotului după formă, sau după faptul că acesta ar putea fi singurul obiect mişcător din mediu se poate soluţiona cu mijloace CNN. La fel şi ţintele, mai ales dacă sunt fixe, pot fi identificate pe baza imaginilor *gray-scale* ale mediului. Pornind de la aceste considerente poate fi setată camera video să capteze direct imaginea *gray-scale* a mediului caz în care organigrama prezentată în Figura 6.2 se simplifică. Odată recunoscut un obiect în imaginea achiziţionată, determinarea punctului central a imaginii acestuia poate fi realizată prin procesări CNN [56].

Metoda câmpului potenţial artificial are dezavantajul că robotul condus prin această metodă se poate bloca în minime locale, dacă în mediul de lucru există obstacole care au forme concave (având concavităţile orientate spre robot), şi care sunt situate în calea acestuia. Rezolvarea acestei probleme prin efectuarea unor procesări asupra imaginii binarizate a mediului cu obstacole astfel încât să fie eliminate eventualele concavităţi din imaginea ce reprezintă obstacolele, constituie în continuare un subiect de cercetare. Pe de altă parte se poate utiliza şi o altă metodă de planificare, spre exemplu în **Capitolul 4**, planificarea traiectoriei s-a realizat în totalitate prin procesări CNN, prin utilizarea template-ului EXPLORE, cu ajutorul căruia se generează o undă în planul imaginii pornind din punctul ţintă. În cazul utilizării acestui template nu apar probleme de genul minimelor locale. Implementarea acestui template în mediul de simulare Matlab, respectiv utilizând toolbox-ul **MatCNN** reprezintă, de asemenea, o temă de studiu.

O altă problemă întâmpinată în decursul testării practice, pe baza imaginii reale a unui mediu de lucru cu obstacole, o constituie modalitatea sa de iluminare. Astfel, dacă iluminarea mediului nu este optimă atunci pot fi identificate ca fiind obstacole şi unele zone întunecate din spaţiile libere cum ar fi umbra obstacolelor sau zonele libere care au culoare închisă. În cazul experimentului practic, suprafaţa pe care s-a deplasat robotul a fost aleasă de culoare deschisă, evitându-se astfel

interpretarea acesteia sau a unor părţi din aceasta ca fiind zonă ocupată de obstacole.

Prezenţa unei bucle de reacţie negativă de la nivelul elementelor de locomoţie spre sistemul de comandă a navigării ar fi benefică pentru o mai bună poziţionare a robotului. Sunt situaţii când datorită suprafeţelor cu rugozitate diferită a mediului apare fenomenul de patinare şi astfel pentru aceeaşi comandă de deplasare dar pe suprafeţe diferite de navigare se poate întâmpla ca robotul să parcurgă distanţe diferite. Prin echiparea robotului mobil cu traductori de deplasare odometrici ar putea fi evitate aceste erori de poziţionare.

Datorită acestor erori de poziţionare, robotul poate să de ciocnească de obstacole, mai ales în cazul în care pe baza unei singure imagini este planificată întreaga traiectorie a robotului urmând apoi elaborarea, pe baza acesteia, a succesiunii de comenzi pentru deplasare. Astfel, dacă robotul nu va efectua deplasarea completă pe un segment de traiectorie, atunci momentul când va primi următoarea comandă, acesta nu va efectua virajul în locul potrivit şi se poate ciocni de obstacole. Evitarea ciocnirii acestora poate fi realizată prin echiparea robotului cu senzori de proximitate (senzori laser sau ultrasonici), care vor semnala apropierea prea mare faţă de un eventual obstacol şi, în acest caz, va fi solicitată o nouă planificare a traiectoriei.

Utilizarea portului paralel pentru comunicarea calculator – robot are dezavantajul că sunt greu de transmis la distanţă datele furnizate de acesta. Un subiect de cercetare îl constituie utilizarea portului serial şi USB pentru transmiterea şi recepţionarea informaţiilor respectiv comenzilor între calculator şi robotul mobil.

Eliminarea firelor de legătură dintre robot şi calculator se poate realiza prin stabilirea unei comunicări în infraroşu sau a unei comunicări prin unde radio şi prin echiparea robotului cu o baterie de acumulatori pentru alimentarea motoraşelor. Pentru a fi complet autonom mai trebuie ca robotul să posede un sistem de calcul care să realizeze toate operaţiile de procesare a imaginilor în scopul planificării traiectoriei, lucru care e greu realizabil prin utilizarea microcontrolerelor clasice.

Deşi "creierul" robotului mobil prezentat aici, se află *off-board* (acesta fiind situat la nivelul unităţii centrale a calculatorului PC) se poate întrezări modalitatea de realizare a unui robot mobil complet autonom, care să utilizeze, pentru procesarea informaţiilor senzoriale, procesoare realizate prin implementarea reţelelor neuronale celulare.

7. CONTRIBUŢII

În **Capitolele 1**, **2** şi **3** sunt studiate şi sintetizate aspectele teoretice actuale precum şi realizări în domeniul navigaţiei roboţilor mobili autonomi şi al reţelelor neuronale celulare.

Capitolele 4, **5** şi **6** sunt bazate, în mare măsură, pe contribuţiile originale ale autorului. Cea mai mare parte dintre aceste contribuţii sunt incluse în cele 18 lucrări prezentate la sesiuni de comunicări ştiinţifice din ţară şi străinătate, în care autorul tezei este coautor şi care sunt citate pe parcursul tezei.

Contribuţiile importante sunt enumerate mai jos, în ordinea prezentării lor în teză.

1. Prezentarea, pe baza bibliografiei citate, a aspectelor teoretice actuale cu privire la roboţii mobili autonomi: structura internă, clasificare, aplicaţii, evoluţia, etc.
2. Studiul comparativ şi sinteza metodelor şi strategiilor de navigare cunoscute pentru roboţii mobili autonomi.
3. Studiul comparativ privind sistemele de conducere a roboţilor mobili (ierarhice, reactive şi deliberativ-reactive).
4. Realizarea şi prezentarea unei modalităţi originale de implementare în mediul de simulare Matlab a metodei câmpului potenţial artificial pentru planificarea traiectoriei roboţilor mobili.
5. Elaborarea şi testarea unei metode de evitare a minimelor locale ce pot apărea în cazul obstacolelor cu forme concave.
6. Studiul şi sinteza aspectelor teoretice actuale ale reţelelor neuronale celulare monostrat şi multistrat.
7. Studiul reţelelor neuronale celulare standard cu operatori de dimensiune 3×3, care au avantajul că pot fi implementate hard.
8. Analiza şi sinteza etapelor principale de realizare a algoritmilor CNN în cazul mediilor de simulare utilizate în teză: MatCNN şi CadetWin.
9. Studiul comparativ al circuitelor integrate CNN existente.
10. Conceperea, elaborarea şi testarea prin simulare a unor algoritmi CNN de planificare, pe baza imaginilor, a traiectoriei roboţilor mobili.
11. Elaborarea şi testarea unei metode originale de alegere a direcţiei optime de deplasare a robotului mobil prin proiectarea, în acest scop, a unui template CNN denumit PATH.
12. Studiul comparativ privind diversele metode de alegere a direcţiei optime şi modul de deplasare a robotului mobil în ceea ce priveşte timpul de rulare a algoritmilor şi numărul de viraje efectuat de către robot.

13. Elaborarea şi testarea unui algoritm CNN pentru planificarea simultană a traiectoriilor pentru doi roboţi mobili care se deplasează spre o ţintă comună.

14. Descrierea unui mod original de concepţie, elaborare şi testare a unui algoritm CNN pentru simularea deplasării unui robot mobil într-un labirint, pe baza imaginii acestuia.

15. Realizarea unei modalităţi originale pentru optimizarea operaţiilor de determinare a direcţiei de deplasare a robotului pe lângă perete, în cazul celor două modalităţi de navigare în labirint (având ca reper peretele din dreapta respectiv peretele din stânga).

16. Optimizarea prin procesări CNN a traiectoriei robotului prin labirint.

17. Analiza şi sinteza principalelor aspecte teoretice din cadrul studierii colectivităţilor de roboţi mobili.

18. Concepţia teoretică originală pentru estimarea distanţelor dintre roboţii mobili aparţinând unei colectivităţi, folosind metode CNN.

19. Elaborarea şi testarea unui algoritm CNN pentru coordonarea deplasării roboţilor mobili din cadrul unei colectivităţi.

20. Prezentarea unei modalităţi originale pentru adunarea roboţilor cu poziţii extreme în jurul roboţilor cu poziţii centrale din cadrul unei colectivităţi.

21. Concepţia şi implementarea practică originală a unui mediu integrat pentru deplasarea unui robot mobil, pe baza imaginilor mediului cu obstacole furnizate de o cameră video.

22. Procesarea imaginilor reale ale mediului cu obstacole utilizând reţele neuronale celulare.

23. Testarea funcţionalităţii practice a mediului integrat pe baza simulatorului analogic de reţele neuronale celulare monostrat, MatCNN, din mediul de simulare Matlab, şi prin utilizarea portului paralel al calculatorului.

24. Elaborarea unor concluzii pe baza realizării unor serii de teste practice privind deplasarea robotului într-un mediu real cu obstacole, pe baza imaginilor, în diverse condiţii de iluminare şi pe suprafeţe cu diferite rugozităţi.

BIBLIOGRAFIE

[1] H. M. Barberá, *A distributed architecture for intelligent control in autonomous mobile,* Ph. D. Thesis, Dept. of Communications and Information Engineering, University of Murcia, January, 2001.

[2] T. Botoş, V. Tiponuţ, *Soluţii cunoscute privind conducerea roboţilor mobili autonomi*, Referat nr. 1 în cadrul pregătirii lucrării de doctorat, Universitatea Politehnică din Timişoara, 1997.

[3] R. D. Schraft, *Mechatronics and Robotics for Service Applications*, IEEE Robotics and Automation Magazine, Vol. 1(4), Dec., pp. 31-35, 1994.

[4] H. Chuo, *Intelligent Mobile Robots: The State of Art and Perspectives*, Romansy-2004, Montreal, Canada, June 14-18, 2004.

[5] A. Meystel, *Autonomous Mobile Robots Vehicles with Cognitive Control*, World Scientific Publishing Co. Pte. Ltd., Singapore, 1991.

[6] H. R. Everett, *Sensors for Mobile Robots - Theory and Application,* A. K. Peters, Wellesley, MA, 1995.

[7] L. S. Lopes, A. Teixeira, *Human-Robot Interaction through Spoken Language Dialogue*, Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, Takamatsu, Japan, Oct./Nov., pp. 528-534, 2000.

[8] F. Mondada, E. Franzi, P. Ienne, *Mobile robot miniaturizations: A tool for investigation in control algorithms,* Experimental Robotics III, Proceedings of the 3-rd International Symposium on Experimental Robotics, Kyoto, Japan, October 28-30, 1993, Springer Verlag, London, pp. 501-513, 1994.

[9] Lynne E. Parker, *Multi-Robot Team Design for Real-World Applications*, Distributed Autonomous Robotic Systems 2, edited by H. Asama, T. Fukuda, T. Arai, and I. Endo, Springer-Verlag, Tokyo, pp. 91-102, 1996.

[10] R. Johansson, *Intelligent Motion Planning for a Multi-Robot System,* Master's Thesis in Computer Science at the School of Computer Science and Engineering, Royal Institute of Technology, Stockholm, Sweden, 2000.

[11] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, *Cooperative Mobile Robotics: Antecedents and Directions*, Autonomous Robots, Kluwer Academic Publishers, Vol. 4, pp. 1-23, 1997.

[12] G. A. S. Pereira, B. S. Pimentel, L. Chaimowicz, M. F. M. Campos, *Coordination of multiple mobile robots in an object carrying task using implicit communication,* VERLab - Laboratorio de Visao Computacional e Robotica, Departamento de Ciencia da Computacao, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

[13] G. A. S. Pereira, A. K. Das, R. V. Kumar, M. F. M. Campos, *Decentralized motion planning for multiple robots subject to sensing and communication constraints,* Published in Multi-Robot Systems: From Swarms to Intelligent Automata, Vol. II, Proceedings of the 2003 International Workshop on Multi-Robot Systems, pp. 267-278, 2003.

[14] P. Zebrowski, *Communication in Multi-Robot Systems,* Simon Fraser University, Spring, 2004.

[15] M. Simoncelli, G. Zunino, H. I. Christensen, *Autonomous Pool Cleaning: Self Localization and Autonomous Navigation for Cleaning*, Autonomous Robots No. 9, pp. 261–270, Kluwer Academic Publishers, Manufactured in The Netherlands, 2000.

[16] S. Thrun, D. Fox, W. Burgard, F. Dellaert, *Robust Monte Carlo Localization for Mobile Robots*, School of Computer Science, Carnegie Mellon University, Pittsburgh, to appear in Artificial Intelligence, Summer, 2001.

[17] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, *MINERVA: A Second-Generation Museum Tour-Guide Robot*, School of Computer Science, Carnegie Mellon University, Pittsburgh and Computer Science Department III University of Bonn, Germany, 1999.

[18] R. A. Russell, *Laying and sensing odor markings as a strategy for assisting mobile robot navigation tasks*, IEEE Robotics and Automation Magazine, pp. 3-9, September, 1995.

[19] D. Lange, J. K. Rosenblatt, M. Heber, *A behavior-based system for off-road navigation*, IEEE Journal of Robotics and Automation, Vol. 10, No. 6, pp. 776-782, Dec., 1994.

[20] S. K. Choi, J. Yuh, *Development of the omni-directional intelligent navigator*, IEEE Robotics and Automation Magazine, Vol. 2, March, 1995.

[21] H. Schempf, B. Chemel and N. Everett, *NEPTUNE: Above ground Storage-Tank Inspection Robot System*, IEEE Robotics and Automation Society Magazine, June, 1995.

[22] H. Lehtinen, P. Kaarmila, M. Blom, I. Kauppi, J. Kerva, *Mobile robots evolving in industrial applications*, VTT Automation, Machine Automation, Mobile Robotics / Mechatronics, //www.vtt.fi/aut/kau, Finland, 2001.

[23] G. Schweitzer, *Mechatronics for the design of human-oriented machines*, IEEE/ASME Transactions on Mechatronics, 1/2, pp. 120-126, June, 1996.

[24] V. Tiponuţ, L. Ţepelea, C. Lar, **I. Gavriluţ**, A. Gacsádi, *An integrated environment for assisted movement of blind persons*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES' 2005), Oradea, pp. 128 -131, 2005.

[25] V. Tiponuţ, A. Gacsádi, L. Ţepelea, C. Lar, **I. Gavriluţ**, *Integrated Environment for Assisted Movement of Visually Impaired*, Proceedings of the 15th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2006), June 15-17, Balatonfüred, Hungary, 2006.

[26] C. R. Weisbin, D. Lavery, *NASA rover and telerobotics technology program*, IEEE Robotics and Automation Magazine, pp.14-21, Dec., 1994.

[27] M. Khatib, *Sensor-based motion control for mobile robots,* Ph. D. thesis, Laboratoire d'Automatique et d'Analyse des Systemes, LAAR-CNSR, Toulouse, France, 1996.

[28] J. V. Dam, *Environment Modelling for Mobile Robots: Neural Learning for Sensor Fusion*, PhD thesis, Dept. of Computer Systems, University of Amsterdam, 1998.

[29] E. Sacks, *Path planning for planar articulated robots using configuration spaces and compliant motion*, IEEE Transactions on Robotics and Automation, Vol. 19, No. 3, pp. 381–390, 2003.

[30] J. Borenstein, Y. Koren, *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*, IEEE Transaction of Robotics and Automation Vol. 7, No. 3, pp. 278-288, June, 1991.

[31] Z. Liu, M. H. Ang Jr., W. K. G. Seah, *A Potential field based Approaches for Multi-Robot Tracking of Multiple Moving Targets*, The First Humanoid, Nanotechnology, Information Technology, Communication and Control Environment and Management (HNICEM) International Conference, March 27-30, Manila, Philippines, 2003.

[32] H. Choset, J. Burdick, *Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph*, The International Journal of Robotics Research Vol. 19, No. 2, pp. 96-125, February, 2000.

[33] W. L. Roque, D. Doering, *Constructing Approximate Voronoi Diagrams from Digital Images of Generalized Polygons and Circular Objects*, Proceedings of WSCG'2003, Febr. 3-7, Plzen, Czech Republic, 2003.

[34] A. G. Lamperski, Owen Y. Loh, Brett L. Kutscher, Noah J. Cowan, *Dynamical Wall-Following for a Wheeled Robot using a Pasive Tactile Sensors*, John Hopkins University, Baltimore, 2004.

[35] P. van Turennout, G. Honderd, L. J. van Schelven, *Wall-following control of a Mobile Robot*, IEEE, International Conference on Robotics and Automation, pp. 280-285, May, 1992.

[36] J. Hertz, A. Krogh, R. G. Palmer. *Introduction to the theory of neural computation.* Addisonn -Wesley Publishing Company, ISBN 0-201-50395-6, 1991.

[37] Z. Yi, *Multi-Ultrasonic Sensor Fusion for Mobile Robots in Confined Spaces*, Ph. D. thesis, School of Electrical & Electronic Engineering, Nanyang Technological University, 2001.

[38] F. Wallner, R. Graf, R. Dillmann, *Real-time Map Refinement by Fusing Sonar and Active Stereo-Vision*, University of Karlsruhe, Institute for Real-Time Computer Systems & Robotics, Karlsruhe, Germany, 1995.

[39] M. Milford, *RatSLAM: A Hippocampal Model for Simultaneous Localisation and Mapping*, Ph. D. Confirmation Report, April 21, 2004.

[40] **I. Gavriluţ**, *A navigation system for autonomous mobile robot*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES'01), Băile Felix, pp. 51-54, 2001.

[41] D. Zhu, J. C. Latombe, *New heuristic algorithms for efficient hierarchical path planning*, IEEE Transactions on Robotics and Automation, No. 7, pp. 9-20, Febr., 1991.

[42] L. Moreno, J. M. Armingol, S. Garrido, A. de la Escalera, M. A. Salichs, *A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors*, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Spain, July, 2001.

[43] A. Georgiev, Peter K. Allen, *Localization Methods for a Mobile Robot in Urban Environments*, IEEE Transactions on Robotics and Automation, Vol. 20, No. 5, pp. 851-864, Oct., 2004.

[44] J. Borenstein, H. R. Everett, L. Feng, D. Wehe, *Mobile Robot Positioning & Sensors and Techniques*, Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots. Vol. 14, No. 4, pp. 231-249, 1996.

[45] R. R. Murphy, *Introduction to AI Robotics*, The MIT Press, ISBN 0-262-13383-0, 2000.

[46] **I. Gavriluţ**, *Path planning of a mobile robot using Potential Field Method*, Proceedings of the International Conference on Renewable Sources and Environmental Electro-Technologies (RSEE 2002), Oradea, pp. 55-58, 2002.

[47] T. Botoş, *Cercetări privind conducerea adaptivă a roboţilor mobili autonomi*, Universitatea Politehnica Timişoara, Teză de doctorat, 2005.

[48] U. Nehmzow, *Mobile Robotics: A Practical Introduction*, Springer - Verlag 2000 ISBN 1-85233-173-9.

[49] L. O. Chua, L. Yang, *Cellular neural networks: Theory*, IEEE Transactions on Circuits and Systems, (CAS), Vol. 35, pp. 1257-1272, 1988.

[50] L. O. Chua, L. Yang, *Cellular neural networks: Applications*, IEEE Transactions on Circuits and Systems, (CAS), Vol. 35, pp. 1273-1290, 1988.

[51] Edward R. Sykes, A. Mirkovic, *A Fully Parallel and Scalable Implementation of a Hopfield Neural Network on the SHARC-NET Supercomputer*, The 19-th International Symposium on High Performance Computing Systems and Applications (HPCS'05), pp. 103-109, 2005.

[52] L. O. Chua, M. Hasler, G. S. Moschytz, J. Neirynck, Autonomous cellular neural network: A unified paradigm for pattern formation and active wave propagation, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, (CAS-I), Vol. 42, pp. 559-577, 1995.

[53] A. Gacsádi, P. Szolgay, *Adaptive Image Enhancement by Using Cellular Neural Networks*, Proceedings of the European Conference on Circuit Theory and Design, Cracow, Poland, Sept., 2003.

[54] A. Gacsadi, **I. Gavriluţ**, L. Ţepelea, Îmbunătăţirea imaginilor utilizând reţele neuronale celulare şi ecuaţii diferenţiale parţiale, Analele Universităţii din Oradea, Fascicula Colegiului Universitar Tehnic, Economic şi de Administraţie, Oradea, 2004.

[55] A. Gacsádi, R. Reiz, **I. Gavriluţ**, L. Ţepelea, V. Tiponuţ, *Noise removal in images by using cellular neural networks*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES' 2005), Oradea, pp. 50-53, 2005.

[56] A. Gacsádi, I. Turcaş, J. Vandewalle, *A CNN algorithm for determining the position of an object in an image*, Proceedings of the 3-th International Conference on Renewable Sources and Environmental Electro-Technologies (RSEE 2000), Oradea, pp. 101-106, 2000.

[57] Z. Nagy, P. Szolgay, *Numerical Solution of a Class of PDEs by Using Emulated Digital CNN-UM on FPGAs*, Proceedings of the European Conference on Circuit Theory and Design, Cracow, Poland, Sept., 2003.

[58] A. Gacsádi, **I. Gavriluţ**, L. Ţepelea, *Rezolvarea ecuaţiilor diferenţiale parţiale utilizând reţele neuronale celulare*, Conferinţa Internaţională, "Tendinţe în dezvoltarea aplicaţiilor ciberneticii", Academia de Cibernetică, "Ştefan Odobleja", Băile Felix, 2004.

[59] A. Gacsádi, V. Tiponuţ, *Reţele neuronale celulare – Aplicaţii*, Editura Universităţii din Oradea, ISBN 973-613-144-0, 2002.

[60] L. O. Chua, T. Roska, *Cellular Neural Networks: Foundation and Primer*, Version 1.7, Lecture notes for the course EE129 at. U.C. Berkeley, 1998.

[61] T. Yang, *Multi-layer Cellular Neural Networks: Theory and Applications to Modeling Nitric Oxide Diffusion in Nervous Systems*, International Journal of Computational Cognition, Vol. 1, No. 2, pp. 1–23, June, 2003.

[62] A. Gacsádi, C. Grava, *Simulation of the discrete time cellular neural network-DTCNN*, Proceedings of the International Conference on Renewable Sources and Environmental Electro-Technologies (RSEE'98), pp. 99-104, Băile Felix, 1998.

[63] R. Carmona, S. Espejo, R. Domínguez-Castro, A. Rodríguez-Vázquez, *CMOS Optoelectronic IC for Radon Transform of Binary Images*, October, 1993.

[64] R. Carmona, S. Espejo, R. Domínguez-Castro, A. Rodríguez-Vázquez, *CMOS Optoelectronic IC for Real-Time Extraction of Image Features,* June, 1999.

[65] Wai-Chi Fang, Bing J. Sheu, H. Venus, R. Sandau, *Smart pixel Array Processors Based on Optimal Cellular Neural Networks for Space Sensor Applications,* Proceedings of the International Conference on Computer Design: VLSI in computers & Processors (ICCD '95), 1995.

[66] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazguez, R. A. Carmona, P. Földesy, Á. Zarándy, P. Szolgay, T. Szirányi, T. Roska, *0.8μm CMOS Two dimensional programmable mixed - signal focal - plane array processor with on - chip binary imaging and instructions storage*, IEEE Journal of Solid State Circuits, Vol. 32, pp. 1013-1026, 1997.

[67] G. L. Cembrano, A. Rodríguez-Vázquez, S. Espejo-Meana, R. Domínguez-Castro: *ACE16k: A 128x128 Focal Plane Analog Processor with Digital I/O*. International Journal Neural Systems, 13(6), pp. 427-434, 2003.

[68] T. Roska, L. O. Chua, *The CNN Universal Machine: An analogical array computer*, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 40, pp. 163-173, 1993.

[69] *** *CadetWin-99 - CNN application development environment and toolkit under Windows,* Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Science, Budapest, 1999.

[70] C. Rekeczky, *MATCNN - Analogic Simulation Toolbox for Matlab*, Version 1.0, DNS-11-1997, Technical Report, Analogical and Neural Computing Laboratory, Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, Sept., 1997.

[71] C. D. Căleanu, V. Tiponuţ, *Reţele neuronale – Aplicaţii*, Editura Politehnică Timişoara, ISBN 973-9389-67-8, 2001.

[72] M. Ghinea, V. Fireţeanu, *MATLAB Calcul numeric – Grafică - Aplicaţii*, Editura Teora, Bucureşti, 1995.

[73] *** *CadetWin-99, VisMouse - CNN Visual mouse software platform for Windows* Reference manual, Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[74] *** *CadetWin-99, TemMaster - Template design and optimization tool for binary input-output CNNs*, User's guide, Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[75] *** *CadetWin-99, SimCNN - Multi-layer CNN simulator for visual mouse platform*, Reference Manual, Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[76] *** *CadetWin-99, CNN Alpha language and compiler*, Reference Manual, Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[77] *** *CadetwWin-99, Extended analogic macro code (AMC) and interpreter*, Reference Manual, Version 3.0, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[78] *** *CCPS, CNN chip prototyping system*, User's Guide, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1997.

[79] ***, *CSL-CNN Software Library (Templates and Algorithms)*, Version 7.3, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1999.

[80] T. Roska, *Analogic CNN computing: architectural, implementation, and algorithmic advances - a review*, Proceedings of IEEE International Workshop on Cellular Neural Networks and their Applications, (CNNA'98), pp. 3-10, London, 1998.

[81] *** *Quick Reference for MatCNN - Analogic CNN Simulation Toolbox for MATLAB* Version: 1.6, 2005.

[82] *** *MatCnn Instant Vision Toolbox for MatLab User's Guide*, Version: 1.0, Eutecus, Inc., 2005

[83] G. Linan, S. Espejo, R. Dominguez-Castro, E. Roca, A. Rodriguez-Vazquez, *CNNUC3: A mixed-signal 64x64 CNN Universal Chip*, Proceedings of the International Conference on Microelectronics for neural, fuzzy and bio-inspired systems, (MicroNeuro99), pp. 61-68, Granada, 1999.

[84] A. Paasio, A. Kananen, K. Halonen, V. Porra, *A 48 by 48 CNN chip operating with B/W images*, Proceedings of IEEE International Conference on Electronics, Circuits and systems, (ICECS'98), pp. 191-194, Lisboa, 1998.

[85] P. Keresztes, T. Bezák, Á. Zarándy, T. Roska, P. Szolgay, T. Hídvégi, P. Jónás, A. Katona, *Design methodology of an emulated digital CCCUM chip*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES'99), pp. 208-216, Băile Felix, 1999.

[86] R. Glasius, A. Komoda, S.Gielen, *Neural network dynamics for path planning and obstacle avoidance*, Department of Medical Physics and Biophysics, University of Nijmegen, The Netherlands, 1994.

[87] A. Gacsádi, T. Maghiar, V. Tiponuţ, *Path planning for a mobile robot in an environment with obstacles using cellular neural network*, International Workshop on Cellular Neural Networks and their Applications, (CNNA 2002), Frankfurt/Main, Germany, pp. 188-194, 2002.

[88] **I. Gavriluţ**, A. Gacsádi, V. Tiponuţ, *Path planning for a cooperative task in case of two mobile robots using cellular neural network*, Proceedings of the 2-nd International Conference on Robotics (Robotica - 2004), Timişoara / Reşiţa, 14-16 Oct., 2004.

[89] **I. Gavriluţ**, *Path tracking by mobile robot using cellular neural network*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES' 2003), Oradea, pp. 54-57, 2003.

[90] A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galan, B. De Lacy Costello, L. Fortuna, M. Frasca and A. Rodriguez-Vazquez, *Reaction-Diffusion Navigation Robot Control: From Chemical to VLSI Analogic Processors*, IEEE Trans. on Circuits and Systems – I: Regular Papers, Vol. 51, No. 5, pp. 926-938, May, 2004.

[91] P. Arena, L. Fortuna, M. Frasca, G. Vagliasindi, M. E. Yalcin, A. Basile, J.A.K. Suykens, *CNN Wave Based Computation for Robot Navigation on ACE16k*, IEEE

International Symposium on Circuits and Systems, ISCAS 2005, pp. 5818-5821, 23-26 May, 2005.

[92] P. Arena, A. Basile, L. Fortuna, A. Virzì, *Visual Feedback by Using a CNN Chip Prototype System*, Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, (CNNA 2002), pp. 291-297, Frankfurt, 2002.

[93] B. Siemiatkowska, A. Dubrawski, *Cellular Neural Networks for Navigation of a Mobile Robot,* L. Polkowski and A. Skowron (Eds.): RSCTC'98, LNAI 1424, pp. 147-154, Springer-Verlag Berlin Heidelberg, 1998.

[94] **I. Gavriluţ**, A. Gacsádi, *Trajectory tracking and predicting using CNN*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES 2003), pp. 58-61, Oradea, Romania, 2003.

[95] D. L. Vilarino, C. Rekeczky, *Shortest path problem with pixels level snakes: Application to robot path planning*, Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, (CNNA 2004), pp. 135-140, Budapest, 2004.

[96] **I. Gavriluţ**, A. Gacsádi, V. Tiponuţ, *Simultaneously paths planning for two mobile robots using cellular neural networks*, Proceedings of the 5-th International Conference on Renewable Sources and Environmental Electro-Technologies (RSEE 2004), Oradea, pp. 64-69, 2004.

[97] A. Gacsádi, *Contribuţii la conducerea adaptivă a roboţilor prin prelucrarea informaţiei vizuale utilizând reţele neuronale celulare*, Teză de doctorat, Universitatea Politehnică Timişoara, 2001.

[98] **I. Gavriluţ**, A. Gacsádi, C. Grava, V. Tiponuţ, *Vision based algorithm for path planning of a mobile robot by using cellular neural networks*, Proceedings of the International Conference on Automation, Quality&Testing, Robotics, AQTR 2006, ISBN 1-4244-0360-X, pp. 306-311, Cluj-Napoca, 2006.

[99] **I. Gavriluţ**, A. Gacsádi, L. Ţepelea, V. Tiponuţ, D. Albu, *Target search by mobile robot in a labyrinth using cellular neural networks*, Proceedings of the International Conference on Engineering of Modern Electric Systems (EMES' 2005), Oradea, pp. 54-59, 2005.

[100] R. Dillmann, J. Kreuzinger, F. Wallner, *PRIAMOS: An experimental platform for reflexive navigation*, Robotics and Autonomous Systems, Vol. 11, No. 3-4, pp. 195-203, Dec., 1993.

[101] Ben J. A. Krose, Kai M. Compagner, Franciscus C. A. Groen, *Accurate estimation of environment parameters from ultrasonic data*, Robotics and Autonomous Systems, Vol. 11, No. 3-4, pp. 221-230, Dec., 1993.

[102] John J. Leonard, Hugh F. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, Boston, 1992.

[103] R. Frizera Vassallo, H. J. Schneebeli, J. Santos-Victor, *Visual navigation: combining visual servoing and appearance based methods*, Proceedings of the International Symposium on Intelligent Robotic Systems, SIRS"98, Edinburgh, Scotland, July, 1998.

[104] L. Chaimowicz, V. Kumar, M. F. M. Campos, *A Paradigm for Dynamic Coordination of Multiple Robots*, Autonomous Robots 17(1), July 7-21, 2004.

[105] T. Huntsberger, G. Hickey, B. Kennedy, and H. Aghazarian, *Task Adaptive Walking Robots for Mars Surface Exploration*, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109-8099, (Terry.Huntsberger@jpl.nasa.gov), 2001.

[106] David Geer, *Small Robots Team Up to Tackle Large Tasks*, IEEE distributed systems online 1541-4922, Published by the IEEE Computer Society, Vol. 6, No. 12; Dec., 2005.

[107] J. A. Shapiro, *Bacteria as Multi-cellular Organisms*, Scientific American, Vol. 258, No. 6, pp. 62-69, 1988.

[108] J. A. Shapiro, *The Significances of Bacterial Colony Patterns*, BioEssays, Vol. 17, No. 7, pp. 597-607, 1995.

[109] C. Ronald Kube, H. Zhang, *Collective Robotics: From Social Insects to Robots*, Adaptive Behavior, Vol. 2, No. 2, pp. 189-219, 1993.

[110] J. M. Pasteels, J. L. Deneubourg, S. Goss, *Self-Organization Mechanisms in Ant Societies: Trail Recruitment to Newly Discovered Food Sources, From Individual to Collective Behavior in Social Insects*, J. M. Pasteels and J. L. Deneubourg (Eds.), Birkhäuser Verlag, Basel, pp. 155-175, 1987.

[111] C. Ronald Kube, H. Zhang, *Collective Robotic Intelligence, From Animals to Animats* 2, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, Meyer J-A, Roitblat H. L., Wilson S. W (Eds.), The MIT Pres., Cambridge, Mass., pp. 460-468, 1992.

[112] R. Beckers, O. E. Holland, J. L. Deneubourg, *From Local Actions to Global Tasks: Stigmergy and Collective Robotics*, Artificial Life IV, R. A. Brooks and P. Maes (Eds.), The MIT Press, Cambridge, Mass., pp. 181-189, 1995.

[113] S. Camazine, *Collective Intelligence in Insect Colonies by Means of Self-Organization*, Proceedings of European Conference on Artificial Life, Brussels, Belgium, pp. 158-173, 1993.

[114] Lynne E. Parker, *Adaptive Action Selection for Cooperative Agent Teams, From animals to animals*, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, J-A Meyer, H. L. Roitblat, S. W. Wilson, (Eds.), The MIT Press, Cambridge, Mass., pp. 12-23, 1992.

[115] C. Ronald Kube, H. Zhang, *Task modelling in collective robotics*, Kluwer Academic Publishers, Autonomous Robots, No. 4, pp. 53-72, Manufactured in The Netherlands, 1997.

[116] A. Agah, G. A. Bekey, *Philogenetic and Ontogenetic Learning in a Colony of Interacting Robots*, Autonomous Robots, Nr. 4, pp. 85-100, 1997.

[117] David C. Brogan, Jessica K. Hodgins, *Group Behaviors for Systems with Significant Dynamics*, Kluwer Academic Publishers, Autonomous Robots, Nr. 4, pp. 137-153, 1997.

[118] M. Vainio, T. Schönberg, A. Halme, P. Jakubik, *Optimizing the Performance of a Robot Society în Structured Environment through Genetic Algorithms,* Advances in Artificial Life, The 3-rd European Conference on Artificial Life, Granada, Spain, June, 1995.

[119] J. Fredslund, M. J. Mataric, *A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication*, IEEE Transaction Robotics and Automation, Vol. 18, No. 5, pp. 837-846, 2002.

[120] A. Halme, P. Jakubic, T. Schönberg, M. Vainio, *The concept of the robot society and its utilization in future robotics*, Advanced Robotics & Intelligent Machines, IEEE, Inc., London, pp. 255-272, 1966.

[121] D. Jung, G. Cheng, A. Zelinsky, *Experiments in Realizing Cooperation between Autonomous Mobile Robots*, Experimental Robotics V - The Fifth International Symposium Barcelona, Catalonia, June, 25-18, pp. 609-619, 1997.

[122] B. Hölldobler, E. O. Wilson, *The Ants*, Belknap of Harvard University Press, 1990.

[123] D. J. Stilwell, J. S. Bay, *Toward the Development of a Material Transport System using Swarms of Ant-like Robots*, Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, USA, pp. 766-771, 1993.

[124] C. Ronald Kube, *Collective Robotics: From Local Perception to Global Action,* Ph. D. Thesis, Computing Science, University of Alberta, 1997.

[125] D. P. Barnes, *A behavior synthesis for co-operant mobile robots*, Advanced Robotics & Intelligent Machines, IEEE, London, pp. 255-272, 1966.

[126] T. Balch, Ronald C. Arkin, *Behavior-Based Formation Control for Multirobot Teams*, IEEE Transactions on Robotics and Automation, Vol. 14, No. 5, Dec. 1998.

[127] V. Gazi, Kevin M. Passin, *Stability Analysis of Social Foraging Swarms*, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 34, No. 1, pp. 539-557, Feb., 2004.

[128] K. Sugihara, I. Suzuki, *Distributed motion coordination of multiple mobile robots*, Proceedings of IEEE International Conference on Robotics and Automation, pp. 138-143, 1990.

[129] P. K. C. Wang, *Navigation strategies for multiple autonomous robots moving in formation*, Journal of Robotics Systems, 8(2), pp. 177-195, 1991.

[130] S. Nakagawa, Y. Kawauchi, M. Buss, *Structure Decision Method for Self Organizing Robots Based on Cell Structures-CEBOT*, Proceedings of the 1989 IEEE International Conference on Robotics and Automation, pp. 695-700, 1989.

[131] Asama H., Matsumoto A., Ishida Y., *Design of an Autonomous and Distributed Robot System: "ACTRESS",* Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems, Tsukuba, Japan, pp. 283-290, 1989.

[132] M. J. Mataric, *Minimizing Complexity in Controlling a Mobile Robot Population*, Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, pp. 830-835, 1992.

[133] M. J. Mataric, *Designing Emergent Behaviors: From Local Interactions to Collective Intelligence,* Second International Conference on Simulation of Adaptive Behavior, (SAB-92), MIT Press, Cambridge, pp. 432-441, 1992.

[134] M. J. Mataric, M. J. Marjanovic, *Synthesizing Complex Behaviours by Composing Simple Primitives*, Preprints from the European Conference on Artificial Life, Brussels, Belgium, pp. 698-707, 1993.

[135] F. Mondada, A. Guignard, M. Bonani, D. Floreano D. Bär, M. Lauria *SWARM-BOT: From Concept to Implementation,* In C. S. George Lee and Junku Yuh, editors, Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS 2003), IEEE Press, Piscataway, NJ, USA, pp. 1626-1631, 2003.

[136] L. Wang, K. Tan, V. Prahlad, *Developing Khepera Robot Applications in a Webots Environment*, International Symposium on Micromechatronics and Human Science, IEEE, pp. 71-76, 2000.

[137] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, USA, ISBN 0-7923-9206-X, 1993.

[138] Y. H. Liu, S. Kuroda, T. Naniwa, H. Noborio, S. Arimoto, *A Practical Algorithm for Planning Collision-Free Coordinated Motion of Multiple Mobile Robots,* Proceedings of Conference on Robotics and Automation, IEEE Comput. Soc. Press, Vol. 3, pp. 1427-32, 1989.

[139] Donald D. Dudenhoeffer, Michael P. Jones, *A formation behavior for large-scale micro-robot force deployment,* Proceedings of the 2000 Winter Simulation Conference, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds., pp. 972-982, 2000.

[140] S. Yuta, S. Premvuti, *Coordinating Autonomous and Centralized Decision Making to Achieve Cooperative Behaviors Between Multiple Mobile Robots*, Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1566-1574, 1992.

[141] R. C. Arkin. *Cooperation without communication: Multi-agent schema based robot navigation*, Journal of Robotic Systems, 9(3), pp. 351-364, April, 1992.

[142] Lynne E. Parker, *The Effect of Action Recognition and Robot Awareness in Cooperative Robotic Teams*, Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, Nagoya, Japan, pp. 212-219, 1995.

[143] A. Martinoli, F. Mondada, *Collective and Cooperative Group Behaviours: Biologically Inspired Experiments in Robotics*, Proceedings of the Fourth International Symposium on Experimental Robotics ISER-95, Khatib O, Salisbury J. K. (Eds.), Stanford, US, June-July, Lecture Notes in Control and Information Sciences, Springer - Verlag, pp. 3-10, 1995.

[144] R. Cohen, D. Peleg, *Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems*, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100, Israel, March, 2004.

[145] V. Tiponuţ, **I. Gavriluţ**, C. Căleanu, A. Gacsádi, *Development of a Neural Network Guided Mobile Robot Collectivity*, WSEAS Transactions on Circuits and Systems Issue 6, Vol. 5, pp. 805-812, ISSN 1109-2734, Cavtat, Croaţia, June, 2006.

[146] V. Tiponuţ, A. Gacsádi, C. Căleanu, **I. Gavriluţ**, *Neural Network Guided Robot Collectivity – An Experimental Setup*, Proceedings of 7-th WSEAS International Conference on Neural Networks (NN'06), June 12-14, Cavtat, Croaţia, 2006.

[147] **I. Gavriluţ**, V. Tiponuţ, O. Neamţu, L. Ţepelea, A. Gacsádi, *Path Planning Methods for Mobile Robot by Using Cellular Neural Networks,* The 6-th International Conference on Renewable Sources and Environmental Electro-Technologies (RSEE 2006), Oradea, June 8-10, 2006.

[148] **I. Gavriluţ**, V. Tiponuţ, A. Gacsádi, *Path Planning of Mobile Robots by Using Cellular Neural Networks*, Proceedings of the 10-th International Workshop on Cellular Neural Networks and Their Applications (CNNA 2006), pp. 108-113, August 28-30, Istanbul, Turkey, 2006.

[149] **I. Gavriluţ**, A. Gacsádi, L. Ţepelea, V. Tiponuţ, *Motion planning for two mobile robots in an environment with obstacles by using cellular neural networks*, Proceedings of the The 7-th International Symposium on Signals, Circuits and Systems, (ISSCS 2005), pp. 801-804, Iaşi, 2005.

[150] A. Soinio, *A Lego-robot with camera controlled by Matlab*, LEGO Group Company, http://www.abo.fi/fak/tkf/rt/robot, 2003.

[151] http://www.isr.ist.utl.pt/~alex/Resources/vfm/.

ANEXA A1

```
;Program AMC pentru planificarea traiectoriei unui robot mobil pe baza alegerii
;direcţiei optime prin comparaţii succesive şi deplasarea robotului atâta timp cât deplasarea pe
;această direcţie asigură apropierea de ţintă.

start: host.load.tem explore.tem TEM1
host.load.tem shifte.tem TEM2
host.load.tem shiftv.tem TEM3
host.load.tem shiftn.tem TEM4
host.load.tem shifts.tem TEM5
host.load.tem shiften.tem TEM6
host.load.tem shiftvn.tem TEM7
host.load.tem shiftes.tem TEM8
host.load.tem shiftvs.tem TEM9
host.load.tem filwhite.tem TEM10
host.load.tem selecten.tem TEM11
host.load.tem selectes.tem TEM12
host.load.tem selectvn.tem TEM13
host.load.tem selectvs.tem TEM14
host.load.tem delen.tem TEM15
host.load.tem deles.tem TEM16
host.load.tem delvn.tem TEM17
host.load.tem delvs.tem TEM18
host.load.tem selecte.tem TEM22
host.load.tem selects.tem TEM23
host.load.tem selectv.tem TEM24
host.load.tem selectn.tem TEM25
host.load.tem dele.tem TEM26
host.load.tem dels.tem TEM27
host.load.tem delv.tem TEM28
host.load.tem deln.tem TEM29

host.load.pic mediu.bmp LLM1 BIN                ;imaginea mediului cu obstacole
host.load.pic tinta.bmp LLM2 BIN                ;imaginea ţintei
host.load.pic start.bmp LLM3 BIN                ;imaginea poziţiei de start
ar.nand.llm LLM2 LLM2 LLM4
ar.nand.llm LLM3 LLM3 LLM3
mov.llm.llm LLM3 LLM6           ;imaginea traiectoriei robotului
host.load.pic mediu.bmp LLM11 BIN
host.display LLM1 1
ar.nand.llm LLM1 LLM1 LLM1
mov.gam.gam 30 GAM10
host.display LLM3 2
host.display LLM4 3

;Evaluarea distanţelor dintre poziţiile libere din mediul de lucru şi poziţia ţintei
ar.tem TEM1 LLM1 LLM2 LAM3 200 1 1 LLM1 NIL
host.display LAM3 4
loop: ar.nand.llm LLM3 LLM3 LLM9
call optdir
```

```
;Deplasarea robotului pe direcţia optimă cât timp se asigură apropierea de ţintă
nordest: ar.tem TEM11 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_ne: ar.tem TEM15 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_ne
jump final
sudest: ar.tem TEM12 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_se: ar.tem TEM16 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_se
jump final
nordvest: ar.tem TEM13 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_nv: ar.tem TEM17 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_nv
jump final
sudvest: ar.tem TEM14 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_sv: ar.tem TEM18 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_sv
jump final
est: ar.tem TEM22 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_e: ar.tem TEM26 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_e
jump final
sud: ar.tem TEM23 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_s: ar.tem TEM27 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_s
jump final
vest: ar.tem TEM24 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
```

```
mov.gam.gam 0 GAM9
del_v: ar.tem TEM28 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_v
jump final
nord: ar.tem TEM25 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
host.display LLM6 5
mov.gam.gam 0 GAM9
del_n: ar.tem TEM29 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_n
jump final
final: ar.nand.llm LLM3 LLM3 LLM5       ;se verifică dacă este atinsă ţinta
ar.xor.llm LLM5 LLM2 LLM8
host.display LLM8 6
conv.llm.lam LLM8 LAM1
conv.lam.gam.avg LAM1 GAM1
sc.rel.equ.gam -1 GAM1 GLM1
host.display LLM6 5
jumpnc GLM1 loop
ar.or.llm LLM6 LLM11 LLM10
host.display LLM10 7
end:end

;Subrutină pentru evaluarea direcţiei optime
optdir: ar.tem TEM2 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM1
ar.tem TEM3 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM2
ar.tem TEM4 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM3
ar.tem TEM5 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM4
ar.tem TEM6 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 -1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM5
ar.tem TEM7 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM6
ar.tem TEM8 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM7
ar.tem TEM9 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM8
;Determinarea direcţiei optime de deplasare prin comparaţii succesive
sc.rel.lt.gam GAM1 GAM2 GLM1
jumpnc GLM1 2
sc.rel.lt.gam GAM1 GAM2 GLM1
jumpnc GLM1 2
sc.rel.lt.gam GAM1 GAM3 GLM1
```

```
jumpnc GLM1 3
sc.rel.lt.gam GAM1 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM1 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM1 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM1 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM1 GAM8 GLM1
jumpnc GLM1 8
jump est
2: sc.rel.lt.gam GAM2 GAM3 GLM1
jumpnc GLM1 3
sc.rel.lt.gam GAM2 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM2 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM2 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM2 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM2 GAM8 GLM1
jumpnc GLM1 8
jump vest
3: sc.rel.lt.gam GAM3 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM3 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM3 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM3 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM3 GAM8 GLM1
jumpnc GLM1 8
jump nord
4: sc.rel.lt.gam GAM4 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM4 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM4 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM4 GAM8 GLM1
jumpnc GLM1 8
jump sud
5: sc.rel.lt.gam GAM5 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM5 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM5 GAM8 GLM1
jumpnc GLM1 8
jump nordest
6: sc.rel.lt.gam GAM6 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM6 GAM8 GLM1
jumpnc GLM1 8
jump nordvest
7: sc.rel.lt.gam GAM7 GAM8 GLM1
```

```
jumpnc GLM1 8
jump sudest
8: jump sudvest
ret
```

ANEXA A2

```
;Program AMC pentru determinarea traiectoriei unui robot mobil printre obstacole prin
;alegerea direcţiei optime la fiecare pas cu template-ul PATH şi deplasarea pixel cu pixel a
;robotului.

start: host.load.tem explore.tem TEM1
host.load.tem treshold.tem TEM2
host.load.tem dilation.tem TEM3
host.load.tem path2.tem TEM4

host.load.pic mediu.bmp LLM1 BIN      ;imaginea binară a mediului cu obstacole
host.load.pic mediul.bmp LLM2 BIN     ;imaginea binară a mediului cu obstacole
host.display LLM2 1
host.load.pic tinta.bmp LLM2 BIN                    ;imaginea ţintei
host.load.pic robot.bmp LLM3 BIN                    ;poziţia de start a robotului
host.load.pic zero.bmp LLM10 BIN
host.load.pic unu.bmp     LAM1
mov.llm.llm LLM1 LLM
mov.gam.gam 0 GAM1
mov.gam.gam 1 GAM2
ar.nand.llm LLM1 LLM1 LLM1
mov.llm.llm LLM2 LLM12
ar.nand.llm LLM2 LLM2 LLM2
mov.llm.llm LLM3 LLM9

;Subrutină pentru deplasarea undei până în punctul ţintă
loop1: ar.tem TEM1 LLM1 LLM2 LAM2 GAM2 1 1 LLM1 NIL
host.display LAM2 2
ar.tem TEM2 LLM1 LAM2 LLM5 1 1 1 NIL NIL
ar.nand.llm LLM5 LLM5 LLM5
ar.and.llm LLM3 LLM5 LLM4
conv.llm.gam.cnt LLM4 GAM3
sc.rel.gt.gam GAM3 GAM1 GLM1                        ;unda a atins robotul?
sc.inc.gam GAM2
jumpnc GLM1 loop1                                   ;daca nu a atins, salt la loop1

;Subrutină pentru deplasarea robotului până la ţintă
loop2: ar.tem TEM3 LLM3 LLM3 LLM4 1 zeroflux zeroflux NIL NIL
mov.llm.llm LLM4 LLM5
ar.nand.llm LLM5 LLM5 LLM5
conv.llm.lam LLM5 LAM4
ar.add.lam LAM1 LAM4 LAM3
ar.add.lam LAM2 LAM3 LAM5
conv.lam.gam.min LAM5 GAM4
conv.gam.lam GAM4 LAM6
ar.sub.lam LAM2 LAM6 LAM6
host.display LAM6 3

;Determinarea direcţiei optime prin aplicarea template-ului PATH
ar.tem TEM4 LAM6 LLM10 LAM7 1 1 1 LLM4 NIL
host.display LAM7 4
```

```
ar.tem TEM4 LAM6 LLM10 LLM7 1 1 1 LLM4 NIL
ar.or.llm LLM9 LLM7 LLM9                    ;imaginea traiectoriei
ar.or.llm LLM9 LLM6 LLM9
host.display LLM9 5
mov.llm.llm LLM7 LLM3
ar.nand.llm LLM2 LLM2 LLM2
ar.and.llm LLM3 LLM12 LLM11
conv.llm.gam.cnt LLM11 GAM5
sc.rel.equ.gam GAM5 GAM1 GLM1               ;robotul a ajuns la ţintă?
jumpc GLM1 loop2
end:end
```

ANEXA A3

```
;Program AMC pentru planificarea simultană a traiectoriilor pentru doi roboţi mobili
;atunci când aceştia se deplasează din poziţii diferite spre o ţintă comună

start: host.load.tem explore.tem TEM1
host.load.tem shifte.tem TEM2
host.load.tem shiftv.tem TEM3
host.load.tem shiftn.tem TEM4
host.load.tem shifts.tem TEM5
host.load.tem shiften.tem TEM6
host.load.tem shiftvn.tem TEM7
host.load.tem shiftes.tem TEM8
host.load.tem shiftvs.tem TEM9
host.load.tem filwhite.tem TEM10
host.load.tem selecten.tem TEM11
host.load.tem selectes.tem TEM12
host.load.tem selectvn.tem TEM13
host.load.tem selectvs.tem TEM14
host.load.tem delen.tem TEM15
host.load.tem deles.tem TEM16
host.load.tem delvn.tem TEM17
host.load.tem delvs.tem TEM18
host.load.tem selecte.tem TEM22
host.load.tem selects.tem TEM23
host.load.tem selectv.tem TEM24
host.load.tem selectn.tem TEM25
host.load.tem dele.tem TEM26
host.load.tem dels.tem TEM27
host.load.tem delv.tem TEM28
host.load.tem deln.tem TEM29
host.load.tem dilation.tem TEM30
host.load.tem filblack.tem TEM31

host.load.pic mediu.bmp LLM1 BIN      ;imaginea mediului de lucru cu obstacole
host.display LLM1 1
ar.nand.llm LLM1 LLM1 LLM1
host.load.pic tinta.bmp LLM2 BIN                    ;imaginea ţintei
ar.nand.llm LLM2 LLM2 LLM4
host.load.pic start1.bmp LLM3 BIN     ;imaginea cu Start1 sau R1 în poziţia iniţială
host.load.pic start2.bmp LLM10 BIN    ;imaginea cu Start2 sau R2 în poziţia iniţială
mov.llm.llm LLM3 LLM6                 ;imaginea traiectoriei pentru R1
mov.llm.llm LLM10 LLM11               ;imaginea traiectoriei pentru R2
host.load.pic zero.bmp LLM16
mov.gam.gam 30 GAM10
mov.glm.glm 0 GLM4
mov.glm.glm 0 GLM5
mov.glm.glm 0 GLM6

;Evaluarea distanţelor dintre poziţiile libere din mediul de lucru şi poziţia ţintei
ar.tem TEM1 LLM1 LLM4 LAM9 200 1 1 LLM1 NIL
mov.lam.lam LAM9 LAM3
```

```
host.display LAM3 10
ar.nand.llm LLM3 LLM3 LLM9
call optdir

loop1: mov.llm.llm LLM3 LLM10                    ;salvarea imaginilor de la R2
mov.llm.llm LLM6 LLM11                           ;traiectoria pentru R2
ar.tem TEM30 LLM10 LLM10 LLM15 2 -1 -1 NIL NIL   ;unda generată de R2
ar.tem TEM31 LAM9 LAM9 LAM8 2 zeroflux zeroflux LLM15 NIL
mov.lam.lam LAM8 LAM3
host.display LAM3 5
mov.llm.llm LLM13 LLM3
mov.llm.llm LLM14 LLM6
ar.nand.llm LLM3 LLM3 LLM9
jumpnc GLM5 optdir
ar.and.llm LLM14 LLM11 LLM4
conv.llm.glm.or LLM4 GLM8
jumpnc GLM8 optdir
ar.tem TEM2 LLM3 LLM16 LLM17 1 zeroflux zeroflux NIL NIL  ;deplasare spre E
ar.tem TEM2 LLM17 LLM16 LLM17 1 zeroflux zeroflux NIL NIL ;deplasare spre E
ar.tem TEM3 LLM3 LLM16 LLM5 1 zeroflux zeroflux NIL       ;deplasare spre V
ar.tem TEM3 LLM5 LLM16 LLM5 1 zeroflux zeroflux NIL NIL   ;deplasare spre V
ar.and.llm LLM17 LLM2 LLM19
conv.llm.glm.or LLM19 GLM8
ar.and.llm LLM5 LLM2 LLM19
conv.llm.glm.or LLM19 GLM9
jumpc GLM9 v1
jumpc GLM8 e1
e1: ar.tem TEM2 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL ;deplasare spre E
ar.or.llm LLM6 LLM3 LLM6
ar.tem TEM2 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL   ;deplasare spre E
ar.or.llm LLM6 LLM3 LLM6
ar.nand.llm LLM3 LLM3 LLM9
call optdir
v1: ar.tem TEM3 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL  ;deplasare spre V
ar.or.llm LLM6 LLM3 LLM6
ar.tem TEM3 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL   ;deplasare spre V
ar.or.llm LLM6 LLM3 LLM6
ar.nand.llm LLM3 LLM3 LLM9
call optdir
loop2: mov.llm.llm LLM3 LLM13                    ;salvarea imaginilor de la R1
mov.llm.llm LLM6 LLM14                           ;traiectoria pentru R1
ar.tem TEM30 LLM13 LLM13 LLM15 2 -1 -1 NIL NIL   ;unda generată de R1
ar.tem TEM31 LAM9 LAM9 LAM8 2 zeroflux zeroflux LLM15 NIL
mov.lam.lam LAM8 LAM3
host.display LAM3 4
mov.llm.llm LLM10 LLM3
mov.llm.llm LLM11 LLM6
ar.nand.llm LLM3 LLM3 LLM9
jumpnc GLM6 optdir
ar.and.llm LLM14 LLM11 LLM4
conv.llm.glm.or LLM4 GLM8
jumpnc GLM8 optdir
ar.tem TEM2 LLM3 LLM16 LLM17 1 zeroflux zeroflux NIL NIL  ;deplasare spre E
ar.tem TEM2 LLM17 LLM16 LLM17 1 zeroflux zeroflux NIL NIL ;deplasare spre E
ar.tem TEM3 LLM3 LLM16 LLM5 1 zeroflux zeroflux NIL NIL   deplasare spre V
ar.tem TEM3 LLM5 LLM16 LLM5 1 zeroflux zeroflux NIL NIL   ;deplasare spre V
ar.and.llm LLM17 LLM2 LLM19
conv.llm.glm.or LLM19 GLM8
```

```
ar.and.llm LLM5 LLM2 LLM19
conv.llm.glm.or LLM19 GLM9
jumpc GLM9 v2
jumpc GLM8 e2
e2: ar.tem TEM2 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL ;deplasare spre E
ar.or.llm LLM6 LLM3 LLM6
ar.tem TEM2 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL   ;deplasare spre E
ar.or.llm LLM6 LLM3 LLM6
ar.nand.llm LLM3 LLM3 LLM9
call optdir
v2: ar.tem TEM3 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL ;deplasare spre V
ar.or.llm LLM6 LLM3 LLM6
ar.tem TEM3 LLM3 LLM16 LLM3 1 zeroflux zeroflux NIL NIL   ;deplasare spre V
ar.or.llm LLM6 LLM3 LLM6
ar.nand.llm LLM3 LLM3 LLM9
call optdir
nordest: ar.tem TEM11 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag1
host.display LLM6 2
jump del_en
imag1: host.display LLM6 3
del_en: ar.tem TEM15 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_en
jump cefinal
sudest: ar.tem TEM12 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag2
host.display LLM6 2
jump del_es
imag2: host.display LLM6 3
del_es: ar.tem TEM16 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_es
jump cefinal
nordvest: ar.tem TEM13 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag3
host.display LLM6 2
jump del_vn
imag3: host.display LLM6 3
del_vn: ar.tem TEM17 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_vn
jump cefinal
sudvest: ar.tem TEM14 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag4
host.display LLM6 2
jump del_vs
imag4: host.display LLM6 3
del_vs: ar.tem TEM18 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
```

```
jumpc GLM2 del_vs
jump cefinal
est: ar.tem TEM22 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag5
host.display LLM6 2
jump del_e
imag5: host.display LLM6 3
del_e: ar.tem TEM26 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_e
jump cefinal
sud: ar.tem TEM23 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag6
host.display LLM6 2
jump del_s
imag6: host.display LLM6 3
del_s: ar.tem TEM27 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_s
jump cefinal
vest: ar.tem TEM24 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag7
host.display LLM6 2
jump del_v
imag7: host.display LLM6 3
del_v: ar.tem TEM28 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_v
jump cefinal
nord: ar.tem TEM25 LAM3 LLM3 LLM7 30 1 -1 NIL NIL
ar.or.llm LLM6 LLM7 LLM6
jumpc GLM4 imag8
host.display LLM6 2
jump del_n
imag8: host.display LLM6 3
del_n: ar.tem TEM29 LLM3 LLM3 LLM3 1 -1 -1 LLM7 NIL
sc.inc.gam GAM9
sc.rel.gt.gam GAM10 GAM9 GLM2
jumpc GLM2 del_n
jump cefinal
cefinal: sc.and.glm GLM5 GLM6 GLM7
jumpc GLM7 end
mov.gam.gam 0 GAM9
sc.not.glm GLM4 GLM4
jumpnc GLM4 final2
final1: ar.tem TEM30 LLM3 LLM3 LLM12 200 -1 -1 NIL NIL    ;vecinătatea lui R1
ar.and.llm LLM12 LLM2 LLM18
ar.and.llm LLM3 LLM2 LLM19
conv.llm.glm.or LLM19 GLM5
ar.or.llm LLM12 LLM2 LLM8
host.display LLM8 8
host.display LLM6 2
```

```
jump loop2
final2: ar.tem TEM30 LLM3 LLM3 LLM12 200 -1 -1 NIL NIL    ;vecinătatea lui R2
ar.and.llm LLM12 LLM2 LLM18
ar.and.llm LLM3 LLM2 LLM19
conv.llm.glm.or LLM19 GLM6
ar.or.llm LLM12 LLM2 LLM8
host.display LLM8 9
host.display LLM6 3
jump loop1
end: end

;Subprogram pentru evaluarea direcţiei de deplasare optime
optdir: ar.tem TEM2 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM1
ar.tem TEM3 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM2
ar.tem TEM4 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM3
ar.tem TEM5 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM4
ar.tem TEM6 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 -1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM5
ar.tem TEM7 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM6
ar.tem TEM8 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM7
ar.tem TEM9 LLM9 LLM9 LLM4 1 1 1 NIL NIL
ar.tem TEM10 LLM4 LAM3 LAM2 1 1 1 LLM4 NIL
conv.lam.gam.max LAM2 GAM8

;Determinarea direcţiei optime de deplasare prin comparaţii succesive
1: sc.rel.lt.gam GAM1 GAM2 GLM1
jumpnc GLM1 2
sc.rel.lt.gam GAM1 GAM3 GLM1
jumpnc GLM1 3
sc.rel.lt.gam GAM1 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM1 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM1 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM1 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM1 GAM8 GLM1
jumpnc GLM1 8
jump est
2: sc.rel.lt.gam GAM2 GAM3 GLM1
jumpnc GLM1 3
sc.rel.lt.gam GAM2 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM2 GAM5 GLM1
```

```
jumpnc GLM1 5
sc.rel.lt.gam GAM2 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM2 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM2 GAM8 GLM1
jumpnc GLM1 8
jump vest
3: sc.rel.lt.gam GAM3 GAM4 GLM1
jumpnc GLM1 4
sc.rel.lt.gam GAM3 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM3 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM3 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM3 GAM8 GLM1
jumpnc GLM1 8
jump nord
4: sc.rel.lt.gam GAM4 GAM5 GLM1
jumpnc GLM1 5
sc.rel.lt.gam GAM4 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM4 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM4 GAM8 GLM1
jumpnc GLM1 8
jump sud
5: sc.rel.lt.gam GAM5 GAM6 GLM1
jumpnc GLM1 6
sc.rel.lt.gam GAM5 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM5 GAM8 GLM1
jumpnc GLM1 8
jump nordest
6: sc.rel.lt.gam GAM6 GAM7 GLM1
jumpnc GLM1 7
sc.rel.lt.gam GAM6 GAM8 GLM1
jumpnc GLM1 8
jump nordvest
7: sc.rel.lt.gam GAM7 GAM8 GLM1
jumpnc GLM1 8
jump sudest
8: jump sudvest
ret
```

ANEXA A4

```
;Program AMC pentru simularea deplasării unui robot mobil într-un labirint prin
;metoda wall- following pe lângă peretele din dreapta apoi pe lângă cel din stânga

start: host.load.tem shifte.tem TEM1
host.load.tem shifts.tem TEM2
host.load.tem shiftv.tem TEM3
host.load.tem shiftn.tem TEM4
host.load.tem shiftes.tem TEM5
host.load.tem shiftvs.tem TEM6
host.load.tem shiftvn.tem TEM7
host.load.tem shiften.tem TEM8
host.load.pic rob.bmp LLM1              ;imagine cu poziţia iniţială a robotului
host.load.pic rob.bmp LLM11
host.load.pic zero.bmp LLM2
host.load.pic obst1.bmp LLM3            ;imaginea binară a labirintului
host.load.pic tinta.bmp LLM6            ;imaginea ţintei
host.load.pic gol.bmp LLM10
host.load.pic gol.bmp LLM12
ar.or.llm LLM1 LLM3 LLM8
host.display LLM8 1
conv.llm.gam.cnt LLM3 GAM1
mov.glm.glm 1 GLM9

;Subrutină pentru verificarea poziţiilor din jurul poziţiei curente a robotului
loop: ar.tem TEM1 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL  ;poziţia spre E
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM1
ar.tem TEM2 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre S
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM2
ar.tem TEM3 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre V
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM3
ar.tem TEM4 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre N
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM4
ar.tem TEM5 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre SE
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM5
ar.tem TEM6 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre SV
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM6
ar.tem TEM7 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL    ;poziţia spre NV
ar.or.llm LLM3 LLM4 LLM5
```

```
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM7
ar.tem TEM8 LLM1 LLM2 LLM4 1 zeroflux zeroflux NIL NIL     ;poziţia spre NE
ar.or.llm LLM3 LLM4 LLM5
conv.llm.gam.cnt LLM5 GAM2
sc.rel.equ.gam GAM2 GAM1 GLM8
jumpc GLM11 stg

;Logica de deplasare a robotului pe lângă peretele din dreapta
sc.and.glm GLM1 GLM4 GLM10
jumpc GLM10 vest
sc.and.glm GLM1 GLM2 GLM10
jumpc GLM10 nord
sc.and.glm GLM1 GLM9 GLM10
jumpc GLM10 nord
sc.and.glm GLM3 GLM2 GLM10
jumpc GLM10 est
sc.and.glm GLM3 GLM4 GLM10
jumpc GLM10 sud
sc.and.glm GLM3 GLM9 GLM10
jumpc GLM10 sud
sc.and.glm GLM4 GLM9 GLM10
jumpc GLM10 vest
sc.and.glm GLM2 GLM9 GLM10
jumpc GLM10 est
sc.and.glm GLM5 GLM9 GLM10
jumpc GLM10 est
sc.and.glm GLM6 GLM9 GLM10
jumpc GLM10 sud
sc.and.glm GLM7 GLM9 GLM10
jumpc GLM10 vest
sc.and.glm GLM8 GLM9 GLM10
jumpc GLM10 nord

;Logica de deplasare a robotului pe lângă peretele din stânga
stg: sc.and.glm GLM1 GLM4 GLM10
jumpc GLM10 sud
sc.and.glm GLM1 GLM2 GLM10
jumpc GLM10 vest
sc.and.glm GLM1 GLM9 GLM10
jumpc GLM10 sud
sc.and.glm GLM3 GLM2 GLM10
jumpc GLM10 nord
sc.and.glm GLM3 GLM4 GLM10
jumpc GLM10 est
sc.and.glm GLM3 GLM9 GLM10
jumpc GLM10 nord
sc.and.glm GLM4 GLM9 GLM10
jumpc GLM10 est
sc.and.glm GLM2 GLM9 GLM10
jumpc GLM10 vest
sc.and.glm GLM5 GLM9 GLM10
jumpc GLM10 sud
sc.and.glm GLM6 GLM9 GLM10
jumpc GLM10 vest
sc.and.glm GLM7 GLM9 GLM10
jumpc GLM10 nord
sc.and.glm GLM8 GLM9 GLM10
```

```
jumpc GLM10 est
est: ar.tem TEM1 LLM1 LLM2 LLM7 1 zeroflux zeroflux NIL NIL          ;deplasare E
ar.or.llm LLM7 LLM3 LLM8
host.display LLM8 2
ar.or.llm LLM10 LLM7 LLM10
mov.llm.llm LLM7 LLM1
jumpc GLM11 fin2
jump fin1
sud: ar.tem TEM2 LLM1 LLM2 LLM7 1 zeroflux zeroflux NIL NIL          ;deplasare S
ar.or.llm LLM7 LLM3 LLM8
host.display LLM8 2
ar.or.llm LLM10 LLM7 LLM10
mov.llm.llm LLM7 LLM1
jumpc GLM11 fin2
jump fin1
vest: ar.tem TEM3 LLM1 LLM2 LLM7 1 zeroflux zeroflux NIL NIL         ;deplasare V
ar.or.llm LLM7 LLM3 LLM8
host.display LLM8 2
ar.or.llm LLM10 LLM7 LLM10
mov.llm.llm LLM7 LLM1
jumpc GLM11 fin2
jump fin1
nord: ar.tem TEM4 LLM1 LLM2 LLM7 1 zeroflux zeroflux NIL NIL         ;deplasare N
ar.or.llm LLM7 LLM3 LLM8
host.display LLM8 2
ar.or.llm LLM10 LLM7 LLM10
mov.llm.llm LLM7 LLM1
jumpc GLM11 fin2
jump fin1
fin1: ar.and.llm LLM1 LLM6 LLM9
conv.llm.gam.cnt LLM9 GAM3
sc.rel.equ.gam 1 GAM3 GLM11
jumpnc GLM11 loop
host.display LLM10 3      ;imaginea traiectoriei pe lângă peretele din dreapta
mov.llm.llm LLM11 LLM1
mov.llm.llm LLM12 LLM10
jump loop
fin2: ar.and.llm LLM1 LLM6 LLM9
conv.llm.gam.cnt LLM9 GAM3
sc.rel.equ.gam 1 GAM3 GLM12
jumpnc GLM12 loop
host.display LLM10 5      ;imaginea traiectoriei pe lângă peretele din stânga
end:end
```

ANEXA A5

```
;Program AMC pentru determinarea robotului sau roboţilor cu poziţie centrală apoi cu
;poziţie extremă, dintr-o colectivitate compusă din patru roboţi, prin estimarea distanţelor în
;ordine crescătoare

start:host.load.tem dilation.tem TEM1
host.load.pic rob1.bmp LLM1                          ;imagine R1
host.load.pic rob2.bmp LLM2                          ;imagine R2
host.load.pic rob3.bmp LLM3                          ;imagine R3
host.load.pic rob4.bmp LLM4                          ;imagine R4
host.load.pic alb.bmp LLM8
host.load.pic insr1.bmp LLM21
host.load.pic insr2.bmp LLM22
host.load.pic insr3.bmp LLM23
host.load.pic insr4.bmp LLM24
host.load.pic centrali.bmp LLM25
host.load.pic extremi.bmp LLM26

mov.llm.llm LLM1 LLM11
mov.llm.llm LLM2 LLM12
mov.llm.llm LLM3 LLM13
mov.llm.llm LLM4 LLM14
ar.or.llm LLM11 LLM8 LLM8
ar.or.llm LLM12 LLM8 LLM8
ar.or.llm LLM13 LLM8 LLM8
ar.or.llm LLM14 LLM8 LLM8
host.display LLM8 1                                  ;afişare toţi roboţii

vecR1: ar.tem TEM1 LLM1 LLM1 LLM5 2 -1 -1 NIL NIL    ;vecinătate R1
ar.or.llm LLM8 LLM5 LLM9
ar.or.llm LLM9 LLM21 LLM9
host.display LLM9 2
ar.and.llm LLM5 LLM12 LLM6
ar.and.llm LLM5 LLM13 LLM7
ar.and.llm LLM5 LLM14 LLM10
mov.llm.llm LLM5 LLM1
jumpnc GLM15 vecR12
jumpnc GLM16 vecR13
jumpnc GLM17 vecR14
vecR12: conv.llm.gam.cnt LLM6 GAM11
sc.rel.equ.gam 1 GAM11 GLM15
sc.inc.gam GAM1                                      ;r12=r21
jumpnc GLM16 vecR13
jumpnc GLM17 vecR14
jump endR1
vecR13: conv.llm.gam.cnt LLM7 GAM11
sc.rel.equ.gam 1 GAM11 GLM16
sc.inc.gam GAM2                                      ;r13=r31
jumpnc GLM17 vecR14
jump endR1
vecR14: conv.llm.gam.cnt LLM10 GAM11
```

```
sc.rel.equ.gam 1 GAM11 GLM17
sc.inc.gam GAM3                                    ;r14=r41
endR1: sc.and.glm GLM15 GLM16 GLM18
sc.and.glm GLM18 GLM17 GLM18
jumpnc GLM18 vecR1
mov.glm.glm 0 GLM16
mov.glm.glm 0 GLM17
vecR2: ar.tem TEM1 LLM2 LLM2 LLM5 2 -1 -1 NIL NIL        ;vecinătate R2
ar.or.llm LLM8 LLM5 LLM9
ar.or.llm LLM9 LLM22 LLM9
host.display LLM9 3
ar.and.llm LLM5 LLM13 LLM6
ar.and.llm LLM5 LLM14 LLM7
mov.llm.llm LLM5 LLM2
jumpnc GLM16 vecR23
jumpnc GLM17 vecR24
vecR23: conv.llm.gam.cnt LLM6 GAM11
sc.rel.equ.gam 1 GAM11 GLM16
sc.inc.gam GAM5                                    ;r23=r32
jumpnc GLM17 vecR24
jump endR2
vecR24: conv.llm.gam.cnt LLM7 GAM11
sc.rel.equ.gam 1 GAM11 GLM17
sc.inc.gam GAM6                                    ;r24=r42
endR2: sc.and.glm GLM16 GLM17 GLM18
jumpnc GLM18 vecR2
vecR3: ar.tem TEM1 LLM3 LLM3 LLM5 2 -1 -1 NIL NIL        ;vecinătate R3
ar.or.llm LLM8 LLM5 LLM9
ar.or.llm LLM9 LLM23 LLM9
host.display LLM9 4
ar.and.llm LLM5 LLM14 LLM6
mov.llm.llm LLM5 LLM3
conv.llm.gam.cnt LLM6 GAM11
sc.rel.equ.gam 1 GAM11 GLM17
sc.inc.gam GAM8                                    ;r34=r43
jumpnc GLM17 vecR3
sc.add.gam GAM1 GAM2 GAM4                          ;r12+r13
sc.add.gam GAM4 GAM3 GAM4                          ;N1=r12+r13+r14
sc.add.gam GAM1 GAM5 GAM7                          ;r21+r23
sc.add.gam GAM7 GAM6 GAM7                          ;N2=r21+r23+r24
sc.add.gam GAM2 GAM5 GAM9                          ;r31+r32
sc.add.gam GAM9 GAM8 GAM9                          ;N3=r31+r32+r34
sc.add.gam GAM3 GAM6 GAM10                         ;r41+r42
sc.add.gam GAM10 GAM8 GAM10                        ;N4=r41+r42+r43
sc.rel.lt.gam GAM4 GAM7 GLM1                       ;N1<N2
sc.rel.lt.gam GAM7 GAM4 GLM2                       ;N2<N1
sc.rel.lt.gam GAM4 GAM9 GLM3                       ;N1<N3
sc.rel.lt.gam GAM9 GAM4 GLM4                       ;N3<N1
sc.rel.lt.gam GAM4 GAM10 GLM5                      ;N1<N4
sc.rel.lt.gam GAM10 GAM4 GLM6                      ;N4<N1
sc.rel.lt.gam GAM7 GAM9 GLM7                       ;N2<N3
sc.rel.lt.gam GAM9 GAM7 GLM8                       ;N3<N2
sc.rel.lt.gam GAM7 GAM10 GLM9                      ;N2<N4
sc.rel.lt.gam GAM10 GAM7 GLM10                     ;N4<N2
sc.rel.lt.gam GAM9 GAM10 GLM11                     ;N3<N4
sc.rel.lt.gam GAM10 GAM9 GLM12                     ;N4<N3
```

```
;Verificare dacă există un robot cu poziţie centrală
sc.and.glm GLM1 GLM3 GLM13                ;N1<N2,N3
sc.and.glm GLM5 GLM13 GLM13               ;N1<N2,N3,N4
jumpc GLM13 R1                            ;R1 central
sc.and.glm GLM2 GLM7 GLM13                ;N2<N1,N3
sc.and.glm GLM9 GLM13 GLM13               ;N2<N1,N3,N4
jumpc GLM13 R2                            ;R2 central
sc.and.glm GLM4 GLM8 GLM13                ;N3<N1,N2
sc.and.glm GLM11 GLM13 GLM13              ;N3<N1,N2,N4
jumpc GLM13 R3                            ;R3 central
sc.and.glm GLM6 GLM10 GLM13               ;N4<N1,N2
sc.and.glm GLM12 GLM13 GLM13              ;N4<N1,N2,N3
jumpc GLM13 R4                            ;R4 central
jump cent2
R1: ar.or.llm LLM11 LLM25 LLM15
host.display LLM15 6                      ;afişare R1
jump ext
R2: ar.or.llm LLM12 LLM25 LLM15
host.display LLM15 6                      ;afişare R2
jump ext
R3: ar.or.llm LLM13 LLM25 LLM15
host.display LLM15 6                      ;afişare R3
jump ext
R4: ar.or.llm LLM14 LLM25 LLM15
host.display LLM15 6                      ;afişare R4
jump ext

;Verificare dacă există 2 roboţi cu poziţii centrale
cent2: sc.nor.glm GLM1 GLM2 GLM13         ;N1=N2
sc.and.glm GLM13 GLM3 GLM13               ;N1=N2<N3
sc.and.glm GLM13 GLM5 GLM13               ;N1=N2<N3,N4
jumpc GLM13 R12                           ;R1 şi R2 centrali
sc.nor.glm GLM3 GLM4 GLM13                ;N1=N3
sc.and.glm GLM13 GLM1 GLM13               ;N1=N3<N2
sc.and.glm GLM13 GLM5 GLM13               ;N1=N3<N2,N4
jumpc GLM13 R13                           ;R1 şi R3 centrali
sc.nor.glm GLM5 GLM6 GLM13                ;N1=N4
sc.and.glm GLM13 GLM1 GLM13               ;N1=N4<N2
sc.and.glm GLM13 GLM3 GLM13               ;N1=N4<N2,N3
jumpc GLM13 R14                           ;R1 şi R4 centrali
sc.nor.glm GLM7 GLM8 GLM13                ;N2=N3
sc.and.glm GLM13 GLM2 GLM13               ;N2=N3<N1
sc.and.glm GLM13 GLM9 GLM13               ;N2=N3<N1,N4
jumpc GLM13 R23                           ;R2 şi R3 centrali
sc.nor.glm GLM9 GLM10 GLM13               ;N2=N4
sc.and.glm GLM13 GLM2 GLM13               ;N2=N4<N1
sc.and.glm GLM13 GLM7 GLM13               ;N2=N4<N1,N3
jumpc GLM13 R24                           ;R2 şi R4 centrali
sc.nor.glm GLM11 GLM12 GLM13              ;N3=N4
sc.and.glm GLM13 GLM4 GLM13               ;N3=N4<N1
sc.and.glm GLM13 GLM8 GLM13               ;N3=N4<N1,N2
jumpc GLM13 R34                           ;R3 şi R4 centrali
jump cent3
R12: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                      ;afişare R1 şi R2
jump ext
R13: ar.or.llm LLM11 LLM13 LLM15
```

```
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R1 şi R3
jump ext
R14: ar.or.llm LLM11 LLM14 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R1 şi R4
jump ext
R23: ar.or.llm LLM12 LLM13 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R2 şi R3
jump ext
R24: ar.or.llm LLM12 LLM14 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R2 şi R4
jump ext
R34: ar.or.llm LLM13 LLM14 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R3 şi R4
jump ext

;Verificare dacă există 3 roboţi cu poziţii centrale
cent3: sc.nor.glm GLM1 GLM2 GLM13       ;N1=N2
sc.nor.glm GLM3 GLM4 GLM14              ;N1=N3
sc.and.glm GLM13 GLM14 GLM13            ;N1=N2=N3
sc.and.glm GLM13 GLM5 GLM13             ;N1=N2=N3<N4
jumpc GLM13 R123                        ;R1, R2 şi R3 centrali
sc.nor.glm GLM1 GLM2 GLM13              ;N1=N2
sc.nor.glm GLM5 GLM6 GLM14              ;N1=N4
sc.and.glm GLM13 GLM14 GLM13            ;N1=N2=N4
sc.and.glm GLM13 GLM3 GLM13             ;N1=N2=N4<N3
jumpc GLM13 R124                        ;R1, R2 şi R4 centrali
sc.nor.glm GLM3 GLM4 GLM13              ;N1=N3
sc.nor.glm GLM5 GLM6 GLM14              ;N1=N4
sc.and.glm GLM13 GLM14 GLM13            ;N1=N3=N4
sc.and.glm GLM13 GLM1 GLM13             ;N1=N3=N4<N2
jumpc GLM13 R134                        ;R1, R3 şi R4 centrali
sc.nor.glm GLM7 GLM8 GLM13              ;N2=N3
sc.nor.glm GLM9 GLM10 GLM14             ;N2=N4
sc.and.glm GLM13 GLM14 GLM13            ;N2=N3=N4
sc.and.glm GLM13 GLM2 GLM13             ;N2=N3=N4<N1
jumpc GLM13 R234                        ;R2, R3 şi R4 centrali
R123: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM13 LLM15 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R1, R2 şi R3
jump ext
R124: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM14 LLM15 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R1, R2 şi R4
jump ext
R134: ar.or.llm LLM11 LLM13 LLM15
ar.or.llm LLM14 LLM15 LLM15
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                    ;afişare R1, R3 şi R4
jump ext
R234: ar.or.llm LLM12 LLM13 LLM15
ar.or.llm LLM14 LLM15 LLM15
```

```
ar.or.llm LLM15 LLM25 LLM15
host.display LLM15 6                          ;afişare R2, R3 şi R4
jump ext
ext: sc.rel.gt.gam GAM4 GAM7 GLM1             ;N1>N2
sc.rel.gt.gam GAM7 GAM4 GLM2                  ;N2>N1
sc.rel.gt.gam GAM4 GAM9 GLM3                  ;N1>N3
sc.rel.gt.gam GAM9 GAM4 GLM4                  ;N3>N1
sc.rel.gt.gam GAM4 GAM10 GLM5                 ;N1>N4
sc.rel.gt.gam GAM10 GAM4 GLM6                 ;N4>N1
sc.rel.gt.gam GAM7 GAM9 GLM7                  ;N2>N3
sc.rel.gt.gam GAM9 GAM7 GLM8                  ;N3>N2
sc.rel.gt.gam GAM7 GAM10 GLM9                 ;N2>N4
sc.rel.gt.gam GAM10 GAM7 GLM10                ;N4>N2
sc.rel.gt.gam GAM9 GAM10 GLM11                ;N3>N4
sc.rel.gt.gam GAM10 GAM9 GLM12                ;N4>N3
;Verificare dacă există un robot cu poziţie extremă
sc.and.glm GLM1 GLM3 GLM13                    ;N1>N2,N3
sc.and.glm GLM5 GLM13 GLM13                   ;N1>N2,N3,N4
jumpc GLM13 R1e                               ;R1 extremă
sc.and.glm GLM2 GLM7 GLM13                    ;N2>N1,N3
sc.and.glm GLM9 GLM13 GLM13                   ;N2>N1,N3,N4
jumpc GLM13 R2e                               ;R2 extremă
sc.and.glm GLM4 GLM8 GLM13                    ;N3>N1,N2
sc.and.glm GLM11 GLM13 GLM13                  ;N3>N1,N2,N4
jumpc GLM13 R3e                               ;R3 extremă
sc.and.glm GLM6 GLM10 GLM13                   ;N4>N1,N2
sc.and.glm GLM12 GLM13 GLM13                  ;N4>N1,N2,N3
jumpc GLM13 R4e                               ;R4 extremă
jump ext2
R1e: ar.or.llm LLM11 LLM26 LLM15
host.display LLM15 7                          ;afişare R1
jump end
R2e: ar.or.llm LLM12 LLM26 LLM15
host.display LLM15 7                          ;afişare R2
jump end
R3e: ar.or.llm LLM13 LLM26 LLM15
host.display LLM15 7                          ;afişare R3
jump end
R4e: ar.or.llm LLM14 LLM26 LLM15
host.display LLM15 7                          ;afişare R4
jump end

;Verificare dacă există 2 roboţi cu poziţii extreme
ext2: sc.nor.glm GLM1 GLM2 GLM13              ;N1=N2
sc.and.glm GLM13 GLM3 GLM13                   ;N1=N2>N3
sc.and.glm GLM13 GLM5 GLM13                   ;N1=N2>N4
jumpc GLM13 R12e                              ;R1 şi R2 extremi
sc.nor.glm GLM3 GLM4 GLM13                    ;N1=N3
sc.and.glm GLM13 GLM1 GLM13                   ;N1=N3>N2
sc.and.glm GLM13 GLM5 GLM13                   ;N1=N3>N4
jumpc GLM13 R13e                              ;R1 şi R3 extremi
sc.nor.glm GLM5 GLM6 GLM13                    ;N1=N4
sc.and.glm GLM13 GLM1 GLM13                   ;N1=N4>N2
sc.and.glm GLM13 GLM3 GLM13                   ;N1=N4>N3
jumpc GLM13 R14e                              ;R1 şi R4 extremi
sc.nor.glm GLM7 GLM8 GLM13                    ;N2=N3
sc.and.glm GLM13 GLM2 GLM13                   ;N2=N3>N1
sc.and.glm GLM13 GLM9 GLM13                   ;N2=N3>N4
```

```
jumpc GLM13 R23e                          ;R2 şi R3 extremi
sc.nor.glm GLM9 GLM10 GLM13               ;N2=N4
sc.and.glm GLM13 GLM2 GLM13               ;N2=N4>N1
sc.and.glm GLM13 GLM7 GLM13               ;N2=N4>N3
jumpc GLM13 R24e                          ;R2 şi R4 extremi
sc.nor.glm GLM11 GLM12 GLM13              ;N3=N4
sc.and.glm GLM13 GLM4 GLM13               ;N3=N4>N1
sc.and.glm GLM13 GLM8 GLM13               ;N3=N4>N2
jumpc GLM13 R34e                          ;R3 şi R4 extremi
jump ext3
R12e: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                      ;afişare R1 şi R2
jump end
R13e: ar.or.llm LLM11 LLM13 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                      ;afişare R1 şi R3
jump end
R14e: ar.or.llm LLM11 LLM14 LLM15
ar.or.llm LLM15 LLM26 LLM15               ;afişare R1 şi R4
jump end
R23e: ar.or.llm LLM12 LLM13 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                      ;afişare R2 şi R3
jump end
R24e: ar.or.llm LLM12 LLM14 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                      ;afişare R2 şi R4
jump end
R34e: ar.or.llm LLM13 LLM14 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                      ;afişare R3 şi R4
jump end

;Verificare dacă există 3 roboţi cu poziţii extreme
ext3: sc.nor.glm GLM1 GLM2 GLM13          ;N1=N2
sc.nor.glm GLM3 GLM4 GLM14                ;N1=N3
sc.and.glm GLM13 GLM14 GLM13              ;N1=N2=N3
sc.and.glm GLM13 GLM5 GLM13               ;N1=N2=N3>N4
jumpc GLM13 R123e                         ;R1, R2 şi R3 extremi
sc.nor.glm GLM1 GLM2 GLM13                ;N1=N2
sc.nor.glm GLM5 GLM6 GLM14                ;N1=N4
sc.and.glm GLM13 GLM14 GLM13              ;N1=N2=N4
sc.and.glm GLM13 GLM3 GLM13               ;N1=N2=N4>N3
jumpc GLM13 R124e                         ;R1, R2 şi R4 extremi
sc.nor.glm GLM3 GLM4 GLM13                ;N1=N3
sc.nor.glm GLM5 GLM6 GLM14                ;N1=N4
sc.and.glm GLM13 GLM14 GLM13              ;N1=N3=N4
sc.and.glm GLM13 GLM1 GLM13               ;N1=N3=N4>N2
jumpc GLM13 R134e                         ;R1, R3 şi R4 extremi
sc.nor.glm GLM7 GLM8 GLM13                ;N2=N3
sc.nor.glm GLM9 GLM10 GLM14               ;N2=N4
sc.and.glm GLM13 GLM14 GLM13              ;N2=N3=N4
sc.and.glm GLM13 GLM2 GLM13               ;N2=N3=N4>N1
jumpc GLM13 R234e                         ;R2, R3 şi R4 extremi
R234e: ar.or.llm LLM12 LLM13 LLM15
ar.or.llm LLM14 LLM15 LLM15
ar.or.llm LLM15 LLM26 LLM15
```

```
host.display LLM15 7                    ;afişare R1, R3 şi R4
jump end
R134e: ar.or.llm LLM11 LLM13 LLM15
ar.or.llm LLM14 LLM15 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                    ;afişare R1, R3 şi R4
jump end
R124e: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM14 LLM15 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                    ;afişare R1, R2 şi R4
jump end
R123e: ar.or.llm LLM11 LLM12 LLM15
ar.or.llm LLM13 LLM15 LLM15
ar.or.llm LLM15 LLM26 LLM15
host.display LLM15 7                    ;afişare R1, R2 şi R3
jump end
end:end
```

ANEXA A6

```
;Program AMC pentru gruparea a doi roboţi mobili extremă (R2, R3) în jurul robotului
;central (R1)

host.load.tem shifte.tem TEM1
host.load.tem shifts.tem TEM2
host.load.tem shiftv.tem TEM3
host.load.tem shiftn.tem TEM4
host.load.tem shiftes.tem TEM5
host.load.tem shiftvs.tem TEM6
host.load.tem shiftvn.tem TEM7
host.load.tem shiften.tem TEM8
host.load.tem dilation.tem TEM9

host.load.pic rob1.bmp LLM1                          ;imagine R1
host.load.pic rob2.bmp LLM2                          ;imagine R2
host.load.pic rob3.bmp LLM3                          ;imagine R3
host.load.pic zero.bmp LLM4          ;imagine cu pixelii de valoare zero
mov.llm.llm LLM1 LLM10
mov.llm.llm LLM2 LLM15                               ;traiectorie R2
mov.llm.llm LLM3 LLM16                               ;traiectorie R3
ar.or.llm LLM1 LLM2 LLM13                                  ;imagine R1 şi R2
ar.or.llm LLM13 LLM3 LLM13                           ;imagine R1, R2 şi R3
host.display LLM13 1                                 ;afişare toţi roboţii
mov.gam.gam 5 GAM1                                   ;r12=5
mov.gam.gam 9 GAM2                                   ;r13=9
mov.gam.gam 0 GAM3
sc.rel.gt.gam GAM1 GAM2 GLM1                         ; r12>r13?
jumpc GLM1 vecR12
jump vecR13

;Vecinătate R1 apoi verificare poziţii din jurul lui R3 şi deplasare cu un pas
vecR13: mov.gam.gam GAM2 GAM4                        ;r13
sc.dec.gam GAM4                                      ;r=r-1
sc.rel.equ.gam 0 GAM4 GLM1                           ;r=0?
jumpc GLM1 end                                       ;dacă r=0 STOP
ar.tem TEM9 LLM1 LLM1 LLM5 2 -1 -1 NIL NIL           ;generare vecinătate R1
mov.llm.llm LLM5 LLM1                                ;actualizare R1
ar.or.llm LLM1 LLM2 LLM14
ar.or.llm LLM14 LLM3 LLM14
conv.llm.gam.cnt LLM1 GAM5           ;nr. de pixeli negri din imag vec. R1
sc.inc.gam GAM3
sc.rel.equ.gam GAM3 GAM4 GLM2
jumpnc GLM2 vecR13
host.display LLM14 2
ar.tem TEM1 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL     ;deplasare E
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 sud3
jump final3
```

```
sud3: ar.tem TEM2 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL        ;deplasare S
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 vest3
jump final3
vest3: ar.tem TEM3 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL       ;deplasare V
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nord3
jump final3
nord3: ar.tem TEM4 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL       ;deplasare N
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 sudest3
jump final3
sudest3: ar.tem TEM5 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL     ;deplasare SE
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM5 sudvest3
jump final3
sudvest3: ar.tem TEM6 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL    ;deplasare SV
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nordvest3
jump final3
nordvest3: ar.tem TEM7 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL  ;deplasare NV
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nordest3
jump final3
nordest3:ar.tem TEM8 LLM3 LLM4 LLM6 1 zeroflux zeroflux NIL NIL    ;deplasare NE
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM8 final3
jump final3

;Vecinătate R1 apoi verificare poziţii din jurul lui R2 şi deplasare R2 cu un pas
vecR12: mov.gam.gam GAM1 GAM4                    ;r12
sc.dec.gam GAM4                                  ;r=r-1
sc.rel.equ.gam 0 GAM4 GLM1                       ;r=0?
jumpc GLM1 end                                   ;dacă r=0 STOP
ar.tem TEM9 LLM1 LLM1 LLM5 2 -1 -1 NIL NIL       ;generare vecinătate R1
mov.llm.llm LLM5 LLM1                            ;actualizare R1
ar.or.llm LLM1 LLM2 LLM14
ar.or.llm LLM14 LLM3 LLM14
conv.llm.gam.cnt LLM1 GAM5
sc.inc.gam GAM3
sc.rel.equ.gam GAM3 GAM4 GLM2
jumpnc GLM2 vecR12
host.display LLM14 2
ar.tem TEM1 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL    ;deplasare E
```

```
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 sud2
jump final2
sud2: ar.tem TEM2 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL          ;deplasare S
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 vest2
jump final2
vest2: ar.tem TEM3 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL         ;deplasare V
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nord2
jump final2
nord2: ar.tem TEM4 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL         ;deplasare N
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 sudest2
jump final2
sudest2: ar.tem TEM5 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL       ;deplasare SE
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM5 sudvest2
jump final2
sudvest2: ar.tem TEM6 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL     ;deplasare SV
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nordvest2
jump final2
nordvest2: ar.tem TEM7 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL    ;deplasare NV
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM1 nordest2
jump final2
nordest2: ar.tem TEM8 LLM2 LLM4 LLM6 1 zeroflux zeroflux NIL NIL     ;deplasare NE
ar.or.llm LLM1 LLM6 LLM7
conv.llm.gam.cnt LLM7 GAM6
sc.rel.equ.gam GAM5 GAM6 GLM1
jumpnc GLM8 final2
jump final2
final3: mov.llm.llm LLM10 LLM1
ar.or.llm LLM6 LLM16 LLM16                    ;actualizare traiectorie R3
ar.or.llm LLM16 LLM15 LLM11
ar.or.llm LLM11 LLM1 LLM11
host.display LLM11 3
mov.llm.llm LLM6 LLM3
sc.rel.equ.gam GAM1 GAM4 GLM1                 ;r12=r13?
sc.rel.gt.gam GAM4 GAM1 GLM2                  ;r13>r12?
mov.gam.gam GAM4 GAM2                         ;r13 actualizat
mov.gam.gam 0 GAM3
jumpc GLM1 vecR12
```

```
jumpc GLM2 vecR13
jump vecR12
final2: mov.llm.llm LLM10 LLM1                ;actualizare imag R1
ar.or.llm LLM6 LLM15 LLM15
ar.or.llm LLM15 LLM16 LLM12
ar.or.llm LLM12 LLM1 LLM12
host.display LLM12 3
mov.llm.llm LLM6 LLM2
sc.rel.equ.gam GAM2 GAM4 GLM1                 ;r13=r12?
sc.rel.gt.gam GAM4 GAM2 GLM2                  ;r12>r13?
mov.gam.gam GAM4 GAM1                         ;r12 actualizat
mov.gam.gam 0 GAM3
jumpc GLM1 vecR13
jumpc GLM2 vecR12
jump vecR13
end:end
```

ANEXA A7

```
% Program Matlab pentru planificarea traiectorie unui robot mobil pe baza imaginilor
% mediului captate cu o cameră video şi procesarea acestora cu CNN

VFM('preview');                                  % urmărire online
VFM('configformat');                             % setare rezoluţie
VFM('configsource');                             % reglaje cameră
I=VFM('grab');                                   % captare imagine color

% Transformarea imaginii color a mediului în imagine gray-scale
GRI=rgb2gray(I); imwrite(GRI,'mediu.bmp'); Me=imread('mediu.bmp');
subplot(231); imShow(Me); Xlabel('Imaginea gray-scale a mediului');

% Procesare CNN pentru obţinerea poziţiei robotului
Ro=I(:,:,1); imwrite(Ro,'rosu.bmp'); SetEnv;
mCNN.INPUT1=lbmp2cnn('rosu.bmp','gray'); mCNN.STATE=mCNN.INPUT1;
LOADTEM('THRES1'); RunTem;
scnn2bmp('pozrob.bmp', mCNN.OUTPUT); R=imread('pozrob.bmp');
subplot(232); imshow(R); Xlabel('Pozitia robotului');

% Procesare CNN pentru obţinerea poziţiei ţintei
Ve=I(:,:,2); imwrite(Ve,'verde.bmp');
mCNN.INPUT1=lbmp2cnn('verde.bmp','gray'); mCNN.STATE=mCNN.INPUT1;
LOADTEM('THRES1'); RunTem;
scnn2bmp('poztin.bmp', mCNN.OUTPUT); T=imread('poztin.bmp');
subplot(233); imshow(T); Xlabel('Pozitia tintei');

% Procesare CNN pentru obţinerea imaginii binare a mediului
mCNN.INPUT1=lbmp2cnn('mediu.bmp','gray'); mCNN.STATE=mCNN.INPUT1;
LOADTEM('THRES'); RunTem;
scnn2bmp('mediubin.bmp', mCNN.OUTPUT); Me1=imread('mediubin.bmp');
subplot(234); imshow(Me1); Xlabel('Imaginea binara');

% Procesare CNN pentru eliminarea zgomotelor din imaginea binară a mediului
mCNN.INPUT1=lbmp2cnn('mediubin.bmp','gray'); mCNN.STATE=mCNN.INPUT1;
LOADTEM('EROSION'); RunTem;
scnn2bmp('mediuero.bmp', mCNN.OUTPUT); Me2=imread('mediuero.bmp');
subplot(235); imshow(Me2); Xlabel('Erodare');

% Procesare CNN pentru dilatare obstacole
i=1;j=0;
while i<4;
   if j==0;    mCNN.INPUT1=lbmp2cnn('mediuero.bmp','gray');
   else        mCNN.INPUT1=mCNN.OUTPUT; end;
   mCNN.STATE=lbmp2cnn('zero1.bmp', 'gray');
   mCNN.TimeStep=1; mCNN.IterNum=1; mCNN.Boundary=2;
   LOADTEM('DILATION'); RunTem;i=i+1;j=j+1; end;
scnn2bmp('mediudil.bmp',mCNN.OUTPUT); M=imread('mediudil.bmp');
subplot(236); imshow(M); Xlabel('Dilatare obstacole');
```

```
% Determinarea coloanei şi liniei pe care se află robotul şi ţinta
[A,B]=max(R); [A,colR]=max(A); P=rot90(R,1); [A,B]=max(P); [A,linR]=max(A);
[A,B]=max(T); [A,colT]=max(A); P=rot90(T,1); [A,B]=max(P); [A,linT]=max(A);
linR=100; colR=15; linT = 40; colT = 130;

%Afişarea imaginii mediului cu obstacole
[m,n]=size(M);
for i=1:1:n; for j=1:1:m; if M(j,i)==255; mediul(j,i)=0;  %inversare imagine mediu
else mediul(j,i)=255; end; end; end;
figure, surf(double(mediul(1:1:m,1:1:n))); hold on; grid on;
axis([0 160 0 120 0 300]); yLabel('j'); xLabel('i');
set(gca,'YDir','reverse'); view(20,80);
title('Mediul cu obstacole');
plot3(colT,linT,10,'g+'); hold on;                          %poziţie ţintă
plot3(colR,linR,10,'r+'); hold on;                          %poziţie robot

% Crearea câmpului potenţial de atracţie
xf=colT; yf=linT; k=3;
for i=1:1:n; for j=1:1:m;
      E=((xf-i)^2+(yf-j)^2)^0.5+0.1*i+0.2*j; Ua=1/2*k*E;
      if   mediul(j,i)==0; Uatr(j,i)=Ua;
      else Uatr(j,i)=255;
end;end;end;
figure, surf(double(Uatr(1:1:m,1:1:n))); hold on; grid on;
axis([0 160 0 120 0 300]); yLabel('j'); xLabel('i');
set(gca,'YDir','reverse'); view(20,50);
title('Potentialul de atractie');

% Crearea câmpului potenţial de repulsie
for i=1:1:n; for j=1:1:m; Urep(j,i)=mediul(j,i); end;end;
   for z=1:1:n; for c=1:1:m; if mediul(c,z)==255;
      for i=z-4:1:z+4; for j=c-4:1:c+4;
         if ((i>0)&(j>0)); if ((z~=i)|(c~=j));
         E=((z-i)^2+(c-j)^2)^0.5; Ur=20*(1/E)^2;
         if E<5; if ((i<=160)&(j<=120));
         Urep(j,i)=Urep(j,i)+Ur; Urep(c,z)=255;
end; end; end; end; end; end; end; end; end;
Urep;
figure, surf(double(Urep(1:1:m,1:1:n))); hold on; grid on;
axis([0 160 0 120 0 600]); yLabel('j'); xLabel('i');
set(gca,'YDir','reverse'); view(20,70);
title('Potentialul de repulsie');

% Crearea câmpului potenţial total
for i=1:1:n; for j=1:1:m; mediu(j,i)=Uatr(j,i)+Urep(j,i); end; end;
figure, surf(double(mediu(1:1:m,1:1:n))); hold on; grid on;
axis([0 160 0 120 0 600]); yLabel('j'); xLabel('i');
set(gca,'YDir','reverse'); view(20,80);
title('Potentialul total si traiectoria robotului');

% Planificarea traiectoriei
pp=digitalio('parallel',1); addline(pp,0:1,'out'); putvalue(pp,[0 0]); pause(.01);
% Iniţializare port
 putvalue(pp,[1 1]);pause(.3); putvalue(pp,[0 0]);          % intrare robot în mediu
 on=0; os=0; oe=0; ov=0; one=1; onv=0; ose=0; osv=0;        % orientare iniţială SE
 i=colR; j=linR; p=1;                                       % pornire robot
 pn=0; ps=0; pe=0; pv=0; pne=0; pnv=0; pse=0; psv=0;        % direcţie optimă
 nf=0; sf=0; ef=0; vf=0; nef=0; nvf=0; sef=0; svf=0;        % nr. paşi pe o direcţie
```

```
dn=0; ds=0; de=0; dv=0; dne=0; dnv=0; dse=0; dsv=0;   % deplasare pe direcţie
 while ((i~=colT)|(j~=linT));
 N=mediu(j-1,i); S=mediu(j+1,i); E=mediu(j,i+1); V=mediu(j,i-1);
 SE=mediu(j+1,i+1); SV=mediu(j+1,i-1); NE=mediu(j-1,i+1); NV=mediu(j-1,i-1);
 X=[N S E V NE NV SE SV]; d=min(X);
 if  d==N; j=j-1; plot3(i,j,N,'r+'); hold on; grid on; pn=pn+1
     sf=ps;ps=0; ef=pe;pe=0; vf=pv;pv=0; nef=pne;pne=0;
     nvf=pnv;pnv=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==S; j=j+1; plot3(i,j,S,'r+'); hold on; grid on; ps=ps+1
     nf=pn;pn=0; ef=pe;pe=0; vf=pv;pv=0; nef=pne;pne=0;
     nvf=pnv;pnv=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==E; i=i+1; plot3(i,j,E,'r+'); hold on; grid on; pe=pe+1
     nf=pn;pn=0; sf=ps;ps=0; vf=pv;pv=0; nef=pne;pne=0;
     nvf=pnv;pnv=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==V; i=i-1; plot3(i,j,V,'r+'); hold on; grid on; pv=pv+1
     nf=pn;pn=0; sf=ps;ps=0; ef=pe;pe=0; nef=pne;pne=0;
     nvf=pnv;pnv=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==NE; i=i+1; j=j-1; plot3(i,j,NE,'r+'); hold on; grid on;  pne=pne+1
     nf=pn;pn=0; sf=ps;ps=0; ef=pe;pe=0; vf=pv;pv=0;
     nvf=pnv;pnv=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==NV; i=i-1; j=j-1; plot3(i,j,NV,'r+'); hold on; grid on;  pnv=pnv+1
     nf=pn;pn=0; sf=ps;ps=0; ef=pe;pe=0; vf=pv;pv=0;
     nef=pne;pne=0; sef=pse;pse=0; svf=psv;psv=0;
 elseif d==SE; i=i+1; j=j+1; plot3(i,j,SE,'r+'); hold on; grid on;  pse=pse+1
     nf=pn;pn=0; sf=ps;ps=0; ef=pe;pe=0; vf=pv;pv=0;
     nef=pne;pne=0; nvf=pnv;pnv=0; svf=psv;psv=0;
 elseif d==SV; i=i-1; j=j+1; plot3(i,j,SV,'r+'); hold on; grid on;  psv=psv+1
     nf=pn;pn=0; sf=ps;ps=0; ef=pe;pe=0; vf=pv;pv=0;
     nef=pne;pne=0; nvf=pnv;pnv=0; sef=pse;pse=0;
 end;

 if (p==115)&(pn>1); nf=pn; end;      if (p==115)&(ps>1); sf=ps; end;
 if (p==115)&(pe>1); ef=pe; end;      if (p==115)&(pv>1); vf=pv; end;
 if (p==115)&(pne>1); nef=pne; end;  if (p==115)&(pnv>1); nvf=pnv; end;
 if (p==115)&(pse>1); sef=pse; end;  if (p==115)&(psv>1); svf=psv; end;

if nf>1;
    ondr=[one oe ose os]; onst=[onv ov osv]; dn=dn+1
    [a,b]=max(ondr); [c,d]=max(onst);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(nf*.015); putvalue(pp,[0 0]);   %deplasare N
    on=1; os=0; oe=0; ov=0; one=0; onv=0; ose=0; osv=0; nf=0;
elseif sf>1;
    osdr=[osv ov onv on]; osst=[ose oe one]; ds=ds+1
    [a,b]=max(osdr); [c,d]=max(osst);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(sf*.015); putvalue(pp,[0 0]);   %deplasare S
    on=0; os=1; oe=0; ov=0; one=0; onv=0; ose=0; osv=0; sf=0;
elseif ef>1;
    oedr=[ose os osv ov]; oest=[one on onv]; de=de+1
    [a,b]=max(oedr); [c,d]=max(oest);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
```

```
    putvalue(pp,[1 1]);pause(ef*.015); putvalue(pp,[0 0]);   %deplasare E
    on=0; os=0; oe=1; ov=0; one=0; onv=0; ose=0; osv=0; ef=0;
elseif vf>1;
    ovdr=[onv on one oe]; ovst=[osv os ose]; dv=dv+1
    [a,b]=max(ovdr); [c,d]=max(ovst);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(vf*.015); putvalue(pp,[0 0]);   %deplasare V
    on=0; os=0; oe=0; ov=1; one=0; onv=0; ose=0; osv=0; vf=0;
elseif nef>1;
    onedr=[oe ose os osv]; onest=[on onv ov]; dne=dne+1
    [a,b]=max(onedr); [c,d]=max(onest);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]); pause(a*b*.16);putvalue(pp,[0 0]);
    putvalue(pp,[1 0]); pause(c*d*.1);putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]); pause(nef*.015); putvalue(pp,[0 0]); %deplasare NE
    on=0; os=0; oe=0; ov=0; one=1; onv=0; ose=0; osv=0; nef=0;
elseif nvf>1;
    onvdr=[on one oe ose]; onvst=[ov osv os]; dnv=dnv+1;
    [a,b]=max(onvdr); [c,d]=max(onvst);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(nvf*.015); putvalue(pp,[0 0]);  %deplasare NV
    on=0; os=0; oe=0; ov=0; one=0; onv=1; ose=0; osv=0; nvf=0;
elseif sef>1;
    osedr=[os osv ov onv]; osest=[oe one on]; dse=dse+1
    [a,b]=max(osedr); [c,d]=max(osest);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(sef*.015); putvalue(pp,[1 1]);  %deplasare SE
    on=0; os=0; oe=0; ov=0; one=0; onv=0; ose=1; osv=0; sef=0;
elseif svf>1;
    osvdr=[ov onv on one]; osvst=[os ose oe]; dsv=dsv+1
    [a,b]=max(osvdr); [c,d]=max(osvst);
    putvalue(pp,[0 0]); pause(.01);
    putvalue(pp,[0 1]);pause(a*b*.16); putvalue(pp,[0 0]);
    putvalue(pp,[1 0]);pause(c*d*.1); putvalue(pp,[0 0]); pause(.1);
    putvalue(pp,[1 1]);pause(svf*.015); putvalue(pp,[0 0]);  deplasare SV
    on=0; os=0; oe=0; ov=0; one=0; onv=0; ose=0; osv=1; svf=0; end;
    p=p+1;
if (p==300); break; end;
end;
p
```