Raport de Cercetare

Grant: Grant CNCSIS-AT tema 2, cod 20, nr. 32940/22.06.2004

Autor: Şef lucr.dr.ing. Cătălin-Daniel Căleanu

Universitatea: Universitatea POLITEHNICA Timişoara

CUPRINS

1. INTRODUCERE

1.1 Tema şi problematica lucrării

Prezentul grant îşi propune studiul problematicii aferente unui robot mobil autonom ce evoluează într-un mediu structurat dar necunoscut apriori, cum ar fi interiorul unui apartament, şi care este destinat asistenței unei persoane umane, de exemplu persoanelor cu handicap locomotor. Se va aborda domeniul navigației roboților mobili autonomi din punctul de vedere al reprezentării mediului înconjurător în formă simbolică, mai concret folosind metode de navigație bazate pe fresce (echiv. lb. engl. *Landscapes*, *Frescoes*) [1]-[3].

Infrastuctura necesară proiectului a fost deja dezvoltată în cadrul Departamentului Sisteme Complexe al Universitatii Evry, Franța sub forma unei arhitecturi ARMAGRA (*Architecture Réseau Multi Agents pour un Groupe de Robots Autonomes*) şi este pusă la dispoziție părții române în scopul finalizării acestui grant [4]. Ceea ce se doreste în continuare este obținerea unei descrieri calitative a mediului înconjurator. Într-un viitor se doreşte ca interacțiunea om-robot să fie facută într-un limbaj natural. Pentru început se doreşte dezvoltarea unui limbaj de nivel mai scăzut, bazat pe o succesiune de markeri topologici. În acest caz, ruta şi/sau destinația pot fi descrise în termeni globali folosind puncte de reper (lb. engl. *Landmarks*). Mediul înconjurator va fi descris prin intermediul unor noțiuni cum ar fi: deschidere, închidere, sfârsitul închiderii, unghiul închiderii, organizate în serii de simboluri denumite fresce, în conformitate cu datele furnizate de senzori.

Una din problemele cele mai importante pentru roboții autonomi, este reprezentarea mediului. De aceea, pe parcursul evoluției acestui domeniu, s-au conceput şi încercat numeroase metode. Odată cu perfecționarea tehnologiei şi a accesibilității sporite a unei largi game de senzori performanți şi a echipamentelor de calcul, aplicațiile roboților au devenit din ce în ce mai numeroase şi se încearcă integrarea acestora în toate domeniile de activitate, precum şi în viața cotidiană a omului. Datorită posibilităților, practic nelimitate, pe care tehnologia le oferă în prezent, se doreşte realizarea unor roboți cu autonomie şi capacități sporite, încercându-se imitarea unor comportamente ale organismelor vii.

Dacă în trecut, mediul robotului se modela în prealabil prin procedee complicate, în prezent pentru a beneficia de versatilitate şi eficiență, un prim deziderat este capacitatea robotului de a se descurca şi opera într-un mediu necunoscut în prealabil, cu posibilitatea de a învăța unele caracteristici ale mediului, cum ar fi unele repere, şi a se adapta anumitor situații. Pe lângă acestea, datorită accesibilității tehnologiei de recunoaştere a vorbirii cu ajutorul calculatorului, această nouă generație de roboți beneficiază în general de facilitatea de a permite comenzi vocale, fapt deosebit de util în aplicații unde se doreşte asistarea unui personal uman sau utilizarea robotului de persoane neinstruite în mod special. Pentru a putea opera într-un mediu necunoscut, robotul trebuie să aibă o percepție asupra mediului cât mai apropiată de cea a ființelor vii, încercându-se, spre exemplu, imitarea modului de orientare al furnicilor [5]. Pentru început se încearcă fundamentarea acestor metode în medii structurate şi închise (ca apartamente sau alte interioare) din cauza simplității naturii acestui tip de mediu. Acestă metodă prezintă avantajul de a nu mai necesita mărimi scalare precise pentru exprimarea pozițiilor unor repere, şi deci se renunță la echipamente cu performanțe deosebite şi costuri mari.

Sistemul de navigație discutat pe larg în această lucrare, se bazează pe descrieri panoramice ale mediului, numite fresce. Acestea sunt imagini simbolice ale mediului generate de senzorii robotului autonom, conținând elementele esențiale pentru robot ale mediului înconjurător. Fiindcă mediul se consideră de tip structurat, elementele componente ale unei fresce pot fi descrise convenabil prin colțuri, deschideri, închideri, sau treceri. Robotul este dotat cu programe ce pot interpreta aceste noțiuni în mod structurat şi poate acționa în consecință, pentru evitarea unor obiecte, “urmărirea” unor elemente de mediu, cum ar fi linia unui un perete, sau pentru reorientare în funcție de repere. Astfel funcționarea robotului nu depinde de un anumit mediu, acesta netrebuind a fi predefinit, şi simplifică implementarea comenzii robotului printr-un limbaj apropiat de cel uman – de exemplu: “urmăreşte coridorul şi intră pe prima uşă din dreapta”. O aplicație a unui astfel de robot este robotul ce ajută şi asistă persoane cu handicap, pentru manipulare de obiecte sau deplasări pe care acestea nu le pot efectua, dar se pot imagina nenumărate aplicații. Robotul discutat în lucrare realizează o descriere a mediului cu ajutorul unui senzor laser 2D, ce măsoară distanțele, digitizând zona vizibilă din mediu în celule, caracteristicile mediului urmând a fi transpuse în noțiuni simbolice, care la rândul lor sunt analizate şi transpuse în descrieri panoramice ale mediului, numite fresce.

În acest context se ridică numeroase probleme, ce se doresc a fi rezolvate prin intermediul cercetarii în cadrul acestui proiect:
- modalități concrete de codare/reprezntare a unei fresce pe baza punctelor de reper; -
- modificarea/transformarea frescelor; În miscarea sa, robotul poate achizitiona prin intermediul senzorilor laser/ultrasonici/video o fresca la câteva secunde. Evident că aceste fresce suferă modificari/transformari eventual sunt contaminate de zgomot deci apare problema selecției acelora care sunt pertinente;
- pentru o descrierea calitativă completă a mediului înconjurator nu toate frescele achiziționate sunt utile şi nu toate trebuie memorate; Apare deci problema selecției frescelor semnificative ce pot furniza o descriere completa a mediului;
- elaborarea unui algoritm care sa permita robotului gasirea parcursului retur pe baza frescelor semnificative.

1.2 Structura lucrării

Având în vedere scopul sus-menționat al prezentului grant, structura acestuia este următoarea:

- **Capitolul 2** prezintă o clasificare şi prezentare generală a metodelor folosite în navigația roboților autonomi, cu precădere asupra celor care folosesc puncte de reper pentru navigație şi interpretări simbolice asupra mediului. Deasemenea se descriu şi tehnologiile disponibile şi cel mai des utilizate pentru implementarea metodelor amintite.

- **Capitolul 3** tratează amplu problema navigației cu ajutorul reprezentării simbolice a mediului, prezentându-se arhitectura de tip ARMAGRA a unui robot ce utilizează o astfel de soluție pentru navigație şi se doreşte a putea fi folosit la asistarea şi deservirea de personal uman, spre exemplu persoane cu handicap. ARMAGRA (Architecture Réseau Multi Agents pour un Groupe de Robots Autonomes) este un prototip dezvoltat de Laboratorul de Sisteme Complexe din cadrul Universității Val d'Essone din Evry, Franța, cu care s-au făcut multiple colaborări pe tema de față.
  O altă problemă abordată în capitolul 3 este explicarea modului în care are loc reprezentarea simbolică a mediului şi implicit crearea unei fresce. Se descriu etapele prin care informațiile din mediu sunt prelucrate şi cum aceste date sun interpretate cu ajutorul unui limbaj simbolic format din puncte de reper abstracte, cu exemple concrete.

- **Capitolul 4** conține partea aplicativă a lucrării de față. Se prezintă metodele utilizate pentru optimizarea numărului de fresce în scopul navigației şi orientării robotului:
  - asemănarea între şiruri;
  - metoda baricentrului;
  - distanța Hamming
  - distanța Levensthein
  - rețele neuronale de tip Hartă de trăsături cu autoorganizare

  Tot în cadrul capitolului 4 se prezintă pe larg, după enumerarea şi descrierea metodelor propuse, rezultatele implemetării acestora pe secvențe reale de fresce preluate de la robotul menționat în lucrare. Sunt analizate performanțele fiecărei metode, pe anumite criterii, cum ar fi: eficiență, simplitate şi viteză, şi se compară cu rezultatele şi performanțele celorlalte metode.
  prezentate în anexa lucrării. Programele utilizate pentru implementarea metodelor discutate şi pentru testarea performanțelor acestora sunt prezentate separat în anexa lucrării.

- **Capitolul 5** – cuprinde concluziile ce se desprind din analizele efectuate în această lucrare şi sugestii pentru eventuale abordări viitoare a problemelor tratate

- **Anexă** – codul sursă al programele dezvoltate în partea experimentală

- **Bibliografie**

2. PRIVIRE DE ANSAMBLU ASUPRA METODELOR FOLOSITE ÎN NAVIGAȚIA ROBOȚILOR MOBILI AUTONOMI

Problema navigației roboților mobili autonomi a fost şi este încă intens studiată. Numeroase aplicații (civile, militare) pot fi identificate, roboții mobili evoluând în spatiu, aer, în medii subacvatice sau terestre.

Pentru navigție au fost folosite numeroase principii [6]: **odometrie** (măsurarea relativă a poziției prin analiza numarului de rotații şi orientarea rotilor) , **navigație inertială** (pe baza masuratorilor relative realizate prin intermediul giroscopului/accelerometru), **ghidare activă** (calculul pozitiei absolute prin masurarea distantei pana la cel putin trei repere), **recunoaşterea punctelor de reper artificiale** (se plaseaza puncte de reper distinctive în pozitii cunoscute ale mediului), **recunoaşterea puncte de reper naturale** (se folosesc puncte de reper existente din mediul inconjurator) etc.

Metodele de navigație a roboților autonomi sunt din cele mai diverse, dată fiind gama largă de utilizare a roboților şi aplicațiile acestora. Mediul înconjurător robotului are o importanță crucială pentru funcționarea şi orientarea acestuia, şi de aceea toate abordările acestei probleme pornesc de la mediu. După acest considerent, se pot evidenția trei principii de navigație:

1. Globală – prin raportare directă prin coordonate absolute la harta mediului înconjurător;
2. Locală – prin determinarea poziției relativ față de obiecte imediat apropiate de robot, staționare sau în mişcare;
3. Individuală – aflarea poziției robotului cu ajutorul unor dispozitive dedicate monitorizării deplasărilor făcute de acesta.

2.1 Navigația globală

În primul caz, întâlnim roboții ce au de parcurs distanțe importante, în spații deschise, fără repere imediate şi la distanțe foarte mari de orice punct de referință. Putem aminti din această categorie roboți folosiți cu precădere de industria militară, cum ar fi avioanele de recunoaştere fără pilot uman de tip UAV, dar şi alte echipamente de pilot automat întâlnite în aeronautică, pe vapoare sau chiar în dotarea automobilelor de ultimă generație.

În majoritatea cazurilor, navigația automată a acestora se face cu ajutorul tehnologiei GPS [7], [8] (Global Positioning System – Sistemul de Poziționare Globală). Acest sistem a fost inițial conceput in cursul anilor 70 de către armata SUA şi a fost folosit exclusiv de aceasta, până când utilitatea acestui sistem a fost necesară şi pentru aplicații civile sau comerciale. Cu toate acestea, varianta disponibilă aplicațiilor civile/comerciale GPS Standard conține în mod intenționat, din motive de securitate, erori ce limitează precizia rezultatelor. Acest sistem utilizează 24 de sateliți aflați pe orbita Pământului la circa 20200 km altitudine, distribuiți astfel încât 4 sateliți GPS să poată fi observați din orice punct al globului, în orice moment. Astfel măsurându-se diferența de timp între semnalele emise de fiecare satelit, şi ținându-se cont de viteza de propagare pentru a afla distanța până la fiecare satelit, se poate calcula poziția subiectului.

Sistemul GPS are trei componente: cei 24 de sateliți aflați pe orbită, sistemele de control aflate la sol, şi utlilizatorul. Sistemul de control constă în stații răspândite pe glob şi asigură buna funcționare a sistemului. Deasemenea, prin stații de recepție fixe se poate obține rezultate mult mai bune a poziționării, prin corecții suplimentare. Această metodă diferențială poartă numele de DGPS. Majoritatea utilizatorilor utilizează DGPS, deoarece sistemul GPS standard, oferă o precizie de până la 100 m, ceea ce pentru navigația unor roboți nu este deloc suficient. Sistemul DGPS este deasemenea potrivit pentru navigație locală, dar acest sistem are dezavantajele de a fi sensibil la mediul înconjurător (undele pot fi deviate sau blocate) şi de aceea nu poate fi folosit cu succes în medii închise. Sistemele de navigație ce folosesc GPS sau DGPS, raportează informația de poziție obținută de la sateliți, la o hartă a mediului detaliată, aflată în prealabil în memoria sistemului.

2.2 Navigația locală

În cazul navigației locale, se folosesc cu precădere metode de detecție vizuală a mediului cu ajutorul a diferiți senzori, cum ar fi optici, laser, sau ultrasonici. [9]-[11]. În cadrul navigării locale se doreşte o modelare şi o interpretare a mediului de către robot, fără ca informațiile despre mediu sa îi fie furnizate în prealabil, ca în cazul anterior. Această interpretare duce la diferite tipuri de reprezentări ale mediului

înconjurător, făcute după modele, funcție de aplicație. Astfel mediul poate fi interpretat mai uşor prin stabilirea unor puncte de reper (landmarks) de către robot prin recunoaşterea unor anumite obiecte sau caracteristici ale mediului. Desemenea aceste repere pot fi stabilite artificial, în puncte cheie, acestea fiind realizate astfel încât să poată fi detectate cât mai uşor. Pe baza interpretării mediului robotul poate realiza hărți bidimensionale sau tridimensionale pentru o orientare mai bună şi prin recunoaşterea anumitor părți din mediu, procesul de navigație poate fi optimizat. Navigația cu ajutorul punctelor de reper va fi tratată pe larg în paragraful următor.

Senzorii utilizați cel mai des, şi totodată cei mai accesibili, sunt cei laser şi camere de achiziție de imagini utilizând senzori CCD sau CMOS (mai ieftine decât CCD). Mediul este perceput sub formă geometrică şi de regulă se foloseşte informația mai multor senzori. Din cauză că senzorii de tip CCD pot obține cea mai importantă cantitate de informație despre mediul înconjurător robotului, acest tip de senzor este tot mai folosit. Prin achiziția de imagini de bună sau foarte bună calitate a mediului, se pot perfecționa şi utiliza cu succes tehnici de recunoaştere a obiectelor şi a mediului înconjurător, interacțiunea robotului cu mediul putându-se realiza printr-o modalitate comună cu cea a omului. De aceea, mai ales prin utilizarea rețelelor neuronale pentru recunoaşterea obiectelor şi interacțiunea cu mediul, problema navigării unui astfel de robot autonom este mult simplificată, permițând roboților de nouă generație să poată fi folosiți cu uşurință şi de personal neinstruit, dar şi o adaptabilitate limitată doar de ingeniozitatea proiectanților.

Navigația bazată pe imagini se poate împărți în următoarele categorii largi de sisteme ce folosesc:

- Telemetre convenționale şi vedere artificială pentru a identifica obiectele [12];
- Rețele neuronale pentru comanda motorului, funcție de informația primită de la senzori. Aceste sisteme încep a structura mediul utilizând concepte ca “intersecție, cameră, colț” şi potrivit acestora generează comenzi de navigație, printr-o unitate de procesare auxiliară [13]. Un dezavantaj al acestei soluții e faptul că rețelele trebuie antrenate în prealabil folosind modele generale şi din această cauză robotul nu se poate descurca cu uşurință în medii structurate complexe.
- Sisteme ce descriu mediul prin trăsături de bază (colțuri, plane) şi totodată, corelează aceste informații la un nivel superior cu noțiuni mai concrete (uşă, masă, cameră). Un dezavantaj al acesteri soluții ar fi faptul că datorită complexității sistemului de vedere ce oferă o cantitate foarte mare de informații, este necesară o putere de calcul sporită, capabilă să proceseze în timp real toate datele. Deasemenea, datele sunt interpretate şi comparate cu modele cunoscute şi fixate în prealabil.
- Sisteme ce interpretează obiectele şi structura mediului din punct de vedere geometric, construind modele, în prealabil deciziei asupra unei căi de urmat [14].

2.3 Navigația individuală sau autonomă

O metodă des folosită, de regula în combinație cu metodele prezentate mai sus, este calcularea poziției robotului relativ la mediu prin masurări directe asupra vitezei şi traiectoriei parcurse de către robot. Această metodă, numită şi odometrie, oferă o corecție mai bună a erorilor de deplasare şi totodată este relativ simplu de implementat, fără costuri importante, soluția regăsindu-se la roboții ieftini sau cu aplicații simple. Probabil cea mai simplă soluție pentru a implementa această metodă este cu ajutorul unui dispozitiv numit odometru. Acesta este cuplat direct la axul roții robotului şi furnizează informații despre viteză sau schimbări de direcție.

Deoarece majoritatea roboților mobili utilizează roți sau şenile, această soluție a devenit practic omniprezentă la aproape toți roboții mobili. Dintre soluțiile de implementare se pot aminti: odometre cu perii, magnetice, inductive, capacitive, optice. Odometrele optice sunt larg răspândite datorită simplității şi a bunei toleranțe la interferențe şi zgomot, dar şi datorită accesibilității lor. Din păcate apar alte surse de erori, cum ar fi datorită: alinierii defectuase a roților, alunecarea roților, modificarea diametrului roții (prin uzura cauciucului sau a umflării/dezumflării acestuia), denivelări ale solului sau interacțiunea nedorită cu diferite obiecte. Aceste erori apar ca urmare a principiului de bază al odometrului – integrarea infomației incrementale de mişcare în timp, ce duce la erori cumulative în timp, dar oferă precizie acceptabilă pe distațe scurte. Deşi acest dezavantaj este important, el poate fi diminuat prin numeroase metode, iar odometrul este unanim considerat ca o parte importantă a sistemului de navigație al oricărui robot.

Odometrul este foarte folosit din mai multe considerente:

- informația odometrului poate fi corectată în anumite puncte cu poziții absolute cunoscute, precum repere artificiale sau puncte de plecare, alimentare, etc;
- odometria se poate folosi între puncte de poziție cunoscute, iar informația corectată prin raportare la diferite repere. Datorită informației odometrului, numărul de repere poate fi redus;

- în unele situații, odometria este singura metodă de referință disponibilă pentru navigație, în cazuri în care, de exemplu, lipsesc repere sau alte puncte de referință din mediu şi robotul nu se poate raporta la nimic altceva.

Alte metode utilizează senzori Doppler sau unde active de tip laser, sonice, radio sau microunde pentru orientare sau măsurarea vitezei. Senzorii Doppler funcționează pe baza efectului cu acelaşi nume, ce priveşte modificarea frecvenței undei radiate funcție de viteza şi direcția emițătorului. Undele active se folosesc cu precădere în spații închise, folosind ca repere pentru robot. Pentru roboții mobili, cele mai utilizate sunt laserele şi ultrasunetele, dar dezavantajul lor constă în faptul că aplicabilitatea robotului rămâne restrânsă la incinta ce găzduieşte aceste repere. Desemenea, mediul trebuie astfel configurat, încât să nu permită blocări ale semnalelor de ghidare sau interferențe.

Pentru aflarea poziției, cea mai utilizată metodă este cea a triangulației, ce presupune măsurarea distanței dintre trei balize şi robotul în cauză. Această metodă prezintă avantajul de a putea măsura şi diferențe de timp, de unde poate rezulta o mai bună precizie în măsurarea distanțelor. Reperele active se folosesc cu precădere în spațiile închise şi în general destinate numai roboților, de exemplu în spații industriale, pentru ca undele emise de aceste repere să nu fie perturbate.

Sistemele de navigație în medii inchise se pot împărți în:
- Sisteme ce se bazează pe hărți ce utilizează modele geometrice şi/sau topologice ale mediului
- Sisteme care construiesc reprezentări geometrice şi/sau topologice ale mediului pe baza datelor obținute de senzori
- Sisteme a căror navigație nu se bazează pe hărți propriu-zise, accentul fiind pus pe recunoaştere complexă a obiectelor şi acțiuni asociate cu acestea [15]

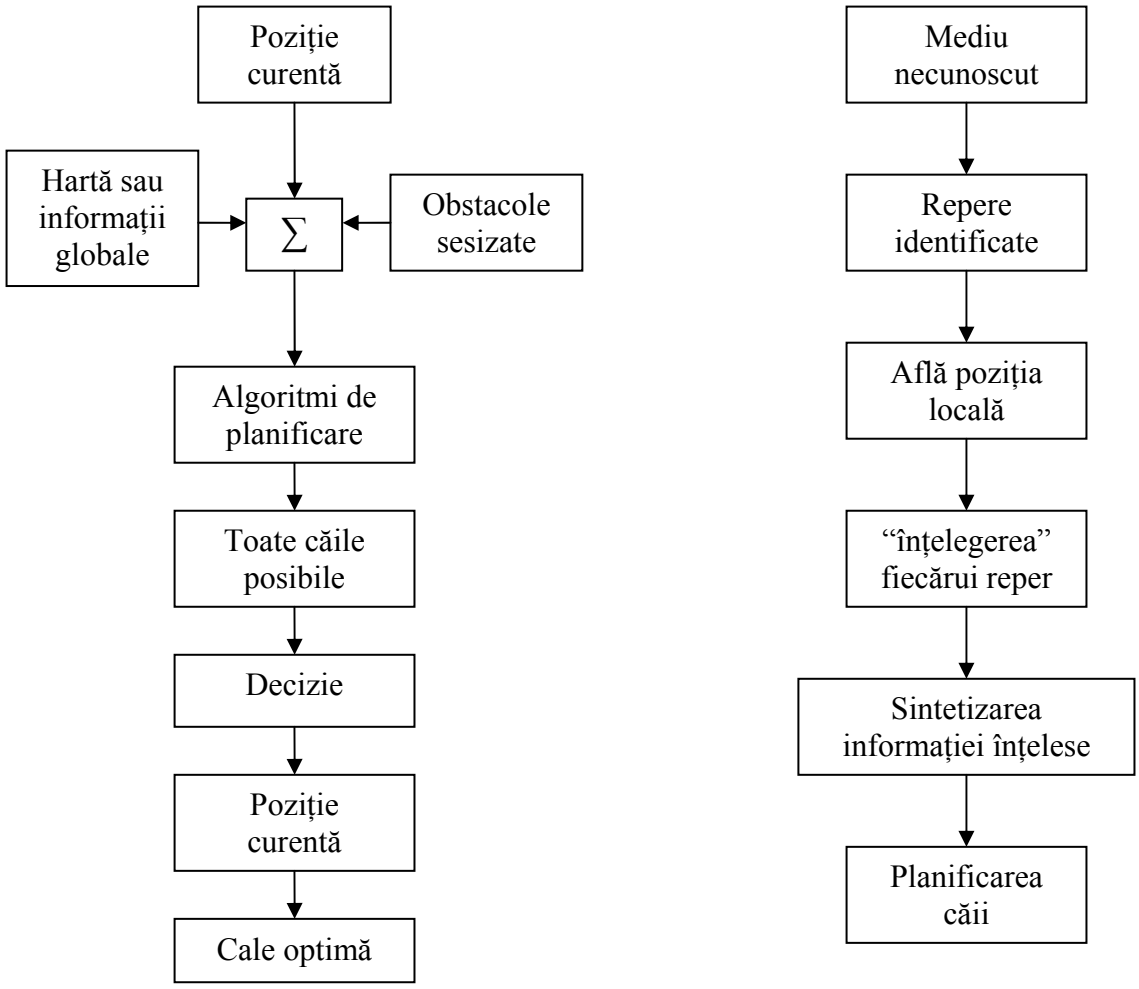
Dintre acestea, în lucrarea de față interesează cu precădere ultimele categorii, deoarece se tratează detaliat tipul de aplicație în care se apelează la navigația pe bază de repere, iar reprezentarea mediului se face simbolic.

Până nu demult, majoritatea metodelor de ghidare ale roboților presupuneau o cunoaştere prealabilă a mediului, într-o formă sau alta. Această abordare duce la o anumită simplificare a problemei şi a ghidării robotului, dar aduce o inflexibilitate aplicației față de schimbări de mediu, fie şi minore, prin cvasi-impunerea unor trasee predefinite sau calculate pe baza unei hărți date. Utilizarea mai multor roboți simultan într-o astfel de situație complică lucrurile foarte mult prin faptul că fiecare robot devine un potențial obstacol mobil pentru ceilalți, ajungându-se la o reducere considerabilă a performanțelor de navigație. Navigația cu ajutorul reperelor rezolvă problema, pentru că fiecare robot trebuie dotat cu capacități de auto-învățare şi auto-localizare, asigurându-se o independență a robotului față de mediu şi schimbările ce pot avea loc în acesta. Capacitatea robotului de a recunoaşte anumite repere din mediu este cheie pentru problema auto-localizării. În timp, robotul va putea construi o hartă a întregului mediu, funcție de “experiența” sa prin acel mediu. Cu cât auto-localizarea se va face mai aproape de realitate, cu atât precizia în navigație va creşte, crescând şi performanța sistemului. În [16] se face următoarea comparație între aceste două abordări în privința planificării traseului, ilustrată de fig. 2.1.

2.4. Metoda de navigație pe bază de puncte de reper (*landmarks*)

Spre deosebire de roboții ghidați, cei autonomi trebuie să fie capabili de a interacționa cât mai bine cu mediul, fiind necesară o interpretare optimă a acestuia. Dacă se doreşte şi o umanizarea dialogului cu robotul pentru o utilizare facilă pentru orice utilizator uman, atunci este necesar ca robotul să realizeze o descriere cât mai structurată a mediului, pornind de la un nivel primar şi abstract al înțelegerii mediului, până la recunoaşterea anumitor caracteristici ale mediului sau obiecte, prin compunerea noțiunilor primare. Acest lucru se poate realiza prin dotarea robotului cu capacitatea de a recunoaşte anumite repere din mediu. Reperele (landmarks) cu care poate opera robotul se definesc ca trăsături distincte pe care robotul le poate recunoaşte cu ajutorul senzorilor săi. Acestea pot fi forme geometrice, de tip dreptunghi, linii, rotunduri, şi după caz, pot conține informații suplimentare, precum coduri de bare sau alte marcaje. În general, se consideră că reperele sunt fixe iar poziția lor este cunoscută, față de care robotul îşi poate afla propria poziție. În unele situații, reperele se aleg astfel încât să poată fi uşor identificate de sezorii robotului – culori contrastante cu mediul sau alte trăsături proeminente. Există însă şi cazuri în care se doreşte ca robotul să învețe să se descurce în aproape orice mediu [17], fără plasarea unor repere artificiale în mod intenționat, pentru ca orientarea robotului să nu depindă de anumite repere predefinite. În prealabil, caracteristicile reperelor trebuie cunoscute, măcar la nivelul cel mai general, şi memorate în computerul robotului. În cadrul navigației, principala sarcină este recunoaşterea cât mai fidelă a reperelor şi aflarea poziției față de acestea.

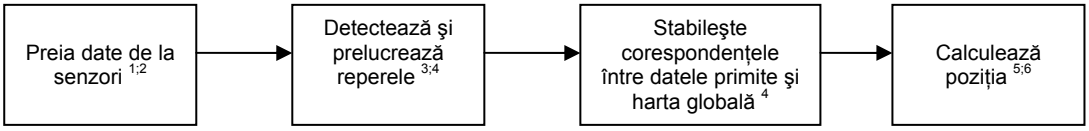
De regulă se optează pentru metode combinate de calculare a poziției, folosindu-se adesea metoda odometriei pentru îmbunătățierea preciziei cu care se determină poziția. Astfel se simplifică procesul de recunoaştere a reperelor, deoarece se pot presupune cunoscute, în mod aproximativ, orientarea şi poziția robotului, diminuându-se aria de căutare a robotului.
Procedeul general de orientare cu ajutorul reperelor este descris în figura 2.2 [6].



Planificarea căii în mod "tradițional"

Procedura pentru un robot într-un mediu necunoscut

Fig. 2.1. În [16] se propune folosirea de repere sub formă de indicatoare care să conțină şi informații asupra sarcinii ce trebuie îndeplinită cât şi a direcției şi distanței de parcurs până în punctul respectiv. Acest lucru presupune totuşi un sistem de interpretare a datelor mai performant.



Note:
1. repere active
2. repere distincte
3. căutarea poate fi simplificată, cu ajutorul unei alte metode ce poate aproxima poziția reală (ex. odometrie)
4. procesul este influențat de complexitatea mediului şi al reperelor.
5. triangulație – eroarea este funcție de pozițiile relative între robot şi repere
6. formă geometrică – eroarea este funcție de distanța şi unghiul dintre robot şi reper

Fig 2.2. Procedeul general pentru poziționarea cu ajutorul reperelor

Reperele se pot împărți ăn două categorii distincte: *naturale* şi *artificiale*. Cele din urmă sunt obiecte special construite şi amplasate pentru a uşura sau asigura navigația corectă a robotului. Acestea se pot compara cu semnele de circulație sau alte inscripții stradale utilzate în traficul rutier, deşi în cazul navigației roboților pot avea trăsături foarte diferite. Reperele naturale sunt cele mai întâlnite, prin faptul că apar în mod implicit în mediul de acțiune al robotului. Acestea nu prezintă caracteristici anume, definite de om, şi care să ajute la navigația robotului iar forma şi natura lor sunt dintre cele mai diferite.

2.4.1 Reperele artificiale

Acestea sunt repere construite de către om şi amplasate în mod convenabil în spațiul de acțiune al robotului, pentru a uşura considerabil orientarea şi poziționarea robotului în mediu. Amplasarea unor astfel de repere este întâlnită în spațiile destinate în mod special roboților, cum ar fi zone industriale sau laboratoare, sau în medii unde operează simultan mai mulți roboți. Reperele artificiale se folosesc adesea împreună cu roboți ce folosesc vedere computerizată şi de aceea reperele conțin adesea forme geometrice şi marcaje vizuale realizate cu culori foarte contrastante. Avantajele reperelor artificiale sunt importante în procesul navigației, deoarece poziția lor este precis cunoscută în prealabil, astfel contribuindu-se la optimizarea navigației. Un alt avantaj este obținut prin amplasarea de marcaje de tip cod de bare (foarte utile pentru roboții ce folosesc senzori laser) care conțin diferite informații legate de mediu, precum indicații sau sarcini de efectuat. Un dezavantaj al utilizării reperelor artificiale este faptul că acestea neexistând într-un mediu obişnuit, roboții ce le utilizează îşi limitează funcționalitatea la zonele în care există aceste repere şi astfel sunt mai puțin adaptabili decât cei care se bazează exclusiv pe recunoaşterea unor repere naturale dintr-un mediu convențional.

2.4.2 Reperele naturale

Utilizarea reperelelor naturale în navigația roboților este o problemă mai complicată decât în cazul reperelor artificiale. Un prim motiv este complexitatea şi diversitatea reperelor naturale, care diferă de la mediu la mediu şi astfel sunt mult mai greu de clasificat şi abstractizat pentru ca un robot să le poată recunoaşte şi deosebi. Din aceste motive, este necesară utilizarea unor senzori ce pot oferi o cantitate cât mai mare de informații, şi de aceea se optează pentru vederea artificială, cu ajutorul senzorilor CCD sau CMOS împreună cu telemetre laser. Cele mai comune repere naturale în vederea computerizată sunt colțurile verticale, precum uşile, joncținile pereților, coridoarele sau forma luminilor din iluminatul artificial amplasat pe tavan. Fiindcă aceste repere sunt foarte frecvente în mediile structurate, sunt şi cele mai folosite pentru navigație şi de aceea, pentru a asigura o detecție optimă a acestora, sistemele de detecție ale roboților prezintă următoarele componente: un senzor sofisticat pentru detecția reperelor (de regulă senzori de achiziție de imagini performanți), programe de recunoaştere a reperelor programe, prin comparație cu modele deja cunoscute, şi metode de calculare a poziției relativ față de repere şi estimare a erorilor de poziționare.

Deoarece tehnologia a evoluat tot mai mult, interesul pentru roboți ce navighează folosindu-se de repere naturale, precum organismele vii, a crescut, şi de aceea apar tot mai multe prototipuri de roboți ce folosesc această metodă de navigație. Interesul pentru această problemă se justifică şi prin faptul că, deoarece tehnologia permite, se doreşte simplificarea până la umanizare a dialogului cu noile generații de roboți şi aducerea acestora la un stadiu cât mai apropiat de comportamentul organismelor vii, în special în ceea ce priveşte navigația şi adaptabilitatea. Aceste calități ar extinde cu mult aplicabilitățile deja foarte numeroase ale roboților şi ar permite apariția unor anumite categorii de roboți autonomi meniți să ajute direct personal uman în diferite situații. Totuşi, pentru a aduce la un numitor comun maşina cu omul, se impune, dinspre robot, o abordare nouă în ceea ce priveşte reprezentarea mediului, aceasta fiind una din problemele cele mai importante pentru funcționarea corectă a unui robot autonom. Astfel, preferându-se metoda orientării cu ajutorul reperelor naturale, se încearcă sistematizarea modului în care robotul percepe obiectele şi trasăturile mediului structurat, prin elaborarea unui set de reguli de bază şi definiții prin care robotul poate defini şi construi reprezentări ale mediului în care operează. Deşi există numeroase abordări, atât utilizând repere artificiale cât şi naturale, în cele ce urmează şi pe parcursul lucrării, se va face referire la reprezentarea mediului prin aşa numitele fresce sau *frescoes*, *landscapes*, cum sunt cunoscute în literatura de specialitate internațională.

2.4.3 Descrierea mediului cu ajutorul frescelor

Pentru a face posibilă o reprezentare calitativă coerentă a mediului este necesară şi o sistematizare a modului în care robotul preia informațiile numeroase ce provin de la senzori. De aceea, trăsăturile mediului sunt analizate şi interperetate pe anumite nivele de prelucrare a informației. După ce imaginile şi datele de la telemetre sunt prelucrate, se extrag şi se organizează informațiile referitoare la reperele existente în mediu. Metoda frescelor se aplică în cazul reperelor naturale, iar o frescă reprezintă o secvență de informații calitative asupra zonei de observație a robotului la un moment dat. Memorarea unor astfel de secvențe cheie oferă posibilitatea robotului de a forma autonom o hartă a mediului şi deasemenea informațiile necesare pentru efectuarea unui traseu de întoarcere.

Pentru ca robotul să se poată orienta în mod complet autonom într-un mediu, acesta trebuie să aibe o privire de ansamblu asupra mediului, sau altfel spus, să aibe în memorie date esențiale asupra mediului. Datorită naturii procesului de preluare a informațiilor din mediu, robotul achiziționează o cantitate mare de fresce, dintre care majoritatea au un caracter redundant pentru navigația robotului. Reducerea numărului acestora prin eliminarea frescelor redundante şi evidențierea celor cheie procesului de orientare şi navigație al robotului, este una din problemele importante ce fac subiectul lucrării de față.

De regulă, mediul se consideră de tip structurat, şi de aceea, elementele componente ale unei fresce pot fi descrise convenabil prin colțuri, deschideri, închideri, sau treceri. Robotul este dotat cu programe ce pot interpreta aceste noțiuni în mod structurat şi poate acționa în consecință, pentru evitarea unor obiecte, “urmărirea” unor elemente de mediu, cum ar fi linia unui un perete, sau pentru reorientare în funcție de repere [4]. Avantajele unui astfel de sistem de interpretare al mediului constau în faptul că funcționarea robotului nu depinde de un anumit mediu, acesta netrebuind a fi predefinit, şi uşurința comandării robotului printr-un limbaj apropiat de cel uman – de exemplu: “urmăreşte coridorul şi intră pe prima uşă din dreapta”. Aceasta contribuie considerabil la simplificarea şi umanizarea cât mai mare a modului de comandă al robotului, ca acesta să poată fi folosit prin comenzi verbale şi de persoane neinstruite în mod special. O aplicație a unui astfel de principiu este un robot ce ajută şi asistă persoane cu handicap, pentru manipulare de obiecte sau deplasări pe care acestea nu le pot efectua [2].

3. NAVIGAȚIA CU AJUTORUL REPREZENTĂRII SIMBOLICE A MEDIULUI

3.1 Prezentarea arhitecturii ARMAGRA

Problema navigației cu ajutorul reprezentării simbolice a mediului, prin intermediul frescelor, are o aplicabilitate imediată în funcționarea corectă a unui prototip de robot conceput cu scopul de a asista şi ajuta omul în diferite situații, într-un mediu similar unui apartament. Acest robot (fig. 3.1) va permite comanda vocală şi va fi capabil de a recunoaşte obiecte pentru a le transporta dintr-un loc în altul sau pentru a le manipula (de exemplu închiderea unei uşi). Un astfel de robot este construit în cadrul proiectului ARMAGRA (Architecture Réseau Multi Agents pour un Groupe de Robots Autonomes), dezvoltat de Laboratorul de Sisteme Complexe din cadrul Universității Val d'Essone - Evry, Franța. Arhitectura numită ARMAGRA (fig. 3.2) este rezultatul îmbunătățirii continue a altor arhitecturi similare, precum AMARA şi AMAGRA [1] şi se doreşte a fi fundamentul unui model de robot viabil pentru asistența persoanelor cu handicap.

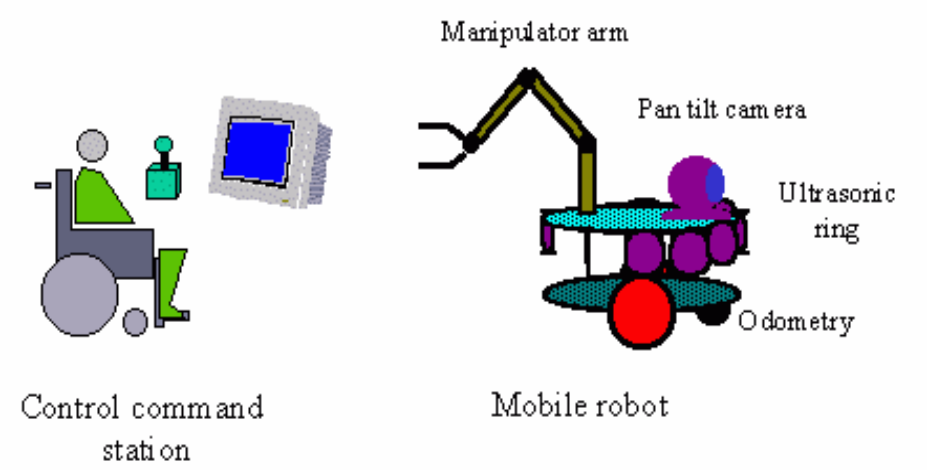


Fig.3.1. Robot mobil destinat asistenței persoanelor cu handicap, dezvltat de către Laboratorul de Sisteme Complexe din cadrul Universității Val d'Essone – Evry.

Printre problemele rezolvate în cadrul acestei arhitecturi se numără:
- navigația în cadrul unui apartament,
- recunoaşterea unei situații,
- executarea autonomă a unei sarcini,
- comunicarea şi interfața cu persoana cu handicap,

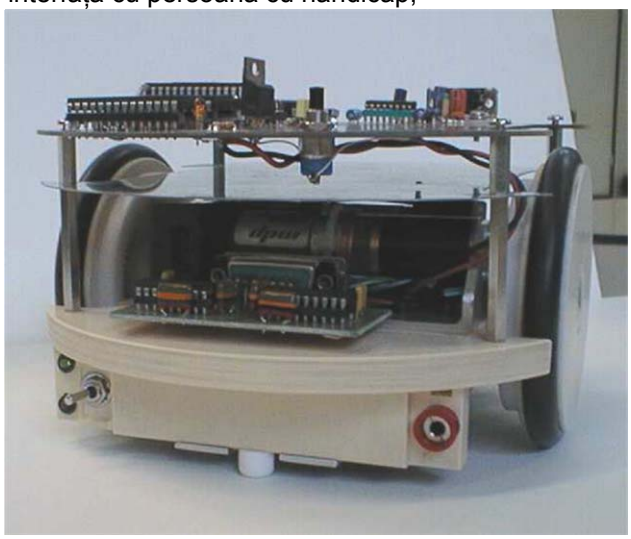


Fig. 3.2 – Prototipul AMAGRA.

Această arhitectură a suferit modificări, plecându-se de la variantele precedente, către adaptarea sistemului pentru lucrul în echipe de roboți mobili, pe o structură utilizând *multi-agenți*. Dat fiind că platforma este în continuă schimbare şi se efectuează mereu îmbunătățiri, se vor prezenta doar trăsăturile generale.

Platforma prezentată în figura 3.2 are dimensiuni reduse față de variantele precedente – 25 cm în diametru şi aproximativ 2,75 kg. Gabaritul redus prezintă un avantaj şi din punctul de vedere al consumului energetic, care în această situație este micşorat, îmbunătățind autonomia robotului. Prototipul ARMAGRA se bazează pe microcontrolere de tip 80C552 şi LM629 pentru controlul motoarelor.

Schimbul de informații se face într-o manieră cât se poate de inovatoare, prin două tipuri de rețele, gândite în scopul structurării fluxului de date:

- o rețea de tip FAN (Flat Area Network) dedicată activității roboților. O astfel de rețea aparține de un utilizator ce gestionează un număr de roboți ce îndeplinesc o sarcină;
- o rețea de tip LAN (Local Area Network) ce permite accesul la resursele partajate, printre care sistemele de decizie, module de analiză şi procesare a sarcinilor, etc. Se are în vedere şi implemetarea unor facilități care să permită controlul de la distanță, pentru teleobservare, telemedicină şi alte aplicații. Această facilitate este utilă în cazul în care există mai multe grupuri de roboți cu sarcini diferite, posibil şi în zone diferite, permițând schimbul de informații între aceştia, dialogul cu operatorul uman, etc.

Structura ARMAGRA combină metode cantitative şi calitative pentru navigație, metoda calitativă având o pondere mai mare. Ca şi în cazul ARMAGRA, în practică se folosesc frecvent metode mixte, abordarea cantitativă fiind folosită pentru distanțe foarte scurte şi evitarea coliziunilor, de regulă prin odometrie. Totuşi, reprezentarea calitativă a mediului se doreşte a fi predominantă, fiind folosită pentru orientarea robotului şi deplasările pe distanțe mari. Acestă opțiune se justifică deoarece este necesară o viziune globală asupra mediului, apoi se facilitează dialogul om-maşină, prin folosirea unui limbaj comun, şi totodată această abordare se apropie de mecanismele de navigație întâlnite la organismele vii, inclusiv om, care se ghidează după puncte de reper.

În cazul ARMAGRA, problema navigației se doreşte a fi rezolvată cu ajutorul *frescelor*. Acestea permit descrieri calitative ale mediului, printr-o suită de puncte de reper naturale, cu scopul final de a permite robotului să determine şi recunoască traiectoria retur.

3.2 Descrierea modului prin care se creează o frescă

Robotul este dotat cu un telemetru laser panoramic amplasat în centrul geometric al robotului, având o rază efectivă de 3 metri şi raza maximă de 10 metri. Deşi senzorul captează mediul pe o zonă circulară, s-a considerat o zonă pătrată de 6 × 6 metri pentru reprezentarea mediului, din dorința de a se simplifica etapa de procesare a informațiilor despre mediu. Robotul este considerat mereu în centrul mediului. Telemetrul utilizat este capabil de a realiza 1024 măsurători pe rotație (5 rpm), dar dintre acestea numai 256 vor fi utilizate în procesul de construcție al frescelor.

Pentru buna funcționare a întregului sistem se consideră că:

- marea parte a mediului se compune din obiecte şi suprafețe cu un coeficient de absorbție scăzut şi nu vor creea interferețe;
- poziția de referință a laserului coincide cu axa principală a robotului;
- măsurătorile se fac în condițiile în care robotul se deplasează;
- precizia este mai mare de 20 cm pentru fiecare măsurare.

Digitizarea mediului se face prin împărțirea mediului după o grilă de 32 × 32 celule, acoperind zona percepută de telemetru. Fiecare celulă reprezintă o zonă de 0.1875 m × 0.1875 m. O astfel de celulă este considerată “activă” în momentul în care unda laser întâlneşte un obiect aflat în celula în cauză şi este “inactivă” în caz contrar. În primă fază, mediul este sintetizat din punct de vedere vizual ca fiind compus din obstacole şi linii de ocluzie. Acestea din urmă pot reprezenta porțiuni parțial vizibile din mediu, o falsă interpretare a unui obiect, sau o deschidere la limita de 3 metri a senzorului. Navigația corectă prin zona asfel digitizată presupune traversarea unei astfel de linii. Deplasarea robotului va îmbunătăți constant percepția asupra naturii acestor linii, iar funcție de anumite caracteristici şi pe baza datelor anterioare, se vor putea anticipa următoarele trăsături ale mediului.

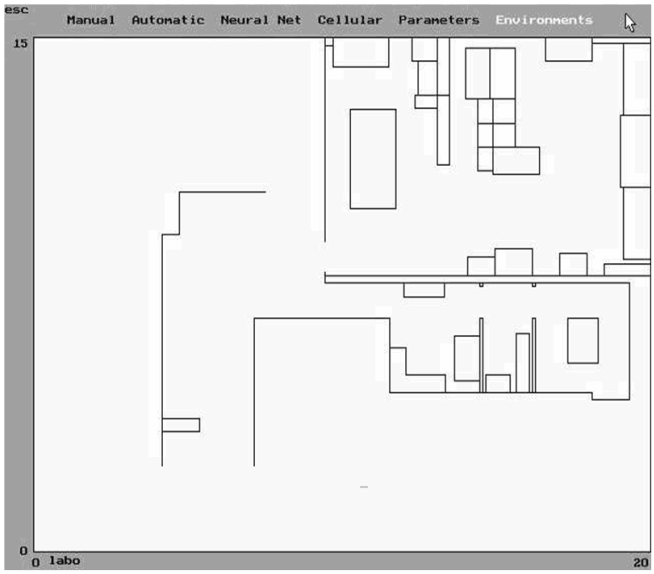


Fig. 3.3. Structura geometrică a mediului.

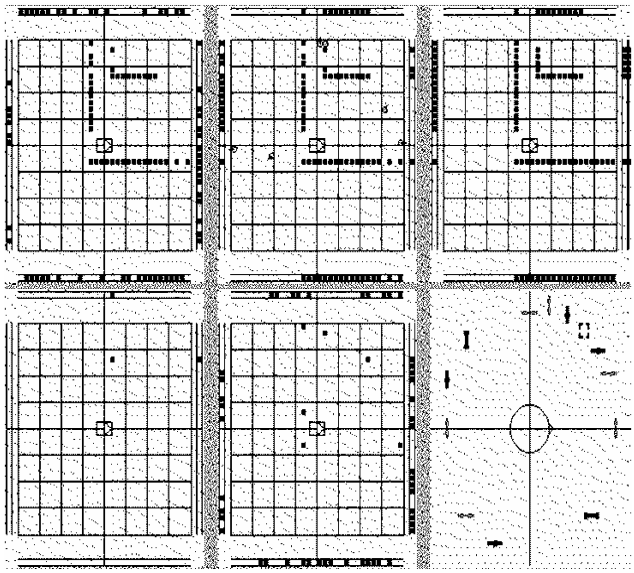


Fig. 3.4 Etape de prelucrare a informației şi obținerea unei fresce (imaginea dreapta-jos).

Mediul în care evoluează robotului, vedere de ansamblu (stânga); etapele de contrucție şi fresca corespunzătoare (dreapta).

Elementele cheie ce urmează a fi obținute sunt deschiderile, închiderile, capătul închiderilor, unghiul închiderilor, respectiv cum au fost numite inițial – Opening, Closure, End_of_Closure, Angle_of_Closure. Acestea se obțin din informațiile metrice şi vor forma repere (lb. engl. *landmarks*) ce vor fi utilizate pentru formarea descrierii calitative a mediului. Robotului îi sunt ataşate două axe perpendiculare, ce ajută în procesul de construcție a mediului digitizat – axa principală, orientată pe lungimea robotului (față – spate), şi axa transversală.

Procesul de reconstrucție al mediului în formă digitizată se împarte pe următoarele etape:

- construcția spațiului în forma împărțită pe celule (32 × 32);
- clasificarea informației senzoriale;
- obținerea segmentelor corespunzătoare celor doua axe principale şi diagonalelor;
- rafinarea datelor;
- reorientarea segmentelor celulare pentru o procesare mai uşoară.

3.2.1 Construirea unei fresce

Acest proces se desfăşoară după următoarele etape:

- obținerea elementelor Opening, Closure, End_of_Closure, Angle_of_Closure;
- construirea frescei;
- validarea frescei obținute.

Prin analizarea spațiului digitizat şi împărțit pe celule (fig. 3.4) se obțin repere de tipul opening, closure, etc, amintite mai sus şi se reconstruieşte imaginea mediului cu ajutorul acestora. Această nouă descriere de tip simbolic elimină noțiunile scalare precum distanțele, descriind mediul doar din punct de vedere calitativ. Fresca obținută conține un limbaj format exclusiv din o serie de repere ce urmează a fi analizate din punct de vedere logic.

Validarea frescei presupune prelucrarea frescei obținute anterior, cu ajutorul unor legi de vecinătate bine definite, între repere. Spre exemplu, în vecinătatea unui reper de tip Angle_of_Closure se pot afla numai repere de tip Angle_of_Closure sau End_of_Closure, prezentate în tabelul 3.1. Aceste verificări se fac pentru fiecare reper în parte iar pentru ca o frescă să fie validată, întregul set de reguli este aplicat. De regulă, numărul de fresce nevalidate este mare în cazul în care este indus zgomot de către o celulă sau mişcarea robotului. Totuşi, pierderea unei sau a câtorva fresce nu este importantă pentru navigația robotului, dat fiind faptul că prin deplasarea sa, robotul achiziționează mereu fresce noi şi oricum acestea sunt sortate şi doar cele relevante sunt memorate şi folosite efectiv pentru navigație.

Inițial, frescele puteau să aibă o lungime variabilă, în funcție de numărul de repere văzute la un moment dat de robot.

În cadrul analizelor efectuate în cadrul acestui grant vizavi de problema reprezentării unei fresce, s-a ajuns la concluzia ca se impune modificarea acesteia în sensul în care, indiferent de numărul de repere perceput de robot să rezulte o frescă de lungime fixă.

Această nouă reprezentare va uşura modalitatea de prelucrare, stocare şi comparare a frescelor aducând şi un plus de informație în privința localizării exacte a unui reper vizavi de poziția robotului.

Concret, fiecare frescă este reprezentată printr-un şir de 64 de caractere, precum acest exemplu, preluat din memoria robotului:

E00C004000000F6EC700006AC4500000E000020000900000000000000000040

În tabelul de mai jos este reprezentată o secvență de câteva astfel de fresce, preluate în ordinea furnizată de calculatorul robotului. Deoarece spațiul înconjurător robotului este descris prin 4 cadrane, iar fiecărui cadran îi corespunde un sfert din secvența de date a unei fresce, frescele s-au reprezentat împărțite pe cadrane, pentru lizibilitate.

Cadranul I	Cadranul II	Cadranul III	Cadranul IV
E000C005F7400090	02A6000000600000	E002090000000000	0000000000000040
E000CC004F740000	00004C0B00060000	E029000000000000	0000000000000040
6E00CC0674F70400	0000004CA6006000	E290000000000000	00000000000000F0

Inițial, o frescă conținea doar reperele, reprezentate prin caracterele de la 1 la F, iar astfel şirurile frescelor aveau lungimi variabile, lucru ce îngreunează prelucrarea şi compararea şirurilor. De aceea s-a optat pentru aducerea tuturor şirurilor la aceeaşi lungime, indiferent de numărul de repere, prin introducerea de zerouri în celulele spațiului aferent robotului ce nu conțin puncte de reper.

Symbol	Landmark	Position	Off-sight	Certainty
	Angle_of_closure			True
	End_of_closure	Lengthwise		True
	End_of_closure	Lengthwise	Off-sight	False
	End_of_closure	Crosswise		True
	End_of_closure	Crosswise	Off-sight	False
	End_of_closure	Diagonal1		False
	End_of_closure	Diagonal1	Off-sight	False
	End_of_closure	Diagonal2		False
	End_of_closure	Diagonal2	Off-sight	False
	45° angle	Lengthwise		False
	45° angle	Crosswise		False
	Opening	Lengthwise		True
	Opening	Crosswise		True
	Breakthrough	Lengthwise		True
	Breakthrough	Crosswise		True

Tabelul 3.1: Limbajul reperelor (*landmarks*) utilizat pentru construcția frescelor.

Pentru sortarea frescelor s-au încercat mai multe metode, dintre care o parte sunt prezentate în capitolul următor. În studiul acestei probleme, s-au interpretat secvențele de fresce drept şiruri de caractere, apelându-se din metode folosite în diferite domenii la prelucrarea şirurilor de caractere (lb. engl. *strings*). Odată sortate, frescele considerate relevante sunt stocate într-o memorie de tip LIFO. Scopul final al acestor prelucrări şi operații este ca robotul să poată folosi informațiile despre mediu memorate pentru efectuarea drumului retur, cât şi pentru o mai bună reprezentare a întregului mediu.

Capacitatea de a efectua în mod autonom drumul de întoarcere, este foarte importantă pentru orice robot mobil, în special pentru cei destinați asistării persoanelor cu handicap. Fiindcă problema este complexă, există mai multe abordări în ceea ce priveşte interperetarea informațiilor despre mediu memorate în prealabil, etapă descrisă anterior.

Deoarece la întoarcere, frescele memorate sunt diferite față de cele curente preluat în acelaşi loc, o primă soluție este rotirea frescelor memorate cu 180°. Din păcate, acest lucru nu este în totdeauna util, deoarece este greu de decis când această rotire trebuie făcută. De aceea, o soluție mai viabilă este deplasarea frescei curente la stânga sau la dreapta pentru a se determina dacă există o potrivire cu una din cele aflate în memorie. O altă metodă consideră gruparea reperelor ce formează o frescă în structuri ce definesc obiecte complexe, precum cele de mobilier, oferind posibilitatea unei descrieri mai calitative a mediului, foarte apropiată de cea umană. Din păcate, această metodă este dezavantajoasă prin cantitatea mare de calcule şi transformări pe unitate de timp, necesare pentru a realiza aceste descrieri. Spre deosebire de aceasta, o altă metodă similară ce se bazează pe urmărirea evoluției unor grupuri restrânse de repere este mai simplă şi mult mai avantajoasă din punct de vedere al resurselor de calcul utilizate. Această metodă presupune ca robotul să fie capabil de a estima trăsăturile viitoare ale mediului, pe măsură ce avansează. Anticiparea trăsăturilor mediului pe parcursul deplasării este o facilitate foarte utilă care, odată implementată, oferă o simplificare a navigației robotului şi o foarte bună capacitate de orientare printr-un mediu cvasi-cunoscut, sau chiar necunoscut.

4. METODE PENTRU ANALIZA REPREZENTARILOR SIMBOLICE ALE MEDIULUI

În cele ce urmează se vor investiga diverse posibilități de analiză (în special metode de selecție ale frescelor semnificative) pentru reprezentările simbolice ale mediului, denumite în acest context, fresce.

Unele din metode sunt inspirate din cele folosite în corectarea automată a unui text (OCR, speller) sau din algoritmi de cautare în baze de date după cuvinte cheie [18]. Altele provin din biologie unde compararea stringurilor reprezintă o operațiune extrem de necesară în special pentru biologia moleculară (ADN, secvențe de amino-acizi) [19], [20]. În fine, sunt propuse şi metode ce folosesc paradigme ale Inteligenței Artificiale, de exemplu Rețelele Neuronale.

4.1 Metoda asemănării

Această metodă (lb. engl. *resemblance*) se foloseşte de o funcție de corelație ce permite calcularea asemănării între două fresce [21]. Metoda a fost testată şi s-a remarcat că principalul obstacol în evaluarea asemănării constă în punctele de reper neclare. Acestea se pot datora zgomotului, erorilor de evaluare sau faptului că unele trăsături ale mediului se modifică pe măsură ce robotul se deplasează – de exemplu, linia unui perete se poate schimba într-o deschidere. De aceea se impune o filtrare cît mai bună a elementelor parazite. Asemănarea între două fresce consecutive se calculează prin considerarea diferențelor ce apar între punctele de reper în fiecare cuadrant al spațiului de observație al robotului de la o frescă la alta. Comparația se face cu ajutorul unui prag predefinit, care prin raportare la diferențele obținute indică dacă o frescă trebuie păstrată sau eliminată.

Algoritmul folosit pentru metoda asemănării este:

1. *construieşte fresca i*
2. *determină numărul de puncte de reper clar definite din fiecare cadran al frescei i, atribuindu-se acestor valori variabilele $N0_i$, $N1_i$, $N2_i$, $N3_i$, pentru cadranele 0, 1, 2, respectiv 3.*
3. *se incrementează valoarea i (i = i + 1)*
4. *construieşte fresca j = i*
5. *determină numărul de puncte de reper clar definite din fiecare cadran al frescei j, atribuindu-se acestor valori variabilele $N0_j$, $N1_j$, $N2_j$, $N3_j$, pentru cadranele 0, 1, 2, respectiv 3.*
6. *calculează asemănarea cu relația:*
   $r_{ij} = N0_i - N1_j + N1_i - N1_j + N2_i - N2_j + N3_i - N3_j$
7. *if $r_{ij} \geq$ prag then:*
   7.1. reține fresca j
   7.2. endif
8. *incrementează valoarea i (i = i + 1)*

4.2 Evaluarea asemănării frescelor prin metoda baricentrului

Metoda de față abordează problema ținând cont de numărul de repere din fiecare cadran şi urmăreşte modificarea acestuia. Acest criteriu are la bază distanța Hausdorff [22], [23] ce măsoară distanța dintre două mulțimi. În cazul de față, metoda baricentrului determină un punct în spațiul celor patru cadrane, pe baza numărului de elemente din fiecare cadran, iar orice modificare al numărului de elemente va determina o deplasare a punctului numit baricentru. Dacă această deplasare este mai mare decât o valoare experimental determinată în prealabil, fresca în cauză este considerată relevantă.

Algoritmul aceste metode:

1. *construieşte fresca i*
2. *determină numărul de puncte de reper clar definite din fiecare cadran al frescei i, atribuindu-se acestor valori variabilele $N0_i$, $N1_i$, $N2_i$, $N3_i$, pentru cadranele 0, 1, 2, respectiv 3.*
3. *calculează numărul total al punctelor de reper clar definite din fiecare cadran al frescei i, $N_{ref} = N0_i + N1_i + N2_i + N3_i$*
4. *se incrementează valoarea i (i = i + 1)*
5. *construieşte fresca j = i*
6. *determină numărul de puncte de reper clar definite din fiecare cadran al frescei j, atribuindu-se acestor valori variabilele $N0_j$, $N1_j$, $N2_j$, $N3_j$, pentru cadranele 0, 1, 2, respectiv 3.*

7. *calculează numărul total al punctelor de reper clar definite din fiecare cadran al frescei j, $N = N0_j + N1_j + N2_j + N3_j$*
8. *calculează baricentrul:*

$$x_{ref} = \frac{N0_i - N2_i}{N_{ref}}$$

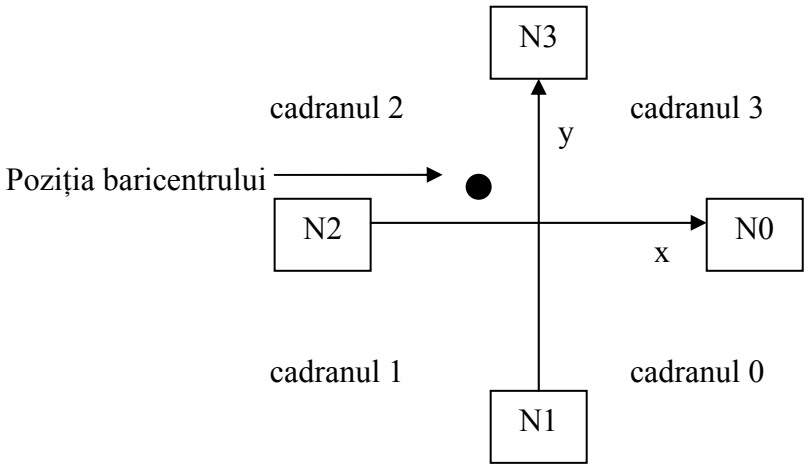
$$y_{ref} = \frac{N1_i - N3_i}{N_{ref}}$$

$$x = \frac{N0_j - N2_j}{N}$$

$$y = \frac{N1_j - N3_j}{N}$$

$$bary_{ij} = \sqrt{(x_{ref} - x)^2 - (y_{ref} - y)^2}$$

9. *if $bary_{ij} \geq prag$ then*
   9.1. reține fresca j
   9.2. endif
10. *incrementează valoarea i (i = i + 1)*



N0: Numărul de repere distincte din cadranul 0
N1: Numărul de repere distincte din cadranul 1
N2: Numărul de repere distincte din cadranul 2
N3: Numărul de repere distincte din cadranul 3

Fig. 4.1. Ilustrarea grafică a principiului baricentrului.

Observații:

Cele două metode au fost testate în medii diferite, pornind de la medii simple, la medii mai complexe, similare cu interiorul unei locuințe. Primul dezavantaj al acestor abordări este faptul că trebuie determinat în prealabil un prag folosit în comparații. Acest prag nu se poate determina şi mai ales fixa după nişte relații, ci empiric. Prin încercările făcute în mediile simple, s-a determinat un prag optim care a fost testat ulterior într-un mediu obişnuit, mai complex, iar în final acesta a fost fixat pe baza acestor teste. Este de remarcat faptul că pentru cele două metode se obțin praguri optime diferite, dar performațele acestor metode sunt similare.

4.3 Distanțe între şiruri de caractere ca metodă de evaluare a relevanței frescelor

De mai mulți ani, redactarea documentelor cu calculatorul este înlesnintă de ajutorul pe care ni-l oferă editoarele de text prin corectarea unor cuvinte scrise greşit sau sugerarea unor variante pentru acestea. Acest lucru util se datorează unor metode de evaluare a distanței între cuvântul presupus greşit şi o

listă de cuvinte cunoscute, păstrată în memoria programului [24]. Cuvintele ce obțin un „scor” cât mai bun într-o astfel de evaluare, sunt propuse pentru corectare.

4.3.1 Distanța Hamming

Distanța Hamming, sau distanța H, se poate defini numai pentru şiruri de aceeaşi lungime [25]. Pentru două şiruri, s1 respectiv s2, distanța Hamming – H(s1,s2) – reprezintă numărul de locuri în care cele două şiruri au caractere diferite (fig. 4.2).

În cazul frescelor, care sunt reprezentate în final prin şiruri de caractere, se poate aplica cu succes această metodă, dat fiind că toate frescele conțin acelaşi număr de caractere, respectiv 64. Din păcate, şi în acest caz, este necesară determinarea în prealabil unui prag de reținere a frescelor.

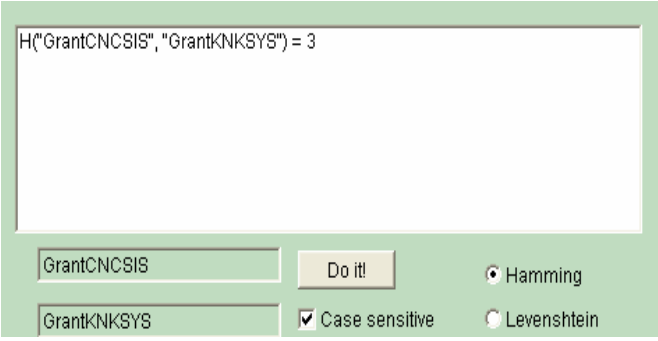


Fig. 4.2. Exemplu de calcul al distanței Hamming.

4.3.2 Distanța Levenshtein

Distanța Levenshtein [26] realizează o evaluare mai complexă decât distanța Hamming. Aceasta poate să opereze şi cu şiruri de lungimi diferite, şi contabilizează diferențele dintre două şiruri date nu doar din punct de vedere al diferențelor ca şi distanța H, ci şi dacă un şir conține un caracter, iar celălalt nu.

Între două şiruri, distanța Levenshtein contabilizează pe rând înlocuirile de caractere, inserțiile, şi eliminările acestora. Rezultatul acestei evaluări este reprezentat prin minimul dintre numărul de înlocuiri, inserții şi eliminări.

$$LD(s1; s2) = \min (n_{ins} + n_{del} + n_{subst})$$

O generalizare a distanței Levenshtein o reprezintă distanța Levensthein generalizată [27] (WLD, *Weighted Levenshtein Distance*), cunoscută şi sub denumirea de distanța de editare [28] (*Edit Distance*). Diferența se manifestă prin faptul că diversele operații de editare pot avea costuri/ponderi diferite şi nu unitare cum presupunea metoda originală:

$$WLD(s1; s2) = \min (w_{ins}n_{ins} + w_{del}n_{del} + w_{subst}n_{subst})$$

Termenul distanță de editare este uneori utilizat pentru un caz special, şi anume când inserția şi ştergerea au costuri unitare iar substituția este considerată ca o pereche ştergere-inserție, avnd deci ponderea 2 [29].

Deoarece această metodă este mai sofisticată, are un algoritm mai complex prin urmare este mai lentă. Acest fapt poate influența în mod important navigația robotului, limitând viteza de reacție a acestuia, mai ales şi datorită faptului că pe lângă aceste operații din cadrul procesului de selecție al frescelor se mai fac şi alte numeroase calcule pretențioase.

4.3.3 Distanța dintre caracteristici

Cunoscută în literatură drept *Feature distance* [30], [31] arată numărul de trăsături/caracteristici în care diferă două şiruir de caractere. Pentru stringuri, N-gramele (stringuri de N consecutive simboluri) sunt o alegere potrivită pentru a reprezenta caracteristici:

FD(s1; s2) = max(N1;N2) - m(s1; s2)

în care N1 şi N2 reprezintă numărul N-gramelor din stringurile s1 respectiv s2 iar m(s1; s2) reprezintă numărul de N-grame ce se potrivesc. Dacă un string este mai lung decât altul atunci N-gramele în plus sunt deasemenea calculate ca şi diferențe.

În general, distanța Levensthein conduce la rezultate uşor mai bune decât distanța dintre caracteristici, dar cu costul creşterii complexității calculelor [32].

4.4 Metoda intercorelație dintre stringuri

Funcția de intercorelație este utilizată în principal în procesările de imagini sau de semnal.
În contextul simbolurilor/şirurilor de caractere metoda clasică ce operează cu numere va trebui modificată astfel încât să poată opera cu stringuri. Practic, funcția va compara „cuvântul" *a* (de lungime *m*) cu *b* (de lungime $n \geq m$) şi va produce un vector W de lungime $l = m + n - 1$ cu elementele Wi (cu i = 0, 1, …, l-1) definite de ecuația:

$$Wi(a,b) = \begin{cases} \sum_{j=0}^{i} S(a_j, b_{(n-1)-i-j}), daca\ i = 0...m-1 \\ \sum_{j=0}^{m-1} S(a_j, b_{(n-1)-i-j}), daca\ i = m...n-1, m \neq n \\ \sum_{j=0}^{(l-1)-i} S(a_{(n-1)-(l-1-i)+j}, b_j), daca\ i = n...l-1 \end{cases}$$

unde

$$S(x,y) = \begin{cases} 1\ daca\ x = y \\ 0\ daca\ x \neq y \end{cases}$$

Această funcție compară două şiruri de simboluri aliniindu-le şi examinând perechile de simboluri corespondente din cele două şiruri.

Elementul vectorului	W_0	W_8	W_9
Alinierea stringurilor	GrantCNCSIS 1GrantCNCSYS 0	GrantCNCSIS 1GrantCNCSYS 000001001	GrantCNCSIS 1GrantCNCSYS 011111111101
Scor	0	2	10

Tab. 1. Exemplu de calcul al elementelor vectorului intercorelației dintre stringurile: *GrantCNCSIS* şi *1GrantCNCSYS*.

4.5 Metode bazate pe tehnici ale Inteligenței Artificiale

Dintre paradigmele Inteligenței Artificiale (Rețele Neuronale Artificiale, Logica Fuzzy, Algoritmi Genetici, Învățarea prin Întărire etc.) Rețele Neuronale Artificiale (RNA) oferă cele interesante soluții relativ la problema navigației roboților mobili bazată pe prelucrărilor frescelor.

Problema principală constă aceea că, în mod obişnuit, RNA operează cu numere şi nu cu simboluri/stringuri. Ca atare, rezultă două posibile abortări:

- convertirea simbolurilor în valori numerice şi prelucrearea acestora din urmă cu arhitecturi neuronale clasice;
- conceperea unor arhitecturi ale RNA (structuri + algoritmi de învățare) dedicate operării în mod direct cu simboluri/stringuri.

În cele ce urmează vor fi prezentate succint cele două posibilități.

4.5.1 Procesarea reprezentărilor simbolice prin arhitecturi RNA clasice

Din analiza literaturii aferente RNA s-a constatat că arhitectura cea mai potrivită pentru selecția frescelor este constituită de tip Kohonen, cunoscute şi sub denumirea de RNA cu Hartă de Trăsături cu Autoorganizare (SOFM-NN, Self Organizing Feature Map – Neural Network).

Se vor prezenta în continuare succint principiile de funcționare ale RNA de tip SOFM. Pentru detalii a se consulta [33] – [35].

Scopul acestei arhitecturi constă în transformarea unui tipar de intrare de dimensiune arbitrară într-o hartă de trăsături (spațiu discret, de regulă 1D sau 2D) ordonată topologic. Cu alte cuvinte, există o corespondență între tipul (caracteristicile) tiparului aplicat la intrare şi locația spațială a neuronului care va fi activat. Corespondența topologică se manifestă în sensul în care la tipare similare aplicate la intrarea RNA vor fi activați neuroni situați în aceeaşi vecinătate (lob sau bulb) a stratului de ieşire.

Arhitectura RNA-SOFM 2D este prezentată în fig.4.3.

Algoritmul de antrenament presupune următorii paşi:

a) Inițializarea ponderilor. Se aleg valori aleatoare mici pentru ponderile sinaptice $w_j(0)$.

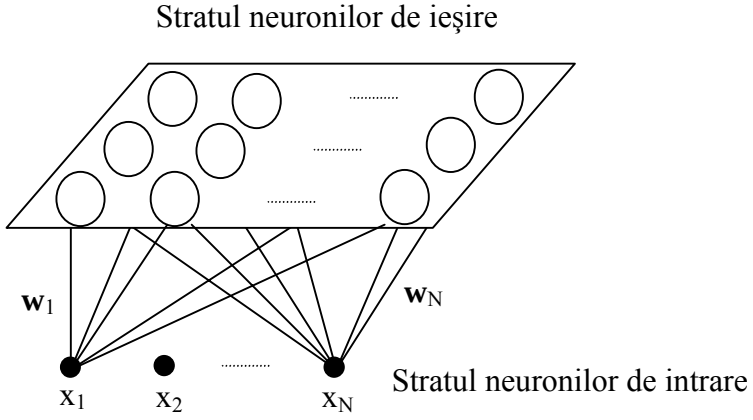


Fig. 4.3. Arhitectura unei RNA-SOFM 2D.

b) Desemnarea neuronului câştigător. Se aplică tiparul x la intrarea RNA, iar pe baza acestuia este selectat neuronul câştigător al competiției:

$$i(\mathbf{x}) = \arg_j \min \left\| \mathbf{x}(n) - \mathbf{w}_j \right\|, \; j = 1,2,...,N$$

c) Ajustarea ponderilor:

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)[\mathbf{x} - \mathbf{w}_j(n)], \text{ daca } j \in \Lambda_{i(x)} \\ 0, \qquad \text{altfel} \end{cases}$$

în care η(n) reprezintă rata de învățare, iar $\Lambda_{i(x)}(n)$ reprezintă vecinătatea topologică a neuronului învingător *i(*x*)*. Dacă se notează cu $\pi_{j,i(x)}$ amplitudinea vecinătății topologice, atunci relația de mai sus poate fi rescrisă astfel:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n)\pi_{j,i(x)}(n)[\mathbf{x}(n) - \mathbf{w}_j(n)]$$

De regulă η(n), $\Lambda_{i(x)}(n)$ şi $\pi_{j,i(x)}$ sunt mărimi dinamice, care variază de-a lungul epocilor de antrenament:

$$\pi_{j,i} = \exp(-\frac{d_{j,i}^2}{2\sigma^2})$$

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_1}\right)$$

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_2}\right)$$

în care $d_{j,i}$ reprezintă distanța de la neuronul "j" la neuronul câştigător "i" iar $\eta_0, \sigma_0, \tau_1, \tau_2$ sunt constante.

Procedura se repetă de la pasul b) de un număr de ori specificat apriori sau până când nu se mai înregistrează schimbări notabile în harta de trăsături .

Aşa cum s-a specificat anterior, problema principală este transformarea elementelor constituente ale frescelor, adică a simbolurilor, în numere naturale. Există numeroase procedee menționate în literatură care se pot folosi în această transformare, unele dintre ele ținând cont de relațiile de ordonare dintre caractere/stringuri ([36]-[39]).

Întru-cât, în cazul nostru particular, numărul maxim de simboluri ce constituie o frescă este de 16, adică o codare de tip hexazecimal (0-9, A-F). Se poate constitui vectorul de intrare pentru RNA în următoare moduri:

a) Codare directă:
- fiecărui simbol îi va corespunde codarea sa binară echivalentă (Ex: 0 = 0000, 1=0001,…,F=1111);

b) Codare exclusivă:
- fiecare simbol este codat cu un vector binar (Ex: 0 = 0…0001, 1=0…0010,…, F=10…0) ce conține doar un element egal cu 1 restul elementelor vectorului fiind 0.

În final, o frescă va fi reprezentată de un vector binar obținut prin concatenarea vectorilor binari corespunzători simbolurilor constituente.

4.5.2 Procesarea reprezentărilor simbolice prin arhitecturi RNA ce operează în mod direct cu simboluri

O idee diferită față de abordarea anterioară o reprezintă organizarea stringurilor în matrici SOFM, astfel încât această organizare să reflecte o măsură a unei distanțe [40] - [43]. În primul rând trebuie definită „similaritatea" sau „distanța" dintre obiecte (în cazul nostru stringuri) şi mai apoi găsirea unor prototipuri reprezentative pentru stringuri.

În antrenamentul SOFM, doi paşi sunt repetați:

- găsirea, pe baza similarității/distanței dintre stringuri, unității ce se potriveşte cel mai bine pentru fiecare element al datelor şi adiționarea elementului la lista unității respective;
- modificarea modelelor nodurilor SOFM: găsirea elementului „median" al uniunii listelor (datele listelor conțin elementele tuturor nodurilor situate în vecinătatea nodului considerat).
  Fie, de exemplu, stringurile s1, s2, s3. Pentru calculul valorii mediane a stringurilor se procedează în felul următor: se alcătuieşte o matrice cu distanțele dintre fiecare două stringuri, de exemplu:

	s1	s2	s3
s1	0	4	1
s2	4	0	2
s3	1	2	0

apoi se calculează suma distanțelor de la un string la toate celelalte:

s1: 0 + 4 + 1 = 5
s2: 4 + 0 + 2 = 6
s3: 1 + 2 + 0 = 3

Cea mai mică sumă de distanțe este pentru elementul s3, deci el reprezintă valoarea mediană a setului de date. În cazul SOFM, distanțele pot fi ponderate în funcție de o anumită formă de vecinătate, aşa cum s-a arătat în paragraful precendent.

Aceasta este modalitatea „lot” de ajustare a hărții de trăsături, ceea ce înseamnă că toate datele sunt prezentate SOFM înainte de a trece la modificarea modelului. În final se obține o hartă de trăsături ce conține în nodurile rețelei neuronale stringuri prototip (fig. 4.4).

certainly	ceainly	saafely	safety	eaited	excited	excited	eemaind	remains	remains
certainly	sainly	safety	safety	safetd	excitd	excind	remats	remains	remains
stfnly	safety	safety	safed	excied	excited	remtnd	remains	remains	remains
toecy	tety	safeh	safed	saied	emaes	remins	remains	remains	remains
touch	touch	touch	ouch	aiay	elaiys	alays	remains	remains	remains
touch	touch	touch	ouch	aluas	alays	always	always	aeways	remais
touch	touch	touch	ouyh	alwas	always	always	always	aeways	depest
tninh	tancg	meanih	mewing	always	always	always	dewayt	deepest	deeest
meaning	meang	meanig	mening	meanis	deways	depeayt	deeest	depest	deeest
meaning	meaning	meaning	meaning	mening	mepint	depest	deepest	depest	deepest

Fig. 4.4. Hartă de trăsături cu autoorganizare, varianta cu stringuri

5. REZULTATE EXPERIMENTALE

Aşa după cum s-a specificat anterior, se pune problema selecţiei unui număr cât mai mic de fresce care să fie reprezentative pentru un anumit mediu. În continuare se vor prezenta rezultatele experimentale obţinute în implementarea câtorva din metodele de selecţie ale fresceclor discutate din punct de vedere teoretic în capitolul anterior. Rezultatele din acest capitol se bazează pe un eşantion real de fresce (Tab. 5.1), extras de la robotul de tip ARMAGRA prezentat în Cap.3, care evoluează într-un mediu structurat (fig. 5.1) reprezentat de incinta unui laborator de cercetare .

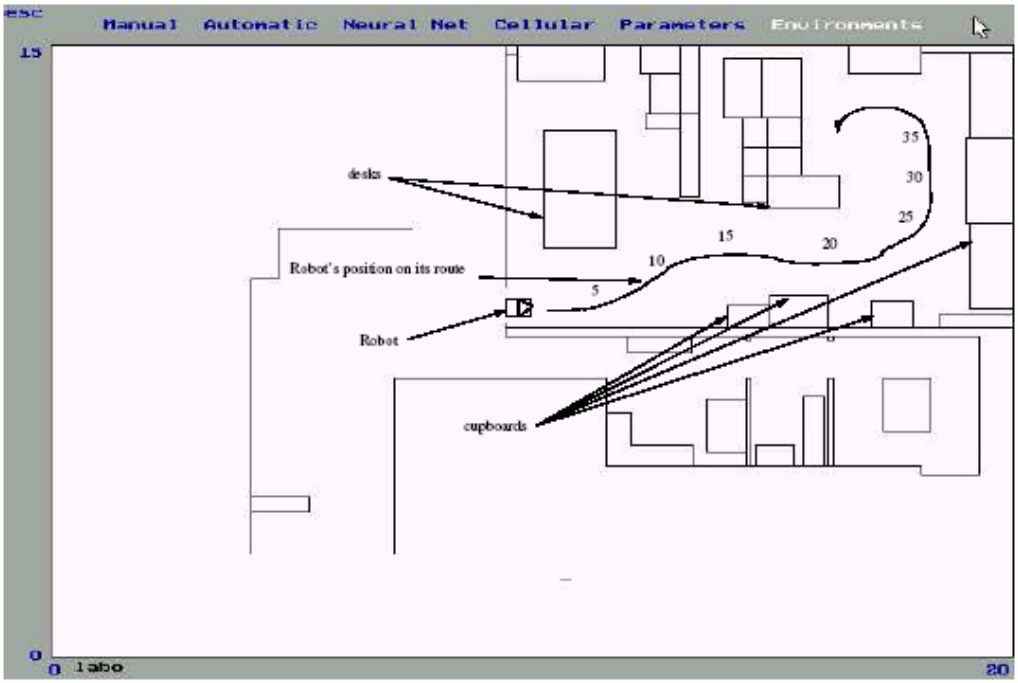


Fig. 5.1. Mediul pentru care a fost extras un număr de 25 de fresce.

E00C004000000F6EC700006AC4500000E000020000900000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E29000000000000000000000000F0
0CC0004F74000000004CA6006000EC40000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC04000000000000000000009000BF040E00
05C000F0704004C00700000F04C0E00050000000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE00050000000048400000900090004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE00005000000990000000000000004CE6
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C00040000000000004C7F4C000000E000504C40000000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000000500
6000E650080004400004C0070F5E6006E005000000000000000000000005E0
BB000E6F600000000FF004C000000000E005000000000000000000000F00
B00000E6F6000000000FF004C0000000E00500000000000000000090B0B

Tab. 5.1 Cele 25 de fresce de lungime 64 de simboluri extrase din mediul prezentat în fig. 5.1.

Rezultatele au fost obținute prin implementarea unor programe în mediul de dezvoltare MATLAB, programele aferente fiecărei metode descrise fiind prezentate în anexele acestei lucrări.

5.1 Metoda asemănării – rezultate experimentale

Această metodă are un prim dezavantaj prin faptul că nu ține cont de natura reperelor, sau mai concret de diferențele dintre caracterele celor două şiruri. De exemplu, două fresce complet diferite, dar cu acelaşi număr de repere ar fi considerate identice folosind acest algoritm, şi nu ar fi păstrate, lucru ce ar conduce la pierderea unor informații utile.

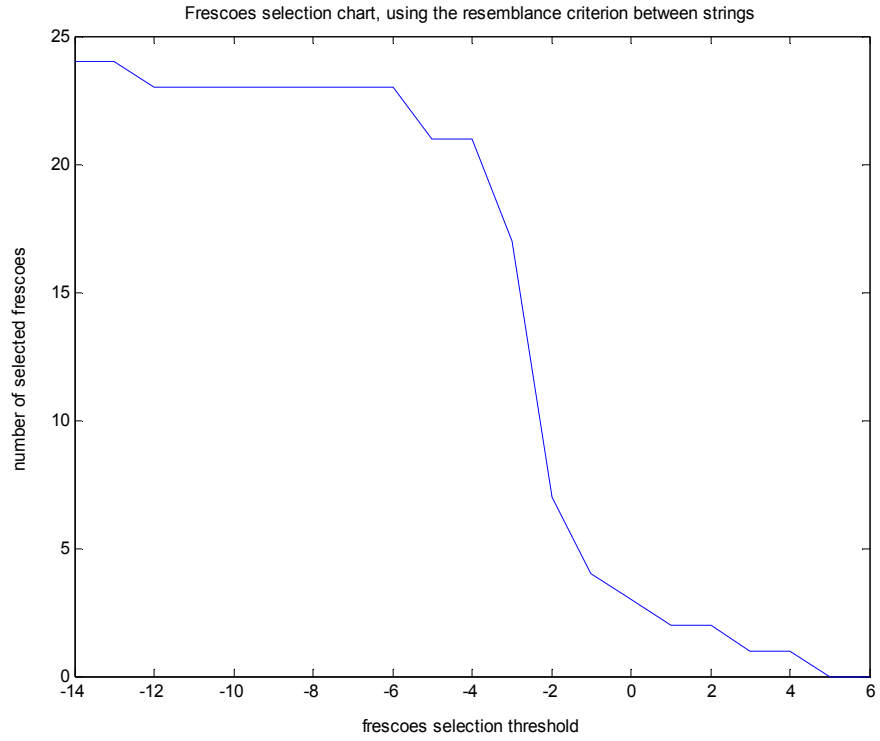


Fig. 5. 2. Dependența numărului de fresce semnificative de pragul de selecție – metoda asemănării.

Pentru a evalua această metodă s-a implementat algoritmul prezentat în Cap.4 (cf. Anexa nr.1, programul **frsel_resmbl.m**) şi s-a trasat caracteristica din fig. 5.2 (cf. Anexa nr.1, programul **chart_resemblance.m**) care reprezintă dependența între numărul de fresce selectate şi pragul ales pentru selecție, eşantionul utilizat totalizând 25 de fresce. Deşi este greu de stabilit empiric un prag optim, este de dorit ca numărul de fresce selectate să reprezinte un procent cât mai redus din numărul total (ex. 25%). De aceea se poate considera optim pragul -2, pentru care se obțin 7 fresce din setul de 25. Restul valorilor oferă fie prea puține fresce fie prea multe. În tab. 5.2 este prezentat eşantionul folosit şi frescele selectate prin această metodă:

E00C004000000F6EC700006AC4500000E000020000900000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E29000000000000000000000000F0
0CC0004F74000000004CA6006000EC40000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC04000000000000000000009000BF040E00
05C000F0704004C00700000F04C0E00050000000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE00050000000484000009000 9004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE00005000000990000000000000004CE6
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C00040000000004C7F4C000000E000504C400000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000500
6000E650080004400004C0070F5E6006E00500000000000000000000005E0
BB000E6F600000000FF004C000000000E00500000000000000000000000F00
B00000E6F6000000000FF004C0000000E00500000000000000000000090B0B

Tabelul 5.2. Frescele selectate prin metoda asemănării

Se poate uşor observa ineficiența aceste metode, prin faptul că şirurile selectate nu doar că sunt foarte similare, dar nici nu sunt reprezentative pentru tot eşantionul dat. Un alt dezavantaj este faptul că pragul trebuie determinat în prealabil, şi există puține variante de praguri convenabile, datorită variației foarte rapide a numărului de fresce selectate tocmai în zona pragurilor „optime”.

5.2 Metoda baricentrului – rezultate experimentale

Această metodă, descrisă anterior, seamănă mult cu metoda anterioară prin modul de manipulare al şirurilor, şi diferă doar prin modul de calcul al similitudinii între şiruri (cf. Anexa nr.2, progr. **frsel_barycenter.m**). Deasemenea, ca şi în cazul metodei asemănării, se contabilizează numărul reperelor şi nu se ține cont şi de natura acestora, fapt ce dezavantajează descrierea calitativă efectuată.

Rezultatele acestei metode se pot considera modeste, dar mai bune decât în cazul metodei asemănării. Totuşi, această metodă prezintă acelaşi dezavantaj, al unui prag care trebuie determinat în prealabil, prin determinări mai mult sau mai puțin empirice.

Metoda baricentrului situează pragul optim de selecție între valorile 0.03 şi 0.05 (cf. Anexa nr.2, programul **chart_barycenter.m**). Rezultatele obținute pe eşantionul dat pentru un prag din această zonă pot fi văzute în fig. 5.3 respectiv tabelul 5..3.

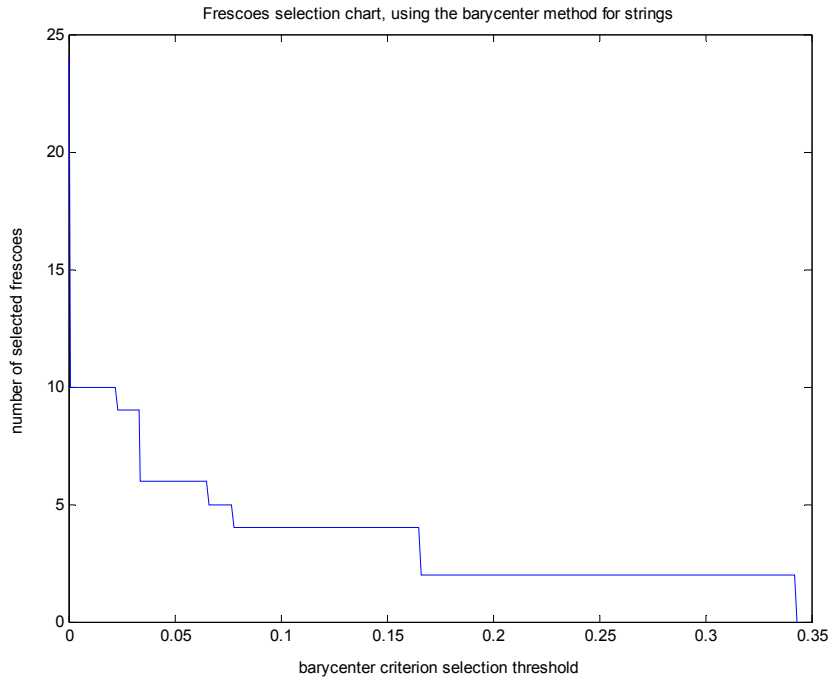


Fig. 5. 3. Dependența numărului de fresce semnificative de pragul de selecție – metoda baricentrului.

E00C004000000F6EC700006AC4500000E000020000900000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E2900000000000000000000000F0
0CC0004F74000000004CA6006000EC4000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC040000000000000000009000BF040E00
05C000F0704004C00700000F04C0E000500000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE000500000000484000009000 9004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE000050000009900000000000004CE6
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C000400000000004C7F4C000000E000504C40000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000000500
6000E650080004400004C0070F5E6006E005000000000000000000000005E0
BB000E6F600000000FF004C000000000E005000000000000000000000000F00
B00000E6F6000000000FF004C0000000E00500000000000000000000090B0B

Tab. 5.3. Frescele selectate prin metoda baricentrului.

5.3 Distanța între şiruri de caractere ca metodă de evaluare a relevanței frescelor

Aceste metode presupun o serie de evaluări calitative a şirurilor, mult mai potrivite aplicației de față. De aceea, prin natura metodelor folosite, rezultatele sunt mai bune decât în cele două cazuri anterioare.

5.3.1 Distața Hamming – rezultate experimentale

Metoda compară două şiruri caracter cu caracter (cf. Anexa nr. 3, programul **frsel_hamming.m**), rezultând un numărul de caractere ce diferă între cele două şiruri. Acest rezultat poate fi interpretat, ca în programul utilizat în această lucrare, sub formă procentuală, drept procentul de asemănare dintre cele două şiruri între care se evaluează distanța H. Alegerea unui prag de asemănare pentru selecție, sub formă procentuală, este mult mai intuitivă decât în celelalte cazuri, exprimând totodată toleranța selecției.

În cazul de față s-a ales din caracteristica din fig. 5.4 (cf. Anexa nr. 3, programul **chart_hamming.m**) un prag minim de 52% diferențe între două fresce. Cu alte cuvinte, orice frescă care este cu peste 52% mai diferită decât cea anterioară, este păstrată. Astfel a rezultat selecția din tab. 5.4. Se poate observa că acestă metodă de selecție este mai bună iar atât utilizarea ei cât şi interpretarea rezultatelor sale, sunt mai intuitive decât metodele anterioare.

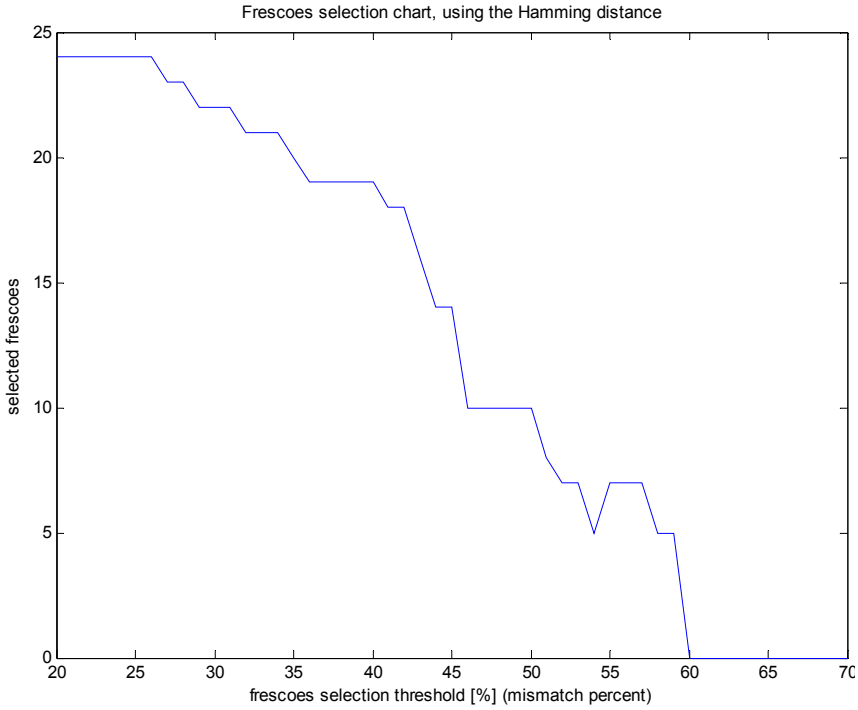


Fig. 5. 4. Dependența numărului de fresce semnificative de pragul de selecție – metoda distanței Hamming

E00C004000000F6EC700006AC4500000E0002000090000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E29000000000000000000000000F0
0CC0004F74000000004CA6006000EC40000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC040000000000000000009000BF040E00
05C000F0704004C00700000F04C0E00050000000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE00050000000484000009000 9004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE00005000000990000000000000004CE6
E00C00040000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C00040000000004C7F4C000000E000504C40000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000000500
6000E650080004400004C0070F5E6006E005000000000000000000000005E0
BB000E6F600000000FF004C000000000E00500000000000000000000000F00
B00000E6F6000000000FF004C0000000E005000000000000000000000090B0B

Tab. 5.4. Frescele selectate prin metoda distanței Hamming, prag 52%.

5.3.2 Distanța Levenshtein – rezultate experimentale

Fiindcă distanța Levenshtein (cf. Anexa nr.4, programul **editdist.m**) este o metodă mai complexă şi flexibilă decât cele anterioare, s-au încercat două abordări. Prima, foloseşte un algorim clasic al distanței Levenshtein (cf. Anexa nr.4, programul **frsel_editdist.m**), cu rezultate foarte bune, dar, datorită complexității algoritmului, această metodă este vizibil mai lentă decît celelalte.

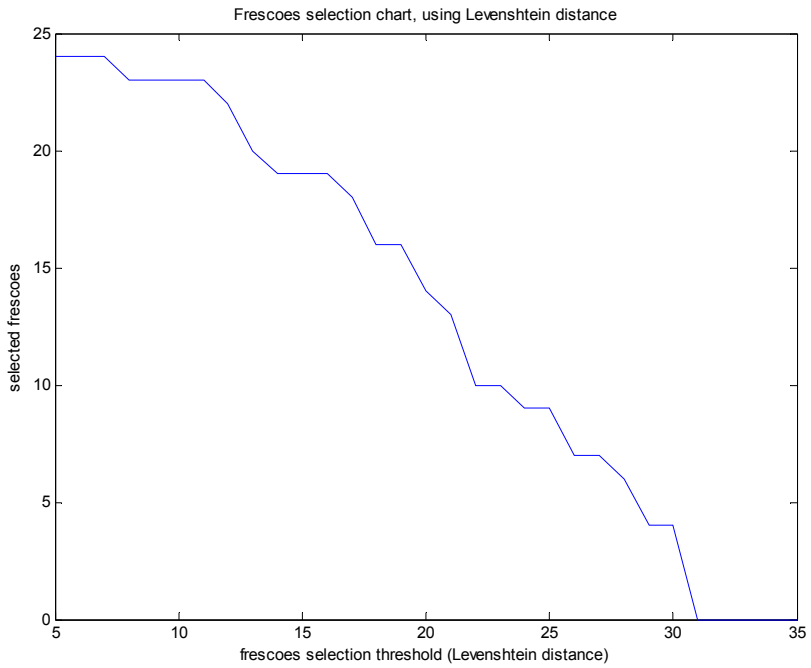


Fig. 5. 5. Dependența numărului de fresce semnificative de pragul de selecție – metoda distanței Levenshtein.

A doua implementare a acestei metode, doreşte o optimizare a timpului de execuţie al programului de selecţie, pe baza analizei structurii setului de fresce. (cf. Anexa nr.4, progr. **frsel_editdistfast.m**). Analizând fresce succesive, s-a observat că datorită deplasării robotului frescele nu îşi schimbă foarte mult informaţia din punct de vedere calitativ, într-un timp relativ scurt. Între unele repere se interpun zerouri, adică celule inactive ce reprezintă în mediul real spaţii, deoarece pe măsură ce robotul se deplasează, perspectiva lui asupra obiectelor se schimbă – dacă dintr-un unghi două repere păreau apropiate, din altul vor arăta că există un spaţiu între ele. În mod evident, aceste fresce sunt practic identice, ca şi conţinut, şi de aceea se consideră că spaţiile au o importanţă mai mică faţă de repere. Din acest motiv, metoda „optimizată" scurtează considerabil şirurile de comparat prin algoritmul distanţei Levenshtein, eliminând zerourile doar pentru această operaţie, fără a altera ireversibil frescele. Micşorând lungimea şirurilor, timpul de execuţie al programului este mult redus, iar rezultatele sunt la fel de bune ca şi în cazul metodei iniţiale.

E00C004000000F6EC700006AC4500000E000020000900000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E290000000000000000000000000F0
0CC0004F74000000004CA6006000EC400000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC040000000000000000009000BF040E00
05C000F0704004C00700000F04C0E00050000000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE000500000000484000009000 9004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE000050000009900000000000004CE6
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C000400000000004C7F4C000000E000504C40000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000000500
6000E650080004400004C0070F5E6006E00500000000000000000000000005E0
BB000E6F600000000FF004C000000000E00500000000000000000000000F00
B00000E6F6000000000FF004C0000000E005000000000000000000000090B0B

Tab. 5.5. Frescele selectate prin metoda distanţei Levenshtein.

Selecţia frescelor din tabelul 5.5, marcate cu bold, s-a obţinut cu un prag de 27 (cf. Anexa nr.4, programul **chart_editdist.m**), cu care se poate face o selecţie convenabilă.

Deoarece caracteristica are o tendinţă liniară, se poate observa că pragul optim se află undeva la mijlocul domeniului pragului, între 20 şi 25. Pentru valori mai mari selecţia este foarte drastică, pierzându-se şi fresce relevante.

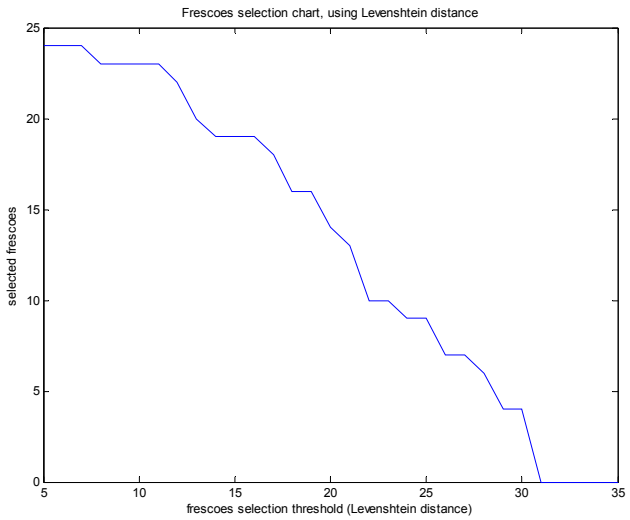


Fig. 5.6 Caracteristica metodei de selecție a frescelor cu distanța Levenshtein

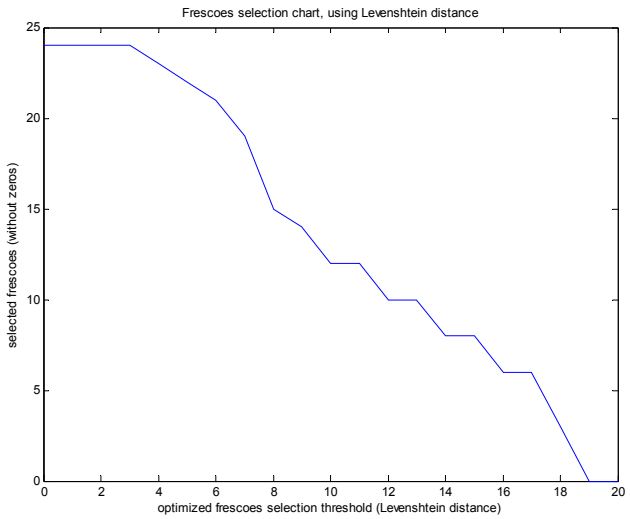


Fig. 5.7 Caracteristica metodei de selecție a frescelor optimizate, cu distanța Levenshtein

0C500F700492C0007000F0004C00EC0400000000000000000000009000BF040E00 0400000000F6EC76A6F4C7000F4CE0005000000004840000090009004CE66E0C E00C0004000000000004C70F04C760000E000050004C40000000000000004CE6F 00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000 000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000 004C4400E000C0504400F6E0C000070F0F6E606A60FFF0000000000000000000 E0000C500044000004C00070F91E6006E06F0000000000000000000000000500	Secvența selectată cu prima variantă („clasică”) de implementare a selecției cu distanță Levenshtein
0400000000F6EC76A6F4C7000F4CE0005000000004840000090009004CE66E0C E0000C0000040000000000004CE6F4C0E000054C44C4000000004C4F0F004000 00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000 000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000 004C4400E000C0504400F6E0C000070F0F6E606A60FFF0000000000000000000 BB000E6F600000000FF004C000000000E005000000000000000000000000F00	Secvența selectată cu a doua variantă (optimizată, mai rapidă) de implementare a selecției cu distanță Levenshtein.

Tab. 5.6. Frescele selectate prin metode de tip distanță Levenshtein.

Din prezentarea comparată de mai sus, se pot observa performanțele foarte apropiate ale celor două implementări, pentru aproximativ acelaşi număr de fresce selectate.

Pe ansamblu, cele mai bune rezultate se obțin cu ajutorul metodei Levenshtein, indiferent de abordare, urmată de metoda de selecție Hamming, care pe lângă rezultatele destul de bune pe care le oferă şi accesibilitatea criteriului de selecție, are şi cel mai simplu algoritm.

5.4 Metoda intercorelație dintre stringuri – rezultate experimentale

O abordare din punct de vedere al corelației (cf. Anexa 5, programul **frsel_corr.m**) definite pentru stringuri presupune selecția frescelor la care similaritatea este mai mică decât un anumit prag.

E00C004000000F6EC700006AC4500000E000020009000000000000000000040
E000C005F740009002A6000000600000E00209000000000000000000000040
E000CC004F74000000004C0B00060000E02900000000000000000000000040
6E00CC0674F704000000004CA6006000E29000000000000000000000000F0
0CC0004F74000000004CA6006000EC40000000000000000000000009B6E00
C05F704F700CE6F0000064CA060E00290000000000000000000000F006A6E
0C500F700492C0007000F0004C00EC040000000000000000009000BF040E00
05C000F0704004C00700000F04C0E00050000000000000000F00F0000040E00
0400000000F6EC76A6F4C7000F4CE000500000000484000009000 9004CE66E0C
E0C00400000000F6EC0B0F4C70000F4CE000050000009900000000000004CE6
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
0E000C000400000000004C7F4C000000E000504C40000000000000064CEF40
E0000C000004000000000004CE6F4C0E000054C44C4000000004C4F0F004000
0A0000060000C040000000004CE6F500EC0044C40000F0F0000000FF0004C4F0
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
00000400000E0C00400F00006E0C000076E6F0F000BB000006E29FF000000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF0000000000000000000
400E0000C05044000F6EC00002BFF06E6000BBFFF000000000000000000000000
E000C0504400F6E0C000070F0F6E606A60FFF000000000000000000000000500
E0000C500044000004C00070F91E6006E06F000000000000000000000000500
6000E650080004400004C0070F5E6006E005000000000000000000000005E0
BB000E6F600000000FF004C000000000E005000000000000000000000000F00
B00000E6F6000000000FF004C0000000E00500000000000000000000090B0B

Tab. 5.7. Frescele selectate prin metoda corelației dintre stringuri, prag 35.

Se constată un număr de fresce selectate comune cu metodele de tip Levenshtein (Tab. 5.8). Dependența de prag a numărului de fresce selectate este prezentată în fig. 5.8 (cf. Anexa 5, programul **chart_corr.m**).

0400000000F6EC76A6F4C7000F4CE000500000000484000009000 9004CE66E0C
E00C000400000000004C70F04C760000E000050004C40000000000000004CE6F
00005000000EC4F0000000600E0CE00066E60F0F00000FF00006E84FF0000000
000000E67600C0000000B0F04C00EC00040F0B094C40004840900B0000000000
00600E0C00000076E6F04C0EC000400F0F048400000484FFF000000000000000
004C4400E000C0504400F6E0C000070F0F6E606A60FFF0000000000000000000
B00000E6F6000000000FF004C0000000E00500000000000000000000090B0B

Tab. 5.8 Fresce comune selecțiilor tip Levenshtein şi corelație.

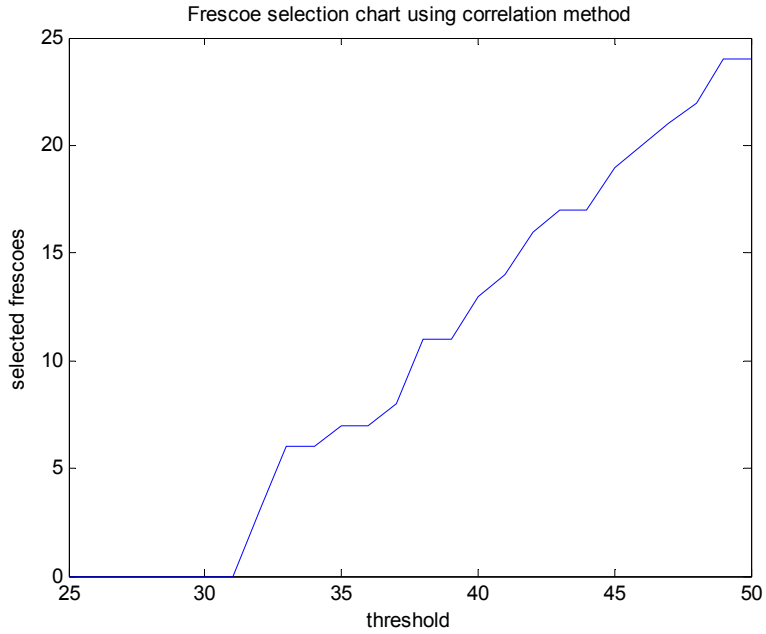


Fig. 5. 8. Dependența numărului de fresce semnificative de pragul de selecție – metoda corelației dintre stringuri.

5.5 Metoda bazată pe arhitectura RNA-SOM clasică – rezultate experimentale

Această abordare fundamental diferită de cele anterioare se bazează pe proprietatea fundamentală a RNA cu autoorganizare tip hartă de trăsături şi anume aceea de a forma o corespondență topologică între tipul vectorului aplicat la intrarea rețelei neuronale şi poziția spațială a neuronul de ieşire câştigător al competiției. Această corespondență este făcută în sensul că la tipare asemănătoare aplicate la intrare se vor activa neuroni de ieşire invecinați din punct de vedere spațial.

Aşa după cum s-a explicat în prezentarea teoretică, deoarece s-a folosit o RNA-SOM clasică, ce operează în mod direct cu numere, se impune mai întâi o conversie a frescelor în vectori de intrare de tip binar. Deoarece s-au folosit în experimentele precedente un număr de 25 de fresce, fiecare cu câte 64 de elemente (simboluri) va rezulta o matrice a vectorilor binari de intrare aplicați rețelei RNA-SOM de dimensiune 25 de coloane şi 64 elemente x 4 biți = 256 linii (tab. 5.9).

11100100011010000000011011
11110100011010000000111100
11110000011010000000011111
00000000000000000000000011
00011010000101000000000010
00011011100100000000000000
00010000000101000000000010
00000001000000000000000010
…….
00000000000100000000011111
00010000010001000000000100
11110100011101000000000100
00010100011001000000000100
00010000000001000000000000
00000100101000000000000001
00000100111000000000000000
00000100011000000000000001
00000000001000000000000001

Tab. 5.9. Colecția de 25 de fresce convertită în vectori binari. Rezultă o matrice cu 256x25 elemente.

Apoi trebuie stabilit numărul de neuroni din stratul de ieşire ce coincide cu numărul de fresce prototip selectate din mulțimea frescelor de intrare. S-a ales ca acest număr să fie egal cu 7, pentru ca să se poată copara cu rezultatele obținute cu metodele anterioare. Practic vecorii pondere ai acestor neuroni vor conține, sub formă numerică, frescele selectate. Pentru a vedea ce vectori prototip s-au format în RNA-SOM va trebui să se facă o conversie inveră vectori binari – fresce. După un proces de antrenament de 500 de epoci, vectorii prototip rezultați pot fi observați în tab. 5.10 (cf. Anexa 6, programul **frsel_clasicSOM.m**).

E000C4000004040000000000E0E6000E025000000000000000000000000440
4000040000040400000000004E0EE000E000004400000000000000004000000000
0000040000004040040000004004E00C6400054C000000000000000000000000
A040040000000050440044040C000000CE004000000008000000000000000000
E000000460000000006004040400000000E000000000000000000000000000A40
8000004460000000000064400440000040000000000000000000000000004E00
0C400060740000000000A0004400EC000000000000000000000000000040E00
Tab. 5.10. Vectorii prototip rezultați din procesul de antrenament al RNA-SOM.

Se constată că vectorii prototip nu sunt total identici cu elementele mulțimii frescelor. Acest lucru se explică prin modalitatea de formare pentru aceşti vectori (vezi § 4.5.1). În concluzie, din cauza unei redundanțe scăzute a vectorilor, această metodă nu dă rezultate satisfăcătoare.

6. CONCLUZII

6.1 Contribuții în domeniul navigației roboților mobili bazate pe reprezentarea simbolică a mediului

Cercetările aferente acestui grant s-au realizat în contextul cooperări dintre Universitatea POLITEHNICA Timişoara, Facultatea de Electronică şi Telecomunicații şi Laboratorul de Sisteme Complexe din cadrul Universității Val d'Essone din Evry, Franța privind modalități de navigare pentru generația de roboți de tip ARMAGRA dezvoltate de partea franceză.

În cadrul acestui grant s-au realizat următoarele:

- În introducere, o trecere în revistă a tehnicilor utilizate în navigația roboților autonomi, un accent deosebit punându-se pe cele ce folosesc repere naturale şi sunt bazate pe reprezentarea simbolică a mediului. Întru-cât elimină numeroase inconveniente legate de reprezentările metrice, s-a ajuns la concluzia conform căreia reprezentarea calitativă a mediului unui robot autonom cu ajutorul limbajului frescelor este foarte utilă mai ales în cazul în care punctele de reper sunt suficient de distanțate de robot.

  În mod cert navigația pe baza reprezentării simbolice va putea conduce la un dialog om-maşină cât mai apropiat de limbajul natural.

- S-a propus o nouă formă de reprezentare a unei fresce: ea să fie formată din 16 reprere x 4 cadrane = 64 de simboluri. Această nouă reprezentare conduce la fresce de lungime egală, mai uşor de comparat. Creşte de asemenea şi informația referitoare la o frescă deoarece apare o localizare clară a unui anumit reprer într-un anumit cadran, la un anumit unghi.

- S-au identificat şi investigat numeroase metode posibil a fi folosite pentru selecția frescelor semnificative:

- Metoda asemănării;
- Metoda baricentrului;
- Distanța Hamming
- Distanța Levenshtein
- Distanța dintre caracteristici
- Metoda intercorelație dintre stringuri
- Procesarea reprezentărilor simbolice prin arhitecturi RNA clasice
- Procesarea reprezentărilor simbolice prin arhitecturi RNA ce operează în mod direct cu simboluri

  De menționat că aceste metode sunt potrivite şi pentru problema identificării traiectoriei retur.

- S-au implementat în cod MATLAB metodele menționate anterior pentru a putea fi evaluate comparativ performanțele acestora. Din experimente s-au desprins informații legate de acuratețea selecției, dependența număr de fresce selectate - prag, valorile optime pentru pragurile de selecție a frescelor semnificative, timpii de execuție etc.

- S-a propus o nouă versiune a metodei Levenshtein, denumită Levenshtein optimizată, ce presupune eliminarea simbolurilor ce reprezintă spații libere. Acest lucru conduce în primul rând la o viteză sporită de execuție dar şi asigură o invarianță la perspectiva robotului conducând astfel la un proces de selecție corespunzător.

Ca şi o concluzie referitoare la aspectele analizate se poate afirma că metodele cu cele mai bune rezultate în procesarea frescelor (selecția frescelor semnificative dar şi identificare a traseului retur) sunt cele de tip Levenshtein optimizată şi metoda intercorelație dintre stringuri.

6.2 Perspective de dezvoltare a temei

Dat fiind faptul că reprezentarea simbolică a mediului în robotică reprezintă un concept în plină şi constantă evoluție, sunt de aşteptat noi abordări şi soluții la problemele aferente. O primă problemă de rezolvat ar fi implementarea unui mod evaluare a pragului de selecție on-line, pe baza eficienței navigației robotului prin mediul respectiv. Acest lucru ar creşte şi mai mult caracterul independent şi flexibil al sistemului de navigație şi orientare al robotului.

Totodată implementarea unei rețele neuronale de tip hartă de trăsături ce să opereze în mod direct cu simboluri ar putea constitui o posibilă direcție viitoare de cercetare.

Pe baza modalității de navigație bazată pe o reprezentare simbolică a mediului este posibil să fie dezvoltat un limbaj mai evoluat de comunicare între om şi robot. Din punct de vedere uman, o exprimare în limbaj natural de genul *Mergi pe coridor, apoi la a III – a uşa ia-o la stanga, intră în încapere etc.* este mult mai firească decât o exprimare metrică.

7. BIBLIOGRAFIE

[1] G. Pradel, S. Avrillon, L. Garbuio, “Landmark interpretation by means of frescoes in mobile robotics,” *Proceedings of MMAR 2000, 6th International conference on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, pp. 585-591, Aug. 28-31, 2000.

[2] G. Pradel, F. Bras, Z. Jin, “2D laser telemetry-based path trend generation and real time envirnment symbolic representation for an autonomous mobile robot,” *Proceedings of the IFAC-IEEE International Conference on Machine Automation*, Tampere, Finland, vol.1, pp. 122-134, 1994.

[3] P. Hoppenot, E. Colle, "Localisation and control of a rehabilitation robot by close human-machine co-operation," *IEEE Transaction on Neural System and Rehabilitation Engineering*, vol. 9, pp. 181-190, June 2001.

[4] P. Hoppenot, G. Pradel, C.D. Căleanu, Nicolas Perrin, Vincent Sommeilly, “Towards a symbolic representation of an indoor environment,” *SEE-CESA2003 - Computing Engineering in Systems Applications*, 9-11 Iulie, Lille, Franța, 2003.

[5] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, R. Wehner, “A mobile robot employing insect strategies for navigation,” *Robotics and Autonomous Systems*, 30, pp. 39–64, 2000.

[6] J. Borenstein, H. R. Everett, L. Feng, *Where am I? Sensors and Methods for Mobile Robot Positioning*, April 1996, University of Michigan.

[7] Y. Bock, N. Leppard, (Eds.), “Global Positioning System: An Overview,” *International Association of Geadesy Symposia*, Springer-Verlag, 1990.

[8] J. Stone, E. LeMaster, J. Powell, S. Rock, “GPS Pseudolite Transceivers and their Applications,” ION National Technical Meeting, San Diego, CA, USA, January 25-27, 1999.

[9] J. Hong, X. Tan, B. Pinette, R. Weiss, E. Riseman, “Image-based Homing,” *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, USA, pp 620-625, Apr. 1991.

[10] A. Kurz, “Constructing maps for mobile robot navigation based on ultrasonic range data,” *IEEE Transactions on Systems, Man, and Cybernetics* 26 (2) 233–242, 1996.

[11] C. Owen, U. Nehmzow, “Landmark-based navigation for a mobile robot,” From Animals to Animats - Proceedings of the SAB98, Zurich, Switzerland, MIT Press, Cambridge, MA, 1998.

[12] G. Wichert, “Selforganizing Visual Perception for Mobile Robot Navigation,” www.rt.e- technik.th-darmstadt.de/ ~georg/pubs/EUROBOTS96/paper.html.

[13] S. Al Allan, *Reconnaissance d'environnement et navigation reactive d'un robot mobile autonome par reseaux de neurones*, PhD. Thesis, Universite d'Evry Val d'Essonne, France, Feb. 1996.

[14] A. Crosnier, *Modelisation geometrique des environnements en robotique mobile*, French Workshop on Robotic Resarch (Journees Nationales de la Recherche en Robotique), Montpellier, France, pp. 83-91, Sept. 1999.

[15] P. Gaussier, C. Joulain, S. Zrehen, A. Revel, *Visual navigation in an open enviroment without map*, Proc. IEEE International Conference on Intelligent Robots systems, September 1997, pp. 237-267.

[16] Jiann-Der Lee, “Indoor Robot Navigation,” *Mathl. Comput. Modelling* Vol. 26, No. 4, pp. 79-89, 1997.

[17] J.F. Diaz, A. Stoytchev, R.C. Arkin, *Exploring Unknown Structured Enviroments*, American Association for Artificial Intelligence AAAI, 2001.

[18] Divita G, Browne A, Tse T, Cheh M, Loane R, and Abramson M. “A Spelling Suggestion Technique for Terminology Servers,” *Poster Session, AMIA Fall Symposium*, Los Angeles, 2000.

[19] Altschul S. “Amino acid substitution matrices from an information theoretic perspective.” *J Mol Biol*, 219**,** 555-65, 1991.

[20] Karlin S, Altschul SF. “Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.” *Proc Natl Acad Science***,** 87**,** 2264-68, 1990.

[21] J. Hong, X. Tan, B. Pinette, R. Weiss, E. Risemann, “Image-based Homing,” *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, USA, pp 620-625, Apr. 1991.

[22] J. Ahuactzin, E. Mazer, P. Bessière, "L'algorithme fil -d'Ariane", Revue d'intelligence artificielle, volume 9 n°1, pp. 7-34, 1995.

[23] D. Huttenlocher, "Comparing images using Hausdorff distance", *IEEE Transactions on pattern analysis and machine intelligence*, volume 15, nr.9, pp. 850-863, 1993.

[24] T.A. Lasko, S.E. Hauser, “Approximate String Matching Algorithms for Limited Vocabulary OCR Output Correction,” Proc. SPIE, Vol.4307, Documnet Recognition and Retrieval VIII, San Jose, CA, , 232-40, January 2001.

[25] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: computer science and computational biology*, Cambridge University Press, New York, 1997.

[26] L. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics–Doklady*, vol. 10, no.7, pp. 707–710, 1966.

[27] T. Kohonen, *Self-Organization and Associative Memory*, Springer Series in Information Sciences, vol. 8, Springer Berlin Heidelberg, 1988.

[28] R. A. Wagner, M. J. Fischer, *The string to string correction problem*, Journal of the ACM 21 168-173, 1974.
[29] G. A. Stepen, *String Searching Algorithms*, World Scientific, 1994.
[30] T. Kohonen, *Content-Addressable Memories*, Springer Series in Information Sciences, vol. 1, Springer Berlin Heidelberg 1987.
[31] T. Kohonen, E. Reuhkala, "A very fast associative method for the recognition and correction of misspelt words, based on redundant hash addressing," in: *Proc. 4th Int. J. Conf. on Pattern Recognition*, Kyoto, Japan, pp. 807-809, 7-10 November 1978.
[32] T. Kohonen, *Median Strings*, Pattern Recognition Letters 3, 309-313, 1985.
[33] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Second Edition, IEEE Press 1999.
[34] C.D. Căleanu, V. Tiponuţ, *Reţele neuronale. Aplicaţii*, Ed. Politehnica, 2001.
[35] V. Tiponuţ, C.D. Căleanu, *Reţele neuronale. Arhitecturi şi algoritmi*, Ed. Politehnica, Timişoara, 2002.
[36] D. W. Aha, D. Kibler, M. K. Albert, "Instance-Based Learning Algorithms," *Machine Learning*, Vol. 6, pp.37-66, 1991.
[37] C. Stanfill, D. Waltz, "Toward memory-based reasoning," *Communication of the ACM*, 29, pp. 1213-1229, 1986.
[38] D. R.Wilson, T. R. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial IntelligenceResearch*, 11, pp. 1-34, 1997.
[39] E. Blanzieri, F. Ricci, "Advanced Metrics for Class-Driven Similarity Search," *International Workshop on Similarity Search*, Firenze, Italy, September 1999.
[40] T. Kohonen, P. Somervuo, *Self-organizing maps of symbol strings*, Neurocomputing 21 pp.19-30, 1998.
[41] T. Kohonen, "Median strings," *Patt. Rec. Lett*. 3 (1985) 309.
[42] T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer, Heidelberg, 1st ed., 1995; 2nd ed., 1997.
[43] T. Kohonen, "Self-organizing maps of symbol strings," *Report A42*, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

ANEXA 1

Codul sursă MATLAB *frsel_resmbl.m* pentru selecția cu ajutorul metodei asemănării

```
function resmblCount = frsel_resmbl(inputMatrix, bias, quiet)
% *** Fresques selection based on Resemblance method.
% argument 1: matrice de 64 coloane de char
% argument 2: prag [-14:6], numere intregi

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help frsel_resmbl for instructions');
   return, end

[numlin, numcol] = size(inputMatrix); % aflare informatii despre matricea de intrare

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
  clc; end

% safe mode :)
if numcol ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);
% initializare; bucla principala
resmblCount=1;
isSelected(1)=false;              % pentru listarea frescelor selectate
for counter=2:numlin
   N0 = NumElem(buffer);
   N1 = NumElem(inputMatrix(counter,:));
   % calculeaza asemanarea prin relatia:
   resmblFactor = N0(1) - N1(1) + N0(2) - N1(2) + N0(3) - N1(3) + N0(4) - N1(4);
   % daca rezultatul este mai mare decat pragul dat (bias) se pastreaza
   % fresca in matricea SEL.
   isSelected(counter)=false;
   if resmblFactor >= bias
      SEL(resmblCount, :) = inputMatrix(counter, :);
      resmblCount=1+resmblCount;
      buffer=inputMatrix(counter,:);
      isSelected(counter)=true;
   end
end

resmblCount=resmblCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
   % afisaza rezultatele
   if resmblCount > 0
      disp('All frescoes:');% pentru listarea frescelor selectate
      for counter=1:numlin
          if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
          else disp(sprintf('%s<<', inputMatrix(counter,:))); % marcare linie sel. cu "<<"
          end
      end
      disp('Selected set:')
      SEL
      disp(sprintf('Number of selected frescoes = %d',resmblCount));
   else
      disp('No frescoes selected, use a smaller bias.')
   end
   disp(sprintf('Bias used = %d',bias));
```

```
end

% se calculeaza numarul de repere din fiecare cadran
function NumberOfElements = NumElem( input )
   NumberOfElements = [0, 0, 0, 0];
   for quadrant=1:4
      for landmark=1:16
         index=quadrant*16 + landmark -16;
         if  input(index) ~= '0'
            NumberOfElements(quadrant) = 1 + NumberOfElements(quadrant);
         end
      end
   end

% end function
```

Codul sursă MATLAB *chart_resemblance.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei asemănării

```
function r_chart = chart_resemblance(inputMatrix)
% grafic – nr. de fresce selectate functie de pragul functiei asemanare
% argument 1: matrice de 64 coloane de char

if nargin == 0 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('use chart_resemblance( 64 char wide matrix )');
   return, end

% safe mode :)
if length(inputMatrix(1,:)) ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

% -14 si 6 = valori aflate empiric
 threshold=-14:6;
max=length(threshold);
for counter=1:max   r_chart(counter)=frsel_resmbl(inputMatrix, threshold(counter), 'q');
end
% deseneaza graficul - numar de fresce selectate functie de prag (threshold, bias)
plot(threshold, r_chart)
ylabel('number of selected frescoes')
xlabel('resemblance criterion selection threshold');
```

ANEXA 2

Codul sursă MATLAB *frsel_barycenter.m* pentru selecția frescelor cu ajutorul metodei baricentrului

```
function resmblCount = frsel_barycenter(inputMatrix, bias, quiet)
% *** Fresques selection based on barycenter method.
% argument 1: matrice de 64 coloane de char
% argument 2: prag (0:2] - numere reale, ex. 0.01

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help frsel_barycenter for instructions');
   return, end

[numlin, numcol] = size(inputMatrix); % aflare informatii despre matricea de intrare

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
  clc; end

% safe mode :)
if numcol ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);
% initializare; bucla principala
resmblCount=1;
 isSelected(1)=false;                % pentru listarea frescelor selectate
for counter=2:numlin
   N0 = NumElem(buffer);
   N1 = NumElem(inputMatrix(counter,:));
   N_ref=sum(N0);
   N_c=sum(N1);
   % calculeaza baricentrul:
   x_ref=(N0(1)-N0(3))/N_ref;
   y_ref=(N0(2)-N0(4))/N_ref;
   x=(N1(1)-N1(3))/N_c;
   y=(N1(2)-N1(4))/N_c;
   barycenter = sqrt( (x_ref - x)^2 - (y_ref-y)^2 );
   % daca rezultatul este mai mare decat pragul dat (bias) se pastreaza
   % fresca in matricea SEL.
    isSelected(counter)=false;
   if barycenter >= bias
      SEL(resmblCount, :) = inputMatrix(counter, :);
      isSelected(counter)=true; % pentru listarea frescelor selectate
      resmblCount=1+resmblCount;
      buffer=inputMatrix(counter,:);
   end
end

resmblCount=resmblCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
   % afisaza rezultatele
   if resmblCount > 0
      disp('All frescoes:');% pentru listarea frescelor selectate
      for counter=1:numlin
          if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
          else disp(sprintf('%s<<', inputMatrix(counter,:)));
                    % marcare linie sel. cu "<<"
          end
      end
```

```
        disp('Selected set:')
        SEL
        disp(sprintf('Number of selected frescoes = %d',resmblCount));
    else
        disp('No frescoes selected, use a smaller bias.')
    end
    disp(sprintf('Bias used = %d',bias));
end

% se calculeaza numarul de repere din fiecare cadran
function NumberOfElements = NumElem( input )
    NumberOfElements = [0, 0, 0, 0];
    for quadrant=1:4
        for landmark=1:16
            index=quadrant*16 + landmark -16;
            if  input(index) ~= '0'
                NumberOfElements(quadrant) = 1 + NumberOfElements(quadrant);
            end
        end
    end

% end function
```

Codul sursă MATLAB *chart_barycenter.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei baricentrului

```
function bc_chart = chart_barycenter(inputMatrix, step);
% *** grafic - numarul de fresce selectate functie de pragul functiei baricentru
% argument 1: matrice de 64 coloane de char
% argument 2: pas pozitiv, real – ex. 0.01

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
    disp('Argument error: type help chart_barycenter for instructions');
    return, end

% aflare informatii despre matricea de intrare
[numlin, numcol] = size(inputMatrix);

% safe mode :)
if numcol ~= 64
    disp('ERROR: input matrix has to be 64 chars wide');
    return, end
if step <= 0
    disp('ERROR: please enter a positive step - example 0.1 ');
    return
elseif step > 2
    disp('ERROR: please enter a smaller step - example 0.1 ');
    return, end

% initializare variabile
counter=1;
threshold(counter)=0;
increment=0;
bc_chart(counter)=frsel_barycenter(inputMatrix, 0, 'q');

% bucla principala
% determina numarul de fresce selectate pt fiecare prag (increment)
while bc_chart(counter) > 0
    counter=1+counter;
    increment=increment+step;
    threshold(counter)=increment;
```

```
    bc_chart(counter)=frsel_barycenter(inputMatrix, increment, 'q');
end

% deseneaza graficul - numar de fresce selectate functie de prag (threshold, bias)
plot(threshold, bc_chart)
ylabel('number of selected frescoes')
xlabel('barycenter criterion selection threshold');
```

ANEXA 3

Codul sursă MATLAB *frsel_hamming.m* pentru selecția cu ajutorul metodei distanței Hamming

```
function resmblCount = frsel_hamming(inputMatrix, bias, quiet)
% *** Fresques selection based on Hamming distance method.
% argument 1: matrice de 64 coloane de char
% argument 2: prag =  procent, reprezinta diferenta minima
%             intre doua fresce

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
  disp('Argument error: type help frsel_hamming for instructions');
  return, end
% aflare informatii despre matricea de intrare
[numlin, numcol] = size(inputMatrix);

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
   clc; end

% safe mode :)
if numcol ~= 64
    disp('ERROR: input matrix has to be 64 chars wide');
    return, end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);
% initializare; bucla principala
resmblCount=1;
isSelected(1)=false;                % pentru listarea frescelor selectate
for counter=2:numlin
    Hd=0;    % reprezinta distanta Hamming intre doua siruri - initial 0
    for index=1:numcol
        if buffer(index) ~= inputMatrix (counter, index) Hd=Hd+1;
        end
    end
   mismatch=100*Hd/numcol;  % calculez valoarea procentuala
    % daca rezultatul este mai mare decat pragul dat (bias) se pastreaza
    % fresca in matricea SEL.
    isSelected(counter)=false;
    if mismatch >= bias
        SEL(resmblCount, :) = inputMatrix(counter, :);
        resmblCount=1+resmblCount;
        buffer=inputMatrix(counter,:);
        isSelected(counter)=true;
    end
end

resmblCount=resmblCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
    % afisaza rezultatele
    if resmblCount > 0
        disp('All frescoes:');   % pentru listarea frescelor selectate
        for counter=1:numlin
            if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
            else disp(sprintf('%s<<', inputMatrix(counter,:))); % marcare linie sel. cu "<<"
            end
        end
        disp('Selected set:')
        SEL
        disp(sprintf('Number of selected frescoes = %d',resmblCount));
    else
```

```
        disp('No frescoes selected, use a smaller bias.')
    end
    disp(sprintf('Bias used minumum %d%s difference between frescoes',bias, '%'));
end
```

Codul sursă MATLAB *chart_hamming.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei distanței Hamming

```
function bias_string = chart_hamming(frescoes)
% *** grafic – nr. de fresce selectate functie de pragul dist. H.
% argument 1: matrice de 64 coloane de char

if nargin == 0 % daca nu exista argumente, avertizeaza utilizatorul.
    disp('Argument error: type help chart_hamming for instructions');
    return, end

% aflare informatii despre matricea de intrare
[numlin, numcol] = size(frescoes);

% safe mode :)
if numcol ~= 64
    disp('ERROR: input matrix has to be 64 chars wide');
    return, end

min_bias=20;
max_bias=70;
k=1;
for n = min_bias : max_bias
    biasrange(k) = frsel_hamming(frescoes, n, 'q');
    k=1+k;
end;
plot(min_bias:max_bias, biasrange)
xlabel('threshold - difference between frescoes [%]')
ylabel('selected frescoes');

bias_string = biasrange;
```

ANEXA 4

Codul sursă MATLAB *frsel_editdist.m* pentru selecția cu ajutorul metodei distanței Levenshtein

```
function response = frsel_editdist(inputMatrix,bias,quiet)
% *** Fresques selection based on Levensthein distance
% argument 1: matrice de 64 coloane de char
% argument 2: prag =  numere naturale pozitive, ex. 23

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help frsel_editdist for instructions');
   return, end

% se stabileste numarul de linii si coloane - numlin respectiv nrcol
[numlin, numcol] = size(inputMatrix);

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
  clc; end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);

resmblCount=1;
isSelected(1)=false;              % pentru listarea frescelor selectate
for counter=2:numlin
   % se afla distanta Levensthein
   Lev_dist = editdist(buffer,inputMatrix(counter,:));
   isSelected(counter)=false;
   % daca distanta L este mai mare decat pragul dat (bias) se pastreaza
   % fresca in matricea SEL.
   if Lev_dist > bias
      buffer=inputMatrix(counter,:);
      % ??? response{resmblCount} = {counter, inputMatrix(counter,:)};
      SEL(resmblCount, :) = inputMatrix(counter, :);
      isSelected(counter)=true;
      resmblCount=resmblCount+1;
   end
end

resmblCount=resmblCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
   % afisaza rezultatele
   if resmblCount > 0
      disp('All frescoes:');% pentru listarea frescelor selectate
      for counter=1:numlin
         if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
         else disp(sprintf('%s<<', inputMatrix(counter,:))); % marcare linie sel. cu "<<"
         end
      end
      disp('Selected set:')
      SEL
      disp(sprintf('Number of selected frescoes = %d',resmblCount));
   else
      disp('No frescoes selected, use a smaller bias.')
   end
   disp(sprintf('Bias used = %d',bias));
end
response=resmblCount;
```

Codul sursă MATLAB *editdist.m* care implementează algoritmul distanței Levenshtein

```
function d = EditDist(s1,s2,varargin)

%Determine the number of inputs. If 2 inputs, set default edit costs to 1.
%Otherwise, make sure there are exactly 5 inputs, and set edit costs
%accordingly.
if ~isempty(varargin)
  if length(varargin) ~= 3
    error('Usage is: EditDist(''string1'',''string2'',DeleteCost,InsertCost,ReplaceCost)');
  end;
  DelCost = varargin{1};
  InsCost = varargin{2};
  ReplCost = varargin{3};
else
  DelCost = 1;
  InsCost = 1;
  ReplCost = 1;
end;

[m1,n1] = size(s1);
[m2,n2] = size(s2);

%Make sure input strings are horizontal.
if ~(ischar(s1) & ischar(s2) & m1 == 1 & m2 == 1)
  error('s1 and s2 must be horizontal strings.');
end;

%Initialize dynamic matrix D with appropriate size:
D = zeros(n1+1,n2+1);


%This is dynamic programming algorithm:
for i = 1:n1
  D(i+1,1) = D(i,1) + DelCost;
end;

for j = 1:n2
  D(1,j+1) = D(1,j) + InsCost;
end;

for i = 1:n1
  for j = 1:n2
    if s1(i) == s2(j)
      Repl = 0;
    else
      Repl = ReplCost;
    end;
    D(i+1,j+1) = min([D(i,j)+Repl D(i+1,j)+DelCost D(i,j+1)+InsCost]);
  end;
end;

d = D(n1+1,n2+1);
```

Codul sursă MATLAB *chart_editdist.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei distanței Levenshtein

```
function bias_string = chart_editdist(frescoes)
% *** grafic - numarul de fresce selectate functie de pragul dist. L.
% argument 1: matrice de 64 coloane de char
```

```
if nargin == 0 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help chart_editdist for instructions');
   return, end

% aflare informatii despre matricea de intrare
[numlin, numcol] = size(frescoes);

% safe mode :)
if numcol ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

min_bias=5;
max_bias=35;
k=1;
for n = min_bias : max_bias
   biasrange(k) = frsel_editdist(frescoes, n, 'q');
   k=1+k;
end;
plot(min_bias:max_bias, biasrange)
xlabel('threshold')
ylabel('selected frescoes');

bias_string = biasrange;
```

Codul sursă MATLAB *frsel_editdistfast.m* pentru selecția cu ajutorul metodei distanței Levenshtein, cu optimizare

```
function resmblCount = frsel_editdistfast(inputMatrix, bias, quiet)
% *** Fresques selection based on Levensthein distance
% argument 1: matrice de 64 coloane de char
% argument 2: prag =  numere naturale pozitive, ex. 10
% OBS. Acest program opereaza cu fresce optimizate, fiind mai rapid

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help frsel_editdistfast for instructions');
   return, end

% aflare informatii despre matricea de intrare
[numlin, numcol] = size(inputMatrix);

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
  clc; end

% safe mode :)
if numcol ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);
String1 = stripzeros(buffer);    % scoatem zerourile din sir
% initializare; bucla principala
resmblCount=1;
isSelected(1)=false;                % pentru listarea frescelor selectate
for counter=2:numlin
   String2=stripzeros(inputMatrix(counter,:));  % scoatem zerourile din sir

   Lev_dist = editdist(String1,String2);
   isSelected(counter)=false;
   % daca distanta L este mai mare decat pragul dat (bias) se pastreaza
```

```
    % fresca in matricea SEL.
    if Lev_dist > bias
        buffer=inputMatrix(counter,:);
        String1 = stripzeros(buffer);
        SEL(resmblCount, :) = inputMatrix(counter, :);
        % disp(sprintf('#%d: %s', resmblCount, String1));
        isSelected(counter)=true;
        resmblCount=resmblCount+1;
    end
end

resmblCount=resmblCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
    % afisarea rezultatelor
    if resmblCount > 0
        disp(' ');
        disp('All frescoes:');% pentru listarea frescelor selectate
        for counter=1:numlin
            if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
            else disp(sprintf('%s<<', inputMatrix(counter,:))); % marcare linie sel. cu "<<"
            end
        end
        disp('Selected set:')
        SEL
        disp(sprintf('Number of selected frescoes = %d',resmblCount));
    else
        disp('No frescoes selected, use a smaller bias.')
    end
    disp(sprintf('Bias used = %d',bias));
end

% scoate zerourile din sir
function zeroless_string = stripzeros(inputString)
    landmarkCount=0;
    for index=1:length(inputString)
        if inputString(index) ~= '0'
            landmarkCount=landmarkCount+1;
            zeroless_string(landmarkCount)=inputString(index);
        end
    end
 % end function
```

Codul sursă MATLAB *chart_editdistfast.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei distanței Levenshtein optimizate:

```
function bias_string = chart_editdistfast(frescoes)
% *** grafic - numarul de fresce selectate functie de pragul dist.  L.
% *** autor Gheorghe Sandor (2004)
% *** OBS. se foloseste acelasi algoritm, dar cu fresce optimizate
% argument 1: matrice de 64 coloane de char

if nargin == 0 % daca nu exista argumente, avertizeaza utilizatorul.
    disp('Argument error: type help chart_editdist for instructions');
    return, end
% aflare informatii despre matricea de intrare
[numlin, numcol] = size(frescoes);
% safe mode :)
if numcol ~= 64
    disp('ERROR: input matrix has to be 64 chars wide');
    return, end
min_bias=0;
max_bias=20;
k=1;
```

```
for n = min_bias : max_bias
    biasrange(k) = frsel_editdistfast(frescoes, n, 'q');
    k=1+k;
end;
plot(min_bias:max_bias, biasrange)
xlabel('threshold')
ylabel('selected frescoes');
bias_string = biasrange
```

ANEXA 5

Codul sursă MATLAB *frsel_corr.m* pentru selecția cu ajutorul metodei corelației dintre stringuri

```
function response = frsel_corr(inputMatrix,bias,quiet)
% *** Fresques selection based on correlation
% argument 1: matrice de 64 coloane de char
% argument 2: prag =  numere naturale pozitive, ex. 35

if nargin < 2 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help frsel_editdist for instructions');
   return, end

% se stabileste numarul de linii si coloane - numlin respectiv nrcol
[numlin, numcol] = size(inputMatrix);

if nargin ~= 3 % daca exista argumentul quiet, nu afecta fereastra de com.
  clc; end

% se introduce prima fresca intr-o variabila temporara buffer
buffer=inputMatrix(1,:);

corrCount=1;
isSelected(1)=false;                % pentru listarea frescelor selectate
for counter=2:numlin
   % se afla distanta Levensthein
   corr_dist = max(xcorrtxt(buffer,inputMatrix(counter,:)));
   isSelected(counter)=false;

   % daca distanta X este mai mare decat pragul dat (bias) se pastreaza
   % fresca in matricea SEL.
   if corr_dist < bias
      buffer=inputMatrix(counter,:);
      SEL(corrCount, :) = inputMatrix(counter, :);
      isSelected(counter)=true;
      corrCount=corrCount+1;
   end
end

corrCount=corrCount-1; % corectie rezultat
if nargin ~= 3 % daca nu exista argumentul quiet atunci
   % afisaza rezultatele
   if corrCount > 0
      disp('All frescoes:');% pentru listarea frescelor selectate
      for counter=1:numlin
         if isSelected(counter)==false disp(sprintf('%s', inputMatrix(counter,:)));
         else disp(sprintf('%s<<', inputMatrix(counter,:))); % marcare linie sel. cu "<<"
         end
      end
      disp('Selected set:')
      SEL
      disp(sprintf('Number of selected frescoes = %d',corrCount));
   else
      disp('No frescoes selected, use a bigger bias.')
   end
   disp(sprintf('Bias used = %d',bias));
end
response=corrCount;
```

```
function [out, matched_str] = xcorrtxt(a, b)
%XCORRTXT Cross correlation of two text strings
%	Usage:
%	out = xcorrtxt(a, b)
%	a: input string 1
%	b: input string 2
%	count: cross correlation (a vector of length length(a)+length(b)-1)

a = a(:).';
b = b(:).';
m = length(a);
n = length(b);
A = a(ones(1,n), :).';
B = b(ones(1,m), :);
matched = A==B;
C = [zeros(m,m-1) matched zeros(m,m-1)];
out = zeros(1, n+m-1);
for i = 1:n+m-1,
	out(i) = sum(C((1:m)+((i:i+m-1)-1)*m));
end

if nargout == 2,
	for i = 1:n+m-1,
		tmp = C((1:m)+((i:i+m-1)-1)*m);
		index = find(tmp==1);
		matched_str{i} = a(index);
	end
end
```

Codul sursă MATLAB *chart_corr.m* pentru trasarea caracteristicii prag – număr de fresce păstrate, prin selecția cu ajutorul metodei corelației dintre stringuri

```
function bias_string = chart_corr(frescoes)
% *** grafic - numarul de fresce selectate functie de pragul corelatiei
% *** chart_corr(inputMatrix);  inputMatrix = 64xN (char)

if nargin == 0 % daca nu exista argumente, avertizeaza utilizatorul.
   disp('Argument error: type help chart_editdist for instructions');
   return, end

% aflare informatii despre matricea de intrare
[numlin, numcol] = size(frescoes);

% safe mode :)
if numcol ~= 64
   disp('ERROR: input matrix has to be 64 chars wide');
   return, end

min_bias=25;
max_bias=50;
k=1;
for n = min_bias : max_bias
   biasrange(k) = frsel_corr(frescoes, n, 'q');
   k=1+k;
end;
plot(min_bias:max_bias, biasrange)
```

```
xlabel('threshold')
ylabel('selected frescoes');
title('Frescoe selection chart using correlation method')
bias_string = biasrange;
```

ANEXA 6

Codul sursă MATLAB *frsel_clasicSOM.m* pentru selecția cu ajutorul metodei bazate pe arhitectura RNA-SOM clasică

```
clear all
close all
clc

load laliste.mat

[linb colb]=size(fresca);
c=charmat2uvect(fresca,'direct');
[linc colc]=size(c);
d=double(c)-48*ones(linc,colc);

l2=0;l1=1;l3=cat(2,l2,l1);
l4=l3;
for i = 1:(linc-1)
l4 = cat(1,l4,l3);
end

nrprot=7;
net = newsom(l4,nrprot);

%afisarea distributiei initiale a vectorilor pondere
plotsom(net.layers{1}.positions)
figure

% antrenamentul retelei
net.trainParam.epochs=500;
net = train(net,d);

plotsom(net.iw{1,1},net.layers{1}.distances)
prf=net.iw{1};

pr1 = round(prf);
pr2 = pr1';
pr3 = dec2base(pr2,2);
binary_prototypes = (reshape(pr3,4,nrprot*colb))'
pr5 = dec2hex(bin2dec(binary_prototypes));
hex_prototypes = (reshape(pr5,colb,nrprot))'



function y = charmat2uvect(x,type)
% Convert a matrix of chars to unary vectors by means of a)direct coding b)excusive coding
x
bin=x';
[nrlin, nrcol] = size(x);
switch type
   case 'direct';
      bin1=dec2bin(hex2dec(bin(:)));
      yr=bin1';
      %yr=reshape(rr(:),4,nrlin*nrcol);
      y=reshape(yr(:),4*nrcol,nrlin);
   case 'exclusive';
      disp('Not implemented yet!')
```

```
end
sprintf('Input matrix %s coded: %d',type)
```