MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

URMĂRIREA UNUI OBIECT PRIN FUZIUNE SENZORIALĂ

LUCRARE DE LICENŢĂ

Absolvent: **Andrei MORARU**
Conducător ştiinţific: **Prof. dr. ing. Eva H. DULF**

2022

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Honoriu VĂLEAN

Absolvent: **Andrei MORARU**

URMĂRIREA UNUI OBIECT PRIN FUZIUNE SENZORIALĂ

1. **Enunţul temei:** *Diverse metode de estimare a traiectoriei prin fuziune senzorială în vederea realizării unei urmăriri autonome a unui obicect în planul bidimensional*
2. **Conţinutul lucrării:** *Cuvânt-înainte, Obiective, Studiu Bibliografic, Analiză şi Fundamentare Teoretică, Proiectare de Detaliu şi Implementare, Testare şi Validare, Manual de Instalare şi Utilizare, Concluzii, Anexele A,B,C*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, Departamentul Automatică*
4. **Consultanţi: Prof. dr. ing. Zsófia LENDEK**
5. **Data emiterii temei:** 1 Octombrie 2021
6. **Data predării:** 1 iulie 2022

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul *Moraru Andrei*, legitimat cu *cartea de identitate* seria *MH* nr. *550346* CNP *1990920055053*, autorul lucrării *URMĂRIREA UNUI OBIECT PRIN FUZIUNE SENZORIALĂ* elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea *Ingineria Sistemelor Automate* din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea *iulie* a anului universitar *2022*, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data	Nume, Prenume
01.07.2022	*Moraru Andrei*

Semnătura

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ

SINTEZA
proiectului cu titlul:
URMĂRIREA UNUI OBIECT PRIN FUZIUNE SENZORIALĂ

1. **Cerinţele temei:** *Proiectarea unei aplicaţii care să simuleze o cursă interactivă cu scopul de a pune în practică diverşi algoritmi de estimare, reprezentări ale orientării în spaţiu şi legi de control*

2. **Soluţii alese:** *Filtre Kalman - liniar, extins şi unscented, cuaternioni, control liniar*

3. **Rezultate obţinute:** *Convengenţa garantată pentru sisteme liniare, soluţii funcţionale pentru modele neliniare, îmbunătăţirea timpului de răspuns comparativ cu implementări alternative*

4. **Testări şi verificări**: *Comparaţia cu soluţii deja existente, validări grafice, validări numerice prin minimizarea erorilor*

5. **Contribuţii personale:** *Implementarea unei aplicaţii complexe bazată pe programare orientată pe obiecte ce se prezintă ca o abordare diferită a fundamentului teoretic, realizarea unei soluţii alternative de estimare a orientării pentru o funtionalitate deja existentă, dezvoltarea interacţiunilor şi animaţiilor din joc, conceperea interfeţei grafice, realizarea conexiunii cu o placă de dezvoltare Arduino Nano într-o configuraţie personală de tip joystick*

6. **Surse de documentare**: *Publicaţii ştiinţifice, articole şi site-uri de specialitate, surse video, consultaţii în timpul şi în afara orelor de curs cu profesorii*

Cuprins

Capitolul 1

Cuvânt-înainte

1.1 Despre ştiinţă şi zgomot

"Atunci când nu ştii ce faci, toate calculele tale sunt zgomot" este dictonul propriu enunţat de Rudolf Emil Kalman într-un colocviu susţinut la Institutul de tehnologie din Massachusetts în 1991 [1], ca o concluzie personală cu privire la aleatoritatea regăsită în fenomenele lumii reale.

Privită în contextul teoriei sistemelor, maxima îsi redobândeşte terenul cedat în favoarea sentenţiozităţii, pentru că spusele autorului îşi regăsesc efectul în lucrări istorice care se bucură şi astăzi de o generală aplicabilitate, lucrări care stau şi la baza prezentei teze de diplomă.

Deşi nu singurul canevas de suport teoretic pentru actuala lucrare, stocasticitatea sistemelor reale este cu siguranţă una din lentilele prin care poate fi analizat acest studiu, dar în acelaşi timp şi un instrument utlil pentru a trage concluzii pertinente cu privire la rezultatele obţinute; şi tocmai de aceea am ales să îi ofer însemnătatea cuvenită în contextul problematicei fuziunii senzoriale pentru sistemele neliniare.

Doresc, pe această cale, să introduc subiectul lucrării mele de licenţă printr-o sinteză a contextului actual ce a iscat interesul personal pentru algoritmii regăsiţi pe ramura statistic-probabilistică a teorei sistemelor, teorie ce reprezintă fundamentul modelelor matematice din spatele funcţionării senzorilor inerţiali.

Am ales să dezvolt acest proiect în jurul unei aplicaţii de urmărire în timp real a unui obiect, deoarece consider că acesta este unul dintre scenarile prin care se poate observa cel mai clar necesitatea îmbinării tututror resurselor senzoriale posibile în vederea obţinerii unui peisaj cât mai clar despre cadrul imediat înconjurător în care se desfăşoară mişcarea. Din acest motiv am decis să proiectez şi o aplicaţie interactivă prin intermediul căreia punerea în practică a noţiunilor teoretice se realizează print-un cadru aparent ingenuu, dar cu implicare directă şi consolidat de o analiză minuţioasă în spate.

[1] https://www.youtube.com/watch?v=i5kTdHT3yBg

1.2 Senzorii inerţiali

Motivul central al acestei lucrări este, fără îndoială, după cum este evocat şi în titlu, reprezentat de senzori, în deosebi de cei inerţiali. Aceştia pot fi priviţi cu interes din mai multe puncte de vedere, în special pentru importanţa din prezent acordată înglobării lor în cât mai multe dispozitive, dar şi pentru evoluţia lor electronică din ultimele decenii. În lucrarea propusă însă, abordarea pe care doresc să o aprofundez se referă la modelele matematice din spatele informaţiei obţinute de la senzori, mai mult decât la senzori în sine, deşi avantajele şi limitările lor vor fi de asemenea amintite.

Senzorii inerţiali sunt o parte vitală a procesului de realizare a sistemelor autonome şi fac parte din ciclul de proceduri SPPA (Sense, Percieve, Plan, Act) în procesele de control ce adresează sectorul roboticii sau al vehiculelor inteligente. Tocmai de aceea am decis să conturez contextul acestui proiect ca unul specific industriei automobilistice, particularizat pentru procedurile de fuziune senzorială. Dedic astfel următoarele capitole unei analize riguroase a prezentelor tehnici şi metodologii folosite actual în domeniu şi ofer şi o implementare bazată pe cunoştiinţele acumulate menite să servească ca schelet pentru ulterioarele proiecte ce vor urma.

1.3 Aprecieri

Prezenta lucrare nu ar fi fost completă fără îndumarea oferită de prof. dr. ing Eva Henrieta Dulf, conducătoarea ştiinţifică a tezei mele de diplomă şi de aceea doresc în acest capitol introductiv să ii mulţumesc pentru îndemnurile oferite şi constantele încurajări.

Tot într-atât de recunoscător ii sunt şi doamnei profesoare dr.ing. Zsófia Lendek pentru ajutorul constant şi implicarea directă acordate în timpul ultimul semestru pentru implementarea algoritmilor de estimare folosiţi în aplicaţia finală, cel mai avansat regăsindu-se într-una din anexele acestui document.

Sunt recunoscător de asemenea şi pentru îndrumarea oferită de MSc. ing. Daniel D. Timiş în cadrul unui proiect care a devenit stadiul incipient al acestei lucrări de licenţă în urmă cu un an universitar.

Capitolul 2

Obiectivele Proiectului

Definită ca schimbarea poziției în raport cu timpul, mișcarea poate fi modelată cu un set de două ecuaţii diferenţiale. Indiferent de tipul și complexitatea traiectoriei pe care o definește corpul în mișcare, și fară a ţine seama de masa ori de volumul corpului imaginat, locul geometric definit de proiecția poziției obiectului animat pe axele unui plan sau ale unui spatiu se supune dinamicii definită întotdeauna de aceleaşi două ecuații specifice. Aceste legi bine definite inițial de Galileo Galilei, și concretizate mai apoi de Isaac Newton, fac posibila descrierea mișcării în univers în general, și a corpurilor de pe pământ în particular. Comportamentul ce se împotrivește inerției poate fi, cu ajutorul soluțiilor ecuațiilor diferențiale, modelat ca un sistem dinamic al cărei componentă cinematică este dată întocmai de acțiunea mișcării. Un astfel de sistem poate fi mai departe supus algoritmilor de estimare și legilor de control formulate de teoria sistemelor în funcţie de problematica domeniului pentru care este modelat.

Am dezvoltat acest proiect cu scopul de a testa și valida diferiți algoritmi de estimare a traiectoriei unui obiect urmărit și legi de control care definesc acțiunile unui obiect urmăritor. O atenție deosebită a fost acordată metodelor de fuziune a măsurătorilor senzorilor și de exprimare a mișcărilor rotative în spaţiul tridimensional. În acest sens, simularea urmăririi unei traiectorii este supleată de conectarea unui microcontroller Arduino care, prin intermediul unui modul extern de senzori inerțiali, permite utilizatorului să simuleze mișcarea obiectului urmărit, întocmai ca într-un joc ce se desfășoară în planul bidimensional al monitorului.

Proiectul se împarte așadar în două parti principale cu importanţă de sine stătătoare, urmărirea traiectoriei simulate pentru obiectul în mișcare și fuziunea datelor de la senzor pentru generarea mișcării. Atât algoritmii de urmărire, cât și cei de fuziune, cu impact individual demn de menționat, sunt înrudiți prin teoria ce stă la baza lor, și formează împreună, în esență, o reprezentație mai fidela a realității care poate fi explorată mai departe în domenii precum conducerea autonomă, realitatea virtuală ori augmentată, precum și viitoarele lor posibile ramificații.

2.1 Estimarea traiectoriei

Pentru a realiza detecția poziției unui corp și pentru a iniția urmărirea acestuia, toți algoritmii folosiți până în prezent au că fundament în alcătuiră lor statistica descriptivă. Fenomenul mișcării, precum și cel al detecției acesteia prin folosirea senzorilor inerțiali, sunt măsurabile în timp pentru un grup de eșantioane de date, ceea ce face posibile concluzile legate de tendința centrala şi dispersia grupului de măsurători folosite, ce duc mai departe la rezultate validate prin acuratețe sau precizie, concepte ce pot fi concret definite prin termeni statistici. Pentru a studia dacă modelul propus menit să descrie fenomenul este unul general, ce se dovedeşte fezabil și pentru aplicații de dimensiuni mai mari, statistica inferentială este de această dată ramura relevantă a științei probabilistice.

Procedeele care se bazează pe noțiuni statistice au evoluat până în prezent pentru a fi folosite în combinație cu o mare gamă de științe precum teoria controlului, informatica aplicată, dar și bilogia, psihologia ori științele umane, cu scopul de a găsi răspunsuri la întrebările din spatele unor fenomene naturale. Statistica aplicată culminează cu apariția metodelor de învățare automata, numita și inteligenta artificială, un obiect de studiu cu orizonturi nenumărate.

Algoritmii clasici de estimare, elocvenți teoriei sistemelor, au, faţă de relativ noile metode de învățare automata, avantajul că funcționarea lor este garantata din prima executare, fară a necesita o perioadă de timp de antrenare, și fară a face greșeli, cu condiția fermă de a avea un model al procesului bine definit pentru sistemul dinamic aferent. Parametrii sistemului configurați pot fi estimați în timpul rulării, ceea ce face aceste proceduri fiabile întocmai pentru aplicații de timp real. Ca urmare, desi robuști prin informația codificată în implementarea lor, dezavantajul lor este că o convergenţă generală nu poate fi asigurată pentru orice scenariu. După cum se va observa în capitolul 5, estimarea traiectoriilor complexe poate să nu fie exacta, și procedura ar putea fi îmbunătățită de metode avansate bazate pe învățare și antrenare în timp.

Sistemul simplificat ce descrie mișcarea unui corp va avea la baza mereu acelăsi set de ecuații diferențiale, indiferent daca mișcarea este a unei drone, a unui vehicul autonom sau a unui obiect dintr-o simulare, ceea ce face posibila extragerea unor concluzii statistice despre traiectoria mișcării și implicit și aplicarea unor algoritmi clasici.

Dat fiind contextul, lucrarea explorează și compară în detaliu trei principali algoritmi statistici pentru estimarea parametrilor sistemului dinamic (soluțiile ecuațiilor diferențiale), folosiți în industria navigației cu acest scop încă de la introducerea lor de către autorul al cărui nume îl și poartă:

- Filtrul Kalman liniar
- Filtrul Kalman extins
- Filtrul Kalman unscented

Folosiți pentru mult timp ca standard în urmărirea traiectoriilor, fie în industrii precum cea defensivă, ori debutanta industrie a condusului ori zborului autonom, variațiile algoritmului inițial dezvoltat de Rudolf E. Kalman se prezintă ca o mixtura de teorie a controlului, procesare de semnal și statistică descriptivă. Mai concret, plecând de la setul de ecuații de mișcare ce a fost definit, numit model matematic, și procesând măsurători din exterior cu un anumit grad de încredere calculat tot în interiorul procedurii, comportarea algoritmului se desfășoară ca o estimare optimă de timp real a parametrilor modelului, și ea poate fi adaptata pentru orice tip de senzor, nu neapărat inerțial.

2.2 Fuziunea senzorială

Pe lângă modelul procesului intern al algoritmilor de urmărire, aceștia primesc și informații despre obiect sub forma măsurătorilor de la senzori. Senzorii folosiți pentru a intercepta mișcarea unui corp se numesc senzori inerțiali sau IMU (Inertial Measurement Unit). Măsurătorile senzorilor în sine sunt un pas important pentru corectarea erorilor, dar convergenţa algoritmilor este mai departe întreţinută de procesul de fuziune al datelor lor.

Un singur senzor, ce descrie un anumit tip de mișcare, prezintă în genere zgomot alb, după cum urmează a fi demonstrat în capitolul 4, și o serie de avantaje și dezavantaje de funcționare în anumite scenarii, comparativ cu alţi senzori ce pot descrie aceeași mișcare. Adăugarea unui senzor identic poate îmbunătăți performanţele, dar problema principală nu este soluţionată. Erorile celor de acum doi sau mai mulți senzori identici vor fi corelate, și dezavantajele vor apărea concomitent în aceleași situații. Soluția de vărf pentru imbunatatinea acurateței măsurătorilor este folosirea diferitor senzori pentru a descrie aceeași mișcare. În acest fel, erorile vor fi prezente în ambele serii de măsurători, dar în situatile în care un senzor nu vă descrie bine mișcarea, algoritmul se poate baza pe alt senzor. Acest concept se numește fuziune senzoriala și, fiind vorba despre un procedeu statistic de acordare a încrederii, o versiune de filtru Kalman a fost programată și pentru a oferi o soluție în acest sens.

Împreună cu o placă de dezvoltare Arduino Nano, a fost folosit un senzor MPU-6050 care conține un accelerometru și un giroscop triaxiale, ce însumează împreună ase grade de libertate (6 DoF - Degrees of Freedom), iar avantajele i dezavantajele fiecărui senzor în parte, precum și îmbunătățirea considerabilă provenită din fuzionarea măsurătorilor vor fi descrise pe larg în capitolele 4 şi 5 .

În urmărirea propriu-zisă, valorile ce descriu poziția obiectului detectat în fiecare moment de timp sunt modelate ca o serie de măsurători cu zgomot puternic, dar Gaussian, asemenea măsurătorilor de GPS (Global Positioning System). După cum va fi demonstrat, adăugarea în simulare a unor măsurători ale unui senzor de tip sonar ce transmit informații despre coordonatele polare ale obiectului urmărit, îmbunătățesc performanţa urmăririi atunci când sunt fuzionate cu cele de tip GPS. Aceastambunatățire semnificativă este şi

fundamentul, dar și motivația procedurii de fuziune a senzorilor.

Dacă implementarea măsurătorilor de tip GPS și sonar sunt simulate, pentru accelerometrul și girocopul prezente pe un hardware extern îmbunătățirea devine și mai clara, iar rezultatele validează ipoteza necesității unei astfel de proceduri.

2.3 Dezvoltarea jocului

În sprijinul validării vizuale și concretizării noțiunilor teoretice ce stau la baza primelor doua subcapitole menționate anterior, o aplicație grafică în două dimensiuni (2D) de tip joc a fost dezvoltată în mediul de programare MATLAB, cu limbajul său propriu.

Contextul urmăririi capătă acum un aspect ludic: corpul urmărit, cel care poate fi controlat de jucător, este o pinipedă din cateogia Phocidaelor, iar urmăritorul este un rechin, animal din ordinul Selachimorpha. Scopul rechinului este de a prinde mamiferul acvatic folosindu-se de diverși algoritmi de estimare și legi de control în funcție de traseul parcurs, și, de fiecare dată când se apropie suficient de mult (în alte cuvinte, atunci când stările estimate converg la cele reale și eroarea este minimizată) acesta își ataca prada, scazandu-i din punctele finite de viață.

Scopul focii, și implicit cel al jucătorului este de a genera traiectorii cât mai complexe, pentru a rămâne cât mai mult timp în viață. Scenariul acvatic se diversifică atunci când în peisaj apare o pasăre acvatica, reprezentată de un pescăruș răpitor, care, oferindu-i ajutor rechinului, îl ghidează prin măsurători mai precise ale coordonatelor focii de la fiecare moment de timp. Pescărușul este astfel modelat ca un senzor de tip sonar, care are în vedere unghiul și raza sub care se mișcă victima, denumite și coordonate polare. Modelul urmăritorului beneficiază de aceste noi măsurători și printr-un proces de corecție își îmbunătățește abilitatea de a luat decizii.

Atunci când aplicaţia nu este rulată în modul de joc, ci sub formă de simulare, utilizatorul poate să aleagă din inferfaţa grafica dezvoltată combinații de trasee și algoritmi de estimare, iar animația devine în esență un mediu de testare și validare al estimatoarelor, ce poate fi folosit mai departe ca suport pentru dezvoltarea unor proiecte sau aplicaţii mai avansate.

Diferenţa dintre rularea aplicaţiei în modul joc sau simulare este la latitudinea utilizatorului. Dacă sunt disponibile un senzor de tip MPU-6050 şi placa de dezvoltare Arduino, din interfaţa grafică se va putea configura opţiunea dorită. Ansamblul electronic e programat în aşa fel încât să funcţioneze ca o consolă de tip joystick, un periferic asociat în general jocurilor video. Scopul folosirii senzorului constă în obţinerea informaţiei sub formă de orientare în spaţiu, ceea ce permite o generare de traiectorii intuitive dependente de mişcarea mâinii utilizatorului. Algoritmii de obţinere a orientării, precum şi algebra din spatele noţiunii vor fi prezentate pe larg în capitolele 4 şi 5.

Capitolul 3

Studiu Bibliografic

Întrucăt reprezintă un punct cheie în aplicaţiile domeniului navigaţiei autonome, studiul detecţiei şi urmăririi corpurilor se bucură de o literatură vastă. Această considerabilă varietate devine şi motivul existenţei diverselor şcoli de găndire în privinţa tipurilor de senzori folosiţi sau algoritmilor de fuziune. Studiul bibliografic elaborat în acest capitol are ca scop expunerea sumară a fundamentului teoretic din spatele diferitelor lucrări care fac legătura cu subiectul expus în capitolul precedent, urmâand ca în capitolul ulterior doar o parte dintre ele să fie expuse pe larg şi considerate pentru implementare. Am vizat astfel să sintetizez esenţialul unui domeniu larg sub forma unei incursiuni istorice care cuprinde atât familiarizarea cu algoritmii clasici ce au revoluţionat tehnica din vremea lor, dar şi metodele de ultimă oră cercetate si dezvoltate în timpul scrierii acestei lucrări.

Introdus pentru prima dată în reputata lucrare [1], filtrul eponim propus de Rudolf E. Kalman pentru a oferi o şolutie de filtrare liniară discretă, deşi supus iniţial unui scepticism general, a devenit curănd o contribuţie notabilă în proiecte ce au lăsat o amprentă majoră în evoluţia stiinţei pănă în acel punct. Algoritmul a fost folosit in programul spaţial Apollo [2] pentru estimarea traiectoriei rachetelor către lună, şi a devenit de atunci un nou standard în industria respectivă precum şi izvorul de inspiraţie pentru soluţiile viitoarelor industrii ce aveau să revoluţioneze statu-quo al tehnologiei din acel moment.

În contextul sistemelor de control liniare şi invariante în timp, Kalman a definit o descompunere a reprezentatării matriciale generale (spaţiul stărilor) care facilitatează explorarea observabilităţii structurilor de reglare. Împreună cu această nouă noţiune, algoritmul, iniţial dezvoltat pentru analiza seriilor temporale în cadrul procesării semnalelor, oferă un model realist simplificat pentru estimarea stărilor actuale ale procesului în cauză şi aplicabilitatea sa a fost regăsită în deosebi în aplicaţii precum planificarea mişcării roboţilor, avioanelor şi vehiculelor, dar şi în studii precum econometria ori neuroştiinţa, care vizează contextul modelării dinamicii sistemului nervos central uman.

In articolul [3], autorii propun folosirea unui filtru de particule pentru fuzionarea datelor de la un senzor de tip LiDAR (Light Detection and Ranging). Aceştii senzori funcţioneaza prin emiterea unor pulsuri de lumină către mediul exterior [4]. Aceste unde

se întorc la senzor si distanţa parcursă este calculată in funcţie de timpul de emisie-receptie. Repetănd acest proces de milioane de ori pe secundă o hartă tridimensională a mediului încojurător este creată. Pentru filtrul de particule au fost necesare mai multe simulări de tip Monte Carlo (variabile aleatorii), ceea ce a dus implicit si la cresterea timpului de execuţie. Autorii menţionează opţiunea folosirii unui filtru Kalman extins cu precizarea că liniarizările in fiecare punct pot duce la erori mari de estimare. Modelarea miscării unui corp este mai departe descrisă simplificat avănd în minte noţiuni de bază ale mecanicii pentru miscările de viraj si funcţionarea senzorului descris anterior.

Autorii lucrării [5] propun rezolvarea aceleiaşi probleme printr-un algoritm de fuziune a măsurătorilor de la un senzor de tip RaDAR (Radio Detection and Ranging) si unul de tip LiDAR ce are la bază un filtru Kalman extins. Ei mentioneaza imbunatăţirea computaţională rezultată din evitarea unei implementări a unui algoritm de felul filtrului de particule. Se poate constata, aşadar, incă dintr-o scurtă incursiune in literatură cum un obiect de stiudiu ce genrează interes va genera mereu şi o scindare a opiniilor cercetatorilor. Lucrarea teoretizează şi validează trei profile diferite de viteză în scopul imbunătăţirii viitoarelor echipamente de ADAS (Adaptive Driver Assistance Systems) şi ramificaţii ale noţiunii de condus automon precum LKA (Lane Keeping Assistance) ori ACC (Adaptive Cruise Control).Funcţionarea senzorilor aleşi, precum şi stările sistemului observat in cadrul algoritmului de fuziune sunt descrise mai apoi pe larg în restul lucrării.

Măsurătorile senzorilor de tip RaDAR fuzionate cu filtre Kalman sunt un topic relevant si pentru industria aerospaţială, după cum observă cercetătorii care au condus studiul [6]. Un scenariu în care miscările neliniare prevalează poate fi modelat cu un filtru Kalman extins, indiferent dacă mişcarea este pe teren sau în aer. O îmbunătăţire a algoritmului clasic ce constă în normalizarea matricei de covarianţă este introdusă de autori ca o necesitate pentru a rezolva limitările filtrului extins în ceea ce priveşte convergenţa negarantată (spre deosebire de cea a unui filtru liniar, care însă nu este un bun estimator pentru un sistem neliniar) şi liniarizările ce pot fi inadecvate. Aceste limitări urmează să fie expuse şi studiate in această lucrare, impreună cu posibile soluţii.

Deşi ţintele urmărite în apă sunt mai afectate de interferenţe decăt cele din scenariile prezentate anterior, filtrul Kalman se dovedeşte o metodă bună de estimare şi în spaţiul subacvatic, dacă senzorii folosiţi sunt adecvaţi. În lucrarea [7], o combinaţie între algoritmul clasic anterior menţionat şi o reţea neuronală de tip LSTM (Long Short Term Memory) este prezentată ca o soluţie pentru urmărirea corpurilor acvatice cu ajutorul senzorilor de tip SoNAR (Sound Navigation and Ranging). Autorii concluzionează cu ideea că filtrul clasic poate fi imbunatăţit prin folosirea unui model de reţea neuronală convoluţionala, intrucâat noul sistem nu necesită informaţii valide a priori, iar robusteţea pentru scenariile neliniare se imbunătăţeşte considerabil.

Urmărirea obiectelor folosind senzori LiDAR este studiată si in articolul [8], în care estimarea poziţiei se realizează prin extinderea vectorului de stări astfel încât algoritmul să urmărească atăt un spaţiu static local, cât şi dinamica obiectelor în mişcare din apropierea

unei maşini. Algoritmul folosit pentru asocierea a unui set mare de măsuratori este unul de tip Branch And Bound, iar urmărirea este realizată cu un filtru Bayesian recursiv, Kalman extins.

Introdus pentru prima dată de Jeffrey Uhlmann şi Simon Julier [9] pentru a rezolva problema divergenţelor prezente în estimările filtrului Kalman extins, filtrul Kalman unscented a devenit un nou standard pentru aplicaţiile ce adresează sisteme neliniare. Algoritmul introduce o nouă metodă de calculare şi propagare a stăriilor si a matricei de covarianţă in timp, pe baza unui set finit de eşantioane de măsuratori , mediate ponderat pentru a aproxima funcţia neliniară pană la echivalentul unei serii Taylor de ordin III [10], spre deosebire de filtrul Kalman extins ce liniarizează sistemul neglijând termenii de ordin superior, fiind bazat pe o expansiune a seriei Taylor de ordin I. Filtrele extinse de ordin II, precum cel propus în lucrarea [11] necesită resurse de calcul în plus pentru matricile de tip Hessian, pe când un mare avantaj al filtrului unscented este ca nu introduce într-atât de multă complexitate computaţionala în plus, şi nici nu există nevoia calculării a priori a unor derivate.

Pentru a întelege pe deplin avantajele oferite de modificarea introdusă de filtrul unscented, transformata neliniară trebuie privită în contextul statistic al distribuţiilor funcţiilor de densitate probabilistică. Această caracteristică este cea prezentată şi de lucrarea [10], şi va fi explicată în detaliu în capitolul 4.

In lucrarea [12], algoritmul anterior menţionat este folosit în combinaţie cu o tehnică de intregrare pentru estimarea poziţiei subacvatice folosind măsuratorile unui SoNAR, in timp ce [13] foloseşte procedura ca o metodă de corecţie. Prima publicaţie explorează imbunataţirile aduse performanţelor unui model hibrid în domeniul defensiv, unde detecţia şi localizarea torpilelor este un element vital în dezarmarea trupelor oponente. Autorul jurnalului trece în revistă posibilele soluţii pentru rezolvarea problemei. Un iniţial filtu Kalman liniarizat, distinct faţa de filtrul liniar sau extins, este evaluat ca o posibilă soluţie. Acesta diferă de metodele anterior menţionate prin faptul că procesul este liniarizat înainte de a trece prin algoritm, ceea ce face procedura în esentă o combinaţie între cele două proceduri standard, care în final prezintă limitări în scenariul dat. Filtrul Kalman extins este apreciat ca o soluţie mai bună, dar problema divergenţei datorată neliniarităţilor severe nu este rezolvată pe deplin. Transformata unscented a filtrului este apreciată pentru îmbunătăţirea semnificativă pe care o aduce convergenţei funcţiei de densitate probabilistică ce modelează covarianţa procesului. Filtrul de particule este notat ca o "implementare pe scală mare a filtrului unscented", cu precizarea că procedura introduce un strat de complexitate problematic pentru aplicaţiile de timp real. În final, cercetătorul propune propriul algoritm hirid fundamentat pe modelul unui filtru unscented.

Neliniarităţile măsuratorilor senzorilor de tip RaDAR sunt luate în considerare în lucrarea [14], ai cărei autori introduc o modificare în calcularea recursivă a covarianţei masurătorilor pentru a renormaliza distribuţia zgomotului după fiecare fază de corecţie. Algoritmul original, precum si predecesorii lui mizează pe prezenţa zgomotelor albe pentru

a putea produce estimări convergente, aşa cum va fi demonstrat în următorul capitol, iar consistenţa acestei aproximări nu este garantată de metode în sine. Lucrarea descrie în introducere metodele clasice de urmărire, printre care se numără intocmai cele menţionate mai sus şi cântăreşte avantajele şi dezavantajele lor în acord cu cele relevate în acest studiu.

Odată cu apariţia noţiunii de "Internet al Lucurilor" (Internet of Things - IoT), după cum observă autorii meta-analizei [15] , senzorii au inceput sa fie categorizaţi în funcţie de capacitatea lor de a condiţiona sau nu comenzi şi acţiuni fară intervenţia umană. Această sciziune a dus la apariţia conceptului de de senzori *deştepţi* şi senzori care nu sunt *deştepţi*, numiţi senzori *de bază*. Autorii descriu senzorii deştepţi ca acei senzori care dispun de resursele computaţionale necesare unei luări de decizii ulterioare în design-ul lor interior. Cu alte cuvinte, dispozitivele precum cele anterior menţionate, LiDAR, RaDAR, SoNAR pot fi considerate inteligente pentru că produc la ieşire o informaţie care poate fi folosită de sine stătător pentru amplificarea percepţiei exterioare. Măsuratorile acestori senzori pot fi deci folosite şi pe cont propriu pentru a lua decizii în aplicaţii precum condusul autonom, iar fuziunea datelor provenite de la ei este în esenţă o metoda de îmbunatăţire a cadrului de decizie, respectiv o prevenţie a erorilor generate de neajunsurile lor.

În această categorie inteligentă de dispozitive pot fi încadraţi şi senzorii de tip accelerometru şi giroscop. Evoluţia lor recentă sub formă de sisteme microelectromecanice (MEM Systems) a permis integrarea lor în telefoane mobile, tablete, ochelari VR (Virtual Reality) şi oricare alte aparate care necesită informaţie de la mediul exterior [16]. Avănd partea electronică integrată, aceşti senzori permit implementarea unui protocol de comunicaţie si a blocurilor logice ce pot fi folosite mai departe de un procesor ce ia parte la luarea unei decizii, după cum evaluaează şi cercetătorul care a realizat lucrarea [17]. Esenţa numeroaselor avantaje provenite din integrarea sistemelor inteligente în senzori poate fi sumarizată prin afirmaţia oferită de autor conform căreia doar un nivel ridicat de programare (însemnănd că aspectul obiectiv mai dificil al programării de nivel jos a fost rezolvat) mai este necesar pentru procesarea datelor obţinute. Algoritmii de fuziune a datelor de la senzori fac parte tocmai din acest nivel înalt de programare ce vizează a obţine încă şi mai mult de la dispozitivele în cauză. Tocmai de aceea, deşi evoluţia electronicii care a adus la apariţia tehnologiilor MEMS până în acest punct ar putea constitui un subiect de teză în sine, lucrarea pe care o propun se va axa pe aspectele fine de nivel înalt ale modelării şi programării unor senzori ce deja pot fi consideraţi inteligenţi.

Unităţile de măsură inerţiale sau IMU (Inertial Measurement Unit) este denumirea pe care o poartă categoria din care fac parte accelerometrele şi giroscoapele. Majoritatea dispozitivelor electronice ce culeg informaţii despre orientarea în spaţiu de obicei conţin această pereche de senzori, căteodată într-un triplet ce include şi un magnetometru. Motivul diviziunii resurselor pentru a oferi o diversitate tipului de unitate de măsură este evident justificat de conceptul de fuziune al senzorilor anterior descris. Atât accelerometrul, cât şi giroscopul şi magnetometrul sunt senzori care oferă informaţii despre orientarea unui corp în spaţiu. Mecanismele lor diferite de funcţionare cauzează inevitabil atât avantaje cât şi dezavantaje comparabile pentru detecţia acelorași tipuri de miscări.

Aceste dezavantaje urmează să fie expuse în restul studiului bibliografic pentru a motiva necesitatea fuziunii datelor. Capitolul 4 va oferi o perspectivă minuţioasă a teoriei din spatele metodelor, umând ca în capitolele 5 şi 6 utilitatea şi imbunătăţirea considerabilă adusă prin fuziune să fie demonstrăte într-un scenariu implicat activ ce cuprinde o testare şi validare pe date reale.

Provenit din limba greacă, cuvântul giroscop a descris incipient un "cerc de observare". Iniţial de factură mecanică, senzorul a evoluat până în stadiul actual pentru a capta informaţii referitoare la viteza unghiulară a corpului de care este ataşat. Funcţionarea senzorului este explicată de efectul Coriolis, aşa cum explică autorii în lucrarea [18]. Viteza sau velocitatea rotativă poate fi exprimată fizic ca derivata poziţiei unghiulare în raport cu timpul. Integrănd această mărime se poate recupera poziţia unghiulară de la un moment dat, dar nu unghiul complet parcurs, un aspect ce se va dovedi esenţial în analiza dezavantajelor senzorului. Funcţia matematică ce defineşte aria de sub curba unui grafic de măsuratori aplicată datelor obţiunte de la un giroscop poate returna deci doar *schimbarea* în unghi ce a avut loc intr-o perioadă de timp, sau unghiurile relative, în contrast cu unghiurile absolute, iar acest aspect face ca informaţia unei referinţe iniţiale să nu poata fi oferită de un giroscop de unul singur. Mai mult, după cum va fi demonstrat în capitolele ce urmeaza integrarea numerică produce un efect de deplasare a masurătorilor statice, numit în literatură *drift*. Acest efect este în deosebi responsabil pentru amplificarea erorilor care apar în zgomotele de frecvenţa redusă care afectează măsurătorile senzorului. O primă încercare de rezolvare a acestei probleme a fost introducerea unui filtru trece-sus pentru a permite doar semnalelor de scurtă durată să fie înregistrate, adică exact a schimbărilor în unghi de interes, dar nu şi a efectului de drift [19].

Accelerometrele, după cum sugerează şi numele, sunt dispozitive ce măsoara acceleraţia. Acestea funcţionează pe baza vibraţiilor [20] şi oferă date atât despre mişcare, măsurănd acceleraţiile liniare, cât şi statice, acceleraţia datorată gravitaţiei. Datorită componentei gravitaţionale, aceşti senzori, spre deosebire de giroscoape, introduc noţiunea unui sistem de referintă. Indiferent de mişcarea obiectului, vectorul gravitaţiei o să fie indreptat mereu în jos, spre centrul pământului. La fel ca în cazul giroscoapelor, şi datele de la accelerometre pot fi procesate pentru a oferi informaţii despre unghiul de mişcare. De data asta nefiind vorba de derivata unei mărimi, o simplă integrare nu poate fi aplicată masurătorilor. Totuşi, acestea pot fi simplu prelucrate prin funcţii trigonometrice ce urmează să fie explicate în capitolele următoare. Un accelerometru funcţionează prin masurarea tuturor forţelor care acţionează asupra lui. Acest aspect, deşi pozitiv în aparenţa pentru acurateţea cu care detectează mişcarea, face ca senzorul să fie afectat de zgomot într-o măsură mai mare decât un giroscop. O rezolvare simplă dar ineficientă comparativ cu metodele de ultimă oră a fost propusă iniţial prin introducerea unui filtru trece-jos (FTJ) care este capabil să elimine zgomotele de frecvenţe înalte cu dezavantajul că introduce o întârziere [19].

După o sumară analiză a celor doi senzori, se poate trage o concluzie bazată pe funcţionalitatea fiecăruia relativ la celălalt. Astfel, dacaăccelerometrele produc măsura-

tori absolute, prin vectorul informativ al acceleraţiei gravitaţionale ce îsi păstrează valoarea de referintă, giroscopul poate oferi doar măsuratori relative, în funcţie de propria lui mişcare. În contextul inerţiei senzoriale, se poate spune că accelerometrul se raportează la un sistem numit cadru mondial, pe când mişcarea giroscopului capătă sens atucni când este privită în cadrul său propriu. Aceste două noţiuni vor avea o importanţă semnificativă în exprimarea orientării în spaţiu, ce urmează să fie descrisă în continuarea studiului bibliografic şi elaborată amănunţit în capitolul ce vizează teoria de la baza tezei. De asemenea, faptul că accelerometrul este afectat de zgomote de frecvenţe înalte îl face nefiabil pentru folosinţa de termen scurt, dar relevant pentru perioade mai lungi. La polul opus, giroscopul produce pe termen lung efectul de drift, dar este fiabil pe termen scurt pentru a oferi informaţii despre schimbarea în rotaţie a obiectului.

Aceste avantaje şi dezavantaje se completează reciproc fascinant şi este motivul pentru care cei doi senzori de obicei sunt folosiţi împreuna În loc de a folosi însa o metodă simplă de control print-un releu bipoziţional care ar putea decide când un senzor este mai favorabil decât celălat, algoritmii de fuziune a datelor au evoluat pâna în momentul de faţă sub formă de metode inteligente capabile să determine singure gradul de încredere ce trebuie acordat fiecarei măsurători prezente în sistem.

Folosirea cuplului de senzori precum şi rezolvările contrastate complemntar prin cele două filtre polar opuse ca funcţionare a dus la o primă tehnica de fuziune elementară care stă la baza unui algoritm arhicunoscut numit sugestiv filtrul complementar, descris tot în lucrarea [19]. Această simplă soluţie are marele avantaj de a fi uşor de implementat în aproape toate limbajele de programare de nivel înalt, şi poate fi suficientă pentru aplicaţii electronice de scală mică. Înmulţirea unghiului extras din măsuratorile accelerometrului cu o constanta şi adunarea produsului cu o înmultirea dintr rezultatul integrării vitezelor unghiulare citite de la giroscop cu o constantă complementară faţă de unitate cu cea a accelerometrului devine o simplă linie de cod, ceea ce face ca problema complexităţii computaţionale să nu facă parte din discuţie.

Prin intermediul unui prim banal dar eficient algoritm de fuziune senzorială încep să apară noi orizonturi de îmbunatăţire a performanţelor unei astfel de metode. Avantajul ei devine incontestabil pentru aplicaţiile senzorilor inerţiali şi subiectul dezbaterii ştiinţifice se îndreaptă mai curând spre exploatarea beneficiilor unei astfel de soluţii decât înspre chestionarea necesităţii ei.

Mai avansaţi, dar şi mai exhaustivi din punct de vedere computaţional sunt algoritmii de fuziune pe baza teoriei probabilităţii şi statisticii. Posibile variaţii ale algoritmului Kalman au fost explorate la începutul acestui studiu bibliografic şi algoritmul se dovedeşte a fi o soluţie bună pentru aproape orice fel de problemă senzorială. În contrast cu filtrul complementar descris anterior, în implementarea căruia programatorul trebuie să decidă singur gradul de încredere acordat fiecărui senzor în parte (constanta de filtrare), filtrul Kalman decide statistic ponderea optimă care trebuie atribuită la fiecare moment de citire al unei măsuratori, prin iniţializarea anumitor parametri de acordare. Toate aceste idei,

precum şi cele de mai sus, îsi găsesc confirmarea în lucrarea [19]. Acordarea respectivilor parametri, precum şi implementarea algoritmulilor în sine vor fi redijate în capitolele următoare.

În completarea perechii de senzori, multe aplicaţii sunt perfecţionate prin adăugarea unui magnetometru. Acest nou senzor a existat sub o formă sau alta timp de milenii, după cum remarcă autorii lucrării [21]. Proprietăţiile magnetice ale celor doi poli geografici de care dispune planeta au fost probabil primul motiv pentru care dezvoltarea sistemelor de navigaţie a putut fi realizată. După cum este redactat în publicaţia citată, o rocă de foc magnetizata pusă într-un bol cu apă a fost primul proiect de succes obţinut de poporul chinez în detecţia polului sud magnetic, din apropiera polului nord geografic.

Magnetometrele sunt, în fond, busole avansate care, tot datorită evoluţiei MEMS pot fi prezente în dispozitive de uz zilnic, cu un rol de mare interes în navigaţia pedestră. Dispunând acum de informaţia referitoare la componenta magnetică a Terrei, azimutul poate fi calculat trigonometric [21], asemenea unui unghi de înclinaţie al accelerometrului. Mai mult, un magnetometru împreună cu un accelerometru devin suficiente pentru a oferi o percepţie completă a cadrului de referinţa în care se află sistemul studiat. Singura acceleraţie care acţionează asupra unui obiect staţionar este cea gravitatîonală şi astfel vectorul de acceleraţie va fi îndreptat în sus, opus ca sens faţă de cel gravitaţional. Magnetometrul arată înspre nord şi astfel sunt obţinute două axe ale sistemului de referintă. A treia axă a spaţiului tridimensional în care se află corpul-obiect poate fi obţinută simplu ca vectorul rezultant din produsul vectorial pe spaţiul definit de cei doi vectori anterior determinaţi şi atfel se obţine o referinţă completă numită în literatură NED (North-East-Down).

Se defineşte astfel, ţinând cont de noţiunile introduse în paragraful precedent orientarea unui obiect ca rotaţia corpului în cauză faţă de sistemul de referinţă NED. Având sistemul de referinţa definit de cei doi senzori, cea mai convenabilă formă de reprezentare a orientării este matricea de rotaţie de 3×3 bazată pe cei trei vectori N-E-D.

Deşi proiectul propus nu presupune utilizarea unui magnetometru, am considerat includerea unei sumare descriei necesară, intrucât aceasta oferă o întelegere mai profundă a sistemelor de orientare, numite în literatura AHRS (Attitude and Heading Reference Systems). Fără folosirea unui magnetometru, componenta ce descrie direcţia de îndreptare în planul longitudinal, numită giraţie [22], nu mai este evaluată, iar noul sistem este limitat la funcţionarea de tip ARS (Attitude Reference System). Acesta este şi sistemul pe care l-am modelat în scopul obţinerii unui comportament de tip joystick pentru microcontroller, şi incursiunea în funcţionalitatea şi importanţa unui senzorului-busolă a fost utilă în a justifica nevoia unuia în general, dar şi futilitatea lui în particular pentru contextul lucrării propuse. Cu alte cuvinte, pentru prezenta teză, obţinerea orientării sistemului folosit nu a solicitat un comportament de busolă, dar principiul descris va putea reprezenta un punct de plecare pentru o viitoare extensie a actualei lucrări.

Introducerea noţiunii de sistem de referinţă deschide calea către un concept de mare importantă în utilitatea dispozitivelor IMU, şi în consecinţă vital şi pentru proiectul propus,

vag menţionat anterior, orientarea unui corp.

Orientarea, adesea numită şi atitudine, defineşte felul în care un corp este poziţionat relativ la un sistem de referinţă. Ea poate fi exprimaţa matematic în mai multe forme cu scopul de dinamica poziţionării unui obiect sau unui corp observat într-un plan sau spaţiu tridimensional. Principalele trei tipuri de reprezentare a orientării sunt unghiurile Euler, introduse ce celebrul matematician al cărui nume îl poartă, matricile de rotaţie şi cuaternionii. Grupul de construcţii matematice este analizat exhaustiv de autorul lucrării [23], din care se pot trage concluzii clare despre avantajele şi dezavantajele fiecărei reprezentări.

Construcţia lui Euler, tripletul format din cele trei unghiuri ce reprezintă rotaţiile corespunzătoare din jurul celor trei axe ale spaţiului de referinţă, reprezintă, după cum observă şi Diebel James, cea mai comună alegere pentru reprezentarea orientării, pentru că întelegerea lor este facilă. Unghiurile sunt adesea numite ruliu, tangaj şi giraţie (roll, pitch, and yaw) în aplicaţiile ce ţin de navigaţie terestră, spaţială sau acvatică. Probabil cel mai important neajuns al simplei reprezentări este vulnerabilitatea la singularităţile provenite din blocarea unei axe, fenomen studiat şi cunoscum sub denumirea de Gimbal-Lock [23]. Concret, atunci căndd două dintre cele trei axe disponibile ajung într-o configuraţie paralelă (cum este cazul perpendicularităţilor), un grad de libertate se pierde deoarece cele două axe oferă aceeaşi informaţie.

Acest dezavantaj este suficient pentru a descalifica folosirea de sine statatoare a reprezentării în aplicaţii ce necesită mişcări perpendiculare. Un proiect fiabil pentru folosirea celor trei coordonate ar putea fi o cameră de filmat programată inteligent, care nu ar trebui niciodată să inregistreze imagini la $90°$, respectiv $-90°$ de grade (tavan şi podea) pentru o referinţa orizontală. Un caz asemănător, un avion care ar atinge una dintre cele două extreme pentru unghiul de tangaj (elevaţie) ar fi în predispoziţie de cădere, deci evitarea gimbal lock-ului nici nu ar avea sens. Aplicaţiile ce nu pot fi constrânse la un interval de valori şi deci necesită o orientare completă în spaţiu, cum este exemplul jocurilor video, sau al realitatăţii virtuale ori augmentate, nu pot fi concepute printr-o implementare indiferentă la astfel de singularităţi, şi impun construcţii mai avansate. Cu toate acestea, tripletul de unghiuri reprezintă baza teoretică pentru următoarele două categorii de reprezentare.

O matrice de rotaţie, adesea numită în literatură DCM (Direction Cosine Matrix) este o matrice de 3×3 care, înmulţită cu un vector dintr-un cadru de referintă, transpune acel vector într-un alt cadru de referinţa, păstrându-i magnitudinea [23]. Pentru fiecare dintre cele trei axe ale sistemului global se poate defini o transformare liniară ce cuprinde informaţii despre fiecare axă din sistemul de referinţă, obţinându-se astfel un sistem de trei ecuaţii cu 9 variabile noi introduse. Aceste nouă variabile sunt exact elementele unei matrici de rotaţie. Interpretarea valorilor este dată geometric de funcţia cosinus a unghiului dintre axele respective de pe fiecare coloană şi de pe fiecae rând al matricii, ceea ce conferă şi explicaţia pentru denumirea lor (Cosine Matrices). Notaţii diferite pentru semnele funcţiilor trigonometrice din interiorul matricilor pot fi regăsite în literatuă deoarece

acestea pot fi reprezentate sub forma a două convenţii, transformarea liniară din cadrul mondial în cadrul obiect, şi invers.

Construcţia avansată derivată din cele trei unghiuri iniţiale prezintă şi dezavantaje. Matricea definită trebuie să fie mereu normalizată la dererminant pentru a exprima o rotaţie corectă [23]. De asemenea, reprezentarea sub forma a nouă elemente în loc de trei va necesita folosirea mai multor resurse de calcul, deşi acest aspect nu reprezintă o problemă de actualitate la momentul scrierii lucrării, când majoritatea limbajelor de programare pot rezolva algoritmi bazaţi pe algebră liniară la fel de rapid ca operaţiile cu scalari sau vectoriale, iar puterea computaţională oferită de procesoare nu este un impediment. De fapt, nu există niciun dezavantaj evident pentru care s-ar evita folosirea matricilor de rotaţie. Singurul motiv pentru care acestea nu sunt mereu încorporate în aplicaţii este pentru că există o reprezentare care este, privind obiectiv din mai multe puncte de vedere, superioară.

Cuaternionii, după cum notează autorul lucării [24] sunt o descoperire a secolului 19 atribuită lui William Rowan Hamilton, care a descoperit cum poate teoretiza o a patra dimensiune pentru a explica rotaţii în spaţiul tridimensional. Algebra din spatele cuaternionilor este un subiect vast căruia îi va fi dedicat o întreagă secţiune teoretică în această lucrare, dar principalul interes pentru teza propusă este folosirea cuaternionilor pentru a defini rotaţii, iar aceasta se exprimă prin cuaternioni unitate (cu norma egală cu unitatea) [24]. Vectorul cvadridimensional descoperit de Hamiltion este nici mai mult, nici mai puţin decăt o extensie a sistemului complex de numere [25] definit de Carl Friedrich Gauss, şi se supune regulilor numerelor imaginare.

În esenţă, prima componentă reală dintre cele patru reprezintă unghiul de rotaţie al corpului, iar celelalte trei componente imaginare reprezintă cele trei axe de rotaţie ale cadrului de referinţă [26]. Astfel se poate imagina un cadrul tridimensional în care se desfăşoară o rotaţie, pentru că o concepţie despre un spaţiu cvadridimensional nu este posibilă pentru mintea umană. Acest nivel ridicat de abstractizare este probabil şi unul dintre puţinele dezavantaje ale acestei reprezentări. Extragerea unui înteles semnificativ pentru cele patru numere nu este intuitivă, iar necesitatea normalizării poate fi un impediment atuncti când se doreşte optimizarea algoritmului deja instanţiat. Totuşi, programaţi pe un calculator, cuaternionii se dovedesc superiori celorlalte două reprezentări, în principal prin capacitatea lor de a fi interpolaţi sferic prin SLERP (Spherical Linear Interpolation) [24], metodă ce permite interpolarea relativ la suprafaţa unei sfere unitate în contrast cu interpolarea clasică printr-o linie. Această tehnică, deşi abstractă, s-a dovedit în timp extrem de utilă pentru schimbarea lină a cadrelor în aplicaţii virtuale, şi astfel cuaternionii au devenit inevitabil reprezentarea de ultimă oră în domenii precum viziunea artificială sau realitatea augmentată.

Scopul în care lucrarea propusă va folosi vectorii cuaternioni rămâne însa fidel obiectivului înitial de a realiza o aplicaţie de urmărire, iar orientarea rotaţiilor este punctul cheie în funcţionarea jocului. Cuaternionii sunt un standard şi pentru proiectele de fuziune sen-

zorială, cum este cel prezentat în lucrarea [27], ce profită de avantajul cinematicii liniare a reprezentării 4D pentru a justifica folosirea unui filtru liniar. Lucrarea [28] oferă o metodă de implementarea filtrului Kalman unscented pentru fuziunea datelor unui IMU în vederea reconstrucţiei unui traseu simulat, folosind cuaternionii ca formă de orientare.

Având astfel definit obiectivul proiectului, precum şi noţiunile teoretice necesare în implementare, propun această teză cu scopul de a oferi o imagine de ansamblu a tehnicilor de fuziune senzorială într-o aplicaţie cu caracter ludic, dar cu nuanţe importante în industria emergentă a vehiculelor autonome. Un model de obţinere al orientării similar cu cel descris în articolul [29] va fi implementat pentru accelerometru, în vreme ce procesul modelului ce va descrie atitudinea obţinută de la giroscop se va baza pe cuaternioni. Folosirea senzorilor reali va presupune, în constrast cu cei simulaţi, o metodă de calibrare. Întrucât calibrarea senzorilor constituie un subiect de teză în sine, după cum este descris întregul proces în lucrarea de disertaţie [30], aspectele de mică importanţă nu vor fi tratate cu aceeaşi minuţiozitate cerută în industrie. Acestea fiind spuse, aplicaţia dezvoltată este complet funcţională şi poate servi ca model pentru arhitectura de la baza unui proiect de scală largă bazat pe urmărire în timp real şi fuziune senzorială, ce mai apoi va putea fi extins pentru a acoperi mai multe aspecte.

Capitolul 4

Analiză şi Fundamentare Teoretică

Propun acest capitol sub forma unei analize minuţioase a conceptelor menţionate anterior în studiul bibliografic în scopul oferirii unei perspective complete asupra ariei de interes înainte de elaborarea modului de dezvoltare a aplicaţiei, rezervată pentru capitolul următor. Doresc pe această cale să creez un cadru tehnic familiar pentru cititor şi să fixez astfel noţiunile necesare întelegerii algoritmilor avansaţi, pornind de la mecanica clasică, cu o trecere spre modelarea matematică sub formă de spaţiu al stărilor şi fundamentele statistice ce stau la baza proceduriilor de nivel înalt.

Tangenţele cu obiectul unor altor studii nu se doresc a fi exhaustive şi explicaţiile necesare vor fi limitate la aplicabilitatea oferită domeniului de interes pentru care această lucrare a fost redactată. În acest sens, capitolul prezent poate constitui un schelet teoretic pentru viitoare implementări ale unor proiecte mai avansate.

4.1 Ecuaţiile mişcării

În *Dialog despre cele două mari sisteme ale lumii* (1632), Galileo Galilei sfidează pentru prima dată ipoteza aristotelică conform căreia o forţă ar imprima unui obiect o viteză. Deşi într-adevăr obiectele în mişcare nu rămân în acest stadiu fără intervenţia unei forţe, principiul interţiei formulat de matematicianul florentin demonstrează că forţa imprimă unui corp acceleraţie, nu velocitate. Cu alte cuvinte, în legea căderii libere, Galileo a observat ca viteza unui corp va continua să crească la infinit, dar mişcarea ei va fi accelerată constant, acea acceleraţie fiind chiar acceleraţia gravitaţională. Mai mult, el a observat că traiectoria unui proiectil accelerat este o parabolă. Astfel, a fost definită ecuaţia căderii libere:

$$p = \frac{1}{2}gt^2 \tag{4.1}$$

Unde p este pozîţia corpului în cădere, g reprezină componenta gravitaţională a acceleraţiei, iar t este componenta temporală, o noutate introdusă de fizician în perspectiva

asupra cinematicii de la acea vreme. Pentru generalizare, o poziţie iniţială p_0 a corpului în cădere, precum şi viteza v care creşte nelimitat sunt introduse astfel:

$$p = p_0 + v + \frac{1}{2}gt^2 \tag{4.2}$$

Galileo a observat nu numai că viteza unui corp cu o acceleraţie constantă creşte la infinit, dar şi cum aceasta creşte liniar în funcţie de timp. A fost definită astfel funcţia de schimbare a vitezei în timp generalizaţa pentru a îngloba şi o viteză iniţială v_0:

$$v = v_0 + gt \tag{4.3}$$

Aceste două simple, dar revoluţionare schimbări de paradigmă din istoria formulării dinamicii obiectelor au fost generealizate de Isaac Newton şi pentru acceleraţii fără componentă gravitaţională în încercarea de a descrie dinamica sistemului solar folosind-se de analiza matematică. În contextul orbitării planetelor în jurul unei stele, cunoaşterea poziţiei unui corp ceresc la un moment dat este un factor cheie în a prezice următoare poziţie de la un următor moment de timp, după cum a descoperit fizicianul englez.

Cum Galileo a definit vag noţiunile de viteză şi acceleraţie ca nişte funcţii de raport în timp, introducere calculului derivativ a propulsat descoperirea într-o nouă formă, Newton definind viteza ca schimbarea în timp a acceleraţiei, iar poziţia ca schimbare în timp a vitezei.

$$v = \frac{dp}{dt}, \quad v = \dot{p} \tag{4.4}$$

$$c, \quad a = \dot{v} = \ddot{p} \tag{4.5}$$

Se poate rescrie astfel prima ecuaţie sub forma generalizată:

$$v = v_0 + \frac{dv}{dt}\Delta t \tag{4.6}$$

Viteza medie $\bar{v}$ a unui corp poate fi scrisă în funcţie de viteza iniţială v_0 şi viteza finală v conform formulei:

$$\bar{v} = \frac{v_0 + v}{2} \tag{4.7}$$

$$v = v_0 + at \tag{4.8}$$

Şi astfel:

$$\bar{v} = v_0 + \frac{1}{2}at \tag{4.9}$$

Ceea ce duce la rescrierea celei de-a doua ecuaţii sub forma:

$$p = p_0 + \bar{v}t \tag{4.10}$$

$$p = p_0 + v_0 t + \frac{1}{2}\frac{d^2 p}{dt^2}\Delta t^2 \tag{4.11}$$

Pot fi deci simulate soluţiile ecuaţiilor diferenţiale prin integrare numerică în timp fără alte cunoştinţe despre proprietăţi ale obiectului în mişcare, şi ipoteza lui Galileo conform căreia traiectoria unui obiect reprezintă o parabolă a unei ecuaţii de gradul II este validată grafic în figura 4.1.

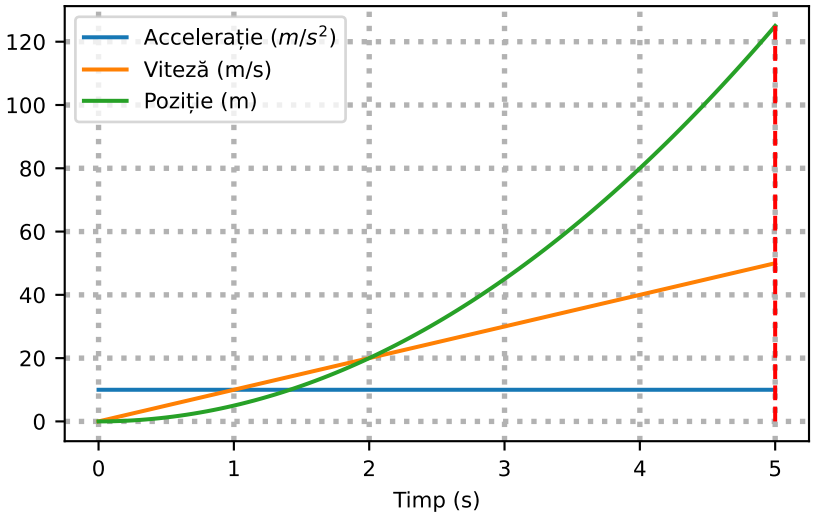


Figura 4.1: Dinamica mișcării pentru o accelerație constantă

Teoretic, starea precedentă (notată ca stare iniţială în formule) şi dinamica descrisă ar trebui să fie suficiente pentru a extrapola poziţia unui obiect de la un moment de timp $t + \Delta t$ următor. În practică, starea măsurată este afectată de zgomot de măsură de la senzori, notat de acum ν, iar dinamica este afectată de zgomot de proces, din cauza turbulenţelor prezente în mediul de observare, notat w. Astfel, celee doua cuaţii devin:

$$v = (v_0 + \nu_v) + \frac{dv}{dt}\Delta t + \omega_v \tag{4.12}$$

$$p = (p_0 + \nu_p) + v_0 t + \frac{1}{2}\frac{d^2 p}{dt^2}\Delta t^2 + \omega_p \tag{4.13}$$

4.2 Dinamica sistemului

Ecuaţille diferenţiale tratate ca un întreg şi evaluate pentru schimbarea valorilor dependente în timp formează reprezentarea cunoscută sub numele de spaţiul stărilor:

4.2.1 Timp continuu

Pornind de la ecuaţiile (4.4) şi (4.5) se poate compne modelul sistemului sub formă matricială astfel:

$$\begin{bmatrix} \dot{v} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} v \\ p \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot a \tag{4.14}$$

Variabilele derivate devin stările sistemului şi ieşirea y care măsoară poziţia unui obiect poate fi definită pentru sistem în felul următor:

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ p \end{bmatrix} \tag{4.15}$$

Pentru un obiect care îsi schimbă poziţia într-un plan bidimensional se poate defini spaţiul stărilor relativ la axele x şi y prin extensia modelului unidimensional anterior definit astfel:

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{p_x} \\ \dot{p_y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ p_x \\ p_y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{4.16}$$

Iar poziţiile vor fi măsurate la ieşire astfel:

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ p_x \\ p_y \end{bmatrix} \tag{4.17}$$

Forma generală a sistemului pentru vectorul de 4 stări x rezultă astfel şi acceleraţia ca intrare u pentru sistem:

$$\dot{x} = Ax + Bu \tag{4.18}$$

$$y = Cx \tag{4.19}$$

Cu A, B, C definite ca matricile de tranziţie, control, respectiv observare.

4.2.2 Observabilitatea sistemului

Pentru ca urmărirea traiectoriei obiectului în mişcare să fie posibilă, sistemul corpului trebuie să fie observabil. R.E. Kalman a definit astfel matricea de observabilitate şi condiţia pentru care sistemul cu $n = 4$ stări să fie complet observabil [31]:

$$\gamma_{A,C} = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}, \quad rang(\gamma_{A,C}) = n \tag{4.20}$$

Un estimator liniar [32] validează observabilitatea sistemului:

$$\dot{\hat{x}} = A\hat{x} + Bu + L\epsilon \tag{4.21}$$

$$\hat{y} = C\hat{x} \tag{4.22}$$

$$\epsilon = y - \hat{y} \tag{4.23}$$

unde ϵ reprezintă eroarea de estimare şi pentru care vectorul L poate fi calculat prin alocare de poli astfel încât rezultatul $A - LC$ să fie Hurwitz [33], adică :

$$det[\lambda I_4 - (A - LC)] = 0 \tag{4.24}$$

Se poate valida printr-o simulare grafică convergenţa stărilor estimate la cele măsurate prin minimizarea erorilor vectorului ϵ atunci când $y - \hat{y} \to 0$ (figura 4.2).

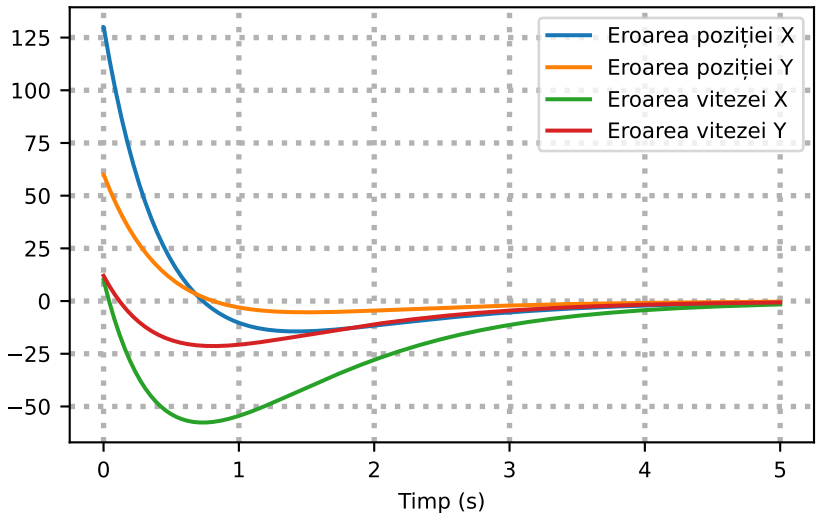


Figura 4.2: Dinamica erorilor

Demonstraţia observabilităţii sistemului (figura 4.3) propus este punctul de plecare pentru urmărirea traiectoriei în timp real pentru o dinamică agnostică în raport cu stările iniţiale.

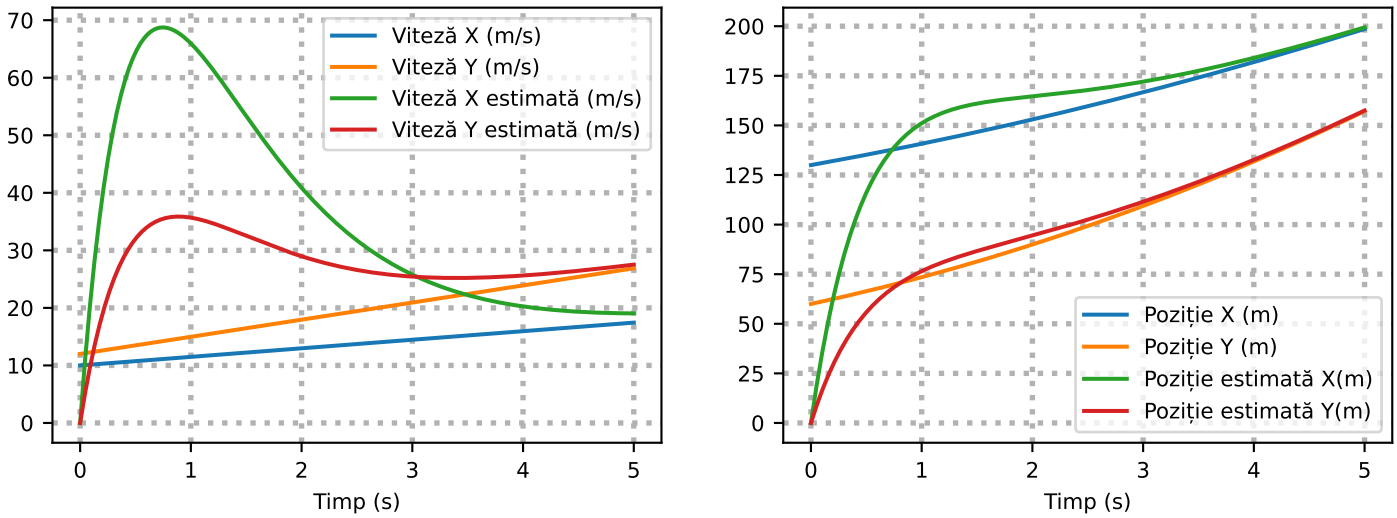


Figura 4.3: Estimarea convergentă a stărilor sistemului

4.2.3 Estimarea perturbaţiilor

Într-un scenariu real, dinamica sistemului simulat va fi afectată de perturbaţii precum bătaia vântului, valuri de maree sau instabilitatea solului în funcţie de contextul în care se desfăşoară urmărirea.

Modelul obţinut se poate extinde pentru a îngloba dinamica perturbaţiei [34]:

$$\dot{\hat{x}} = A\hat{x} + Bu + Ed \tag{4.25}$$
$$\hat{y} = C\hat{x} \tag{4.26}$$
$$\tag{4.27}$$

Acesta poate fi augmentat pentru a observa perturbaţia astfel încat condiţia de observabilitate, rang $(\gamma_{A_e,C_e}) = n + 1$ să fie îndeplinită [34]:

$$\dot{\hat{x}_e} = A_e\hat{x}_e + B_eu + L\epsilon \tag{4.28}$$
$$\hat{y} = C_e\hat{x}_e \tag{4.29}$$
$$\epsilon = y - \hat{y} \tag{4.30}$$

unde [34]:

$$A_e = \begin{bmatrix} A & E \\ 0 & 0 \end{bmatrix}, \; B_e = \begin{bmatrix} B \\ 0 \end{bmatrix}, \; C_e = \begin{bmatrix} C & 0 \end{bmatrix} \tag{4.31}$$

Plasarea polilor estimatorului astfel încât matricea $(A_e - LC_e)$ să fie Hurwitz se realizează cu funcţia *place* [35] din modulul *control* extern de Python pentru polii $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ cu partea reală negativă aleşi astfel:

$$L = place({A_e}^T, {C_e}^T, [\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5]) \tag{4.32}$$

Simularea sistemului pentru un set de condiţii iniţiale demonstrează conform figurii 4.4 că modelul extins este capabil să estimeze perturbaţii constante sau deterministe.

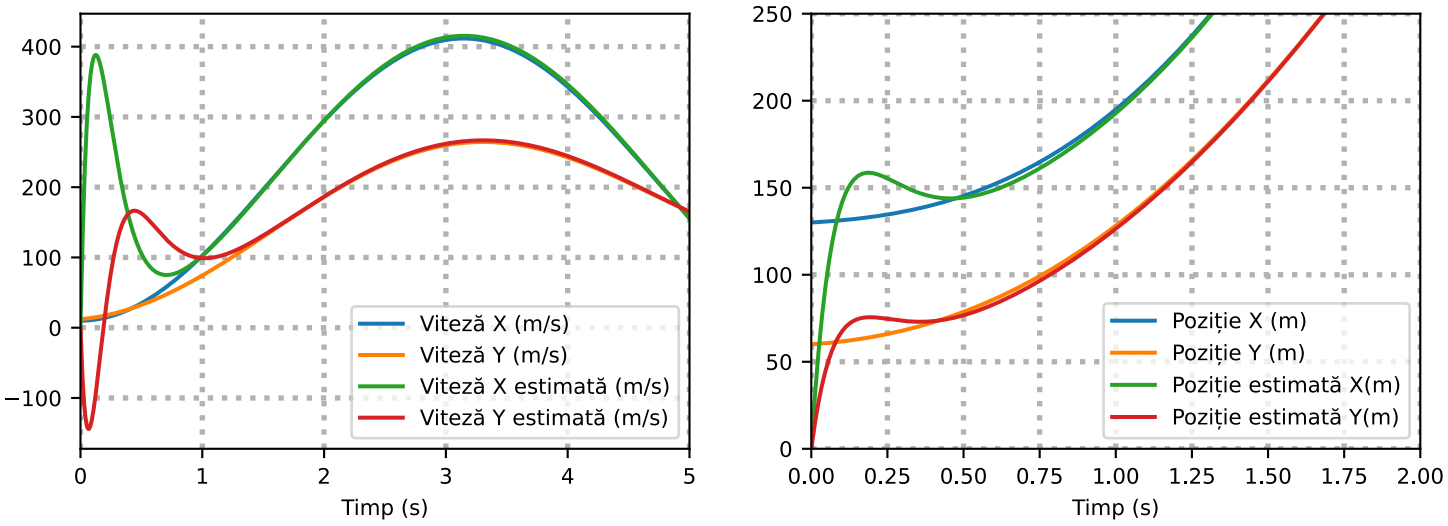


Figura 4.4: Estimarea stărilor sistemului

Limitarea pe care extensia modelului liniar o prezintă este incapabilitatea observatorului de a aproxima perturbaţii stohastice. De cele mai multe ori, zgomotul prezent în modelul procesului, dar şi în modelul măsurătorilor este de natură probabilistică, ceea ce deschide calea pentru studierea unor estimatoare mai avansate.

Perturbaţiile liniare sau liniare pe părţi pot fi privite ca o intrare externă a sistemuli şi de aceea estimarea lor nu necesită algoritmi probabilistici. Sinusoidele, deşi neliniare, au un comportament detrminist în timp în funcţie de perioada de oscilaţie şi amplitudine, care rămân componente constante pentru funcţie. Dezechilibrele prezente în scenariile de urmărire a obiectelor sunt, după cum se va vedea, sporadice, şi comportamentul lor trebuie încadrat statistic într-o distribuţie de valori posibile, după cum urmează să fie demonstrat.

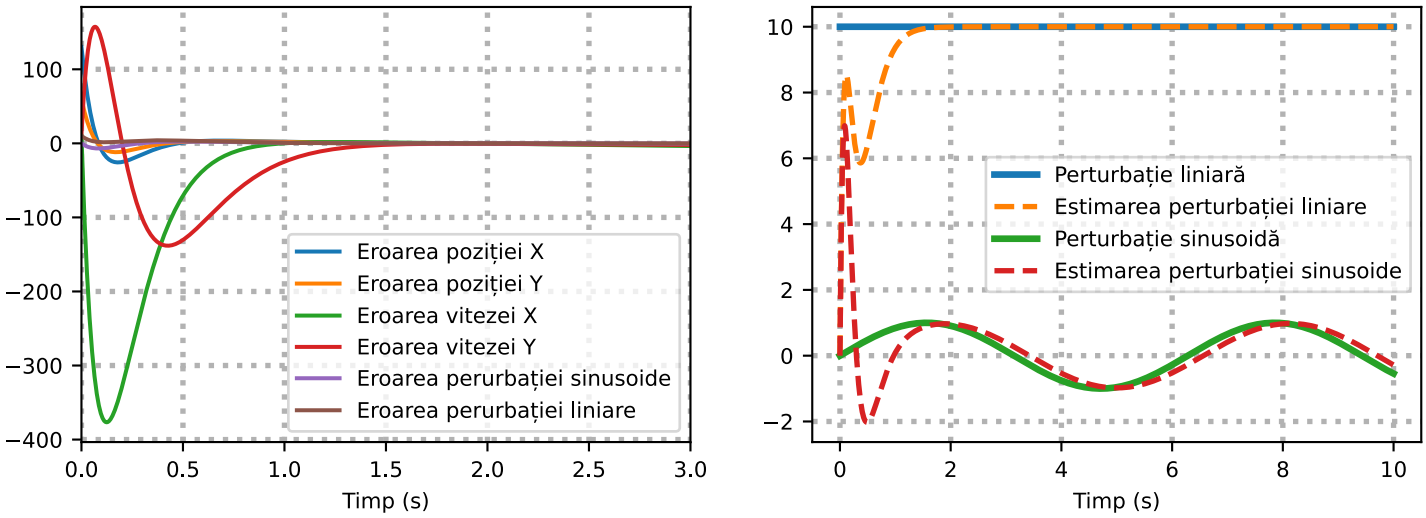


Figura 4.5: Dinamica erorilor şi estimarea perturbaţiilor

4.2.4 Estimatorul Luenberger

O posibilă metodă de rejectare a destabilizărilor este decuplarea lor cu totul de la intrarea sistemului. De asemenenea, ele nu vor mai fi integrate în vectorul de stare, şi prin urmare nu vor fi nici estimate.

În 1966, David Luenberger propune o nouă formă canonică [36], ca un compromis dintr-un estimator liniar şi invariant în timp, cu o dinamică a polilor mai rapidă decăt a sistemului obsevat şi un sistem aproximativ al ieşirilor măsurate pentru o rejectare rapidă, dar degradată de zgomotul aditiv din măsurători.

O perspectivă asupra motivaţiei autorului poate fi desprinsă şi din figura 4.5, în care se observă că erorile care converg rapid au în dinamica lor şi un suprareglaj semnificativ, din cauza agresivităţii cu care observatorul liniar alocă poli negativi cu valoare absolută mare. Acest lucru este fiabil pentru simulări, dar nu şi pentru majoritate aplicaţiilor reale.

Luenberger a propus atfel pentru modelul liniar, invariant în timp:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.33}$$

$$y(t) = Cx(t) \tag{4.34}$$

sistemul de observare guvernat de [37]:

$$\dot{z}(t) = Fz(t) + Gx(t) + TBu \tag{4.35}$$

pentru perechea (A, C) complet observabilă.

Pentru acest sistem, s-a definit controlul intrării [37]:

$$u(t) = Kx(t) = (K1 + K2)x(t) \tag{4.36}$$

Concret, pentru vectorul de stare estimat [38]:

$$\hat{x} = z + Hy \tag{4.37}$$

pentru care H poate fi rescris în funcţie de pseudo-inversa matricei (CE), notată $(CE)*$ [38], sub forma:

$$H = E(CE)* \tag{4.38}$$

se poate obţine setul de ecuaţii [38]:

$$K1 = place(TA^T, C^T, [\lambda_1, \lambda_2, \lambda_3, \lambda_4]) \tag{4.39}$$
$$T = I_4 - HC \tag{4.40}$$
$$F = TA - K1^T C \tag{4.41}$$
$$K2 = FH \tag{4.42}$$
$$\tag{4.43}$$

Figura 4.6 arată cum pentru o perturbaţie stocastică de tip zgomot alb estimarea stărilor nu diverge. Un dezavantaj al estimatorului Luenberger este necesitatea îndeplinirii condiţiei $rank(E) = rank(CE)$ [38] care limitează posibilităţiile modelării dezechilibrelor.

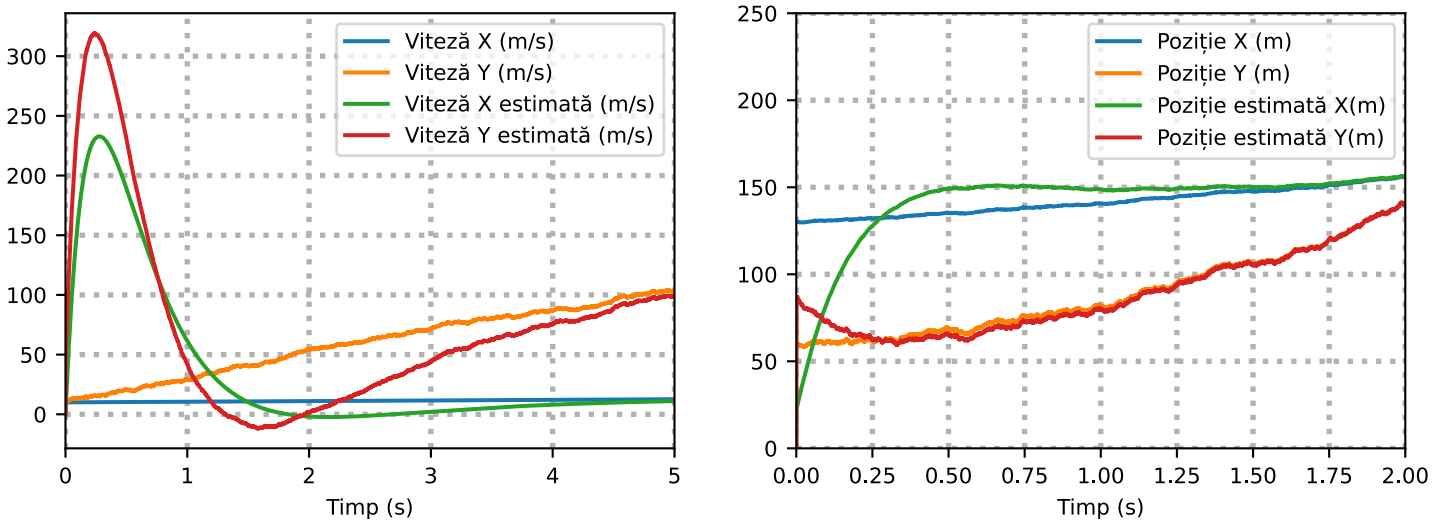


Figura 4.6: Estimarea stărilor folosind forma Luenberger

Figura 4.7 captează dinamica erorilor de estimare.

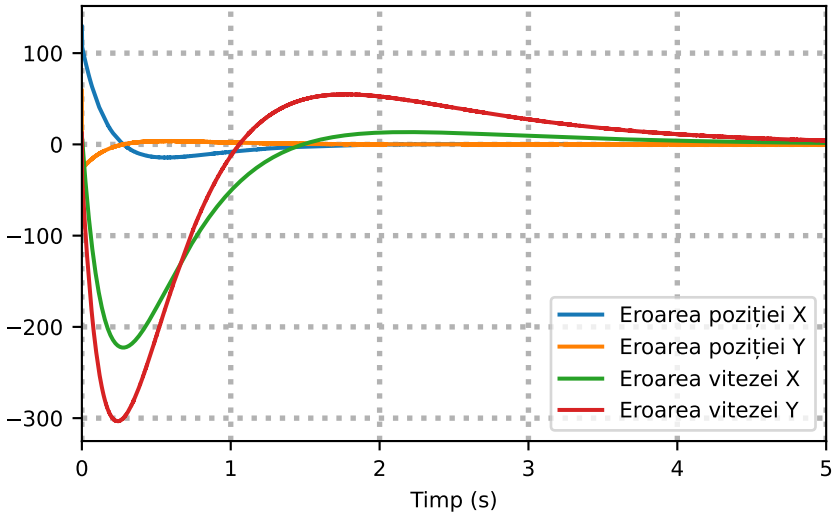


Figura 4.7: Convergenţa erorilor estimatorului Luenberger

4.3 Filtrul Kalman

Am considerat analizarea observabilităţii sistemului necesară oferirii unui context pentru problema estimării unei traiectorii. Cu toate acestea, algoritmii propuşi nu pot fi consideraţi soluţii de ultimă oră. Decuplarea perturbaţiilor funcţionează în cazul în care perturbaţia în sine nu este o intrare pentru model. în cazul procesului definit, dacă este prezent zgomot de măsură pentru acceleraţie, aceasta nu poate fi eliminată cu totul, pentru că variabilele de stare sunt dependente de ea.

Din acest motiv doresc să introduc în acest subcapitol motivarea, precum şi analiza filtrelor Kalman care au ajuns şi în implementarea finală descrisă în capitolul următor. Pentru aceasta voi realiza o trecere prin fundalul teoretic necesar ancorării unui astfel de algoritm.

4.3.1 Discretizarea procesului

Mişcarea unui obiect este de cele mai multe ori un proces continuu, cum este şi cazul care a motivat elaborarea acestei teze. Cu toate acestea, pentru sistemele automate, măsurătorile care vin de la senzori precum şi semnalele de comandă iniţiate de actuatoare sunt eşantionate pe momente de timp, şi au comportament discret, ceea ce înseamnă ca modelul spaţiului stărilor anterior definit se modifică.

Modelul liniar matricial pe care l-am dezvoltat pornind de la ecuaţiile (4.4) şi (4.5) poate fi integrat în timp folosind metode numerice precum Euler sau Runge Kutta [39] pentru aproximarea ecuaţiilor diferenţiale. Avantajul reprezentării propuse de Galileo-Newton este că sistemul definit de ecuaţiile (4.12) şi (4.13) este deja discretizat, sub o formă pe care Leonard Euler a redescoperit-o ulterior.

Astfel, se recuperează sistemul discret:

$$v_1 = v_0 + a\Delta t \tag{4.44}$$

$$p_1 = p_0 + v_0\Delta t + \frac{1}{2}a\Delta t^2 \tag{4.45}$$

pentru care perioada de observare în timp capătă noul sens de perioadă de eşantionare.

Spaţiul stărilor se rescrie pentru un moment de timp $t = k + 1$ sub forma:

$$\begin{bmatrix} p \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p \\ v \end{bmatrix}_k + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \cdot a \tag{4.46}$$

Care este extins pentru a acapara mişcarea în raport cu cele două axe ale planului traiectoriei [40]:

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \end{bmatrix}_k \tag{4.47}$$

Am definit astfel modelul procesului pentru care, în lipsa unui accelerometru, acceleraţia poate fi modelată ca zgomot, unde F devine matricea de tranziţie, iar L denotă matricea de propagare a zgomotului pe stările din proces:

$$x_{k+1} = Fx_k + La_k \tag{4.48}$$

Urmărirea traiectoriei unui obiect presupune în summum estimarea poziţiei corpului relativ la axele de mişcare. Din modelul procesului se poate vedea că o informaţie despre acceleraţii ar fi suficientă pentru a realiza obiectivul. Integrată o dată, măsuratoarea acceleraţiei devine o estimare pentru velocitate, care prin integrare reprezintă o ipotetică poziţie.

Deşi integrarea reduce zgomotul, un dezavantaj mare al procedurilor numerice îl reprezintă erorile de aproximare [39]. Cum nu se poate implementa analitic, o aproximare numerică de sine stătătoare va introduce dezechilibru, iar o dublă integrare va spori efectul. Acest fapt impune înglobarea comportamentului stocastic al nesiguranţelor din proces sub o formă predictibilă şi este punctul de plecare pentru analiza statistică ce urmează.

4.3.2 Probabilistica variabilelor aleatorii

Se defineşte probabilitatea unui eveniment x sub condiţia [41] :

$$0 \leq P(x) \leq 1 \tag{4.49}$$

astfel încât $P(x) = 0$ înseamnă că evenimentul nu va avea loc niciodată, iar pentru $P(x) = 1$ evenimentul va avea loc cu certitudine absolută.

Probabilitatea lui x poate fi calculată exact pentru o mulţime de evenimente n ce cuprinde totalitatea rezultatelor posibile [41]:

$$P(x) = \frac{100 \times n_x}{\sum n}[\%] \tag{4.50}$$

Pentru un set de evenimente X se defineşte probabilitatea obţinerii elementului x ca o funcţie cu restricţiile aferente [41]:

$$f(x) = P(X == x) \tag{4.51}$$
$$P(X == x) \geq 0 \tag{4.52}$$
$$\sum P(X == x) = 1 \tag{4.53}$$

Această funtie poartă numele de *probabilitate de masă* şi se aplică seturilor de evenimente discrete.

Cum însumarea unei muţimi de evenimente discrete infinitezimale ca mărime şi efect este rezultatul unei integrale, se defineşte pentru evenimente continue conceptul de *densitate probabilistică* ca o funcţie $f(X)$ se supune pentru un interval de observare dat $[a, b]$ următoarei egalităţi:

$$P(a \leq X \leq b) = \int_a^b f(X)dx \tag{4.54}$$
$$P(X == x) = 0 \tag{4.55}$$

Funcţia definită reprezintă distribuţia evenimentelor aleatorii din setul total de posibilităţi, iar intergrala ei aproximează aria de sub grafic.

În contextul modelării incertitudinii unei mărimi, măsurătorile senzorilui de la fiecare moment de timp k din numărul total N de experimente pot fi aproximate naiv prin mediere astfel:

$$\mu = \frac{1}{N} \sum_{k=1}^{N} x_k \tag{4.56}$$

Legea numerelor mari [42] observă pentru un număr de experimente $N \to \infty$ tendinţa mediei eşantioanelor de a converge la media adevărată a populaţiei. Figura 4.8 prezinţa rezultatul unei simulări de măsurători aleatorii cuprinse într-o distribuţie uniformă. Deşi media populaţiei (media teoretică) nu e neapărat o reprezentare bună pentru fiecare medie de eşantion, ea este cea mai bună reprezentare pentru media tutoror mediilor eşantioanelor.

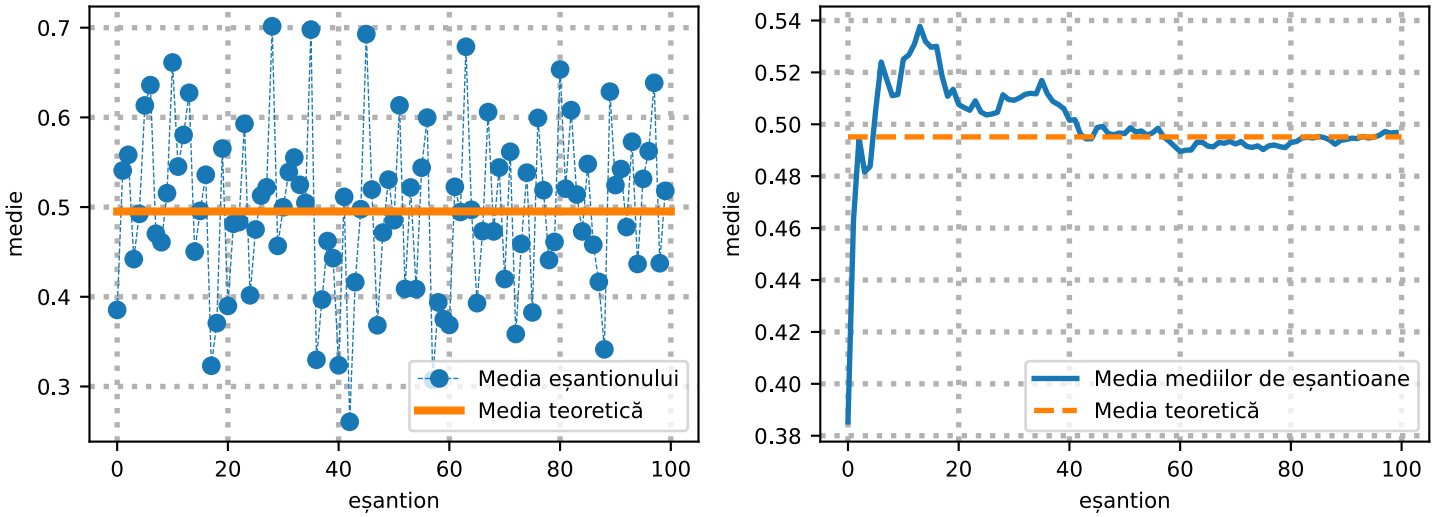


Figura 4.8: Legea numerelor mari

Chiar mai relevantă pentru topicul subiectului este implicaţia care se desprinde din ideea enunţată anterior. Teorema limitei centrale [43] implică pentru acelaşi număr de experiente $N \to \infty$ o distribuţie normală a mediilor eşantionelor.

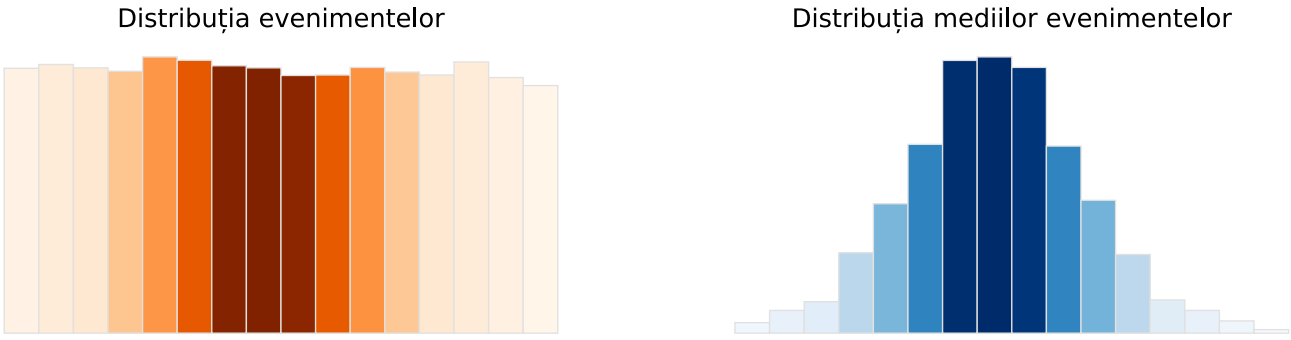


Figura 4.9: Teorema limitei centrale

Histogramele experimentului măsurătorilor zgomotoase din Figura 4.9 demonstrează ca pentru orice tip de distribuţie, distribuţia mediilor evenimentelor va fi mereu normală.

Această arie în formă de clopot a fost una dintre contribuţiile matematicianului Carl Friedrich Gauss şi stă la baza majorităţii algorimilor statistici, şi constituie implicit şi un detaliu teoretic necesar implementării lucrării propuse.

Omniprezenţa distribuţiei Gaussiene în estimarea statistică se remarcă prin proprietăţi ale funcţiei de densitate probabilistică ce devin parametri utili în descrierea fenomenelor normal-aleatorii.

4.3.3 Dispersia şi tendinţa centrală

Gauss a definit astfel funcţia de densitate probabilistică [44] pentru abaterea standard σ şi media μ sub forma [1]:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{4.57}$$

Se defineşte astfel *valoarea aşteptată* ca o măsuratoare a tendiţei centrale şi se calculează în funcţie de probabilitatea unui eviniment [1]:

$$E[X] = \sum_{k=1}^{N} x_k P(x_k) \tag{4.58}$$

Pentru evenimente la fel de probabile (improbabile), cu proprietatea $P(x)_{\forall x \in N} = \frac{1}{N}$ se observă că:

$$E[X] = \mu \tag{4.59}$$

unde media μ capătă conotaţia de prim moment statistic al distribuţiei anterior definite.

O altă prprietate importantă a unei distribuţii este cât de mult variază distribuţia relativ la valoarea aşteptată. Astfel, s-a definit *varianţa* ca al doilea moment statistic [45]:

$$\sigma^2{}_X = E[(X - \mu)^2] = \frac{1}{N} \sum_{k=1}^{N} (x_k - \mu)^2 \tag{4.60}$$

Tot ca o măsurătoare a dispersiei se defineşte abaterea standard ca rădăcina pătrată a varianţei:

$$\sigma_x = \sqrt{\sigma^2{}_X} \tag{4.61}$$

Se defineşte *covarianţa* pentru două seturi de variabile ca produs al dispersiilor [45]:

$$Cov(X, Y) = E[X, Y] - \mu_x \mu_y \tag{4.62}$$

unde pentru seturi de evenimente independente X şi Y există relaţia [45]:

$$Cov(X, Y) = 0 \tag{4.63}$$

Noţiunile statistice introduse concretizează acum un început pentru modalitatea de rezolvarea a obiectivului propus la începutul lucrăii.

În contextul urmăririi traiectoriei, stocasticitatea zgomotului prezent în măsurătorile senzorilor de mişcare poate fi înglobată într-o distribuţie normală şi se pot trage următoarele concluzii empirice:

$$P(\mu - \sigma \leq X \leq \mu + \sigma) = 0.68 \tag{4.64}$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) = 0.95 \tag{4.65}$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) = 0.99 \tag{4.66}$$

Astfel, pentru orice măsurătoare zgomotoasă de la un senzor se poate spune cu certidine aproape absolută că există probabilitatea de 99% să fie încadrată în trei deviaţii standard de la media tuturor măsurătorilor precedente.

Această demonstraţie statistică a fost demonstrată programatic pentru o funcţie de densitate probabilistică analitică după cum se observă în figura 4.10. Înglobarea probabilistică a comportamentului evenimentelor stocastice este şi motivul pentru care senzorii, deşi prezintă măsurători afectate de zgomot, produc un randament bun atunci când algoritmii de fuziune sunt programaţi corect.

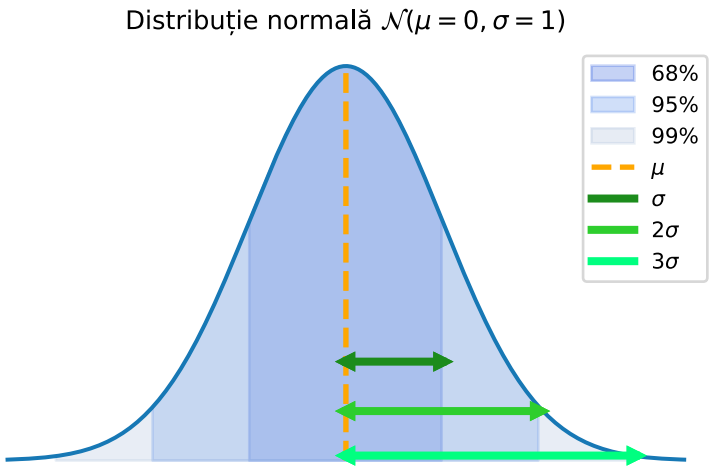


Figura 4.10: Proprietăţiile distribuţiei Gaussiene

4.3.4 Predicţie

Transformarea liniară a funcţiilor de densitate probabilistică

Demonstraţie
Se consideră setul de date distribuit normal:

$$x \sim \mathcal{N}(\mu = \bar{x}, \sigma^2 = {\sigma_x}^2) \tag{4.67}$$

şi transformarea liniară:

$$y(x) = ax + b \tag{4.68}$$
$$y = z(x) \tag{4.69}$$

aplicată mediei:

$$y(\mu) = a\mu + b \tag{4.70}$$

Se defineşte pentru setul X funcţia de densitate probabilistică rezultată din ecuaţia (4.57). Se recuperează variabila x [46] din ecuaţiile (4.68), (4.69):

$$f(x) = f(x, \mu, \sigma_x) \tag{4.71}$$
$$x = z(y)^{-1} = Z(y) = \frac{y-b}{a} \tag{4.72}$$

Se transformă liniar [46] prin y funcţia de densitate probabilistică f pentru a recupera probabilitatea conform ecuaţiei (4.54):

$$f(y) = f[Z(y)]Z'(y) \tag{4.73}$$
$$Z'(y) = \frac{\partial Z}{\partial y}(\frac{y-b}{a}) = \frac{1}{a} \tag{4.74}$$

Se recalculează funcţia de densitate probabilistică [46] ţinând cont de (4.70) - (4.74):

$$f(y, \mu, \sigma_y) = \frac{1}{a}\frac{1}{\sigma_x\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{(\frac{y-b}{a})-a\mu-b}{\sigma_x})^2} \tag{4.75}$$

De unde se poate trage concluzia:

$$\sigma_y = a\sigma_x \tag{4.76}$$

Aşadar, se observă că transformarea liniară a unei distribuţii normale:

$$y \sim \mathcal{N}(\bar{y} = a\mu + b, \sigma_y = a^2 {\sigma_x}^2) \tag{4.77}$$

este tot o distribuţie normală cu media şi varianţa schimbate.

Propagarea Covarianţei

Conform (4.62) - (4.77), pentru un vector de stări X [1]:

$$Cov(X) = E[X, X^T] - \mu^2 x^2 \tag{4.78}$$

Astfel , rezultă [45] [47]:

$$Cov(X) = E[(X - \mu)(X - \mu)^T] \tag{4.79}$$

unde se respectă (4.48) pentru X:

$$X = FX + La \tag{4.80}$$

Astfel, ţinând cont de transformarea liniară (4.70):

$$Cov(X) = E[(FX + La - F\mu - La)(FX + La - F\mu + La)^T] \tag{4.81}$$

$$Cov(X) = E[F(X - \mu)(X - \mu)^T F^T] \tag{4.82}$$

Folosind relaţia de transpunere a matricilor [47], rezultă:

$$Cov(X) = FE[(X - \mu)(X - \mu)^T]F^T \tag{4.83}$$

Se notează $Cov(X) = P$ şi se desprinde concluzia pentru evoluţia liniară a incertitudinii în timp [47]:

$$P_{k+1} = FP_kF^T \tag{4.84}$$

Această relaţie denotă cea de-a doua ecuaţie din algoritmul elaborat de Kalman şi ea reprezintă propagarea în timp a matricei de covarianţa aferentă vectorului de stări. Echivalentul unei transformări liniare (4.68) asupra unei variabile echivalează cu o înmulţire matricială pentru un vector de variabile.

Pentru că procesul modelat nu este echivalent în deplinătate cu realitatea, varianţa zgomotului de proces generat de incertitudinea intrării (acceleraţiei) este aditivă pentru covarianţa stărilor şi astfel se obţine ecuaţia completă [1]:

$$P_{k+1} = FP_kF^T + L{\sigma_a}^2 \tag{4.85}$$

4.3.5 Corecţie

Modelul măsuratorilor

Se defineşte pentru modelul ales în (4.47) - (4.48) ieşirea observată a procesului:

$$y_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k+1} \tag{4.86}$$

astfel încât se respectă obiectivul estimării poziţiei corpului relativ la axele planului. Comprimat, modelul măsurătorii se notează:

$$y_{k+1} = Hx_{k+1} \tag{4.87}$$

unde H este invariantă în timp şi propagă la ieşire o transformare liniară prin selecţia primelor două stări.

Cum în realitate măsurătorile provin de la senzori cu zgomot prezent, se adaugă termenul de *bias* ce variază în timp pentru fiecare măsurătoare:

$$y_{k+1} = Hx_{k+1} + \omega_{k+1} \tag{4.88}$$

pentru ca proprietăţile probabilistice de predicţie expuse anterior să fie aplicabile, se presupune că ω provine dintr-o distribuţie normală cu media zero (0):

$$\omega \sim \mathcal{N}(\mu = 0, \sigma^2) \tag{4.89}$$

ce reprezintă o clasă particulară de zgomote, numite zgomote albe Gaussiene.

Regresie

Se reconsideră transformarea liniară introdusă în (4.68), rescrisă vectorial:

$$y(x) = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{4.90}$$

Se generalieză pentru a include zgomotul de măsură:

$$y(x) = \begin{bmatrix} a & b & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \\ \omega \end{bmatrix} \tag{4.91}$$

$$y(X) = \Theta X \tag{4.92}$$

unde Θ reprezintă vectorul de proprietăţi al dreptei — panta, interceptul [48] şi prezenţa zgomotului, iar X este vectorul de stări. Am atribuit zgomotul măsurătorilor, şi

nu ecuaţiei liniare în sine.

Justificarea zgomotului alb

Se ia în considerare cazul particular al unei constante ($a = 0$) astfel încăt estimările sunt perturbate doar de zgomotul ω:

$$y(x)_k = x + \omega_k \tag{4.93}$$

S-a demonstrat (Figura 4.8) că media măsurătorilor este probabilistic mai reprezentativă pentru proces decât oricare măsurătoare în parte. Astfel încât se evaluează:

$$\mu_y = \frac{1}{N}\sum_{k=1}^{N} y(x) = \frac{1}{N}\sum_{k=1}^{N}(x + \omega_k) \tag{4.94}$$

Deoarece x este constant, se deduce:

$$\mu_y = E[x] + E[\omega] \tag{4.95}$$

$$E[X] = \mu_x = x \tag{4.96}$$

Cum cea mai bună estimare pentru seria de măsurători zgomotoase este chiar media μ_y, reiese că trebuie să aibă loc neapărat egalitatea:

$$E(\omega) = 0 \tag{4.97}$$

Această asumpţie va fi făcută de acum pentru restul lucrării.

Minimizarea erorilor

Pentru cazul general în care măsurătorile observă o schimbare în dinamică, panta dreptei estimate este nenulă ($a \neq 0$), şi eroarea de estimare dată de zgomot este [49]:

$$\epsilon = y(X) - \Theta X \tag{4.98}$$

Se defineşte funcţia de cost C [1] ca sumă a erorilor fiecărei măsurători [48]:

$$C = \sum_{k=1}^{N} {\epsilon^2}_k \tag{4.99}$$

$$C = \mathcal{E}\mathcal{E}^T \tag{4.100}$$

pentru care pătratul erorilor este evaluat pentru a anula funcţia de semn.

Pentru cele mai bune estimări trebuie ca funcţia de cost să fie minimizaţa [48], iar minimul poate fi calculat ca punct extrem prin evaluarea derivatei funtiei în 0 [50]:

$$\frac{\partial C}{\partial \hat{X}} = \frac{[\partial(y(X) - \Theta X)(y(X) - \Theta X)^T]}{\partial \hat{X}} \tag{4.101}$$

Şi astfel se obţine cea mai bună estimare [50] [48] [49]:

$$\hat{X} = (\Theta^T \Theta)^{-1} \Theta^T y(X) \tag{4.102}$$

Ponderare

Deşi valoarea aşteptată a zgomotelor de măsură este 0, variantă lor nu este neapărat invariantă în timp. În acest sens, erorile vor fi calculate în raport de cu varianţa [51], iar (4.99) devine:

$$C = \sum_{k=1}^{N} \frac{\epsilon^2{}_k}{\sigma_k^2} \tag{4.103}$$

$$C = \mathcal{E} \Sigma \mathcal{E}^T \tag{4.104}$$

Ceea ce impune modificarea pentru (4.102) conform [51]:

$$\hat{X} = (\Theta^T \Sigma^{-1} \Theta)^{-1} \Theta^T \Sigma^{-1} y(X) \tag{4.105}$$

Recursivitate

Atât pentru optimizarea performanţei, cât şi pentru îmbunătăţirea acurateţei, estimarea punctului următor poate fi rescrisă pentru a păstra atât informaţia estimării de la punctul curent, cât şi corecţia erorii din prezent. Astfel, cea mai bună estimare se rescrie recursiv [1]:

$$\hat{X}_{k+1} = \hat{X}_k + K_{k+1}(y_k(X_k) - \Theta \hat{X}_k) \tag{4.106}$$

unde K reprezintă un factor de corecţie al erorrii (inovaţiei) [52], cunoscut în literatură sub denumirea de *căştig Kalman*.

Căştigul K poate fi calculat conform [52] ca raportul dintre varianţa actuală a procesului şi varianţa modificată prin introducere zgomotului de măsură. Astfel, pentru covarianţa obţinută în (4.84), (4.85), câştigul care minimizează înovaţia devine:

$$K_{k+1} = P_{k+1}^- \Theta (\Theta P_{k+1}^- \Theta^T + \Sigma)^{-1} \tag{4.107}$$

$$S_{k+1} = \Theta P_{k+1}^- \Theta^T + \Sigma \tag{4.108}$$

unde S este notaţia simplificată a varianţei inovaţiei.

Calculată recursiv conform [53] [1], eroarea de la punctul următor se exprimă ca:

$$\mathcal{E}_{k+1} = X_{k+1} - \hat{X}_{k+1} \tag{4.109}$$
$$\mathcal{E}_{k+1} = X_k - \hat{X}_k - K_{k+1}(y_k(X_k) - \Theta\hat{X}_k) \tag{4.110}$$
$$\mathcal{E}_{k+1} = \mathcal{E}_k - K_{k+1}(\Theta X_{k+1} - \Theta\hat{X}_k) \tag{4.111}$$
$$\mathcal{E}_{k+1} = (I - K_{k+1}\Theta)\mathcal{E}_k \tag{4.112}$$

Am obţinut astfel noua matrice de propagare în timp a covarianţei [53]:

$$P^+_{k+1} = (I - K_{k+1}\Theta)P^-_{k+1} \tag{4.113}$$

şi astfel au fost recuperaţi toţi paşii algoritmului propus de R. E. Kalman.

Implementarea pe paşi a fiecărei ecuaţii va fi descrisă în capitolul următor pentru estimarea traiectoriei liniare, iar rezultatelor vor fi validate în capitolul 6.

4.4 Mişcarea Neliniară

Filtrul Kalman este din punct de vedere optim-algebric cel mai bun estimator pentru orice sistem liniar. Privind obiectivul realist însă, obiectele de interes nu se mişcă întotdeauna liniar, aşa cum nici majoritatea proceselor reale nu au un comportament liniar în afara teoriei.

Apare astfel nevoia modelării neliniarităţilor din proces şi măsurători. Traiectoria mişcării neliniare se schimbă sub un unghi B şi modelul procesului se modifică implicit. Acest subcapitol va prezenta modelarea neliniară a dinamicii sistemului şi va motiva folosirea filtrelor Kalman extins şi unscented, a căror funtionalitate va fi demonstrată în următoarele două capitole.

4.4.1 Dinamica procesului extins

Se consideră dinamica discretă a schimbării unghiului în plan obţinută prin integrare numerică Euler astfel [19]:

$$\beta_{k+1} = \beta_k + \dot{\beta}\Delta t \tag{4.114}$$

unde $\dot{\beta}$ este viteza unghiulară măsurabilă de un senzor de tip giroscop.

care schimbă vectorial vitezele relativ la axe [54]:

$$v_x = v \times \cos\beta \tag{4.115}$$
$$v_y = v \times \sin\beta \tag{4.116}$$

Astfel, noul proces devine [3] [54]:

$$\begin{bmatrix} p_x \\ p_y \\ \beta \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} p_x \\ p_y \\ \beta \\ v \end{bmatrix}_{k} + \Delta t \cdot \begin{bmatrix} v \times \cos\beta_k \\ v \times \sin\beta_k \\ \dot{\beta}_k \\ a_k \end{bmatrix} \tag{4.117}$$

astfel încât acum se urmăreşte şi schimbarea unghiului în timp.

Noul proces nu poate fi trecut sub formă matricială liniară deoarece există o interdependeţă neliniară de produs dintre stări pentru cele două funcţii trigonometrice. Acesta se poate rescrie simplificat [3]:

$$x_{k+1} = f(x_k, a_k, \dot{\beta}_k, \nu_k) \tag{4.118}$$

unde f este o funcţie neliniară ce descrie modelul (4.117) care are în componenţa vectorului de stări acceleraţia a şi viteza unghiulara $\dot{\beta}$, precum şi zgomotul alb de proces $\nu \sim \mathcal{N}(0, \sigma^2)$.

Pentru ca filtrul Kalman să poată fi aplicat, aşadar, este nevoie ca procesul să fie liniarizat.

4.4.2 Liniarizare

Se reconsideră ecuaţia unei drepte enunţată în (4.68). Aceasta poate fi rescrisă ţinând cont de proprietate derivatelor de a descrie panta unei drepte în punctul evaluat:

$$y(x) = f'(x)x + f(x) \tag{4.119}$$

unde f este o funcţie neliniară care evaluată în x echivalează cu o valoare de $defazaj$ (intercept).

Datorită proprietăţiilor derivatei, este important de notat că această aproximare va fi valabilă doar pentru anumite puncte de funcţionare x_0 astfel [48]:

$$y(x) = f'(x_0)(x - x_0) + f(x_0) \tag{4.120}$$

Aplicată funcţiei obţinută în (4.118) pentru un punct de echilibru $(x_0, a_0, \dot{\beta}_0, \nu_0)$, transformarea liniară devine:

$$f(x_k, a_k, \dot{\beta}_k, \nu_k) = \nabla f(x_0, a_0, \dot{\beta}_0, \nu_0)[X_k - X_0] + f(x_0, a_0, \dot{\beta}_0, \nu_0) \tag{4.121}$$

$$[X_k - X_0] = [(x_k, a_k, \dot{\beta}_k, \nu_k) - (x_0, a_0, \dot{\beta}_0, \nu_0)] \tag{4.122}$$

Această expansiune este un caz particular al seriei Taylor [48], folosită pentru a aproxima comportamentul neliniar al unei funcţii în timp ţinând cont şi de termenii de ordin superior (T.O.S)

$$f(x) = f'(x_0)(x - x_0) + f(x_0) + T.O.S \tag{4.123}$$

Se liniarizează în acest fel procesul descris în (4.117) şi se obţin [11]:

$$p_x = \frac{\partial p_x}{\partial p_x}(x - x_0) + \frac{\partial p_x}{\partial p_y}(x - x_0) + \frac{\partial p_x}{\partial \beta}(x - x_0) + \frac{\partial p_x}{\partial v}(x - x_0) + p_x(x_0) \tag{4.124}$$

$$p_y = \frac{\partial p_y}{\partial p_x}(x - x_0) + \frac{\partial p_y}{\partial p_y}(x - x_0) + \frac{\partial p_y}{\partial \beta}(x - x_0) + \frac{\partial p_y}{\partial v}(x - x_0) + p_y(x_0) \tag{4.125}$$

$$\beta = \frac{\partial \beta}{\partial p_x}(x - x_0) + \frac{\partial \beta}{\partial p_y}(x - x_0) + \frac{\partial \beta}{\partial \beta}(x - x_0) + \frac{\partial \beta}{\partial v}(x - x_0) + \beta(x_0) \tag{4.126}$$

$$v = \frac{\partial v}{\partial p_x}(x - x_0) + \frac{\partial v}{\partial p_y}(x - x_0) + \frac{\partial v}{\partial \beta}(x - x_0) + \frac{\partial v}{\partial v}(x - x_0) + v(x_0) \tag{4.127}$$

Astfel, matricea F care propaga liniar procesul iniţial introdus în (4.48) devine:

$$\nabla f = \begin{bmatrix} 1 & 0 & -\Delta t \times v \times \sin\beta & \Delta t \times \cos\beta \\ 0 & 1 & \Delta t \times v \times \cos\beta & \Delta t \times \sin\beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.128}$$

pentru care se poate rescrie propagarea liniarizată a covarianţei introdusă în (4.85):

$$P_{k+1} = \nabla f P_k \nabla f^T + \nabla N Q \nabla N \tag{4.129}$$

$$Q = (0, 0, \sigma_{\dot{\beta}}, \sigma_a)^2 \tag{4.130}$$

unde $\nabla N = I_4$, matricea identitate, pentru zgomotul considerat aditiv care este considerat prezent numai pentru acceleraţie şi viteza unghiulară. Noul sgomot de proces Q este diferit de cel înitia considerat L (4.85), deoarece aceste acum înglobează şi nesiguranţa din estimarea giraţiei, şi nu numai varianţa acceleraţiei.

Modelul măsurătorilor primite pentru poziţii prin matricea de selecţie H enunţaţa în (4.86) , (4.87) poate fi considerat liniar pentru simplificare. Astfel, ca pentru un fel de senzor de GPS constrâns la planul bidimensional, distanţa dintre două puncte (mă-

surătoarea curentă şi măsurătoarea anteriorară) va fi mereu o dreaptă, deci corecţia este implicit liniară. De notat este însă că în realitatea spaţiului tridimensional, senzorii de tip GPS se modelează de obicei nelinar, tot sub un unghi [55].

4.4.3 Sonar

Expansiunea neliniară a filtrului permite şi integrarea măsurătorilor unui senzor cu comportamnet neliniar. O nouă fază de corecţie în plus faţă de măsurătorile liniare ale poziţiei este adaugată pentru procesul extins.

Astfel, măsurarea poziţiei se poate face şi în coordonate polare [56], în afara măsurătorilor carteziene procesate liniar. Corecţia neliniară adăugată va îmbunătăţii considerabil rezultatele după cum se va demonstra în următoarele două capitole.

Modelul neliniar al măsurătorilor devine [56] [57] [58]:

$$\gamma = \arctan \frac{O_y - \hat{p_y}}{O_x - \hat{p_x}} - \hat{\beta} \tag{4.131}$$

$$r = \sqrt{(O_y - \hat{p_y})^2 + (O_x - \hat{p_x})^2} \tag{4.132}$$

unde O este un punct de observare static, definit de coordonatele specifice x şi y, iar γ şi r sunt reprezentarea polară a poziţiei în plan, relative la unghiul de giraţie β.

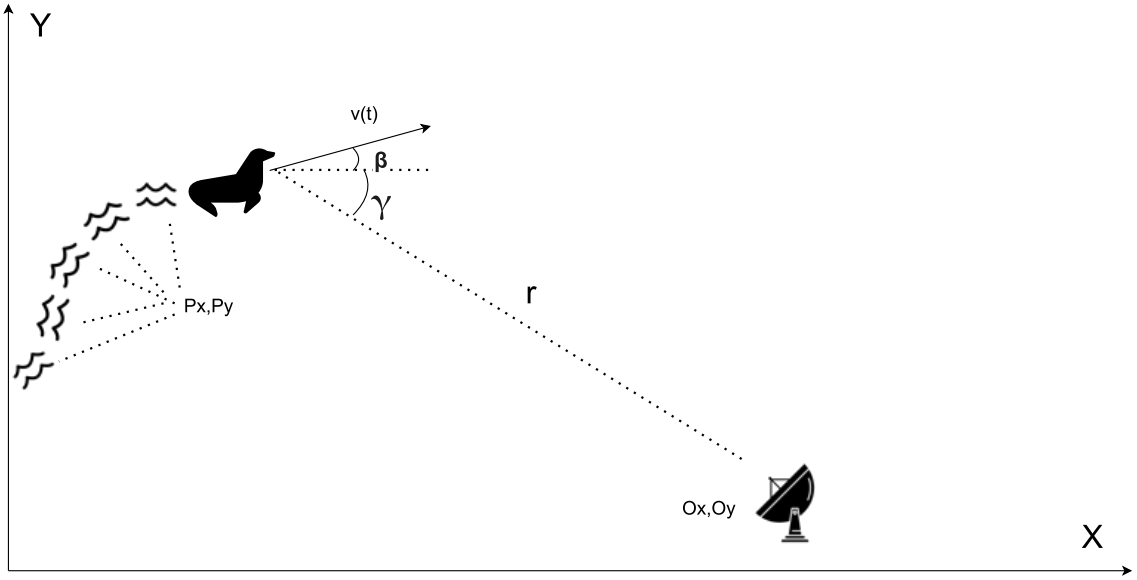


Figura 4.11: Reprezentarea mişcării în plan sub observare

Figura 4.11 prezintă geometric schema de urmărire a coordonatelor polare oferite de sonar şi a fost inspirată din [54] şi [58]. Poziţiile Px şi Py pot fi estimate recursiv printr-o

regresie liniară, iar pentru estimarea razei r şi unghiului γ se concepe o nouă etapă de corecţie extinsă.

Se liniarizează ecuaţiile (4.130) şi (4.31) în raport cu stările modelului (px, py, v, β) prin aproximarea Taylor de ordin I (4.123):

$$r = \frac{\partial r}{\partial p_x}(x - x_0) + \frac{\partial r}{\partial p_y}(x - x_0) + \frac{\partial r}{\partial \beta}(x - x_0) + \frac{\partial r}{\partial v}(x - x_0) + r(x_0) \tag{4.133}$$

$$\gamma = \frac{\partial \gamma}{\partial p_x}(x - x_0) + \frac{\partial \gamma}{\partial p_y}(x - x_0) + \frac{\partial \gamma}{\partial \beta}(x - x_0) + \frac{\partial \gamma}{\partial v}(x - x_0) + \gamma(x_0) \tag{4.134}$$

$$\tag{4.135}$$

de unde se obţine jacobianul matricei de selecţie [58]:

$$\nabla h = \begin{bmatrix} \frac{\hat{p_x} - O_x}{\sqrt{(O_y - \hat{p_y})^2 + (O_x - \hat{p_x})^2}} & \frac{\hat{p_y} - O_y}{\sqrt{(O_y - \hat{p_y})^2 + (O_x - \hat{p_x})^2}} & 0 & 0 \\ \frac{-\hat{p_y} - O_y}{(O_x - \hat{p_x})^2 + (O_x - \hat{p_x})^2} & \frac{\hat{p_x} - O_x}{(O_y - \hat{p_y})^2 + (O_x - \hat{p_x})^2} & -1 & 0 \end{bmatrix} \tag{4.136}$$

pentru care se pot rescrie ecuaţiile etapei de corecţie (4.106) - (4.113) astfel încât se face modificarea $\Theta \to \nabla h$.

Pentru transformarea neliniară $h(X)$ (4.131) - (4.132), se recuperează ecuaţiile:

$$S_k = \nabla h P_k \nabla h^T + \nabla M R \nabla M \tag{4.137}$$

$$K_{k+1} = P_{k+1}^- \nabla h {S_{k+1}}^{-1} \tag{4.138}$$

$$\hat{X}_{k+1} = \hat{X}_k + K_{k+1}(h(X_k) - h(\hat{X}_k)) \tag{4.139}$$

$$P_{k+1}^+ = (I - K_{k+1} \nabla h) P_{k+1}^- \tag{4.140}$$

$$\tag{4.141}$$

La fel ca în cazul predicţiei, zgomotul de măsură se consideră aditiv, deci se poate simplifica calcularea covarianţei inovaţiei prin consideraţia $\nabla M = I_2$.

Diferenţa demnă de notat pentru aceeaşi ecuaţie este că zgomotul R al corecţiei neliniare este diferit de zgomotul Σ iniţial considerat pentru procesul liniar ($R \neq \Sigma$), deoarece senzorul de tip sonar este diferit de senzorul de tip GPS şi este de aşteptat să aibă variaţiuni şi performanţe diferite.

Modificările de liniarizare aduse celor două etape ale algoritmului însumează ecuaţiile unui nou filtru, cunoscut drept filtrul Kalman *extins*. Întrucât se bazează pe aproximări infinitezimale, procedura extinsă nu poate garanta convergentă pentru estimare, iar rezultatele pot fi divergente. Dezavantajul este suficient de prezent pentru a motiva explorarea unei noi implementări şi mai avansate a filtrului neliniar.

4.4.4 Transformata unscented

Noul filtru sau de ce copacii se numesc copaci

La finele mileniului trecut, Jeffrey Uhlman, unul dintre cei doi autori ai algoritmului neliniar modificat, se întâneşte la Oxford cu Hugh Durrant-Whyte, unul dintre notabilii pionieri ai algoritmilor de tip SLAM (Simultaneous Localization and Mapping) [59] pentru a punele bazele a ceea ce urma să devină un punct cheie în localizarea corpurilor în conducerea autonomă.

În interviul [60] acordat site-ului Engineering and Technology History Wiki, Uhlman povesteşte cum liniarizarea sistemului prin aproximare jacobiană nu adresează rădăcina problemei pentru că, de fapt, ceea ce contează pentru un filtru Kalman sunt transformările celor două momente ale distribuţiei, media şi (co)varianţa. El propune astfel o mapare determinstă a acestora, fără a impune calcularea prealabilă a derivatelor ecuaţiilor sistemului.

Explicaţia autorului pentru denumirea neobişnuită a noului filtru este că *termenii tehnici sunt acceptaţi de public pur şi simplu ca atare, aşa cum niciun om nu se întrebă de ce copacii se numesc copaci şi nu altfel* [60].

Premisă

Încadrarea distribuţiei unui set de măsurători este, aşa cum demonstrează faza de predicţie a filtrului Kalman (4.84) - (4.85), o transformare liniară. Pentru o distribuţie neliniară, filtrul Kalman extins va încerca de fapt tot o aproximare liniară a distribuţiei, ceea ce este, după cum a remarcat Uhlman, adevărată cauză a divergenţelor. Am generat figura 4.12 modificând codul pus la dispoziţie de [61] pentru a arăta diferenţa dintre o încadrare Gaussiană a unui set de date liniar în constrast cu unul neliniar.

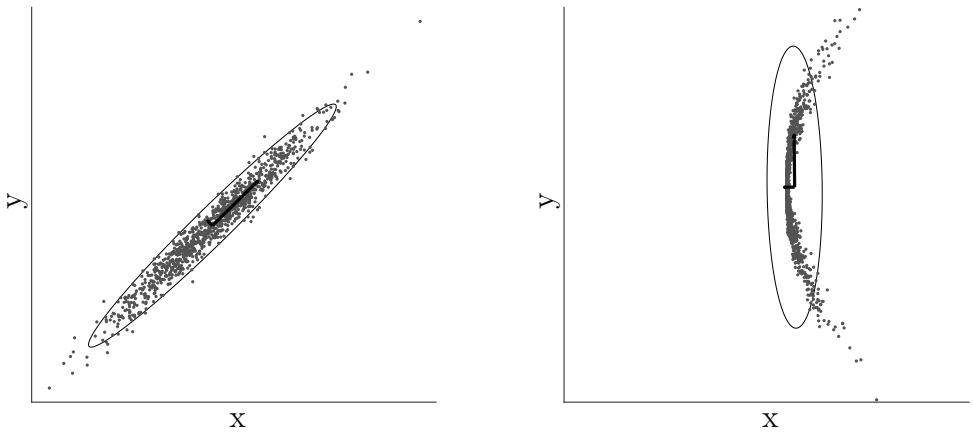


Figura 4.12: Comparaţie între încadrarea datelor liniare şi neliniare în elipse de covarianţă

Se observă în a doua figură cum elipsa de încredere nu este o aproximare ideală pentru distribuţia reală, aşa cum nici derivata (panta) unei funcţii nu este mereu o aproximare bună pentru acea funcţie.

În schimb, transformata unscented adresează cele două momente ale distribuţiei astfel [9]:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.142}$$

$$\Sigma_X = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{X})(x_i - \bar{X})^T \tag{4.143}$$

unde $\bar{X}$ reprezinţa media datelor şi $\Sigma_X = P$ covarianţa.

O aproximare mai bună decât medierea naivă este aproximarea pe setul dublat ($N \to 2N$) definită prin *puncte* σ ce stă la baza transformatei unscented [9]:

$$x = \bar{x} + x^{\sigma} \tag{4.144}$$

$$x^{\sigma_i} = \sqrt{NP_i} \tag{4.145}$$

$$x^{\sigma_{N+i}} = \sqrt{NP_i} \tag{4.146}$$

$$W = \frac{N}{2} \tag{4.147}$$

unde W defineşte un set de ponderi.

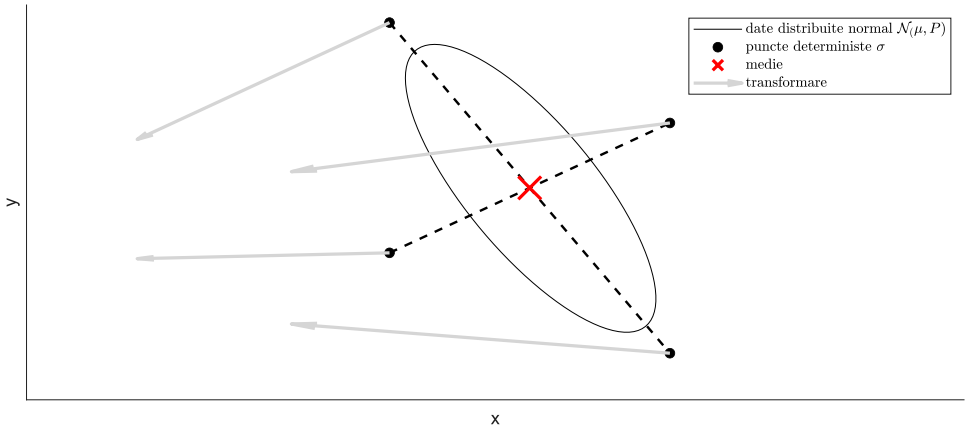


Figura 4.13: Forma de interes a distribuţiei obţinută print-un set minim de 4 puncte

Algoritm

Figura 4.13 arată cum elipsa poate fi aproximată de două puncte şi de oglinditele lor. De aici apare şi motivaţia dublării setului pentru ponderare. Transformata poate fi generalizată pentru a îngloba cunoştinţe a priori despre distribuţie [10]:

$$x = \bar{x} + x^{\sigma} \tag{4.148}$$
$$x^0 = 0 \tag{4.149}$$
$$x^{\sigma_i} = \sqrt{(N+\kappa)P} \tag{4.150}$$
$$x^{\sigma_{N+i}} = \sqrt{(N+\kappa)P} \tag{4.151}$$

pentru care se definesc ponderile unui κ ales de obicei empiric $\kappa = 3 - N$[10]:

$$W^0 = \frac{\kappa}{\kappa + N} \tag{4.152}$$
$$W^i = \frac{1}{2(\kappa + N)} \tag{4.153}$$

Astfel, se recuperează faza de predicţie [10] [9]:

$$\hat{X}_k^- = \sum_{i=0}^{2N} W^i \hat{X}_k^i \tag{4.154}$$
$$P_k^- = \sum_{i=0}^{2N} W^i (\hat{X}_k^i - \hat{X}_k^-)(\hat{X}_k^i - \hat{X}_k^-)^T \tag{4.155}$$

precum şi faza de corectîe pentru modelul măsurătorilor neliniar (4.141 - 4.132) [10]:

$$\hat{z}_k = \sum_{i=0}^{2N} W^i \hat{z}_k^i \tag{4.156}$$
$$S_k = \sum_{i=0}^{2N} W^i (\hat{z}_k^i - \hat{z}_k^-)^T (\hat{z}_k^i - \hat{z}_k^-) \tag{4.157}$$
$$P_{x_k,z_k}^+ = \sum_{i=0}^{2N} W^i (\hat{X}_k^i - \hat{X}_k^-)^T (\hat{z}_k^i - \hat{z}_k^-) \tag{4.158}$$

unde P_{x_k,z_k}^+ este *cros-covarianţa* trasnformată pentru stări - măsurători, iar S_k este covarianţa ponderată pentru modelul măsurătorilor z_k [9].

În continuare câştigul Kalman se defineşte ca raport dintre varianţe [10]:

$$K_k = P^+_{x_k,z_k} S_k^{-1} \tag{4.159}$$

iar actualizarea stării şi a covarianţei devine [10]:

$$\hat{X}_k^+ = \hat{X}_k^- + K_k(h(X_k) - \hat{z}_k) \tag{4.160}$$

$$P_k^+ = P_k^- - K_k S_k {K_k}^T \tag{4.161}$$

şi astfel se va realiza implementarea ulterioară a filtrului Kalman unscented.

4.5 Controlul urmăritorului

Pentru obiectul în mişcare au fost implementaţi cei trei algoritmi de urmărire în funcţie de scenariul ales. Pentru o mişcare liniară, filtrul Kalman simplu este de preferat întrucât este garantat să conveargă. O traiectorie neliniară creează o concurenţă între modelul extins şi transformata unscented a filtrului şi va fi elaborată în capitolul 5, pentru ca mai apoi rezultatele să fie validate în capitolul 6.

Dinamica obictului urmăritor a fost considerată liniară pentru a putea aplica legi de control simplificate. Astfel, aproape identică cu modelul procesului filtrului liniar, s-a definit ecuaţia matricială a spaţiului stărilor:

$$X^c_{k+1} = FX^c_k + Ga_k \tag{4.162}$$

unde matricea de tranziţie şi matricea de control descriu:

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{4.163}$$

Pentru a contura un scenariu realist, direcţia de atat a obiectului urmăritor se bazează pe estimările filtrului, fie liniar, extins sau unscented. Astfel, pentru estimările liniare se dfineşte vectorul de eroare:

$$\mathcal{E}^c_{k+1} = X^c_{k+1} - \hat{X}_{k+1} \tag{4.164}$$

unde $\hat{X}_{k+1}$ reprezintă estimările filtrului Kalman liniar.

În cazuk celor două filtre neliniare, deoarece modelul procesului obiectului urmărit (4.117) diferă print vectorul de stare de urmăritor, erorile trebuie recuperate explicit:

$$\epsilon_{k+1}^{p_x} = p_{k+1}^{x_c} - \hat{p}_{k+1}^{x} \tag{4.165}$$
$$\epsilon_{k+1}^{p_y} = p_{k+1}^{y_c} - \hat{p}_{k+1}^{y} \tag{4.166}$$
$$\epsilon_{k+1}^{v_x} = v_{k+1}^{x_c} - \hat{v}_{k+1} \times \cos\hat{\beta}_{k+1} \tag{4.167}$$
$$\epsilon_{k+1}^{v_y} = v_{k+1}^{x_c} - \hat{v}_{k+1} \times \sin\hat{\beta}_{k+1} \tag{4.168}$$

pentru care se recompune vectorul de erori:

$$\mathcal{E}_{k+1} = \begin{bmatrix} \epsilon_{k+1}^{p_x} \\ \epsilon_{k+1}^{p_y} \\ \epsilon_{k+1}^{v_x} \\ \epsilon_{k+1}^{v_y} \end{bmatrix} \tag{4.169}$$

Astfel, au fost implementate două câştiguri pentru intrare, fie prin alocare de poli sau reglare liniar-pătratică discretă (LQR) cu ajutorul funcţiilor MATLAB *place* [62] şi *dlqr* [63]:

$$\eta = place(F, G, [\lambda_1, \lambda_2, \lambda_3, \lambda_4]) \tag{4.170}$$

ori:

$$\eta = dlqr(F, G, \mathcal{Q}, \mathcal{R}) \tag{4.171}$$

pentru care $\mathcal{Q}$ reprezintă matricea diagonală de 4×4 de penalizare a erorilor pe stări, iar $\mathcal{R}$ matricea diagonală de 2×2 de penalizare a efortului pentru intrări. întrucât legea de control se bazează pe stări estimate, elementele matricei $\mathcal{Q}$ au fost alese empiric de 100 de ori mai mari decât cele ale matricei $\mathcal{R}$.

Astfel, se defineşte legea de control:

$$u_{k+1} = -\eta\mathcal{E}_{k+1} \tag{4.172}$$

intrare ce se aplică direct pe acceleraţiile procesului (4.162):

$$a_{k+1} = u_{k+1} \tag{4.173}$$

Conform implementării, controlul realizat este proporţional, şi pare limitat. Totuşi, reamintind procesul modelului, acceleraţiile de pe intrări sunt integrate de două ori numeric pentru a obţine poziţiile, deci sunt prezente şi elementele integratoare. Mai mult, în vectorul de stare al urmăritorului sunt prezente şi pozitîile pe cele două axe, dar si derivatele lor, vitezele, ceea ce denotă existenţa unui element derivativ. Astfel, sistemul întreg se comportă ca un regulator PID (Proporţional, Integrator, Derivativ) şi singura limitare este constrângerea ecuaţiilor la un proces liniar.

Astfel, scenariul urmăririi se desfăşoară conform figurii 4.14. Urmăritorul işi reglează acceleraţia print-una din cele două legi de control pentru a minimiza diferenţa dintre poziţiile Px şi Sx, respectiv Py şi Sy. Acesta este ajutat de măsurătorile zgomotoase primite de la un senzor intrinsec de tip GPS şi de coordonatele polare transmise de sonar.

Am constatat empiric că legea de alocare a polilor este mai fiabilă pentru scenariul neliniar decât reglarea liniar-pătratică. Totuşi, cum scenraiile simulate sunt de tip Monte - Carlo (cu variabile aleatorii pentru acceleraţie, giraţie, traseu), o concluzie analitică nu poate fi obţinută, ci doar una statistică.

Pentru implementarea regulatorului, polii alocaţi sunt de un ordin redus de 100 de ori faţă de echivalentul unei implementări reale, pentru a fi mai aproape de centrul cercului unitate.

De asemenea, pentru implementarea regulatorului liniar-pătratic, penalizările pe erorile stărilor au fost considerate de $100 \rightarrow 1000$ de ori mai mari decât cele pe intrări. Este demn de menţionat că niciuna dintr-e aceste legi nu ar fi fiabilă pentru un control real, dar funcţionează pentru simulare.

Proiectarea efectivă bazată pe scenariul descris va fi analizată în capitolul următor.

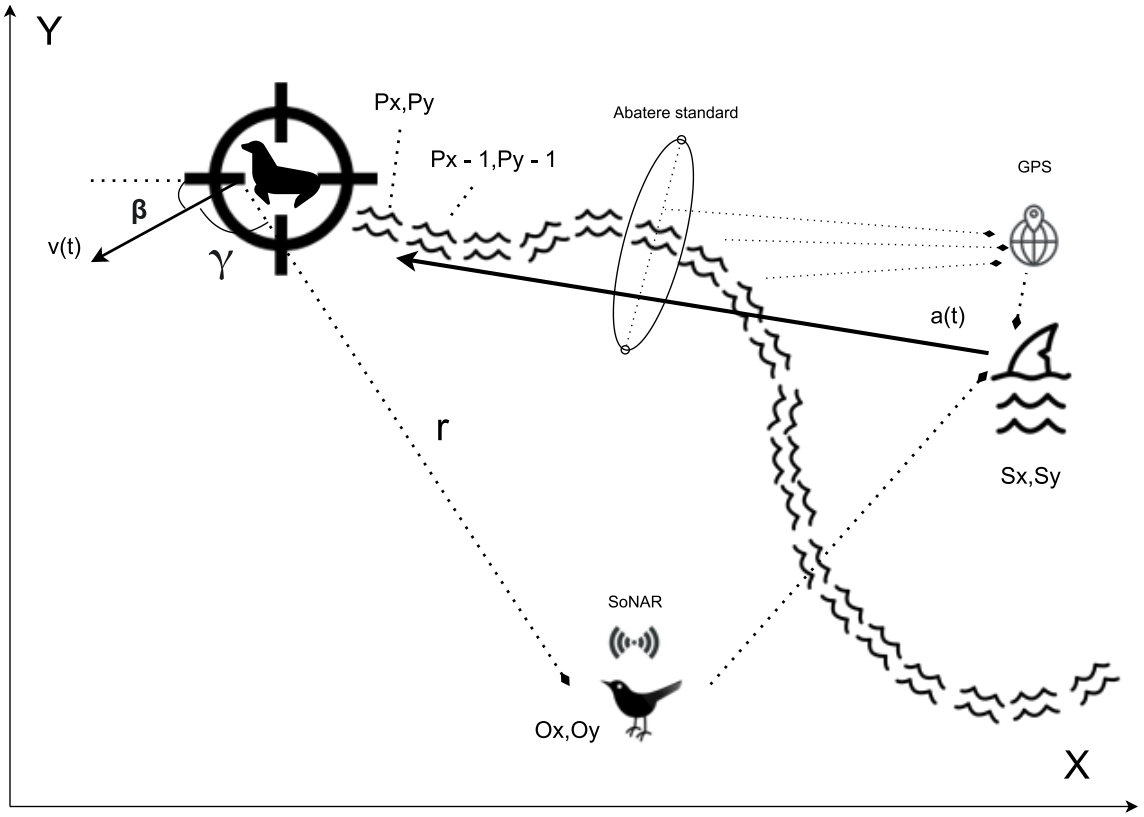


Figura 4.14: Scenariul urmăririi

4.6 Estimarea orientării

Odată cu implemntarea legii de control şi a celor trei algoritmi de estimare a pozitiei, teoria din spatele scenariului simulat poate fi considerată încheiată. Spre sfârşitul acestui capitol, propun o analiză a domeniului informaţiei senzorilor inerţiali, în vederea realizării joystick-ului, a cărui funcţionalitate va fi descrisă în particular în următorul capitol.

Aşadar, doresc ca în continuare să dedic acest capitol fundamentului teoretic din spatele rezultatelor finale ce urmează să fie expuse, şi am decis în acest sens să dedic următoarele secţiuni analizei sistemelor dinamice din spatele atitudinii, informaţia de interes obţinută de la senzori.

Această parte din proiect se bucură în mod particular de o aplicabilitate directă datorită implementării algoritmului propus pe un microcontroller Arduino Nano în legătură cu un bloc de senzori MPU-6050, spre deosebire de senzorii de tip GPS şi sonar a căror comportament a fost simulat. Cu toate acestea, teza propusă nu se doreşte a fi un expozeu al sistemelor înglobate de acest tip. Subiectul de interes este şi în acest caz dinamica datelor obţinute. Faptul că algoritmii de fuziune funcţionează şi pe date reale de la senzori este doar o coroborare a eficacităţii lor, şi de altfel şi motivul pentru care au fost gândiţi în primul rând.

4.6.1 Atitudinea

Atitudinea, termen pe care îl voi folosi interşanjabil cu noţiunea de *orientare* pentru restul tezei, reprezintă o mărime măsurabilă care exprimă poziţia unui corp sau a unei forme în spaţiu sau în plan. Orice definiţie dată orientării sugerează că ea are sens doar ca mărime relativă, nu absolută, şi de aceea atitudinea trebuie să fie conturată relativ la un sistem de coordonate [23].

Deşi aplicaţia prezentată ca joc se desfăşoară în planul monitorului, microcontrollerul ale cărui mişcări vor determina comportamentul de joystick este unul real, în spaţiul tridimentional, aflat la îndemâna jucătorului. Din acest motiv, cadrul mondial în care se reprezintă atitudinea senzorilor va fi cadrul cartezian tridimensional al realităţii mesei de lucru, deorece acest cadru poate fi considerat fix în comparaţie cu senzorii, ceea ce îl transformă într-un sistem de referintă [23].

Se pot deci recupera două cadre de importanţa, antedefinitul cadru de referintă, mondial, precum şi cadrul *obiect*, cadrul propriului al sistemului observat relativ la cadrul mondial [23]. Atitudinea se scrie astfel matematic ca acea mărime $\mathcal{A}$ evaluată de o functîe f care transformă un set de date din cadrul mondial $\mathcal{C}^m$ în cadrul obiect $\mathcal{C}^o$:

$$f(\mathcal{A}) : \mathcal{C}^m \to \mathcal{C}^o \tag{4.174}$$

4.6.2 Forme de reprezentare

Orientarea este, după cum reiese din definiţie, o mărime măsurabilă, dar formele de replicare ale mărimii sunt diferite, întrucât noţiunea este de interes pentru o varietate largă de domenii, pentru multe dintre ele fiind chiar singura punte comună.

Cele trei forme principale de reprezentare a atitudinii sunt unghiurile Euler, matricile de rotaţie şi cuaternionii [23]. Fiecare dintre aceste reprezentări prezintă avantaje şi dezavantaje, acesta fiind şi motivul pentru care nu există o singură reprezentare generală, acceptată ca etalon.

Unghiurile Euler

Definite de obicei prin convenţia Tait-Bryan XYZ [23], unghiurile Euler reprezintă secvenţa celor trei rotaţii din jurul axelor cadrului cartezian. Desi numele sugerează originea lor, Leonhard Euler a gândit iniţial o secventă de rotaţii de tip ZXZ, ceea ce nu va fi considerat în această lucrare. Pentru secvenţa XYZ, se folosesc denumirile de *ruliu(X), tangaj(Y) şi giraţie(Z).*Întrucât celelalte reprezentări sunt noţiuni abstracte, raportarea lor la unghiurile Euler devine cheia înţelegerii lor.

Matricile de rotaţie

Matricile cosinusoide de rotaţie sunt o transformare liniară a nui vector din spaţiu sau plan. Datorită proprietăţiilor lor de ortogonalitate, ele păstrează lungimea vectorului [23], doar modificându-i orientarea. Aceleaşi proprietăţi de ortogonalitate restricţionează însă şi adecvarea lor pentru a reprezenta o rotaţie. Mai mult, deşi convenabilă pentru dinamica liniară, această reprezentare introduce redundanţă [24], şi nu va fi folosită decât intermediar în proiectarea acestei părţi.

Cuaternionii

William Rowan Hamilton a descoperit la sfârşitul secolului al XIX-lea că pentru a extinde planul complex bidimensional nu este suficient să introducă o dimensiune în plus, ci două, pentru ca în total să poată opera produse cu patru componente. Noul sistem pe care inventatorul l-a descoperit şi l-ar fi sculptat pe un pod funcţionează după următoarea lege [24]:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{4.175}$$

pentru care se defineşte cuaternionul [24]:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{4.176}$$

$$q = [q_0, q_1, q_2, q_3] \tag{4.177}$$

Cuaternioni Unitate

Pentru a putea reprezenta o rotaţie în spaţiu cu un cuaternion, se desrpinde următoarea constatare despre norma euclidiană a vectorului [24] [23]:

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{4.178}$$

Cu alte cuvinte, nu toţi cuaternionii reprezintă o rotaţie, ci doar cei care au lungimea 1, deoarece, la fel ca în cazul matricilor de rotaţie, nu trebuie să schimbe lungimea vectorului transformat, ci doar orientarea lui.

Cheia intrepretării un cuaternion unitate este reprezentarea geometrică a rotaţiei ca parte reală a vectorului (q_0) relativă la descriptorii celor trei axe ale sistemului (q_1, q_2, q_3) [25] pentru care s-au obţinut următoarele egalităţi [64] [65]:

$$q_0 = \cos\frac{\delta}{2} \tag{4.179}$$

$$q_1 = \cos\alpha \sin\frac{\delta}{2} \tag{4.180}$$

$$q_2 = \cos\beta \sin\frac{\delta}{2} \tag{4.181}$$

$$q_3 = \cos\gamma \sin\frac{\delta}{2} \tag{4.182}$$

Figura 4.15 reprezintă o posibilă intrerpretare a unei rotaţii în interiorul spaţiului tridimensional menită să simplifice întelegerea unei noţiuni cvadimensionale [64] [65].

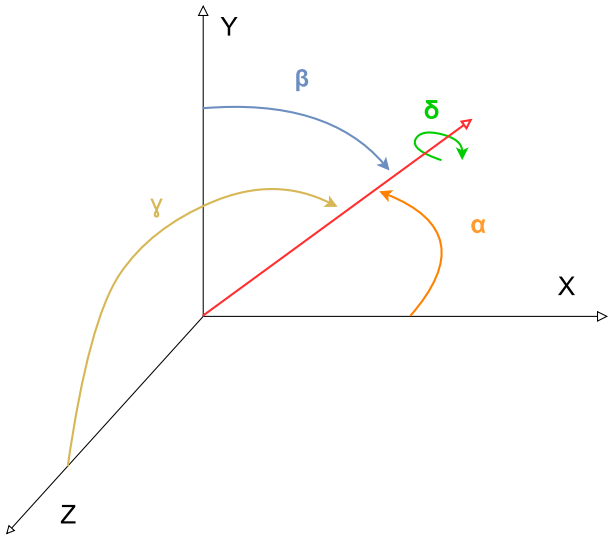


Figura 4.15: Interpretarea geometrică a cuaternionilor unitate

Pentru ca rezultatul să poată fi folosit în proiect, este nevoie ca o serie de transformări să fie aplicate cuaternionului unitate pentru a recupera secventă de unghiuri Euler, cu a căror informaţie se poate ulterior opera controlul animaţiei prin senzori. Interpretarea celor patru elemente scrisă desfăşurat pentru cele trei unghiuri este [25]:

$$q_0 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{4.183}$$

$$q_1 = \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{4.184}$$

$$q_2 = \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \tag{4.185}$$

$$q_3 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \tag{4.186}$$

pentru care ϕ, θ, ψ reprezintă unghiurile de ruliu, tangaj şi giraţie (*roll, pitch, yaw*) în această ordine.

Transformarea inversă din cuaternion unitate în unghiuri Euler necesită trecerea printr-o matrice intermediară de rotaţie [26]:

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{4.187}$$

din care se recuperează secvenţa de unghiuri [23]:

$$\phi = -\arctan\frac{R_{23}}{R_{33}} \tag{4.188}$$

$$\theta = -\arcsin R_{13} \tag{4.189}$$

$$\psi = \arctan\frac{R_{12}}{R_{11}} \tag{4.190}$$

Dacă orientarea poate fi definită ca o mărime ăsurabilă pentru contextul oferit în (4.174), schimbarea orientării în timp poate fi modelată ca un sistem dinamic folosind calcul derivativ. Astfel, o schimbare în rotaţie poate fi convenabil definită cu ajutorul cuaternionilor [23]:

$$\dot{q} = \frac{1}{2}\mathcal{W}\vec{w} \tag{4.191}$$

unde matricea $\mathcal{W}$ reprezintă transformarea liniară a vectorului $\vec{w}$ ce curpinde cele

trei derivate ale unghiurilor de rotaţie, vitezele unghiulare [23].

$$\mathcal{W} = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & a_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} \tag{4.192}$$

Avantajul major introdus de cinematica cvadimensională a reprezentării prin cuaternioni este imunitatea pe care aceasta o conferă la singularităţiile introduse de reprezentarea atitudinii prin secvenţa Euler. Aceste singularităţi reprezintă valori critice pentru ecuaţiile dinamicii unghiurilor, şi fenomenul pe care îl suportă poartă denumirea de *gimbal lock* [23].

Am simulat [65] în figura 4.16 dinamica schimbării de rotaţie prin ambele reprezentări pentru un set identic de condiţii iniţiale şi viteze unghiulare, şi se observă cum atunci când unghiul de tangaj converge la punctul critic de $-90°$ [23], valorile celorlalte două unghiuri explodează. Aceste divergenţe pot fi restrâse într-un interval dorit (ex. $\pm 180°$), dar atunci cele două axe vor coincide, iar sistemul va pierde un grad de libertate, ceea ce produce evenimentul de *gimbal lock*.

Cuaternionii, după cum se observă în al doilea grafic din figură, neagă acest efect printr-o compensaţie adusă de celelalte două axe de rotaţie (ruliu şi giraţie), acest lucru fiind posibil prind calitatea orientării de a putea fi reprezentată în mai multe feluri pentru acelaşi tip de mişcare.

Acest avantaj a dus la decizia de a folosi reprezentarea sub formă de cuaternioni pentru modelul procesului algoritmului de estimare a orientării

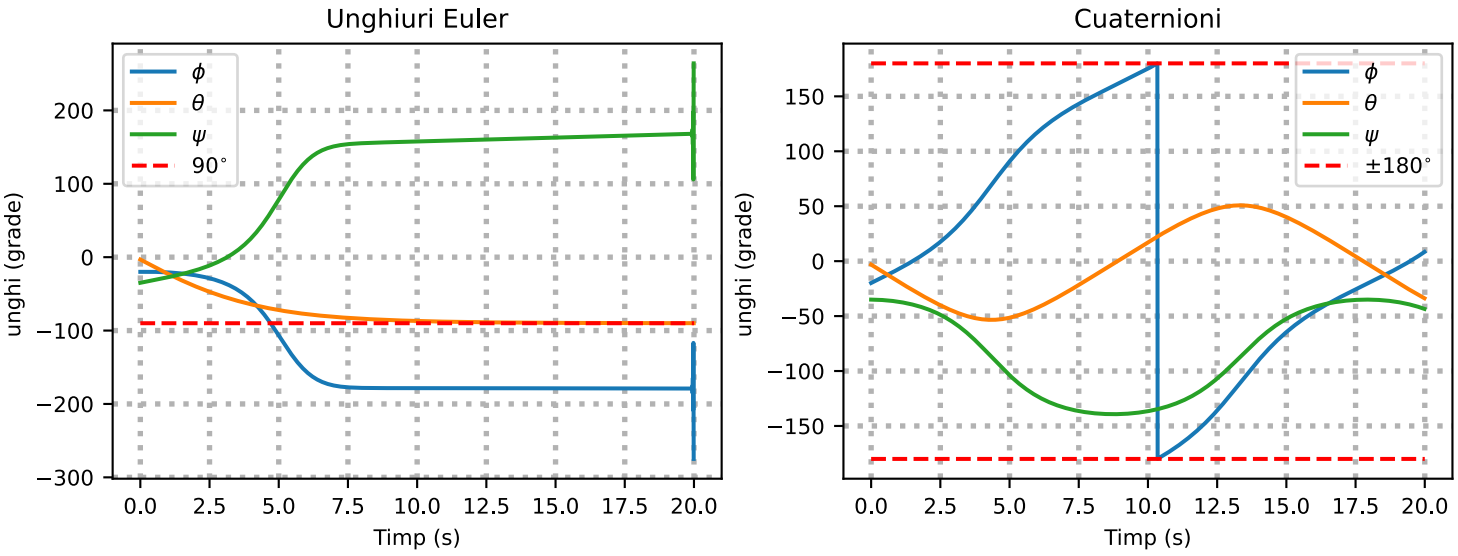


Figura 4.16: Comparaţie unghiuri Euler - Cuaternioni pentru dinamica orientării

Capitolul 5

Proiectare de Detaliu şi Implementare

5.1 Giroscopul - Modelul procesului

Pentru estimarea orientării am implementat un filtru Kalman liniar adaptat celor doi senzori reali prezenţi. Faza de predicţie a filtrului propagă în timp modelul procesului bazat pe transformările cuaternionilor, dar în acelaşi timp va primi şi măsurători de la giroscop înainte de o fază de corecţie, care va fi elaborată în următoarea secţiune. Astfel, informaţia primită de filtru va fi corpul vitezelor unghiulare:

$$\vec{w} = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{5.1}$$

unde p, q, r reprezintă vitezele unghiulare citite de pe cele trei axe ale vectorului.

După cum a fost menţionat şi în studiul bibliografic, unghiul de rotaţie din jurul axei Z nu va fi trecut prin procesul de corecţie, şi deci nu va fi verificat pentru convergenţă. Datorită lipsei de referinţă pentru axa de giraţie, fără ajutorul unui magnetometru care poate indica nordul, sistemul de senzori compus dintr-un accelerometru şi un giroscop triaxiale nu returnează o valoare relevantă pentru un utilizator.

Măsurătoarea provenită de la giroscop poate fi integrată în timp pentru a obţine unghiul, iar datele de la accelerometru pot fi prelucrate pentru a returna o valoare locală, dar fără un sistem de referintă, unghiul măsurat a fi mereu relativ la pozitîa în care senzorul a fost conectat, ceea ce nu este de dorit Tocmai de aceea am decis să constrâng folosirea microcontroller-ului la cele informaţia oferită.

Cu toate acestea, pentru a verifica validitatea unui cuaternion unitate (4.178), dinamica unghiului de giraţie trebuie inclusă, iar faza de corecţie trebuie modificată corespunzător pentru a evalua doar două dintre cele trei componente ale vectorului.

Astfel, vitezele unghiulare citite sunt transformate în dinamica 4D prin (4.192), după care mărimea este integrată în timp prin metoda Euler înainte conform unghiului simulat de giraţie (4.114) din animaţie:

$$q = q_0 + \dot{q}\Delta t \tag{5.2}$$

pentru care q_0 este iniţializat cu un vector nul, iar Δt este chiar perioada de prelevare a datelor de la senzor:

$$q_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.3}$$

şi cuaternionul obţinut q este normalizat cu norma (4.178) pentru defini o rotaţie:

$$q \leftarrow \frac{q}{\|q\|} \tag{5.4}$$

după care este transformat în secvenţa Euler corespunzătorare conform (4.187) - (4.190) pentru a intra într-o ulterioară fază de corecţie.

Pentru modelul procesului se defineşte matricea de covariantă P care modelează ataţ erorile provenite de la giroscop, cât şi cele de transformare ale vectorului cuaternion. Ea este iniţializată ca o matricea diagonală a varianţelor fiecărei componente a cuaternionului:

$$P_0 = \begin{bmatrix} p\sigma_{q_0}^2 & 0 & 0 & 0 \\ 0 & p\sigma_{q_1}^2 & 0 & 0 \\ 0 & 0 & 0p\sigma_{q_2}^2 & 0 \\ 0 & 0 & 0 & p\sigma_{q_3}^2 \end{bmatrix} \tag{5.5}$$

astfel încât pentru o incertitudine mare variabila p poate fi setată la un ordin relativ la valorile apropiate de unitate ale cuaternionului, iar pentru siguranţă absolută poate fi egalată cu zero.

Zgomotul de proces este aditiv pentru covarianţă şi deci asemănător ca în (4.85) se poate propaga în timp:

$$P = FP_0F^T + Q \tag{5.6}$$

unde F este matricea identitate I_4 întrucât mărimea cuaternionului nu se transformă, iar Q înglobează zgomotul de proces al transformării şi al măsurătorii de la senzor:

$$Q = \begin{bmatrix} \sigma_{q_0}^2 & 0 & 0 & 0 \\ 0 & \sigma_{q_1}^2 & 0 & 0 \\ 0 & 0 & 0\sigma_{q_2}^2 & 0 \\ 0 & 0 & 0 & \sigma_{q_3}^2 \end{bmatrix} \tag{5.7}$$

Şi astfel au fost obţiunute ecuaţiile predicţiei filtrului de fuziune.

Notă

O restricţie de firmware a legăturii matlab Arduino a fost observată pentru alegerea frecvenţei de citire a datelor de la senzor. Pentru ca aplicaţia să răspundă cât mai rapid

comenzilor utilizatorului, frecvenţa de prelevare maximă posibilă a fost aleasă:

$$f = 200Hz \tag{5.8}$$

$$\Delta t = \frac{1}{f} = 0.005s \tag{5.9}$$

a cărei inversă va fi chiar perioada de eşantionare a procesului discret al modelului.

Un giroscop oferă informaţie despre unghiuri prin integrarea numerică a vitezelor rotative. Componenta integratoare reduce zgomotul de frecvenţe mari din proces, ceea ce este de dorit, dar integrează pe de altă parte şi eroarea de măsură constantă (numită *bias* [66]), ceea ce duce la îndepărtarea progresivă a măsurătorilor de adevăr (*drift* [19]). Tocmai de aceea procesul trebuie corectat de un alt senzor ale cărui erori nu sunt corelate cu acelaşi fenomen.

5.2 Accelerometrul - Modelul măsurătorilor

Folosit de unul singur, giroscopul poate integra vitezele unghiulare pentru a obţine un unghi reativ la sistemul de referintă pentru care a fost iniţializată conexiunea. Fără un alt senzor care oferă o informaţie despre sistemul de referintă, măsurătorile obţinute nu pot fi absolute, şi consola de joc nu ar putea funcţiona corespunzător.

Un accelerometru triaxial măsoară atât acceleraţia gravitaţională, cât şi acceleraţiile liniare de pe cele trei axe. Măsurătoarea axei gravitaţionale se opune axei de jos din sistemul de referinţă NED (North-East-Down) şi astfel senzorul are o informaţie despre cadrul de referinţă în care se află.

Pe cont propriu, folosind transformări trigonometrice, accelerometrul este fiabil pentru a oferi informaţii despre unghurile de înclinaţie ale obiectului de care este ataşat. Aşadar, pentru setul de trei măsurători provenite de la accelerometru:

$$m_{acc} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \tag{5.10}$$

se pot normaliza acceleraţiile axelor la norma euclidiană gravitaţională [66]:

$$\phi = \arcsin \frac{a_y}{g} \tag{5.11}$$

$$\theta = \arcsin \frac{a_x}{g} \tag{5.12}$$

unde:

$$g = \sqrt{a_x^2 + +a_y^2 + a_z^2} \tag{5.13}$$

Demonstraţia ecuaţiilor trigonometrice poate fi făcută geometric conform figurii 5.1. Pentru un senzor MPU-6050 se consideră sensibilitatea de $8192LSB/g$ [67] se consideră rotaţia de 45° a axei Xo a obiectului în raport cu axa Xm mondială. Astfel, pentru a faciliza explicaţia, se poate considera că sensibilitatea este *bifurcată* în numerele egale cu 4096. Următorul calcul poate fi evaluat pentru a recupera unghiul de mişcare:

$$\frac{4096}{\sqrt{4096^2 + 0^2 + 4096^2}} = 0.7071 \tag{5.14}$$

$$\arcsin 0.7071 = 45^\circ \tag{5.15}$$

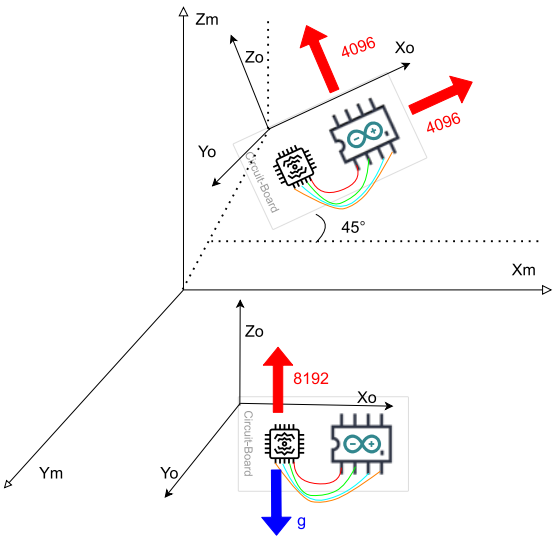


Figura 5.1: Interpretarea geometrică a unghiurilor de înclinaţie ale accelerometrului

Măsurătorile transformate trigonometric (4.202) - (4.203) sub forma de unghiuri sunt introduse în faza de corecţie a filtrului de fuziune. Vectorul secvenţei Euler e augmentat cu o valoare nulă pentru măsurătoarea giraţiei, pentru ca transformările (4.183) - (4.186) să poată avea loc.

Nou cuaternion provenit din măsurătorile accelerometrului este comparat cu cel obţinut prin procesul modelului giroscopului, iar diferenţa (inovaţia) dintre cei doi vectori este ponderă de câştigul Kalman pentru a realiza fuziunea. Aşadar, au loc ecuaţiile:

$$S_{k+1} = HP^-_{k+1}H^T + R \tag{5.16}$$

$$K_{k+1} = P_k H^T {S_{k+1}}^{-1} \tag{5.17}$$

$$\hat{x}^+_{k+1} = \hat{x}^-_{k+1} + K_{k+1}(transformare([\phi, \theta, 0] - H\hat{x}^-_{k+1}) \tag{5.18}$$

$$P^+_{k+1} = P^-_{k+1} - K_{k+1}HP^-_{k+1} \tag{5.19}$$

unde H, asemenea lui F, este matricea unitate I_4, şi poate fi eliminat din ecuaţii, iar matricea R reprezinţa zgomotul măsurătorilor şi poate fi modelată cu acelaşi ordin pentru parametrii ca şi Q:

$$R = \begin{bmatrix} \sigma_{q_0}^2 & 0 & 0 & 0 \\ 0 & \sigma_{q_1}^2 & 0 & 0 \\ 0 & 0 & 0\sigma_{q_2}^2 & 0 \\ 0 & 0 & 0 & \sigma_{q_3}^2 \end{bmatrix} \tag{5.20}$$

Pentru un zgomot de măsură R mai mic decât zgomotul de proces Q ($trace(R) < trace(Q)$), filtrul va acorda mai multă încredere accelerometrului. Inversând rolurile ($trace(Q) < trace(R)$), modelul procesului giroscopului va fi prioritar pentru filtru. Procedura descrisă se numeşte acordare şi este o etapă importantă a fuziunii senzoriale. În acest context, *trace* defineşte suma valorilor de pe diagonala matricei.

Notă

Pe cont propriu, accelerometrul oferă informaţii relevante pentru un senzor aproape static. În mişcare, măsurătorile lui sunt zgomotoase şi funcţionează mai bine în tandem cu un semnal de la giroscop filtrat prin integrarea numerică.

Filtrul Kalman conceput are recursivitate doar pentru covarianţă, dar nu şi pentru vectorul de stare, deoarece, în fiecare nouă etapă următoare, informaţia despre stare este primită de la celălalt senzor, şi astfel se realizează o estimare în buclă semi-deschisă, conform schemei generale din figura 5.2.

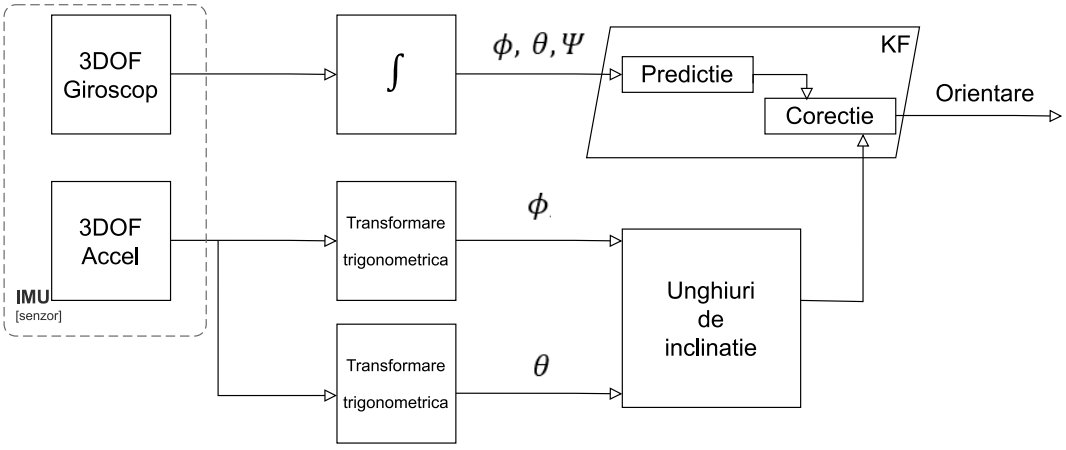


Figura 5.2: Schema generală de fuziune a datelor de la senzori

Pentru figura 5.3 am aplicat algoritmul de fuziune pentru placa de dezvoltare şi senzori în poziţie statică. Se poate observa că filtrul Kalman propus este un compromis dintre cele două măsurători individuale, iar stările sale sunt mai aproape de adevăr (0°). Acest fapt va fi confirmat în detaliu în capitolul următor dedicat validării rezultatelor.

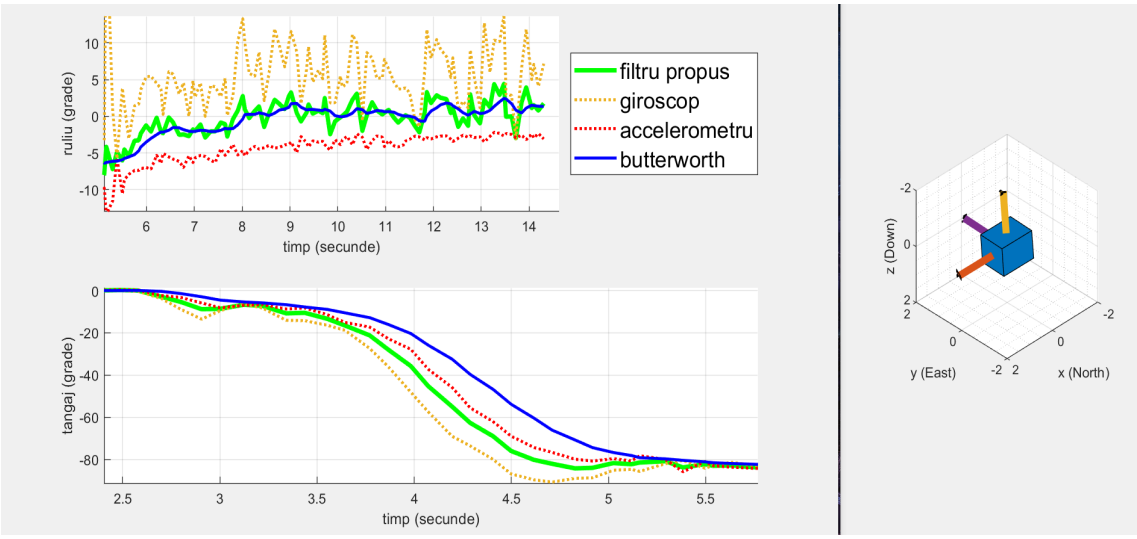


Figura 5.3: Fuziunea senzorială prin Kalman comparată cu măsurătorile individuale

Pentru a reduce şi mai mult eroarea, un filtru Butterworth [68] de tip trece jos poate fi folosit recursiv pe baza unei funcţii de transfer de ordin I, cu o frecventă de tăiere aleasă empiric de 10 ori mai mică decâfrecventă nyquist ($f_{nyquist} = f/2 = 100Hz$). Principalul dezavantaj adus de suplimentarul FTJ este că acesta introduce o întârziere măsurabilă, fapt pe care l-am considerat nefiabil pentru o aplicaţie de timp real, şi pe care deci nu l-am introdus în implementarea finală.

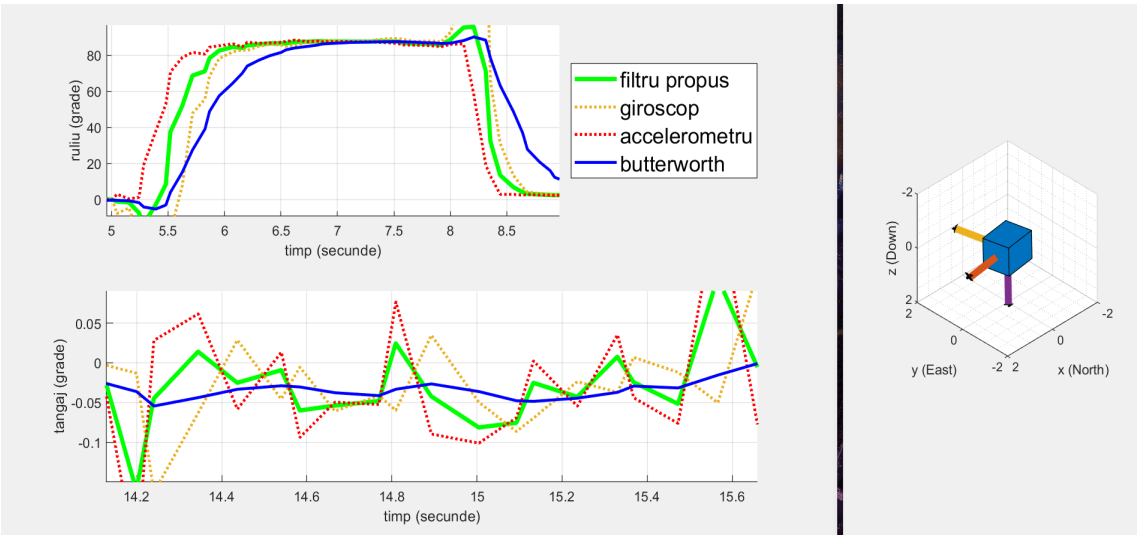


Figura 5.4: Fuziunea senzorială prin Kalman comparată cu măsurătorile individuale

Convergenţa algoritmului este garantată de filtrul liniar pentru ambele unghiuri, şi estimarea orientării funcţionează şi în regim dinamic, după cum se vede în figura 5.4.

5.3 Schema generală de funcţionare

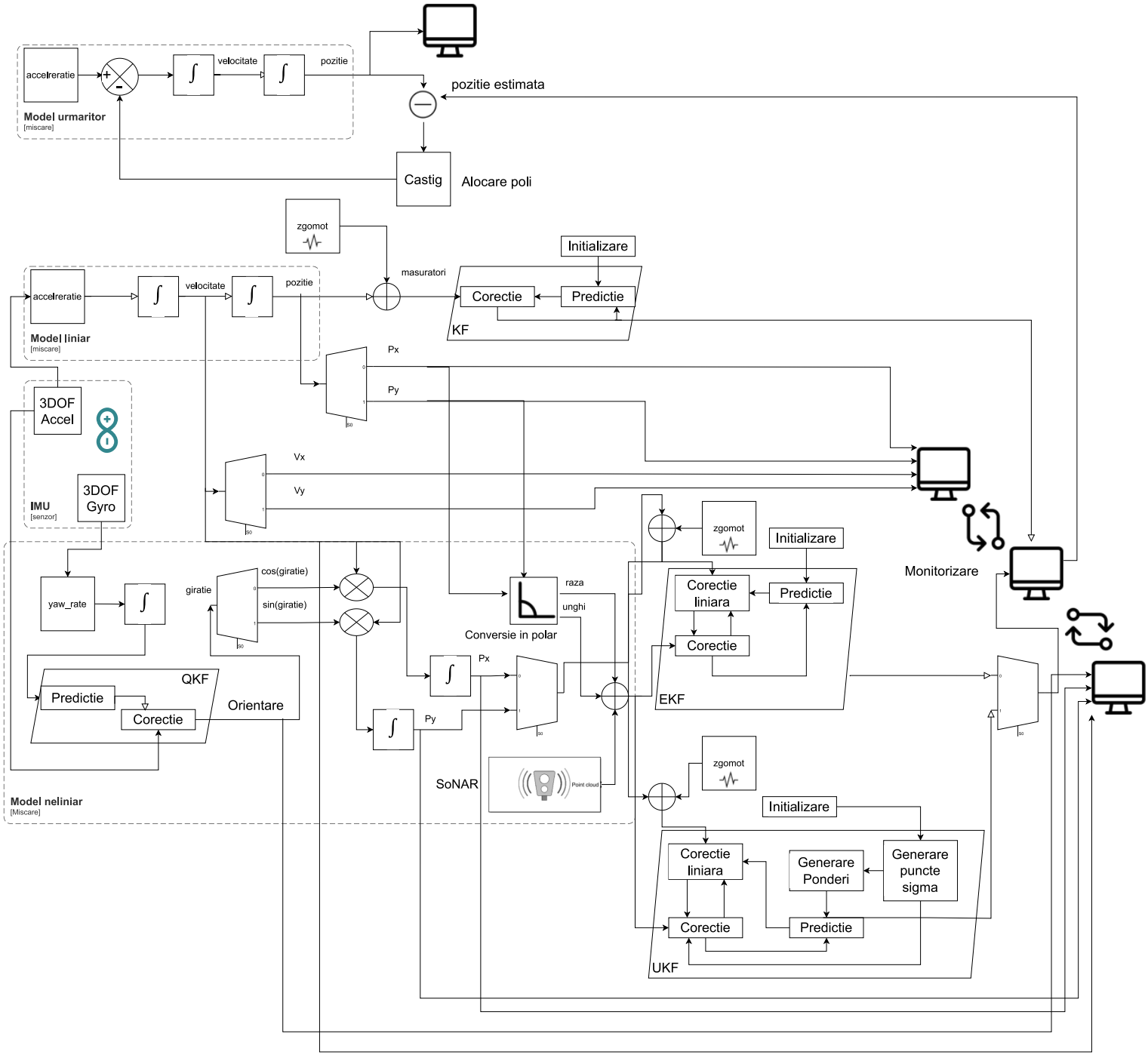


Figura 5.5: Schema bloc generală de funcţionare a aplicaţiei

Schema din figura 5.5 oferă o privire de ansamblu asupra scenariului de derulare al jocului. Obiectul urărit este controlat prin orientarea plăcii de dezvoltare obţiunită prin fuziunea datelor celor doi senzori prezenţi, îar trei metode de estimare sunt posibile în funcţie de tipul de mişcare realizat. Fuziunea este realizată conform schemei din figura 5.2, prin algoritmul de fuziune QKF (filtrul Kalman cu cuaternioni) prezentat treptat de la începutul capitolului.

Pentru un proces liniar, filtrul Kalman simplu (KF) este de preferat, dar umărirea se poate realiza şi prin filtrele extins sau unscented. Acest lucru e posibil pentru că şi obiectul în mişcarea liniară dispunde de o viteză de rotaţie, doar că se consideră nulă. Filtrul simplu nu are la dispoziţie o fază de corecţie bazată pe măsurători polare, pentru că acestea nu pot fi restrânse la ecuaţii algebrice liniare.

Pozitia unui obiect relativă la cele două axe de mişcare poate fi transformată în coordonate polare pentru a intra în faza de corecţie a unuia dintre cele două filtre neliniare, unde măsurătorile vor fi comparate cu informaţia de tip Point Cloud a sonarului, care are mult mai multă acurateţe decât măsurătorile de tip GPS primite în faza de corecţie liniară. Astfel, atât pentru EKF (filtrul Kalman extins), cât şi pentru UKF (unscented), fuziunea senzorială se realizează prin aditivitatea celor două faze de corecţie. Iniţial, măsurătorile de tip GPS confirmă sau corectează eroarea estimării obţinute prin propagarea procesului modelului, după care şi mai fiabilele măsurători ale sonarului au o intervenţie în algoritm.

În faza de predicţie a fitrelor neliniare, pot fi primite măsurători ale vitezei unghiulare zgomotoase de la giroscop pentru modelare, dar nu este imperios necesar, deorece această mărimea poate fi modelată ca zgomot de proces, aşa cum este şi acceleraţia. Se realizează astfel trei algoritmi de estimare în buclă închisă, ale caror erori de estimare sunt obţinute prin compararea cu adevărul prin ceea ce e reprezentat schematic ca *monitorizare*, iar această eroare devine intrarea ponderată în dinamica procesului urmăritor.

Filtrul unscented este cel mai complex algoritm prezent în aplicaţie, deoarece a necesitat în plus două funcţii de calculare a punctelor sigma şi a ponderilor lor (explicate în capitolul precedent), iar acesta poate fi contruit fie pe baza aceluiaşi model de proces folosit şi pentru filtrul extins, fie pe baza unuia modificat, după cum se va vedea.

Am considerat procesul urmăritorului ca fiind liniar pentru toate cazurile de mişcare, pentru ca astfel să se poată aplica fie metoda de alocare a polilor schiţată în figură, fie prin reglare liniar - pătratică. Un punct de dezolvatare viitor care rezultă de aici este explorarea metodelor de control neliniare în vederea obţinerii unor rezultate mai aproape de adevăr.

Rezultatul final al algoritmul, şi implicit obiectivul proiectului propus este convergenţa stărilor urmăritorului la valorile de stare ale obiectului urmărit. Astfel se realizează o urmărire de traiectorie complet autonomă, fără intervenţia utilizatorului, iar în restul capitolului voi explica implementarea programatică activă a cumulului de noţiuni teoretice prezentate în capitolul dedicat analizei teoretice.

5.4 Diagrama claselor

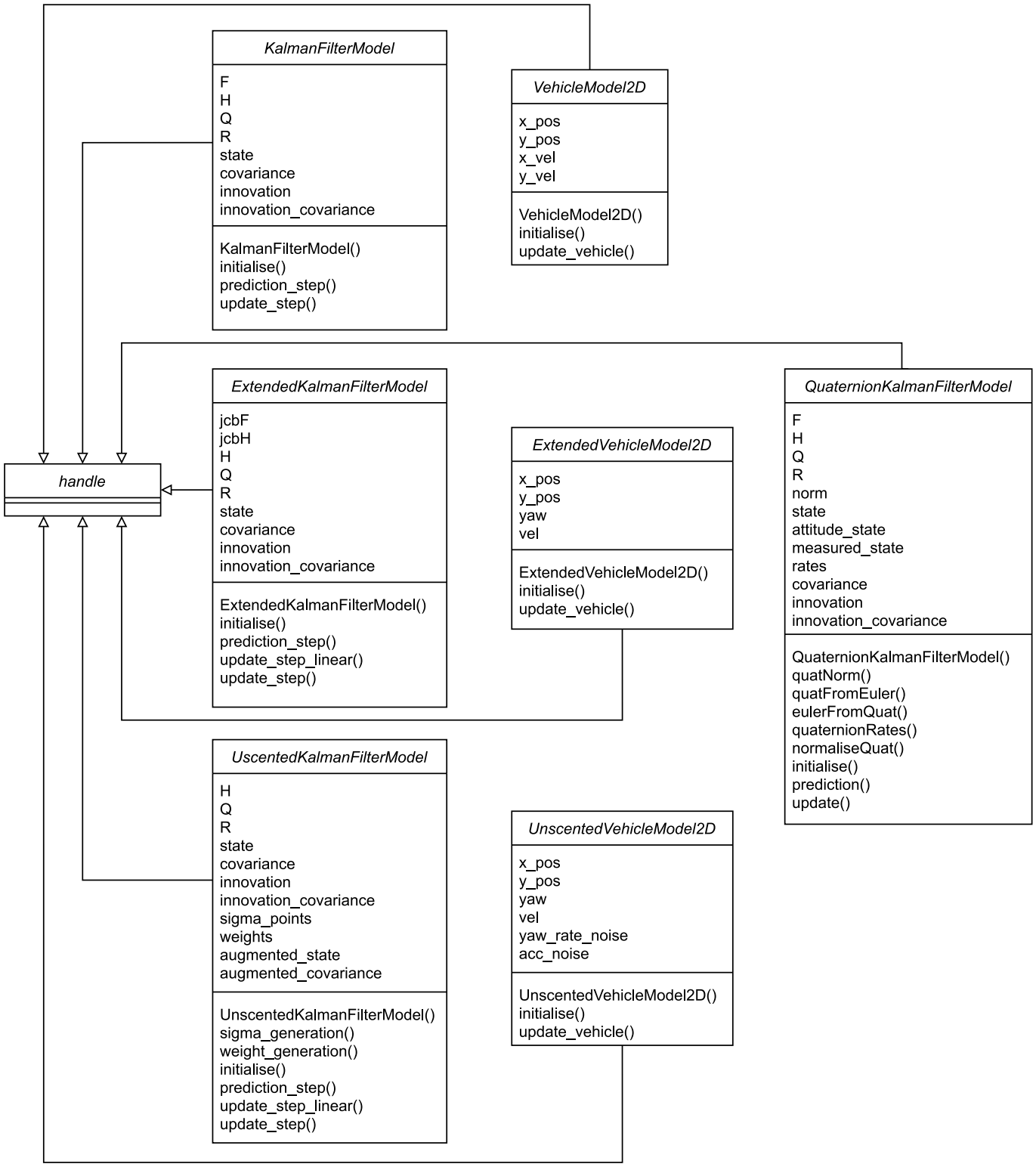


Figura 5.6: Diagrama claselor folosite în aplicaţie

Figura 5.6 prezintă totalitatea claselor implementate în MATLAB pe care le-am considerat necesare realizării jocului. Abstractizarea obiectivului prin obiecte şi clase a fost inspirată de proiectul [69]. Folosind structurile POO (programare orientată pe obiecte) puse la dispoziţie de limbajul de programare am coceput trei scenarii posibile de urmărire simulate: mişcare liniară, circulară şi aleatorie, iar cel mai înalt grad al performanţei este scenrariul mişcării comandate de utilizator, care se poate prezenta ca o combinaţie dintre celelalte trei.

Astfel, prin moştenire directă, toate clasele au acces la metodele superclasei *handle* [70]. Această clasă unică din limbaj este necesară pentru referirea, ştergerea sau modificarea instanţelor unei clase, şi astfel este indispensabilă modelării dinamicii obiectelor în timp. Această relaţie de moştenire este de altfel singura legătură între clase, întrucât am intenţionat funcţionarea lor independentă în funcţie de scenariul ales.

Aşadar, cele trei variaţiuni ale filtrului Kalman analizate în capitolul precedent se regâsesc în cele trei clase *KalmanFilterModel*, *ExtendedKalmanFilterModel* şi *UnscentedKalmanFilterModel*. Fiecare dintre filtre a fost implementat pentru a estima un model de mişcare diferit, iar în acest sens am implementat cele trei clase *VehicleModel2D*, *ExtendedVehicleModel2D* şi *UnscentedVehicleModel2D*.

Prima dintre cele trei modelează după cum se observă şi din atribute (numite proprietăţi în MATLAB) cazul de mişcare liniară şi poate fi estimată cu succes de toate cele trei filtre. Atât modelul extins, cât şi modelul unscented descriu mişcarea neliniară sub un unghi a obiectului şi singura diferenţă dintre cele două este admisia zgomotului pentru intrările procesului unscented (acceleraţia şi viteza unghiulară). Am inten;tionat în acest fel să creez o estimare mai robustă atât la zgomotele reale provenite de la senzor cât şi la cele simulate în aplicaţie. Cu toate aceastea, modelul extins este suficient pentru ambele filtre neliniare, pentru că se bazează pe acelaşi vector de stare.

Pentru fuziunea senzorială a datelor de la giroscop şi accelerometru demonstrată la începutul capitolului am implementat clasa *QuaternionKalmanFilterModel* care implementează ecuaţiile de transformare şi propagare a dinamicii orientării prezentate în (4.178) - (4.192) şi (5.2) - (5.9) pentru modelul procesului giroscopului, şi corecţia prin unghiurile de înclinaţie obţinute din datele accelerometrului regăsite în (5.10) - (5.20).

Toate clasele construite sunt instanţiate într-un script (cod) principal care iniţiază şi întreţine şi animaţia, după cum se va vedea în continuarea capitolului. Scriptul principal în MATLAB nu este o clasă, şi deci nu se regăseşte în diagrama din figura precedentă, dar funcţiile folosite sunt demne de explicat pentru a contura contextul programatic al aplicaţiei, şi nu numai cel ştiinţific.

Acest script principal este responsabil de asemenea de realizarea controlului urmăritorului, obiect care va fi mereu o instanţă a clasei *VehicleModel2D*, întrucât procesul său este modelat liniar. Tot aici are loc şi generarea graficelor de perfomantă care vor fi puse la dispoziţie în capitolul 6, iar implementarea structurilor de estimare pe clase facilitează comparatîa stărilor pentru o analiză în detaliu a erorilor.

5.5 Diagrama de flux

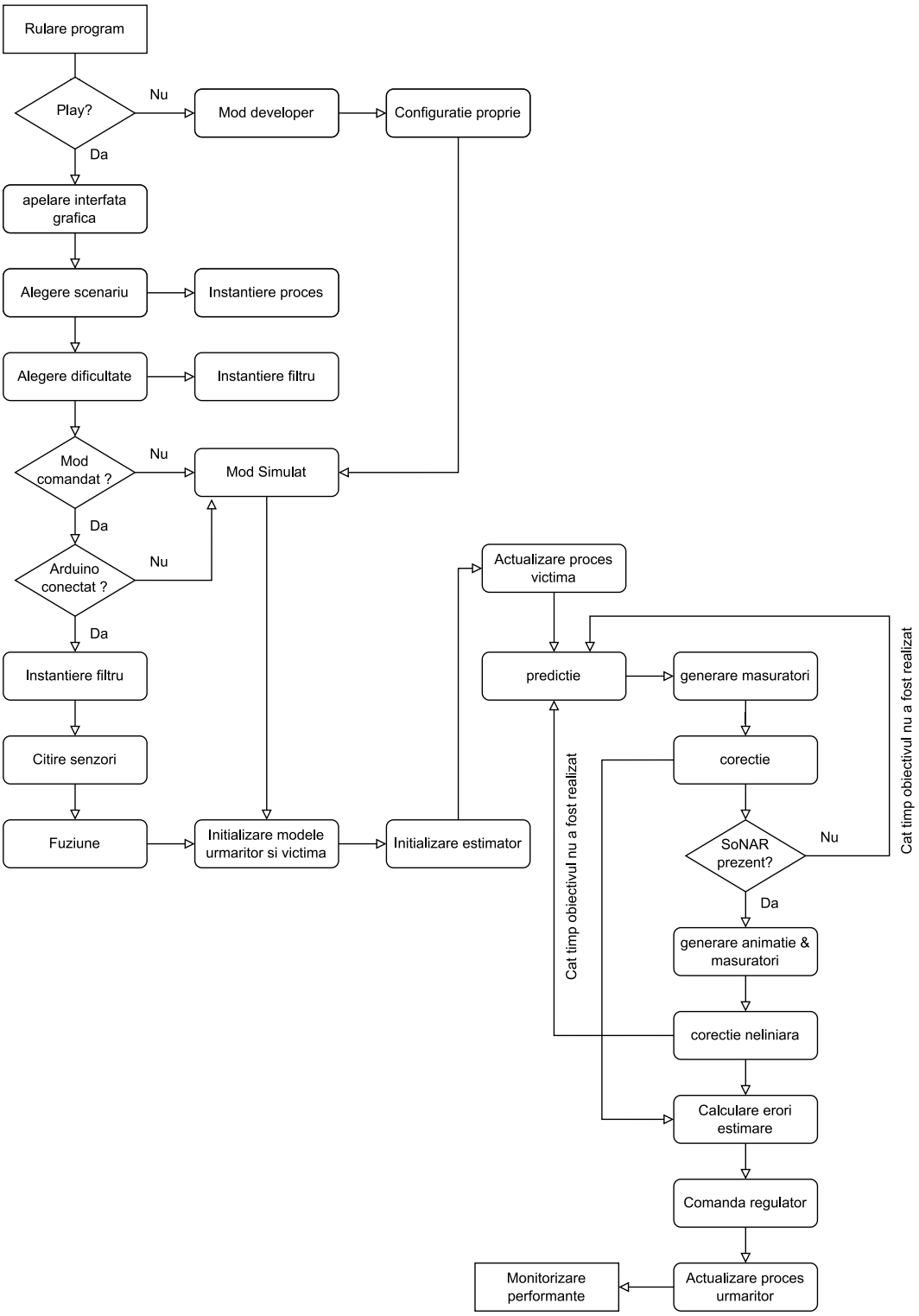


Figura 5.7: Diagrama fluxului de execuţie al programului

Execuţia sistematică a aplicaţiei în mai multe etape este prezentată în figura 5.7. Porgramul se desfăsoară ca un joc, cu o interfaţă grafică specifică, din care utiliziatorul poate alege scenariul urmăririi (traiectoria mişcării) şi nivelul de dificultate (algoritumul de estimare). În caz că placa de dezvoltate nu este conectată, jocul rulează ca o simulare în care se pot explora performanţele fiecărui filtru.

Modul de dezvoltare (*developer*) porneşte script-ul fără interfaţa grafică, şi rolul lui este de a oferi o configuraţie proprie simulării, în vederea acordării filtrelor. Astfel, pentru modelul liniar, se pot seta stările iniţiale ale procesului astfel încât să fie fiabile şi pentru procesul neliniar:

$$x_0 = \begin{bmatrix} p_{x_0} & p_{y_0} & v_0 & \beta_0 \end{bmatrix} \tag{5.21}$$

de unde se recuperează vitezele pentru cele două axe:

$$v_x = \cos\beta_0 \times v_0 \tag{5.22}$$

$$v_y = \sin\beta_0 \times v_0 \tag{5.23}$$

De notat este că unghiul *beta* setat la început este unghiul sub care se va desfăşura mişcarea liniară, însemnând că acesta nu se mai modifică pe parcursul traseului, iar dinamica lui este constantă. Pentru stările prelucrate se pot seta abaterile standard:

$$\sigma = \begin{bmatrix} \sigma_p & \sigma_v & \sigma_a & \sigma_m \end{bmatrix} \tag{5.24}$$

unde, pentru simplificare, abaterea standard a ambelor poziţii şi cea a ambelor viteze sunt evaluate cu câte o valoare. Deviaţia ambelor acceleraţii este înglobată în σ_a, iar σ_m reprezintă abaterea măsurătorilor primite de la GPS.

Am configurat şi un steag cu valoare booleană care stabileşte dacă filtrul va fi initializat cu stare x_0 a procesului, sau dacă acesta nu deţine cunoştiinte a priori despre proces. În ambele cazuri, filtrul liniar va converge. Acelaşi steag controlează şi iniţializarea parametrilor urmăritorului. Astfel, matricea de covarianţă a stării se programează:

$$P = \begin{bmatrix} \sigma_m^2 & 0 & 0 & 0 \\ 0 & \sigma_m^2 & 0 & 0 \\ 0 & 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \; sau \; P = \begin{bmatrix} \sigma_p^2 & 0 & 0 & 0 \\ 0 & \sigma_p^2 & 0 & 0 \\ 0 & 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \tag{5.25}$$

în funcţie de setarea steagului. Zgomotul de proces Q înglobează dinamica celor două acceleraţii primite ca intrări în proces şi este initializat ca în (4.47) folosind varianţa acceleraţiei (pătratul abaterii, considerat identic pentru ambele axe spre simplificare) în locul acceleraţiilor propriu zise. Zgomotul de măsură R va depinde mereu de varianţa măsurătorilor de tip GPS:

$$R = \begin{bmatrix} \sigma_m^2 & 0 \\ 0 & \sigma_m^2 \end{bmatrix} \tag{5.26}$$

Componentele vectorului de acceleraţie sunt setate stocastic cu funtia MATLAB *randn* [71] ce returnează numere aleatorii distribuite Gaussian, folosită şi pentru a genera zgomotul ce afectează măsurătorile exacte ale poziţiilor victimei, ce devin intrările funcţiei de corecţie (4.86) - (4.87). Atât vectorul de stare cât şi matricea de covarianţă sunt rescrise recursiv în ambele etape ale algoritmului şi astfel se realizează o estimareîn buclă închisă conform fazelor de predicţie (4.85) şi corecţie a filtrului liniar (4.106) - (4.113). Modelul procesului liniar, dar şi cel al urmăritorului sunt bazate pe legile de mişcare (4.44) - (4.45), scrise matricial ca în (4.162) - (4.163).

Faza de predicţie a procesului extins conţine în plus informaţii despre varianţa vitezei de giraţie a victimei. Pentru a face algoritmul mai eficient, am inclus o modificare a măsurătoarii giroscopului de pe axa X (prin care va comanda jucătorul giraţia propriu-zisă) pentru care am adăugat zgomot alb, deci se poate considera că filtrul primeşte informaţie în plus de la giroscop. Din moment ce algoritmul neliniar e vulnerabil la divegentă, am considerat obligatorie opţiunea de iniţializare a filtrului cu primele măsurători, deşi este demn de mneţionat că într-un scenariu real acest lucru nu este mereu posibil. Astfel, am iniţializat matricea de covarianţă.

$$P = \begin{bmatrix} \sigma_m^2 & 0 & 0 & 0 \\ 0 & \sigma_m^2 & 0 & 0 \\ 0 & 0 & \sigma_\beta^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \tag{5.27}$$

Prin liniarizarea distribuţiei zgomotului de proces (4.77), noua varianţă celor două intrări - viteza de giraţie şi acceleraţia poate fi înglobată astfel (4.114), (4.130) [69]:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta t^2 \sigma_{\dot{\beta}}^2 & 0 \\ 0 & 0 & 0 & \Delta t^2 \sigma_a^2 \end{bmatrix} \tag{5.28}$$

Faza de corecţie liniară funcţionează ca în cazul precedent, iar adăugarea sonarului este optională, pentru acesta modelându-se un zgomot de măsură R_{extins}:

$$R_{extins} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} \tag{5.29}$$

iar algoritmul decurge în continuare conform ecuaţiilor (4.128) - (4.141).

Pentru filtrul Kalman unscented, vectorul de stare se poate augmenta asstfel încât să conţină şi informaţie atât despre cele patru abateri ale stărilor, cât şi despre abaterile celou două măsurători neliniare de rază şi unghi oferite de sonar:

$$X_{aug} = [X, \nu_{p_x}, \nu_{p_y}, \nu_{\dot{\beta}}, \nu_a, \omega_r, \omega_\gamma] \tag{5.30}$$

unde ν şi γ reprezintă zgomotul de proces, respectiv măsură.

Se obţine astfel un vector augmentat ce conţine 10 stări, care este trecut prin transformata unscented (4.142), iar covarianţa lui este construită prin augmentarea matricei anterioare P astfel:

$$P_{aug} = \begin{bmatrix} P & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix} \tag{5.31}$$

şi aceasta este propagată conform (4.143), unde punctele σ şi ponderile au fost calculate conform (4.150) - (4.153) pentru 20 de valori. Pentru faza de corecţie neliniară, dacă este prezentă, matricea de covarianţa augmentată se rescrie astfel încât să înglobeze incertitudinea măsurătorilor sonarului:

$$P_{aug} = \begin{bmatrix} P & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R_{extins} \end{bmatrix} \tag{5.32}$$

şi funcţia de generare a punctelor sigma este apelată din nou, dar nu şi cea a ponderilor, care, conform (4.152) - (4.153) rămân constante pe parcursul estimării. Filtrul parcurge toate etapele (4.154) - (4.161). Sonarul este ales static pentru un punct de observare fix în raport cu care se măsoară raza şi unghiul obiectului urmărit pentru fiecare iteraţie.

Se observă cum transformata unscented introduce complexitate în plus pentru calcularea ponderilor şi punctelor σ, buclele de calcul devenind $\mathcal{O}(20n)$. Motivul pentru care cele 10 stări ale vectorului agumentate sunt dublate este oglindirea parametrilor conform figurii (4.13) în vederea recuperării noii elipse de covarianţă. Pentru aplicaţia prezentă de tip joc, această compelxitate este neglijabilă în raport cu numărul de iteraţii folosite pentru fiecare mişcare, şi filtrul unscented devine o opţiune avantajoasă pentru unele dintre scenarii.

Dinamica animaţiei depinde de perioada de eşantionare, şi acesta este şi motivul pentru care procesul continuu teoretizat la începutul capitolului precedent nu poate fi integrat într-o aplicaţie de acest fel, cum de altfel şi procesele înglobate pe sisteme de calcul sunt mereu discrete. Astfel, perioada de eşantionare şi frecvenţa de resetare a animaţiei au fost alese empiric ţinând cont de următoarea regulă:

$$\frac{1}{f} < pause < \Delta t \tag{5.33}$$

unde f reprezintă frecventă datelor de citire de la senzorul real MPU-6050, iar *pause* perioada de pauză dintre cadre. În acest fel, se asigură o tranziţie netedă de la comanda

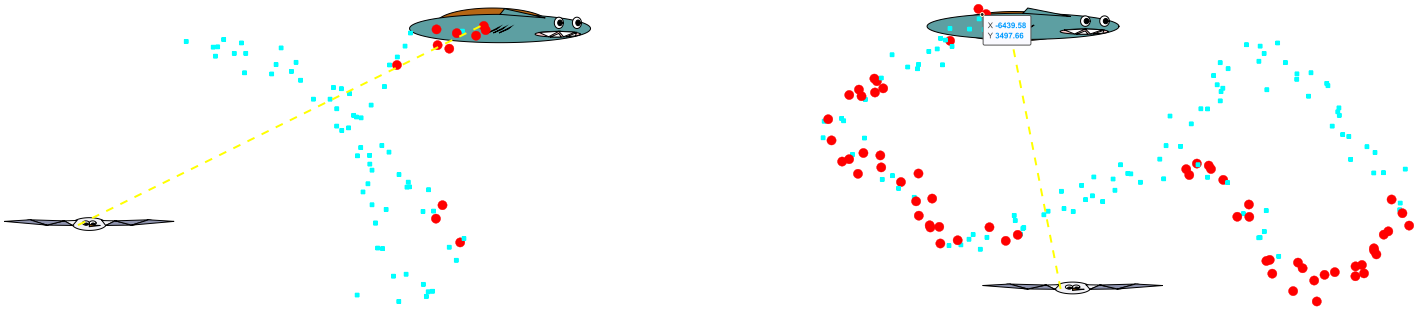
joystick-ului la animaţie, care la rândul ei permite observarea desfăşurării dinamicii procesului şi a urmăriltorului. În figurile 5.8 - 5.10 se pot observa diferitele scenarii în acţiune.

(a) Estimarea poziţiei (b) Convergenţa autonomă

Figura 5.8: Urmărirea traiectoriei liniare

(a) Estimarea poziţiei (b) Convergenţa autonomă

Figura 5.9: Urmărirea traiectoriei circulare



(a) Estimarea poziţiei (b) Convergenţa autonomă

Figura 5.10: Urmărirea traiectoriei aleatorii

Capitolul 6

Testare şi Validare

Pentru dinamica estimatorului, a procesului victimă şi a urmăritorului sunt salvate vectorial stările în fiecare moment de timp, şi astfel, la finalul jocului, atunci când rechinul prinde foca (prin epuizarea punctelor roşii de viată), se pot trage concluzii asupra performanţei urmăririi. Aceste grafice au fost folosite pentru a decide nivelul de acordare suplimentar cerut de regulator sau de filtru.

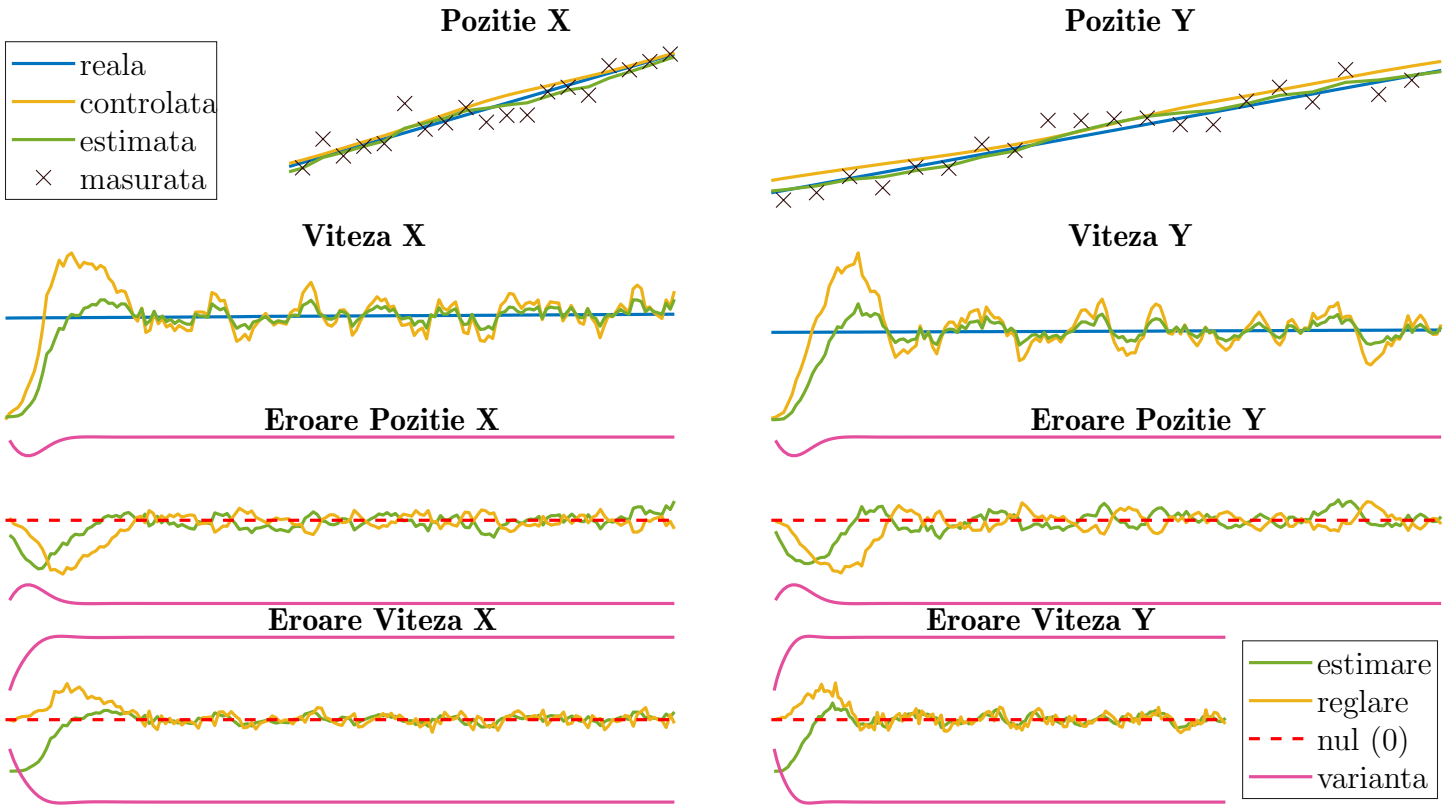


Figura 6.1: Evaluarea performantelor procesului liniar

Se poate observa în figura 6.1 cum stările estimatorului converg la cele reale ale procesului, respectiv cum stările urmăritorului converg prin control la estimările făcute de filtru. Pentru ambele poziţii, atât estimările cât şi reglarea sunt mai aproape de adevăr decât măsurătorile zgomotoase, ceea ce înseamnă că algoritmul şi-a îndeplinit obiectivul.

Mai mult, după cum se vede în ultimele 4 figuri, erorile de estimare ale fiecărei stări converg la 0, şi sunt înglobate de incertidudinea de $\pm 3\sigma$ a deviaţiei standard rezultate din extragerea radicalului varianţei. Pentru fiecare fază de predicţie incertitudinea din proces este crescută de zgomotul de proces Q, conform (4.85), pentru ca mai apoi să fie întreţinută prin faza de corecţie (4.113). În acest exemplu, covarianţa stărilor nu este minimizată, ci rămâne aproximativ constantă din cauza noilor incertiduni apărute în proces, dar atăt stările, cât şi erorile converg, ceea ce înseamnă că filtrul liniar are succes.

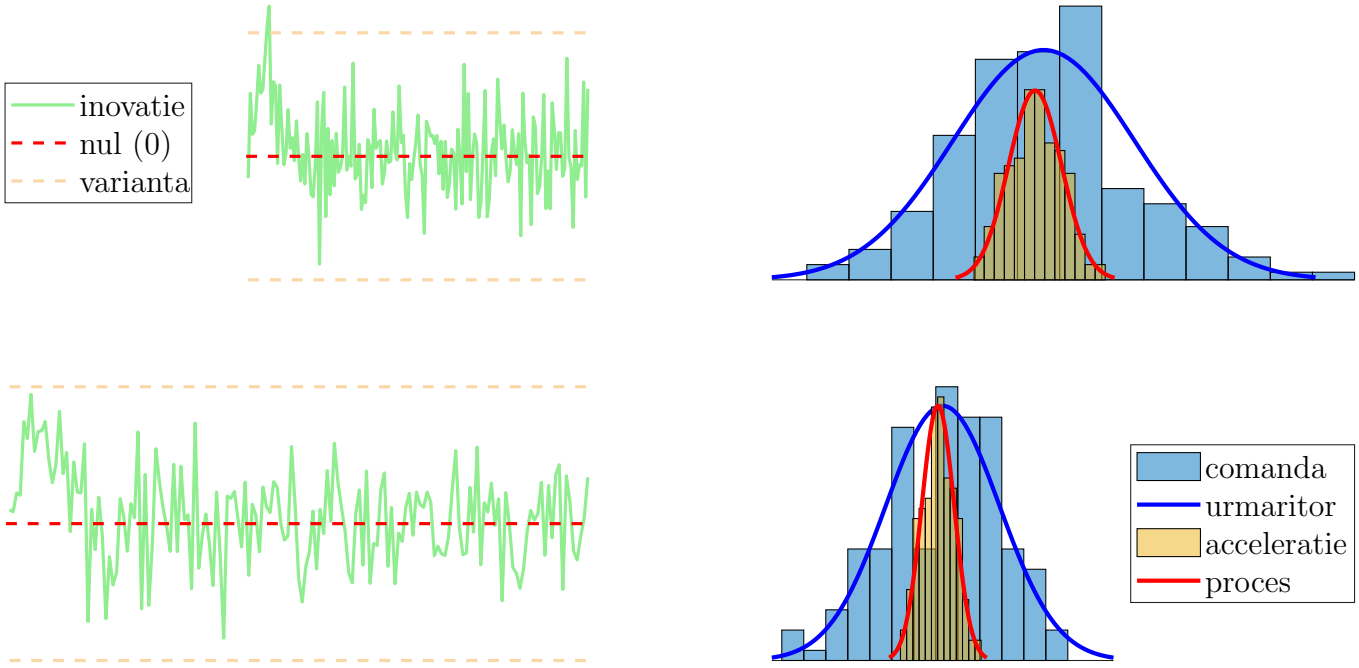


Figura 6.2: Evaluarea performantelor procesului liniar

Figura 6.2 prezintă pe de o parte inovaţia (eroarea) măsurătorilor de tip GPS admise în faza de corecţie, care poate fi încadrată în gama mai strictă de $\pm\sigma$ (o deviaţie standard) prevăzută înainte de rularea aplicaţiei. Importantă pentru distribuţia erorilor este şi media cu valoare nulă, iar acest lucru se transpune şi în intrarea dată regulatorului ce comandă mişcarea rechinului. Astfel, comanda dată este capabilă să mimeze forma distribuţiei acceleratîei adevărate ale victimei de pe ambele axe, dar cu o varianţă observabil mai

mare, ceea ce însemnă că efortul depus de rechin pentru a prinde foca este semnificativ, iar o astfel de comandă ar putea să nu fie fiabilă pentru un regulator real. Se poate nota de asemenea cum înainte de convergentă generală, eroarea din proces tinde să crească, până când informaţia primită prin măsurători este evaluată.

Acceleraţia pentru acest experiment a fost setată cu o valoare neglijabilă în raport cu poziţia şi viteza iniţială, şi de accea poziţia creşte aproximativ liniar, iar viteza rămâne constantă. Pentru o dinamică mai complexă însă, performanţele algoritmilor se diversifică, după cum va fi evidenţiat şi în experimentele următoare.

Un caz de mişcare circulară este prezentat în figura 6.3, pentru care filtrul Kalman extins doar cu o fază de corecţie liniară a fost folosit ca estimator. O astfel de mişcare presupune o schimbare continuă vitezei şi a giraţiei, şi se poate observa cum şi incertidunea din proces pentru poziţii se modifică din cauza neliniarităţiilor introduse. Se vede cum în acest caz faza de maximizare (predicţie), respectiv cea de minimizare (corecţie) a covarianţei modifică vizibil forma distribuţiei în timp. Orientarea poate fi estimată cu succes datorită informaţiei despre rata de giraţie a procesului integrată în faza de predicţie a modelului, iar varianţa vitezei rămâne constantă din cauza stocasticităţii acceleraţiei necesară producerii virajelor.

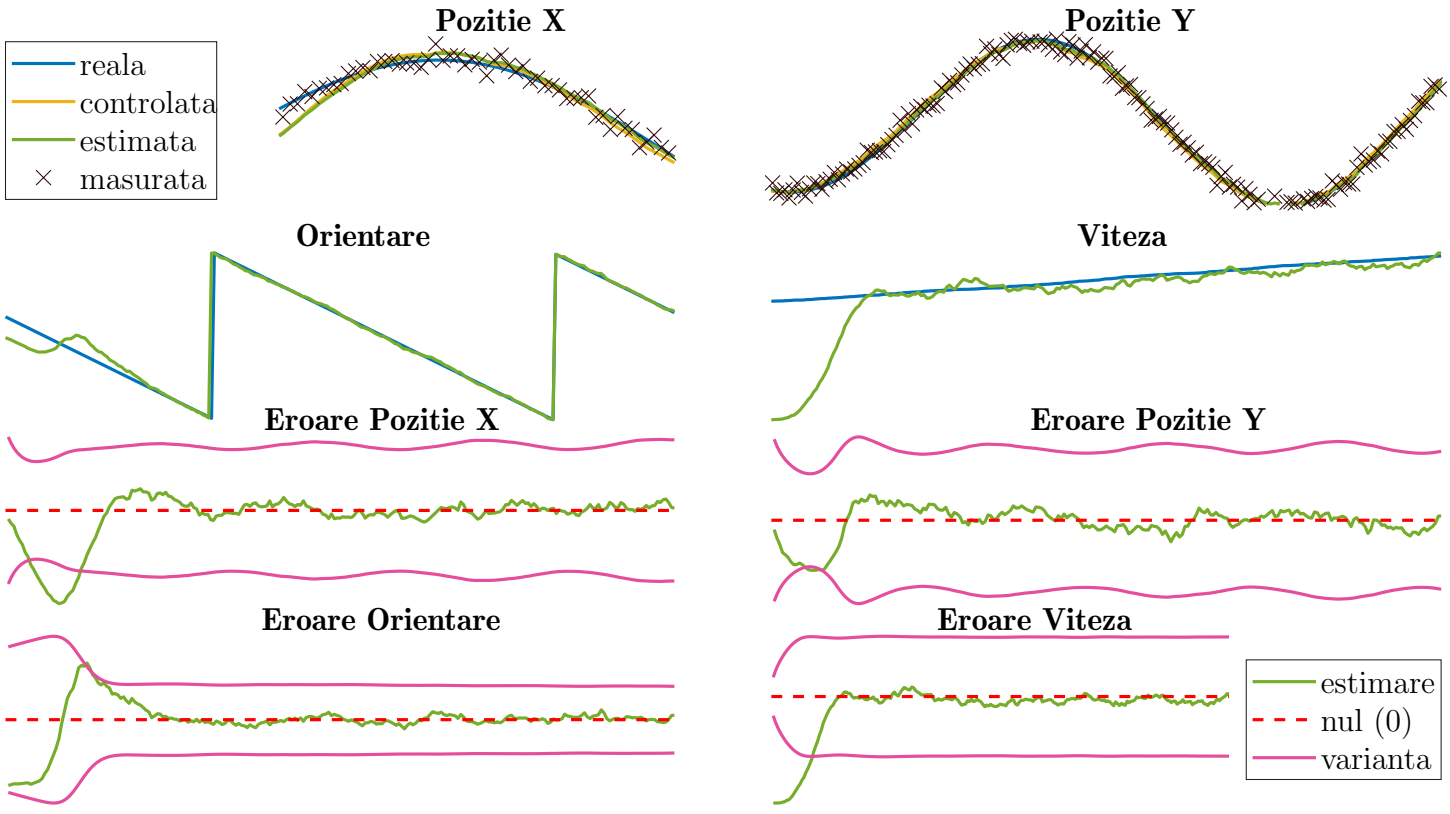


Figura 6.3: Filtrul Kalman extins pentru o traiectorie circulară

Pentru a limita intervalul valabil pentru atitudinea procesului victimă am folosit funcţia MATLAB *wrapTo180* [72] , şi astfel de ficare dată când se termină un cerc complet de mişcare se poate observa pe grafic o linie verticală. Poziţiile axelor X şi Y vor echivala cu funcţiile cosinusoidă, respectiv sinusoidă dependente de magnitudinea acceleraţei.

Îmbunătăţirea majoră introdusă de adăugarea fazei de corecţie neliniare se observă în figura 6.4 în primul rând pentru acurateţea suplimentară regăsită în estimarea orientării. Nu numai că estimarea are o convergenţă mai netedă şi mai rapidă, dar şi varianţa măsurătorii este minimizată considerabil prin introducerea noului senzor. Conform ecuaţiilor fazei de corecţie extinse (4.137) - (4.141), se poate observa că matricea de covarianţa este minimizată indiferent de convergenţa vectorului de stare, ceea ce înseamnă că atât execuţia algoritmului, cât şi acordarea a priori au fost de succes.

Şi de această dată fazele succesive de maximizare, respectiv minimizare ale varianţelor pozitîilor pot fi obsevate mai clar în cele două grafice, faţă care estimarea converge independent cu mai putînă incertidune la stare. Modelul procesului urmăritorului este în continuare unul liniar, şi de aceea performanţele lui nu sunt comparate pentru noile stări prezente în figură, ci decât pentru stările de interes final, poziţiile, pentru care se observă din nou o apropiere mai bună decât cea obţinută de măsurătorile neprocesate.

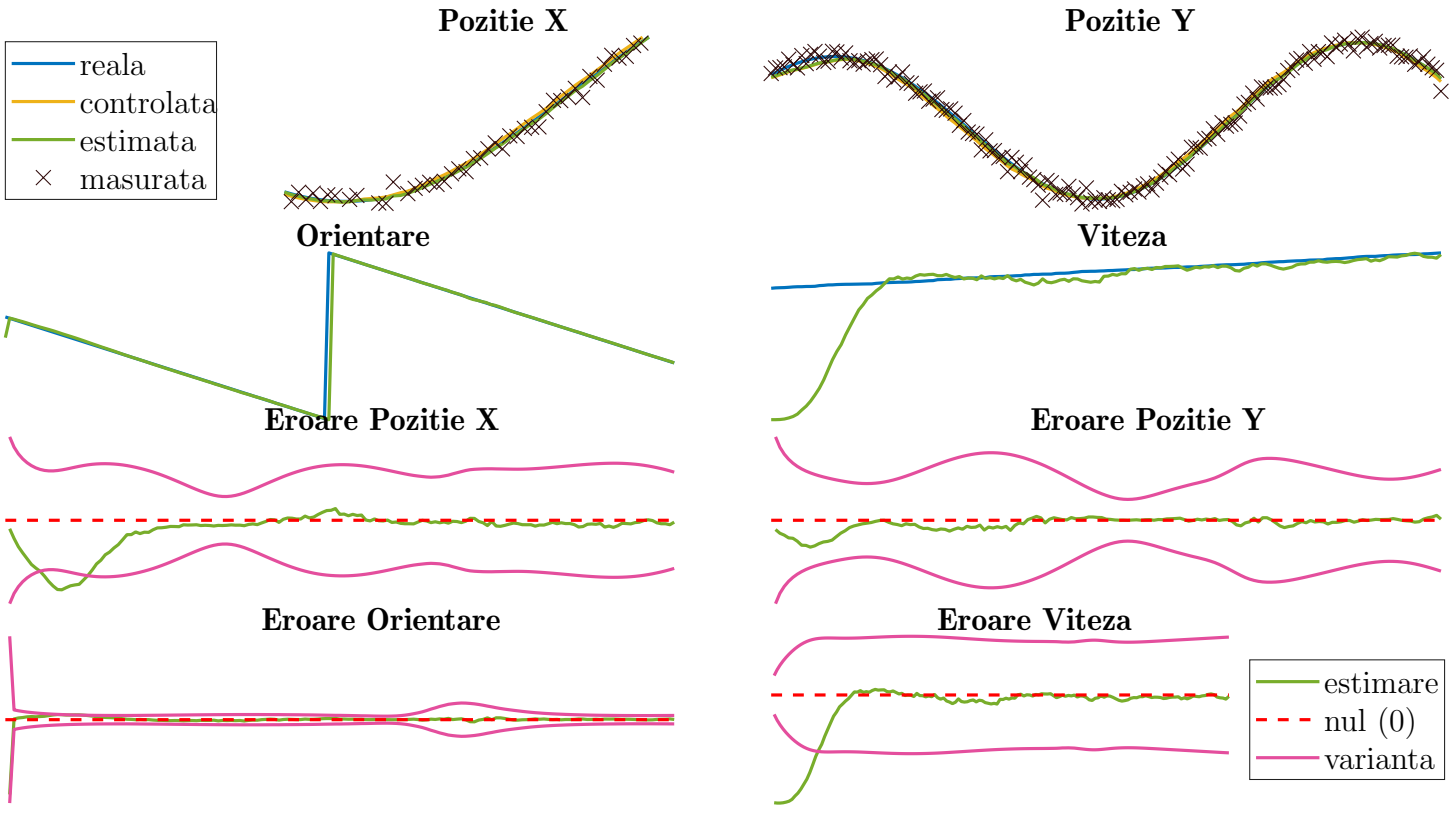


Figura 6.4: Filtrul Kalman extins+ SoNAR pentru o traiectorie circulară

Orientarea şi viteza par să fie estimate la fel de bine şi fără ajutorul SoNAR-ului, dar, la o privire mai atentă, se poate constata că în al figura de sus doar o singură linie verticală este prezentă în graficul atitudinii, faţă de cele două din graficul precedent. Acest lucru înseamnă ca foca a reusît să efectueze un singur cerc complet înainte să piardă toate punctele de viată prin prindere (convergenţa estimatorului şi a regulatorului), faţă de cele recordul anterior de două rotaţii complete.

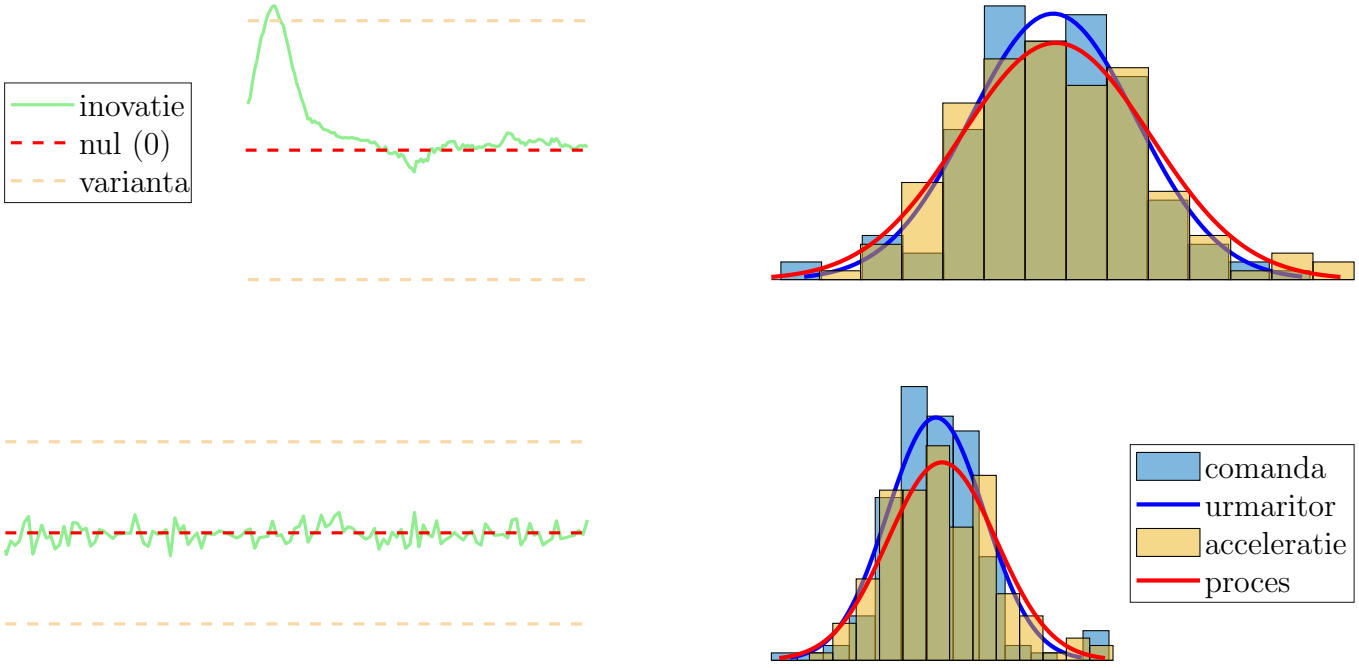


Figura 6.5: Filtrul Kalman + SoNAR extins pentru o traiectorie circulară

Probabil cea mai pertinentă concluzie poate fi trasă prin analizarea figurii 6.5, unde, cu ajutorul celor patru grafice de performanţă, se observă cum distribuţia comenzii regulatorului o imită aproape perfect pe cea reală a acceleraţiei aleatorii a procesului. De aici rezultă că rechinul nu depune un efort considerabil mai mare decât este necesar pentru a prinde foca, iar graficele din stânga figurii întăresc şi mai mult această convingere prin expunerea erorilor. Inovaţia măsurătorilor este şi de această dată zero-centrică, distribuită Gaussian, însă chiar mai mult, în plus faţă de rezultatul anterior, ea se regăseşte foarte strict sub limita procesată în matricea S a covarianţei erorilor, calculată prin consideraţia iniţială a zgomotului de măsură R. Acest aspect relevă rolul vital jucat de măsurătorile exacte ale unui sonar comparativ cu cele foarte zgomotoase ale unui senzor de tip GPS în creşterea acurateţii estimărilor.

Acest scenariu este probabil unul dintre cele mai bune care pot fi obţinute pentru o traiectorie neliniară, însă procesul este în continuare deterministic prin natura sa, chiar dacă factorii structurali sunt stocastici. Un experiment mai aproapiat de realitate trebuie să permită mişcării neliniare să iasă din repetitivitatea rotirii în cerc.

Am simulat un astfel de model de mişcare pentru figura 6.6, unde se poate observa că orientarea obiectului nu mai este liniară nici măcar pe părti, şi tocmai de aceea am ales să aplic transformata unscented estimatorului. Pentru că filtrele neliniare nu pot garanta convergenţa, a trebuit să acordez parametrii specific pentru acest tip de mişcare, ceea ce ar putea însemna că o structură generală nu poate fi oferită pentru toate mişcările posibile.

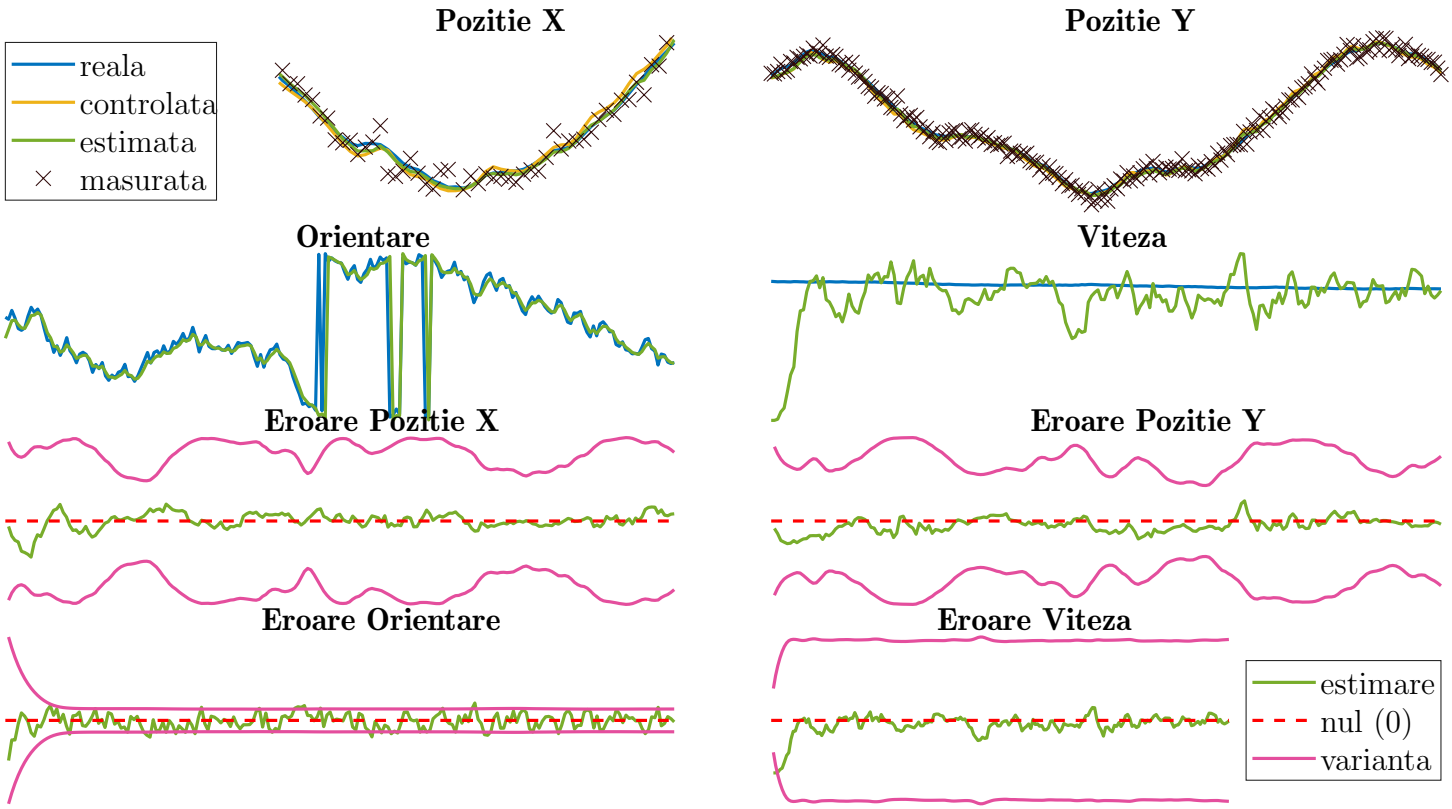


Figura 6.6: Filtrul Kalman unscented + SoNAR pentru o traciectorie aleatorie

După cum se observă, valoarea neglijabilă, cvasinulă a acceleraţiei determină un comportament aproape constant al velocităţii, dar neliniaritatea puternică provine din schimbările bruşte de orientare. Acest exemplu nu ar fi convers (nu într-o perioadă de timp rezonabilă) fără intervenţia sonarului în faza de corecţie neliniară. Ţinând cont că modelul mişcării în cauză prezintă cel mai înalt grad de stocasticitate, etapele de fluctuaţia ale varianţelor poziţiilor sunt şi mai evidente decât în cazurile precedente. Evidentă în acest caz este şi stabilitatea neclară a vitezei estimate comparativ cu cea adevărată, acest lucru fiind datorat creşterii zgomotului de proces prin acordarea filtrului.

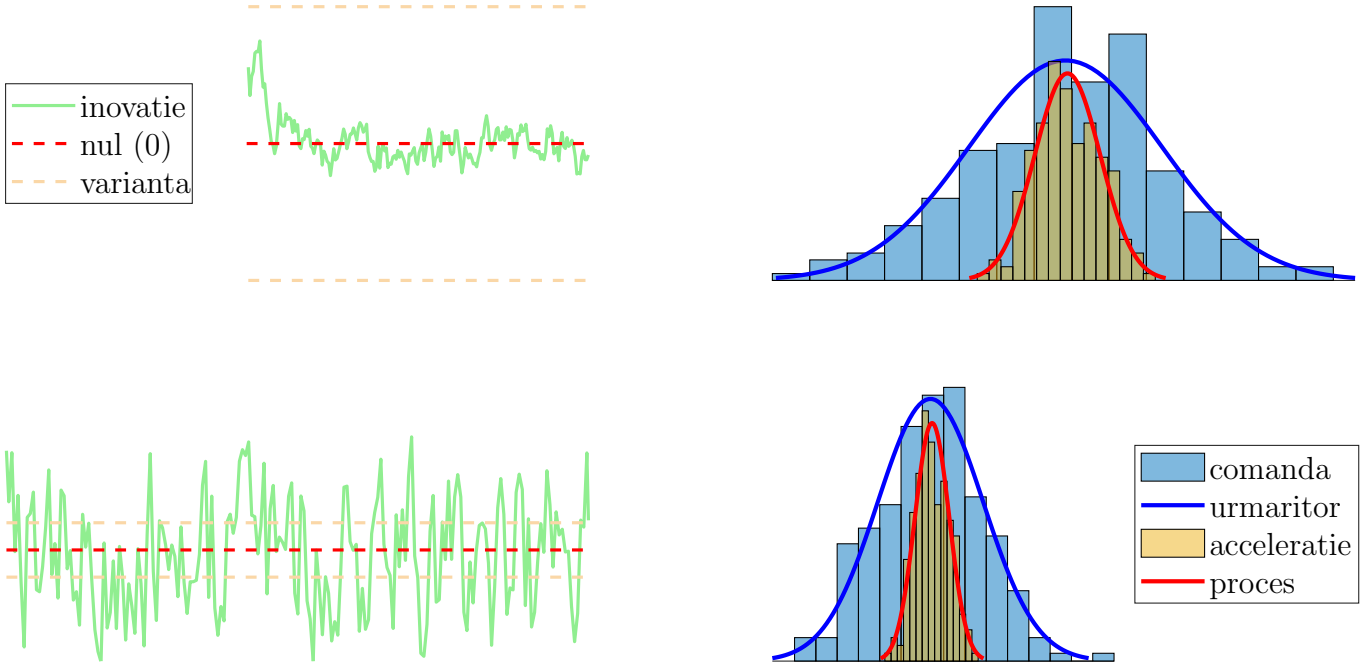


Figura 6.7: Filtrul Kalman unscented + SoNAR pentru o traiectorie aleatorie

La fel de importante sunt şi concluziile ce se pot trage din graficele puse la dispoziţie în figura 6.7 Deşi aspectul Gaussian este recuperat de distribuţia comenzii regulatorului procesului urmăritor, distributîa în sine este mai lată, ceea ce denotă o abatere mai mare a comenzilor faţă de medie, făcând controlul dificil de implementat în practică. De data acesta, erorile măsurătorilor de pe axa Y nu au fost bine încadrate în varianţa calculată pentru inovaţie, iar acest lucru se transpune în efortul pe care urmăritoul îl depune pentru a prinde victima. Filtrul în cauză mai are nevoie să fie acordat, sau chiar modificat pentru a face faţă neliniarităţilor puternice prezente în proces.

Acest ultim exemplu de traiectorie este şi cel relevant pentru mişcarea pe care o poate comanda un jucător, întrucât se aşteaptă ca aceasta să fie oricât de neliniară posibil, şi de aceea, atât prin privirea aspectului teoretic, cât şi prin demonstraţiile grafice din acest capitol, consider modelul extins augmentat al filtrului Kalman unscented cel mai fiabil pentru joc în sine.

Pentru algoritmul de fuziune al datelor de la giroscop şi accelerometru am realizat o comparaţie cu funcţia MATLAB preimplementată *imufilter* [73] care porcesează şi fuzionează măsurătorile celor doi senzori. Am constatat că filtrul propriu rezolvă o problemă de latenţă în convergentă estimărilor. Acest lucru se datorează faptului că funcţia

oferită de toolbox-ul *Sensor fusion and Tracking* aplică un filtru dur rezultatelor pentru a garanta minimizarea erorii de estimare în favoarea vitezei.

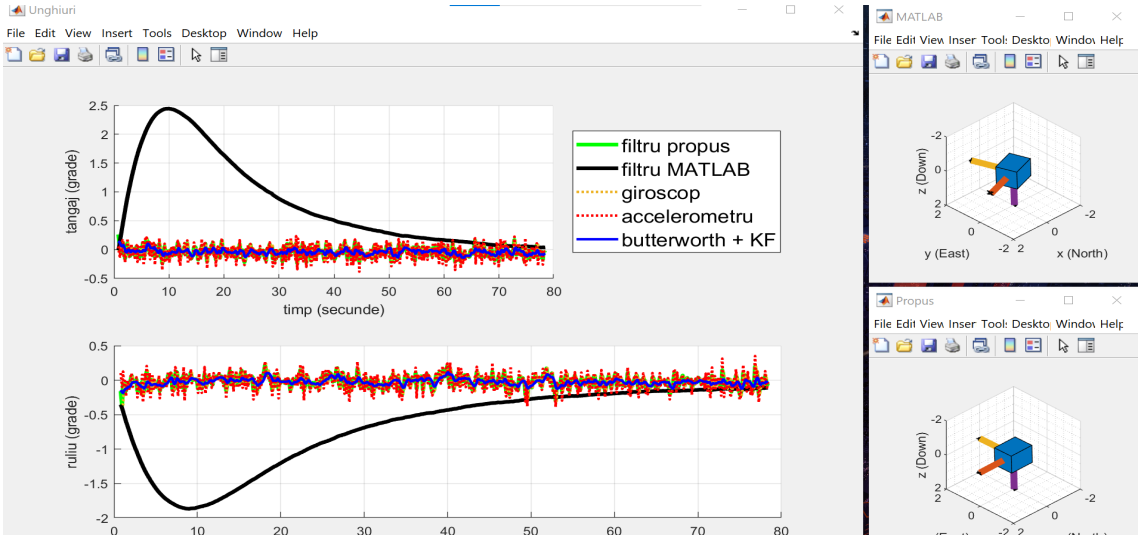


Figura 6.8: Fuziunea senzorială prin Kalman comparată cu măsurătorile individuale

Se poate vedea în figura 6.8 cum toate măsurătorile senzorului static converg la valoarea de 0, însă în cazul funţiei *imufilter* acest lucru durează mai mult de un minut. Acest aspect este desigur inadmisibil pentru o aplicaţie de timp real, cum este şi cazul jocului dezvoltat,şi de aceea am ales să folosesc implementarea proprie.

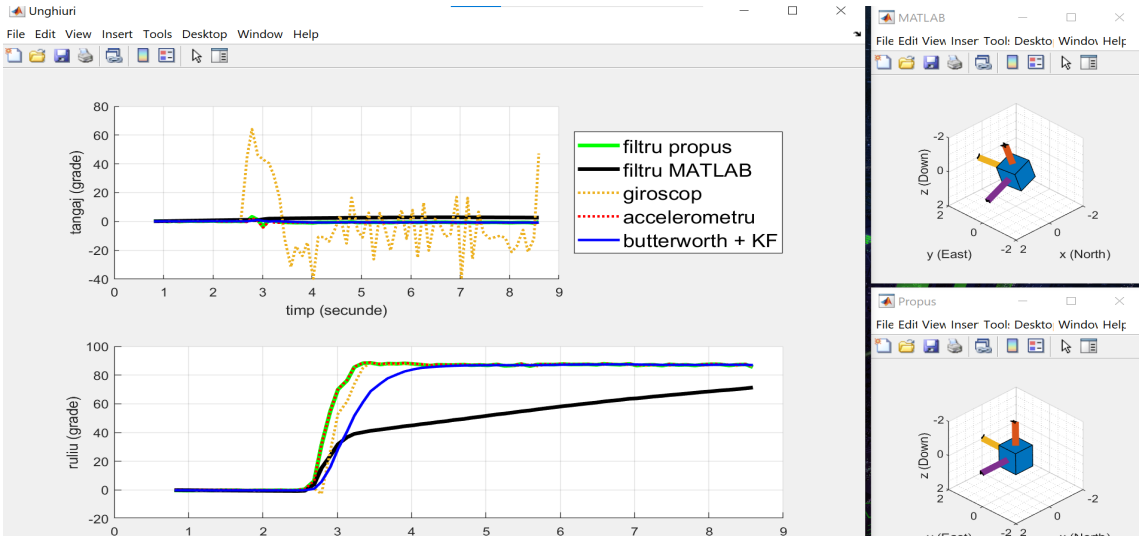


Figura 6.9: Fuziunea senzorială prin Kalman comparată cu măsurătorile individuale

Dezavantajul posibil obţinut prin propria implementare este că fără un filtru suplimentar, orientarea poate fi greşit estimată din cauza zgomotului senzorilor. În cazul

figurii 6.9, se poate observa cum pentru o rotaţie bruscă pe axa unghiului de ruliu, giroscopul a integrat viteza de rotaţie mare măsurată şi astfel estim;arile sale sunt eronate pentru axa de tangaj faţă de care senzorul era static. Această problemă poate fi rezolvată prin acordarea unei mai mari încrederi în măsurătorile accelerometrului prin diminuarea abaterilor matricei zgomotului de măsură R, sau similar prin mărirea elementelor matricei zgomotului de proces Q, ce modelează erorile giroscopului. Astfel se obţine un rezultat satisfăcător, pentru care am aplicat un filtru propriu Butterworth, însă pe care am ales să nu îl folosesc în implementarea finală, dată fiind întărzierea de o secundă observabilă în figură.

Am comparat concret rezultatele obţinute pentru senzorul static cu fiecare metodă folosind funcţia *immse* [74] din toolbox-ul de procesare de imagini pus la dispoziţie de MATLAB şi am realizat astfel tabelul 6.1. Se poate constata că într-adervăr, datorită filtrării suplimentare, funcţia *imufilter* este mai exactă. În afară de acest aspect, se poate vedea că fuziunea proprie a celor două măsurători independente de la giroscop şi accelerometru oferă o estimare mai puţin eronată decât oricare senzor evaluat separat, ceea ce însemnă că scopul algoritmului a fost îndeplinit.

Eroarea medie pătratică			
Metodă	Fază	Ruliu	Tangaj
Giroscop - Integrare numerică	Predicţie	0.0342	0.0081
Accelerometru - Înclinaţie	Corecţie	0.0373	0.0141
Filtru Kalman cuaternioni	Fuziune	0.0307	0.0076
imufilter (MATLAB + toolbox)	Fuziune	0.0054	0.00065
Filtru propus + Butterworth	Filtru IIR	0.0255	0.0022

Tabelul 6.1: Analiză performanţe

Capitolul 7

Manual de Instalare şi Utilizare

7.1 Resurse hardware

Pentru realizarea joystick-ului au fost folosite un microcontroller Arduino în combinaţie cu un modul de senzori MPU-6050 (accelerometru şi giroscop triaxiale).

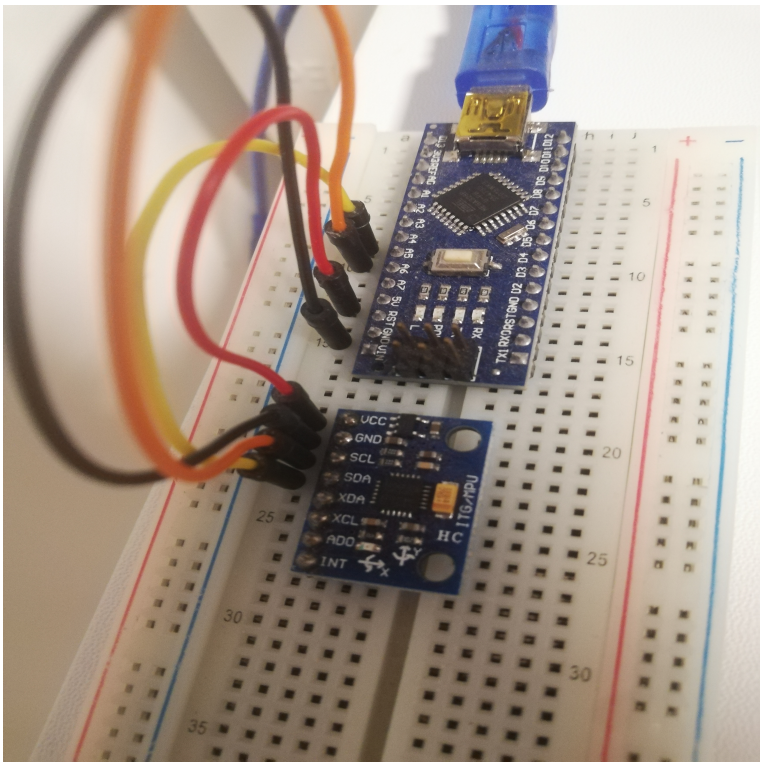


Figura 7.1: Configuraţie pini

Configuraţia se realizează conform figurii 7.1 şi astfel se alcătuiesc conexiunile:

Arduino	MPU-6050
A5	SCL
A4	SDA
VCC	5V
GND	GND

Tabelul 7.1: Conexiune

7.2 Resurse software

Pentru implementarea aplicaţiei a fost folosit programul MATLAB (versiunea 2020b) cu limbajul de programare intrinsec. Toolbox-urile extra folosite sunt *Control System Toolbox* pentru funcţiile *lqr* şi *place*, *Statistics and Machine Learning Toolbox* pentru vizualizarea histogramelor, pachetul *MATLAB Support Package for Arduino Hardware* pentru conexiunea cu placa [75] şi senzorii [76]. Opţional pot fi folosite resursele din *Sensor Fusion and Tracking Toolbox* pentru vizualizarea 3D a orientării şi *Image Processing Toolbox* pentru facilizarea calculării erorii medie pătratice. Restul funcţionalităţilor sunt implementai proprii realizate doar cu pachetul individual pus la dispoziţie şi concepute cu ajutorul resurselor bibliografice.

Pentru rularea aplicaţiei este necesar ca scripturile de jos să fie în acelaşi director:

- QuaternionKalmanFilterModel.m
- KalmanFilterModel.m
- VehicleModel2D.m
- ExtendedKalmanFilterModel.m
- ExtendedVehicleModel2D.m
- UnscentedKalmanFilterModel.m
- UnscentedVehicleModel2D.m
- SharkDrawer.m
- SealDrawer.m
- SeagullDrawer.m
- TrackerUI.m
- ObjectTracker.m

ObjectTracker.m este script-ul principal şi singurul care trebuie apelat direct de către utilizator. *TrackerUI.m* este funcţia responsabilă pentru realizarea interfeţei grafice, în timp ce fişierele de tip *AnimalDrawer.m* desenează vectorial şi integrează cele trei personaje în animaţia principală. Clasele filtrelor Kalman şi ale diferitelor modele de mişcare sunt cele introduse în figura 5.6

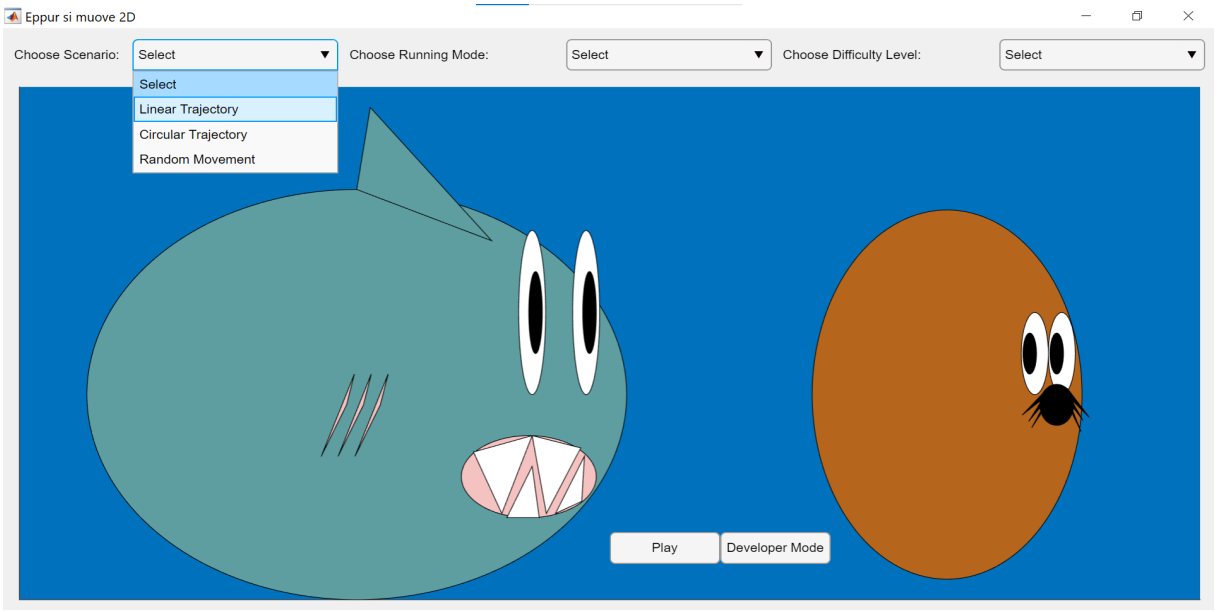


Figura 7.2: Alegere scenariu

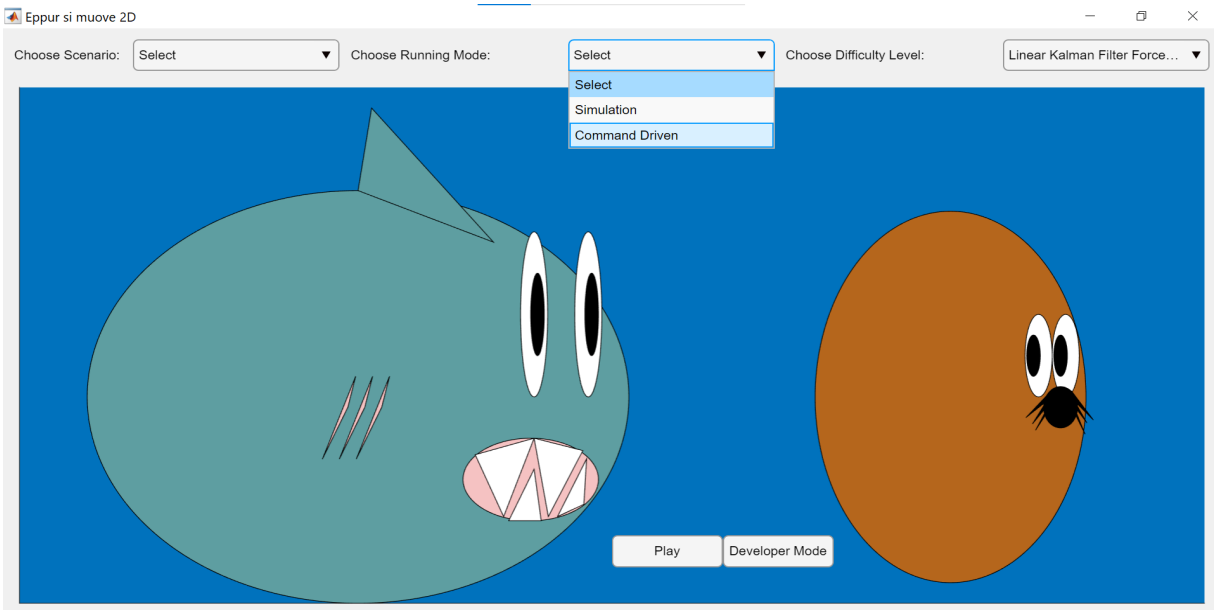


Figura 7.3: Alegere modalitate de joc

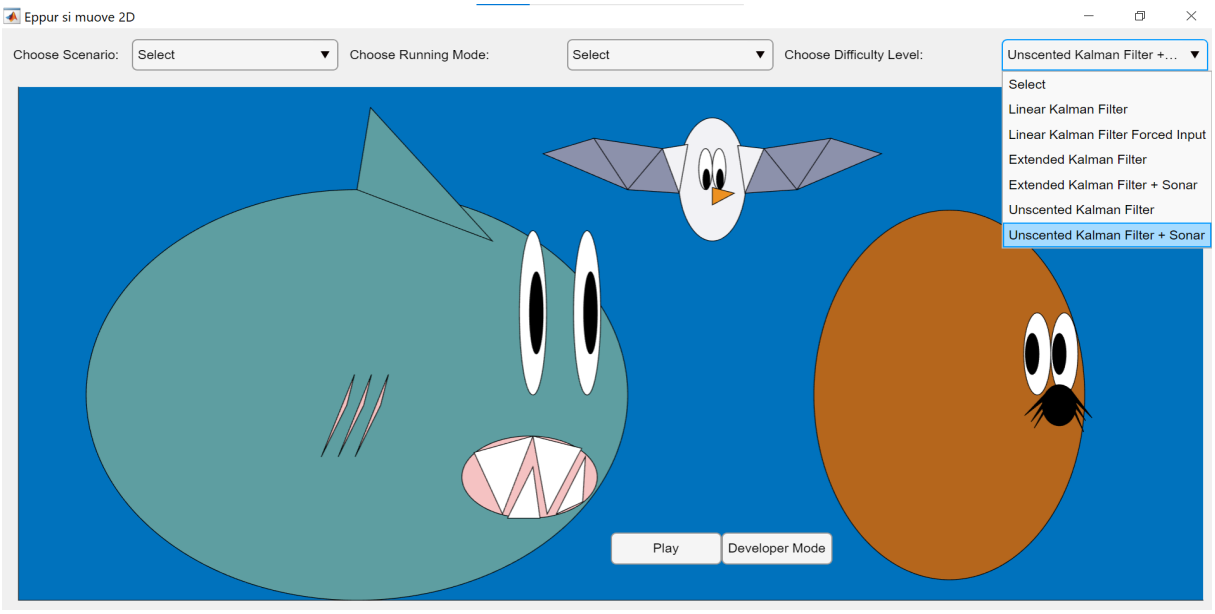


Figura 7.4: Alegere estimator

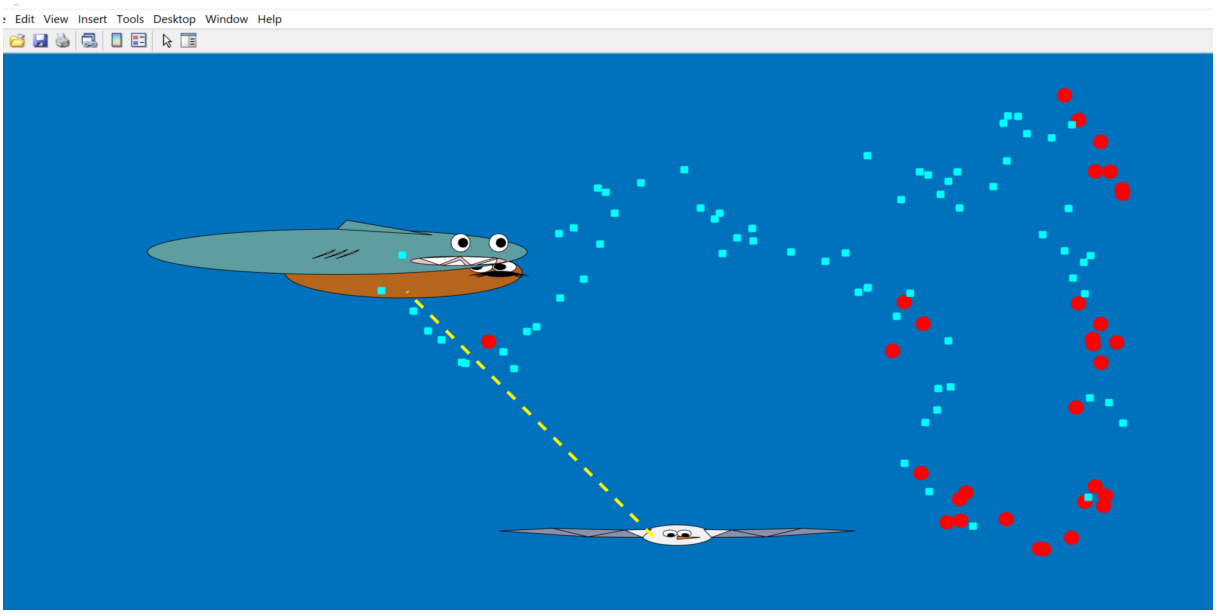


Figura 7.5: Desfăşurare urmărire

Figurile 7.2, 7.3 şi 7.4 prezintă opţiunile oferite utilizatorului de interfaţa grafică. Figura 7.5 prezintă o captură a animaţiei din timpul urmăririi.

Capitolul 8

Concluzii

În încheierea acestei lucrări, doresc să expun o analiză critică a rezultatelor la care am ajuns până în acest punct, presupusă să întocmească o imagine a dezvoltărilor ulterioare pe care intenţionez să le aduc proiectului prezent. Voi recapitula deci contribuţiile personale şi rezultatele obţinute în scopul sintetizării lucrării, dar şi în vederea viitoarelor îmbunătăţiri.

8.1 Contribuţii

Am încercat prin prezenta lucrare o punere în practică a diverselor noţiuni teoretice regăsite în lista bibliografică referitoare la algoritmii de estimare şi fuziune senzorială. Am oferit în acest sens o implementare personală în MATLAB pentru modele matematice deja existente în literatură sau în proiecte cu tematică, tehnologii sau limbaje diferite.

Am decis să dau o notă accesibilă conceptelor livreşti prin scenariul ludic gândit pentru aplicaţie şi am descoperit, în acelaşi timp, că un mediu de simulare poate fi folosit şi pentru dezvoltarea unor aplicaţii diverse prin îmbinarea noţiunilor de teorie a sistemelor, controlului şi estimării cu un limbaj de programare de nivel înalt, o interfeţa grafică şi o vizualizare dinamică a performanţelor.

Am oferit o soluţie proprie pentru fuzionarea datelor de la un accelerometru şi un giroscop organizând şi punând cap la cap conceptele teoretice însuşite din mai multe arii de interes. Această soluţie s-a dovedit mai rapidă decât preimplementarea oferită de mediul de dezvoltare ales, dar nu la fel de exactă, şi cu sigurantă nu la fel de complexă.

Am reuşit prin prezenta teză să adun conceptele teoretice pe care le-am învăţat în aceşti ani universitari într-un singur loc, ce pot servi drept suport teoretic implementării unor viitoare aplicaţii.

8.2 Critică

Dezvoltarea aplicaţiei prezente mi-a permis să observ, pe lângă performanţele algoritmilor folosiţi, şi diverse dezavantaje provenite din încercarea de a simplifica problematica pentru a ajunge la un rezultat dorit. Simulările dinamice şi interfaţă animată dezvoltată mi-au permis să observ limitări într-un fel în care nu aş fi putut să o fac doar prin analiză teoretică.

Constat astfel cu un ochi critic că legile de control liniare folosite sunt limitate pentru scenariul prezent, urmăritorul nefăcând faţă de fiecare dată schimbărilor bruşte de mişcare. Acest lucru este agravat de faptul că regulatoarele simulate dispun de nişte performanţe care nici nu ar putea fi implementate în practică prin natura agresivităţii lor.

Analizând partea de estimare, constat cum imii neliniari nu performează pe atât de bine pe cum ar trebui. Deşi într-adevăr convergenţa lor nu este garantată intrinsec, consider că implementări mult mai atente ar putea rezolva unele dintre prezentele probleme, cum ar fi convergenţa covarianţei odată cu starea. Scenariul restrâns bidimensional, de asemenea, devine de la un anumit punct incapabil să reproducă realitatea. Măsurătorile de tip GPS în realitate nu apar pur şi simplu pe hartă, ci sunt recepţionate prin satelit, deci un estimator al lor de obicei funcţionează tot prin filtrarea extinsă a unor coordonate transformate în polar. Modelul point cloud de reproducere al viziunii sonarului este cu mult simplificat faţă de viziunea computerizată oferită de tehnologii dominante în industrie precum senzorii LIDAR.

Pentru modulul de senzori MPU-6050 nu am oferit o estimare a unghiului de giraţie, dar trebuie menţionat că funcţia MATLAB *imufilter* oferă una, chiar dacă nu este chiar exactă. Se pote deduce funcţii trigonometrice pentru unghiul de înclinaţie al accelerometrului şi viteza de rotaţie de pe axa Z a giroscopului poate fi integrată numeric în timp pentru a oferi o informaţie despre acesta, dar rezultatul nu era stabil prin lipsa referinţei.

Poate chiar mai important, şi de interes vital pentru o aplicaţie reală este faptul că senzorii nu prezintă o calibrare riguroasă, ci doar o eliminare a bias-ului printr-o citire offline anterioară. După cum am menţionat şi la începutul lucrării, calibrarea senzorilor inerţiali ar putea constitui o teză în sine şi cu siguranţă reprezintă ceva ce va trebui analizat minuţios în detaliu. Dacă senzorii nu sunt calibraţi corespunzător, un algoritm de fuziune nu ar trebui să intre încă în discuţie.

De menţionat este şi că implementarea proprie a filtrului Kalman bazat pe cuaternioni este liniară, ceea ce înseamnă că estimatorul, deşi garantat să conveargă, nu va capta întreaga dinamică a mişcărilor interceptate de senzori, iar acest lucru ar putea fi uşor depistabil într-o aplcaţie de scală mai largă.

8.3 Dezvoltare ulterioară

Analiza critică a limitărilor prezentei implementări constituie şi punctul de plecare pentru viitoarele îmbunătăţiri ale aplicaţiei pe care le consider necesare în vederea realizării unui proiect de succes complet.

Doresc pe aceasţa cale să încep dezvoltarea unei aplicaţii cu o grafică mai avansată tridimensională, iar pentru acest lucru intenţionez să folosesc medii de dezvoltare specifice jocurilor video (ex: unreal engine) cu limbaje de programare aferente (C_{++}, blueprints) pentru a oferi o perspectivă cât mai aproape de realitate a scenariului unei urmăriri.

Voi avea în vedere pentru imediata dezvoltare legi de control mai avansate, cum ar fi controlul predictiv bazat pe model (MPC) sau controlul prin inteligenţă artificială, iar pentru estimatoare voi încerca diferiţi algoritmi de nivel înalt, cum ar fi filtrul de particule, sau algoritmi dedicaţi pentru SLAM.

Doresc de asemenea să achiziţionez în viitorul apropiat un modul de senzori MPU-9050 ce extinde capacitatea actualului MPU-6050 folosit prin încorporarea unui magnetometru. În acest sens, intenţionez să realizez o estimare completă în spaţiul tridimensional al orientării pentru conceperea unui joystick mai avansat. Acest lucru va impune, bineînteles, şi augmentarea algoritmului prezent de fuziune senzorială prin adgarea unei noi faze de corecţie care să adreseze noile măsrători.

Tot legat de acest aspect, consider că pot explora mai în detaliu avantajele folosirii cuaternionilor pentru reprezentarea orientării, întrucât proiectul de faţă nu a intrat în profunzimea câştigurilor grafice ce pot fi realizate prin tehnici precum SLERP (interpolare sferică) pentru netezirea tranziţiilor dintre imaginile unei animaţii, iar acest lucru este cât se poate de realizabil cu fundamentul teoretic pus la dispoziţie de acestă lucrare.

Cel mai avansat obiectiv pe care doresc să îl realizez e includerea în scenariul urmăririi unor elemente de viziune computerizată, învăţare inteligentă a unui traseu în scopul ocolirii unor obstacole sau eficientizării urmăririi. Doresc astfel să realizez un cadru teoretic complet pentru funcţionarea vehiculelor autonome din actuala industrie emergentă.

Bibliografie

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems,' *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[2] M. S. Grewal and A. P. Andrews, "Applications of kalman filtering in aerospace 1960 to the present [historical perspectives],' *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 69–78, 2010.

[3] M. Thuy and F. Puente Leon, "Non-linear, shape independent object tracking based on 2d lidar data,' in *2009 IEEE Intelligent Vehicles Symposium*, 2009, pp. 532–537.

[4] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems,' *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[5] H. Lee, H. Chae, and K. Yi, "A geometric model based 2d lidar/radar sensor fusion for tracking surrounding vehicles,' *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 130–135, 2019, 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319303908

[6] M. d. F. Coelho, K. Bousson, and K. Ahmed, "An improved extended kalman filter for radar tracking of satellite trajectories,' *Designs*, vol. 5, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2411-9660/5/3/54

[7] M. Wang, C. Xu, C. Zhou, Y. Gong, and B. Qiu, "Study on underwater target tracking technology based on an lstmndash;kalman filtering method,' *Applied Sciences*, vol. 12, no. 10, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/10/5233

[8] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2d lidar,' *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1039–1063, 2015. [Online]. Available: https://doi.org/10.1177/0278364914562237

[9] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation,' *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[10] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation,' in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[11] S. Yang and M. Baum, "Extended kalman filter for extended object tracking,' in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4386–4390.

[12] D. Ravi Kumar, "Hybrid unscented kalman filter with rare features for underwater target tracking using passive sonar measurements,' *Optik*, vol. 226, p. 165813, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402620316363

[13] A. Salcedo-Bosch, F. Rocadenbosch, and J. Sospedra, "A robust adaptive unscented kalman filter for floating doppler wind-lidar motion correction,' *Remote Sensing*, vol. 13, no. 20, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/20/4167

[14] Z. Liu, Y. Chen, and Y. Lu, "Mid-state kalman filter for nonlinear problems,' *Sensors*, vol. 22, no. 4, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/4/1302

[15] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review,' *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140

[16] M. L. Fung, M. Z. Q. Chen, and Y. H. Chen, "Sensor fusion: A review of methods and applications,' in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 3853–3860.

[17] S. Gervais-Ducouret, "Next smart sensors generation,' in *2011 IEEE Sensors Applications Symposium*, 2011, pp. 193–196.

[18] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective,' *Sensors*, vol. 17, no. 10, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/10/2284

[19] P. Gui, L. Tang, and S. Mukhopadhyay, "Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion,' in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, 2015, pp. 2004–2009.

[20] I. Faisal, T. Purboyo, and A. Ansori, "A review of accelerometer sensor and gyroscope sensor in imu sensors on motion capture,' *Journal of Engineering and Applied Sciences*, vol. 15, pp. 826–829, 11 2019.

[21] V. Renaudin, M. H. Afzal, and G. Lachapelle, "New method for magnetometers based orientation estimation,' in *IEEE/ION Position, Location and Navigation Symposium*, 2010, pp. 348–356.

[22] P. Neto, "Kalman filter-based yaw angle estimation by fusing inertial and magnetic sensing: a case study using low cost sensors,' *Sensor Review*, vol. 35, pp. 244–250, 06 2015.

[23] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors,' *Matrix*, vol. 58, 01 2006.

[24] Y.-B. Jia, "Quaternions and rotations *,' 2015.

[25] L. Perumal, "Quaternion and its application in rotation using sets of regions,' *International Journal of Engineering and Technology Innovation (IJETI)*, vol. 1, pp. 35–52, 10 2011.

[26] S. Sarabandi and F. Thomas, "Accurate computation of quaternions from rotation matrices,' in *Advances in Robot Kinematics 2018*, J. Lenarcic and V. Parenti-Castelli, Eds. Cham: Springer International Publishing, 2019, pp. 39–46.

[27] D. Choukroun, I. Bar-itzhack, and Y. Oshman, "A novel quaternion kalman filter,' *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, pp. 174 – 190, 02 2006.

[28] E. Kraft, "A quaternion-based unscented kalman filter for orientation tracking,' in *Sixth International Conference of Information Fusion, 2003. Proceedings of the*, vol. 1, 2003, pp. 47–54.

[29] F. Abyarjoo, A. Barreto, J. Cofino, and F. R. Ortega, "Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors,' in *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*, T. Sobh and K. Elleithy, Eds. Cham: Springer International Publishing, 2015, pp. 305–310.

[30] D. Royo Serrano, "Development of a calibration procedure for gyroscopes in cubesat missions,' Masterş thesis, Luleå University of Technology, Space Technology, 2021.

[31] L. Aguirre, "Controllability and observability of linear systems: some noninvariant aspects,' *Education, IEEE Transactions on*, vol. 38, pp. 33 – 39, 03 1995.

[32] C.-H. Lien, "Robust observer-based control of systems with state perturbations via lmi approach,' *Automatic Control, IEEE Transactions on*, vol. 49, pp. 1365 – 1370, 09 2004.

[33] "Pole Placement,' http://lendek.net/teaching/EC/ch4_poles.pdf, accessed: 2022-06-11.

[34] "Estimarea unei intrari necunoscute,' http://lendek.net/teaching/EC/ec_lab8ro.pdf, accessed: 2022-06-12.

[35] "control.place - Python Control Systems,' https://python-control.readthedocs.io/en/latest/generated/control.place.html, accessed: 2022-06-18.

[36] D. Luenberger, "Observers for multivariable systems,' *Automatic Control, IEEE Transactions on*, vol. AC-11, pp. 190 – 197, 05 1966.

[37] D. Luenberger, "An introduction to observers,' *Automatic Control, IEEE Transactions on*, vol. 16, pp. 596 – 602, 01 1972.

[38] "Decuplarea intrarilor necunoscute,' http://lendek.net/teaching/EC/ec_lab9ro.pdf, accessed: 2022-06-12.

[39] A. Islam, "A comparative study on numerical solutions of initial value problems (ivp) for ordinary differential equations (ode) with euler and runge kutta methods,' *American Journal of Computational Mathematics*, vol. 05, pp. 393–404, 01 2015.

[40] "Multidimensional Kalman Filter,' https://www.kalmanfilter.net/kalmanmulti.html, accessed: 2022-06-13.

[41] P. Rocchi and M. Burgin, "An essay on the prerequisites for the probability theory,' *Advances in Pure Mathematics*, vol. 10, pp. 685–698, 01 2020.

[42] T. Verhoeff, "The laws of large numbers compared,' 05 2003.

[43] S. Kwak and J. Kim, "Central limit theorem: The cornerstone of modern statistics,' *Korean Journal of Anesthesiology*, vol. 70, p. 144, 04 2017.

[44] L. da F. Costa, "Comparing probability densities: A comparative approach,' 05 2022.

[45] "Background Break,' https://www.kalmanfilter.net/background2.html#covExp, accessed: 2022-06-14.

[46] M. Boutahar and D. Pommeret, "Testing for equality between two transformations of random variables,' 10 2011.

[47] "Covariance Extrapolation Equation,' https://www.kalmanfilter.net/covextrap.html, accessed: 2022-06-13.

[48] A. Long, "Linear regression and non-linear regression,' 11 2016.

[49] S. A. van de Geer, *Least Squares Estimation*. John Wiley Sons, Ltd, 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470013192.bsa199

[50] M. Kong, D. Li, and D. Zhang, "Research on the application of improved least square method in linear fitting,' *IOP Conference Series: Earth and Environmental Science*, vol. 252, p. 052158, 07 2019.

[51] J. Romano and M. Wolf, "Resurrecting weighted least squares,' *Journal of Econometrics*, vol. 197, 12 2014.

[52] "Kalman Filter in One Dimension,' https://www.kalmanfilter.net/kalman1d.html, accessed: 2022-06-14.

[53] "Covariance Update Equation,' https://www.kalmanfilter.net/covUpdate.html, accessed: 2022-06-14.

[54] M. Thuy and F. Puente León, "Non-linear multimodal object tracking based on 2d lidar data,' *Metrology and Measurement Systems*, vol. Vol. 16, nr 3, pp. 359–369, 2009.

[55] H. Qi and J. B. Moore, "2 a direct kalman filtering approach for gpwins integration,' 1998.

[56] P. Gibbens and S. Dumble, "Efficient terrain-aided visual horizon based attitude estimation and localization,' *Journal of Intelligent and Robotic Systems*, vol. 78, 03 2014.

[57] D. Inaibo, "Design of extended kalman filter for object position tracking,' *International Journal of Engineering Research and*, vol. V7, 07 2018.

[58] "Using an Extended Kalman Filter for Object Tracking in Simulink,' https://www.goddardconsulting.ca/simulink-extended-kalman-filter-tracking.html, accessed: 2022-06-14.

[59] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i,' *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[60] "First-Hand:The Unscented Transform,' https://ethw.org/First-Hand:The_Unscented_Transform, accessed: 2022-06-17.

[61] "How to draw a covariance error ellipse?' https://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/, accessed: 2022-06-17.

[62] "Pole placement design,' https://www.mathworks.com/help/control/ref/place.html, accessed: 2022-06-18.

[63] "Linear-quadratic state feedback regulator for discrete time state-space systems,' https://www.mathworks.com/help/control/ref/dlqr.html, accessed: 2022-06-18.

[64] N. H. Hughes, "Quaternion to euler angle conversion for arbitrary rotation sequence using geometric methods,' 2009.

[65] "Complete Guide to Rotations and Transformations,' https://www.udemy.com/course/complete-guide-to-rotations-and-transformations/, accessed: 2021-12-20.

[66] Z. Wu, Z. Sun, W. Zhang, and Q. Chen, "Attitude and gyro bias estimation by the rotation of an inertial measurement unit,' *Measurement Science and Technology*, vol. 26, p. 125102, 12 2015.

[67] "MPU-6050 Datasheet,' https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf, accessed: 2022-06-20.

[68] "Butterworth filter design,' https://www.mathworks.com/help/signal/ref/butter.html, accessed: 2022-06-20.

[69] "Advanced Kalman Filtering and Sensor Fusion,' https://github.com/technitute/AKFSF-Simulation-CPP/tree/main/src, accessed: 2022-06-20.

[70] "The Handle Superclass,' https://www.mathworks.com/help/matlab/matlab_oop/the-handle-superclass.html, accessed: 2022-06-20.

[71] "random normal distribution,' https://www.mathworks.com/help/matlab/ref/randn.html, accessed: 2022-06-20.

[72] "Wrap angle to 180 degrees,' https://www.mathworks.com/help/map/ref/wrapto180.html, accessed: 2022-06-20.

[73] "Orientation from accelerometer and gyroscope readings,' https://www.mathworks.com/help/fusion/ref/imufilter-system-object.html, accessed: 2022-06-20.

[74] "Mean-squared error,' https://www.mathworks.com/help/images/ref/immse.html, accessed: 2022-06-22.

[75] "Connection to the Arduino and Arduino-compatible ESP32 hardware,' https://www.mathworks.com/help/supportpkg/arduinoio/ref/arduino.html, accessed: 2022-06-21.

[76] "Connect to MPU-6050 sensor on Arduino hardware I2C bus,' https://www.mathworks.com/help/supportpkg/arduinoio/ref/mpu6050-system-object.html, accessed: 2022-06-21.

Anexa A

Clasa UKF - Implementare în MATLAB

```
classdef UnscentedKalmanFilterModel < handle

    properties
        H
        Q
        R
        state
        covariance
        innovation
        innovation_covariance
        sigma_points
        weights
        augmented_state
        augmented_covariance
    end

    methods

        function [obj] = UnscentedKalmanFilterModel()
            obj.state = [];
            obj.covariance = zeros(4,4);
            obj.innovation = [];
            obj.innovation_covariance = [];
            obj.sigma_points = [];
            obj.weights = [];
            obj.augmented_state = [];
            obj.augmented_covariance = [];
        end

        function [obj] = sigma_generation(obj)
            number_of_states = size(obj.augmented_state,2);
            kappa = 3 - number_of_states;
            root_cov = sqrtm(obj.augmented_covariance);
            sigma0 = obj.augmented_state;
            sigma{1} = sigma0;

            for i = 1 : number_of_states
                obj.sigma_points{i} = obj.augmented_state + sqrt(kappa + number_of_states) * root_cov(i,:);
                obj.sigma_points{number_of_states + i} = obj.augmented_state - sqrt(kappa + number_of_states) * root_cov(i,:);
            end

            for i = 2 : 2 * number_of_states + 1
                sigma{i} =  obj.sigma_points{i-1};
            end

            obj.sigma_points = sigma;
        end

        function [obj] = weight_generation(obj)
            number_of_states = size(obj.augmented_state,2);
            kappa = 3 - number_of_states;
            w0 = kappa / (kappa + number_of_states);
            wi = 0.5 / (kappa + number_of_states);
            obj.weights(1) = w0;

            for i = 2 : 2 * number_of_states + 1
                obj.weights(i)  = wi;
            end
        end
```

```
function [obj] = initialise(obj, accel_std, yaw_std, meas_std, init_on_meas, p_std, v_std, y_std, measurement, varargin)
    obj.H = [1 0 0 0;
        0 1 0 0];
    obj.Q = diag([0.001, 0.001, yaw_std*yaw_std, accel_std*accel_std]);
    obj.R = diag([meas_std*meas_std,meas_std*meas_std]);

    if init_on_meas == false
        obj.state = [0 0 0 0];
        obj.covariance = diag([p_std*p_std,p_std*p_std,y_std*y_std,v_std*v_std]);
    else
        obj.state = [measurement(1) measurement(2) 0 0];
        obj.covariance = diag([p_std*p_std, p_std*p_std, y_std*y_std, v_std*v_std]);
    end
end

function [obj] = prediction_step(obj, time_step, yaw_rate)
    dt = time_step;

    if ~isempty(obj.state)
        x = obj.state;

        P = obj.covariance;

        number_of_states = size(x,2);

        P_aug = blkdiag(P,obj.Q,obj.R);

        obj.augmented_covariance = P_aug;
        obj.augmented_state = [x, 0, 0, 0, 0, 0, 0];

        obj.sigma_generation()
        obj.weight_generation()

        sigma_points_predict = [];

        for i = 1 : size(obj.sigma_points,2)
            px = obj.sigma_points{i}(1);
            py = obj.sigma_points{i}(2);
            psi = obj.sigma_points{i}(3);
            v = obj.sigma_points{i}(4);
            yaw_rate_noise = obj.sigma_points{i}(7);
            accel_noise = obj.sigma_points{i}(8);

            px_upd = px + dt * v * cos(psi) + obj.sigma_points{i}(5);
            py_upd = py + dt * v * sin(psi) + obj.sigma_points{i}(6);
            psi_upd = psi + (yaw_rate_noise + yaw_rate) * dt ;
            v_upd = v + accel_noise * dt ;

            sigma_points_predict{i} = [px_upd, py_upd, psi_upd, v_upd];
        end

        x_predict = zeros(1, number_of_states);
        for i = 1 : size(sigma_points_predict,2)
            x_predict = x_predict+ obj.weights(i) * sigma_points_predict{i};
        end

        P_predict = zeros(number_of_states,number_of_states);

        for i = 1 : size(sigma_points_predict,2)
            deviation = sigma_points_predict{i} - x_predict;
            P_predict = P_predict + (obj.weights(i) * (deviation' * deviation));
        end

        obj.state = x_predict;
        obj.covariance = P_predict;
    end
end

function [obj] = update_step_linear(obj, measurement)

    if ~isempty(obj.state) && ~isempty(obj.covariance)
        x = obj.state;
        P = obj.covariance;
        H = obj.H;
        R = obj.R;

        z = [measurement(1),measurement(2)];
        z_hat = H * x';
        z_hat = z_hat';

        y = z - z_hat;
        S = H*P*H' + R;
        K = P*H'/S;

        x_update = x + (K * y')';
```

```
            P_update = P - K*H*P;

            obj.innovation = y;
            obj.innovation_covariance = S;
            obj.state = x_update;
            obj.covariance = P_update;

        else
            obj.state = ([measurement(1), measurement(2),0,0]);
            obj.covariance = diag([obj.R(1,1),obj.R(2,2),0.5,0.5]);
        end
    end

    function [obj] = update_step(obj, sight_x, sight_y, range, heading, range_std, heading_std)

        if ~isempty(obj.state) && ~isempty(obj.covariance)

            x = obj.state;

            P = obj.covariance;

            z = [range,heading];

            R_polar = diag([range_std*range_std; heading_std*heading_std]);

            number_of_states = size(x,2);
            noise_vars = 2;

            P_aug = blkdiag(P,obj.Q,R_polar);
            obj.augmented_covariance = P_aug;
            obj.augmented_state = [x, 0, 0, 0, 0, 0, 0];
            obj.sigma_generation()

            z_sigma = [];

            for i = 1 : size(obj.sigma_points,2)
                px = obj.sigma_points{i}(1);
                py = obj.sigma_points{i}(2);
                psi = obj.sigma_points{i}(3);
                range_noise = obj.sigma_points{i}(9);
                heading_noise = obj.sigma_points{i}(10);

                dx = sight_x - px;
                dy = sight_y - py;

                line_of_sight = sqrt(dx^2 + dy^2) + range_noise;
                orientation = atan2(dy,dx) - psi + heading_noise;

                z_sigma{i} = [line_of_sight, orientation];
            end

            z_mean = zeros(1,2);

            for i = 1 : size(z_sigma,2)
                z_mean = z_mean + obj.weights(i)* z_sigma{i};
            end

            S = zeros(noise_vars,noise_vars);

            for i = 1 : size(z_sigma,2)
                deviation = z_sigma{i} - z_mean;
                S = S + obj.weights(i) * (deviation' * deviation);
            end

            crossCov = zeros(number_of_states,noise_vars);

            for i = 1 : 2 * 4 + 2
                state_dev = obj.sigma_points{i}(1:4) - x;
                meas_dev = z_sigma{i} - z_mean;
                crossCov = crossCov + obj.weights(i) * state_dev' * meas_dev;
            end

            K = crossCov/S;
            y = z - z_mean;
            x_update = x + (K * y')';
            P_update = P - K*S*K';

            obj.innovation = y;
            obj.innovation_covariance = S;
            obj.state = x_update;
            obj.covariance = P_update;
        end
    end
  end
end
```

Anexa B

Calibrarea giroscopului prin învăţare automată

Întrucât nu am epuizat subiectul calibrării senzorilor folosiţi în lucrarea propriu-zisă, am decis să dedic această anexă pentru redijarea complexului fenomen, deoarece consider că întelegerea lui nu poate fi privită unilateral fată de algoritmii de fuziune senzorială.

Măsurătorile senzorilor giroscopici sunt afectate de temperatură [1], iar acest lucru provocă fenomenul de *drift* menţionat la începutul lucrării. De obicei, dacă temperatura senzorului sau a plăcii de care este ataşat creşte constant, atunci şi *drift-ul* va creşte constant, şi poate fi aproximat simplu printr-o regresie liniară (4.90) - (4.102), sau integrat în vectorul de stare al filtrului Kalman pentru a fi estimat odată cu orientarea.

Doresc să prezint însă în această parte a lucrării un caz neliniar de creştere a temperaturii ce produce ridicai şi scăderi în media măsurătorilor pentru o perioadă mai lungă de timp. Am realizat astfel un experiment prin pornirea şi oprirea unui ventilator extern pentru laptopul la care erau conectate placa de dezvoltare şi modulul de senzori şi am înregistrat datele citite de la senzori pentru 30000 de iteraţii (aproximativ 60 de minute).

Se poate constata cum măsurătorile giroscopului de pe axa X au fost deviate în figura B.1. Măsurătorile accelerometrului în schimb nu au fost deloc afectate de schimbarea de temperatură pentru nicuna dintre cele trei axe. Pentru că zgomotul prezent era mult prea mare, am aplicat setului de date un filtru de mediere pentru o fereastră de ± 10 valori, şi am testat experimentul doar pentru axa ruliu:

$$f(X) = \frac{\sum_{i=t-10}^{t+10} X_i}{2 \times 10 + 1} \tag{B.1}$$

[1] Nez, Alexis Fradet, Laetitia & Laguillaumie, Pierre Monnet, Tony Lacouture, Patrick. (2018). Simple and efficient thermal calibration for MEMS gyroscopes. Medical Engineering Physics. 55.10.1016/j.medengphy.2018.03.002.

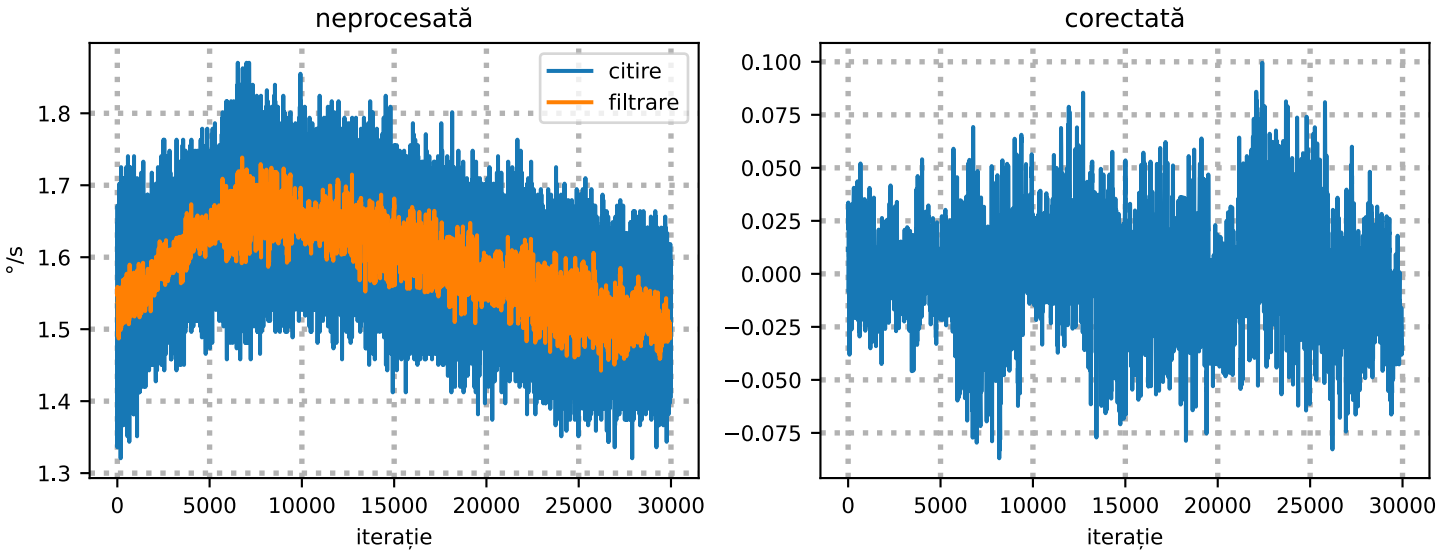


Figura B.1: Corecţie măsurători

Am proiectat apoi pentru datele filtrate o reţea neuronală cu ajutorului submodulului *nn* din PyTorch folosind funtii de activare liniare de bază şi funcţia ReLU [2] (Rectified Linear Unit) pentru a putea descoperi neliniarităţiile în propagarea reţelei înapoi.

Pentru ca reţeaua să fie capabilă să mimeze toate iregularităţile prezente a fost nevoie de introducere unui număr mare de neuroni pentru straturile ascunse, şi am găsit empiric că un număr potrivit este de 25 de neuroni ca lătîme pentru o adâncime de două straturi suplimentare. Am conceput astfel reţeaua neuronală conform figurii B.2 [3]:

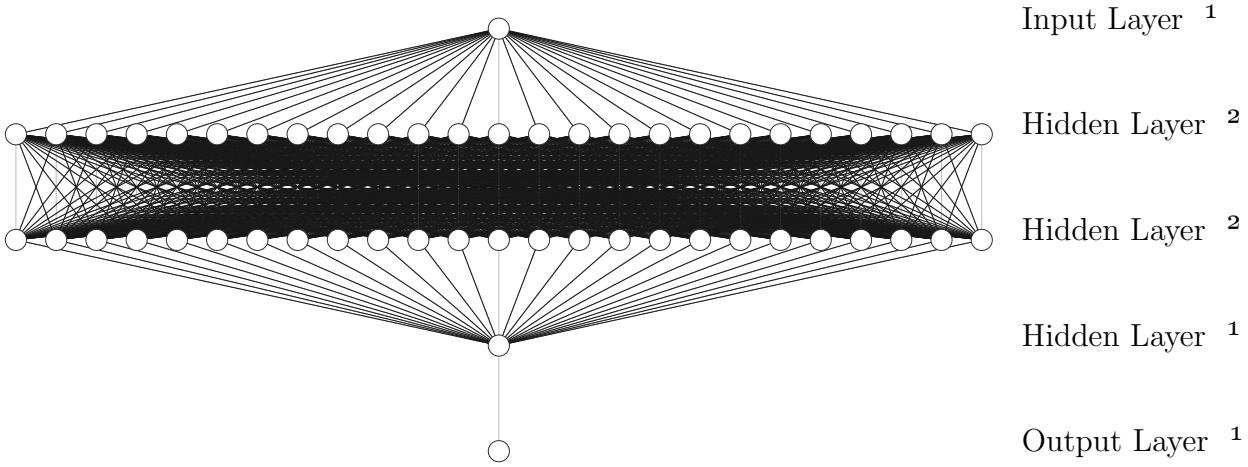


Figura B.2: Reţea neuronală

[2] https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html
[3] https://alexlenail.me/NN-SVG/index.html

Unde penultimul strat ascuns format dintr-un singur neuron are doar rolul să mai treacă o dată rezultatul prin activarea neliniară înainte de procesarea finală. Astfel, pentru un neuron de la intrare (pentru un element din vectorul de date) se repetă propagarea până la obţinerea rezultatului (neuronul de ieşire, valoarea punctului transformată).

Deoarece corelaţia dintre doi vectori poate fi obţinută prin calcularea erorii dintre fiecare element asociat unei iteraţii specifice, am ales să folosesc funcţie *MSELoss* [4] ce returnează erorea medie pătratică dintre vectorul iniţial şi rezultatul procesat. Astfel, am proiectat algoritmul în felul următor:

Algorithm 1 Propagare reţea neuronală (Pseudo-Python)

Require: x = vector iteraţii, y = vector valori giro
Ensure: $y, x = float \wedge y, x = torch.tensor$
Ensure: MSELoss()
Ensure: $nn(Linear(1, 25), ReLU(), Linear(25, 25), ReLU(), Linear(25, 1), ReLU(), Linear(1, 1)$
Ensure: Optimizator = Stochastic Gradient Descent
  $pas \leftarrow 0.05$
  $epoci \leftarrow 700$
  for (epocă din epoci) **do**
    $\hat{y} \leftarrow nn(x)$
    MSELoss($\hat{y}, y$)
    MSELoss.backward()
    Optimizator.gradient_zero()
    Optimizator.propagare(pas)
  end for

Pentru care *pas-ul* reprezintă valoarea cu care înaintează funcţia de gradient (optimizatorul) pentru minimizarea erorii pătratice, iar funcţiile de tip *backward* [5], *gradient_zero* [6] şi *propagare* [7] reprezintă faza de evaluare inversă a reţelei în scopul calculării ponderilor prin funcţiile de activare neliniare ReLU. Aceste ponderi rezultate sunt cele care decif magnitudinea răspunsului final pentru fiecare epocă şi astfel se obţine un vector ca în figurra B.3, în al cărei prim grafic se observă că valorile tensorului iniţial au fost extrapolate corect de reţea cu o valoare de corelaţie de 89% [8] între adevăr şi predicţie.

A doua parte a figurii prezintă minimizarea funcţiei de cost (eroarea medie pătratică atinge valoarea minimiă posibilă) şi se constată cum un număr mai redus de epoci ar putea fi suficient pentru antrenarea reţelei, întrucât scăderea nu mai este drastică după primele 200 de epoci.

[4] https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html
[5] https://pytorch.org/docs/stable/generated/torch.Tensor.backward.html
[6] https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.zero$_g rad.html$
[7] https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.step.html
[8] https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html

Rezultatul final obţinut prin scăderea fomei recuperate din datele reale se regăseşte în al doilea grafic din figura B.1 şi se poate constata că astfel s-a redus şi bias-ul static, dar şi evoluţia driftului.

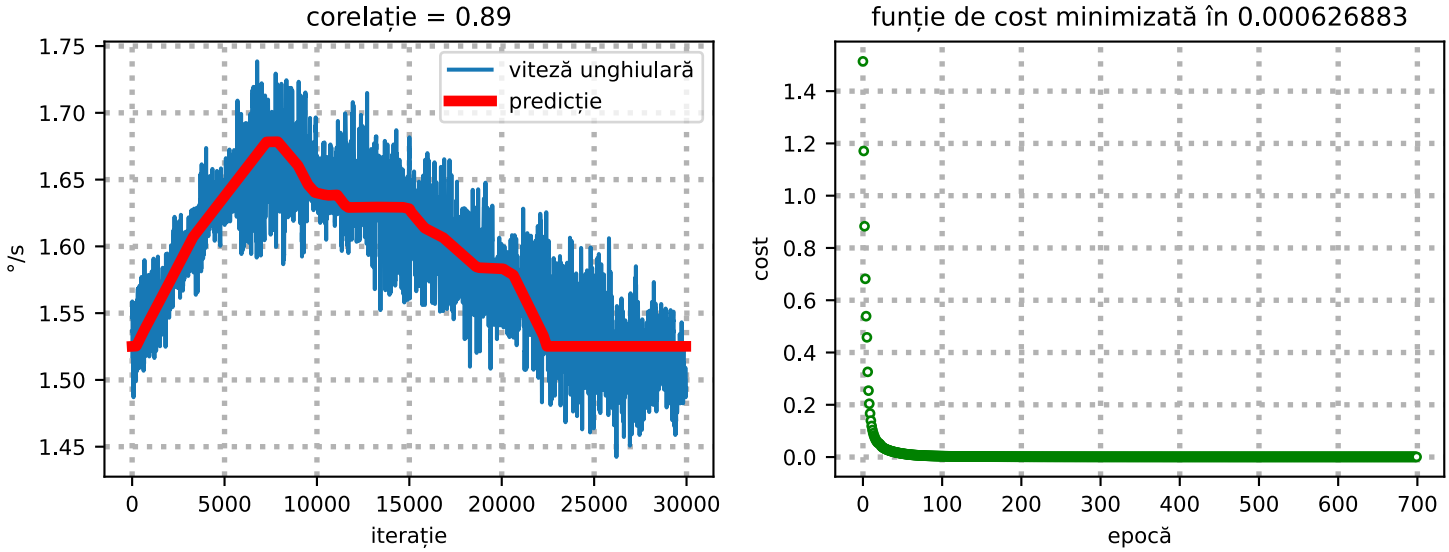


Figura B.3: Predicţie reţea neuronală

Desigur, pentru un astfel de exemplu trebuie menţionat că o aproximare polinomială ar fi funcţionat probabil mai bine şi mai eficient, fără a avea nevoie de timp de antrenare, şi fără a putea oferi rezultate greşite. Un model matematic bazat pe un proces va fi mereu de preferat în locul unuia bazat pe învăţare, dacă procesul este determinist.

Cu toate acestea, consider că acest tip de metode de calibrare deschide noi perspective pentru senzorii inerţiali, întrucât extrapolarea inteligentă a unui semnal se poate adapta şi în cazul în care acesta îsi schimbă comportamentul într-un mod stocastic.

Deoarece acest comportament al măsurătorilor senzorului a fost obţinut într-un timp de lungă durată comparativ cu minutele de obicei alocate unui joc, şi pentru că obţinerea drift-ului a necesitat intervenţie exterioară prin modificarea activă a temperaturii, acest model nu a fost inclus în aplicaţia finală, dar doresc, prin încheierea acestei anexe, să pun bazele unei posibile dezvoltări ulterioare a unui algoritm de estimare a orientării bazat pe inteligenţă artificială.

Anexa C

Lucrări publicate

A. Moraru, ”Quaternion Based Kalman Filter for Open Loop Attitude Estimation”, AQTR Student Forum, p. 29, Aprilie 2022