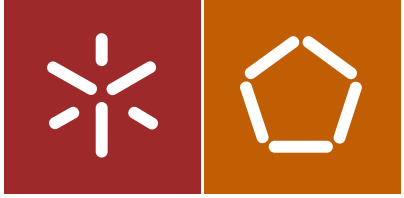
Universidade do Minho
Escola de Engenharia

Jorge Bruno Ferreira da Silva

Generating Timed Trajectories for an Autonomous Robot

janeiro de 2015



Universidade do Minho
Escola de Engenharia

Jorge Bruno Ferreira da Silva

Generating Timed Trajectories for an Autonomous Robot

Tese de Doutoramento
Programa Doutoral em Engenharia Electrónica e Computadores

Trabalho efectuado sob a orientação de
Professora Doutora Cristina Santos
Professor Doutor João Sequeira

janeiro de 2015

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, _____

Full name: _____

Signature: _____

Preface

This manuscript describes the work performed throughout the author's doctoral thesis while member of the Adaptive System Behavior Group (ASBG), within the Doctoral Program in Electrical and Computer Engineering at the University of Minho in Portugal. The sponsoring of the project was supported by *Fundação para a Ciência e Tecnologia* (FCT).

The supervisors of the project were Cristina Santos from University of Minho and João Sequeira from Instituto Superior Técnico.

The main topic of this research was the development and stability analysis of an architecture with time constraints to guide autonomous mobile robots in real environments.

Jorge Silva
January 2015

Summary

The inclusion of timed movements in control architectures for mobile navigation has received an increasing attention over the last years. Timed movements allow modulating the behavior of the mobile robot according to the elapsed time, such that the robot reaches a goal location within a specified time constraint. If the robot takes longer than expected to reach the goal location, its linear velocity is increased for compensating the delay. Timed movements are also relevant when sequences of missions are considered. The robot should follow the predefined time schedule, so that the next mission is initiated without delay. The performance of the architecture that controls the robot can be validated through simulations and field experiments. However, experimental tests do not cover all the possible solutions. These should be guided by a stability analysis, which might provide directions to improve the architecture design in cases of inadequate performance of the architecture.

This thesis aims at developing a navigation architecture and its stability analysis based on the Contraction Theory. The architecture is based on nonlinear dynamical systems and must guide a mobile robot, such that it reaches a goal location within a time constraint while avoiding unexpected obstacles in a cluttered and dynamic real environment. The stability analysis based on the Contraction Theory might provide conditions to the dynamical systems parameters, such that the dynamical systems are designed as contracting, ensuring the global exponential stability of the architecture. Furthermore, Contraction Theory provides solutions to analyze the success of the mission as a stability problem. This provides formal results that evaluate the performance of the architecture, allowing the comparison to other navigation architectures.

To verify the ability of the architecture to guide the mobile robot, several experimental tests were conducted. The obtained results show that the proposed architecture is able to drive mobile robots with timed movements in indoor environments for large distances without human intervention. Furthermore, the results show that the Contraction Theory is an important tool to design stable control architectures and to analyze the success of the robotic missions as a stability problem.

Resumo

A inclusão de movimentos temporizados em arquitecturas de controlo para navegação móvel tem aumentado ao longo dos últimos anos. Movimentos temporizados permitem modular o comportamento do robô de tal forma que ele chegue ao seu destino dentro de um tempo especificado. Se o robô se atrasar, a sua velocidade linear deve ser aumentada para compensar o atraso. Estes movimentos são também importantes quando se consideram sequências de missões. O robô deve seguir o escalonamento da sequência, de tal forma que a próxima missão seja iniciada sem atraso. O desempenho da arquitectura pode ser validado através de simulações e experiências reais. Contudo, testes experimentais não cobrem todas as possíveis soluções. Estes devem ser conduzidos por uma análise de estabilidade, que pode fornecer direcções para melhorar o desempenho da arquitectura.

O objectivo desta tese é desenvolver uma arquitectura de navegação e analisar a sua estabilidade através da teoria da Contracção. A arquitectura é baseada em sistemas dinâmicos não lineares e deve controlar o robô móvel num ambiente real, desordenado e dinâmico, de tal modo que ele chegue à posição alvo dentro de uma restrição de tempo especificada. A análise de estabilidade baseada na teoria da Contracção pode fornecer condições aos parâmetros dos sistemas dinâmicos de modo a desenha-los como contracções, e assim garantir a estabilidade exponencial global da arquitectura. Esta teoria fornece ainda soluções interessantes para analisar o sucesso da missão como um problema de estabilidade. Isto providencia resultados formais que avaliam o desempenho da arquitectura e permitem a comparação com outras arquitecturas.

Para verificar a habilidade da arquitectura em controlar o robô móvel, foram conduzidos vários testes experimentais. Os resultados obtidos mostram que a arquitectura proposta é capaz de controlar robôs móveis com movimentos temporizados em ambientes interiores durante grandes distâncias e sem intervenção humana. Além disso, os resultados mostram que a teoria da Contracção é uma ferramenta importante para desenhar arquitecturas de controlo estáveis e para analisar o sucesso das missões efectuadas pelo robô como um problema de estabilidade.

Acknowledgements

This thesis is the end of a journey on obtaining my PhD. At this moment of accomplishment, I would like to thank everyone who helped me on making this thesis possible and a remarkable experience.

Primarily, I am grateful to my PhD advisors, Professors Cristina Santos and João Sequeira. I wish to thank them for their constant support, fruitful discussions, proofreading and interesting suggestions. In particular, I am thankful to Professor Cristina Santos, who invited me to join her lab, allowing me to pursue a research project on mobile navigation and nonlinear dynamical systems. Further, I am grateful to Professor João Sequeira for sharing his knowledge on stability theory and for receiving me at the IST in Lisbon.

A good daily support was essential to finish my journey. I am grateful to all the elements of the *ASBG* lab team, in special those who shared the lab with me while pursuing their PhDs, Miguel Oliveira, Vitor Matos, Pedro Silva, Carolina Vilares, Maria Martins and Ricardo Campos. I would like to mention a special recognition to Vitor Matos, for all the fruitful discussions we had about robotics and for the support that he gave me on solving my problems. I also have a special thanks to Carolina Vilares and Maria Martins for their valuable proofreading. I am indebted to all the persons that participated in my experiments, in particular Vitor Faria, João Macedo, César Ferreira, Carlos Teixeira and David Barbosa.

The support outside the lab was fundamental to keep me on track of my goals. I especially thank my mom, dad and brother for their love, encouragement and inspiration. I also would like to show gratitude to my girlfriend, Isabel, for the weeks we were not together. Without their constant support, this thesis was not possible.

Finally, I thank the financial support provided by the Portuguese Science and Technology Foundation (FCT) through the PhD grant SFRH/BD/68805/2010, and the bureaucratic support provided by the University of Minho, in particular, the research center ALGORITMI.

Contents

List of Figures **xiii**

List of Tables **xix**

List of acronyms **xxi**

1 Introduction **1**
1.1 Motivation . . . 1
1.1.1 Timing in Hospital Delivery Missions . . . 2
1.1.2 Nonlinear Dynamical Systems Theory . . . 3
1.1.3 Stability in Nonlinear Robotic Controllers . . . 5
1.1.4 Contraction Theory . . . 5
1.1.5 Performance as a Stability Measure . . . 6
1.2 Quick View of the Proposed Solution . . . 6
1.3 Main Contributions . . . 7
1.4 Thesis Organization . . . 8
1.5 List of publications . . . 10

2 Context and Related Work **11**
2.1 Service Mobile Robots . . . 11
2.1.1 Cleaning and Lawn Mower Robots . . . 12
2.1.2 Tele-presence and Assistive . . . 13
2.1.3 Smart Walkers . . . 14
2.1.4 Delivery Robots . . . 14
2.1.5 Hospital Mobile Robots . . . 15
2.1.6 Comparative Analysis . . . 18
2.2 Control Architectures . . . 19
2.2.1 Local Path Planning . . . 20
2.2.2 Global Path Planning . . . 25
2.2.3 Timing Control . . . 28
2.2.4 Localization . . . 31
2.3 Robotic Architectures Paradigms . . . 35
2.4 Stability Analysis of Nonlinear Systems . . . 37
2.4.1 Phase Plane Analysis . . . 37

2.4.2 Lyapunov Theory . . . 38
2.4.3 Input-Output Stability . . . 39
2.4.4 Contraction Mapping Theory . . . 40
2.5 Problem Statement . . . 41

3 Overall Architecture 43
3.1 Global System Overview . . . 43
3.2 Dynamic Approach to Motion Control . . . 45
3.2.1 World Representation . . . 46
3.2.2 Transition Function M . . . 47
3.2.3 Total Cost of an Arc . . . 48
3.2.4 Global Planner . . . 54
3.2.5 Look-up Table . . . 55
3.2.6 Global - Local Integration . . . 59
3.3 Dynamic Approach to Local Control . . . 60
3.3.1 Behavior Variables . . . 61
3.3.2 Target Orientation . . . 62
3.3.3 Obstacle Avoidance . . . 63
3.3.4 Target Orientation and Obstacle Avoidance Integration . . . 66
3.3.5 Detection of Obstacles . . . 68
3.3.6 Target Orientation and Obstacle Avoidance Behaviors . . . 68
3.4 Dynamic Approach to Timing Control . . . 70
3.4.1 Velocity . . . 70
3.4.2 Timing Adaptation . . . 73
3.4.3 Obstacle Profile . . . 79
3.4.4 Behavior Switching . . . 81
3.4.5 Parameter Modulation . . . 84
3.4.6 Adiabatic Elimination . . . 86

4 Fundamentals for Stability and Success of the Global System 89
4.1 Contraction Mapping Theory . . . 89
4.2 Stability and Mission Success . . . 91
4.3 Stability Analysis . . . 95
4.3.1 Motion Control - Contraction Analysis . . . 97
4.3.2 Local Control - Contraction Analysis . . . 99
4.3.3 Timing Control - Contraction Analysis . . . 101

5 Experimental Setup 107
5.1 Constraints of Hospital Environments . . . 107
5.1.1 Robot Hardware Design . . . 108
5.1.2 Navigation and Safety . . . 109
5.1.3 Localization . . . 110
5.1.4 Interface . . . 110
5.1.5 Cost Effectiveness . . . 111
5.2 Indoor Environment . . . 112

5.3 Robot and Sensors . . . 114
5.4 Frame Assignment . . . 115
5.5 Simulator . . . 116
5.6 Localization System . . . 119
5.6.1 Vision System . . . 119
5.6.2 Landmark Placement . . . 122
5.6.3 Sensor Fusion - An Extended Kalman Filter Approach . . . 123
5.7 Dependability of the Localization System . . . 132
5.7.1 Accuracy and Precision . . . 132
5.7.2 Convergence Time . . . 133
5.7.3 Integrity . . . 133
5.8 External module . . . 138

6 Experiments **141**
6.1 Simulation Experiments . . . 143
6.1.1 Simulation 1 . . . 143
6.1.2 Simulation 2 . . . 147
6.2 Real Experiments . . . 151
6.2.1 Experiment 1 . . . 152
6.2.2 Experiment 2 . . . 153
6.2.3 Experiment 3 . . . 157
6.2.4 Experiment 4 . . . 160
6.2.5 Experiment 5 . . . 163
6.2.6 Experiment 6 . . . 165
6.2.7 Experiment 7 . . . 166
6.2.8 Experiment 8 . . . 169
6.2.9 Experiment 9 . . . 172
6.2.10 Experiment 10 . . . 175
6.2.11 Experiment 11 . . . 175
6.2.12 Experiment 12 . . . 180
6.3 Discussion of the experiments . . . 181

7 Conclusions **183**
7.1 Addressed Subjects . . . 183
7.2 Summary of Contributions . . . 184
7.3 Outlook . . . 187

A **191**
A.1 World Representations . . . 191
A.2 Orthogonal Projection . . . 191
A.3 Stuart-Landau oscillator . . . 192
A.4 C^1 Dynamical Systems . . . 195

B **203**

B.1 Robot Features 203
B.1.1 Kinematics 204
B.2 Sensors 205

Bibliography **206**

List of Figures

3.1 Graphical representation of the global system. 44
3.2 Schematic of the K, f_{robot} and $f_{\text{supervisor}}$ blocks containing the modules of control Motion, Local and Timing. 45
3.3 Schematic of the Motion Control module. 46
3.4 Space of beliefs for the robot's position. 47
3.5 Sequence of algorithms used to build the transition function, M, which describes topologically the environment. 47
3.6 Topological representation of the simulated environment. 48
3.7 Deterministic cost between regions r_1 and r_2. 49
3.8 a) Obstacles partially blocking the corridor. b) Obstacles totally blocking the corridor. 51
3.9 a) Two alternative paths connecting region r_1 to region r_8. b) The robot selects the lowest cost path (path 2). 53
3.10 a) Two alternative paths connecting region r_1 to region r_8. b) The robot selected path 2, because it is the lowest cost path. 54
3.11 Evolution of the total cost of path 1 (red dashed line) and path 2 (green continuous line). 54
3.12 Robot starts its mission in region r_3 (red circle) and goal location, P_{g} (green cross) is located in region r_{16}. 56
3.13 Example of a trajectory followed by the robot when it is close to a boundary between regions. 57
3.14 A corridor of an environment composed by the sequence of regions, $\rho = \{r_1, r_2, r_3, r_4\}$. 58
3.15 a) Robot moves from region r_3 to region r_2 by crossing critical line $l_{3,2}$ at time $t = k$. b) Robot circumnavigates an unexpected obstacle (red rectangle) at time $t = k + 1$ and the local goal P_{b} is updated according to (3.18) and (3.19). 60
3.16 Schematic of the Local Control module. 61
3.17 Schematic representing the direction to the goal location, ψ_{tar} and the robot's heading direction, ϕ, relative to the allocentric reference frame, $\{W\}$. 62
3.18 Vector field of the target orientation contribution. 63
3.19 Robot and laser range finder seen through a top perspective. 64
3.20 Range of the laser step, $La_{,341}$, detecting an obstacle at a distance $d_{\text{l},341}$. 66
3.21 Vector field of the obstacle avoidance contribution. 66
3.22 Vector field resulting from the superposition of the attractive and repulsive forces. 67

3.23 Worst case scenario in which an obstacle lies between the robot and the goal location. 69
3.24 a) Vector field of the target orientation contribution. b) Vector field of the obstacle avoidance contribution. c) Resulting vector field of the heading direction dynamics. 70
3.25 Schematic of the Timing Control module. 71
3.26 m trajectory modulation (solid red line) and parameters' role. 72
3.27 Solution m of the Stuart-Landau oscillator 73
3.28 Single oscillation profile generated by the Stuart-Landau oscillator. . . 74
3.29 Blue continuous line depicts the frequency of the oscillator, ω, dashed grey line depicts solution m and dashed dotted green line depicts the oscillator offset, O_{m}. 75
3.30 Example of how to calculate the distance D(0) between the robot's initial position, $P_{\mathrm{r}}(0)$, and the goal location, P_{g}. 78
3.31 A is the radius of the Stuart-Landau oscillator 79
3.32 Bifurcation diagram and phase space of the competitive dynamics without the competitive term. 82
3.33 Triplet of variables u_i. 85
3.34 Parameter μ according to the triplet of variables u_i. 86
3.35 Behavior of the robot according to the triplet of variables u_i. 87

4.1 T is contracting. r is the radius of ball $B(x, r)$ and s is the radius of ball $B(Tx, s)$. 90
4.2 Graphical representation of the global system with feedback of the upper bound of the stability indicator. 94
4.3 Block diagram representing the global system. 96
4.4 Phase portrait of $D(f_4)$ ($D(f_5)$) and f_4 (f_5) for $\lambda_{\mathrm{tr}} < 0$. 98
4.5 Phase portrait of $D(f_8)$ and f_8 for $\lambda_{\mathrm{tar}} < 0$. 100
4.6 Phase portrait of $D(f_8)$ and f_8 for $\lambda_{\mathrm{obs},i_k} > 0$ and $\lambda_{\mathrm{tar}} < 0$. 101
4.7 Phase portrait of $D(f_{15})$ and f_{15} for $\beta_i > 0$ and $\beta_i < 0$. 103

5.1 Map of the indoor environment in which the robot performs the field experiments. 113
5.2 Snapshots from the indoor environment. 114
5.3 Pioneer 3-DX and remaining hardware used to control the robot: camera, laptop and laser. 115
5.4 Reference frames. 116
5.5 Simulated model of the Pionner 3-DX in Webots. 117
5.6 Webots simulated environment. 117
5.7 Simulated objects usually found in hospitals. 118
5.8 Camera on the robot's top and camera's field of view. 120
5.9 Example of a line segment divided into N equal parts (black divisions) and N equidistributed points (red circles). 122
5.10 Red circles identify the landmarks distributed on the environment. . . . 124
5.11 Localization System module composed by a direct EKF. 125

5.12 Error of the robot's pose calculated by the camera when compared to the ground truth. 130
5.13 Precision *vs* time: convergence of the localization system. 134
5.14 Minimum distance measured by the laser mounted on the robot and reference velocity, generated by the Timing Control module. 135
5.15 Estimates of the robot's pose $(\hat{x}_{\mathrm{r}}, \hat{y}_{\mathrm{r}}, \hat{\phi})$. 136
5.16 Uncertainty of the localization system when the percentage of detected landmarks is changed. 137
5.17 Uncertainty *vs* time: convergence of the localization system when the robot is kidnapped during an interval of time (green area). 138
5.18 Schematic of the External module that represents the interface between the user and the robot. 139

6.1 Different situations faced by the robot during simulation 1. 144
6.2 Solutions m (blue continuous line) and n (red dashed line) generated by the Stuart-Landau oscillator, remaining distance between the robot and the goal location, D (black continuous line) and variables responsible for the robot's motor behavior. 145
6.3 Evolution of the bound of $\|D(f_{\mathrm{supervisor}})\|$ along the three missions of simulation 1. 146
6.4 Trajectory followed by the robot during simulation 1. 147
6.5 Estimates given by the EKF of the trajectory followed by the robot in simulation 1. 148
6.6 Error between the ground truth of the robot's position and the one estimated by the EKF. 148
6.7 Trajectory followed by the robot during simulation 2. 149
6.8 Evolution of the bound of $\|D(f_{\mathrm{supervisor}})\|$ and sequence of f_i Jacobians along the three missions of simulation 2. 150
6.9 Trajectory followed by the robot during the two failed missions of simulation 2. 151
6.10 Robot's linear velocity, v, during simulation 2. 151
6.11 Snapshots of the robot performing the mission of experiment 1. 152
6.12 Solutions of the Stuart-Landau oscillator m (blue continuous line), n (red dashed line), amplitude of the oscillator, A (black dashed-dotted line) and distance between the robot and the goal location (black continuous line). 153
6.13 Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). 154
6.14 Evolution of the bound of $\|D(f_{\mathrm{supervisor}})\|$ along the mission of experiment 1. 154
6.15 Snapshots of the robot performing the mission of experiment 2. 155
6.16 Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.15. 155

6.17 Solution m (blue continuous line) and amplitude of the Stuart-Landau oscillator, A (black dashed-dotted line), direction that the robot should follow to reach the goal location and potential function indicating the presence of obstacles, $U(\phi)$. 156
6.18 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 2. 157
6.19 Snapshots of the robot performing the mission of experiment 3. 158
6.20 Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.19. 158
6.21 Velocity followed by the robot, v (red continuous line), solution m of the Stuart-Landau oscillator (blue continuous line), distance between the robot and the goal location (black continuous line) and target orientation F_{tar} (blue continuous line) and obstacle avoidance F_{obs} (red dashed line) contributions. 159
6.22 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 3. 160
6.23 Snapshots of the robot performing the mission of experiment 4. 161
6.24 Snapshots of the surroundings of the robot when it performs its mission in experiment 4. 161
6.25 Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.23. 162
6.26 Robot's linear velocity, v (red continuous line), solution m generated by the Stuart-Landau oscillator (blue dashed line), distance between the robot and the goal location, D (black continuous line) and distance in which it is assumed that the robot has reached the goal location (red dashed line). 162
6.27 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the single mission of experiment 4. 163
6.28 Snapshots of the robot performing the mission of experiment 5. 164
6.29 Robot's linear velocity, v (red continuous line), solution m (blue dashed line), distance between the robot and the goal location, D (black continuous line) and distance in which it is assumed that the robot reaches the goal location (red dashed line). 165
6.30 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along experiment 5. 166
6.31 Solution m (blue continuous line) and the amplitude of the oscillator, A (black dashed-dotted line), set of variables responsible for the behavior of the robot. 167
6.32 (a) Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along experiment 6. (b) $\|D(f_{\text{timing}})\|$ identifies if the mission is completed within the time constraint. 167
6.33 Snapshots of the robot performing the two missions in experiment 7. The robot is kidnapped during the second mission. 168
6.34 Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). 169

6.35 (a) Solutions m (blue continuous line) and amplitude of the oscillator, A (black dashed-dotted line). (b) Distance covered by the robot throughout the two missions. 170
6.36 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 7. 170
6.37 Snapshots of the robot performing the mission in experiment 8. During this mission, the robot is kidnapped and its vision system is disabled. . 171
6.38 (a) Robot's linear velocity, v (red continuous line). (b) Distance covered by the robot during the two missions. 172
6.39 Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). 173
6.40 Limit-cycle of the Stuart-Landau oscillator when the robot is kidnapped and its amplitude is increased to compensate the provoked delay. 174
6.41 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 8. 174
6.42 Snapshots of the robot performing a mission in which it is kidnapped and does not reach the goal location. 175
6.43 Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). 176
6.44 (a) Robot's linear velocity, v (red continuous line) and solution m generated by the Stuart-Landau oscillator (blue dashed line). (b) Distance covered by the robot during the two missions. 176
6.45 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 9. 177
6.46 Snapshots of the robot moving in a narrow passage. 177
6.47 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 10. 178
6.48 Snapshots of the robot performing the long-term mission. 178
6.49 Estimates given by the EKF of the trajectory followed by the robot. . . 179
6.50 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the sequence of missions. 179
6.51 Goal locations P_{g} of the missions performed by the robot during experiment 12. 180
6.52 Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the sequence of missions. 181

A.1 Representation of the simulated environment using occupancy grid, topological and landmark-based representations. Crosses represent landmarks. 191
A.2 Projection of the robot's position, P_{r}, onto the critical line, $l_{i,i+1}$. P_{1i} and P_{2i} are the extremities of the respective critical line. 192
A.3 Phase portraits of the Stuart-Landau oscillator depicted in (3.35)-(3.36). 193
A.4 The bifurcation diagram and the one-parameter family of limit cycles Γ_μ resulting from the Hopf bifurcation. 193
A.5 Smooth modulation of the generated trajectory amplitude (top panel) by modifying parameter μ (bottom panel). 194
A.6 Harmonic oscillations with different periods. 194
A.7 Limit-cycles for different signals of frequency ω. 195
A.8 Offset O_{m} and state variable m superimposed. 195

A.9 Derivatives of the feed-through map f_{12} relative to variables m and n. 201
A.10 Derivative of the feed-through map f_{12} relative to variable O_{m}. 202

B.1 Lateral and top perspectives of the robot. Measures are in centimeters. 203
B.2 Distribution of the 8 sonar sensors mounted in front of the robot. . . . 204
B.3 (a) Laser Range Finder Hokuyo URG-04LX-UG01. (b) PsEye camera. . 205

List of Tables

2.1 Comparative analysis in terms of navigation capabilities (✓ - fulfilled, X - not fulfilled) 19
2.2 Expected error according to the exteroceptive sensor used for robot indoor localization. 34

3.1 Distance required to circumnavigate obstacles type O_1. 51
3.2 Total cost of path 1 and path 2 during 9 traversals. 53

4.1 Stability parameters. 104

5.1 Specifications considered for delivery mobile robots (✓ - fulfilled, X - not fulfilled). 112
5.2 Results of the experimental field tests to verify the odometry uncertainty. 129
5.3 Accuracy and precision of the Localization System. 132

6.1 Summary of the simulations and experiments. 143
6.2 Results of the sequential missions performed in experiment 12. 181

List of acronyms

2D two-dimensional 25

3D three-dimensional 13

AGVs Automatic Guided Vehicles 14

CAD Computer Aided Design 17

CPGs Central Pattern Generators 29

CVM Curvature-Velocity Method 23

DMP Dynamic Movement Primitive 31

DSA Dynamical Systems Approach 21

DWA Dynamic Window Approach 23

EKF Extended Kalman Filter 30

GPS Global Positioning System 33

LCM Lane-Curvature Method 23

LPA* Lifelong Planning A* 27

NDV Nearest Diagram Navigation 24

ODE Open Dynamics Engine 116

ORM Obstacle-Restriction Method 24

PVO Probabilistic Velocity Obstacle 23

RAPs Reactive Action Packages 36

RFID Radio Frequency Identification 32

SPA Sense-Plan-Act 36

TVDW Time-Varying Dynamic Window 24

UWB Ultra Wide Band 32

VFF Virtual Force Field 21

VFH Vector Field Histogram 22

VO Velocity Obstacle 23

WLAN Wireless Local Area Network 32

To my family...

Chapter 1

Introduction

The main aim of this thesis is to develop an architecture able to guide an autonomous mobile robot whose missions are performed with time constraints in a real and dynamic environment. The modules of the architecture are based on nonlinear dynamical systems. Contraction Theory is a tool used to show under what conditions the proposed architecture is stable and able to drive the mobile robot successfully.

Real experiments show the performance of the architecture in a lab indoor environment. This environment has similarities with hospitals, which are the main focus for the navigation with time constraints proposed in the thesis. However, the proposed architecture is indeed generic and can be applied to mobile robots in common indoor environments.

1.1 Motivation

Navigation is the ability of a mobile robot to move towards its goal locations based on sensory data and knowledge about the environment. In cluttered and dynamic indoor environments, the robot can easily be blocked by obstacles and become unable to pursue its mission until the obstacles have been removed. A well-designed control architecture is required to successfully guide mobile robots in such environments.

The architecture has to deal with both global and local path planning problems. Global path planning involves calculating a trajectory that will lead the robot towards the goal location when the map of the environment is known. Typically, this problem is solved before the robot starts its mission. Local path planning means locally modulate the trajectory, given real-time sensory data acquired by the robot of its surroundings. Local planning includes obstacle avoidance and target orientation behaviors. These behaviors allow the robot to avoid collisions, so that it can safely reach the goal location.

In autonomous navigation, global and local path planning abilities are not sufficient to handle timing problems. If a robot has to obey a time constraint to reach its goal location, it is necessary to include the ability to generate timed movements into its architecture. The robot will be allowed to complete its missions according to an *a priori* defined time constraint, despite varying environmental conditions or bounded perturbations (*e.g.* obstacles). If a perturbation delays the robot, its movement is compensated to comply as much as possible the time constraint.

The majority of approaches addressing navigation problems does not handle the generation of timed movements. Time is not considered as a parameter to the control architectures and robot actions are initiated and finished independently of the elapsed time. If the robot takes longer than expected to complete its mission because of unforeseen disturbances, this change of timing is not compensated along its trajectory. For instance, obstacles detected during the mission may force the robot to cover a larger distance than expected or even stop and wait for the clearance of the path. The time lost to handle these disturbances is not compensated and the robot finishes its mission after the expected time. Therefore, a timed movement approach where the robot accelerates along its trajectory is necessary for compensating the possible delays.

The reason for including timed movements in mobile navigation is even more relevant when sequences of missions are considered. If a mission is delayed by external factors, the robot should follow the predefined time schedule and complete the mission within the expected predefined time, so that the next mission is initiated without delay. A potential application for the generation of timed movements for sequence of missions is the automation of deliveries in hospitals. The delivery time can be critical for robots in hospitals, because in general, drugs and meals have scheduled times to be delivered.

Timed movements are important for other robotic applications, mainly those that require rhythmic movements, such as locomotion for legged robots [1–3], dancing, catching, hitting, juggling and human-robot interaction [4]. Time independent movements are for instance ambulatory displacements over the environment, wherein there is no specific time to reach a particular goal point.

1.1.1 Timing in Hospital Delivery Missions

Several studies [5–7] have shown that automated deliveries of goods in hospitals improve their transportation efficiency and overall organization, while reducing the logistic costs. Robots can transport non-urgent supplies, such as food, dishes, linens

and wastes. Nurses and housekeepers can spend more time on patient care tasks instead of doing repetitive delivery tasks. This increases patient satisfaction and reduces operational costs. Furthermore, the service quality offered by the hospital is improved.

Manual hospital delivery tasks are usually bounded by the limitations of available workers. Transportation routes, load volumes, load weights and travel frequencies are planned according to the available means of transportation framework and personnel. An automated system improves flexibility, as transportation routes can be optimized and deliveries can be planned for night and daytime.

Current delivery robots navigate in hospital facilities at a constant velocity and stop when imminent collisions are detected. After the obstacles have been removed, the robot keeps moving with the previous constant velocity. Since time constraints are not considered as a requirement to complete the delivery missions, the delay is not compensated for by accelerating the robot and the goal location is reached after the scheduled time. This delay can be very significant for the reason that robots move in corridors along with human traffic, ride elevators, open automatic doors and avoid obstacles. For instance, the time that elevators need to travel between floors depends on the number of persons who call them and if they are far from the floor where they were requested to arrive. These perturbations will cause several interruptions to the movement of the robots and they will fail to reach their destinations when they are needed.

Controlling the velocity of the robot instead of setting a constant velocity has multiple advantages. For instance, regardless of the initial position of the robot and the final destination, the robot can perform its delivery missions within a specific time. Performing delivery time schedules is important when considering drugs or meals, whose delivery times are in general predefined. Thus, if the robot is late on a delivery mission, it must compensate the delay, such that the following missions are initiated according to the time schedule. Another reason for generating timed movements is to ensure that robots complete similar missions at different periods of the day with the same time constraints, even if the density of obstacles changes throughout the day. In addition, controlling the velocity of the robot can save operational time that can be used by the robot to perform more delivery tasks.

1.1.2 Nonlinear Dynamical Systems Theory

In the last years, nonlinear approaches have been widely used by researchers to design and build control systems. Even though linear control has been successful in a wide range of industrial applications, nonlinear control offers useful tools for analyzing and

modeling patterns of stability in the behavior of a system. For instance, the generation of timed movements in robotics has been following a standard approach based on nonlinear dynamical systems. The work in [4] suggested that all timing models can be represented by limit cycles in nonlinear oscillators.

The majority of physical and engineering systems is inherently nonlinear. Numerous sources of nonlinearities that arise in such systems, *e.g.*, friction (Coulomb, hysteresis), input constraints (*e.g.*, saturation), gyroscopic and kinematic effects must be accounted for in the system analysis and control design. These nonlinearities are handled by nonlinear controllers in large or small range operations, while linear controllers have to rely on the small range operation where the linearity assumption is valid. Linear controllers may be unstable if the operation range is large or if the discontinuities in the nonlinearities do not allow linear approximation [8]. Nonlinear control provides techniques to estimate the behavior of the system even in the presence of nonlinearities. Linear controllers may require extensions to handle systems with significant nonlinearities, such as high quality actuators and sensors. On the other hand, nonlinear control may use less expensive actuators with nonlinear characteristics and that does not compromise the system in terms of stability and performance [8]. Well-designed nonlinear controllers handle uncertainties on the parameters because of their robustness and adaptability. On the contrary, linear control requires that the parameters of the model are well known. If they are inaccurate, linear control may exhibit instability and degradation in the performance.

Nonlinear systems provide richer and more complex behaviors than linear systems. The variables of a nonlinear system might converge to an equilibrium state, blow to infinity, exhibit irregular chaotic patterns, bifurcate from one pattern to another, repeat periodic patterns or behave randomly. Furthermore, a single nonlinear dynamical system can exhibit different patterns depending on its initial conditions and control parameters. Thus, the stability analysis of nonlinear dynamical systems is much more difficult. In fact, there are nonlinear dynamical systems that can not be solved analytically. Stability theory can be used to predict the behavior of nonlinear dynamical systems by analyzing the trajectory of their state variables over time, the set of their possible bifurcations and their control parameters. In fact, the small number of control parameters in nonlinear dynamical systems reduces the dimensionality of the control problem [9]. Thus, they are amenable to formal analysis, namely, when considering stability conditions in real robots.

1.1.3 Stability in Nonlinear Robotic Controllers

Nonlinear systems can be analyzed under several properties, such as controllability, observability, invertibility and stability [10]. All properties play a fundamental role in the behavior of the systems. Among them, stability can be used to estimate the behavior of the systems over time.

If a control system is stable, its output will follow its input at a certainty distance. An unstable control system can not guarantee that its output follows its input under bounded perturbations. The amplitude of at least one system coordinate or output variable of an unstable system may be unbounded, even if the input of the system is bounded. Usually, this drives the system to saturation and to undesirable consequences, such as breakdowns or damages. For these reasons, the stability of control systems must be guaranteed.

In the case of mobile navigation, it is clear that experimental tests, such as simulations and real experiments, of nonlinear control architectures are very important to validate the performance of the mobile robots over time. However, experimental tests do not cover all the possible solutions, since nonlinear systems exhibit a wide range of behaviors that depend on initial conditions, inputs and perturbations. A stability analysis may provide conditions under which the solutions of the nonlinear systems can be predicted, despite the different initial conditions and bounded perturbations.

1.1.4 Contraction Theory

There is no single concept of stability and many different definitions are possible. A number of strong stability concepts, such as global asymptotic stability or global exponential stability are required to the navigation problem. Global asymptotic stability ensures that all solutions tend to an equilibrium state of interest. This concept of stability is motivated by the fact that stability and performance properties of many control systems are independent of the initial time. However, global asymptotic stability does not quantify the rate of convergence and in some applications, it is insufficient to know that the control system converges to an equilibrium point in finite time. In robot navigation, there is a need to estimate how fast the mobile robot approaches the goal location.

The concept of global exponential stability ensures that the convergence of the system is bounded by an exponential decay. If a system is exponentially stable, then it is asymptotically stable and stable in the sense of Lyapunov. The converse is not necessarily true. If a system composed by a nonlinear controller and a robot is globally

exponentially stable, then it is able to drive the robot from its initial position to a neighborhood arbitrarily small centered on any goal position in finite time.

Contraction Theory is a recent tool (see [11]) that ensures global exponential stability of nonlinear control systems. This theory states that if a nonlinear system is contracting, then its initial conditions are somehow forgotten and transient bounded disturbances vanish exponentially. Thus, bounded external perturbations have no effect on the convergence, and this is a fundamental property when dealing with realistic control systems.

1.1.5 Performance as a Stability Measure

As the complexity in robotics is increasing, it becomes necessary to establish performance measures that enable the comparison among robotics research results. A typical performance criterion that evaluates the ability of a control architecture to guide mobile robots is the mission success, *i.e.*, the number of successful missions completed by the robot.

Contraction Theory provides a combination property, that can be used to evaluate the mission success as a stability problem. Combination property guarantees the contraction of a control architecture if all of its nonlinear dynamical systems are contracting. Furthermore, the knowledge of the internal organization of the architecture is not required, as the contraction of the architecture holds over a wide range of possible combinations of its dynamical systems [12].

A contraction criterion can be derived through the combination property, and viewed as a stability indicator that identifies the success of the mission. If the overall system is contracting, it reaches its unique fixed point and the mission is successfully completed. Otherwise, the mission may fail.

To the best of the author's knowledge, performance criteria measuring the ability of a navigation system to drive mobile robots in complex dynamic environments are obtained through simulation tests or real experiments. In general, there is a lack of formal results. On the other hand, analyzing the performance criteria as a stability problem based on Contraction Theory provides formal results that evaluate the performance of the navigation system.

1.2 Quick View of the Proposed Solution

A previous architecture [13], where obstacle avoidance and timed movements were successfully achieved for short displacements, in the order of a couple of meters, using

a toy-like mobile robot, supports the proposed architecture. The new architecture includes three modules of control, Motion Control for path planning, Local Control for local path planning and Timing Control for generating timed movements. Each module is based on a mesh of C^1 nonlinear dynamical systems and feed-through maps [1].

The Motion Control module receives the map of the environment and the position of the robot, in order to generate the goal direction that the robot should follow to reach the goal location. This module converts the representation of the environment from a physical perspective to a mathematical perspective, in order to treat the environment as a mathematical model. The Local Control module is responsible for the obstacle avoidance and target orientation tasks. It receives the goal direction from the Motion Control and verifies based on sensory information if the robot can follow this direction. This module generates the robot's angular velocity. The Timing Control module generates the linear velocity, so that the robot accomplishes its missions within their respective time constraints. The pose of the robot (position and orientation) is estimated based on vision information and odometry by the localization system. The position of the robot is included into the Timing Control module to verify if the robot is on time, on advance or delayed to complete the mission.

1.3 Main Contributions

This thesis provides three main contributions. First, it proposes an architecture that combines in a novel way navigation abilities (local and global path planning) with the generation of timed movements for a mobile robot capable of performing missions in cluttered and dynamic environments.

The second contribution consists on the exploitation of the Contraction Theory properties. This contribution is divided into two directions. The first one provides stability conditions that define the control parameters of the dynamical systems composing the architecture. Hence, the architecture is designed on top of stability conditions. The second direction extends the combination property of the Contraction Theory to derive a contraction criterion that identifies if the robot completes with success its missions. This property includes a norm bounded constraint augmented with a term corresponding to the time constraint, which corresponds to a desired level of performance. Thus, the mission success is identified as a stability problem. If the global system[2] converges to its unique fixed point, then, the robot successfully completes the mission. An important aspect of this contribution is that it provides an *a*

[1] C^1 stands for the set of continuously differentiable functions.

[2] Closed loop composed by the robot, its controlling architecture and the interacting environment.

priori roadmap to design stable control systems that can be extended to a wide range of other control systems.

Results obtained from field experiments in a real environment are the final contribution. Several experiments were conducted to show that the architecture successfully drives the mobile robot to its goal locations in due time. Furthermore, it is expected that the stability indicator identifies possible successful and unsuccessful missions. This final contribution also provides a dependability analysis that concludes about the robustness of the proposed architecture. The dependability analysis considers the reliability, integrity and safety of the robot. Long-term experiments, covering a wide range of realistic scenarios in a typical university indoor environment, validate the ability of the architecture to guide the robot towards its goal location while respecting the time constraint. These experiments assess if the proposed architecture is able to drive a mobile robot for large periods of time in realistic environments, such as hospitals. In this final contribution, Monte Carlo tests show the ability of the localization system to provide accurate robot poses estimates under different environment conditions.

The main contributions of this thesis are summarized as follows:

- Extension of a control architecture with time constraints based on nonlinear dynamical systems and feed-through maps capable of guiding autonomous mobile robots in dynamic and cluttered environments, such as hospitals;
- Extension of the dynamic approach that generates and modulates the timing control of the robot to deal with global and local path planning;
- Stability analysis of the architecture through the Contraction Theory;
- Definition of a stability indicator based on the combination property of Contraction Theory that identifies the ability of the robot to successfully complete its missions;
- Results obtained from the conducted long-term experiments that covered a wide range of realistic scenarios in a lab environment with similarities to hospital environments.

1.4 Thesis Organization

This section briefly reviews the different chapters of this thesis.

Chapter 2

This chapter presents a review of literature related to this work. A brief introduction to autonomous mobile robots is given. This review compares navigation techniques, sensors for obstacle avoidance, real-time localization and assigned tasks between delivery hospital robots and other autonomous mobile robots, namely, assistive/telepresence, cleaning and mowing robots. The chapter continues with a description about methods for obstacle avoidance, global path planning, timing control and localization techniques. A description of stability theory methods for control architectures concludes this chapter.

Chapter 3

This chapter fully details the modules that compose the control architecture: Motion, Local and Timing. They are responsible for generating the linear and angular velocity, so that the robot completes the missions within their respective time constraints.

Chapter 4

A stability analysis based on Contraction Theory for each module is described in this chapter. This analysis provides stability conditions that constrain the control parameters in the dynamical systems. Furthermore, a stability indicator that identifies the ability of the robot to complete its missions is developed based on the combination contraction property.

Chapter 5

Hospital environments tend to constrain robot autonomy. Challenges as safety, obstacle detection, hardware design or localization must be considered during the process of designing the robot. This chapter includes an overview about the hospital constraints and a description of the indoor environment and robot used to run the field experiments. This environment shares several characteristics with hospitals, such as people walking around, passages through doors and static obstacles. Then, a Kalman filter based approach is adopted to fuse sensory information from different sources, in order to obtain an accurate robot's pose estimated by the localization system. The performance of the localization system is evaluated through Monte Carlo tests.

Chapter 6

In this chapter, both simulations and field experiments are conducted to verify if the proposed architecture successfully produces smooth and uninterrupted robot motion over long distances without human supervision. Single and long-term missions compose the experiments. During the experiments, the stability indicator is used to verify if the robot successfully completes the assigned missions in time. The environment contains unpredictable obstacles that appear in the trajectory of the robot.

Chapter 7

The last chapter of this thesis presents the conclusions obtained from the results, the summary of contributions and guidelines for future works.

1.5 List of publications

Conference papers

- Jorge Silva, Cristina Santos and João Sequeira, *Developing a timed navigation architecture for hospital delivery robots*, In *IEEE 3rd Portuguese Meeting in Bioengineering (ENBENG)*, (pp. 1 - 4), 2013
- Jorge Silva, Cristina Santos and João Sequeira, *Navigation Architecture for Mobile Robots with Temporal Stabilization of Movements*, In *9th IEEE International Workshop on Robot, Motion and Control (RoMoCo)* (pp. 209-214), 2013
- Jorge Silva, João Sequeira and Cristina Santos, *A Stability Analysis for a Dynamical Robot Control Architecture*, In *Intelligent Autonomous Vehicles (IAV)*, (Vol. 8, No, 1, pp. 225 - 230), 2013
- Jorge Silva, Cristina Santos and João Sequeira, *Timed Trajectory Generation Combined with an Extended Kalman Filter for a Vision-Based Autonomous Mobile Robot*, In *Intelligent Autonomous Systems 12* (pp. 67 - 79). Springer Berlin Heidelberg, 2013
- João Sequeira, Cristina Santos and Jorge Silva, *Dynamical Systems in Robot Control Architectures: A Building Block Perspective*, In *12th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 82 - 87), 2012
- Jorge Silva, Cristina Santos and João Sequeira, *Timed Trajectory Generation for a Vision-based Autonomous Mobile Robot in Cluttered Environments*, In *9th International Conference on Informatics in Control, Automation and Robotics (ICINCO) (pp. 431 - 434)*, 2012
- Jorge Silva, Vitor Matos and Cristina Santos *Generating Timed Trajectories for Autonomous Robots*, In *International Workshop on Bio-Inspired Robots*, 2011

Chapter 2

Context and Related Work

This chapter begins by describing service mobile robots commercially available or expected to be on the market in the near future. The description specifies their physical features, sensors, tasks and navigation techniques. A comparative analysis between each category of service mobile robots and hospital delivery robots is given.

Service mobile robots must be able to autonomously navigate in dynamic and cluttered environments, in order to complete their missions. The control architectures that rule these robots should integrate several modules of control, such that the missions are successfully completed. Section 2.2 gives an overview about the control modules that architectures should have, namely techniques and approaches to address local and global path planning, localization and timing control.

Once the modules that compose the architecture are identified, it is fundamental to detail the intrinsics and connect them, such that the architecture deals with real-time applications and unexpected events. Section 2.3 reviews robotic architecture paradigms, fundamental to obtain a well-designed architecture.

Additionally, it is essential to ensure that the architecture converges to an equilibrium state. Section 2.4 describes methods for stability analysis of nonlinear systems. The chapter ends with a summary describing the weaknesses of the related work addressed in this thesis.

2.1 Service Mobile Robots

In the last years, the number of service mobile robots has significantly increased [14]. From cutting grass to floor cleaning, transporting laundry in hospitals or materials in industrial facilities, mobile robots are performing some of the trivial tasks that humans regularly do.

In the following section, a description of available or expected commercial robots, including cleaning, lawn mowers, tele-presence, assistive, smart walkers and delivery robots is given. This summary aims to provide the sensors characteristics, goals and limitations of each robotic application and compare them with delivery hospital robotic applications.

2.1.1 Cleaning and Lawn Mower Robots

Cleaning robots perform tasks such as brushing, scrubbing and vacuuming dirt and/or dust on the floor [15–18]. These robots are designed to be inexpensive and small. Simple heuristic rules define their random navigation patterns. Typically, infrared and contact sensors are employed to detect obstacles. Simplest cleaning robots use infrared sensors to obtain their current location and to prevent them from falling down the stairs [19]. Most expensive cleaning robots have other sensors for mapping and navigation, such as ultrasonic sensors, cameras and lasers [20–22]. To estimate their localization, they use gyroscopes or accelerometers.

More sophisticated navigation algorithms allow robots to avoid cleaning the same spot repeatedly. When running out of power or after finishing the cleaning tasks, they return to the dock charging station and continue cleaning where they left off in the last session. Delimited areas defined on the floor through magnetic strips or virtual walls ensure that the cleaning robots do not leave the specific area.

Large cleaning robots [23] are more expensive than small ones and have a physical portrait similar to hospital delivery robots. When obstacles are detected in the cleaning path, robots wait for the clearance of the corridor before continuing. This option imposes constraints in the environment hallways and rooms, as they should be completely free of obstacles, in order to ensure a proper cleaning operation. This is a suitable solution for cleaning robots, as cleaning missions can be continued after the obstacles have been removed.

Lawn mower robots have sensory hardware and control navigation techniques similar to cleaning robots. However, they are specially designed for outdoor environments. Simplest lawn mower robots [24–31] use ultrasonic sensors and bumpers to detect obstacles. Additionally, random algorithms based on heuristics are used for navigation. When obstacles are detected, these robots stop and only move after the obstacles have been removed. A more intelligent navigation system developed for the robot described in [25] measures the garden and calculates the shortest possible route to perform the lawn mower mission. In [32], the robot uses a combination of accelerometers, gyroscopes or electronic compasses to keep straight its trajectory.

Typically, lawn mower robots require a boundary wire around the lawn to define the mowing area. In some cases, the wire defines the location of the charging dock station.

The random navigation algorithms for cleaning and lawn mower robots are unfeasible for hospital delivery applications. As previously mentioned, when obstacles are detected, these robots stop and wait for the clearance of the path or change their movement based on heuristics. They assume that the cleaning and mowing tasks will be eventually completed, regardless of the direction followed after avoiding obstacles. However, this solution is not feasible in hospital hallways, since delivery hospital robots have to reach a specific location even when obstacles are detected. In terms of security, delivery hospital robots need a higher number of sensors to detect obstacles, as they are in general larger and taller than cleaning or mowing robots.

2.1.2 Tele-presence and Assistive

Nowadays, tele-presence, assistive and multi-task robots are navigating in hospitals and other public spaces, making surveillance missions, providing meetings between patients and doctors and assisting elderly or physically disabled people.

There are several tele-presence robots on the market, but only the RP-VITA [33] and, more recently, the Ava 500 from iRobot [34] are completely autonomous. Users specify the desired destination and these robots automatically navigate towards it without human intervention. Both robots use laser, three-dimensional (3D) imaging and sonars to safely avoid obstacles in cluttered environments, without bumping into people or objects. RP-VITA robots are already installed in 50 hospitals.

FURO-S [35] and FURO-K [36] are multi-service robots already operating in airports and malls, as guides or receptionists. These robots autonomously navigate in the environment using ultrasonic sensors and bumpers to detect obstacles, reaching velocities up to 1 m/s.

Expected to be commercially available in 2015, the Kompai robot [37, 38] is being designed to assist elderly and disabled people. This robot autonomously navigates to specific locations given by the user. To detect obstacles, rear and front bumpers, sonars, infrared, laser and 3D vision are applied. Furthermore, traditional techniques for laser-based simultaneous localization and mapping are used.

Giraff [39] is another assistive robot expected to be launched to the market in 2015. It uses a laser range finder on its base for obstacle avoidance. If Giraff detects an obstacle under 30 cm, it warns the user about the existence of an obstacle and stops to avoid the collision. Giraff uses an *a priori* map built offline from data acquired by the laser.

Most known non-commercially solutions for assistive health care, domestic or tele-presence robots include several research projects. The latest version of the health care robot Care-O-bot [40] is equipped with omnidirectional drives, range and image sensors for object learning and autonomous navigation. AMIGO robot [41] was designed for domestic service tasks. This omni-directional robot uses a 3D Kinect camera and a laser to detect obstacles. The Nursebot Pearl [42] is a personal mobile service robot that assists elderly people in domestic tasks, makes tele-presence and social interaction. This robot uses sonars and a laser for obstacle avoidance. CoBot [43] is a robot that interacts with humans in indoor environments, being able to achieve autonomous localization and navigation using a 3D Kinect camera, WiFi and a laser.

SIGAs robots [44] are operating in the facilities of a banking group as guides to clients. These robots are equipped with odometers, gyroscopes and RF sensors for localization. To detect obstacles, each robot uses 16 sonars mounted on a ring around it to detect obstacles. They are able to move up to six hours and automatically return to their charging station when the batteries are low.

Assistive, domestic and tele-presence robots have similar features to hospital delivery robots, as the moving velocity, navigation techniques and sensors for localization. They

autonomously navigate to specified destinations while avoiding bumping into people and other obstacles. After completing the missions, they automatically return to their docking stations for charging.

2.1.3 Smart Walkers

Smart walkers work as a physical support and navigational aid for frail and visually impaired people. Several research groups aim to integrate navigation properties in the smart walkers. Moreover, only a pair of these robots with autonomous navigation capabilities is commercially available: the Siemens c-Walker [45] and Guido [46].

c-Walker is equipped with multiple vision sensors, enabling the mobile walker to monitor its spatial surroundings and react to obstacles located in the trajectory. This walker guides the users to their destinations. On the other hand, Guido is able to select the best trajectory based on an *a priori* map, which is updated in real-time through a laser. To detect obstacles, Guido uses sonars and a laser.

Based on these characteristics, hospital delivery robots should present navigation and localization capabilities similarly to Guido and c-Walker. However, both smart walkers are tailored for low velocity applications. In hospital environments, the delivery robots are expected to reach higher velocities, leading to the necessity of more sensors to detect obstacles and faster navigation techniques.

2.1.4 Delivery Robots

In the back years, the development of Automatic Guided Vehicles (AGVs) mainly aimed to transport materials between different locations in controlled environments. The robots are able to follow a path, calculated at the beginning of the mission. If some event disturbs the robot movement, the mission can fail.

Kiva mobile robots from Amazon [47] compose the most common delivery automated systems for warehouses. These robots can lift approximately 450 kg and travel at 1.3 m/s. They follow bar-code stickers on the floor detected through onboard cameras. As the robots move, encoded information to estimate their coordinates in the warehouse is being read. Simultaneously, the navigation system corrects the position of the robots if they are not centered at the stickers.

Based on this simple navigation approach, robots are mechanically simpler and, subsequently, cheaper. This system presents good performance in controlled environments, such as warehouses and distribution centers. In hospitals, this navigation approach is unfeasible, once it is not possible to fill the ground with tapes and assure that obstacles are never placed over them. Obstacles appear in unpredictable locations and the navigation approach must be able to deal with this uncertainty.

2.1.5 Hospital Mobile Robots

Robots have already acquired a place in the health-care world. In 2013, approximately 1000 robots were performing delivery missions in hospitals. Those numbers are expected to grow 10 times in the next five years. The good reputation of delivery robots is leading hospital managers to "hire" them for their services as workers. Due to the need for delivery robots, as well as potential strategies for their implementation in hospitals, several studies have been made [5–7, 48, 49]. These studies have shown that automated deliveries improve the efficiency of hospital transportation, overall organization, cost reduction, and allow nurses and housekeepers to focus their attention on health-care services, rather than doing delivery tasks. Expenses in relation to logistic account are approximately 30 - 46% of a hospital's total budget and nurses spend 10% of their time performing logistics, rather than patient care tasks. These numbers can be reduced by adopting automated delivery strategies.

The analysis in [50] showed that the installation of delivery robots in a mid-size hospital has better performance than its current system of 3 clinical laboratory delivery employers. Robots are able to deliver over 85% of all goods in hospitals, such as blood samples, linen, trash, food and medical material according to [7]. Mobile robot solutions are cost effective and delivery performances are similar to human based delivery systems, in terms of average time, but with a significant reduction in the standard deviation [51].

Preliminary remarks about delivery robots already operating in hospitals are presented in [52, 53]. In [53], robots were responsible for transporting blood samples between different locations. The hospital staff showed some familiarity with the robots, but they did not fully rely on them. The way that organizational factors in a hospital environment affect the response of the staff to delivery robots are addressed in [52]. This study was conducted for 15 months in a hospital where 7 robots were operating since 2003. The study revealed distinct opinions according to the staff health-care service. Medical units were reluctant to the benefits introduced by delivery robots. They enhanced the frequent interruptions and collisions caused by them. On the other hand, postpartum units accepted positively the delivery robots because heavy loads were no longer transported by them, but by the robots. Even though delivery robots provide a collective benefit to the hospital organization, some members needed to adjust their workflow and do additional work, which was not always well accepted. These studies suggested that future delivery robots must cause fewer interruptions to the staff workflow and improve social interactions.

The task of designing delivery robots for hospital environments should consider some important challenges, in order to fulfill the requirements identified by the survey in [6]: (1) safety - robots should not collide with people and obstacles. (2) Obstacle detection - the robot should have onboard sensors such that the scanned area is maximized. (3) Path planning - the robot should plan the shortest path to go from the starting point to the destination point. (4) Navigation - the robot must continuously know its pose and do not look clumsy or hostile when it moves. (5) Velocity - the robot can travel at high velocities or slowing down

in some areas. (6) Logistics - cooperation between multiple robots. (7) Automatic doors and elevators - the robot should communicate with automated devices, in order to be able to move between floors and to other areas. (8) Installation of charging and parking stations. (9) User interface - hospital staff communicates with the robot through an interface, in order to specify the delivery destination. (10) The robot must deal with emergencies, such as evacuations.

Some academic research groups have tried to develop solutions for mobile robots focused on delivery tasks for hospital environments. The work developed by [54] proposed the use of natural landmarks of the environment, namely the florescent lamps on the ceiling, to localize the robot instead of distributing artificial landmarks along the environment. This robot is efficiently operating in a hospital in Hong Kong since 2001. The robot uses 24 sonars, 24 infrared sensors and bumpers to detect the presence of obstacles. A potential field approach for obstacle avoidance based on [55] has been adopted. The maximum velocity of the robot is 0.3 m/s.

The i-Merc mobile robot developed by [56] delivers meals in health care services. It has more capabilities than the robot proposed by [54], since i-Merc communicates with other mobile robots, elevators, automatic doors and with the management system. Ultrasonic sensors and bumpers are used to detect obstacles. i-Merc reduces its velocity when an obstacle is detected. If i-Merc is unable to avoid the obstacle, it stops and waits for the clearance of the path. A combination of odometry, an inertial sensor and landmarks detected by an optical sensor is used to obtain the robot's pose.

The Muratec Keio robot proposed by [57] is an omni-directional mobile robot that is under development. The robot includes a wagon truck to accommodate the hospital stuff. Laser and ultrasonic sensors are used for obstacle avoidance. Navigation is based on the potential fields method proposed by [55], and different time scales are defined to deal with unexpected events. This robot reaches a maximum velocity of 0.5 m/s, higher than the robots proposed in [54, 56].

The Merry Porter in [58] has a unicycle structure with a front steering wheel that reaches a maximum speed of 1.5 m/s, carries goods until 180 kg and handles overtakes on the floor up to 3 cm. It has a telescopic turret for placing sensors at a variable height, in order to avoid occlusions due to surrounding people. Merry Porter uses a laser on top of the turret to localize itself and another laser and 16 ultrasound sensors to detect obstacles. When the robot is not able to deal with obstacles, it stops and waits for the clearance of the path. Navigation is based on the standard potential field approach [55]. During navigation, the robot is allowed to deviate from its trajectory to avoid obstacles, but it is never allowed to exit a convex area defined by the roaming stripes algorithm [59].

Robotic commercial solutions that accomplish the aforementioned challenges are already

implemented in hospital environments. One of the pioneer robots deployed in hospital environments is the HelpMate robotic courier proposed by [49, 60] and manufactured by HelpMate Robotics Inc.. More than 150 HelpMate robots are being used in hospitals in Europe, Japan, Canada and United States. HelpMate robots combine laser, sonars, bumpers and infrareds mounted around the robot to detect obstacles and the ground. They use an offline built map with the desired stop locations embedded into its onboard memory to autonomously navigate throughout the hospital facilities. Radio signals provide communication with elevators. HelpMate robots apply a simple rule to avoid dynamic objects: they are allowed to leave their defined paths within a specified distance to obstacles or stop until the obstacles have disappeared. Its pose is estimated through an odometric system and natural landmarks. HelpMate robots have a flashing warning light to notice their presence and a stopping button to be used in emergency cases.

RoboCart robots [61] are produced by California Computer Research Inc. and designed to operate in clinical laboratory environments. These robots have more limited resources than HelpMate robots. They are able to follow a fixed path, specified by a tape placed in the floor. When their onboard sonars detect obstacles, they stop and wait for the clearance of the path. A RoboCart robot occupies about the same floor area as a human and moves at approximately 0.58 m/s.

TUG robots [62] commercialized by Aethon Inc. are being used in hospitals to deliver medical records and supplies (specimen, food and laundry) since 2008. They make approximately 50000 deliveries each week. TUG robots are equipped with laser range finders, ultrasound and infrared sensors to detect obstacles and safely navigate in the hospital. When several obstacles obstruct the corridor, TUG robots wait for clearance. Their operating system converts Computer Aided Design (CAD) drawings of the hospital into a map understandable by them. TUGs' onboard sensors track their pose within the embedded map of the building. They do not require the installation of radio markers, magnetic strips or other space delineating technologies like RoboCart. TUG robots use the hospital existing WiFi system to communicate with elevators, automatic doors, fire alarms and with other TUG robots (for optimal delivery and performance).

RoboCourier [63] and SpeciMinder [64] are made by CCS Robotics and commercialized by Swisslog Inc.. These robots have the same appearance and overall performance. They perform approximately 32000 delivery tasks and travel around 3200 Km per year. Lasers, sonars and bumpers are used to detect obstacles and, consequently, navigate in dynamic environments in a safe way. When obstacles are detected, they calculate the fastest path to circumnavigate them and proceed to the next goal. If the obstruction prevents the circumnavigation, they wait for a specified period for the clearance of the objects. When stuck, these robots emit a warning signal. Batteries are recharged during the interval between missions. Verbal announcements are emitted to alert people of their presence and intentions.

Matsushita's HOSPI robots [65] are commercialized by Panasonic Inc. and have suffered

several improvements along the past years. HOSPI robots are already operating in several Japanese hospitals. They use laser range finders to detect obstacles on their way and stop moving when obstacles are detected within a specified area around them. Otherwise, they automatically adjust their route. They know the map of the building *a priori* and the map system is sufficiently flexible to handle extensions to the existing facilities. HOSPI robots call and take elevators automatically. They can move at a maximum velocity of 1 m/s and work up to 7 hours before needing to recharge.

QC BOT [66] is produced by VECNA Inc. and save approximately 50 hours per week spent on medicine transportation. QC BOT robots use lasers and a camera to detect the environment and avoid obstacles. The navigation of QC BOT robots does not require modifications in the hospital facilities. Furthermore, they include cameras for video conferences between patients and medical staff.

Swisslog's TransCar [67] uses laser, ultrasonic, bumpers and photoelectric sensors for navigation. Safety and warning zones are defined around the robots. Transcar robots stop immediately when an obstacle is detected within the safety zone or emit sounds and warning lights when an obstacle is detected within the warning zone. TransCar robots reach a maximum velocity of 1.6 m/s. Wireless communication is used to communicate with elevators and doors.

This literature review underlines that current hospital delivery robots present multiple similarities, which reveals efforts to achieve a unified solution. To navigate, lasers are suitable sensors, since they cover a large area with greater detail of the robot surroundings. Furthermore, a stopping area around the robot must be defined for safety. Delivery robots stop and wait for the clearance of the path, always that an obstacle is detected in the safety area. This safety measure prevents the robots to change abruptly its trajectories or to collide with obstacles. The map of the environment is provided *a priori*, since the map is not built or updated while the robots are performing the missions. Although the success of commercial solutions to perform delivery missions in hospitals, the cooperation between robots and hospital staff needs to be improved [52]. Furthermore, current delivery robots do not consider time constraints as a condition to successfully complete their missions. Robots can be stuck for large periods of time and do not compensate the delay after the clearance of the path.

2.1.6 Comparative Analysis

This section overviews a comparative analysis (see table 2.1) in terms of navigation capabilities between each category of service mobile robots and hospital mobile robots. In terms of local path planning, delivery robots as AGVs follow a predefined path and stop when obstacles are detected. The remaining service mobile robots have sufficient capabilities to detect and circumnavigate obstacles. The ability to plan global paths is not present in cleaning and mowing robots. They usually follow heuristic rules and assume that sooner or later their

missions are performed. In terms of the capability to reach high velocities, only smart walkers are tailored for low velocity applications. The remaining service mobile robots can reach velocities up to 1 m/s. In terms of localization, all service robots have sensors to estimate their poses.

TABLE 2.1: Comparative analysis in terms of navigation capabilities (✓ - fulfilled, X - not fulfilled)

Ability	Cleaning/ mowing robots	Tele-presence/ assistive robots	Smart Walkers	Delivery robots	Hospital robots
Local Planning	✓	✓	✓	X	✓
Global Planning	X	✓	✓	✓	✓
High Velocities	✓	✓	X	✓	✓
Localization	✓	✓	✓	✓	✓

2.2 Control Architectures

Control architectures for autonomous mobile robots must be able to solve the robot navigation problem. In general, this problem is handled by answering three questions,

- Where am I? The robot must know its current location to make decisions about its next movement.
- Where am I going? The robot has to know the goal location where its mission ends.
- How do I get there? Once the robot knows the goal location, it should be able to find a path to get there.

These questions are solved by including into the robotic architecture modules that address each specific question. The first question is answered by the localization module, which provides an estimate of the robot's pose. The second question is addressed by an entity that specifies the location where the mission ends. This entity can be the user or a module of the architecture responsible for defining the goal locations. The last question is solved by including local and global path planning modules. Information about the environment is used to prevent the robot to be stuck in deadlock situations. The inclusion of both local and global path planning modules provides the advantages of both methods operating solely.

In this work, time is a variable used to ensure that the robot performs its missions within a specified time constraint. The problem of navigation becomes more complex and another question needs to be solved,

- How long do I have? The robot needs to know if it is in time, delayed or in advance to accomplish its mission, and act according that.

This question is solved by adding a timing module responsible for generating velocity commands, allowing the robot to reach the goal location within the specified time constraint.

This section describes different techniques and approaches used to address local and global path planning, localization and timing control.

2.2.1 Local Path Planning

This section overviews a taxonomy of obstacle avoidance methods for mobile robots. It shows that there is not a unified solution addressing all local path planning problems. The solution depends on the robotic requirements. For instance, there are solutions that provide a smooth behavior in narrow passages and cluttered environments. Other solutions consider the dynamics of the robot or the velocity of obstacles, and others are well suited for robots moving at high velocities.

In cluttered environments, such as hospitals, it can be desirable that the robot presents a smooth behavior while avoiding obstacles. Moving at high velocities is not a critical point because the robot will move at low velocities when obstacles are detected or even stop if obstacles are too close.

Obstacle avoidance methods can be divided into two groups: methods that compute the motion in one-step and in multiple steps. One-step methods reduce the sensory information to a motion control, and can be divided into heuristic and physical analogies methods. Motion controls can be a set of directions or velocities. Multiple steps methods compute an intermediate set of candidate motion controls, over which the best candidate motion control is selected.

2.2.1.1 Heuristic Methods

Heuristic methods were the first techniques used to generate motion in two-dimensional scenarios based on sensory information. Being heuristic, these methods can neither be exhaustively tested nor proved effective in all cases. In fact, poor solutions or no solutions at all may result from this method [68]. However, in easy environments, heuristic methods have shown to be effective and, when reaching a solution, it is quickly calculated.

Main heuristic methods derive from classical planning methods. For instance, the Bug algorithm [69] defines a set of rules for obstacle avoidance. When an obstacle is detected, the robot follows the obstacle boundary and repeats the same procedure if new obstacles are detected. The Collision Cone [70] adopts a set of strategies to avoid collisions when the robot and an obstacle are on a collision route. Additional heuristic methods based on classical planning methods can be seen in [71–73].

2.2.1.2 Physical Analogies Methods

Physical analogies methods apply mathematical equations from physics to sensory information, in order to compute the motion commands. The most known methods are the potential field approach [55] and the Dynamical Systems Approach (DSA) [74].

Potential Fields

The potential field approach considers that the robot is viewed as a particle that moves in space under the influence of a force field. Attractive forces that attract the robot towards the target and repulsive forces that move away the robot from obstacles constitute this force field.

The major problem of potential fields is when attractive and repulsive forces cancel out each other and the robot stops before reaching the goal location. This is the so-called local minima problem. Other problems include the instability of the robot dynamics in narrow corridors and when the robot moves at high speeds.

Simple heuristic solutions attempt to avoid the local minima problem by adding rotational fields around obstacles or random fields to the force field. However, these solutions might be insufficient to eliminate the local minima problem. On the other hand, more complex solutions such as, harmonic functions [75], circulatory field approach [76] and biharmonic potential fields [77] compute a potential field free of local minima.

One of the most known methods that stem from the potential field approach is the Virtual Force Field (VFF) method [78]. This method includes certainty grids [79] for obstacle representation and potential fields for navigation. Each cell of the grid contains a probability indicating whether the respective area encoded by the cell is free or occupied by obstacles. Each cell exerts a virtual repulsive force proportional to its value. The robot will avoid cells with repulsive forces and follow empty cells. However, in some circumstances, the robot is unable to pass through doorways because of the repulsive forces from both sides of the doorway. Furthermore, a smooth control signal is required to the steering motor because drastic changes in the resultant forces cause fluctuations in the steering control.

Dynamical Systems Approach

The conceptual framework of the dynamical systems approach (DSA) is based on the theory of nonlinear dynamical systems (see [80]). This approach was proposed by [74] to replace the transient solutions of the potential field approach by attractor solutions of a dynamical system. DSA provides a smooth control signal to the steering motor control. The dynamical system is modeled by a system of differential equations with attractors and repellers, standing for goals and obstacles, respectively.

An intelligent choice of planning variables makes possible to guide the robot over complex trajectories from stationary stable states. In [74], the robot's heading direction was selected as the planning variable to control and guide the robot towards the goal location while, simultaneously, avoiding obstacles. The planning variable lies in or near a fixed point attractor at all times, and this is a major advantage over potential fields whose point of

attraction is only accomplished if the robot reaches the global minimum. Once a behavior is generated from nonlinear dynamical systems, theoretical tools from mathematical theory can be used to obtain a suitable robot's behavior. The dynamical systems approach offers robustness against small perturbations, providing the possibility to fuse new inputs into the system without changing its properties.

Over the last years, several researchers have adopted the DSA method to address local navigation of mobile robots. In [81], DSA was combined with a neural dynamical field to endow a mobile robot with memory ability, enabling it to store the positions of obstacles. In [82, 83], DSA was extended for real-time path planning on a mobile robot working with low level sensory information. Dynamical systems were used to control the velocity and direction of a mobile robot, able of obstacle avoidance and target attraction. This approach was later used in several mobile robots [1, 84–86].

Other robotic behaviors were generated using the DSA method, such as corridor following and walls avoidance [87], modeling formations [88], predicting routes [89] or decision among different behaviors [90]. Furthermore, nonlinear dynamical systems can include harmonic functions [91] or neural networks [92] to obtain customizable behaviors in real-time environments. Other variations to the obstacle avoidance behavior include the addition of new variables to the motion equation [93].

DSA was extended for obstacle avoidance and target attraction for an anthropomorphic arm [94, 95], and [96] showed that DSA could be applied to perform obstacle avoidance in Cartesian and joint spaces.

2.2.1.3 Multiple step Methods

This section presents the most known multiple steps methods for obstacle avoidance.

Vector Field Histogram

The drawbacks of the VFF method motivated the development of the Vector Field Histogram (VFH) [97]. VFH uses a two-stage data reduction to provide detailed information about the obstacles in the environment. This representation allows the robot to identify narrow passages and move across them without oscillations. Fluctuations in the steering control are also eliminated. The main disadvantage of VFH is the necessity to define a threshold that selects the candidate direction for the robot. If the threshold value is too large, the robot can achieve positions very close to obstacles and eventually collide with them. If the threshold value is too low, some potential candidate directions will be excluded and the robot will not move through narrow passages. Additionally, VFH does not consider the robot dynamics, which can be problematic in cluttered environments.

An improvement to VFH, named as VFH+ is proposed in [98]. This method applies a four-stage data reduction to compute the direction of motion and to solve several VFH disadvantages. For instance, VFH+ considers the robot dynamics to reduce the parameters inherent to VFH. Nonetheless, two new thresholds are introduced to obtain smooth and more

reliable trajectories. VFH+ applies a cost function to select the desired direction, reducing the hysteresis provoked by oscillatory movement in narrow passages.

Curvature-Velocity

The Curvature-Velocity Method (CVM) [99] selects the best linear and angular velocity from a set of candidate velocities, in order to maximize an objective function that guarantees a trade-off between safety, speed and direction to the goal. CVM calculates in real-time a curvature trajectory that considers the dynamics of the robot and, simultaneously, avoids hitting the obstacles.

The Lane-Curvature Method (LCM) [100] combines the CVM with a directional method called lane method to yield a collision-free and smooth motion. LCM divides the environment into lanes and selects the best lane to follow the desired direction. Then, the CVM uses the selected lane to determine the linear and angular velocities. However, both CVM and LCM do not consider the velocity of obstacles to calculate the set of candidate velocities.

Dynamic Window Approach

The Dynamic Window Approach (DWA) developed by [101] selects a velocity for the robot from a set of candidate velocities. The search space is restricted to velocities that allow the robot to stop safely, without colliding with obstacles, and to the admissible velocities that the robot can reach within the next time interval. DWA considers the robot dynamic constraints, such as maximum velocities and torque limits. A cost function, based on a set of heuristics, selects the best candidate velocity. DWA keeps low complexity even at high velocities. However, this method is not able to deal with possible future points of collisions and considers that obstacles are avoided if the robot can stop without colliding with them.

The Global Dynamic Window proposed in [102] combines the DWA with a grid based global navigation function, in order to provide goal directness and free space connectivity.

Both DWA and CVM yield good results for obstacle avoidance, even at high speeds. The local minima problem in these approaches can be solved by combining other techniques [75, 77].

Velocity Obstacle

The Velocity Obstacle (VO) proposed by [103] is an extension of DWA and CVM. VO considers the velocity of the obstacles to calculate the candidate velocities, being well suited for dynamic environments. VO method reduces the representation of the robot to a single particle and enlarges the obstacles by considering the robot's radius. Then, the velocity of each obstacle is estimated. A collision cone is created and the set of colliding velocities is defined. The best candidate velocity outside the range of colliding velocities is selected by a cost function based on a set of heuristics.

Probabilistic Velocity Obstacle

The Probabilistic Velocity Obstacle (PVO) was proposed to handle with the uncertainty on the velocity and on the radius of the obstacles [104]. Both dynamic and kinematic constraints of the robot are considered to obtain the current velocity space. Similarly to VO

method, the velocity candidate is selected based on a set of heuristics.

In [105], the PVO method was combined with occupancy grids [79], in order to consider the uncertainty in the perception system. The probability to collide with obstacles is used to obtain the velocity space over which a candidate velocity is selected.

Time-Varying Dynamic Window

The Time-Varying Dynamic Window (TVDW) method developed by [106] associates the DWA and VO methods. TVDW combines the benefit of using the well-structured steps of the DWA method and the ability to operate in dynamic environments provided by the VO method. TVDW considers the holonomic constraints of car-like robots. Furthermore, the velocity of obstacles was discretized to allow operating at high velocities. A set of heuristics, similar to the DWA method, calculates the best candidate velocity.

Nearest Diagram Navigation

The Nearest Diagram Navigation (NDV) [107] computes some high-level information as an intermediate step to calculate the robot's motion. The idea behind this approach is to apply the "divide and conquer" strategy, in order to reduce the complexity of the environment and the difficulty of the navigation problem. This method allows robots to move in complex, cluttered and dense scenarios, where other methods possibly would fail.

Obstacle-Restriction

The Obstacle-Restriction Method (ORM) developed by [108] suggests to use the available obstacle information along with the two-steps iteration process. First, based on the obstacle distribution, a local procedure selects the set of candidate motion directions. The second step associates a motion restriction to each obstacle, in order to calculate the selected direction.

This method avoids technical problems inherent to classical methods, such as local trap situations (U-shape obstacles), oscillations or instable motions. ORM has no internal parameters to tune, which simplifies its implementation. Similar results to the NDV method were obtained in open spaces, but in dense and complex scenarios, ORM shows a better performance [108].

Fuzzy Logic

Fuzzy logic [109, 110] has been applied in mobile navigation, due to its ability to cope with large amounts of uncertainty inherent to real environments. However, fuzzy logic lacks in terms of correctness, consistency and completeness of its rules.

Artificial intelligence techniques have been used to overcome the insufficient information of the environment. Combinations of fuzzy logic with reinforcement learning [111, 112], evolutionary algorithms [113], genetic algorithms [114] and neural networks [115] have been adopted to extract a set of fuzzy rules after a training phase. Furthermore, fuzzy logic is combined with classical techniques for obstacle avoidance, such as potential fields [116].

2.2.2 Global Path Planning

Global path planning is responsible for generating a collision-free path that connects the initial position of the robot to the goal location. Before planning the global path, the robot must know the geometry of the environment. Global planning does not consider the vehicle stability or the existence of unexpected obstacles. These are left to the local planning controller.

Global path planning can be separated into two problems. First, available information of the environment has to be represented by a configuration space. Second, a search path algorithm should find the best path based on the user's criteria, such as the distance or the time to reach the goal location. This is the so-called search path problem.

2.2.2.1 Representation of the environment

Methods to represent the environment characterize its geometrical information by a mathematical model understandable by the robot. The representation of the environment can be constructed offline if the map is already known, or online from sensory information acquired by the robot

Two-dimensional (2D) grids can simplify the representation of indoor environments, since these environments are mainly composed by linear structures, such as lines and planes. Even though several methods can build 2D world representations, they must share important features. First, they must be compacted, in order to be efficiently used by other modules. Second, they should be adapted according to the type of the environment. Finally, the representation of the environment must contain the uncertainty inherent to sensory information.

Cell Decomposition

Cell decomposition methods divide the free space of the environment into cells that compose a grid (metric representation). A non-direct graph, called connectivity graph, represents the adjacency relation between the cells. The outcome of these methods is a sequence of cells called channel.

Cell decomposition methods do not rely on any predefined features of the environment and offer a constant access to the grid cells. On the other hand, when dealing with large environments, potential discretization errors and high memory requirements are problematic for real-time applications.

Line Maps

Line maps represent indoor environments by connecting a set of data points (Cartesian coordinates), in order to create a line that minimizes the square distance to all points. Line maps have several advantages over cell decomposition methods as they require less memory, are better scalable with the size of the environment, do not suffer from discretization problems and are more accurate. On the other hand, the large number of lines needed to represent the environment, as well as the assignment of data points to individual lines are the major

disadvantages. A common solution to overcome these drawbacks is to use the split-and-merge algorithm developed by [117]. Unfortunately, this algorithm does not guarantee that the resulting model is optimal, *i.e.*, the minimization of the square distance to all points in not guaranteed.

Landmark-based Maps

In environments with sufficient distinguishable features, methods based on landmarks detection have been extensively used, as for instance, the FastSLAM algorithm [118]. The locations of the landmarks are known *a priori* through two-dimensional Gaussians. The robot estimates its position, over time, based on the detected landmarks. However, this representation largely fails when the environment lacks of distinguishable features.

Roadmap Methods

Roadmap methods consist of capturing the connectivity of the free space in a network of one-dimensional curves (called the roadmap), such that all starting and goal points of the free space are connected by a path. The roadmap is a set of standardized paths that reduce the path planning problem to the problem of assigning the starting and final location to points in the roadmap [119]. Most known roadmap methods include visibility graphs [120], Voronoi diagrams [121], probabilistic roadmaps [122] and rapidly exploring random trees [123].

The major disadvantage of roadmap methods is their representational incompleteness. If a path exists in the map, there is no guarantee that it will be found using a roadmap method.

Topological Maps

In contrast to geometric structures methods such as line maps or cell decomposition, topological maps represent the environment through a set of regions. A node represents a region, and nodes are connected through arcs. According to [124], a region can be a corridor or a room, and arcs represent doors, stairways or elevators. More generally, the environment can be viewed as a graph-like structure and regions can result from a tessellation procedure of the environment.

Topological maps are computationally lighter than geometric structural methods. They provide simpler representations of complex and large environments, which is suitable for navigation purposes [125–127]. On the other hand, topological maps provide a coarse representation of the environment, which makes it unsuitable for mapping.

Combination of Topological and Metric representations

The majority of works addressing autonomous navigation combine topological and metric representations into the same architecture. A metric map represents the environment and several topological maps, partitioned by a Voronoi diagram, are obtained from the underlying metric map [126]. The planning depends on the size and shape of the topological region where the robot is located. The search algorithm is applied to the metric map, but the topological map provides the points that define the robot's initial and final location. In [128], the partition of the environment is performed on a graph-cut clustering model, rather than on a Voronoi diagram. Consequently, the computational cost is reduced and the average loss

in the paths' optimality is not deteriorated. The resultant performance is similar to [126], showing that, planning based on topological maps leads to path lengths only a few percent greater than the metric based ones.

In [127], a topological map is used for global path planning and a local metric map for local path planning. The set of waypoints that the robot must reach is generated in the local metric map. Only this map is re-planned at each instant of time, rather than the global metric map, as in [126]. Consequently, this work contributes to the simplification and design of navigation architectures.

In the work proposed by [129], topological and metric maps are alternated with the help of visual landmarks and in [130], they are alternated according to the precision needed for navigation. In large areas, precision is less important than robustness and global consistency. Thus, topological maps are well applied. In small areas, precision is important and local metric maps are used.

Some of the aforementioned approaches use metric maps in small sections of the environment and topological maps in large and dynamic environments. However, in some approaches, the search path algorithm successfully solves the global path planning in topological maps. For the purposes of navigation, a metric map is not required to calculate the robot trajectory and the topological map is sufficient to drive the robot towards the goal locations [131, 132].

2.2.2.2 Search Path Problem

The search path problem consists on selecting the path that minimizes or maximizes a specific cost function. One of the first algorithms that solve the shortest path problem is the Dijkstra's algorithm [133]. The original Dijkstra's algorithm solves the problem for a graph with nonnegative edge costs in computational time $O(|V|^2)$, in which $|V|$ is the number of nodes. Later, a common implementation of this algorithm applies a priority queue that runs in $O(|E| + |V| \log |V|)$, in which $|E|$ is the number of edges connecting the regions.

The A* algorithm proposed in [134] is considered as a generalization of the Dijkstra's algorithm. The main difference lies in the fact that A* uses a distance-plus-cost heuristic function that determines the order in which each node in the graph is explored. Consequently, it reduces the size of the sub-graph that remains unexplored. A* uses a best-first search algorithm to find the minimum cost path between regions, similarly to the Dijkstra's algorithm. The time complexity of the A* algorithm depends on the heuristic complexity. The higher the heuristic knowledge about the goal distance, the faster the search problem is solved.

In the Lifelong Planning A* (LPA*) algorithm [135], both incremental and heuristic information are combined to reduce re-planning times. LPA* finds the shortest path from a given start node to a goal node, while the edges costs and vertices can be changed. Even though

this method provides the ability to re-plan its paths after discovering unknown obstacles, the resultant planning time can be in the order of minutes for large environments.

Later, the D* algorithm (Focused Dynamic A*) introduced by [136] allowed to solve search problems, with a reduction of computational complexity of one to two orders of magnitude over the previously described search algorithms. It uses a clever heuristic search method over which repeated A* searches by locally modifying previous search results.

Recently, the D* Lite algorithm proposed by [137] is implemented based on the same navigation strategy as D*, but being algorithmically different. It uses a single criterion for comparing priorities, which simplifies their maintenance. D* and its variants solve the search problem faster than other algorithms, which makes them suitable for real-time applications.

In a delivery robot, considering that a topological map represents the environment, the search path problem consists on calculating the sequence of topological regions that the robot should follow to reach the final location. The number of regions in hospital environments is in the order of hundreds. Therefore, the computational time required by any of the search path algorithms is not relevant, and any of the algorithms previously described can be used in the present application. Furthermore, the search path problem is solved before the robot starts its missions.

2.2.3 Timing Control

The generation of timing movements has been addressed in different ways according to the control architecture paradigm that rules the robot [3, 9]: classical approaches, in which planning and control are conceptually separated [119]; and behavior-based approaches, in which exteroceptive sensory information connects perception and action [138].

Classical approaches imply that space and time constraints on the robot motion must be known *a priori*, which makes it difficult for robots to operate in unknown environments constantly subject to disturbances. Despite the efficiency of classical planning algorithms in theory, the path planning remains separated from exteroceptive perception and control. Such systems are inflexible and do not allow online adjustments to the planned actions, *i.e.*, if external perturbations disturb the generation of timing movements, this timing change is not rectified. The overall result is a robotic system with a lack of responsiveness and limited real-time capabilities.

Behavior-based approaches allow a continuous coupling to exteroceptive sensory information. The dynamics of uncontrolled environments are considered by the robot, so that it can adapt its behavior accordingly [138]. Consequently, perturbations on the timed movement of the robot can be corrected in real-time. However, the generation of timed movements is more difficult to handle in behavior-based approaches than in classical approaches.

Solutions to the generation of timed movements, namely, the way that rhythmic movement are generated, have been biologically inspired by analogies with nervous systems [139, 140].

A specialized neural network, called Central Pattern Generators (CPGs), located in the vertebral spine [141, 142] generates self-adjusted rhythmic activity. CPGs do not require sensory feedback or higher-level commands to initiate rhythmic movement. This rhythmic activity has been mathematically described in literature by nonlinear dynamical systems with limit cycle solutions.

The dynamical systems theory was initially proposed to understand the patterns of rhythmic movements coordination in [143]. The basic concepts were motivated by physical theories of pattern formation [144]. Coordination patterns of rhythmic movements are represented as collective variables, and the identification of pattern dynamics are described as equations of motion. The dynamical systems theory provides theoretical concepts, such that a single model integrates the theory to initiate, generate and modulate the rhythmic movement. The work in [143] showed that complex nonlinear biological systems could be suitably described by simple equations. Furthermore, it showed that stability is an important property of a coordination pattern. Measures of stability, namely near phase transitions in the generation of movement have led to prediction of a coordination rhythmic pattern.

Later, the work suggested by [145] described a theoretical attempt to extend the dynamical systems theory with stable limit cycle solutions, and their adaptation to understand the coordination of discrete movements.

The work proposed by [146] provided a dynamic theory that includes online linkage with time varying sensory information. The dynamic theory of discrete and rhythmic movement is based on [145, 147] and expresses coordination movement patterns (postural states) in terms of motion equations. Variables that characterize motion are identified as attractor solutions.

In [81], a dynamic theoretical approach is proposed to endow task-related variables with simple fixed points and with limit cycle dynamics. Vision information is coupled into the dynamics of the action variables, such that the timing movements are elicited by such information. Compensatory movements are generated in response to perturbations of the visual motion. This work showed that the contributions to the action dynamics can be viewed as finite individual forces. Therefore, it is possible to understand how such contributions may interact and cooperate without losing their characteristic properties when coupled.

One of the first approaches focused on generating rhythmic movement for wheeled mobile robots was proposed in [140]. This work suggested using limit cycle attractors to generate a single continuous pattern of rhythmic movement. However, the integration of discrete movements into the generated patterns was not addressed.

The theoretical approach proposed by [81] was applied in two manipulators to generate rhythmic and temporal discrete movements by coupling two dynamical systems [148]. The manipulators achieved synchronization, revealing an independency relatively to the specification of their individual parameters. The model consists of a timing layer that uses a nonlinear oscillator with a Hopf bifurcation to generate rhythmic movement, and attractor states to achieve discrete movement. By coupling sensory information, sensor driven initiation and

termination of movement are enabled.

The work in [1] was based on previous work [148] and applied temporal stabilization and sequences of movements for an autonomous mobile robot with low-level sensory information to steer action. This work demonstrated the robustness of nonlinear dynamical systems to deal with small perturbations, while achieving good performance on the mission success. Stability and controllability of the architecture is weakly verified by ensuring the time scale separation between the multiple dynamical systems. This approach was then extended to generate temporal coordination between two simulated mobile robots in [149]. Both robots were able to complete missions within non-structured environments. In [150], temporal movement was generated for a robotic arm whose mission was catching a ball acquired by vision information. The dynamical solution was used to control the velocity of the arm, instead of controlling the spatial coordinates, as in [1].

By the same time, the work proposed in [3] claimed that the mechanism of temporal stabilization proposed in [1] was not able to preserve the timing constraint. They proposed to use the dynamical solution to control the velocity of the robot, rather than controlling the spatial coordinates. In addition, its proposal was able to reach a moving target while circumnavigating obstacles, as well as to deal with other disturbances. Theoretically, they stated that the time allocated for the movement was strictly preserved by an adaptive rule. Mobile robots performing simple travel missions demonstrated the success of this approach.

Later, the challenge of controlling the velocity of a mobile robot suffered additional adaptations. The work in [13] proposed to explore the bifurcation theory to switch the qualitative dynamics of the oscillators. This allowed using a single dynamical system to generate rhythmic and discrete movement, instead of switching between different dynamical systems. Furthermore, based on the current state of the oscillators, a new adaptive mechanism for frequency modulation of the velocity profile was proposed. This allowed specifying the time intervals for the robot acceleration and deceleration. The new velocity profile takes into account the robot physical perspectives by requesting lower maximum velocities.

The work in [85] used the same dynamical system proposed in [13]. However, the sensory loop was closed by acquiring the target location through an onboard camera. This improvement showed that the dynamical system responsible for generating the timed movement was robust to the noise introduced by the vision information. As the localization of the target is acquired by the robot's vision system, there might be situations in which the target is not detected by the camera (*e.g.* something is occluding the ball or the robot needs to change its direction). The estimation of the target location is a major problem addressed in [151, 152]. An Extended Kalman Filter (EKF) is applied to estimate the target location relative to the robot's current position. Additionally, this work showed through experimental field trials that including an EKF into the architecture does not degrade the generation of timed movements for the robot.

The aforementioned works addressed simple missions. The goal locations could be reached

by the robot in a simple way, despite small perturbations, *e.g.*, obstacles placed in the environment. Moreover, only short distances, in the order of meters, between the goal location and the initial position of the robot were considered. In hospitals or in other large environments, the robot could be summoned to a far location. In such cases, a global path planning is required to endow the robot with knowledge relatively to the best route to follow. Works proposed in [86, 153] included a control module responsible for global path planning. These works demonstrated that the generation of timed movements could be extended to more complex and longer missions.

The aforementioned approaches deal with temporal stabilization by generating rhythmic and discrete solutions for the spatial coordinates of the robot movement or for the robot's velocity. In both cases, the good performance of the dynamical systems approach encourages researchers to continue applying them in other robotic applications. In fact, the generation of timed movements is not exclusive to mobile navigation or to robotic arms, and other robotic frameworks have also addressed timed movements, in order to perform both rhythmic and discrete actions.

Robotic frameworks that include learning by human demonstration [154–158] have extended the Dynamic Movement Primitive (DMP) [159] to achieve rhythmic, discrete and superposing of both primitives (rhythmic and discrete). The goal was to generate synchronized movements for robots with multiple degrees of freedom. These authors claimed that representing a movement with differential equations has the advantage to automatically correct perturbations by the system dynamics.

The success of nonlinear dynamical systems to model mathematically the role of the spinal cord to generate motor patterns through CPGs networks, have led researchers to apply them in robotic locomotion, including quadruped [160–163], biped [164–166], hexapod [167–169] and swimming robots [170–172].

Performer robots in a wide variety of activities, including drumming [173, 174], rehabilitation [175], modular robotics [176–178], ball paddling [179, 180], reaching objects [181, 182] have also used dynamical systems to generate movement.

2.2.4 Localization

Robot localization is a key problem on the development of autonomous mobile robots, since the robot should know its location to decide its next movement. The localization problem can be divided into three different subproblems, namely position tracking, global positioning and kidnapped problem [183, 184].

The position tracking problem consists on keeping track the robot's pose from a known initial pose. The main difficulty of this problem is to compensate the unbounded errors in the robot's odometry. Algorithms for position tracking consider assumptions on the maximum error and on the shape of the robot's uncertainty.

The global positioning problem is a more challenging localization problem that consists on determining the robot's pose under global uncertainty. This problem can be divided in two phases. The first phase determines the possible robot's initial poses based on sensory information. In cases where multiple poses are candidates, the second phase eliminates the incorrect candidates. In this phase, the robot should travel the minimum distance necessary to calculate its location, which depends on the size and on the symmetry level of the environment.

In the kidnapped problem [185, 186], the robot may be well localized but suddenly it is moved to another location without be informed about. The robot has to detect the "kidnapping" and calculate its new pose. Solving this problem is very important to verify the ability of the robot to recover from catastrophic localization failures.

These localization problems can be solved in two steps: the first one consists on installing sensors and/or landmarks over the environment; the second one consists on selecting the localization technique that combines noisy sensory information to estimate the robot's pose.

2.2.4.1 Sensors for the localization problem

The localization of the robot can be estimated through proprioceptive and/or exteroceptive sensors. Proprioceptive sensors measure physical variables on the robot. These sensors include encoders, gyroscopes, accelerometers and compasses. On the other hand, exteroceptive sensors measure the relation between the robot and the environment through natural or artificial objects. For short displacements, proprioceptive sensors are accurate. However, for long-term missions, as uncertainty grows exponentially, measurements become inaccurate. In general, exteroceptive sensors provide data with a constant uncertainty and are not available at all instants of time. In mobile robotics, a common strategy is to use proprioceptive sensors to estimate the robot's pose during short displacements, and exteroceptive sensors to correct the measurements provided by the proprioceptive sensors. Exteroceptive sensors include cameras, laser range finders, radars, infrared, Wireless Local Area Network (WLAN), Ultra Wide Band (UWB), pseudolites, Bluetooth and Radio Frequency Identification (RFID). The survey presented in [187] compares different wireless sensors for indoor applications.

Infrared

Infrared is a short-range wireless technology mainly used in combination with other technologies for robot localization. The EIRIS system [188] combines infrared and ultra-high frequency to obtain approximately 1 m of accuracy. A localization uncertainty of approximately 2 m with a combination of infrared and Bluetooth was obtained in [189].

RFID

RFID have limited ranges, typically approximately 12 m, and the cost of the readers is relatively high. This technology has particular problems, like interferences and reflections of the signal, missing tag range and a high number of false negative readings. Metal objects absorb the energy sent by the RFID reader and, consequently, tags attached to metallic

objects will only be detected in a short range. RFID readings are generally affected by high uncertainty and, in the case of passive tags, the RFID reader can only determine whether a tag is present within the reading range. RFIDs do not require direct line-of-sight and multiple tags can be detected simultaneously. Best location results using RFID achieve errors in the order of 0.5 m [190–192].

UWB

Ultra wide band sends short duration pulses that passes easily through walls, equipment or clothing. However, metallic and liquid materials cause UWB signal interference, which can be minimized by placing more UWB readers. An indoor localization accuracy about 0.2-0.3 m was obtained in [193, 194].

WLAN

WLAN is becoming the standard technology for indoor wireless communication because many buildings are already equipped with WLAN infrastructures. Wi-Fi devices are distributed along the environment to act as beacons and the signal strength is used to localize the receiver sensor placed on the mobile robot. However, the WLAN signal is absorbed by water and corrupted by random effects, such as noisy signals, which decreases the signal strength. Several works have obtained an accuracy in the order of 1-10 m [195–197].

Bluetooth

Bluetooth is a short-range wireless technology whose accuracy for robot localization is subject to the influence of obstacles in the environment. This technology is not suitable for large distances between the reader and the receiver devices. Work [198] showed that localization with Bluetooth provides an accuracy of approximately 1 m.

Pseudolite

Pseudolite is a technology that overcomes the limitations of the conventional Global Positioning System (GPS) in indoor environments. A pseudolite system consists of several transmitters that act as pseudo-satellites. A minimum of four pseudolite transmitters should be placed in known positions, in order to obtain unambiguous estimates of the robot localization. Pseudolite suffers from multipath problems, affecting the accuracy of the robot's localization. Approaches using pseudolites for localization [199–201] obtained uncertainty errors less than 0.4 m.

RGB and infrared Cameras

The main goal of a vision localization system is to estimate the robot's pose based on natural or artificial landmarks. In general, vision techniques for robot localization are combined with estimator algorithms, in order to improve the accuracy and provide estimates even when the vision signal is lost. Several works have adopted vision for robot localization in dynamic environments [202–204]. Surveys for robot localization based on vision sensors can be consulted in [205, 206]. The localization system Stargazer [207] and approach [208] present uncertainty errors in indoor environments under optimal conditions close to approximately 0.1 m.

Laser range finder

2D laser range finders provide a scan of the surrounding environment that can be compared against an *a priori* map. Several works have reported that accurate laser scanners can obtain localization errors between 0.05 m and 0.25 m [186, 209–211].

Sensors accuracy comparison

In hospital environments, delivery mobile robots should perform the missions with a maximum error of approximately 0.4 m (see section 5.1 for further details). In table 2.2, it is shown the error of the aforementioned sensors used for indoor localization. Infrared sensors can localize the robot with an error of approximately 1 m, which is higher than the maximum error allowed for localization in hospital environments. RFIDs present a lower error than infrared technology. Nevertheless, this error level is not sufficiently low to be considered suitable for hospital environments. Furthermore, hospitals are non-controlled environments and plenty of interferences will deteriorate the RFID signals. Both WLAN and bluetooth are not feasible for robot localization in hospitals. Pseudolites and UWB can be suitable for mobile robots in hospitals. However, UWB and pseudolite readers are very expensive, which may be economically unviable for large environments. Even though the cameras localization errors were achieved in controlled environments, vision is a favorable solution for robot localization in hospitals. Laser range finders have a well-suited range of localization errors and can be adopted as a solution for hospital environments.

TABLE 2.2: Expected error according to the exteroceptive sensor used for robot indoor localization.

Exteroceptive sensors	Expected error (m)
Infrared	1
RFID	0.5
UWB	0.2-0.3
WLAN	1-10
Bluetooth	1
Pseudolite	0.4
RGB and infrared cameras	0.1
Laser	0.05-0.25

2.2.4.2 Techniques

Several techniques have been developed over the last years to solve the localization problem (see a survey [212]). One of these techniques is the so-called dead-reckoning. This technique uses simple geometric equations, based on odometry, to compute the position of the robot relative to its initial position. However, dead-reckoning is unable to provide accurate estimates for long distances.

Other algorithms solve the localization problem by combining relative and absolute position measurements. The fusion of information from multiple sensors can provide more

accurate estimates of the robot's pose. In general, approaches for data fusion when applied to robots are probabilistic approaches [183], such as the Kalman filters family [213], probabilistic grids and sequential Monte Carlo techniques.

The Kalman filter is a recursive estimator that provides a continuous state that evolves over time, based on non-continuous observations of the state. Kalman filters are well suited to deal with multi-sensor estimates. The use of statistical measures of uncertainty makes it possible to evaluate the role that each sensor plays in the overall system performance. Nevertheless, Kalman filters require that the starting position of the robot is known *a priori*. This limitation makes Kalman filters unfeasible for global localization problems [186]. Algorithms as multi-hypothesis Kalman filters, Grid-based Markov or Monte Carlo filters are able to overcome this limitation.

Multi-hypothesis Kalman filters represent beliefs through a mixture of Gaussians, which allows representing multiple and distinct hypothesis of the robot's pose, each one represented by a single Gaussian [214]. However, the assumption that noise is modeled by Gaussian functions remains. Grid-based Markov localization uses a probabilistic framework to maintain a position probability density over the set of possible robot poses. This algorithm deals with multi-modal and non-Gaussian densities, making it able to handle ambiguous situations, such as the re-localization of the robot in case of failures. Markov localization assumes that the environment does not change over time, which can lead it to fail in dynamic environments.

Monte Carlo filters describe probability distributions as a set of weighted samples of an underlying state space. These techniques are well suited to problems where state-transition models and observation models are highly nonlinear, since they can represent general probability densities. In addition, multimodal or multiple hypothesis density functions are well handled. In general, the number of samples required to accurately model a given density increases exponentially with state-space dimension. However, the effect of dimensionality can be reduced by re-sampling samples, as particle filters do [215].

2.3 Robotic Architectures Paradigms

Robotic systems are based on a predefined architectural structure that specifies how the architecture is divided into subsystems, and how they interact to each other. A well-conceived architecture offers significant advantages in the specification, execution and validation of the robotic system.

Robotic systems should fulfill important requirements, as the need to interact asynchronously, in real-time and in dynamic environments. These requirements are satisfied by including capabilities into the robotic systems, such as, simultaneous support for actuators and sensors and integration of high-level planning with low-level control. These capabilities can be implemented using different architectural structures, each one with their inherent benefits.

Another important feature of robotic architectures is modularity. This consists on the ability to decompose a complex system into a group of simple subsystems. Decomposition of complex architectures provides multiple architectural styles that support the particular needs of different applications. The decomposition can be hierarchical, in which subsystems are built on top of other subsystems. Temporal dimension decomposes architectures according to the operation frequency. Architectures decomposed by abstraction have subsystems at one layer invoked by other layers. In mobile navigation, abstraction decomposition may divide the architecture into local and global navigation.

One of the first known paradigms used to decompose robotic architectures is the Sense–Plan-Act (SPA) [216]. The architecture is decomposed into three functional groups: sensing, planning and executing. The sensing group is responsible for translating environmental information into an internal model. The planning group generates the sequence of actions sent to the robot by the executing group. The main advantage of this paradigm is that the plan is executed without using the sensory information that created the model. However, the execution of a plan without considering sensory information is critical in dynamic worlds. Herein, new robotic paradigms include a reactive planning, in which sensory information is considered on the mission planning.

Subsumption paradigm proposed by [217] considered that the architecture is built from layers of interacting finite-state machines, called behaviors. This paradigm allows robots to constantly sense the world. Several robotic behavior architectures were built based on the subsumption paradigm [218, 219].

The behavior-based architecture [220] was built based on biological inspiration, allowing to connect motor and perceptual schemas. Due to the lack on planning capabilities, this paradigm presents several difficulties to generate behavior for long-range goals. Robots need the planning capabilities of early architectures, as well as the reactivity of the behavior-based architectures. This leads to the development of a new architecture paradigm: layered robot control.

One of the first architectures integrating planning, reactive and sequencing modules was the Reactive Action Packages (RAPs) developed by [221]. This architecture was the first one to outline a three-layered (3T) architecture. Several 3T paradigms based architectures have been developed since then, such as, MITRE [222], ATLANTIS [223], LAAS [224], CLARAty [225]. Later, the 3T paradigm was extended to multi-robot coordination in the Syndicate architecture [226]. Each layer was not restricted to interact only with the layers located above and below, but also with layers at the same level. By this way, distributed control loops could be designed at multiple levels of abstraction.

Hospital delivery robots need the planning and reactive capabilities that the 3T paradigm offers. In fact, this paradigm has been widely used to design the control architectures for current autonomous mobile robots.

2.4 Stability Analysis of Nonlinear Systems

Robotic architecture paradigms conceive well-organized architectures that control robots such that they are able to complete successfully its missions. However, some issues relative to the behavior of the robots can not be answered by these paradigms. For instance, what happens with the system during long periods? Is the architecture able to deal with different initial conditions and bounded perturbations along the mission? Answering these issues is relevant when dealing with mobile robots in unpredictable and dynamic environments.

Stability theory provides tools that can be used to address these issues. However, despite the great importance of stability theory for control architectures, no general method has been adopted to the design of stable nonlinear controllers in robot navigation. Considerable efforts have been made to develop a wide range of alternative theoretical methods in the stability theory, each one more suitable to be applied to particular nonlinear control problems.

In linear control, a system can be analyzed by standard techniques in frequency or in time domain. However, these standard techniques are useless for nonlinear control, since direct solutions of nonlinear systems are in general impossible to determine and transformations in the frequency domain are not applied. In the following sections, some relevant methods for stability analysis of nonlinear systems are described.

2.4.1 Phase Plane Analysis

The basic idea of phase plane analysis is to solve graphically second-order nonlinear systems through a phase portrait, rather than seeking for an analytical solution. The phase plane analysis provides a set of system motion trajectories on a two-dimensional plane. This allows visualizing the behavior of a nonlinear system starting from different initial conditions, without having to calculate analytically the solution of the nonlinear system. The qualitative features of the motion trajectories are sufficient to provide intuitive insights about the stability and other motion patterns existent in the system.

The analysis of the fixed points (equilibrium points) is very important in the phase plane, as they can reveal the behavior of the system. Linear systems only have one fixed point, and their stability is characterized by the nature of the respective fixed point. On the other hand, the more complicated patterns of nonlinear systems, such as limit cycles, chaotic attractors or multiple fixed points can be seen in the phase plane.

Some methods for constructing the phase portrait for linear or nonlinear systems include: Isocline, delta method, Pell's method, Lienard's method and the Vector field diagram.

Phase plane analysis is suitable to second-order dynamics, since their motion trajectories are well represented by carves in a plane. This restrains the generality of the analysis to simple nonlinear systems based architectures. Extensions to third-order systems can be achieved through computer graphics. However, the graphical study of higher-order systems is geometrically and computationally complex.

Phase plane analysis has been adopted for designing stable controllers in robotics. The work proposed in [227] discussed how the phase plane analysis describes the overall behavior of single and multiple autonomous robotic vehicles with finite state machine rules. A controller for mobile navigation based on the dynamical system theory demonstrated its stability through the phase plane analysis [82].

Appraches [228, 229] proved the stability in the phase plane of a limit cycle that controls the periodic motion of a mechanical biped robot. Additionally, other biped [230–232], quadruped [233], robotic hands [234], hopping robots [235, 236] and snake-like robots [237, 238] used phase plane analysis to design and model their controllers.

The applicability of the phase plane stability analysis has been extended to biologic problems, as the study of the rat exploratory behavior [239] and to mathematics, as the study of the Kuramoto-Sivashinsky equations [240].

2.4.2 Lyapunov Theory

Lyapunov theory is an approach widely used to study the stability of nonlinear systems [241]. Lyapunov theory includes two methods for stability analysis, namely, direct and indirect methods.

The indirect method proves that the stability of a nonlinear system can be inferred by analyzing the stability in the close vicinity of its fixed points. This method uses the linearization of the nonlinear system in the neighborhood of its fixed points to obtain the stability of the nonlinear system. However, Lyapunov's indirect method does not provide information about how the system behaves when the initial condition is not in the neighborhood of the fixed point.

Due to its generality, Lyapunov's direct method has been widely used to design nonlinear controllers in robotics and adaptive control. The basic idea of this method is to define an overall energy function (called Lyapunov function) as the sum of the energy functions associated to all controllers in the system. Then, a control law that makes this energy function to decrease is selected. This idea is based on the generalization of the energy concepts associated to a mechanical system, in which its motion is stable if its total mechanical energy decreases during all the time instants. Similarly, if the total energy of the system is continuously dissipated, eventually, the system converges to an equilibrium point.

Lyapunov's direct method can be applied to all types of control systems, whether linear or nonlinear, autonomous (time-invariant) or non-autonomous (time-varying), finite or infinite dimensional, continuous or discrete, in small or large motion. Furthermore, it is useful in many engineering problems where stability in the sense of Lyapunov (indirect method) is not enough. These problems require that, when disturbed, the state does not remain close to its solution, but rather converges gradually to it [8].

Lyapunov theory works equally well in showing the boundedness of solutions when no equilibrium points exist [242]. This constitutes an advantage over phase plane stability analysis.

Lyapunov theory has been successful adopted to guarantee stability of switched and hybrid controllers with hard nonlinearities [243]. Several researchers have adopted Lyapunov theory to ensure stability for their mobile navigation controllers [244–248], including both global and local stability [249]. Other robotic frameworks applying Lyapunov theory include manipulators [250], formation of multi-mobile robots [251], lawn mower robot [252], and the stability of dynamical systems to reproduce human arm movements [159].

Other works addressed the convergence of tracking errors [253, 254], and the stability of neural networks [255]. The improved dynamic window approach (DWA) in [256] incorporated Lyapunov stability criteria to guarantee global and asymptotic convergence.

The main limitation of the Lyapunov's direct method lies in the fact that there is no generic way of finding Lyapunov functions for nonlinear systems. Some methods as Krasovskii's method or the variable gradient method can be used to facilitate the search for Lyapunov functions [8]. Control designers use these methods, experience, intuition, trial and error, and physical insights to search for a suitable Lyapunov function. Furthermore, Lyapunov theory does not provide freedom to the architecture design. If a new module is included into the architecture, it is not guaranteed that the same Lyapunov function works. Even though several architectures have been proven to be stable based on Lyapunov theory, these architectures only addressed local navigation purposes. Global navigation and generation of timed movements were not considered for the stability analysis.

2.4.3 Input-Output Stability

Lyapunov stability and phase plane analysis study stability through the mathematical modeling of the dynamical systems. Input-output stability is an alternative approach to the mathematical modeling of dynamical systems based on input-output notions. This approach analyzes the stability of a given system by relating the output directly to the input of the system, without knowing the internal state of the system. The system is treated as a black box, only accessed through its inputs and output terminals. A system is stable in the input-output sense if small input signals produce small output signals.

Input - output stability analysis was previously developed for linear feedback systems. The most notable contributions to generalize this analysis to nonlinear systems were made by [257, 258]. Small-gain and passivity are important theorems that provide conditions for the preservation of input-output stability in a feedback configuration.

Concepts from input-output stability and Lyapunov theory were combined to derive the input-to-state stability [259], in which inputs and states are considered for the stability analysis.

Input-output stability is useful for studying the stability of interconnected systems, since the gain of a system can be used to track how the norm of a signal increases or decreases as it passes through the system. Input-output stability provides a method to anticipate the qualitative behavior of a feedback nonlinear system with scarce information about the feedback components. This leads to notions of robustness of feedback stability and motivated many of the recent developments in modern control theory.

This type of stability is mainly used for network and time-varying delayed systems [260]. Applications of the input-output stability for mobile navigation controllers are scarce. In [249] input-output stability was used to assure stability of a controller for local navigation. In [261], a smooth nonlinear feedback was obtained to achieve asymptotical input-output stability. This stability method was also addressed to design a stable tracking controller [262].

The main disadvantage of input-output stability approach is the fact that the system description is unable to deal with physical system interconnections. Hence, the mathematical modeling based on state space models is of fundamental importance in the description of physical dynamical systems.

2.4.4 Contraction Mapping Theory

Searching for a contraction metric is an alternative to phase plane analysis, Lyapunov and input-output stability theories. While phase plane analysis must be constructed over a known equilibrium point, Lyapunov analysis need a Lyapunov function and input-output analysis is unable to deal with physical system interconnections, Contraction Mapping Theory implies indirectly the existence of a single stable equilibrium. This is particularly useful if the equilibrium point changes from locations depending on the unknown dynamics.

Contraction Theory is a useful method that provides a set of tools to analyze the stability of both linear and nonlinear systems. Contraction analysis is inspired by fluid mechanics, being different from the classical point of view of stability [11, 263, 264]. Using this analysis, stability is not viewed relative to some equilibrium point. Alternatively, a system is stable in some region if the initial conditions or temporary bounded disturbances are forgotten, *i.e.*, the solution of the system converges to the unique equilibrium point independently of the initial conditions. Contraction is an incremental form of stability, *i.e.*, stability of the system trajectories with respect to one another [265], besides being attracted towards an equilibrium position. Incremental stability concept is useful to analyze the robustness and performance properties of nonlinear closed-loop systems, and to verify if the system holds suitable steady-state properties [266, 267]

As an alternative to Lyapunov theory, Contraction Theory establishes exponential stability of systems [11] without having to select a Lyapunov function. Additionally, it explicitly incorporates the control input in the process of stability analysis [268], which is similar to the input-to-state stability. Systems that are universally contracting with respect to the inputs

are stable with respect to the input-to-state stability [268]. Contraction Theory studies stability when external bounded perturbations do not put the system out of contraction bounds. This implies these perturbations have no effect on the global exponential convergence, which is an important advantage when dealing with real-time applications. These perturbations might represent noise and uncertainty in the system control.

The combination property [269] inherent to Contraction Theory provides freedom to the architecture design, as the overall architecture remains stable if a new module included into the architecture is contracting. The analysis of the remaining modules does not change, as in the case of Lyapunov theory, for which a new Lyapunov function may be necessary. Therefore, contraction is preserved through a large variety of systems combinations (*e.g.*, parallel, feedback or hierarchies), which makes this theory useful for analyzing the stability of large complex real-time control architectures. Contraction Theory is a powerful tool to study synchronization and the dynamic behavior of coupled nonlinear systems and oscillators. Furthermore, contraction analysis is extended to autonomous and non-autonomous systems.

Contraction Theory has been widely used to derive stable controllers for several robotic frameworks, including robotic flight control [270, 271], underwater vehicles [11], flexible robot arm [272], tracking [273], indoor mobile navigation [274] and dynamical movement primitives for locomotion [275]. However, global navigation was not considered on the stability analysis. Later, the approach addressed in [153] included the global navigation into the architecture presented in [274] and successfully demonstrated the stability of the architecture in simulated missions. Even though the referred works have used Contraction Theory to prove the stability of control architectures, they lack of robustness when the robot and the environment include external bounded perturbations into the global system.

Contraction Theory has also been applied to the analytical development of observers for problems such as inertial navigation [276] and dynamic positioning of surface vessel [277], to the nonlinear stability of complex dynamic network systems [278, 279], and to verify the exponential convergence of the EKF [263].

2.5 Problem Statement

This section summarizes the main limitations of the literature previously described.

- Timing constraints: even though several current commercial delivery mobile robots are already navigating in hospitals, none of them includes timing constraints as requirements for their delivering missions. This limitation is also verified in other aforementioned commercial robots. Current works including timing control into a navigation architecture based on nonlinear dynamical systems with local and global planners for mobile robots were developed within the scope of this thesis.

- Navigation control: delivery mobile robots typically stop in cases where obstacles are detected in a safety zone, waiting for the clearance of the path. The robot will be delayed if the mission must be completed within a time constraint. With the architecture proposed in this thesis, the robot must slow down when obstacles are detected and try to circumnavigate them. The robot only stops when obstacles are detected in a close neighborhood to it.

- Stability conditions: despite the extensive research on stability theory, there are no works focused on formal stability analysis of complex navigation architectures with time constraints, when controlling robots in real environments.

- Duration and complexity of the experiments: there is a lack of formal results demonstrating the stability and performance of robots in indoor environments. This is special important for long-term missions. Dependability tests should validate the robustness of the architecture in real field experiments.

Chapter 3

Overall Architecture

This chapter starts with a review of each block of the global system, namely, the architecture, the robot and the environment. The following sections explore the several modules that compose the architecture.

Section 3.2 details the Motion Control module. This section starts with a description of the module responsible for representing the environment. Later, the process to obtain the trajectory that leads the robot to its goal location and how this trajectory is converted into a direction that the robot must follow is described.

Section 3.3 describes the Local Control module. The section details the target orientation and obstacle avoidance contributions. Furthermore, it describes how the robot detects a zone with obstacles and what should be the contributions of the target orientation and obstacle avoidance behaviors.

This chapter is concluded with an overview of the Timing Control module, including its description and parameterization, and the calculation of the distance that the robot needs to reach the goal location. Furthermore, it is explained the module responsible for specifying the robot's motor behavior and the adiabatic elimination.

3.1 Global System Overview

Fig. 3.1 shows the global system: f_{robot} stands for the robot, $f_{\text{supervisor}}$ stands for the architecture that controls the robot, and block K represents the interacting environment.

Maps $f_{\text{supervisor}}$ and f_{robot} are assumed to be described by C^1 discrete dynamical systems[1] of the form,

$$x_{k+1} = f_{\text{supervisor}}(x_k, P_{\text{g}}, e_{k+1}), \tag{3.1}$$

$$q_{k+1} = f_{\text{robot}}(q_k, x_{k+1}), \tag{3.2}$$

$$e_{k+1} = K(e_k, q_{k+1} \oplus \eta_{k+1}), \tag{3.3}$$

[1]Current robotic systems are computer-based and hence intrinsically discrete.

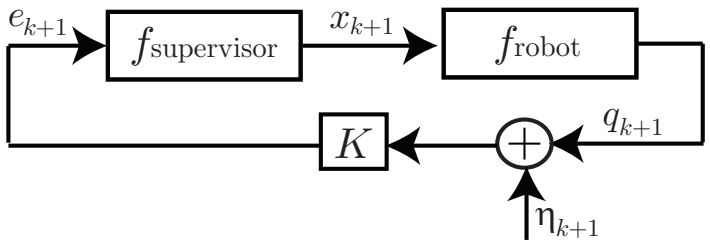


FIGURE 3.1: Graphical representation of the global system. The robotic application can be viewed in general terms as the robot f_{robot}, the architecture $f_{\text{supervisor}}$ and the interacting environment K.

where e_{k+1}, q_{k+1} and η_{k+1} stand for the observations, interactions and disturbances, respectively. $\oplus$ represents the combination of signals q_{k+1} and η_{k+1}. P_{g} stands for the goal location where the mission ends, and x_{k+1} represents commands sent by $f_{\text{supervisor}}$ that trigger actions on the robot. The C^1 assumption is reasonable since in practical terms situations, C^0 or C^{-1} maps can be replaced by C^1 maps using splines (see for instance [280] or for other regularization methods [281, 282]).

Hereafter, the feed-through maps and dynamical systems that compose the global system are represented through their discrete form, and k is the discretization index. The discretization method is the forward Euler method.

A detailed schematic of the global system is shown in fig. 3.2. The architecture (represented by $f_{\text{supervisor}}$) is responsible for guiding the robot along the missions. Three modules of control compose the architecture, namely, Motion (green area), Local (blue area) and Timing (red area). Each module of control works independently of the others, however information flows among them.

The architecture outputs the angular, ϖ, and linear velocity, v, to the robot. Both velocities safely guide the robot towards its goal location. The Motion Control calculates the trajectory that connects the current robot's position, $\hat{P}_{\text{r}}$, to the goal location, P_{g}. This trajectory is used to obtain the direction ψ_{tar} that the robot should follow to reach P_{g}. The Local Control guides the robot by setting its angular velocity, ϖ. It receives the direction that the robot should follow, ψ_{tar}, and verifies based on sensory information, $\psi_{\text{obs},i}$, if the direction ψ_{tar} can be followed by the robot without colliding with obstacles. Timing Control is in charge of setting the robot's linear velocity, v. This module receives the remaining distance to the goal location (encoded by A), the time constraint, MT, and sensory information (laser, La_i and sonars, S_i).

The Localization System is responsible for estimating the robot's pose, $\hat{P}_{\text{r}}(\hat{x}_{\text{r}}, \hat{y}_{\text{r}}, \hat{\phi})$ through an Extended Kalman Filter (EKF) that merges the sensory information provided by odometry and vision information [2]. The External module represents the user that interacts with

[2]Notation $\hat{}$ stands for estimates provided by the EKF.

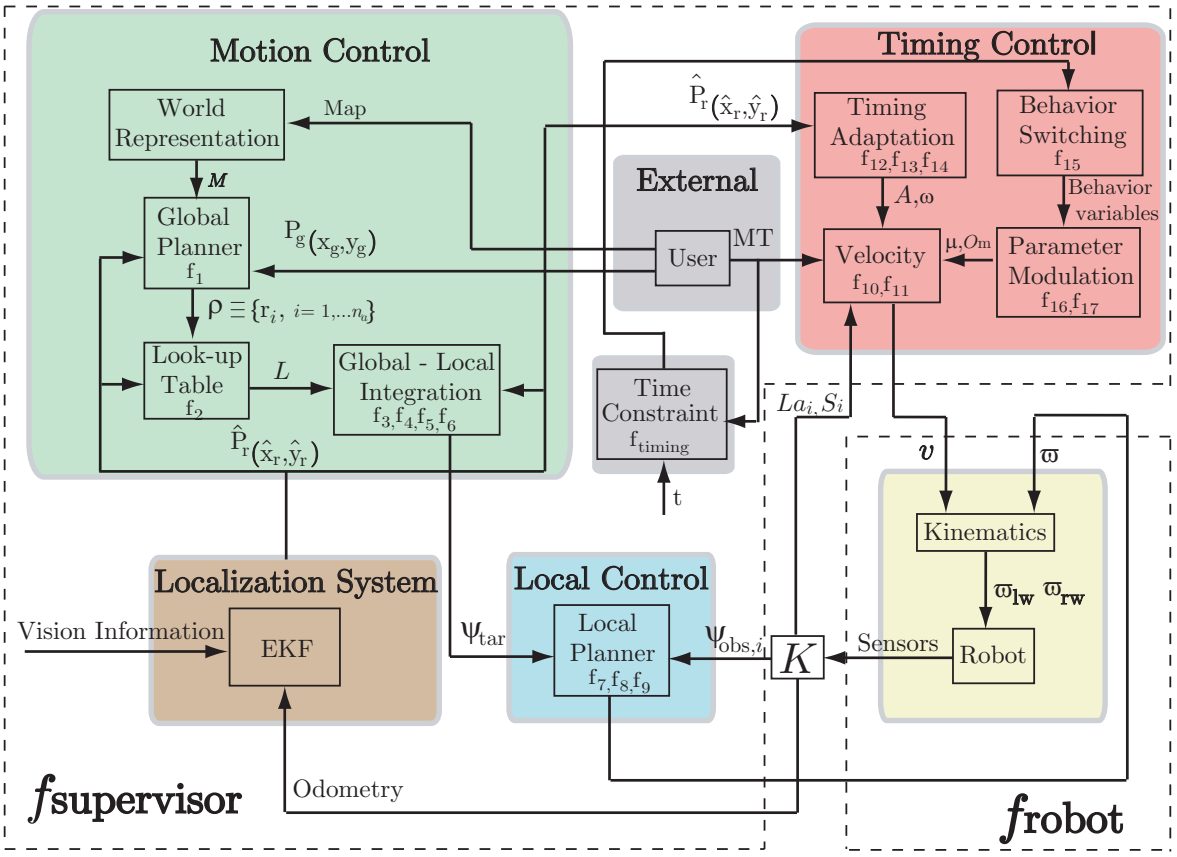


FIGURE 3.2: Schematic of the K, f_{robot} and $f_{\text{supervisor}}$ blocks containing the modules of control Motion, Local and Timing.

the robot. The user sends the time constraint, MT, and the goal location, P_{g}, to the architecture. In a practical perspective, this represents the time to complete the mission and the location on the environment for which the robot is requested to move. The user also provides the map of the environment before the robot initializes its first mission.

f_{robot} is composed by the physical robot and the kinematics module. The kinematics are feed-through maps that convert the linear velocity, v, and angular velocity, ϖ into the angular velocities for left, ϖ_{lw}, and right, ϖ_{rw}, wheels. The kinematics block depends on the robot used to carry out the missions.

Block K represents the environment that interacts with the robot. It maps the f_{robot} outputs into the $f_{\text{supervisor}}$ inputs.

3.2 Dynamic Approach to Motion Control

The Motion Control is responsible for calculating the trajectory that the robot should follow to complete its mission. This module provides the goal direction ψ_{tar} that the robot should track to follow the obtained trajectory. The map of the environment, *e.g.*, corridors, rooms and distance between zones is considered to calculate the trajectory, such that the expected

time spent by the robot to cover the trajectory is minimized. Real-time sensory information acquired by the laser and sonars is not considered. The Motion Control (see schematic in fig. 3.3) is composed by four blocks: World Representation, Global Planner, Look-up Table and Global - Local Integration, which are detailed in the following.

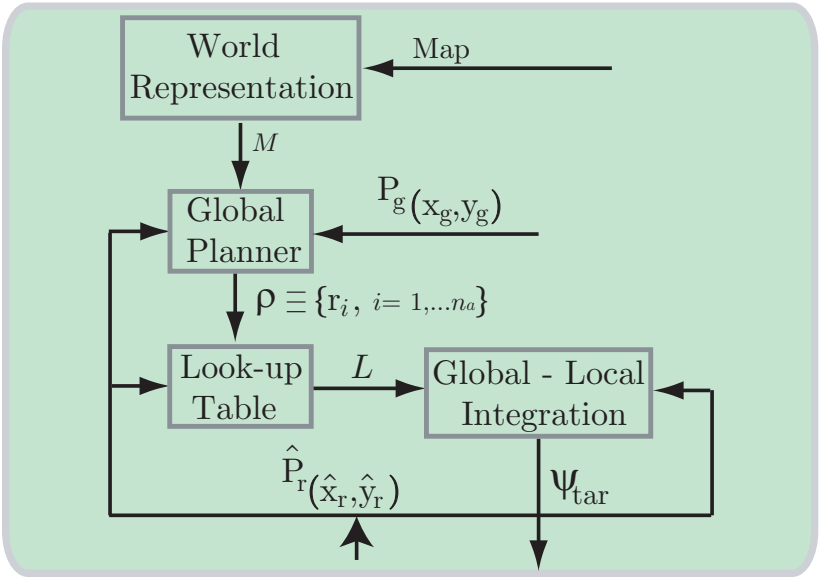


FIGURE 3.3: Schematic of the Motion Control module. Inputs of this module are the goal location, P_g, the robot's location, $\hat{P}_r$, and the map of the environment. The output is ψ_{tar}.

The Motion Control module should deal with sensory and state uncertainty, τ. State uncertainty stands for the unmodeled external influences on the motion of the robot, and sensory uncertainty includes partial or noisy measurements of the robot's pose. One typical solution to model the uncertainty is to consider the robot's pose as the space of all beliefs. Representing the robot's pose as a space of beliefs was already addressed in the coverage problem [283, 284], and assuming the worst-case model to bound the uncertainty was addressed in [285]. In fact, for navigation purposes, it is sufficient to know that the robot is in some bounded region. In practical terms, the Localization System provides the robot's position $\hat{P}_r$, with a maximum uncertainty τ (see fig. 3.4). The true robot's position is within the neighborhood $B\left(\hat{P}_r, \tau\right)$.

3.2.1 World Representation

The World Representation block is responsible for representing the map of the environment as a mathematical model, M, understandable by the robot. The environment is represented by a topological map, and known by the robot before starting its missions. Section A.1 in appendix A shows an example of the simulated environment illustrated under different

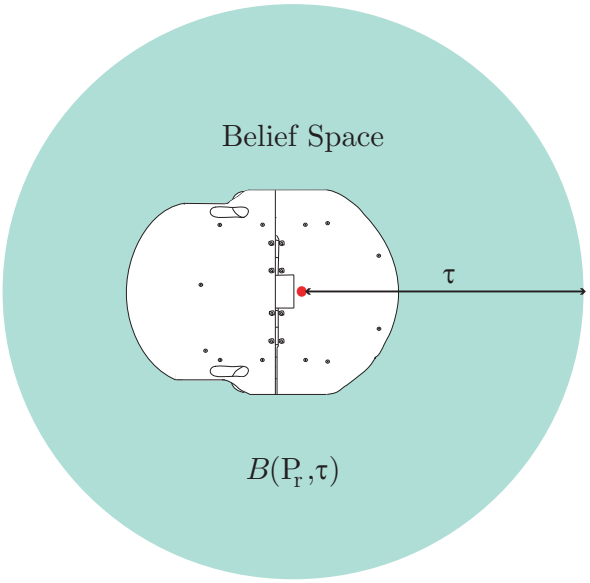


FIGURE 3.4: Space of beliefs for the robot's position, $B(P_r, \tau)$. The true robot's position is within a circular area with radius τ.

representations. The following sections describe how a mathematical module can be used to represent the topological features of an environment.

3.2.2 Transition Function M

The topological representation is obtained through a sequence of algorithms similar to the work described in [126]. The sequence is illustrated in fig. 3.5.

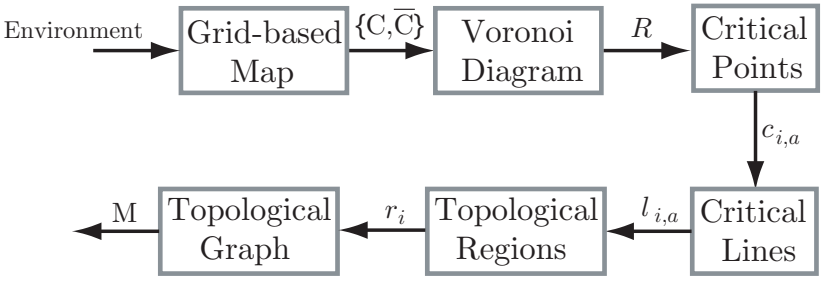


FIGURE 3.5: Sequence of algorithms used to build the transition function, M, which describes topologically the environment.

The first step to obtain a topological representation is to discretize the environment in cells, as in the Occupancy Grid method [97, 286, 287]. Each cell has a binary occupancy value, 0 for clear and 1 for occupied. Clear cells are considered free-space and denoted by C. Occupied cells are denoted by $\bar{C}$.

Based on the set $\{C, \bar{C}\}$, the Voronoi diagram (see [119] for details) calculates a region space R composed by a set of points in the free-space, C, equidistant to at least two obstacles

(see blue line in fig. (3.6)). Nevertheless, any other method to divide the free-space into regions could be used. The Voronoi diagram R contains points, called critical points $c_{i,j}$ (see red empty circles in fig. (3.6)), that identify the center of passages from large areas to narrow areas.

Critical points are connected to their basis points (points that intersect the walls of the environment) through critical lines, l (see red dashed lines in fig. (3.6)). Critical lines divide the free-space C into regions r_i. These regions can be rooms or corridors. The critical line that divides regions r_i and r_j is denoted as $l_{i,j}$. The center of each region is represented by a filled black circle. Black thick lines represent occupied cells $\bar{C}$ and white space represents free cells C.

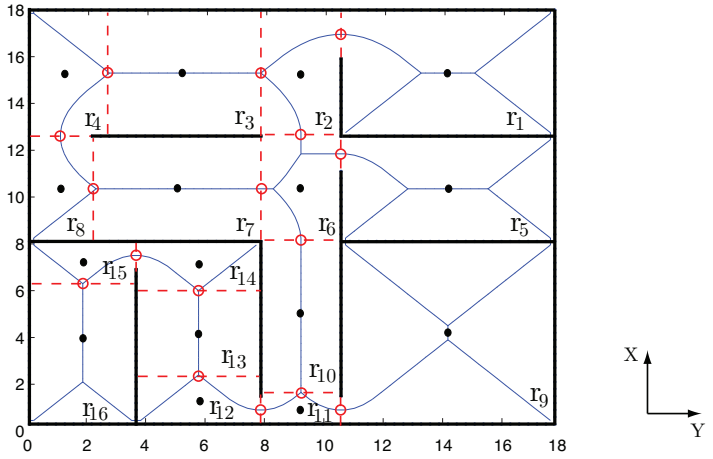


FIGURE 3.6: Topological representation of the simulated environment. Voronoi diagram (blue line) is drawn over a grid-representation. Critical points are depicted by empty red circles, critical lines by red dashed lines and the center of reach region by filled black circles. Regions are numbered from 1 to 16, $\{r_1, ..., r_{16}\}$.

Each duplet of neighbor regions, r_i and r_j, is connected by an arc, named a_{ij}. Each arc has an associated cost C_{ij}. The higher the cost, the greater the expected time to traverse the arc. The transition function, M, receives a duplet of regions $\{r_i, r_{i+1}\}$ and provides the cost to move from region r_i to r_{i+1}.

3.2.3 Total Cost of an Arc

The topological representation allows representing obstacles and unforeseen disturbances that may occur in the environment by dynamically updating the cost of the arcs.

The cost of each arc includes two components: a deterministic cost, d_{ij}, defined *a priori* through the geometrical information included in the map, *e.g.* walls, doors and distance between regions, and an uncertainty cost, u_{ij}, which represents the appearance of unpredictable

obstacles in the environment,

$$C_{ij} = d_{ij} + u_{ij} \ . \tag{3.4}$$

In practical terms, the deterministic cost, d_{ij}, represents the distance that the robot has to cover between regions, and the uncertainty cost, u_{ij}, represents the additional distance that the robot expects to travel when unpredictable obstacles appear.

3.2.3.1 Deterministic Cost

The deterministic cost d_{ij} of an arc connecting regions, r_i and r_j, is obtained by summing the Euclidean distance, $d_{i,c}$, between the center of the region, r_i, and the critical point that divides both regions, $c_{i,j}$, plus the distance, $d_{c,j}$, between $c_{i,j}$ and the center of the region r_j,

$$d_{ij} = d_{i,c} + d_{c,j}. \tag{3.5}$$

Fig. 3.7 shows a representation of the deterministic cost of the arc connecting regions r_1 and r_2, d_{12}, given as follows,

$$d_{12} = d_{1,2} + d_{2,1}, \tag{3.6}$$

where $d_{1,2}$ is the distance between the center of region r_1 and the critical point $c_{1,2}$, and $d_{2,1}$ is the distance between the center of region r_2 and the critical point $c_{1,2}$. [3] The deterministic cost is viewed as the absolute value of the line segments illustrated by the blue lines.

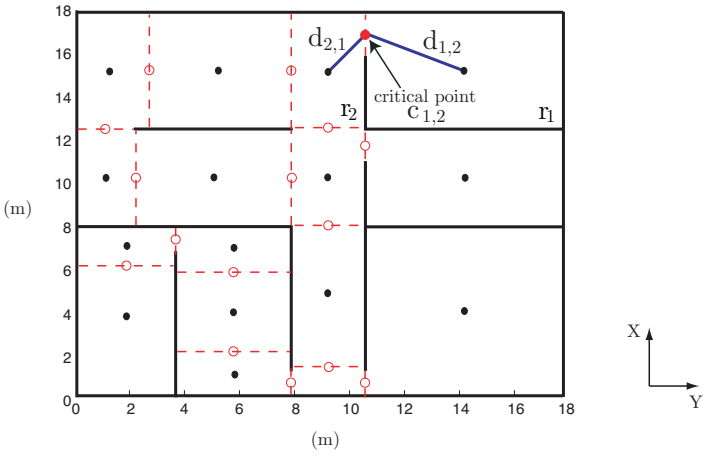


FIGURE 3.7: Deterministic cost between regions r_1 and r_2. Red empty circles represent critical points and black filled circles represent the center of the regions.

[3] Note that d_{12} is equidistant to d_{21}, but the same is not true for $d_{1,2}$ and $d_{2,1}$.

3.2.3.2 Uncertainty Cost

The uncertainty cost u_{ij} of an arc represents a probability of unexpected obstacles to appear in regions r_i and r_j. Initially, if unexpected obstacles do not appear in the environment, $u_{ij} = 0,\ \forall i, j \in \mathbf{R}$. While the robot is moving and acquiring data, the uncertainty cost u_{ij} is being updated if obstacles are detected. If an obstacle is detected in region r_i, the uncertainty cost of all arcs containing this region is updated, $u_{i,a},\ \forall a \neq i \in \mathbf{R}$.

Onboard sensors are used to calculate the number of zones with obstacles. A potential function, $U(\phi)$ (see section 3.3.5 for details), verifies if obstacles are obstructing the trajectory of the robot. If $U(\phi) < 0$, the robot is in a repulsion zone created by obstacles. When $U(\phi) \geq 0$, no obstacles are in the surroundings of the robot. Thus, always that $U(\phi)$ triggers from zero to a negative value, a zone with obstacles is detected and must be considered for the uncertainty cost.

Obstacle Types

In a real environment, two types of obstacles can be identified,

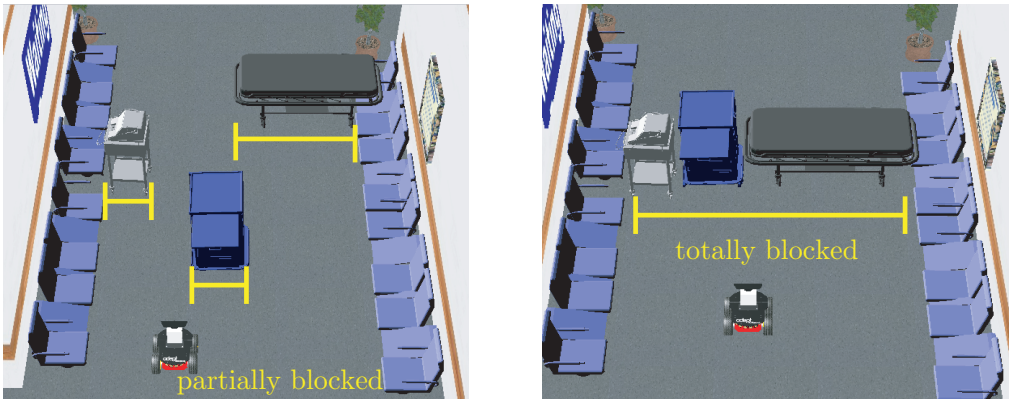
- O_1 - obstacle partially blocking the path.
- O_2 - obstacle totally blocking the path.

When an obstacle partially obstructs the path, the robot can circumnavigate it and follow the same path. However, the robot travels a larger distance than expected to reach the goal location. Consequently, a re-planning of the path is not necessary. If an obstacle blocks a corridor or a passage, the robot needs to find an alternative path to reach its goal location. Fig. 3.8 depicts an example showing the two types of obstacles, O_1 and O_2. Fig. 3.8 (a) depicts the robot in a partial obstructed corridor. The robot has to circumnavigate the obstacles, but it is still able to follow the same path. Fig. 3.8 (b) depicts a case in which obstacles are blocking the corridor, preventing the robot to follow the path. An alternative path leading the robot to its goal location should be planned. The robot must select the path with the minimum expected number of obstacles type O_1 and avoid paths with obstacles type O_2. If no alternative paths are available, the robot must stop and wait for the clearance of the corridor.

The more obstacles the robot faces in a corridor, the greater the distance the robot must travel to traverse the corridor. Thus, the uncertainty cost, u_{ij}, should depend on the number and type of the detected unexpected obstacles,

$$u_{ij} = d_1 v_{ij_1} + d_2 v_{ij_2}, \tag{3.7}$$

where d_1 and d_2 are the average distance needed by the robot to circumnavigate obstacles type O_1 and O_2, respectively. v_{ij_1} and v_{ij_2} indicate the number of obstacles O_1 and O_2 in arc a_{ij}, respectively.



(a) (b)

FIGURE 3.8: a) Obstacles partially blocking the corridor. b) Obstacles totally blocking the corridor.

The distance d_1 required by the robot to circumnavigate obstacles type O_1 was empirically calculated through field experiments. The experiments consist of performing a mission without obstacles, and then repeating the same mission with obstacles type O_1. In both missions, the robot follows a constant velocity of 0.4 m/s. Table 3.1 resumes the obtained results. First column depicts the number of obstacles detected by the robot. Second column refers the expected distance that the robot should traverse and the third column stands for the effective distance traveled by the robot. On average, the robot had to travel approximately more 0.4 m to circumnavigate each obstacle type O_1. Thus, it was defined that $d_1 = 40$.

TABLE 3.1: Distance required to circumnavigate obstacles type O_1.

Number of obstacles type O_1	Expected distance (m)	Traveled distance (m)
1	2	2.36
2	4	4.78
3	5	6.26
10	10	14.10

Obstacles type O_2 force the robot to stop and find a new path to complete its mission. Thus, the robot should not select paths in which an obstacle type O_2 was detected, except when no alternatives paths are available. For that, the uncertainty cost of an arc with an obstacle type O_2 is set as the maximum cost of the arcs in the environment, $d_2 = \max\{C_{ij}\}$. This ensures that an alternative path is selected, rather than the obstructed path.

The number of unexpected obstacles type O_1 and O_2 is given by a probability distribution that defines the mean, v_{ij}, and variance, σ, of their occurrence in any arc of the environment. The adopted probability distributions are similar to the ones presented in [288]. The number of unexpected obstacles type O_1 in a corridor is proportional to its length. On the other hand, the presence of unexpected obstacles type O_2 in a corridor can be viewed as a discrete

event, because only two possible outcomes are available: the presence or the absence of an obstacle type O_2.

The average number of obstacles type O_1 in arc a_{ij}, v_{ij_1}, is defined as,

$$\begin{aligned} v_{ij_1} &= E[O_1] \\ &= \frac{\alpha_1}{\beta_1}, \end{aligned} \tag{3.8}$$

where $E[.]$ denotes the expected value and α_1 gives the number of obstacles detected along the arc a_{ij} during β_1 missions.

The average number of obstacles type O_2 in arc a_{ij}, v_{ij_2}, is defined as,

$$\begin{aligned} v_{ij_2} &= E[O_2] \\ &= \frac{\alpha_2}{\alpha_2 + \beta_2}. \end{aligned} \tag{3.9}$$

where α_2 is the number of detected unexpected obstacles type O_2 and β_2 is the number of missions in which the robot did not detect an obstacle O_2 in arc a_{ij}.

The uncertainty cost, u_{ij} is obtained by substituting (3.8) and (3.9) into (3.7),

$$u_{ij} = d_1 \left(\frac{\alpha_1}{\beta_1} \right) + d_2 \left(\frac{\alpha_2}{\alpha_2 + \beta_2} \right). \tag{3.10}$$

The cost C_{ij} is obtained by substituting (3.10) into (3.4),

$$C_{ij} = d_{ij} + d_1 \left(\frac{\alpha_1}{\beta_1} \right) + d_2 \left(\frac{\alpha_2}{\alpha_2 + \beta_2} \right). \tag{3.11}$$

Uncertainty Cost - Simulation Results

A set of simulated missions is performed to verify how unexpected obstacles modify the path followed by the robot. Fig. 3.9 illustrates the simulations set-up. The robot starts its mission in region r_1 and should reach the goal location P_g (black cross) in region r_8. Two alternative paths are available: path 1, composed by the set of regions $\{r_1, r_2, r_3, r_4, r_8\}$ (red dashed line), and path 2 composed by the set of regions $\{r_1, r_2, r_6, r_7, r_8\}$ (green dashed line) (fig. 3.9 (a)). Before the first mission, there is no knowledge about the occurrence of obstacles type O_1 and O_2. Thus, parameters are initialized as: $\alpha_1 = 0, \alpha_2 = 0, \beta_2 = 0, \beta_1 = 0$. Furthermore, for the purposes of this simulation, it is assumed that the total cost of path 1 and path 2 is 1993 and 1984, respectively. Consequently, the robot follows path 2 to complete its mission (fig. 3.9 (b)).

In the first mission, the robot follows path 2 and detects an unexpected obstacle in region r_7 (see fig. 3.9 (b)). According to (3.10), the uncertainty cost is updated with $d_1 = 40$, since the robot only detected one obstacle type O_1. The number of performed missions and the number of detected obstacles are updated to $\beta_1 = 1$ and $\alpha_1 = 1$, respectively. The cost of the arcs containing region r_7 ($a_{6,7}$ and $a_{7,8}$) are updated. Thus, $v_{67_1} = 1$ and $v_{78_1} = 1$. The

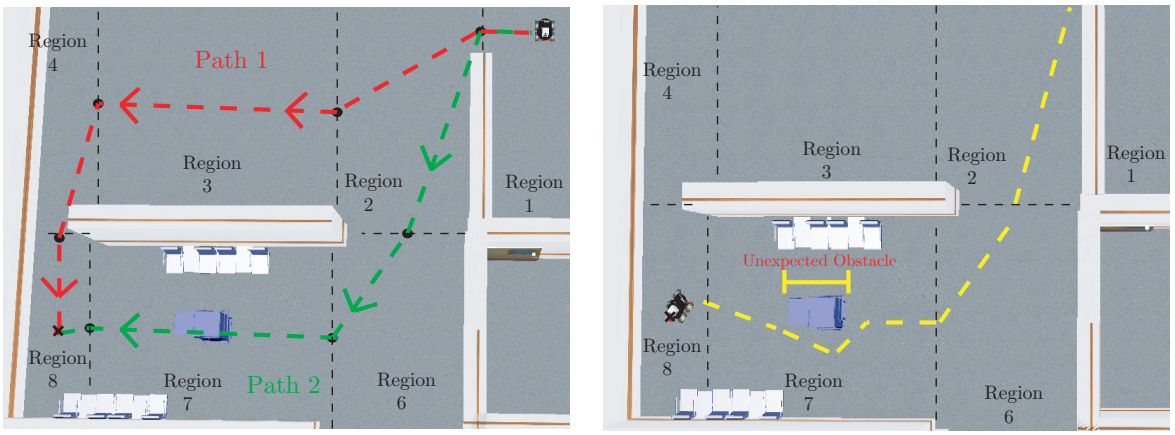


FIGURE 3.9: a) Two alternative paths connecting region r_1 to region r_8. b) The robot selects the lowest cost path (path 2).

total cost of path 2 after the first mission is updated to 2064, which is higher than the cost of path 1. Consequently, for the next missions, the robot will select path 1 to travel from region r_1 to region r_8. Table 3.2 depicts the evolution of the total cost of path 1 and path 2.

Parameter β_i increases as the robot performs missions, and α_1 remains unchanged since no obstacles were detected. Consequently, the cost of path 2 decreases and after the 9^{th} mission, it is lower than the cost of path 1. After this mission, the robot will follow path 2 to move from region r_1 to region r_8.

TABLE 3.2: Total cost of path 1 and path 2 during 9 traversals.

Path	$\beta_1 = 0$	$\beta_1 = 1$	$\beta_1 = 2$	$\beta_1 = 3$	$\beta_1 = 4$	$\beta_1 = 5$	$\beta_1 = 6$	$\beta_1 = 7$	$\beta_1 = 8$	$\beta_1 = 9$
1	1993	1993	1993	1993	1993	1993	1993	1993	1993	1993
2	1984	2064	2024	2010.6	2004	2000	1997.4	1995.4	1994	1992.8

The next simulation verifies the effect of the detection of an obstacle type O_2 on the uncertainty cost. The same initial conditions of the previous simulation are repeated. Parameters α_2 and β_2 defining the number of detected obstacles type O_2 and the number of missions in which no obstacles type O_2 are detected, respectively, are set to 0. Once again, the robot must travel from region r_1 to region r_8 through path 2. Fig. 3.10 illustrates the simulation set-up. During the first mission, the robot detects an obstacle in region r_7 that blocks the corridor. Thus, arcs containing region r_7 will be set with the maximum cost of the arcs in the environment, $\max\{C_{ij}\}$. After the cost update, path 2 is no longer an alternative path to follow, since its cost is very high. Parameters $\alpha_2 = 1$ and $\beta_2 = 1$ are updated. Fig. 3.11 shows the evolution of the total cost of both paths connecting region r_1 to region r_8. Initially, the cost of path 1 (red dashed line) is higher than the cost of path 2 (green continuous line), and the robot follows path 2. After an obstacle type O_2 has been detected, the total cost of path 2 increases and for approximately 55 travels, this cost is the highest one. Consequently,

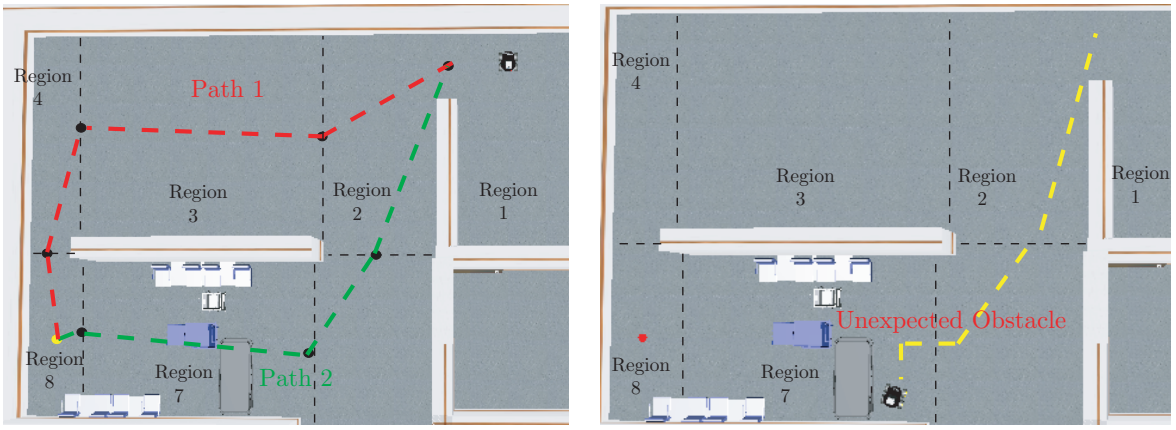


FIGURE 3.10: a) Two alternative paths connecting region r_1 to region r_8. b) The robot selected path 2, because it is the lowest cost path. During the mission, the robot detects an obstacle totally blocking the passage.

the robot follows path 1. Parameter α_2 is not updated since no obstacles type O_2 are detected, and β_2 increases to 55. The exponential decay of the appearance of an

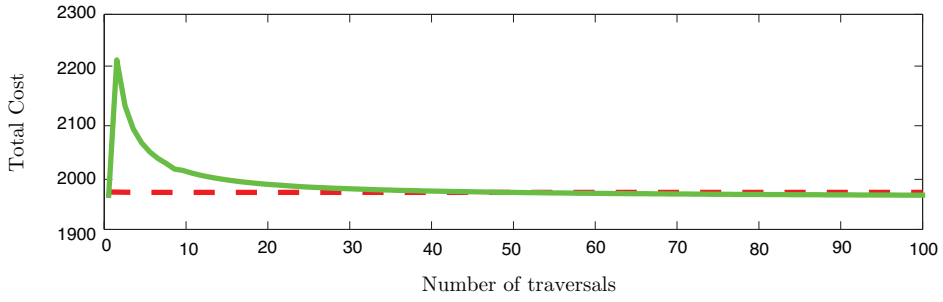


FIGURE 3.11: Evolution of the total cost of path 1 (red dashed line) and path 2 (green continuous line).

obstacle type O_2 allows the robot to forget the influence of past unexpected obstacles and to re-select the paths where such obstacles were previously detected.

3.2.4 Global Planner

The Global Planner block aims at finding a sequence of regions, ρ, connecting the robot's position, $\hat{P}_{\mathrm{r}}$, to the neighborhood of the goal location, $B(P_{\mathrm{g}}, \epsilon)$. ϵ is a suitable radius that defines the neighborhood in which the robot reaches P_{g}. This value is calculated based on the expected uncertainty on the robot's pose and allows the robot to stop in the neighborhood of the goal location, despite the inertia on the robot.

The sequence of regions $\rho \equiv \{r_1, \ldots, r_{n_a}\}$ is found by selecting the path with the minimum cost C_a connecting $\hat{P}_{\mathrm{r}}$ to P_{g},

$$J = \min\{C_a\}, \ a \in \{1, ..., N\}, \tag{3.12}$$

where N is the number of possible paths connecting $\hat{P}_{\mathrm{r}}$ to P_{g}, n_a is the number of regions r_i in sequence ρ and C_a is defined as,

$$f_1 \triangleq C_a = E\left[\sum_{k=0}^{n_a-1} M(r_k, r_{k+1})\right], \tag{3.13}$$

subject to constraints $r_0 = r_s$, the starting region; and $r_{n_a} = r_g$ the goal region. $E[.]$ denotes the expectation operation on the transition function M and $M(r_k, r_{k+1})$ gives the cost $C_{k,k+1}$ of the arc connecting r_k to r_{k+1}.

Minimization of the cost function is achieved by using a search path algorithm, namely the Djikstra's algorithm. This algorithm receives the transition function, M, from the World Representation block, the current robot's position, $\hat{P}_{\mathrm{r}}$, from the Localization System and the goal location, P_{g}, provided by the user. As output, the Djikstra's algorithm provides the sequence of regions, ρ, which satisfies the minimization and connects $\hat{P}_{\mathrm{r}}$ to P_{g}. If no sequence of regions is available, the region where P_{g} lies is unreachable by the robot. The choice of the Djisktra's algorithm was motivated because it is simple and yields similar results to others.

Fig. 3.12 shows an example of the search path problem. The robot starts its mission in region r_3 (red circle) and the goal location P_{g} (green cross) lies in region r_{16}. The Djisktra's algorithm calculates the number of different paths that successfully connect $\hat{P}_{\mathrm{r}}$ to P_{g}. In this example, two possible paths are available, $N = 2$, each one with an associated cost, C_a. Path 1 (blue line) is composed by the sequence of regions $\rho_1 = \{r_3, r_2, r_6, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15},\ r_{16}\}$ and path 2 (yellow line) is composed by the sequence of regions $\rho_2 = \{r_3, r_4, r_8, r_7, r_6, r_{10}, r_{11},\ r_{12}, r_{13}, r_{14}, r_{15}, r_{16}\}$. The path with the minimum cost will be selected.

3.2.5 Look-up Table

The Look-up Table block is composed by a feed-through map responsible for the transition between the regions of ρ. For instance, when the robot is in region r_i, it must know which region is the next one to move. In order to ensure the transition between regions, the Look-up Table provides a local goal $P_{\mathrm{b}}(x_{\mathrm{b}}, y_{\mathrm{b}})$ that the robot should follow to cross to the next region. This point belongs to the critical line $l_{i,i+1}$ that divides the region where the robot is, r_i, and the next region to be crossed, r_{i+1}. In a general view, the Look-up Table provides a sequence of local points P_{b} that guides the robot to the final region.

A critical line $l_{i,i+1}$ can be represented through a linear equation, where its extremities are the points that intersect the walls of the environment, $P_{2,i}(x_{2i}, y_{2i})$. Note that

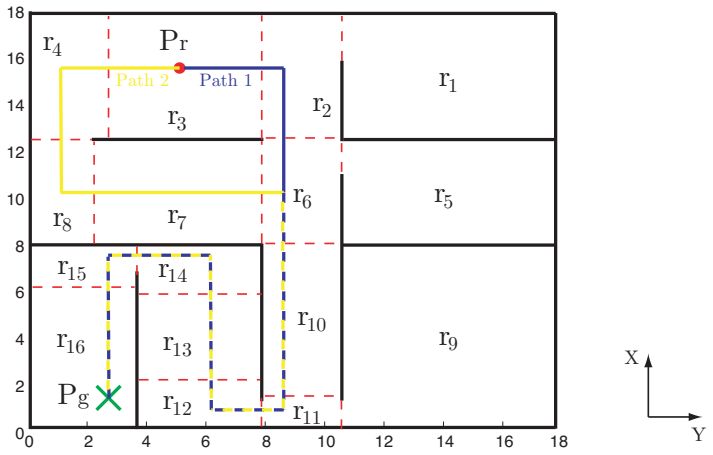


FIGURE 3.12: Robot starts its mission in region r_3 (red circle) and goal location, P_g (green cross) is located in region r_{16}. Path 1 is depicted by a blue line and path 2 is depicted by a yellow line.

$l_{i,i+1}$ is named as critical line, however in a mathematical sense, $l_{i,i+1}$ is a line segment. The number of critical lines to be crossed is equal to $n_a - 1$.

Critical lines are selected according to the region where the robot is. Considering that the robot is in region r_1 and the next region to reach is r_2, the selected critical line should be the one that defines the border between r_1 and r_2, $l_{1,2}$.

The robot's position, $\hat{P}_r$, can be used to verify in which region the robot is, and therefore what is the next region to move. However, the inaccurate robot's pose provided by the Localization System, which contains the uncertainty τ (see fig.3.4), could lead the robot to present oscillations on its motor behavior. For instance, when the robot is close to a boundary between regions, the estimated robot's position can alternate between neighbor regions. If the selection of the critical line depends on the perceived region, the alternation between critical lines will cause an oscillatory behavior on the robot. Fig. 3.13 shows an example of the robot's position (black circle) when the robot is close to a boundary (red dashed line) between regions. The space of beliefs is represented by the green circle. In $t = 1$ s, the robot considers that it is in region r_1 and selects the critical line $l_{1,2}$. At $t = 2$ s, the robot has moved to another position and considers that it already has crossed to region r_2. However, note that according to the space of beliefs, the robot's true position can be in region r_1. The robot selects the critical line diving region r_2 and the next region. At $t = 3$ s, the robot considers that it is in region r_1. Consequently, critical line $l_{1,2}$ is selected again and this causes an oscillation on the critical lines selections. Therefore, the robot can exhibit oscillations in its motor behavior.

In order to avoid the oscillatory behavior, the space of beliefs of the robot's position

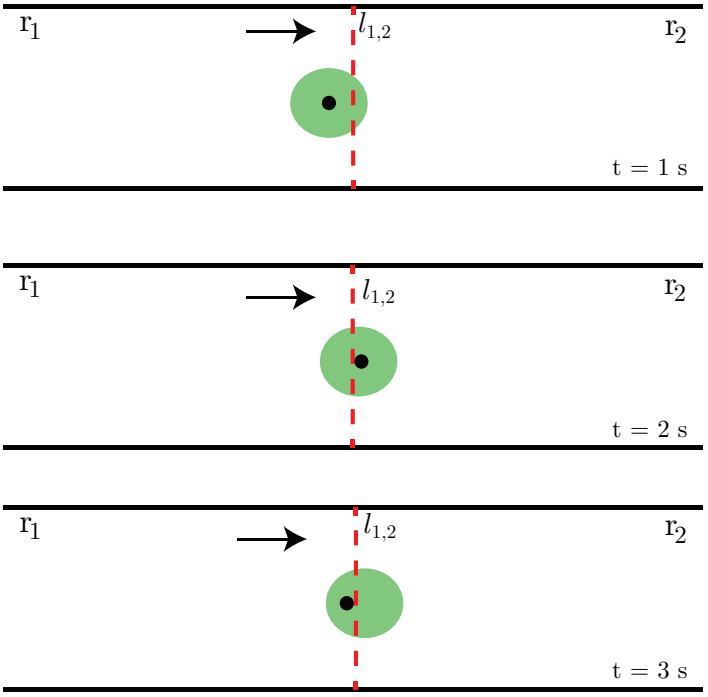


FIGURE 3.13: Example of a trajectory followed by the robot when it is close to a boundary between regions. Arrows indicate the direction of the robot's movement.

is used to know in which region the robot is. Thus, the robot's position is represented as an area, and therefore, the selection of the critical lines must be obtained according to the area where the robot is. Areas where the robot can wrongly perceive its current region are named as critical areas, r_{ci}. They depend on τ and on the extremities of the respective critical line, $P_{1,i}$, $P_{2,i}$, as follows,

$$r_{ci} = \left\{ (x, y) \mid x(u) = \frac{w}{2} \operatorname{sgn}(\cos(u)) \ , \ y(u) = \frac{h}{2} \operatorname{sgn}(\sin(u)), \ 0 \leq u \leq 2\pi \right\}, \quad (3.14)$$

where h is the Euclidean distance between $P_{1,i}$ and $P_{2,i}$, $w = 4\tau$ and $\operatorname{sgn}(x)$ is defined as in (A.7). The greater the uncertainty τ, the larger the size of the critical areas.

The next critical line should be selected only when the robot has certainty that it is in a critical area. The robot is in a critical area if its space of beliefs, $B\left(P_{\mathrm{r}}, \tau\right)$, is completely within a critical area. This is verified if the Euclidean distance between the robot's position, $\hat{P}_{\mathrm{r}}$, and the local goal, P_{b}, defined as,

$$u_k = \|\hat{P}_{\mathrm{r}_k} - P_{\mathrm{b}_k}\|, \quad (3.15)$$

is less than 2τ. A C^1 feed-through map that verifies if $u_k < 2\tau$ can be given as follows,

$$f_2 \triangleq e_k = \frac{1 + \tanh\left(b\left(u_k - 2\tau\right)\right)}{2}, \tag{3.16}$$

which gives 0, if $u_k < 2\tau$ (the space of beliefs of the robot's position is outside a critical area), and 1, if $u_k > 2\tau$ (the space of beliefs of the robot's position is within a critical area). The selection of the critical line that the robot should cross, L, can be given through the following condition,

$$f_3 \triangleq L_k = \tanh\left(b\left(1 - e_k\right)\right) l_{i,i+1} + \tanh\left(b(e_k)\right) l_{i+1,i+2}. \tag{3.17}$$

If the robot is outside of a critical area, the local goal $P_\mathrm{b} \in l_{i,i+1}$ is selected. On the other hand, if the robot is within a critical area, $P_\mathrm{b} \in l_{i+1,i+2}$ is selected. The selection of the next critical line through the critical areas augments the transition zone between regions from a line to an area. Note that other C^1 feed-through maps could be used to represent f_2 and f_3.

Fig. 3.14 depicts an example of a sequence of regions, $\rho = \{r_1, r_2, r_3, r_4\}$, in which the critical lines dividing neighbor regions are identified by red dashed lines and the critical areas are represented by green areas. Considering that the robot (red circle identifies the space of beliefs of the robot's position) starts its mission in r_1. To reach the goal location P_g, it must cross $l_{1,2}$ to reach r_2, $l_{2,3}$ to reach r_3, and finally cross $l_{3,4}$ to reach the final region r_4 where P_g (green cross) lies. In region r_1, while the robot

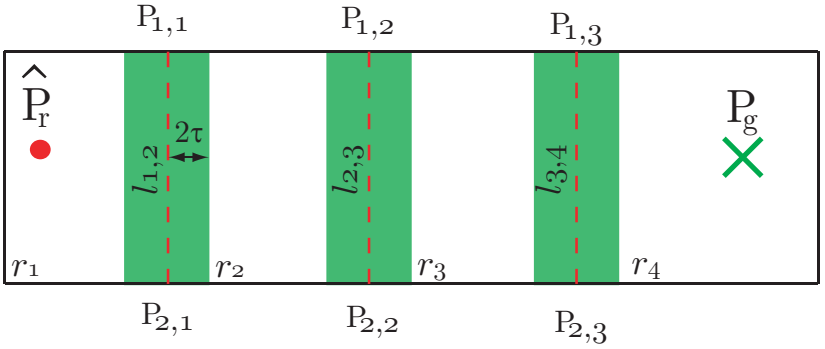


FIGURE 3.14: A corridor of an environment composed by the sequence of regions, $\rho = \{r_1, r_2, r_3, r_4\}$. Red dashed lines depict the critical lines, green areas represent critical areas and points $P_{1,i}, P_{2,i}$ hold for the extremities of the critical lines.

is outside the critical area, the local goal P_b belongs to the critical line $l_{1,2}$. When the robot approaches the border between region r_1 and r_2, it eventually enters the critical area. In this case, $l_{2,3}$ is selected and $P_\mathrm{b} \in l_{2,3}$. This local goal remains selected, until the robot enters the next critical area around the border between regions r_2 and r_3.

The same procedure holds for the following regions, until the robot reaches the final region r_4.

3.2.6 Global - Local Integration

The Global - Local Integration block receives the critical line $l_{i,i+1}$ or $l_{i+1,i+2}$ depending on whether the space of beliefs of the robot's position is inside or outside of a critical area, and provides the direction ψ_{tar} that the robot should follow. In order to calculate ψ_{tar}, it is required to obtain the local goal $P_{\text{b}}(x_{\text{b}}, y_{\text{b}})$, which is calculated at each time step as the orthogonal projection of $\hat{P}_{\text{r}}$ onto the received critical line. See appendix A.2 for more details.

While the robot traverses regions by following the local goals $P_{\text{b}}(x_{\text{b}}, y_{\text{b}})$, abrupt changes in the critical lines will occur. Without losing generality and to maintain the consistency with C^1 dynamical systems, the local goal P_{b} is chosen by a smooth function. Local goal P_{b} can be calculated by integrating the following dynamical systems,

$$f_4 \triangleq x_{\text{b}_{k+1}} = x_{\text{b}_k} + \lambda_{\text{tr}} \left(x_{\text{b}_k} - x_{\text{b}}\right) d_k, \tag{3.18}$$

$$f_5 \triangleq y_{\text{b}_{k+1}} = y_{\text{b}_k} + \lambda_{\text{tr}} \left(y_{\text{b}_k} - y_{\text{b}}\right) d_k, \tag{3.19}$$

where $\lambda_{\text{tr}} = \frac{1}{\tau_{\text{tr}}}$ defines the relaxation rate of the dynamical systems and d_k is the discretization step. These smooth functions allow analyzing the stability during the transitions of regions. Otherwise, only the stability inside each region r_i would be assured.

In the last region (where the goal location lies), the point P_{b} does not belong to a critical line, but it is assumed as the final goal location, $P_{\text{b}} = P_{\text{g}}$.

Finally, the direction that the robot should follow, ψ_{tar}, to move across the critical lines between regions is calculated as follows,

$$f_6 \triangleq \psi_{\text{tar}_k} = \arctan\left(\frac{y_{\text{b}_k} - y_{\text{r}_k}}{x_{\text{b}_k} - x_{\text{r}_k}}\right). \tag{3.20}$$

Fig. 3.15 depicts an example in which the robot has to move from region r_3 to region r_2 by crossing critical line $l_{3,2}$. The point P_{b} is calculated as the orthogonal projection of the robot's position $\hat{P}_{\text{r}}$ onto $l_{3,2}$. In fig. 3.15 (a), no obstacles appear in the environment and the robot follows a straight trajectory to reach P_{b}. In fig. 3.15 (b), an unexpected obstacle (red rectangle) suddenly appears, and the robot needs to circumnavigate it. As the robot moves, P_{b} is being updated according to (3.18) and (3.19).

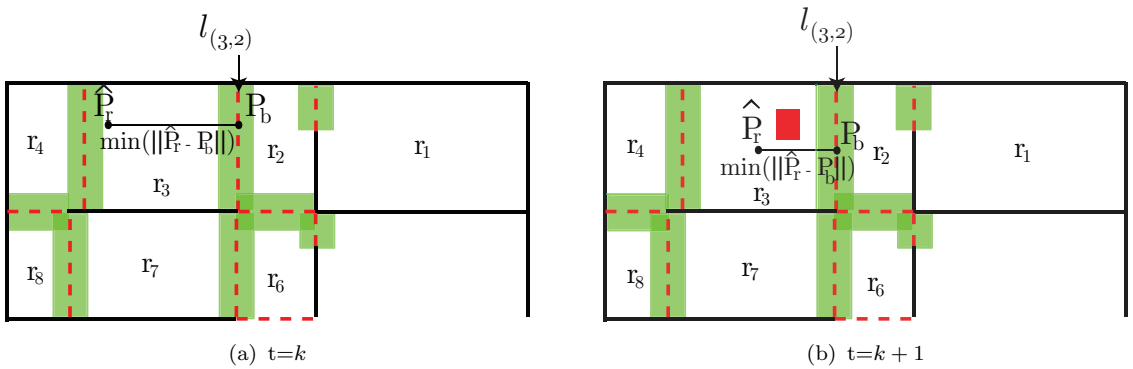


FIGURE 3.15: a) Robot moves from region r_3 to region r_2 by crossing critical line $l_{3,2}$ at time $t = k$. b) Robot circumnavigates an unexpected obstacle (red rectangle) at time $t = k + 1$ and the local goal P_b is updated according to (3.18) and (3.19). Green areas represent the critical areas.

3.3 Dynamic Approach to Local Control

This section describes the Local Control module composed by a dynamic approach responsible for controlling the robot's angular velocity, ϖ. The robot should be able to navigate in disturbed environments, full of obstacles like people and other objects. The robot has no knowledge about the position of unexpected obstacles. However, it should be able to detect and circumnavigate them.

The Local Control module receives the direction, ψ_{tar}, that the robot should follow to reach P_g, and sensory information that identifies the direction of obstacles, $\psi_{\text{obs},i}$. This module adapts the robot's heading direction, ϕ, to guide the robot towards the goal location, P_g, while avoiding obstacles.

The choice of this dynamical approach [74, 82] as the local planner method was motivated since its properties allow the integration for sensory-motor feedback, which provides a closed-loop control. In addition, it is consistent in terms of C^1 dynamical systems. However, for navigation purposes, any other local planner method could be used as well.

Fig. 3.16 depicts the schematic of the Local Control module. It includes the Local Planner block that uses the dynamic approach to provide the robot's angular velocity, ϖ.

In the following, the basic concepts of the dynamic approach to obstacle avoidance and target orientation for mobile robots are reviewed (see [74] for an extended review).

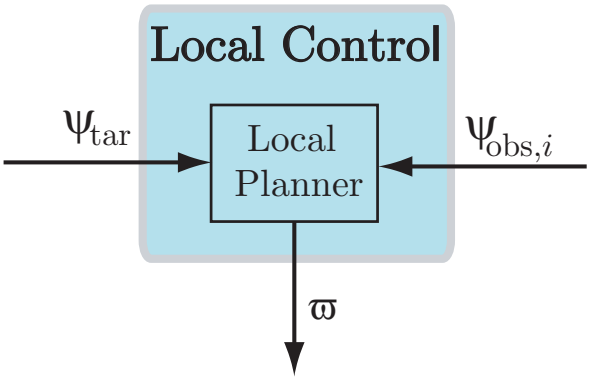


FIGURE 3.16: Schematic of the Local Control module. It receives as inputs the direction to the goal location, ψ_{tar}, and the direction where obstacles are detected, $\psi_{\text{obs},i}$. It provides the robot's angular velocity, ϖ.

3.3.1 Behavior Variables

The dynamical systems theory can be used as a theoretical language and tool to design, specify, analyze and implement the heading dynamics of robotic platforms [84]. In the context of the dynamical systems approach, it is fundamental to find variables that explicitly describe, parameterize and represent the state of the system. These variables are called behavioral variables. In the robotic framework, they represent the behavioral state of the robot and the necessary requirements for the robot to execute its missions. Each behavioral variable is governed by a nonlinear vector field.

The robot's heading direction is defined as the behavioral variable, because at all instants of time, the heading direction is near or in a resulting attractor created by the obstacle avoidance and target orientation contributions. The complete behavioral dynamic that governs the robot's heading direction, ϕ, is achieved by summing the individual contributions (target orientation and obstacle avoidance). Each individual contribution is characterized by three different parameters: the behavioral variable, a relaxation rate responsible for defining the strength of attraction or repulsion, and the range of the behavioral variable over which the force exerts its effect, $[0, 2\pi[$.

As the robot moves, the directions to obstacles and the goal location change, so that the resulting attractor shifts, pulling the robot's heading along. Because the angles are related to the allocentric reference frame, $\{W\}$[4], the contributions to the dynamical system of the heading direction do not depend on the robot's heading direction, ϕ.

When an obstacle is detected, a repulsive force is erected at direction $\psi_{\text{obs},i}$, relative to the allocentric reference frame, $\{W\}$. The repulsive force erects an unstable fixed point on the dynamics of the behavioral variable. Similarly, the existence of a goal

[4]The reference frames can be consulted in section 5.4

location in the environment erects an attractive force at direction ψ_{tar}, which creates a stable fixed point and attracts the robot's heading direction to the direction where the goal location lies. The complete behavioral dynamics were implemented on a physical mobile robot in [13, 84, 85, 151], and are described in the following.

3.3.2 Target Orientation

In an undisturbed environment, the robot's heading direction ϕ closely follows ψ_{tar} (given by (3.20)), since the movement of the robot is not disturbed by obstacles.

Fig. 3.17 depicts the angle ψ_{tar} between the robot's position, $\hat{P}_{\text{r}}$, and a goal location, P_{g}. The orientation ψ_{tar}, specifies the fixed point with an attractive force that attracts

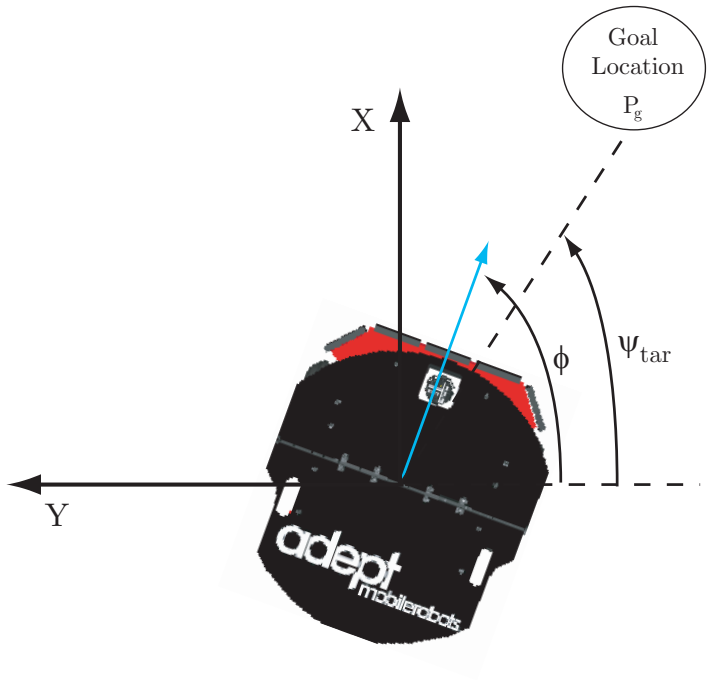


FIGURE 3.17: Schematic representing the direction to the goal location, ψ_{tar} and the robot's heading direction, ϕ, relative to the allocentric reference frame, $\{W\}$.

the robot's heading direction, ϕ,

$$F_{\text{tar}}(\phi) = \lambda_{\text{tar}} \sin(\phi - \psi_{\text{tar}}), \tag{3.21}$$

where $\lambda_{\text{tar}} = \frac{1}{\tau_{\text{tar}}}$ defines the strength of attraction that the orientation ψ_{tar} exerts on the robot's heading direction. τ_{tar} gives the relaxation rate. This nonlinear system has 2 fixed points, ψ_{tar} and $\psi_{\text{tar}} + \pi$. If $\lambda_{\text{tar}} < 0$, ψ_{tar} behaves as an attractor and $\psi_{\text{tar}} + \pi$ as a repeller. Otherwise, if $\lambda_{\text{tar}} > 0$, the fixed point $\psi_{\text{tar}} + \pi$ behaves as an attractor, and the fixed point ψ_{tar} behaves as a repeller. In this case, the robot's heading direction,

ϕ is shifted from the direction ψ_{tar}. $F_{\text{tar}}(\phi)$ results on a sinusoidal contribution to provide the same dynamics when complete rotations (2π rad) are performed.

Fig. 3.18 depicts an example for a negative λ_{tar}. An attractor is erected at the direction where the goal location lies, ψ_{tar}, and a repeller is erected at the opposite direction, $\psi_{\text{tar}} + \pi$. The robot's heading direction, ϕ is forced to converge to ψ_{tar}, independently of the initial robot's heading direction within the interval of time $[0, 2\pi[$.

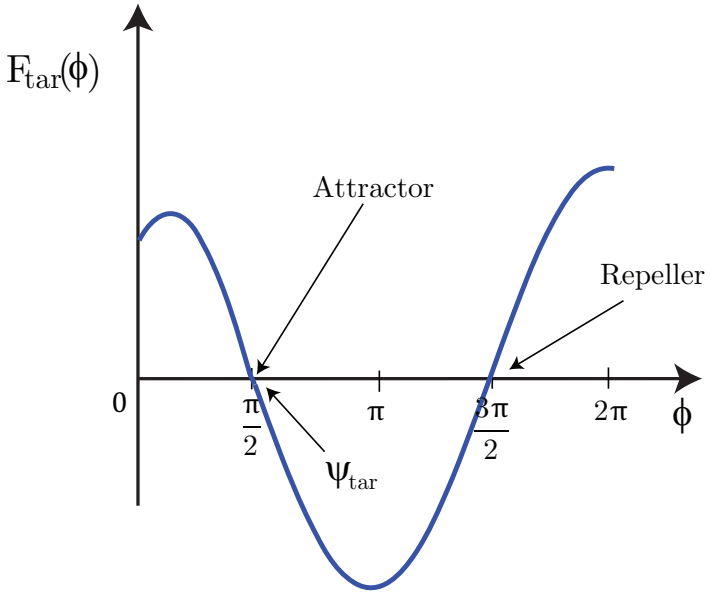


FIGURE 3.18: Vector field of the target orientation contribution. At the location $\phi = \psi_{\text{tar}}$ exists a zero crossing whose rate of change has negative slope. Thus, an attractive force at that location is erected.

3.3.3 Obstacle Avoidance

The robot senses its surroundings through a laser range finder and sonars. The laser measures 682 steps (see appendix B.1), $N_l = 682$, and each laser step, La_i, has an angle θ_i relative to the robot coordinate reference frame $\{R\}$. Hence, relatively to the allocentric reference frame, each laser step measures the environment into the direction $\psi_{\text{obs},i} = \theta_i + \phi$.

8 sonars are mounted on the robot, $N_s = 8$, as depicted in fig. B.2. Each sonar, S_i, has an angle $\theta_{\text{s},i}$ relative to reference $\{R\}$. Obstacles are detected relatively to the allocentric reference frame into the direction $\psi_{\text{sobs},i} = \theta_{\text{s},i} + \phi$.

The robot's heading direction, ϕ, is not required to the obstacle avoidance algorithm, since only the angles, θ_i and $\theta_{\text{s},i}$ are necessary. Note that $-\theta_i = \phi - \psi_{\text{obs},i}$ and $-\theta_{\text{s},i} = \phi - \psi_{\text{sobs},i}$. Fig. 3.19 depicts a top perspective of the robot and the laser mounted on

it. The yellow shadow represents the area over which the laser senses its surroundings. The obstacle is detected into the interval of directions $\{\psi_{\text{obs}_{128}}, \ldots, \psi_{\text{obs}_{170}}\}$.

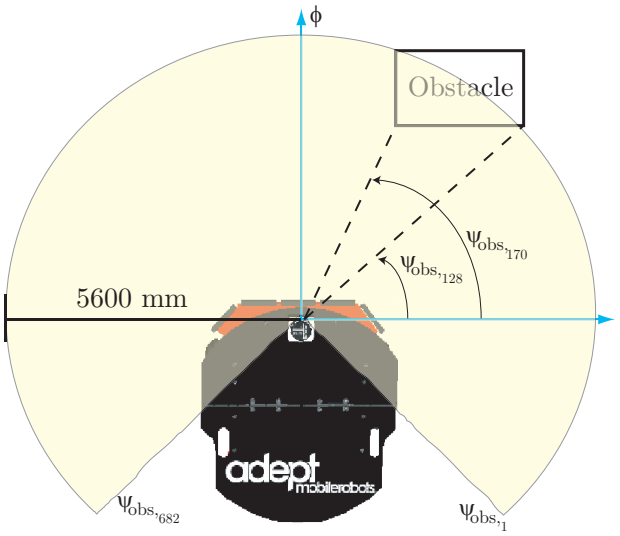


FIGURE 3.19: Robot and laser range finder seen through a top perspective. Yellow shadow depicts the area over which the laser measures its surroundings. Each laser step measures the environment into direction $\psi_{\text{obs},i}$ relatively to the allocentric reference frame. Two laser steps are represented, $\psi_{\text{obs},_{128}}$ and $\psi_{\text{obs},_{170}}$.

When a laser step, La_i, detects an obstacle, a direction representing that obstacle $\psi_{\text{obs},i}$ is specified and a repulsive force $f_{\text{obs},i}$ centered at $\psi_{\text{obs},i}$ is erected,

$$f_{\text{obs},i} = \lambda_{\text{obs},i}\,(\phi - \psi_{\text{obs},i}) \exp\left[\frac{-(\phi - \psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i}}\right], \quad i = 1, ..., N_l, \tag{3.22}$$

where i subscribes the laser step. When a sonar sensor S_i detects an obstacle at direction $\psi_{\text{sobs},i}$, a repulsive force $f_{\text{sobs},i}$ is erected at that direction,

$$f_{\text{sobs},i} = \lambda_{\text{sobs},i}\,(\phi - \psi_{\text{sobs},i}) \exp\left[\frac{-(\phi - \psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i}}\right], \quad i = 1, ..., N_s, \tag{3.23}$$

where i subscribes the sonar index and $N_s = 8$. Parameters $\lambda_{\text{obs},i}$ and $\lambda_{\text{sobs},i}$ are responsible for defining the intensity of each repeller according to the distance to the detected obstacle. The closer the robot, the higher the values of $\lambda_{\text{obs},i}$ and $\lambda_{\text{sobs},i}$, as well as the intensity of the respective obstacle repulsing forces, $f_{\text{obs},i}$ and $f_{\text{sobs},i}$. This allows the robot to avoid obstacles more quickly when they are near than far.

Parameters $\lambda_{\text{obs},i}$ and $\lambda_{\text{sobs},i}$ are mathematically defined as follows,

$$\lambda_{\text{obs},i} = \beta_{\text{obs},1} \exp\left[\frac{-d_{\text{l},i}}{\beta_{\text{obs},2}}\right], \quad i = 1, ..., N_l, \tag{3.24}$$

$$\lambda_{\text{sobs},i} = \beta_{\text{sobs},1} \exp\left[\frac{-d_{\text{s},i}}{\beta_{\text{sobs},2}}\right], \quad i = 1, ..., N_s, \tag{3.25}$$

where $d_{\text{l},i}$ and $d_{\text{s},i}$ stand for the distance to an obstacle measured by the laser step La_i and sonar S_i, respectively. $\beta_{\text{obs},1}$ and $\beta_{\text{sobs},1}$ define the maximum value for parameters $\lambda_{\text{obs},i}$ and $\lambda_{\text{sobs},i}$, respectively. The relaxation rates of the dynamics are defined as $\tau_{\text{obs},i} = \frac{1}{\lambda_{\text{obs},i}}$ and $\tau_{\text{sobs},i} = \frac{1}{\lambda_{\text{sobs},i}}$.

Parameters $\beta_{\text{obs},2}$ and $\beta_{\text{sobs},2}$ define the decay rate of the obstacle repulsion force. Obstacles farther than $\beta_{\text{obs},2}$ (or $\beta_{\text{sobs},2}$) are repelled weakly than nearby obstacles. Both contributions decay exponentially with distances $d_{\text{l},i}$ and $d_{\text{s},i}$.

Parameters $\sigma_{\text{obs},i}$ and $\sigma_{\text{sobs},i}$ define the angular range over which the obstacle repulsion forces, $f_{\text{obs},i}$ and $f_{\text{sobs},i}$, exert their effect on the robot,

$$f_{7_a} \triangleq \sigma_{\text{obs},i_k} = \arctan\left[\tan\left(\frac{\Delta\theta}{2}\right) + \frac{\text{R}_{\text{robot}}}{\text{R}_{\text{robot}} + d_{\text{l},i}}\right], \tag{3.26}$$

$$f_{7_b} \triangleq \sigma_{\text{sobs},i_k} = \arctan\left[\tan\left(\frac{\Delta\theta_{\text{s}}}{2}\right) + \frac{\text{R}_{\text{robot}}}{\text{R}_{\text{robot}} + d_{\text{s},i}}\right], \tag{3.27}$$

where $\Delta\theta$ and $\Delta\theta_{\text{s}}$ are constant values and stand for the angular range of laser step La_i, and sonar sensor S_i. They are defined as $\Delta\theta = 0.0061$ rad and $\Delta\theta_{\text{s}} = 0.52$ rad. Parameter R_{robot} subscribes the radius of the robot. Fig. 3.20 depicts an example of the repulsion range, $\sigma_{\text{obs},341}$, when an obstacle appears in front of the robot. The obstacle is detected by several laser steps, however for visual purposes it is only illustrated the laser step $La_{,341}$ and its measured distance to an obstacle, $d_{\text{l},341}$.

For the obstacle avoidance behavior, all contributions from the laser steps and sonars sensors are considered. The obstacle contributions are summed, resulting in the repeller standing for the direction that the robot should avoid,

$$F_{\text{obs}}(\phi) = \sum_{i=1}^{N_l} f_{\text{obs},i}(\phi) \quad i = 1, ..., N_l, \tag{3.28}$$

and

$$F_{\text{sobs}}(\phi) = \sum_{i=1}^{N_s} f_{\text{sobs},i}(\phi) \quad i = 1, ..., N_s. \tag{3.29}$$

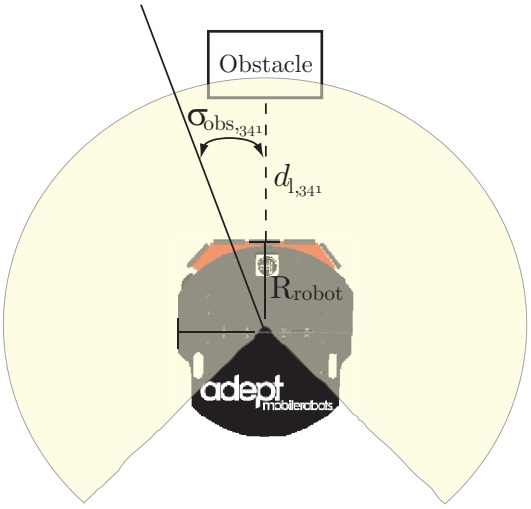


FIGURE 3.20: Range of the laser step, $La,_{341}$, detecting an obstacle at a distance $d_{l,341}$. A repulsive force is erected at $\psi_{obs,341}$ and the repulsion range is $\sigma_{obs,341}$.

Fig. 3.21 depicts an example of the repeller created by summing the obstacle contributions from the laser and sonars. The repeller is located at $\psi_{obs} = \frac{\pi}{2}$ and this is the direction that the robot should avoid.

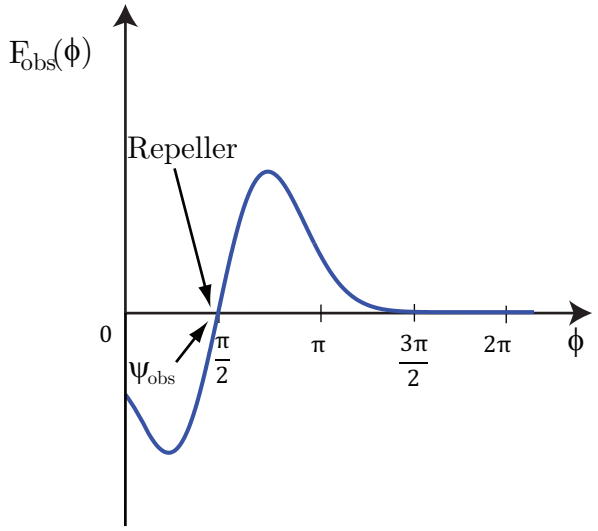


FIGURE 3.21: Vector field of the obstacle avoidance contribution. At location $\phi = \psi_{obs}$, there is a zero crossing and the rate of change has positive slope. Thus, a repulsive force at that location is erected.

3.3.4 Target Orientation and Obstacle Avoidance Integration

The integration of obstacle avoidance, $F_{obs}(\phi)$ and $F_{sobs}(\phi)$, and target orientation $F_{tar}(\phi)$ contributions is achieved by adding each contribution to the vector field that

governs the heading direction dynamics,

$$\begin{aligned} F_{\text{res}}(\phi) &= \dot{\phi} = F_{\text{obs}}(\phi) + F_{\text{sobs}}(\phi) + F_{\text{tar}}(\phi) + F_{\text{stoch}}, \\ f_8 \triangleq \phi_{k+1} &= \phi_k + \left(\sum_{i=1}^{N_l} \lambda_{\text{obs},i} \left(\phi_k - \psi_{\text{obs},i}\right) \exp\left[\frac{-(\phi_k - \psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}} \right] \right. \\ &+ \sum_{i=1}^{N_s} \lambda_{\text{sobs},i} \left(\phi_k - \psi_{\text{sobs},i}\right) \exp\left[\frac{-(\phi_k - \psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i_k}} \right] \\ &+ \left. \lambda_{\text{tar}} \sin(\phi_k - \psi_{\text{tar}}) + F_{\text{stoch}} \right) d_k \end{aligned} \tag{3.30}$$

As the robot moves, both sensory information and heading direction dynamics change. Eventually, the sum of the target orientation and obstacle avoidance contributions generates local minima, and the robot's heading direction may lie over an unstable state. To overcome this problem, a typical solution consisting on adding a stochastic component, F_{stoch}, to the vector field has been proposed [1, 74, 289],

$$F_{\text{stoch}} = \sqrt{Q\xi_n}, \tag{3.31}$$

where ξ_n is a Gaussian white noise with zero mean and unit variance, and Q the effective variance. This solution ensures an escape from unstable states in finite time.

Fig. 3.22 depicts the vector field resulting from the integration between the attractive contribution, $F_{\text{tar}}(\phi)$ (see fig. 3.18) and the obstacle avoidance contributions $F_{\text{obs}}(\phi)$ and $F_{\text{sobs}}(\phi)$ (see fig. 3.21). In this example, an attractor is erected at $\phi = \frac{\pi}{2}$ rad and a repeller at $\phi = \frac{3\pi}{2}$ rad.

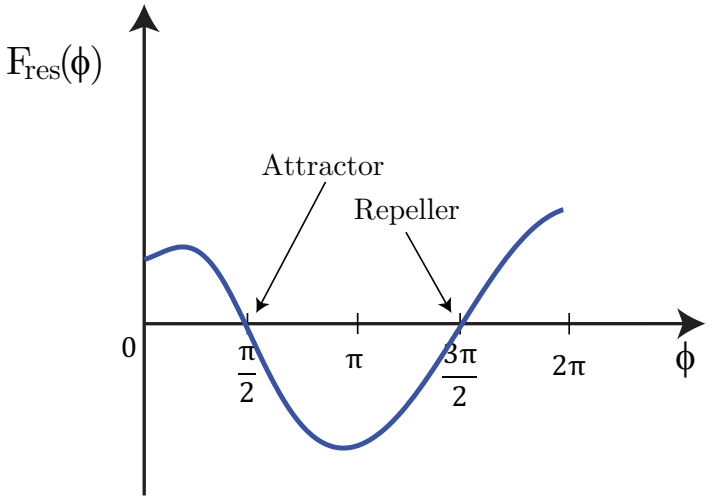


FIGURE 3.22: Vector field resulting from the superposition of the attractive and repulsive forces.

3.3.5 Detection of Obstacles

A systematic way to indicate if obstacles are present in the environment is to construct a function, $U(\phi)$, by integrating the laser obstacle contribution given by (3.28),

$$f_9 \triangleq U_k(\phi_k) = \sum_{i=1}^{N_l} \left(\lambda_{\text{obs},i} \sigma^2_{\text{obs},i_k} \exp \left[\frac{-(\phi_k - \psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}} \right] - \frac{\lambda_{\text{obs},i} \sigma^2_{\text{obs},i_k}}{\sqrt{e}} \right). \tag{3.32}$$

If $U(\phi) \geq 0$, the repulsion from the obstacle contribution is weak and the robot does not change its direction. If $U(\phi) < 0$, the robot is on a repulsion zone created by obstacles. See detailed information about $U(\phi)$ in [74, 289]. The same procedure is applied to construct the function, $U_s(\phi)$, which integrates the sonar obstacle contribution given by (3.29).

3.3.6 Target Orientation and Obstacle Avoidance Behaviors

Obstacle avoidance behavior should predominate over the target orientation one. In a sense, it is more important that the robot avoids collisions and does not reach the goal location, than colliding with an obstacle when trying to reach the goal. Thus, people and physical components of the robot are protected against damages.

Obstacle avoidance behavior predominates over the target orientation if its contribution is stronger than the target orientation one, $F_{\text{obs}}(\phi) > F_{\text{tar}}(\phi)$ and $F_{\text{sobs}}(\phi) > F_{\text{tar}}(\phi)$. By construction,

$$\sum_{i=1}^{N_l} \beta_{\text{obs},1} \exp \left[\frac{-d_{\text{l},i}}{\beta_{\text{obs},2}} \right] > \lambda_{\text{tar}}, \tag{3.33}$$

$$\sum_{i=1}^{N_s} \beta_{\text{sobs},1} \exp \left[\frac{-d_{\text{s},i}}{\beta_{\text{sobs},2}} \right] > \lambda_{\text{tar}}, \tag{3.34}$$

ensures that the intensity of each individual obstacle avoidance contribution is stronger than the target orientation contribution. The first terms in (3.33) and (3.34) define the maximum force exerted by an individual obstacle contribution and the exponential terms depend on the distance measured to obstacles, $d_{\text{l},i}$ and $d_{\text{s},i}$. If the obstacles are thin, only a low number of laser steps, La_i, detect them. On the other hand, if the obstacles are large, a high number of laser steps will detect them. A thin obstacle should erect a sufficient repulsive force allowing the obstacle circumnavigation. A large obstacle should not erect an excessive repulsive force. Consequently, parameters defining the precedence of obstacle avoidance over target orientation depend on the

environment configuration. Empirically, these parameters are defined such that conditions (3.33) and (3.34) are validated, $\beta_{\text{obs},1} = 0.7$, $\beta_{\text{sobs},1} = 5$, $\lambda_{\text{tar}} = 1$, $\beta_{\text{obs},2} = 0.9$ and $\beta_{\text{sobs},2} = 0.5$.

The worst case scenario consists on a situation in which an obstacle is located between the robot and the goal location, P_{g} (see fig. 3.23). Consider that P_{g} attracts

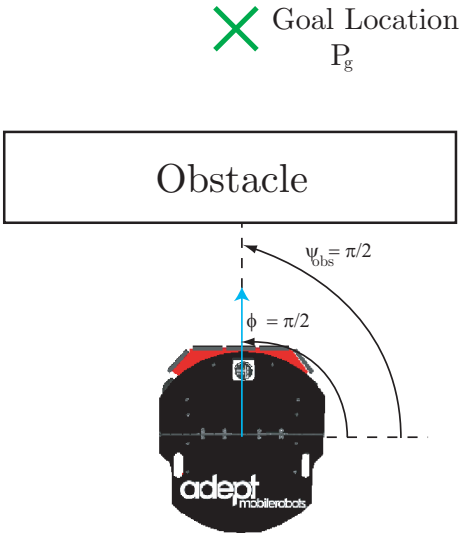


FIGURE 3.23: Worst case scenario in which an obstacle lies between the robot and the goal location.

the robot's heading direction to $\psi_{\text{tar}} = \frac{\pi}{2}$ and the obstacle repels it to $\psi_{\text{obs}} = \frac{\pi}{2}$. In this situation, the direction that the robot should follow coincides with the one that it should avoid. Fig. 3.24 depicts the vector field contributions for this scenario. To ensure that the robot does not collide with the obstacle, the obstacle contribution has to be stronger than the target orientation contribution. Consequently, conditions (3.33) and (3.34) are verified. Panel (a) depicts the vector field of the target orientation contribution. Note that the robot's heading direction $\phi = \frac{\pi}{2}$ is near the attractive fixed point, meaning that the robot is in the correct direction to reach the goal location. Panel (b) depicts the vector field of the laser obstacle avoidance contribution, $F_{\text{obs}}(\phi)$. The obstacle is in front of the robot and a repeller is created at $\psi_{\text{obs}} = \frac{\pi}{2}$. Panel (c) depicts the resulting vector field of the heading direction dynamics. As the obstacle avoidance contribution is stronger than the target orientation contribution, a resulting repeller is erected at direction $\phi = \frac{\pi}{2}$ and the robot's heading direction, ϕ moves away from $\phi = \frac{\pi}{2}$ rad.

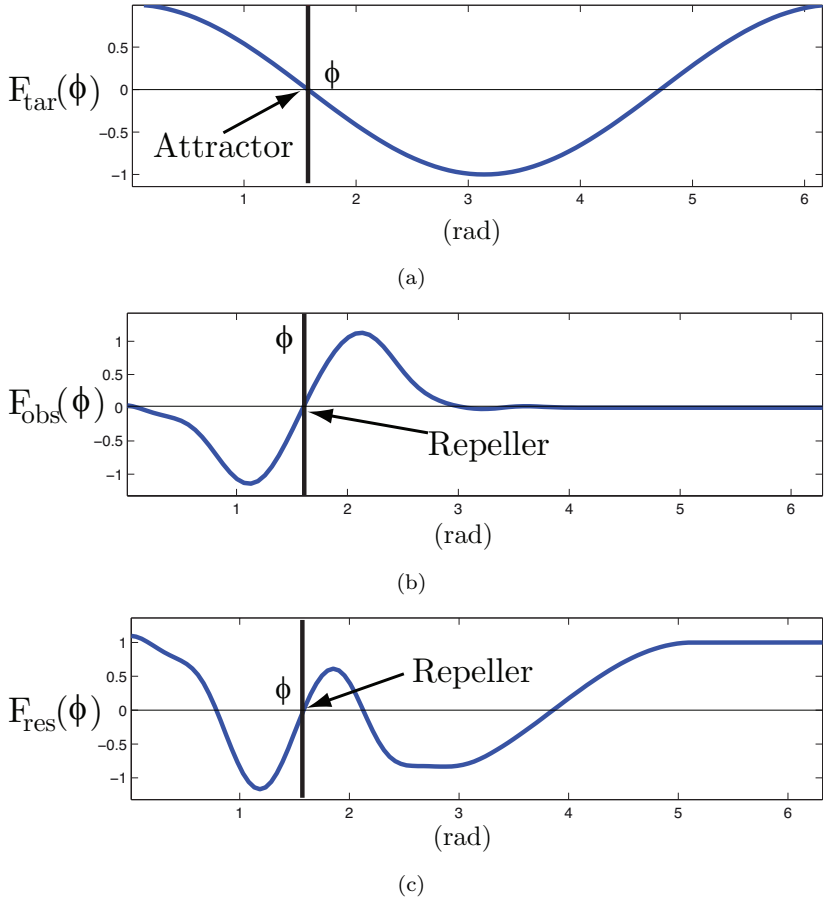


FIGURE 3.24: a) Vector field of the target orientation contribution. b) Vector field of the obstacle avoidance contribution. c) Resulting vector field of the heading direction dynamics.

3.4 Dynamic Approach to Timing Control

This section describes the Timing Control module responsible for generating the robot's linear velocity, v, so that the robot completes its mission under a time constraint.

This module (see fig. 3.25) is developed based on the Stuart-Landau nonlinear oscillator, whose solutions are used to set in a straightforward way the robot's linear velocity, v. The module is composed by four blocks: Velocity, Timing Adaptation, Parameter Modulation and Behavior Switching.

3.4.1 Velocity

The generation of the robot's linear velocity, v, is fundamental to allow the robot to complete its mission in the specified time constraint, MT. A robotic mission is considered successful if the robot moves from its initial position to the final position, within the specified time constraint, MT.

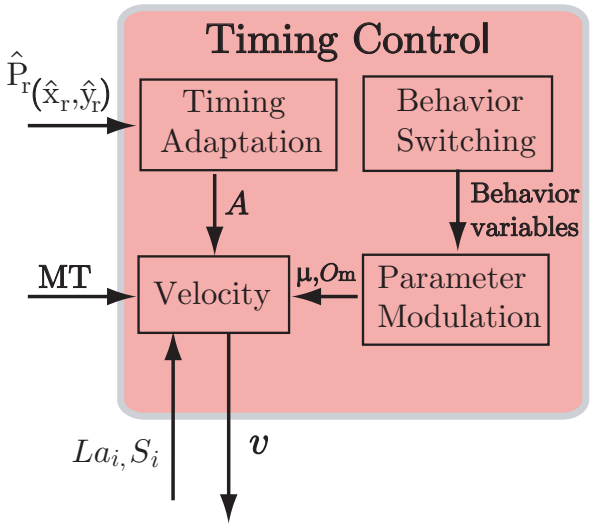


FIGURE 3.25: Schematic of the Timing Control module. It receives as inputs the time constraint, MT, the robot's position, $\hat{P}_r$, and sensor information, La_i (laser) and S_i (sonars). The output is the robot's linear velocity, v.

To generate the robot's linear velocity, the Stuart-Landau nonlinear oscillator whose solution is a single limit cycle (a periodic oscillation with cycle time $T = \frac{2\pi}{\omega}$ and finite amplitude A) is adopted,

$$f_{10} \triangleq m_{k+1} = m_k + \left(\alpha\left(\mu_k - r_k^2\right)\left(m_k - O_{m_k}\right) - \omega_k n_k\right) d_k, \tag{3.35}$$

$$f_{11} \triangleq n_{k+1} = n_k + \left(\alpha\left(\mu_k - r_k^2\right) n_k + \omega_k \left(m_k - O_{m_k}\right)\right) d_k, \tag{3.36}$$

$$r_{k+1} = \sqrt{\left(m_k - O_{m_k}\right)^2 + n_k^2}, \tag{3.37}$$

where m and n are the state variables and μ, O_m, ω and α are control parameters. This oscillator can be written in polar coordinates as follows,

$$r_{k+1} = r_k + r_k \alpha(\mu_k - r_k^2) d_k, \tag{3.38}$$

$$\theta_{k+1} = \theta_k + \omega d_k, \tag{3.39}$$

where r_k and θ_k define the radius and the angle of the oscillations, respectively.

This oscillator was selected, because it is analytically treatable to a large extend and provides a complete control over its states. This benefits the specification of parameters and represents a definitive advantage over other oscillators. In addition, it enables to explicitly modulate the generated solutions according to parameters, while keeping the general features of the original movements. These features are useful for generating stable solutions, applied in robotics [1, 2, 13, 85, 149]. A full description of the Stuart-Landau oscillator can be viewed in appendix A.3.

When the parameters are changed, the oscillator promptly changes the frequency, amplitude and offset of its solutions, resulting in smooth and responsive trajectories. To summarize, the trajectories generated by the Stuart-Landau oscillator can be defined over time as,

$$\begin{bmatrix} m\left(t\right) \\ n\left(t\right) \end{bmatrix} = \begin{cases} \begin{bmatrix} O_{\mathrm{m}} \\ 0 \end{bmatrix}, \mu < 0, \\ \begin{bmatrix} O_{\mathrm{m}} + \sqrt{\mu}\cos\left(\omega t\right) \\ \sqrt{\mu}\sin\left(\omega t\right) \end{bmatrix}, \mu > 0, \end{cases} \tag{3.40}$$

and the oscillator is able to generate different solutions depending on its parameters (see fig.3.26):

1. a discrete movement to a time-varying offset O_{m}, if $\mu < 0$ (Region A);
2. a rhythmic movement around O_{m}, if $\mu > 0$ (Regions B,C and D).

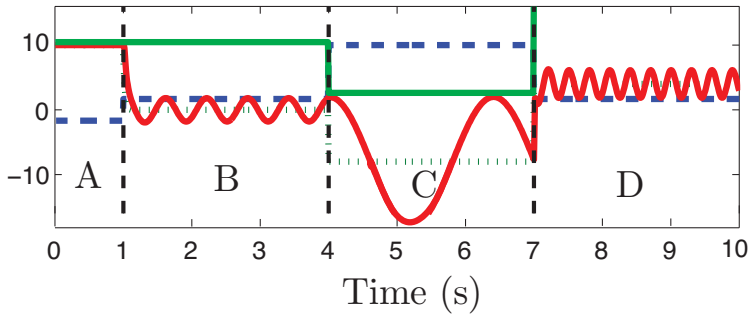


FIGURE 3.26: m trajectory modulation (solid red line) and parameters' role. (A): because $\mu = -2.8^2$ (dashed blue) the oscillatory behavior is turned off due to the Hopf bifurcation, leading to a discrete movement towards the value of $O_{\mathrm{m}} = 10$ (dotted dark green). (B), (C), (D): when $\mu > 0$ a rhythmic movement is obtained, with frequency ω (light green) around O_{m}. (B): $\mu = 2.8^2$, $O_{\mathrm{m}} = 0$ and $\omega = 10\mathrm{rad.s}^{-1}$. (C): $\mu = 10^2$, $O_{\mathrm{m}} = -7$ and $\omega = 3\mathrm{rad.s}^{-1}$. (D): $\mu = 2.8^2$, $O_{\mathrm{m}} = 5$ and $\omega = 15\mathrm{rad.s}^{-1}$.

The state variable m of the Stuart-Landau oscillator was used to directly control the robot's linear velocity, while the state variable n is required to enable the oscillator to undergo periodic motion. Thus, the robot's linear velocity, v, is given as follows,

$$v = m. \tag{3.41}$$

The online adjustment of the robot's linear velocity is performed by tuning the oscillator parameters $(\mu, O_{\mathrm{m}}, \alpha, \omega)$, so that the robot is able to complete its mission within MT.

3.4.2 Timing Adaptation

This section describes how to modulate the limit cycle solution provided by the Stuart-Landau oscillator, in order to generate the required robot's linear velocity profile. When $\mu < 0$, the oscillator (3.35),(3.36) generates a rhythmic solution whose period equals the time constraint $MT = \frac{2\pi}{\omega}$. To control the robot's linear velocity during a mission, only one period of the solution m is required. Fig. 3.27 depicts an example of solution m, wherein one period is highlighted in red.

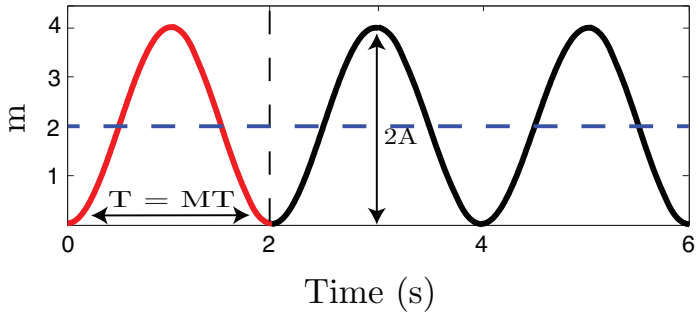


FIGURE 3.27: Solution m of the Stuart-Landau oscillator with T = 2 s, during 6 s, for A = 2 and $O_{\mathrm{m}} = 2$ (blue dashed line). Only one period of the solution m is used, *i.e.* only the red part is generated and used to control the robot's linear velocity.

Considering that $O_{\mathrm{m}} = A$, the time constraint, MT, is the time required by solution m to go from zero to twice the oscillator amplitude and back to zero again, performing a full sinusoidal cycle. Fig. 3.28 (top) depicts an example of solution m with a constant frequency ω during the interval of time $MT = 2$ s. Nonetheless, ascending and descending parts of the oscillator cycle have equal durations, meaning that a large amount of time is spent accelerating towards the maximum required velocity and decelerating back to zero again. This velocity profile can be given as follows,

$$v\left(t\right) = A\left(1 - \cos\left(\omega t\right)\right), \tag{3.42}$$

where A is the amplitude of the oscillator and ω defines the angular frequency.

From a robot physical perspective, it would be beneficial if minor maximum velocities are requested. The velocity profile should be kept approximately constant, as long as possible. As minor maximum velocities are required, less performance of the robot's servos, motor and other physical components is demanded, increasing their lifetime. This velocity profile (fig. 3.28 (bottom)) has shorter acceleration and deceleration times, but more abrupt. Ideally, it should be possible to select the acceleration and deceleration durations for the mission. These intervals of time can be set according to the mechanical features of the robot. In the first interval, $[0, t_1[$, the robot acceler-

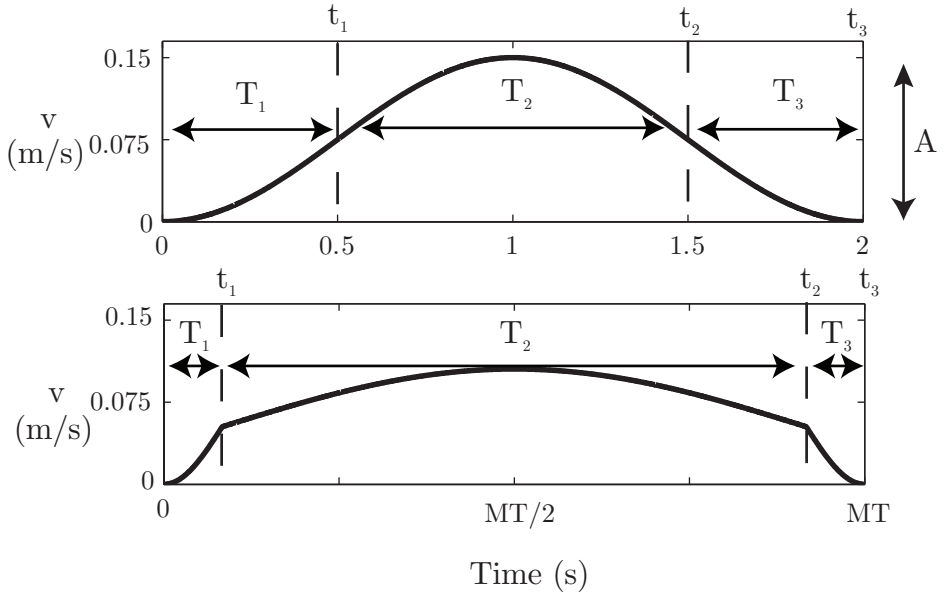


FIGURE 3.28: Top) Single oscillation profile generated by the Stuart-Landau oscillator. On this profile, T_1 and T_3 are longer, resulting in a curve with a higher top velocity. Bottom) Modulated oscillation profile where periods T_1 and T_3 are shorter, resulting in a smaller top velocity.

ates; in the second interval, $[t_1, t_2[$, the robot keeps a velocity approximately constant; in the third time interval, $[t_2, t_3[$, the robot decelerates until stop. Each interval of time can be defined with a different duration, such their sum equals the time constraint, $MT = T_1 + T_2 + T_3$. Note that in both velocity profiles, the robot should cover the same distance for equal intervals of time.

During the first interval of time, $[0, t_1[$, it is considered that the oscillator covers the first quarter of the cycle, (T_1), half of the cycle is covered during the second interval of time, $[t_1, t_2[$, (T_2), and the last quarter of the cycle is covered in the last interval of time, $[t_2, t_3[$, (T_3).

Each interval of time has an angular frequency, $(\omega_1, \omega_2, \omega_3)$, such that, in the overall, the oscillation is performed within the correct timing $\omega = \dfrac{2\pi}{MT}$,

$$\omega_1 = \frac{\pi}{2T_1},\ \omega_2 = \frac{\pi}{T_2 - t_a},\ \omega_3 = \frac{\pi}{2T_3}, \tag{3.43}$$

where t_a is the amount of time in which the condition $m < O_{\mathrm{m}}$ is verified within the interval of time $[t_1, t_2[$. This situation occurs when the robot detects an obstacle and eventually reduces its velocity to safely circumnavigate it. Hence, t_a is defined as

follows,

$$t_a = \sum_{t=0}^{MT} \frac{d_k}{(1+\exp\left[-b(m-O_{\mathrm{m}})\right])\left(\exp\left[-b(t-T_1)\right]\right)\left(\exp\left[b(t-T_1-T_2)\right]\right)}, \tag{3.44}$$

where d_k is the discretization step and $b = 500$ is an empiric constant that defines the speed between transitions. Fig. 3.29 depicts an example in which the initial frequency of the oscillator is ω_1. Then, in the interval of time $[t_1, t_2[$, ω_2 is selected. At approximately $t = 35$ s, the robot's linear velocity is reduced and $m < O_{\mathrm{m}}$. Dashed gray line depicts the solution m and dashed dotted green line depicts the offset O_{m}. To ensure a quick acceleration, ω_1 is again selected. During this period, t_a is being incremented according to (3.44). When $m > O_{\mathrm{m}}$, ω_2 is selected again, but with a different value, since t_a was updated. Finally, in the last interval, ω_3 is selected.

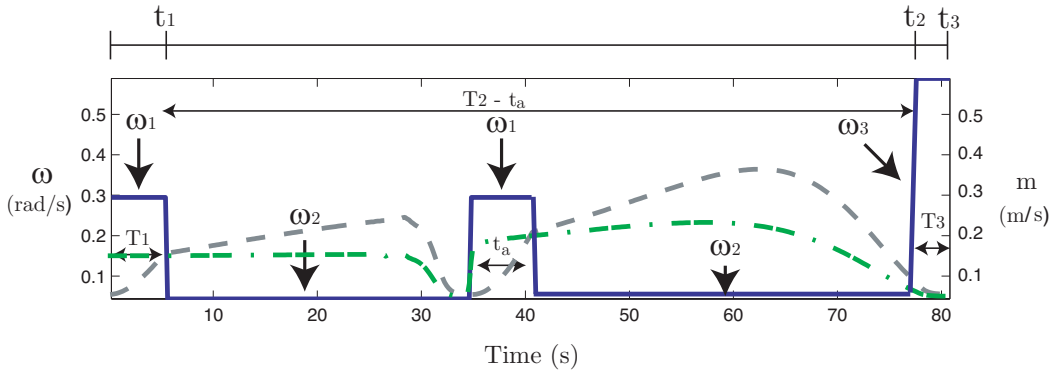


FIGURE 3.29: Blue continuous line depicts the frequency of the oscillator, ω, dashed grey line depicts solution m and dashed dotted green line depicts the oscillator offset, O_{m}.

The distance s covered during each interval of time is calculated by integrating (3.42) for a fixed amplitude, A, during the given intervals of time,

$$s_1 = \int_0^{T_1} A(1-\cos(\omega_1 t))dt = \frac{A\left(\frac{\pi}{2}-1\right)}{\omega_1}, \tag{3.45}$$

$$s_2 = \int_0^{T_2} A(1-\cos(\omega_2 t))dt = \frac{A\left(\pi+2\right)}{\omega_2}, \tag{3.46}$$

$$s_3 = \int_0^{T_3} A(1-\cos(\omega_3 t))dt = \frac{A\left(\frac{\pi}{2}-1\right)}{\omega_3}. \tag{3.47}$$

In an undisturbed scenario, in which no disturbances affect the trajectory of the robot, the total distance covered by the robot is the distance required to reach the

final position, P_{g}, at the beginning of the mission,

$$D\left(0\right) = s_1 + s_2 + s_3, \tag{3.48}$$

$$D\left(0\right) = \frac{A\left(\frac{\pi}{2} - 1\right)}{\omega_1} + \frac{A\left(\pi + 2\right)}{\omega_2} + \frac{A\left(\frac{\pi}{2} - 1\right)}{\omega_3}, \tag{3.49}$$

where $D\left(0\right)$ is the initial distance between the robot's position and the final location, P_{g}.

If no obstacles disturb the trajectory of the robot, the amplitude of the oscillator, A, does not need to be adapted and can be calculated as,

$$A = \frac{D\left(0\right)}{\frac{\left(\frac{\pi}{2}-1\right)}{\omega_1} + \frac{\left(\pi+2\right)}{\omega_2} + \frac{\left(\frac{\pi}{2}-1\right)}{\omega_3}}. \tag{3.50}$$

However, in real environments, disturbances of many types may occur, forcing the robot to change its trajectory. The expected distance in such cases is no longer the distance calculated in the beginning of the mission. Thus, it is necessary to adapt in real-time the amplitude A of the Stuart-Landau oscillator. This adaptation must consider the remaining distance, D, between the robot's position, $\hat{P}_{\mathrm{r}}$, and the goal location, P_{g}. In the following, it is depicted the process to calculate the distance D and amplitude A at each instant of time.

3.4.2.1 Distance Calculation

The distance, D, that the robot has to cover can be used to verify whether the robot is on the desired position, delayed or advanced relatively to the time constraint of the mission.

Consider that $D(0)$ is equal to the sum of the remaining distance D and the distance already covered by the robot, D_c,

$$D(0) = D + D_c. \tag{3.51}$$

In previous works [1, 3, 13, 85, 151], the distance D was calculated through the Euclidean distance between P_{g} and the robot's position, $\hat{P}_{\mathrm{r}}$. However, for complex environments where the robot moves between several corridors and rooms, the Euclidean distance between $\hat{P}_{\mathrm{r}}$ and P_{g} is unfeasible to calculate D.

A three-step iterative algorithm is used to determine the distance that the robot has to cover to reach P_{g}. This algorithm follows the same approach as in appendix A.2,

as the next position of the robot is the projection onto the next critical line. The first iteration consists on calculating the distance d_1 between the robot's initial position, $P_{\mathrm{r}}(0)$, and its projection, $B_{0,1}$, onto the critical line $l_{1,2}$ that divides region r_1 and r_2. The second iteration consists on calculating the distance $d_{i,i+1}$ between all projections onto the respective critical lines, starting by calculating the distance between the first projection, $B_{0,1}$, and the projection of $B_{0,1}$ onto the critical line $l_{2,3}$, $B_{1,2}$. The last iteration consists on calculating the distance $d_{n,g}$ between the last projection $B_{n-1,n-2}$ and the goal location, P_{g},

$$D(0) = d_1 + \sum_{i=2}^{n_a} d_{i,i+1} + d_{n,g}, \tag{3.52}$$

where n_a is total number of regions in the environment that the robot has to traverse to reach the goal location. Note that in cases the robot starts its mission in the same region of the goal location, $D(0) = d_{r,g}$, where $d_{r,g}$ is the distance between the robot's initial position, $P_{\mathrm{r}}(0)$, and P_{g}.

Consider for instance the example of calculating the distance the robot has to cover in fig. 3.30. Initially, the robot is inside region r_1, and the first step is to calculate the distance d_1. Then, the distance between consecutive projections is calculated while the robot does not reach the final region r_{10}, where P_{g} lies. Finally, it is calculated the distance $d_{n,g}$ between the last projection and P_{g}.

At the beginning of the mission, the remaining distance, D, that the robot has to cover is equal to the initial distance, $D(0)$ that the robot has to cover. As the robot moves towards the goal location, the remaining distance D changes as follows,

$$D = D(0) - \left(d_1 + \sum_{i=2}^{n_r} d_{i,i+1} \right) + d_{\mathrm{robot},i+1}, \tag{3.53}$$

where n_r is the current region of the robot and $d_{\mathrm{robot},i+1}$ is the distance between the current robot's position and the next critical line. Note that when the initial region is equal to the final region, $D = d_{\mathrm{robot},g}$, *i.e.*, the distance between $\hat{P}_{\mathrm{r}}$ and P_{g}.

The adaptive rule to calculate the amplitude A is obtained by substituting (3.50) into (3.51) and by integrating (3.42),

for $0 < t < t_1$:

$$A_1 = \frac{D}{\frac{\frac{\pi}{2}-1+\sin(\omega_1 t)}{\omega_1} + \frac{\pi+2}{\omega_2} + \frac{\frac{\pi}{2}-1}{\omega_3} - t}, \tag{3.54}$$

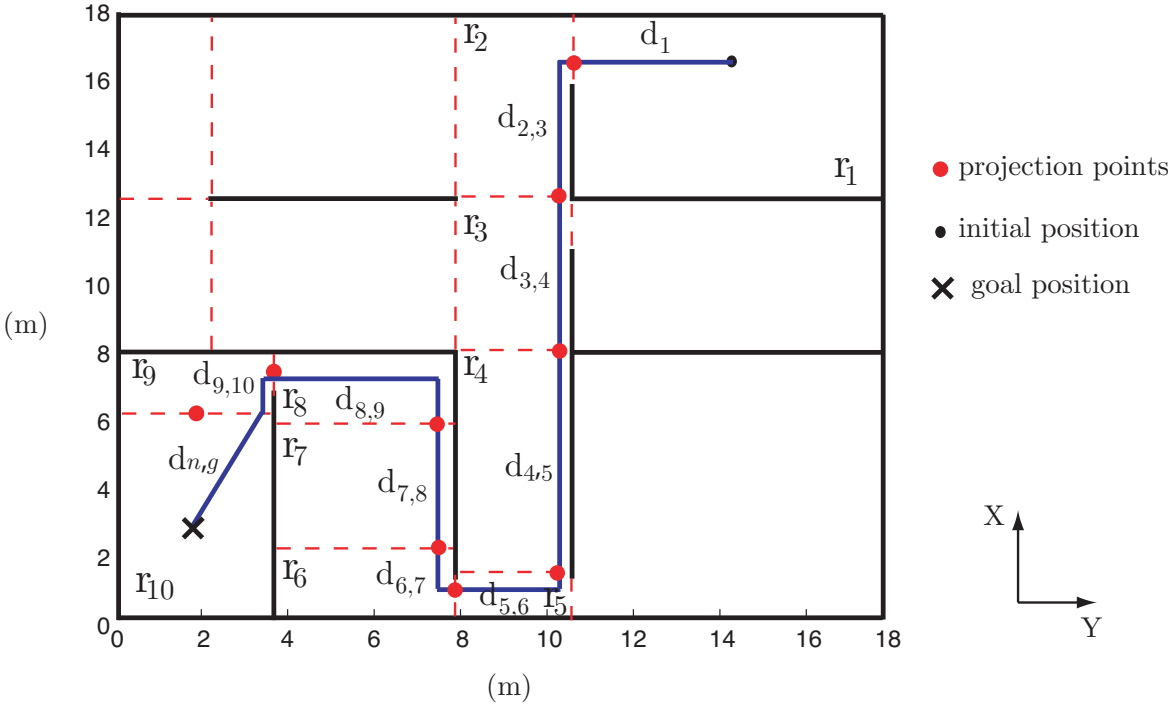


FIGURE 3.30: Example of how to calculate the distance D(0) between the robot's initial position, $P_{\mathrm{r}}(0)$, and the goal location, P_{g}. Blue line represents the calculated path.

for $t_1 < t < t_2$:

$$A_2 = \frac{D}{\frac{\frac{\pi}{2}}{\omega_1} + \frac{\pi+1+\cos(\omega_2(t-T_1))}{\omega_2} + \frac{\frac{\pi}{2}-1}{\omega_3} - t}, \tag{3.55}$$

for $t_2 < t < t_3$:

$$A_3 = \frac{D}{\frac{\frac{\pi}{2}}{\omega_1} + \frac{\pi}{\omega_2} + \frac{\frac{\pi}{2}-\cos(\omega_3(t-T_1-T_2))}{\omega_3} - t}. \tag{3.56}$$

The velocity profile is modulated in amplitude and frequency by simply changing both A and ω parameters, respectively. The idea is to explicitly change these parameters according to the oscillator current state, as follows,

$$\begin{aligned} f_{12} \triangleq A'_k &= \frac{A_{1_k}}{(1+\exp\left[b\left(m_k - O_{\mathrm{m}_k}\right)\right])\left(1+\exp\left[bn_k\right]\right)} + \frac{A_{2_k}}{1+\exp\left[-b\left(m_k - O_{\mathrm{m}_k}\right)\right]} \\ &+ \frac{A_{3_k}}{(1+\exp\left[b\left(m_k - O_{\mathrm{m}_k}\right)\right])\left(1+\exp\left[-bn_k\right]\right)}, \end{aligned} \tag{3.57}$$

where A_1, A_2 and A_3 are defined in (3.54), (3.55), (3.56), respectively. A' is equal to A_1 when m is smaller than O_{m} and n is negative. A' is equal to A_2 always that m is

greater than O_{m}. A' is equal to A_3 when m is smaller than O_{m} and n is positive. b controls the alternation speed between these values.

The same procedure is used for parameter ω as follows,

$$f_{13} \triangleq \omega_k = \frac{\omega_1}{(1+\exp\left[b\left(m_k - O_{\mathrm{m}_k}\right)\right])\left(1+\exp\left[bn_k\right]\right)} + \frac{\omega_2}{1+\exp\left[-b\left(m_k - O_{\mathrm{m}_k}\right)\right]} + \frac{\omega_3}{(1+\exp\left[b\left(m_k - O_{\mathrm{m}_k}\right)\right])\left(1+\exp\left[-bn_k\right]\right)}. \tag{3.58}$$

Both A' and ω values are selected from A_i and ω_i $(i = 1, 2, 3)$ respectively, according to the state of m and n. Fig. 3.31 illustrates an example of a limit cycle where parameters A and ω are selected according to the values of O_{m}, m and n.

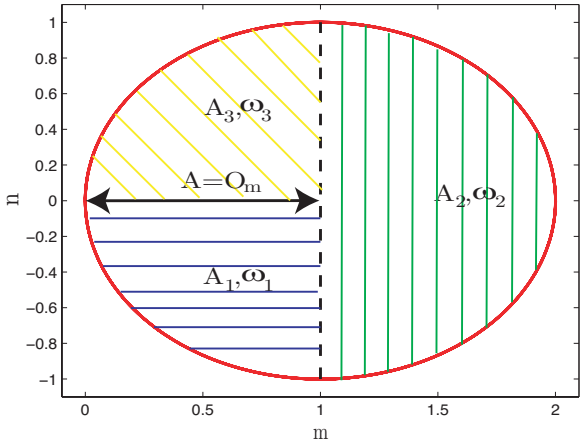


FIGURE 3.31: A is the radius of the Stuart-Landau oscillator (in this example $A = 1$ and $O_{\mathrm{m}} = 1$). It takes different values according to the oscillator state. $A = A_1$ when state variable m is smaller than O_{m} and state variable n is negative. $A = A_2$, if m is greater than O_{m}. $A = A_3$, when n is positive and the value of m is smaller than O_{m}.

3.4.3 Obstacle Profile

During the missions, the robot detects obstacles that may force it to modify its path and cover a larger distance than initially expected. If the robot moves at a high velocity, collisions with obstacles might not be avoided. Due to its physical constraints, *e.g.* inertia, the robot might not be able to decelerate and stop before hitting the obstacle. Thus, the robot should consider its surroundings, namely, the distance to nearby obstacles, in order to adapt its velocity, v. This allows the robot to circumnavigate safely the obstacles or eventually stop in case of eminent collisions.

As the robot moves in the environment, the distance to obstacles and the angular range subtended by them vary, such that their locations change relative to the robot's

position. The same happens to the direction of the goal location in the environment. Herein, the attractor (direction to reach the goal location) gradually shifts in the space of heading direction. If the relaxation rate of the attractor movement is sufficiently slow compared to the relaxation rate of the timing dynamics, the system relaxes to the new attractor position before it moves again. This is achieved by setting the relaxation rate of the heading dynamics much faster than the relaxation rate of the attractor movement. The difference in the relaxation rates makes it possible to treat the attractor (ψ_{tar}) as a constant value from the viewpoint of the heading dynamics. In such case, the heading variable, ϕ, is in or near the resulting attractor of the dynamical system responsible for the heading direction control.

The maximal shift rate of the fixed points, $\dot{\psi}_{\max}$, is a function of the robot's linear velocity, v, and the minimum distance between the robot and an obstacle, $\min\{La_i, S_i\}$, (see [74, 289] for a full discussion),

$$\dot{\psi}_{\max} \approx \frac{v}{\min\{La_i, S_i\}}. \tag{3.59}$$

The function $\min\{La_i, S_i\}$ can be approximated by a smooth function as follows,

$$\min\{La_i, S_i\} \approx \frac{1}{2}\left(La_i + S_i - \sqrt{(La_i + S_i)^2}\right). \tag{3.60}$$

This maximal rate is obtained when the obstacle is seen sideways and it is assumed constant, such that the system is able to track the moving attractor. The robot's linear velocity, v, should be decreased when the minimum distance to an obstacle decreases. This is achieved by adapting the amplitude of the oscillator, A. Empirically, the following condition is suitable to reduce the robot's linear velocity,

$$f_{14} \triangleq A_k = A'_k\left(\frac{1}{1+\exp\left[-bU_k(\phi_k)\right]} + \frac{d\left(\frac{1}{2}\left(La_i + S_i - \sqrt{(La_i + S_i)^2}\right)\right)^c}{1+\exp\left[bU_k(\phi_k)\right]}\right). \tag{3.61}$$

If no obstacles are detected, $U(\phi) \geq 0$, a reduction of the amplitude is not required and $A = A'$. If obstacles are detected, $U(\phi) < 0$, the amplitude A is reduced according to (3.61), which depends on the minimum distance to any obstacle detected by the laser or sonars. d and c are control parameters empirically selected, such that the robot reduces its velocity when obstacles are close to it.

3.4.4 Behavior Switching

The Behavior Switching block is responsible for specifying the motor behavior of the robot. The robot performs three different motor behaviors: *stop*, when no movement is generated and the robot is stopped; *execution*, when timed movement is generated; and *rescue*, an escape behavior responsible for dealing with situations in which the goal location, P_{g}, cannot be reached by the robot within the time constraint, MT. Furthermore, the switch between these behaviors should be autonomously elicited according to sensory information. The selected behavior is achieved through a nonlinear dynamical system that reproduces a competition among variables. This dynamical system was already used in previous works [74, 290]. This dynamical competition was applied because it provides stability against bounded perturbations in the input signals and the environment can easily elicited an autonomous switch. Furthermore, the concordance in terms of dynamical systems is preserved. Each motor behavior contributes with a value to the vector field. However, only one motor behavior should be active at each instant of time, while the others are disabled.

3.4.4.1 Competitive Dynamics

Each motor behavior of the robot is represented by a behavioral variable u_{i} (i = *stop, execution, rescue*). The competitive dynamics used to represent the competition among these variables is formulated as follows,

$$f_{15} \triangleq u_{i_{k+1}} = u_{i_k} + \frac{\left(\beta_i u_{i_k} - \mid \beta_i \mid u_{i_k}^3 - \nu \sum_{a \neq i} u_{a_k}^2 u_{i_k} + F_{\mathrm{stoch}}\right)}{\alpha_u} d_k, \tag{3.62}$$

where variables u_{i} can go "on" ($u_i = \pm 1$) or "off" ($u_i = 0$) and $\tau_u = \frac{1}{\alpha_u}$ defines the relaxation rate. These dynamics enforce competition among behaviors depending on parameter β_i. The variable u_i with the highest competitive advantage, $\beta_i > 0$, is likely to win the competition. Note that $|\beta_i|$ is a non-smooth function, which can be approximated by the smooth function $\sqrt{\beta_i^2 + \eta}$. The smaller the η, the more accurate is the approximation.

The first two terms of the competitive dynamics represent the normal form of a degenerate *pitch-fork* bifurcation (see fig. 3.32) (a). The third term containing parameter ν ensures the competition among variables u_i and destabilizes any attractors in which more than one variable u_i is active.

Each variable u_i (i = *stop*, *execution* and *rescue*) converges to a solution of the competitive dynamics. A detailed analysis in [290] calculated the fixed points of this competitive dynamics, $u_{\mathrm{fp1}} = 0$, $u_{\mathrm{fp2}} = 1$ and $u_{\mathrm{fp3}} = -1$. The absolute value of u_{fp} is

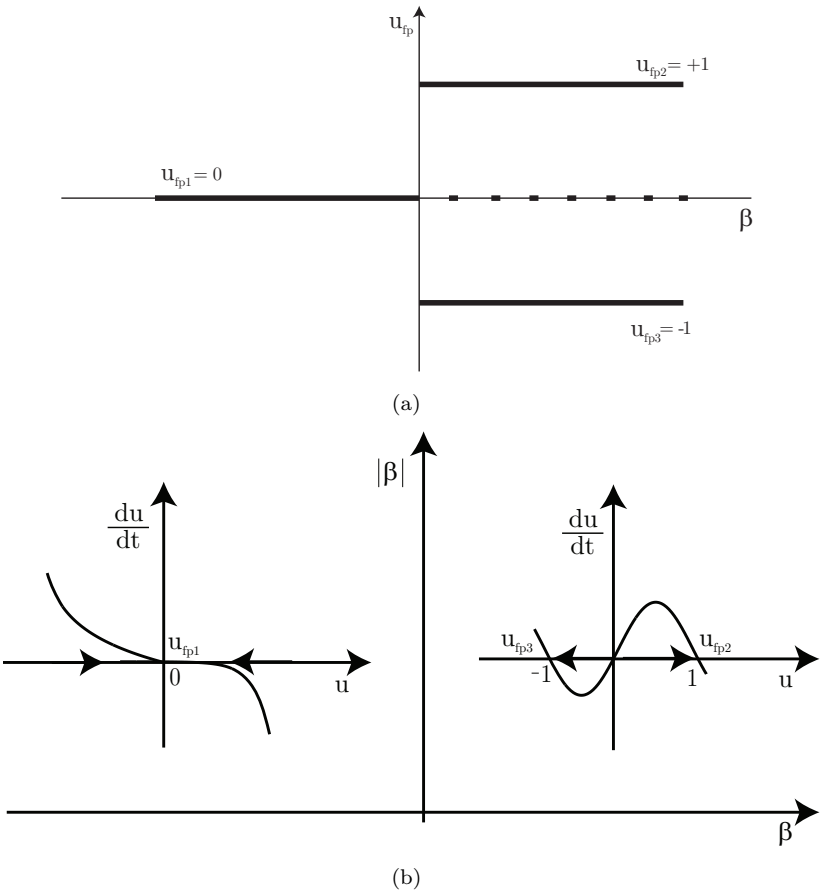


FIGURE 3.32: a) Bifurcation diagram of the competitive dynamics without the competitive term, $\nu = 0$. b) Phase space for the competitive dynamics without the competitive term, $\nu = 0$. For $\beta < 0$ there is a fixed point at $u_{\text{fp1}} = 0$. For $\beta > 0$, this fixed point becomes unstable and two new fixed points appear, $u_{\text{fp2,3}} \pm 1$.

adopted, so that both u_{fp2} and u_{fp3} are equivalent to the "on" state, while $u_{\text{fp1}} = 0$ is equivalent to the "off" state. The activation of the fixed points depends on β_i (see fig. 3.32) (b). Fixed point $u_{\text{fp1}} = 0$ is the single attractor for $\beta_i < 0$. It becomes unstable for $\beta_i > 0$ and two new attractors appear at $u_{\text{fp2}} = 1$ and $u_{\text{fp3}} = -1$.

Considering the competitive term, ν, a new set of possible solutions arise. A detailed stability analysis of each possible solution is provided in [290]. They demonstrated that the stability of the fixed points depends on parameters β_i and ν. Considering that a variable u_i is active and the other variables u_j are disabled, then the condition $\nu > \beta_j,\ \forall j \neq i$ must be true. The active variable u_i has the highest β_i and inhibits the other variables. Parameters $\beta_{\text{stop}}, \beta_{\text{execution}}, \beta_{\text{rescue}}$ and ν are defined to ensure that only one variable u_i and consequent motor behavior is active.

3.4.4.2 Motor Behavior Activation

Functional forms for parameters β_i and ν are designed so that the competitive dynamics bifurcates suitably for the different types of behavior. While the robot is performing a mission, its state (moving or stopped) changes and parameters β_i should reflect these changes. β_i is empirically defined within the interval, $1.5 \leq \beta_i \leq 3.5$, and the offset of this interval is 2.5. If $\nu > 2.5$, it is guaranteed that only one variable u_i is active (see [291] for a detailed explanation). Otherwise, if $\nu < 2.5$, multiple variables u_i may be active. It was defined that $\nu = 3$, in order to guarantee the existence of a single motor behavior.

Parameter β_i varies between 1.5 and 3.5, as follows,

$$\beta_i = 1.5 + 2b_i, \quad (i = stop, execution, rescue), \tag{3.63}$$

where b_i are "quasi-boolean" variables that alternate between 0 and 1. The motor behavior switching is generated by converting sensory conditions and constraints into variables b_i (see [291] for a description and [1, 9, 149] for examples).

Variables b_i are mathematically expressed as follows,

$$b_{\text{stop}} = \frac{\tanh\left(b\left((1 - t_{\text{start}}) + (t_{\text{start}})(t_{\text{reached}})\right)^2 - 0.5\right) + 1}{2}, \tag{3.64}$$

$$b_{\text{execution}} = \frac{\tanh\left(b((t_{\text{start}})(1 - t_{\text{reached}})(g_{\text{reachable}}))^2 - 0.5\right) + 1}{2}, \tag{3.65}$$

$$b_{\text{rescue}} = \frac{\tanh\left(b((t_{\text{start}})(1 - g_{\text{reachable}}))^2 - 0.5\right) + 1}{2}, \tag{3.66}$$

where t_{reached}, t_{start} and $g_{\text{reachable}}$ are mathematically defined as sigmoid functions. t_{reached} returns 1 when the robot reaches the neighborhood of the goal location $B(P_{\text{g}}, \epsilon)$, and 0 otherwise,

$$t_{\text{reached}} = \frac{\tanh(b(-D + \epsilon)) + 1}{2}. \tag{3.67}$$

t_{init} is a period of time that the robot must wait before starting its timed movement. During this period, the robot rotates towards the local goal, P_{b}, and only starts moving after t_{init} has been elapsed. t_{start} returns 1 when $t > t_{\text{init}}$ and 0 when $t < t_{\text{init}}$,

$$t_{\text{start}} = \frac{\tanh(b(t - t_{\text{init}})) + 1}{2}. \tag{3.68}$$

The goal location, P_{g}, is considered reachable if the robot is able to complete its mission within MT. Mathematically, this condition is verified as follows,

$$g_{\mathrm{reachable}} = \frac{\tanh((b(\frac{D}{v_{\max}} - (MT + t_{\mathrm{init}} - t))) + 1)}{2}, \tag{3.69}$$

where parameter $g_{\mathrm{reachable}}$ returns 1 if P_{g} is reachable, and 0 if it is unreachable by the robot within MT. $v_{\max}$ defines the maximum velocity reached by the robot and t is the elapsed time since the beginning of the mission.

3.4.5 Parameter Modulation

The Parameter Modulation block is responsible for defining the topological type (fixed point or limit cycle behavior) of the solutions generated by the Stuart-Landau oscillator. In addition, this block defines the offset O_{m} according to the variable u_i received by the Behavior Switching module.

Small parameter changes in the Stuart-Landau oscillator modulate the generated trajectories with respect to their amplitude, frequency and offset, in order to achieve the desired robot behavior. Therefore, the oscillator parameters have to be set according to their roles in the final modulation.

Different triplets of variables u_i, $(u_{\mathrm{stop}}, u_{\mathrm{execution}}, u_{\mathrm{rescue}})$, lead to different robot behaviors, namely: no movement, timed movement and movement with constant velocity. Each triplet must be mapped onto different values for the set of parameters. To activate each motor behavior (stop, execution and rescue), the triplets of variables u_i are defined as,

- "stop" $\rightarrow (u_{\mathrm{stop}}, u_{\mathrm{execution}}, u_{\mathrm{rescue}}) = (\pm 1, 0, 0)$,
- "execution" $\rightarrow (u_{\mathrm{stop}}, u_{\mathrm{execution}}, u_{\mathrm{rescue}}) = (0, \pm 1, 0)$,
- "rescue" $\rightarrow (u_{\mathrm{stop}}, u_{\mathrm{execution}}, u_{\mathrm{rescue}}) = (0, 0, \pm 1)$.

3.4.5.1 Offset Modulation

The offset of solution m of the Stuart-Landau oscillator, O_{m}, is modulated according to the values of variables u_i as follows,

$$f_{16} \triangleq O_{\mathrm{m}_k} = \tanh\left(b\, u_{\mathrm{stop}_k}\right) O_{\mathrm{s}} + \tanh\left(b\, u_{\mathrm{execution}_k}\right) O_{\mathrm{e}} + \tanh\left(b\, u_{\mathrm{rescue}_k}\right) O_{\mathrm{r}}, \tag{3.70}$$

where $O_{\mathrm{s}}, O_{\mathrm{e}}$ and O_{r} are respectively the offsets when the robot is stopped, executing timed movement or moving with a constant velocity. The offset values are set according to desired behaviors: $O_{\mathrm{s}} = 0$, $O_{\mathrm{e}} = A$ and $O_{\mathrm{r}} = 0.1$.

When variable u_{stop} is active, the robot stops. When variable $u_{\mathrm{execution}}$ is active, the offset O_{m} is set according to the necessity of the robot to accelerate or decelerate its velocity, indicated by parameter A. When variable u_{rescue} is activated, the timing nature of the mission is no longer important and the robot is unable to finish the mission within MT. Thus, this variable sets the oscillator offset with a constant value, such that the robot may reach the goal location P_{g} with a constant velocity, $v = 0.1$ m/s.

Fig. 3.33 illustrates how the different triplets of variables u_i, $(u_{\mathrm{stop}}, u_{\mathrm{execution}}, u_{\mathrm{rescue}})$, modulate the offset parameter O_{m}.

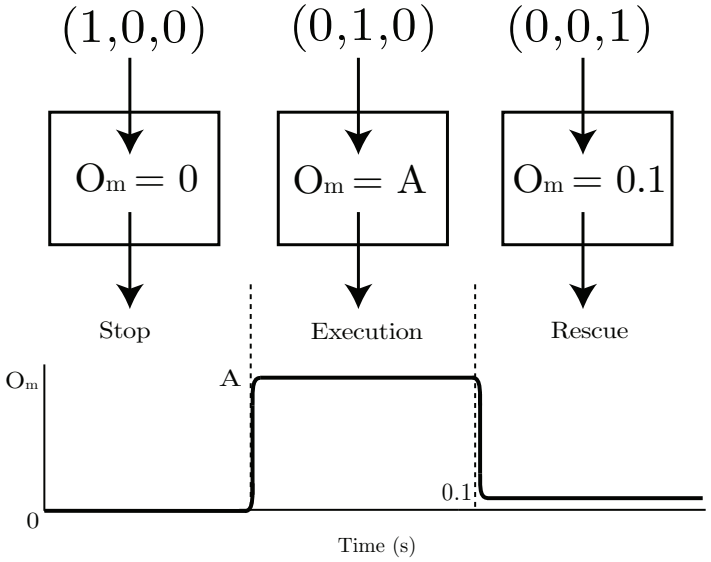


FIGURE 3.33: When the triplet of variables u_i is (1,0,0), u_{stop} is "on" and $O_{\mathrm{m}} = 0$. When the triplet is (0,1,0), $u_{\mathrm{execution}}$ is "on" and $O_{\mathrm{m}} = A$. Finally, when the triplet is (0,0,1), u_{rescue} is "on" and $O_{\mathrm{m}} = 0.1$.

3.4.5.2 Qualitative Behavior

By modifying parameter μ, the system switches between the fixed point at $(m, n) = (O_{\mathrm{m}}, 0)$ (for $\mu < 0$) and the limit cycle (for $\mu > 0$). In addition, μ controls the amplitude of the oscillations, and therefore the amplitude of the robot's linear velocity, v. The value of μ is set according to the competitive dynamics as follows,

$$f_{17} \triangleq \mu_k = -\tanh\left(b\,(u_{\mathrm{stop}_k} + u_{\mathrm{rescue}_k})\right)\frac{A_k^2}{2} + \tanh\left(b\,u_{\mathrm{execution}_k}\right)A_k^2, \tag{3.71}$$

When u_{stop} or u_{rescue} are "on", μ is negative $\left(\mu = -\frac{A^2}{2}\right)$ and the robot's linear velocity is constant (0 m/s in case of u_{stop} is "on" or 0.1 m/s in case u_{rescue} is "on"). When $u_{\text{execution}}$ is "on", μ is positive ($\mu = A^2$), and the timing movement is generated by the Stuart-Landau oscillator.

Fig. 3.34 illustrates how different triplets of variables u_i ($u_{\text{stop}}, u_{\text{execution}}, u_{\text{rescue}}$) modulate the parameter μ.

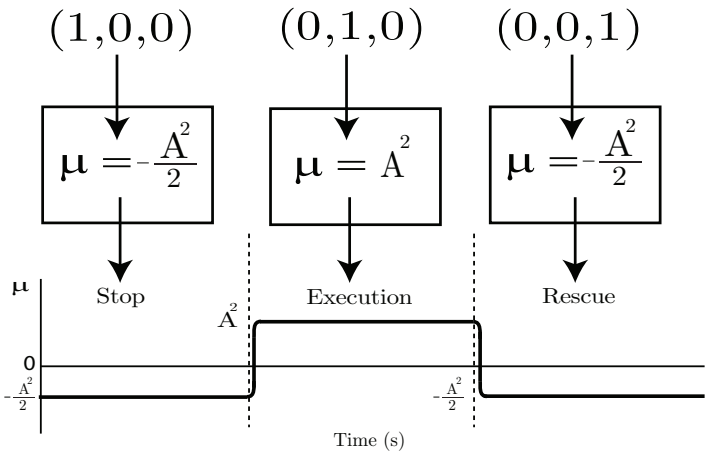


FIGURE 3.34: When u_{stop} or u_{rescue} are "on", (1,0,0) and (0,0,1), respectively, μ parameter is negative, $\mu = -\frac{A^2}{2}$, and the generated trajectory will be constant. On the other hand, when $u_{\text{execution}}$ is "on" (0,1,0), μ parameter is positive, $\mu = A^2$, and the generated trajectory will be timed.

Fig. 3.35 illustrates the process of generating the trajectory for the robot according to the different triplets of variables u_i ($u_{\text{stop}}, u_{\text{execution}}, u_{\text{rescue}}$) that modulate both A and μ parameters.

3.4.6 Adiabatic Elimination

In architectures with coupled dynamical systems, a common solution for approximating the dynamics of the system by eliminating irrelevant coupled levels is adiabatic elimination. This solution consists on providing a hierarchy of relaxation times, such that the faster dynamics are treated as parameters in the slower ones [291]. Conversely, from the viewpoint of the slower dynamics, the faster dynamics can be assumed to have already relaxed to their corresponding fixed points.

The smallest relaxation rate of the global system is defined by the integration cycle, $d_k \approx 0.024$ s, which defines the discretization step. This step defines a lower bound to the relaxation rates of the dynamical systems. Also, the highest relaxation rate of the dynamical systems is upper bounded by the time constraint, MT. Therefore, the relaxation rates of the dynamical systems should be higher than d_k, and lower

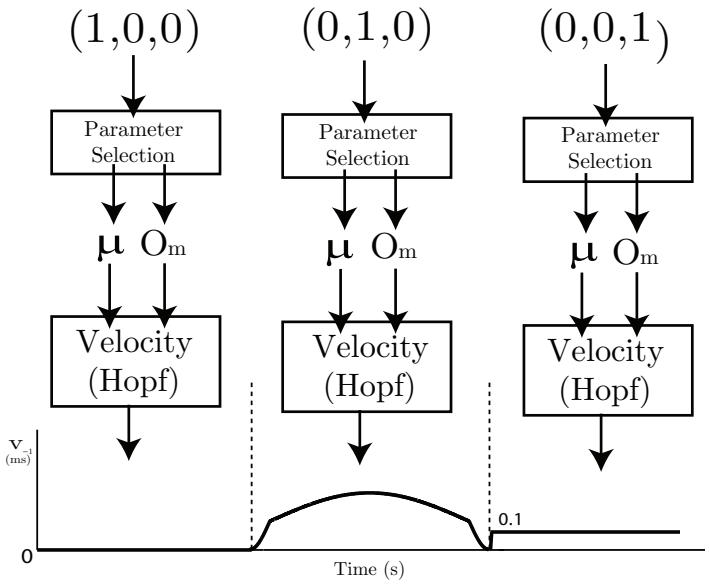


FIGURE 3.35: When the triplet is (1,0,0), u_{stop} is active and the robot does not move, because the generated velocity is 0 m/s. When the triplet is (0,1,0), $u_{\text{execution}}$ is active and the robot performs the timed movement according to the task constraints. Finally, when the robot cannot reach the goal location within the time constraint, the triplet is (0,0,1), and u_{rescue} is active. In this case, the robot will move towards to the goal location with a constant velocity of 0.1 m/s.

than MT. Note that the problem of defining the relaxation rates is not a conceptual problem, but rather a limitation imposed by implementation considerations.

In the architecture modules, the following relaxation rates are identified: τ_{tar}, τ_{obs} and τ_{sobs} on the Local Control, τ_{tr} on the Motion Control, τ_u and $\frac{1}{2\alpha\mu}$ on the Timing Control.

For relaxation purposes, the rate $\frac{1}{2\alpha\mu}$ is not important since, no other relaxation rates depend on it. The competitive dynamics should be defined as the fastest ones, since they are used as parameters in the other dynamics. Consequently, its relaxation rate, τ_u, must be the smallest one.

As defined in section 3.3.6, conditions $\sum_{i=1}^{N_l} \lambda_{\text{obs},i} > \lambda_{\text{tar}}$ and $\sum_{i=1}^{N_s} \lambda_{\text{sobs},i} > \lambda_{\text{tar}}$ should be verified. Consequently, the relaxation rate of the target orientation contribution, τ_{tar} should verify the following conditions, $\tau_{\text{tar}} > \tau_{\text{obs},i}$ and $\tau_{\text{tar}} > \tau_{\text{sobs},i}$.

The goal point P_{b} should have already converged to the desired value when the robot's linear velocity, v, and the robot's heading direction, ϕ changes. P_{b} is considered as a parameter for the calculus of ϕ. Thus, the relaxation rate τ_{tr} must be lesser than the largest relaxation rate of the heading dynamics, τ_{tar}. Velocity depends on the distance to the goal location, P_{g}, which in turn depends on the robot's heading direction, ϕ. Thus, the largest relaxation rate τ_{tar} must be lesser than the time constraint

MT.

The following hierarchy of relaxation rates ensures that the outputs of the faster dynamical systems are viewed as parameters to the slower ones and the obstacle avoidance contribution is stronger than the target orientation one.

$$d_k << \tau_u, \max\left\{\tau_{\text{obs},i}, \tau_{\text{sobs},i}\right\} < \tau_{\text{tar}}, \tau_{\text{tr}} << \tau_{\text{tar}}, \tau_{\text{tar}} << MT. \tag{3.72}$$

Chapter 4

Fundamentals for Stability and Success of the Global System

Contraction Mapping Theory is a framework commonly used in stability analysis [11]. This theory addresses the existence and uniqueness of a single stable equilibrium state. If a nonlinear dynamical system is contracting, its initial conditions and temporary disturbances are forgotten exponentially fast, *i.e.*, the trajectories of the nonlinear system return to their equilibrium state with an exponential convergence rate.

The unique equilibrium state of a contracting system depends smoothly on the contraction and simultaneously presents exponentially convergence [11]. Moreover, this equilibrium state is stable under bounded perturbations, and this is an important robustness property for control architectures (see Propositions 2.2.20 and 2.6.14 in [292] for perturbations on fixed points).

This chapter begins with mathematical concepts about Contraction Mapping Theory, namely the contraction principle and the definition of contraction. The chapter continues with the construction of a stability indicator that is identified with the ability of the robot to complete the mission with success. This condition follows from the combination property of the Contraction Mapping Theory. The chapter is concluded with a stability analysis for each module of the architecture based on the Contraction Mapping Theory. This analysis provides stability conditions used as guidelines to design the architecture.

4.1 Contraction Mapping Theory

Contraction Mapping Theory states that if a map $f(x_i) = x_{i+1} : X \mapsto X$ is contracting with respect to an Euclidean space X, then f converges exponentially to its

unique solution, x_0, called fixed point, *i.e.*, $f(x_0) = x_0$ (see the contraction principle in proposition 2.2.10 [292]). A map f is contracting (see definition 2.2.1 in [292] if the following property is verified,

$$d(f^n(x), f^n(y)) < \lambda^n d(x, y) \quad \wedge \quad \lambda \in [0, 1[, \ \forall x, y \in X, \tag{4.1}$$

where $n \in \mathbb{N}$[1] and d [2] stands for the Euclidean distance with Cartesian coordinates.

A contraction brings every two points x and y in X closer together. In particular, for every $x \in X$, and any $r > 0$, all points y in the ball $B(x, r)$, are mapped into a ball $B(Tx, s)$, with $s < r$ (see fig. 4.1).

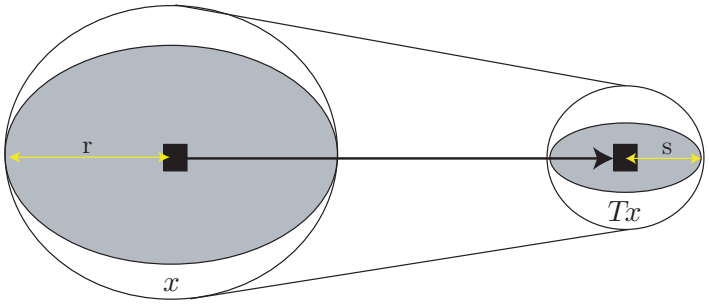


FIGURE 4.1: T is contracting. r is the radius of ball $B(x, r)$ and s is the radius of ball $B(Tx, s)$.

The contraction property of a map f in the Euclidean space can be extended to a multi-variable analysis through the derivative of f. Let $C \subset \mathbb{R}^n$ be an open strictly convex set and $\bar{C}$ its closure, then according to theorem 2.2.16 in [292], f is contracting and has a unique fixed point $x_0 \in \bar{C}$ for every $x \in \bar{C}$ if $f : \bar{C} \mapsto \mathbb{R}^n$ is differentiable on C, continuous on $\bar{C}$ and

$$\|Df\| \leq \lambda < 1, \tag{4.2}$$

where D refers to the Jacobian matrix and $\|\cdot\|$ refers to the Euclidean norm, defined as $\sqrt{\lambda_{\max}(f^T f)} = \sigma_{\max}(f)$, and $\lambda_{\max}$ the maximum eigenvalue, f^T stands for the transpose of f and $\sigma_{\max}$ stands for the maximum singular value.

The Contraction Theory can be applied under weaker hypotheses according to definition 2.6.11 [292], in which a map $f(x_i) = x_{i+1}$ is eventually contracting if there is a constant $G > 0$, such that,

$$d(f^n(x), f^n(y)) < G\lambda^n d(x, y) \quad \wedge \quad \lambda \in [0, 1[, \ \forall x, y \in X, \tag{4.3}$$

[1] $\mathbb{N}$ is the set of integers. This follows the same notation as in [292].

[2] $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$.

for all $n \in \mathbb{N}$.

Condition (4.2) can be rewritten for an eventually contracting map as follows,

$$\|Df\| \leq G\lambda. \tag{4.4}$$

The definition of eventually contracting maps is important to analyze the stability of global systems, since there might be dynamical subsystems of the resulting dynamical system in the global system that are not contracting in the entire space X, but converge to a unique fixed point in a part of X. Thus, the eventually contracting dynamical system obeys to condition (4.2) in that part of space X, over which it converges to the unique fixed point.

4.2 Stability and Mission Success

When a map $f(x_i)$ represents a closed loop process, such as the global system in fig. 3.1, the stability of its unique fixed point can be identified with the mission success.

Definition 4.1. The mission is considered successful if the resulting dynamical system in the global system, which may include contracting and eventually contracting dynamical systems, is contracting and thus converge to the unique fixed point of the global system $f_{\text{supervisor}} \circ f_{\text{robot}} \circ K$ (see fig. 3.1) [3].

In practical terms, the stability of the global system means that the mission is successfully completed, *e.g.*, the robot reaches the goal location P_{g} within the time constraint, MT.

The robot is deployed in an indoor environment with a topology allowing the successful execution of its missions. In a sense, it is assumed that the environment is neutral. Neutrality can be defined in multiple forms. A simple form is to identify the map K with the identity map. A less restrictive form is that of a isomorphism. Since a isomorphism between metric spaces preserves distances at the input to the output [293], this means that the environment does not scale the distances at the input. Hence, if a scaling is required, it can be accommodated by $f_{\text{supervisor}}$ (see below). Consequently, the global system is simplified and might be analyzed through the composition $f_{\text{supervisor}} \circ f_{\text{robot}}$ or $f_{\text{robot}} \circ f_{\text{supervisor}}$.

[3] Symbol $\circ$ stands for function composition.

Theorem 4.2. *For a generic robot for which there is no information on its structure, assume that,*

$$||D(f_{\text{supervisor}})|| << 1. \tag{4.5}$$

Then (4.5) means that it can be safely assumed as a sufficient condition to ensure the contraction of the global system and hence the existence of a fixed point, independently of the model of the robot.

Proof. Considering that O is an Euclidean space, the global system composed by $f_{\text{supervisor}} : O \mapsto X$ and $f_{\text{robot}} : X \mapsto O$ (see fig. 3.1), under the aforementioned assumptions on differentiability and environment neutrality, is considered contracting if it verifies condition (4.2),

$$\|D(f_{\text{robot}} \circ f_{\text{supervisor}})\| < 1. \tag{4.6}$$

Using the differentiation rule, the left-hand term in (4.6) can be written as (assuming norm compatibility, see for instance [294]) ,

$$\|D(f_{\text{robot}} \circ f_{\text{supervisor}})\| \leq \|D\left(f_{\text{robot}}\right) \circ f_{\text{supervisor}}\| \, \|D(f_{\text{supervisor}})\|. \tag{4.7}$$

Using (4.7) and (4.6) and adopting a conservative design approach (4.5), yields,

$$\|D\left(f_{\text{robot}}\right) \circ f_{\text{supervisor}}\| \, \|D(f_{\text{supervisor}})\| < 1. \tag{4.8}$$

This ensures the contraction of the global system and hence the existence of a fixed point, independently of the model of the robot. □

Theorem 4.2 is a weak condition for mission success as it does not consider the information of the robot in f_{robot}. Nevertheless, it represents an important design tool due to its generality to a wide range of robotic applications.

Contracting systems offer desirable combination properties [12]. The contraction of a mesh of nonlinear systems is preserved under many types of combinations. According to [295], the knowledge of the internal organization of the mesh is not required to establish the contraction of the architecture. If the nonlinear systems are connected to each other, *e.g.* through a parallel, feedback or hierarchical combination, the global system remains contracting if each individual dynamical system is contracting or eventually contracting [12].

The global system can be represented by a sequence of f_i modules (see fig. 3.2) and $\|D(f_{\text{supervisor}})\|$ is given as follows,

$$\begin{aligned} ||D(f_{\text{supervisor}})|| &= ||D(f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17})|| \\ &\leq \prod_{i=1}^{17} \|D(f_i)\|. \end{aligned} \tag{4.9}$$

The combination property can be extended to the global system composed by a mesh of nonlinear dynamical systems and feed-through maps, f_i, if the following conditions are verified: (i) the dynamical systems and feed-through maps are connected (parallel, feedback or hierarchical combination); (ii) the dynamical systems are contracting or eventually contracting; (iii) the Jacobians of the feed-through maps are upper bounded; and (iv) the Jacobian norms of the dynamical systems when composed with the feed-through maps (that can have Jacobians with upper bounds greater than 1) result in a value below 1. Otherwise, the contraction of the global system is not verified and the mission may not be completed with success.

It must be noted that bounds such as (4.9) do not embed information on the interconnection among the feed-through maps and nonlinear dynamical systems. In a sense, this corresponds to assuming that each f_i block is meaningfully interconnected to others such that $f_{\text{supervisor}}$ will work towards completing the mission within MT.

In stability terms, bounding the time the robot has to complete the mission requires that the global system converges to the fixed point fast enough. For instance, a dynamical system with initial condition q_0 and final condition q_f is said to verify a time constraint MT if,

$$\forall \epsilon > 0,\ \exists u(t)\ :\ q(t) \in B(q_f, \epsilon),\ t \geq MT \tag{4.10}$$

where $u(t)$ are command inputs of the system and $B(q_f, \epsilon)$ is a ball of radius ϵ centered at q_f. However, $||D(f_{\text{supervisor}})||$ given by (4.9) does not include any block concerning the verification of the time constraint. Including a block concerning the verification of the time constraint into $||D(f_{\text{supervisor}})||$ can be achieved by several C^1 feed-through maps, provided that they verify some conditions. They must return approximately 1 when the mission is in time, $t \leq MT$, and a value sufficiently large to set $||D(f_{\text{supervisor}})|| >> 1$ when the mission is not finished in MT. Consequently, condition (4.5) no longer holds and the mission can be considered failed, as the fixed point of the global system is no longer stable. One possible C^1 feed-through map, which verifies if the mission is completed within the time constraint can be described

as follows,

$$f_{\text{timing}} = 1 + \exp\left[-b(MT - t)\right], \tag{4.11}$$

where b defines the rate of transition and is suitably selected. The same representation as in (4.9) is used to include the Jacobian of the map f_{timing} into the architecture,

$$||D(f_{\text{supervisor}})|| \;\; \leq \;\; \|D(f_{\text{timing}})\| \prod_{i=1}^{17} \|D(f_i)\|, \tag{4.12}$$

where the right-hand side is the upper bound of $||D(f_{\text{supervisor}})||$ and is denoted as $||\overline{D(f_{\text{supervisor}})}||$.

The stability indicator, $\|D(f_{\text{supervisor}})\|$, or more generally its upper bound can be used as feedback on the global system (see fig. 4.2), such that information of the current state of the mission (a continuum between failure or success) is used to steer actions on the robot. For instance, the robot can follow at a constant velocity to the goal location in cases that the mission is failed or follow its mission in cases the mission is considered successful. Block H receives the upper bound of the stability indicator $||\overline{D(f_{\text{supervisor}})}_k||$ and produces commands, h_{k+1}, that can be used in the architecture, $f_{\text{supervisor}}$. Block H must have into account properties such that it does not disturb the contraction or eventual contraction of the global system.

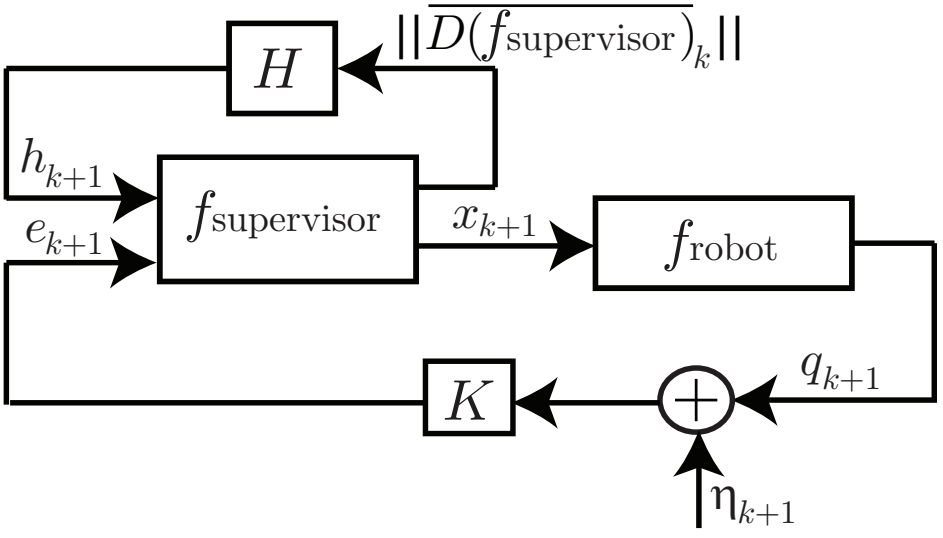


FIGURE 4.2: Graphical representation of the global system with feedback of the upper bound of the stability indicator, $\|\overline{D(f_{\text{supervisor}})}_k\|$, into block M.

This stability indicator, $\|D(f_{\text{supervisor}})\|$, can also be used to estimate at a given instant of time $t < MT$, if the mission will be completed with success.

Theorem 4.3. *The mission of the robot is completed with success if,*

$$\exists\ t_1 < MT,\ \forall t \in\ ]t_1, MT] : ||D(f_{\text{supervisor}})|| << 1, \quad \text{subject to } \eta, \tag{4.13}$$

where t_1 stands for an instant of time and η is a bounded perturbation.

Proof. If the expected perturbations of the global system, η, verify the neutrality of the environment, then the success of the mission only depends on the ability of the global system to converge to a unique solution. This is verified if condition $||D(f_{\text{supervisor}})|| << 1$ is verified at least after a certain point t_1, even if during other instants of time, only the eventually contracting is verified, $||D(f_{\text{supervisor}})|| \leq G\lambda$, with $G\lambda$ approximately 1. However, for $t \in\ ]t_1, MT]$, an unbounded perturbation η, could lead the global system to diverge from the unique fixed point and the mission may be considered failed. □

The Timing Control module already embeds f_i blocks that adjust the velocity and make $\|D(f_{\text{supervisor}})\|$ to increase. However, this increase may not be large enough, leading to a small $\|D(f_{\text{supervisor}})\|$ (a typical example is a saturation). In such cases, it is necessary to use a performance timing index, such as f_{timing}, which observes the system in greater detail.

Including additional constraints in the architecture can be easily done using the above principle. This shows the usefulness of Contraction Theory, in particular the combination property used in (4.9) and in (4.12), to analyze the stability of complex real-time control architectures.

The global system can be assumed as a hybrid system, *i.e.*, contains continuous-time and discrete-time elements. Sufficient conditions to extend the combination property to hybrid systems can be found in theorem 4 in [296]. Hybrid systems are said to be contracting if and only if their dynamics (continuous and discrete) are uniformly exponentially stable.

4.3 Stability Analysis

The internal organization of the global system is depicted in fig. 4.3, where rectangular shapes stand for dynamical systems whereas triangles represent feed-through maps.

The behavior of each block shown in fig. 3.2 can be written more compactly as [4],

[4]Right-hand term variables in (4.14), (4.15), (4.16) and (4.17) are already depicted in chapter 3.

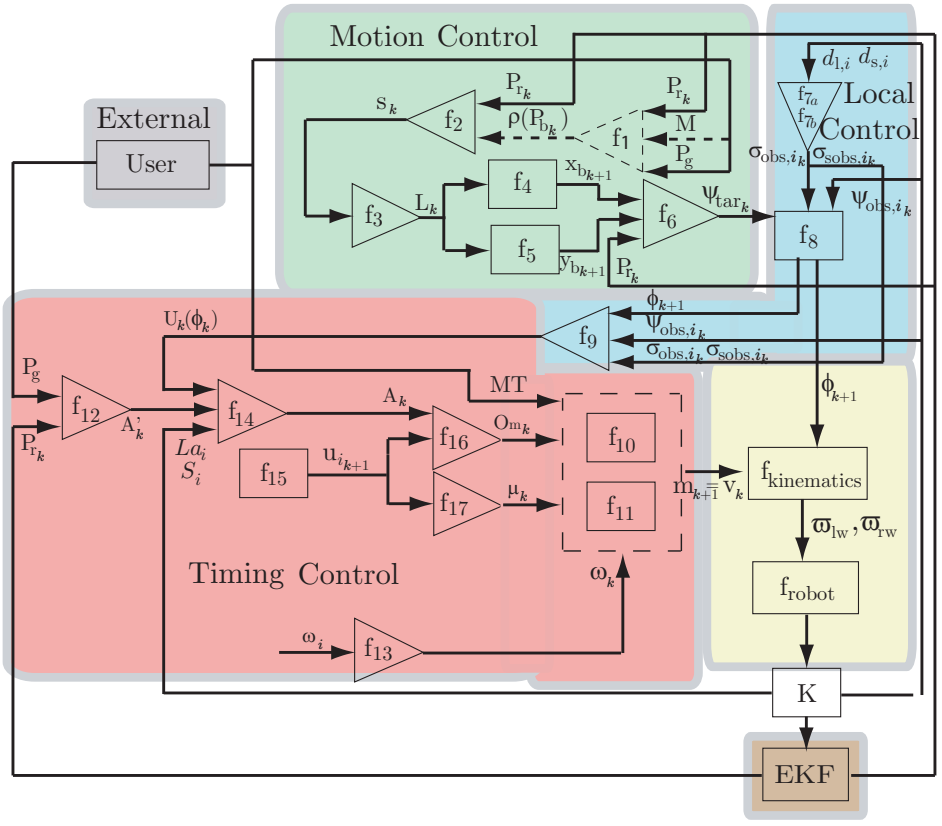


FIGURE 4.3: Block diagram representing the global system, wherein only variables exchanged between blocks are represented. Block f_1 is dashed represented because the mission begins only when ρ (sequence of regions) is provided to block f_2.

Motion Control

$$
\begin{aligned}
f_1(M, P_{\mathrm{r}_k}, P_{\mathrm{g}}) &= \rho \\
f_2(P_{\mathrm{r}_k}, P_{\mathrm{b}_k}) &= s_k \\
f_3(s_k, l_{i,i+1}, l_{i+1,i+2}) &= L_k \\
f_4(x_{\mathrm{b}_k}, L_k, \lambda_{\mathrm{tr}}) &= x_{\mathrm{b}_{k+1}} \\
f_5(y_{\mathrm{b}_k}, L_k, \lambda_{\mathrm{tr}}) &= y_{\mathrm{b}_{k+1}} \\
f_6(P_{\mathrm{r}_k}, P_{\mathrm{b}_k}) &= \psi_{\mathrm{tar}_k}
\end{aligned}
\tag{4.14}
$$

where the i in f_3 is a region index. Block f_1 returns a set of indexes identifying the sequence of regions to be traversed before the robot starts its mission. f_2 is a map that constantly checks in which region the robot is, and it can be constructed as a smooth interpolation between the pairs of points P_{1i}, P_{2i}.

Local Control

$$\begin{aligned} f_{7_a}(R_{\text{robot}}, d_{\text{l},i}) &= \sigma_{\text{obs},i_k} \\ f_{7_b}(R_{\text{robot}}, d_{\text{s},i}) &= \sigma_{\text{sobs},i_k} \\ f_8(\phi_k, \psi_{\text{obs},i_k}, \sigma_{\text{obs},i_k}, \sigma_{\text{sobs},i_k}) &= \phi_{k+1} \\ f_9(\phi_{k+1}, \psi_{\text{obs},i_k}, \sigma_{\text{obs},i_k}, \sigma_{\text{sobs},i_k}) &= U_k(\phi_k) \end{aligned} \tag{4.15}$$

where i identifies the index of each sensor used to measure the environment.

Timing Control

$$\begin{aligned} f_{10}(m_k, n_k, O_{\text{m}_k}, \mu_k, \alpha_k, \omega_k) &= m_{k+1} \\ f_{11}(m_k, n_k, O_{\text{m}_k}, \mu_k, \alpha_k, \omega_k) &= n_{k+1} \\ f_{12}(L_{\text{a}}, d, c, A'_k) &= A'_k \\ f_{13}(\omega_i, m_k, n_k, O_{\text{m}_k}, b), \forall\ i \in \{1,2,3\} &= \omega_k \\ f_{14}(A_{i_k}, A'_k, m_k, n_k, O_{\text{m}_k}, b, D_k), \forall\ i \in \{1,2,3\} &= A_k \\ f_{15}(u_{i_k}, \beta_i, \nu), \forall\ i \in \{1,2,3\} &= u_{i_{k+1}} \\ f_{16}(u_{i_{k+1}}, O_{\text{s}}, O_{\text{e}}, O_{\text{r}}), \forall\ i \in \{1,2,3\} &= O_{\text{m}_k} \\ f_{17}(u_{i_{k+1}}, A_k), \forall\ i \in \{1,2,3\} &= \mu_k \end{aligned} \tag{4.16}$$

where $i \in \{1,2,3\}$ encodes the robot's motor behavior (*stop*, *execution*, *rescue*).

Robot Module

$$\begin{aligned} f_{\text{robot}}(q_k, \omega_{\text{lw}_k}, \omega_{\text{rw}_k}) &= q_{k+1} \\ K(q_k) &= \left(\psi_{\text{obs},i_k}, P_{\text{r}_k}, La_i, S_i, d_{\text{l},i}, d_{\text{s},i}\right). \end{aligned} \tag{4.17}$$

4.3.1 Motion Control - Contraction Analysis

Motion Control module is composed by a series composition of direct feed-through maps (f_2, f_3, f_6) and dynamical systems (f_4, f_5). This module is contracting if the dynamical systems converge to a unique fixed point (contracting if verify condition (4.2), or eventually contracting if verify condition (4.4)) and the Jacobians of the feed-through maps are upper bounded.

The Jacobian of dynamical systems f_4, f_5 are as follows,

$$D(f_4) = \frac{\partial f_4}{\partial x_{\mathrm{b}_k}} = 1 + \lambda_{\mathrm{tr}} d_k, \tag{4.18}$$

$$D(f_5) = \frac{\partial f_5}{\partial y_{\mathrm{b}_k}} = 1 + \lambda_{\mathrm{tr}} d_k. \tag{4.19}$$

Verifying if $D(f_4)$ and $D(f_5)$ validate (4.2) yields,

$$\|1 + \lambda_{\mathrm{tr}} d_k\| < 1, \tag{4.20}$$

and this condition is true when,

$$-2 < \lambda_{\mathrm{tr}} d_k < 0. \tag{4.21}$$

Assuming sufficient small d_k, (4.21) holds for $\lambda_{\mathrm{tr}} < 0$. Fig. 4.4 (a) illustrates Jacobians $D(f_4)$ and $D(f_5)$ when $\lambda_{\mathrm{tr}} < 0$. Clearly, both Jacobians are below 1. Thus, f_4 and f_5 are contracting maps and converge to the unique fixed point (see fig. 4.4 (b)).

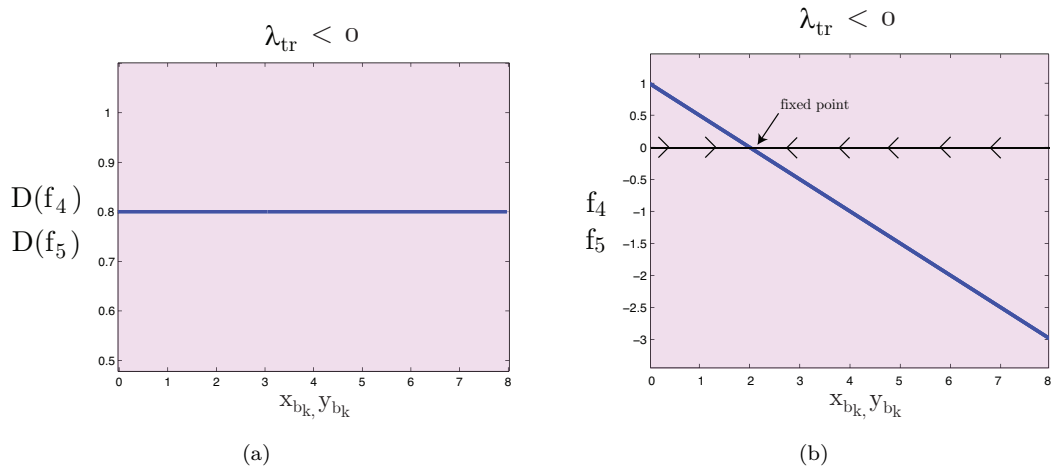


FIGURE 4.4: Phase portrait of $D(f_4)$ ($D(f_5)$) and f_4 (f_5) for $\lambda_{\mathrm{tr}} < 0$.

Feed-through maps f_2 and f_3 are continuous in $\mathbb{R}$ through a suitable choice of parameter b (see (3.16) and (3.17)). Thus, their Jacobians, $D(f_2)$ and $D(f_3)$ are upper bounded. Feed-through map f_6 is continuous in $\mathbb{R}$ (see (3.20)) and consequently $D(f_6)$ is upper bounded.

4.3.2 Local Control - Contraction Analysis

The Local Control module composed by block (f_7, f_8, f_9) is contracting if the dynamical system f_8 converges to a unique fixed point (f_8 is contracting or eventually contracting), and the Jacobians of feed-through maps f_7 and f_9 are upper bounded. Note that f_7 is composed by f_{7_a} and f_{7_b}. Nevertheless, their stability analysis is similar, and thus considered only as a f_7 block.

The Jacobian of f_8 is defined as,

$$\begin{aligned} D(f_8) = \frac{\partial f_8}{\partial \phi_k} &= 1 + \left(\lambda_{\text{tar}} \cos\left(\phi_k - \psi_{\text{tar}}\right) + \sum_{i=1}^{N_l} \lambda_{\text{obs},i_k} \exp\left[-\frac{a^2}{2\sigma^2_{\text{obs},i_k}} \right] \left(1 - \frac{a^2}{\sigma^2_{\text{obs},i_k}} \right) \right. \\ &+ \left. \sum_{i=1}^{N_s} \lambda_{\text{sobs},i_k} \exp\left[-\frac{a_1^2}{2\sigma^2_{\text{sobs},i_k}} \right] \left(1 - \frac{a_1^2}{\sigma^2_{\text{sobs},i_k}} \right) \right) d_k, \end{aligned} \tag{4.22}$$

where $a = (\phi_k - \psi_{\text{obs},i})$ and $a_1 = (\phi_k - \psi_{\text{sobs},i})$. If $\|D(f_8)\| < 1$, f_8 is contracting. Verifying if (4.22) validates (4.2),

$$\begin{aligned} \left\| 1 + \left(\lambda_{\text{tar}} \cos\left(\phi_k - \psi_{\text{tar}}\right) + \sum_{i=1}^{N_l} \lambda_{\text{obs},i_k} \exp\left[-\frac{a^2}{2\sigma^2_{\text{obs},i_k}} \right] \left(1 - \frac{a^2}{\sigma^2_{\text{obs},i_k}} \right) \right. \right. \\ \left. \left. + \sum_{i=1}^{N_s} \lambda_{\text{sobs},i_k} \exp\left[-\frac{a_1^2}{2\sigma^2_{\text{sobs},i_k}} \right] \left(1 - \frac{a_1^2}{\sigma^2_{\text{sobs},i_k}} \right) \right) d_k \right\| < 1, \end{aligned} \tag{4.23}$$

which is true if the following condition is verified,

$$\begin{aligned} -2 < \left(\lambda_{\text{tar}} \cos\left(\phi_k - \psi_{\text{tar}}\right) + \sum_{i=1}^{N_l} \lambda_{\text{obs},i_k} \exp\left[-\frac{a^2}{2\sigma^2_{\text{obs},i_k}} \right] \left(1 - \frac{a^2}{\sigma^2_{\text{obs},i_k}} \right) \right. \\ \left. + \sum_{i=1}^{N_s} \lambda_{\text{sobs},i_k} \exp\left[-\frac{a_1^2}{2\sigma^2_{\text{sobs},i_k}} \right] \left(1 - \frac{a_1^2}{\sigma^2_{\text{sobs},i_k}} \right) \right) d_k < 0 \end{aligned} \tag{4.24}$$

Considering the situation where obstacles are not detected, ($\lambda_{\text{obs},i_k} = 0,\ i = 1, ..., N_l$ and $\lambda_{\text{sobs},i_k} = 0,\ i = 1, ..., N_s$), condition (4.24) can be rewritten as,

$$-2 < \left(\lambda_{\text{tar}} \cos\left(\phi_k - \psi_{\text{tar}}\right) \right) d_k < 0. \tag{4.25}$$

For the target attraction behavior, ϕ_k should converge to ψ_{tar} and this is verified when $\lambda_{\text{tar}} < 0$. Thus, assuming that d_k is sufficient small and $\lambda_{\text{tar}} < 0$, condition (4.25) is verified during the interval: $2\pi n - \frac{\pi}{2} < \phi_k - \psi_{\text{tar}} < 2\pi n + \frac{\pi}{2},\ n \in \mathbb{N}$. Thus, in these intervals, condition (4.2) is verified. This is illustrated in fig. 4.5 (a) by the shadow areas.

However, during intervals: $\frac{-3\pi}{2} + 2\pi n < \phi_k - \psi_{\text{tar}} < -\frac{\pi}{2} + 2\pi n$ and $\frac{\pi}{2} + 2\pi n < \phi_k - \psi_{\text{tar}} < \frac{3\pi}{2} + 2\pi n$ (white areas), (4.25) is not verified. In these intervals, $D(f_8)$ verifies (4.4). Thus, f_8 is eventually contracting, since $D(f_8) < G\lambda$ with $G = \|\lambda_{\text{tar}}\| d_k$ and $\lambda \in [0, 1[$. However, note that the repulsive fixed points, $\phi_k - \psi_{\text{tar}} + \pi n$ and $\phi_k - \psi_{\text{tar}} - \pi n$, push the solution to the fixed point $\phi_k - \psi_{\text{tar}}$ in such intervals (see fig. 4.5 (b)). Thus, considering the interval $-\pi < \phi_k - \psi_{\text{tar}} < \pi$, f_8 maintains the contraction principle and converges to the unique fixed point ψ_{tar}. Different solutions will arise in other intervals, but they can be mapped to the interval $]0, 2\pi[$.

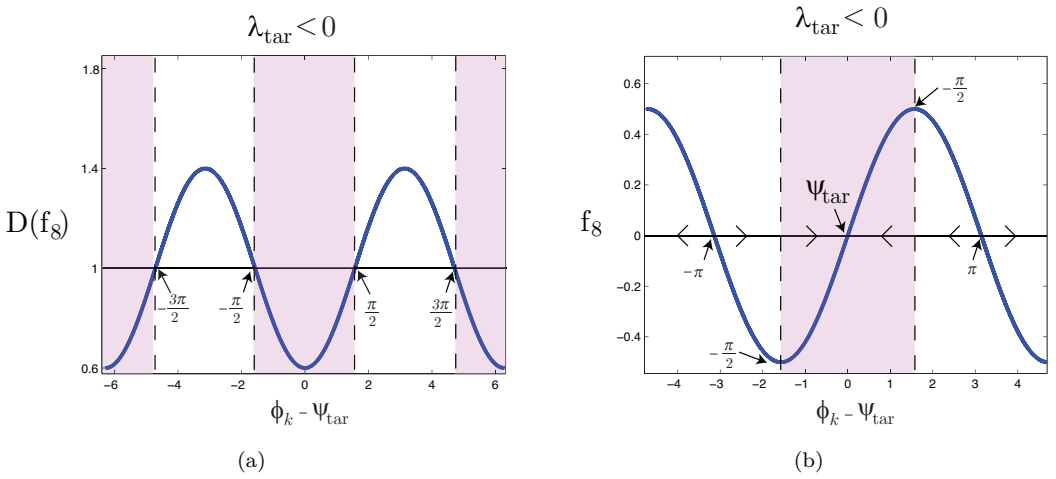


FIGURE 4.5: Phase portrait of $D(f_8)$ and f_8 for $\lambda_{\text{tar}} < 0$.

In cases where obstacles are detected, $\lambda_{\text{obs},i_k} \neq 0\ \forall i$ or $\lambda_{\text{sobs},i_k} \neq 0\ \forall i$, the robot must avoid the direction where they are located. The second and third terms of (4.24) are responsible for moving away the robot from the direction where obstacles are located. This is achieved by setting $\lambda_{\text{obs},i_k} > 0,\ \forall k, i = 1, ..., N_l$ and $\lambda_{\text{sobs},i_k} > 0,\ \forall k, i = 1, ..., N_s$. Since the first term of (4.24) was already verified to be eventually contracting, the other terms are contracting if the following condition is verified,

$$\begin{aligned} -2 \ < \ & \left(\sum_{i=1}^{N_l} \lambda_{\text{obs},i_k} \exp\left[-\frac{a^2}{2\sigma^2_{\text{obs},i_k}} \right] \left(1 - \frac{a^2}{\sigma^2_{\text{obs},i_k}} \right) \right. \\ + \ & \left. \sum_{i=1}^{N_s} \lambda_{\text{sobs},i_k} \exp\left[-\frac{a_1^2}{2\sigma^2_{\text{sobs},i_k}} \right] \left(1 - \frac{a_1^2}{\sigma^2_{\text{sobs},i_k}} \right) \right) d_k < 0, \end{aligned} \tag{4.26}$$

which is true if the following sufficient conditions are verified,

$$\begin{aligned} a \ &> \pm\sigma_{\text{obs},i_k}, \quad \forall k,\ i = 1, \dots N_l \\ a_1 \ &> \pm\sigma_{\text{sobs},i_k}, \quad \forall k,\ i = 1, \dots N_s \end{aligned}$$

$$\left(\sum_{i=1}^{N_l}\lambda_{\text{obs},i_k}\exp\left[-\frac{a^2}{2\sigma^2_{\text{obs},i_k}}\right]\left(1-\frac{a^2}{\sigma^2_{\text{obs},i_k}}\right)\right.$$
$$\left.+\sum_{i=1}^{N_s}\lambda_{\text{sobs},i_k}\exp\left[-\frac{a_1^2}{2\sigma^2_{\text{sobs},i_k}}\right]\left(1-\frac{a_1^2}{\sigma^2_{\text{sobs},i_k}}\right)\right)d_k>-2. \tag{4.27}$$

Assuming that d_k is sufficient small, $\lambda_{\text{tar}}<0$, $\lambda_{\text{obs},i_k}>0$, $\lambda_{\text{sobs},i_k}>0$, $a>\pm\sigma_{\text{obs},i_k}$ and $a_1>\sigma_{\text{sobs},i_k}$, f_8 is contracting if (4.27) is verified. However, (4.27) is not verified in the interval $\psi_{\text{obs},i}-\sigma_{\text{obs},i_k}<\phi_k-\psi_{\text{obs},i}<\psi_{\text{obs},i}+\sigma_{\text{obs},i_k}$. Nevertheless, f_8 is eventually contracting since $D(f_8)$ verifies (4.4), $D(f_8)<G\lambda$ with $G=\sum_{i=1}^{N_l}\lambda_{\text{obs},i_k}+\sum_{i=1}^{N_s}\lambda_{\text{sobs},i_k}$. The repulsive fixed point $\psi_{\text{obs},i}$ in the interval $\psi_{\text{obs},i}-\sigma_{\text{obs},i_k}<\phi_k-\psi_{\text{obs},i}<\psi_{\text{obs},i}+\sigma_{\text{obs},i_k}$ pushes the solution f_8 to the unique fixed point ψ_{tar} (see fig. 4.6 (b)). Therefore, when obstacles are detected, f_8 is eventually contracting but maintains the contraction principle and converges to the unique fixed point ψ_{tar} in the interval $0<\phi_k-\psi_{\text{obs},i}<2\pi$.

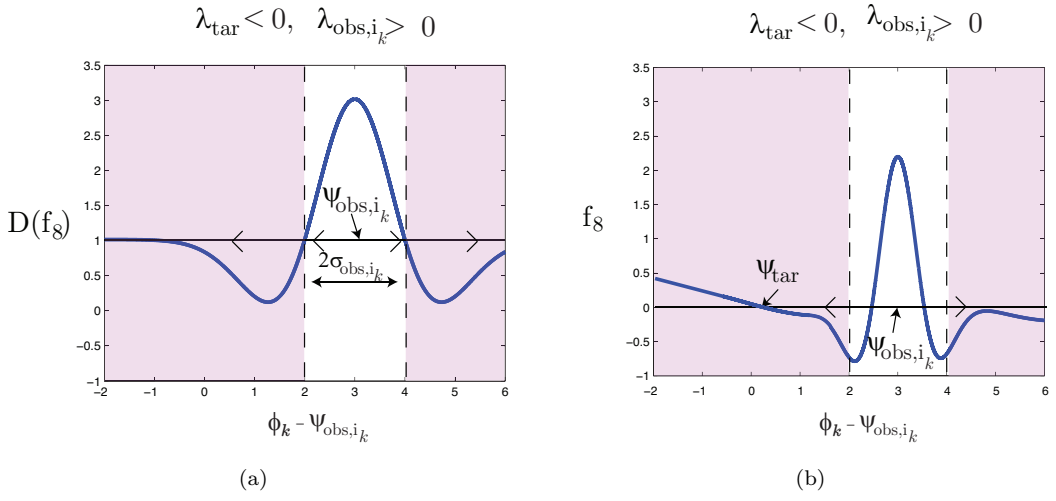


FIGURE 4.6: Phase portrait of $D(f_8)$ and f_8 for $\lambda_{\text{obs},i_k}>0$ and $\lambda_{\text{tar}}<0$.

Feed-through maps f_7 (including f_{7_a} and f_{7_b}) (see (3.26)-(3.27)) are continuous in $\mathbb{R}$. Thus, their Jacobians are upper bounded. Note that $\tan\left(\frac{\Delta\theta}{2}\right)$ and $\tan\left(\frac{\Delta\theta_s}{2}\right)$ are replaced by constant values. Feed-through map f_9 see (3.32)) is also continuous in $\mathbb{R}$. Consequently, its Jacobian, $D(f_9)$, is upper bounded.

4.3.3 Timing Control - Contraction Analysis

Conditions for contraction of the Timing Control module require that the dynamical systems (f_{10}, f_{11}) and f_{15} converge to a unique fixed point (contracting or eventually contracting), and the set of maps $\|D(f_{12}, f_{13}, f_{14},\ f_{16}, f_{17})\|$ is upper bounded.

Dynamical system f_{15} is contracting if its Jacobian verifies (4.2), ($\|D(f_{15})\| < 1$),

$$D(f_{15}) = \frac{\partial f_{15}}{\partial u_{i_k}} = 1 + \left(\frac{\beta_i - 3|\beta_i| u_{i_k}^2 - \nu \sum_{a \neq i} u_{a_k}^2}{\alpha_u} \right) d_k. \tag{4.28}$$

Verifying if (4.28) validates (4.2),

$$\left\| 1 + \left(\frac{\beta_i - 3|\beta_i| u_{i_k}^2 - \nu \sum_{a \neq i} u_{a_k}^2}{\alpha_u} \right) d_k \right\| < 1, \tag{4.29}$$

which is true if the following condition is verified,

$$-2 < \left(\frac{\beta_i - 3|\beta_i| u_{i_k}^2 - \nu \sum_{a \neq i} u_{a_k}^2}{\alpha_u} \right) d_k < 0. \tag{4.30}$$

For $\beta_i > 0$ and $\alpha_u > 0$, f_{15} is contracting if the following conditions are verified,

$$\|u_{i_k}\| > \left\| \sqrt{\frac{1}{3} - \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|, \tag{4.31}$$

$$\|u_{i_k}\| < \left\| \sqrt{\frac{2\alpha_u}{3\beta_i d_k} + \frac{1}{3} - \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|, \tag{4.32}$$

which are true if $\beta_i > \nu \sum_{a \neq i} u_{i_k}^2$ and $\frac{2\alpha_u}{3\beta_i d_k} + \frac{1}{3} > \frac{\nu \sum a \neq i u_{i_k}^2}{3\beta_i}$, respectively.

For $\beta_i < 0$, f_{15} is contracting if the following conditions are verified,

$$\|u_{i_k}\| > \left\| \sqrt{\frac{1}{3} + \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|, \tag{4.33}$$

$$\|u_{i_k}\| < \left\| \sqrt{\frac{2\alpha_u}{3\beta_i d_k} + \frac{1}{3} - \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|, \tag{4.34}$$

which are true if $\nu > 0$ and $\frac{2\alpha_u}{3\beta_i d_k} + \frac{1}{3} > \frac{\nu \sum a \neq i u_{a_k}^2}{3\beta_i}$.

For visual purposes, fig. 4.7 depicts the phase plane of $D(f_{15})$ and f_{15}, respectively for $\beta_i > 0$ and $\beta_i < 0$. For $\beta_i > 0$, condition (4.31) is illustrated in figs. 4.7 (a) (c) by vertical dashed lines. However, note that during the interval $\|u_{i_k}\| < \left\| \sqrt{\frac{1}{3} - \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|$, condition (4.31) is not verified. Consequently f_{15} is not a contraction when $\beta_i > 0$. However, f_{15} is eventually contracting since $D(f_{15})$ verifies (4.4), $\|D(f_{15}\| < G\lambda$ with $G = \beta_1 - \frac{\nu \sum a \neq i u_{a_k}^2}{\alpha_u}$. For the interval $\|u_{i_k}\| < \left\| \sqrt{\frac{1}{3} - \frac{\nu \sum_{a \neq i} u_{a_k}^2}{3\beta_i}} \right\|$, u_{i_k}, the repulsive fixed point, 0, pushes away the solution to the attractor fixed points, $-\frac{\nu \sum u_{i_k}^2}{\beta_1}$ and $\frac{\nu \sum u_{i_k}^2}{\beta_1}$. However, both fixed points identify the same active behavior for the robot, and thus

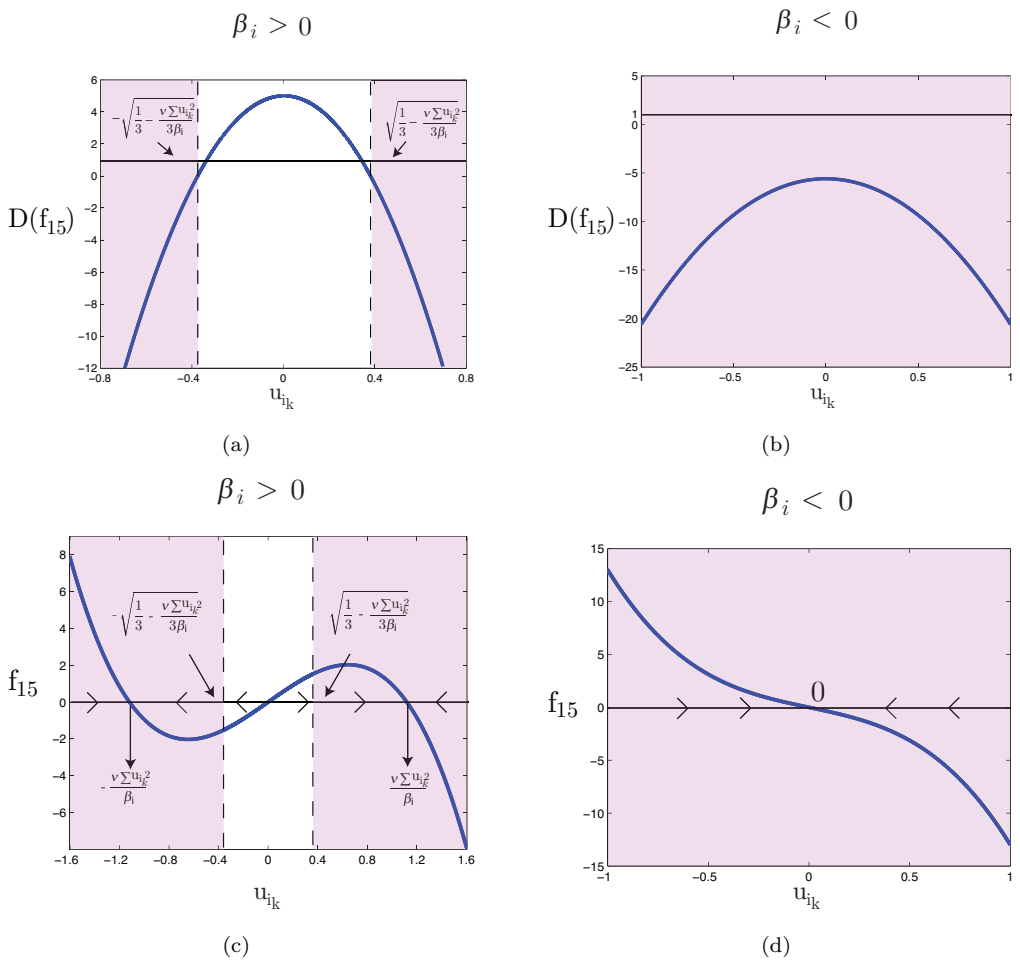


FIGURE 4.7: Phase portrait of $D(f_{15})$ and f_{15} for $\beta_i > 0$ and $\beta_i < 0$.

$\|u_{i_k}\|$ is adopted. Consequently, for this robotic application, it is not relevant the sign of the fixed point but rather, its Euclidean norm. Thus, f_{15} maintains the contraction principle for $\beta_i > 0$ and converges to the unique fixed point $\left\| \frac{\nu \sum u_{i_k}^2}{\beta_1} \right\|$ independently of the initial value of u_{i_k}.

For $\beta_i < 0$, condition (4.33) is always verified if $\nu > 0$. Thus, $D(f_{15})$ is below 1 for the entire domain (see fig. 4.7 (b)), and consequently f_{15} is contracting and converges to the unique fixed point, 0, independently of the initial value of u_{i_k}. Fig. 4.7 (d) illustrates the convergence of f_{15} when $\beta_i < 0$.

A Hopf bifurcation is expected to occur in block (f_{10}, f_{11}) (see chapter 4 in [80]) from a stable fixed point at $(m, n) = (O_m, 0)$, when $\mu < 0$, to a stable limit cycle when $\mu > 0$ (see example 1 in section 4.4 in [80]). To simplify the analysis, oscillator (f_{10}, f_{11}) is written in polar coordinates,

$$f_{10} \doteq r_{k+1} = r_k + r_k \alpha(\mu - r_k^2) d_k, \tag{4.35}$$

$$f_{11} \doteq \theta_{k+1} = \theta_k + \omega d_k. \tag{4.36}$$

According to remark 2.4.9 and proposition 2.4.10 in [292], the orbit of a periodic point p is a stable limit cycle if one eigenvalue is 1 and the others have absolute value below 1. The Jacobian of block (f_{10}, f_{11}) is given as,

$$D(f_{10}, f_{11}) = \begin{bmatrix} \frac{\partial f_{10}}{\partial r_k} & \frac{\partial f_{10}}{\partial \theta_k} \\ \frac{\partial f_{11}}{\partial r_k} & \frac{\partial f_{11}}{\partial \theta_k} \end{bmatrix} = \begin{bmatrix} 1 + \alpha(\mu - 3r_k^2)d_k & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.37}$$

This block has two eigenvalues, $\sigma_1 = 1 + \alpha(\mu - 3r_k^2)d_k$ and $\sigma_2 = 1$. Thus, there is a stable limit cycle around point p if,

$$\|1 + \alpha(\mu - 3r_k^2)d_k\| < 1. \tag{4.38}$$

This condition is true if,

$$-2 < \alpha(\mu - 3r_k^2)d_k < 0, \tag{4.39}$$

is verified. If $\alpha > 0$, $\mu > 0$, $3r_k^2 > \mu$ and $\alpha(\mu - 3r_k^2)d_k > -2$ are true, the absolute value of eigenvalue σ_1 is below 1. Assuming that d_k is sufficient small, $\alpha > 0$, $\mu > 0$ and $3r_k^2 > \mu$ are sufficient conditions to ensure that block (f_{10}, f_{11}) is contracting to a stable limit cycle and every point is positively asymptotic to its orbit.

Once conditions that ensure the contraction of the block of dynamical systems (f_{10}, f_{11}) and (f_{15}) were derived, the Jacobian of the maps $(f_{12}, f_{13}, f_{14}, f_{16}, f_{17})$ must be upper bounded.

Through a suitable choice of parameters a, b and c, f_{12}, f_{13} and f_{14} are continuous in $\mathbb{R}$, and therefore their Jacobians are upper bounded. O_s, O_e and O_r are finite values and u_{stop}, $u_{\text{execution}}$ and u_{rescue} ($\|u_{i_k}\|$) were demonstrated to converge to a unique fixed point if conditions (4.31) and (4.33) are true. Therefore, f_{16} and f_{17} are continuous in $\mathbb{R}$ and their Jacobians are upper bounded.

Parameters that ensure the contraction or eventual contraction of the dynamical systems derived in this section are summarized in table 4.1.

TABLE 4.1: Stability parameters.

Module	Conditions	Values
Motion Control	$\lambda_{\text{tr}} < 0$	$\lambda_{\text{tr}} = -0.9$
Local Control	$\lambda_{\text{tar}} < 0,\ \lambda_{\text{obs},i} > 0,\ \lambda_{\text{sobs},i} > 0$ $\phi_k - \psi_{\text{obs},i} > \pm\sigma_{\text{obs},i}$, $\phi_k - \psi_{\text{sobs},i} > \pm\sigma_{\text{sobs},i}$	$\lambda_{\text{tar}} = -0.6,\ \lambda_{\text{obs},i} = 0.7,\ \lambda_{\text{sobs},i} = 4$ $0.0359 < \sigma_{\text{obs},i} < 0.7869$, $0.2905 < \sigma_{\text{sobs},i} < 0.9023$
Timing Control	$\beta_i > \nu \sum_{a \neq i} u_{a_k}^2$ if $\beta_i > 0$ $\alpha > 0,\ \mu > 0,\ 3\, r_k^2 > \mu$	$\nu = 1.7,\ \beta_i = 3.5$ if $\beta_{\text{i}} > 0$, $\alpha = 100,\ \mu = A_k^2$

In the overall, this chapter illustrates the principles of the analysis of the proposed architecture.

Chapter 5

Experimental Setup

This chapter starts by describing the constraints that mobile robots usually face when delivering goods in hospitals. The indoor environment used for the field experiments is explained in section 5.2. Section 5.3 describes the hardware composed by the robot Pioneer-3DX and its on-board sensors. Later on, the frame assignment used to represent the position and orientation of the robot is depicted in section 5.4.

Before the field experiments, the architecture that guides the robot must be tested in a simulation software. This software and the simulated hospital environment are described in section 5.5. The chapter continues with a description and dependability analysis of the Localization System in sections 5.6 and 5.7, respectively. This chapter is concluded by describing the External module that represents the interaction between the robot and users.

5.1 Constraints of Hospital Environments

Mobile robots must be able to navigate in cluttered and dynamic environments, accounting for several constraints. These constraints should be satisfied by the robot, both at software and hardware levels. In case of hospital environments, such constraints can be partitioned as,

- Robot hardware design: a number of hardware features on the robot must be considered to address the physical problems of the environment like slopes, size of the rooms, corridors and doorways;
- Navigation and security: robots must navigate in different and cluttered areas of the hospital, which may be full of unpredictable obstacles and collisions must be

avoided. Thus, the security system must have higher priority over the navigation system, such that collisions are avoided at any cost;

- Localization: a localization system must provide an estimate of the robot's pose with sufficient accuracy to allow navigation. Localization failures are not acceptable, as in general, they require a technical intervention to re-locate the robot;
- Cost effectiveness: from a customer's perspective, hospital robots must be less expensive and more efficient than nurses or auxiliary staff when performing the same delivery missions;
- User interface: a simple interface allows users to visualize the information of the robot's current state and to manage robot missions.

5.1.1 Robot Hardware Design

Knowing the hospital layout before starting the missions is essential for mobile robots to know what routes they should take to complete the missions. In fact, most indoor environments can be represented by a set of predefined locations. For instance, in hospital environments, these locations can be patient, nurse or store rooms, *etc.*. Furthermore, these locations are connected by a network of corridors and elevators. Information of the locations and paths connecting them must be included into the map of the environment.

During missions, the robot should interact with several automated devices, such as elevators and automated doors. The robot should be able to call an elevator, select the required floor and communicate with automated doors, such that they can be opened, allowing the robot to pass through.

In terms of hospitals layout, corridors can usually be 1.8 m to 2.4 m wide, with many doors along the walls [48]. Some corridors have a low density of obstacles, but others are so cluttered with medical equipment that navigation is impossible until the equipment has been removed. The physical design of the robots must consider that its size can not exceed the dimensions of doors, corridors and elevators. Furthermore, they must be able to safely maneuver in the corridors.

Hospitals grounds are usually flat, such that steps and other irregularities do not prevent robots to complete the missions. For instance, if one robot detects stairs during its trajectory, it will be unable to finish its mission. However, the robot's wheels should be able to overcome small steps, as the alignment between elevators and floors, which might not be perfect.

The distance traveled by a delivery robot should be at least similar to the average distance covered during a working day by nurses or auxiliary staff. Furthermore, robots must be able to navigate 24 hours per day, excluding charging periods.

Robots must have a suitable container to carry out goods. According to [297], a container with the size of 60x50x45 cm, and a payload up to 50 kg is sufficient to carry out the typical hospital goods. Locking the containers is an important issue to ensure the security of the goods transported by the robot. During the transportation process, the container should be locked and it can only be opened by authorized people.

5.1.2 Navigation and Safety

Contrary to AGVs, the navigation of delivery robotic solutions for hospital environments can not be restricted to a rigid network of wires or tapes attached to the ground, as AGVs normally do in automated factories. Dynamic and static obstacles would interfere with the track-guided motion, leading to potential collisions. Therefore, hospital delivery robots must leave their routes in case of necessity. Dynamical obstacles include hospital staff, visitors, patients or other moving obstacles. Static obstacles include walls, stairs, beds, food trays, wheelchairs, stretches and other medical equipment. However, in hospitals, which are usually crowded and cluttered, the obstacle avoidance module can trap the robot in narrow areas that are difficult to handle. In such situations, it might be preferable that the robot stops and waits for the clearance of the area than trying to circumnavigated them. This assumes that eventually the obstacles are removed, allowing the robot to continue its mission. In fact, an emergency stop device is necessary according to European laws (EN 1525:1997) and it is activated only when necessary. In the unlike case an obstacle is not detected, bumpers mounted on the robot's base will sign the eventual collision and trigger the emergency stop.

The velocity that robots follow during missions is an important safety factor. Robots moving slowly interfere with the normal hospital workflow. On the other hand, faster robots have a higher probability of colliding with obstacles. The average walking velocity for older pedestrians is 0.92 m/s, and 1.2 m/s for younger pedestrians [298]. For short distances and short periods of time, a worker can reach an average velocity up to 1.6 m/s [5]. However, the velocity of a pedestrian is a function of the pedestrian density in the respective environment [299]. Even though the robot should follow a desirable velocity similar to humans, safety must be preserved. For instance, the Helpmate robot [48] navigates along the corridors at a velocity of approximately 0.6 m/s when the path is clear and 0.3 m/s when obstacles are detected. These values are

consistent to the approach suggested in [5], where the robot should adopt different velocities according to the area of the hospital: the highest velocity for non-patient areas, the medium velocity for patient areas and the lowest velocity for interactions with elevators.

Noise can have an adverse effect on patient sleep patterns. Robots should be as quiet as possible and do not emit noise and sound levels that may disturb patients, particularly at night.

5.1.3 Localization

Robots need to continuously know their poses while performing missions. A typical method to estimate the robot's pose in cluttered environments is to distribute artificial landmarks along them. However, in hospitals, it might be forbidden to place artificial landmarks. Using natural landmarks such as walls, lamps or corners is preferable, but it may hamper the accuracy of the robot's pose estimates. Odometry can be an auxiliary technique to estimate the robot's pose. However, uneven floors, wheel slippage, skidding or external forces might make unreliable the information provided by the wheel encoders for large distances. Thus, a fusion of multiple sensors can be used to reduce the errors on the robot's pose estimates.

The workspace of hospitals can be characterized by many narrow passages that the robot must go through, such as elevators and doors. The robot's pose should be estimated with a positioning error, such that the robot passes through doors and reaches elevators with success. Standard width for indoor doors is approximately 0.86 m. To successfully move through a door, the robot should navigate with a maximum error of half the door's width, considering that the robot passes through the center of the door. Thus, robots should navigate with a maximum expected error of 0.43 m. In the case that this accuracy can not be achieved, robots can fail to navigate through doors or reach the elevators. Note that the radius of the robot is not considered to calculate the maximum localization error. If the robot is close to the doorjambs, it will interpret them as obstacles and avoid them.

5.1.4 Interface

The interface is important for users to set up scheduling missions for the robot, monitor its execution and program the map of the environment. Scheduling missions consists on defining the locations (for instance the rooms) that the robot should reach, and the time assigned to complete each mission. Both the map and the missions must be provided to the robot only by authorized staff, such as nurses or auxiliary staff.

The interface should notify users with other relevant information about the robot's state. For instance, it can reveal the robot's position relative to the environment, the goal location, battery status, the current velocity and the status of the mission, *i.e.* if the mission is in time or delayed.

5.1.5 Cost Effectiveness

Robots should be able to perform delivery missions during 24 hours per day in a hospital environment. Nevertheless, if they are not cost efficient, hospital managers will not buy them. Therefore, a delivery robot must be less expensive and more efficient than a corresponding team of human workers when performing the same delivery task.

One typical solution to maximize the efficiency of delivery robots is to ensure that they follow the fastest route to reach their goal location. In general, the shortest path is also the fastest one. However, there are situations where obstacles may obstruct the shortest path, provoking a delay on the timing scheduling. The number of expected obstacles that can appear in a corridor must be considered to calculate the fastest path. Furthermore, the fastest path must be calculated at the beginning of each mission and it can change along the day. Moreover, the robot only initiates its delivery mission if there is at least one path connecting the initial location to the goal location.

There are some problems not considered in the scope of this thesis. Nevertheless, they are relevant when considering the implementation of mobile robots in hospital environments. For instance, how do robots should behave in case of a catastrophic emergency (fire, earthquakes, *etc.*), or get the attention of busy staff when they arrived at their goal locations and await for the unloading.

In conclusion, the specifications that hospital delivery robots must fulfill can be viewed in terms of physical design, mechanical performance and sensory performance. In terms of physical design, robots should be approximately 1 m long and 0.5 m wide, in order to navigate with sufficient space in corridors and pass through doors. They must have a container to store approximately 50 kg of goods. Moreover, robots must be able to overtake steps of approximately 3 cm, as the floors can be rugged or the alignment between elevators and floors can be not perfect [58]. In terms of mechanical performance, robots should reach different levels of velocity, according to the area of the hospital. In obstacle-free areas, robots should be able to reach velocities up to 1 m/s. In crowded areas, they should move slower. Even though their velocity levels, robots must be as silent as possible. In terms of sensory performance, several sensors must be mounted on robots to ensure that they do not collide with any obstacle. Also,

the localization system must provide estimates about their pose, with a maximum error of 0.43 m, such that robots navigate for long periods of time with success.

Specifications on the physical features of environments must also be assumed. They should have wide space enough for the robot navigation, flat floors and elevators aligned with the floor, such that robots can use them. Passages to all rooms must be clear, in order to ensure that robots are always able to reach the destination goals. Table 5.1 resumes the specifications for delivery mobile robots and identifies those that were fulfilled in this thesis.

TABLE 5.1: Specifications considered for delivery mobile robots (✓ - fulfilled, X - not fulfilled).

Design Issues	Specifications	
Robot Hardware	Container to carry out goods	X
	Bumpers	X
	Robot's size satisfies dimensions of doors and corridors	✓
	Robot's wheels overcome small steps	✓
	Low-level noise	✓
	Robot emits sound to notify its presence	X
Navigation	Interaction with automated devices	X
	Robot is able to navigate 6/7 Km during a working day	✓
	Leave the predefined route in case of necessity	✓
	Selection of the fastest route	✓
Safety	Robot's linear velocity changes according to the presence of obstacles	✓
	Specific behavior in emergency cases	X
	Emergency stop condition	✓
Localization	Robot's pose estimate < 0.43 m	✓
	Natural landmarks	X
Interface	Visual interface	X

5.2 Indoor Environment

The indoor environment selected for the field experiments is part of the Industrial Electronic Department of the University of Minho, located in Guimarães, Portugal. This environment mimics several hospital features, such as corridors with approximately the same width, passage between doors, cluttered rooms, static obstacles and people unfamiliar with the robot's behavior.

A schematic of the indoor environment is illustrated in fig. 5.1. Striped regions identify forbidden areas while empty regions identify free areas that can be used by the robot. The environment has approximately 130 m^2 of free area. Letters correspond to

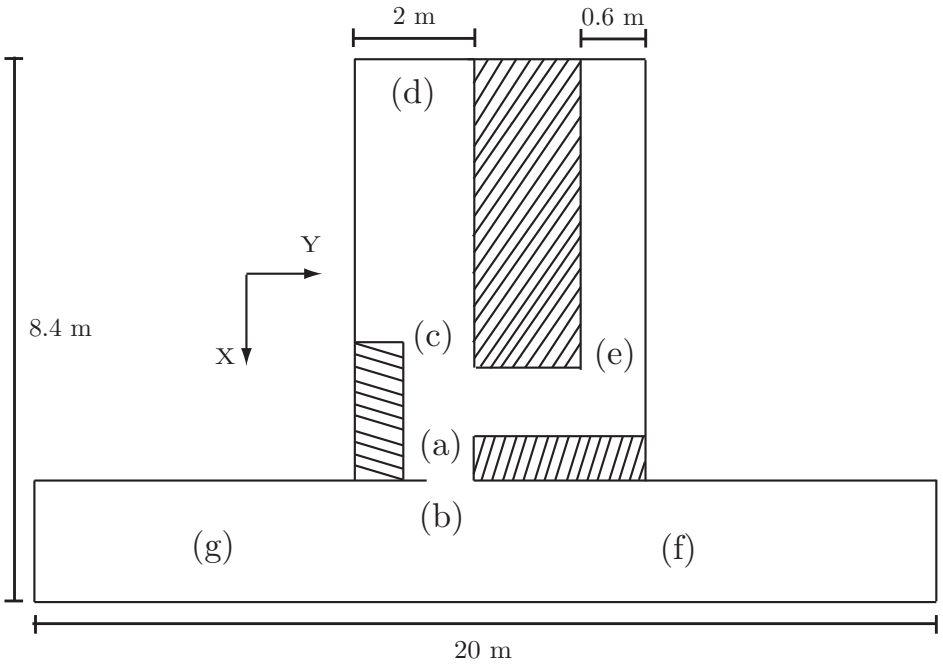


FIGURE 5.1: Map of the indoor environment in which the robot performs the field experiments.

the sequence of images of the indoor environment illustrated in fig. 5.2. Snapshots (a) and (b) depict the passage through a door between the corridor and the laboratory. Snapshots (c), (d) and (e) illustrate parts of the laboratory from different perspectives. Snapshots (f) and (g) depict the corridor.

During a working day, many people whose behavior is unpredictable move in the corridor. While some persons try to obstruct the robot, others treat it as an obstacle and keep a proper trajectory to avoid colliding with it. People move at a velocity of approximately 1.2 m/s in the corridor. Inside the laboratory, people move with lower velocities as it is a smaller and more cluttered area. In addition, they are familiarized with robots and do not interfere with the behavior of the robot.

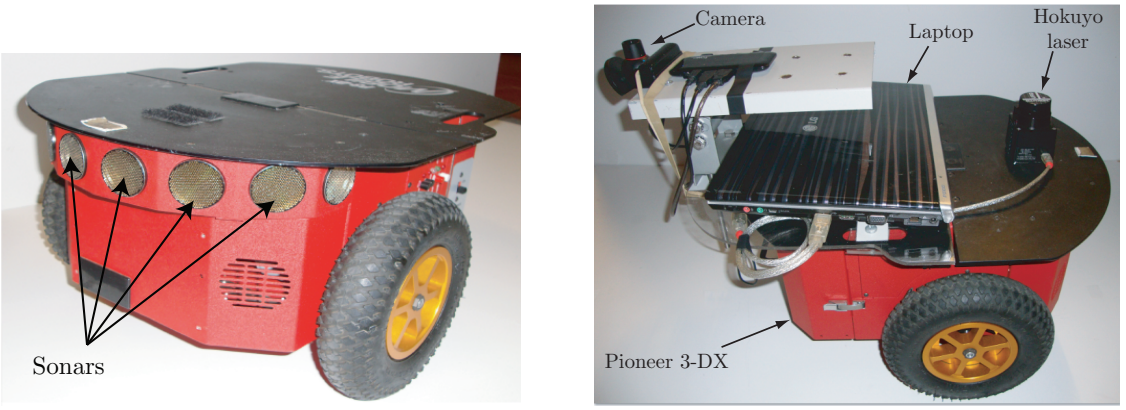
Corridors are 2.40 m wide (see panels (f) and (g) in fig. 5.2), which is enough for the robot to perform the obstacle avoidance maneuvers. The door that separates the laboratory from the corridor is 0.8 m width (see panels (a) and (b) in fig. 5.2), which is sufficient for the robot to pass through it if its localization error is at most 0.4 m.

FIGURE 5.2: Snapshots from the indoor environment: (a),(b) passage through a door, (c),(d),(e) areas of the laboratory, (f),(g) corridor.

5.3 Robot and Sensors

The adopted mobile robot is a Pioneer 3-DX (see fig. 5.3 (a)) developed by Adept MobileRobots Corporation.

Pioneer 3-DX has an oval shape with maximum and minimum diameters of 0.455 m and 0.381 m, respectively, and 0.237 m height. It can reach a maximum forward/backward velocity of 0.8 m/s and a rotation speed of 300°/s. 8 ultrasonic sensors are mounted on a ring and used to measure the robot surroundings. More details about



(a) (b)

FIGURE 5.3: Pioneer 3-DX and remaining hardware used to control the robot: camera, laptop and laser.

the Pioneer 3-DX features can be read in appendix B.1.

The robot uses a laser range finder (Hokuyo URG-04LX-UG01) and a RGB camera (PSEye), mounted onboard the robot to detect obstacles and to identify artificial landmarks in the environment, respectively. More details of these sensors can be seen in appendix B.2. Fig. 5.3 (b) shows the Pioneer 3-DX robot, the laptop that runs the navigation architecture, the camera and the laser.

5.4 Frame Assignment

Coordinate systems are defined to represent position and orientation in space. Each frame is characterized by a position vector and a rotation matrix. Five different coordinate frames (see fig. 5.4) have been applied,

1. The environment or allocentric reference frame $\{W\}$ (black axes);
2. The robot coordinate frame $\{R\}$, centered on the mobile robot (blue axes);
3. The camera coordinate frame $\{C\}$, fixed to the camera mounted on top of the robot and facing upwards (red axes);
4. The laser coordinate frame $\{L\}$, fixed on the robot's front. The laser frame lies along the robot's heading direction (green axes);
5. Each sonar sensor has a reference frame fixed pointing forward, $\{S_i\}$. The distance to obstacles measured by each sonar is relative to the center of their individual reference frames (yellow axes).

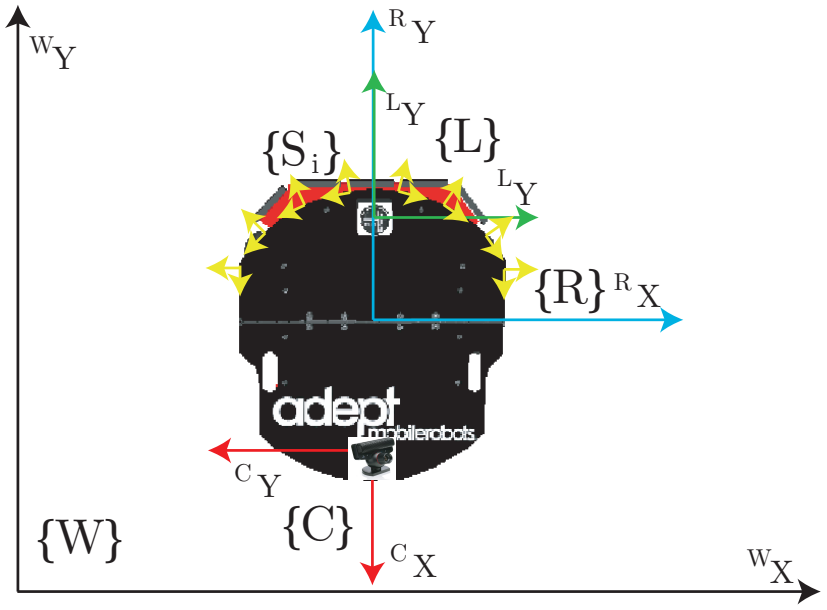


FIGURE 5.4: The robot's pose is measured relative to the allocentric reference frame, $\{W\}$ (black axes). Laser data is acquired relative to the coordinate frame $\{L\}$ (green axes). Landmarks detected on the ceiling are measured relative to $\{C\}$ (red axes) and converted *a posteriori* to $\{R\}$ (blue axes). Sonar data are acquired relative to the respective coordinate frame $\{S_i\}$ (yellow axes).

5.5 Simulator

Simulations are used to assess the ability of the architecture to fulfill the proposed aims. They save time in the design of the control architecture, as they increase the reliability level in the planning process and program development. Previewing the behavior of the robot in a simulated environment allows for a variety of configurations, controllers and "what-if scenarios" to be tried and tested, before implementing the architecture in the real robotic application.

The selected simulator is Webots [300]. This simulator provides real-time motion simulation of several mobile robots using their kinematic models and the geometry of the environment. Indoor environments can be customized with 3-D objects, simulating as close as possible real hospital environments.

Webots provides a large choice of simulated sensors and actuators to equip robots. Interactions between robots and environments are simulated with the Open Dynamics Engine (ODE). The sonar sensors mounted on Pioneer 3-DX robot are simulated through an algorithm reminiscent of ray-tracing. Errors on odometry include slip and encoder noise. Slip noise is simulated by adding a noise component to the command for each simulation step. The added noise is different for each wheel and presents a

uniform distribution. The encoder noise is simulated by adding a cumulative uniform noise to the encoder values at each simulation step. Both sonars and laser are also simulated with Gaussian white noise. The simulated model of the Pionner 3-DX is illustrated in fig. 5.5.

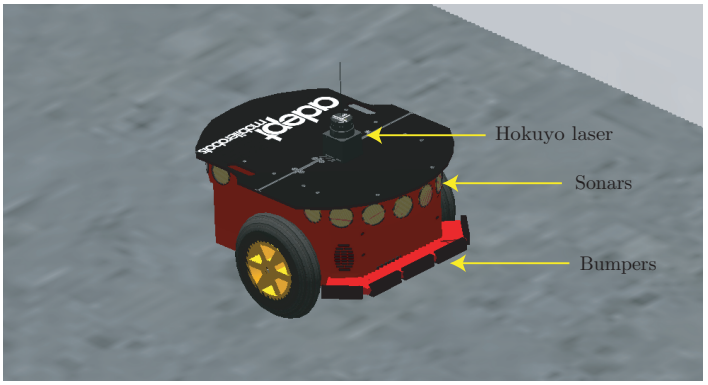


FIGURE 5.5: Simulated model of the Pionner 3-DX in Webots.

The simulated hospital environment is illustrated in fig. 5.6. The environment simulates a part of a hospital floor with an area of 324 m^2. It is composed by a

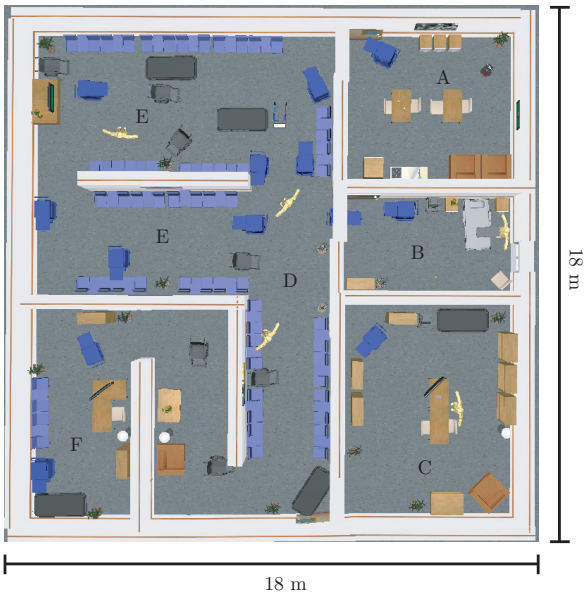


FIGURE 5.6: Webots simulated environment. Capital letters identify different regions of the hospital.

kitchen/eating room (letter A), a patient room (letter B), a doctor room (letter C), a corridor (letter D), waiting rooms (letter E) and a dispensary (letter F). All parts of the environment are connected by corridors or doors, such that the robot is able to reach any part of the environment regardless of its initial position.

Typical obstacles commonly found in hospitals are simulated through 3-D models, in order to achieve more realistic simulations (see fig. 5.7). The sensors mounted on the robot should be able to detect the different obstacles, which are very different in terms of physical structure. Nevertheless, the Local Control module should guide the robot, such that any collision is avoided.

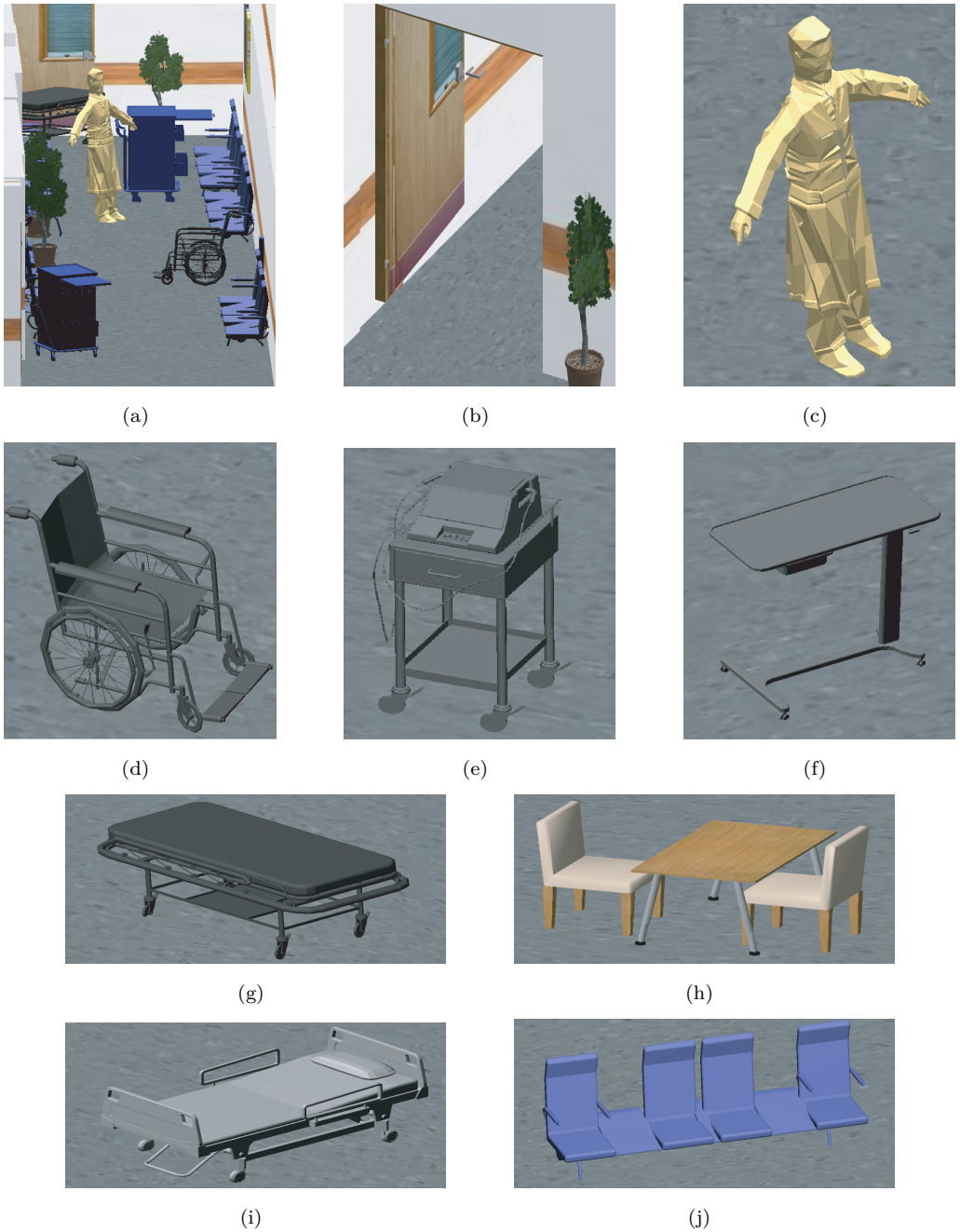


FIGURE 5.7: Simulated objects usually found in hospitals: (a) corridors with obstacles, (b) doors, (c) persons, (d) wheelchairs, (e) medical equipment, (f) food trays, (g) stretchers, (h) tables and chairs, (i) beds and (j) seats.

5.6 Localization System

This section describes the robot's localization system, which combines vision information and odometry through an Extended Kalman Filter (EKF). The vision system detects artificial landmarks distributed along the environment, such that their placement follows an optimized distribution.

5.6.1 Vision System

Using odometry to estimate the robot's pose lacks in accuracy for long distances, as odometry errors grow unbounded. A possibility to reset the odometry data and continue to provide an accurate robot's pose is to use an exteroceptive sensor. As described in section 2.2.4.1, cameras, UWB and pseudolites are used in several works with acceptable noise levels for hospital environments. In this work, vision is adopted, since simple RGB cameras are relatively cheap when compared to other exteroceptive sensors, such as pseudolites receptors or UWB. Furthermore, their uncertainty level on the robot's pose estimates is similar.

Vision has long been used to solve the robot localization problem by capturing natural or artificial landmarks placed in the environment. The robot knows the landmarks' position and when one landmark is detected, its position and distance to the robot is used to estimate the robot's pose. This requires that each landmark is unique.

In this work, landmarks are distributed on the ceiling, since, in populated environments, occlusions of the landmarks are usually less frequent when they are placed on the ceiling than in walls. The detection of the artificial landmarks is performed by the ARToolkit software library [301]. This software provides video tracking libraries that calculate in real-time the position and orientation of the camera relative to the detected landmarks. The landmarks information is stored in a database used by the ARToolkit to identify them and obtain their identification code. The detected landmark with the highest degree of confidence is the one used to estimate the robot's pose.

Some disadvantages should be considered when vision is used to estimate the robot's pose. The uncertainty level of the robot's localization depends on factors such as lighting conditions, camera's resolution and on the distance between the camera and the landmarks.

5.6.1.1 Vision System Configuration

The vision system consists on a camera mounted on the robot's top (see fig. 5.3 (b)) and pointing upwards to detect the landmarks on the ceiling. Fig. 5.8 (a) depicts the schematic of the camera mounted on the robot and the camera's field of view. The distance between the camera and the ceiling is approximately 2.8 m. Fig. 5.8 (b) represents the camera's field of view projected into the ceiling of the environment. The field of view has a minimum side of $m_s = 1.2$ m and an area of approximately 1.92 m^2. Only the landmarks placed within the camera's field of view are detected.

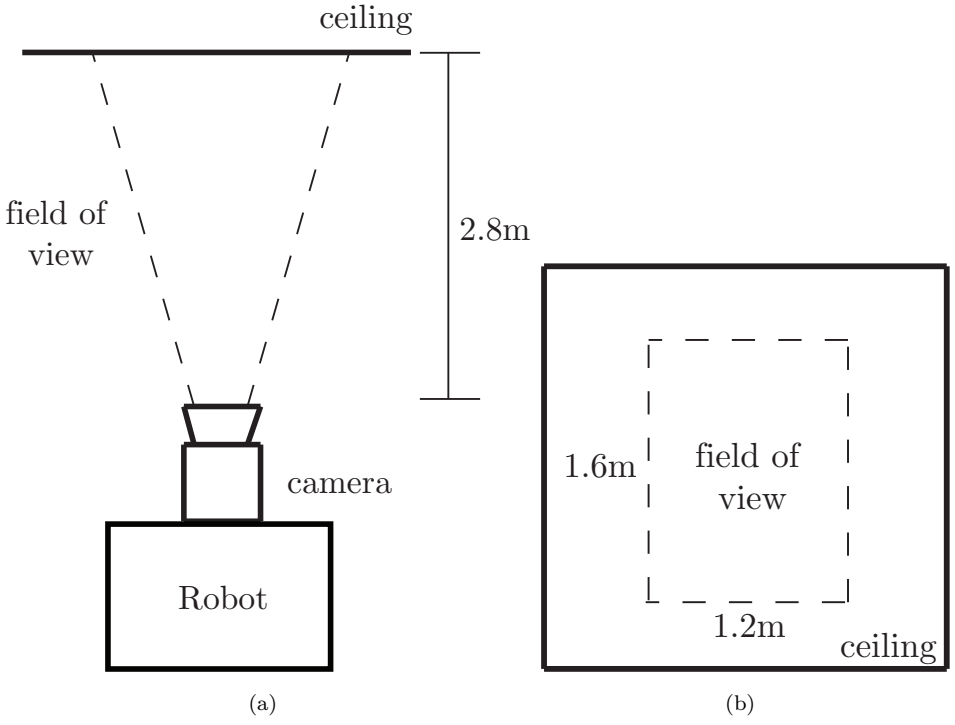


FIGURE 5.8: a) Representation of the camera on the robot's top pointing towards the ceiling. b) Camera's field of view projected into a part of the environment's ceiling.

5.6.1.2 Low Discrepancy Sequence

The number of available artificial landmarks to distribute on the environment is limited, either by computational efforts or by the high cost of the landmarks. Hence, the distribution of landmarks should follow an optimization process, in order to obtain the minimum number N of landmarks.

The optimal landmarks distribution problem has been addressed by several authors in literature. The approaches can be divided into two categories: online and offline. In the online selection, the robot selects at each time step the subset of visible features to

maximize a predefined policy. This can be performed through several methods as neural networks [126], entropy concepts [302] and reinforcement learning [303]. However, these methods rely on the fact that robots usually repeat the same trajectory.

In dynamic environments, obstacles can obstruct the trajectory of the robots and a trajectory replanning might be required to avoid collisions. Thus, an online landmark distribution is not optimized to every possible trajectory followed by the robot. In dynamic environments, offline methods present advantages over online ones. The set of landmarks in offline methods is selected independently of the robot's trajectory and before the robot starts moving. Offline methods [304–306] consider that the landmarks should be distributed to maximize their joint coverage of the environment, and at least one landmark lies in the direct line of sight to the robot.

In this work, an offline method to distribute the landmarks is adopted. Landmarks are equidistributed on the ceiling, and at least one landmark is always visible by the robot. These assumptions are already verified in [305].

An equidistributed sequence of points inside an arbitrary set B has a low discrepancy if the proportion of points in B is proportional to the measure of B. A bounded sequence of points $\{s_1, \ldots, s_n\}$ on an interval $[a, b]$, is said to be equidistributed if for any subset $[c, d]$ of $[a, b]$ we have,

$$\lim_{N\to\infty} \frac{|\{s_1, \ldots, s_N\} \cap [c, d]|}{N} = \frac{d-c}{b-a}, \tag{5.1}$$

and the discrepancy $D(L)$ of a sequence of points $\{s_1, \ldots, s_N\}$ with respect to the interval $[a, b]$ can be defined as,

$$D(L) = \sup_{a\le c\le d\le b} \left| \frac{\{s_1, \ldots, s_N\} \cap [c, d]}{N} - \frac{d-c}{b-a} \right|, \tag{5.2}$$

where L is the set of landmarks and the discrepancy $D(L)$ tends to 0 as N tends to infinity if the sequence is equidistributed.

The minimum number of N landmarks depends on the minimum side of the camera's field of view, $m_s = 1.2$ m. This constraint limits the projection area of the camera on the ceiling. The larger the projection area, the lesser the number of required landmarks. m_s can be viewed as the maximum distance between the landmarks that ensures that one landmark is always visible by the robot.

A typical process to obtain a low-discrepancy sequence is the rectangle rule. This rule consists on dividing a s-dimensional interval into N equal parts. The equidistributed sequence of points is then defined as the center position of the N equal parts. Fig. 5.9 depicts an example of a line segment defined in the interval [a,b] and divided

into N equal parts (black divisions). The equidistributed N points (red circles) are placed in the middle of adjacent equal parts.

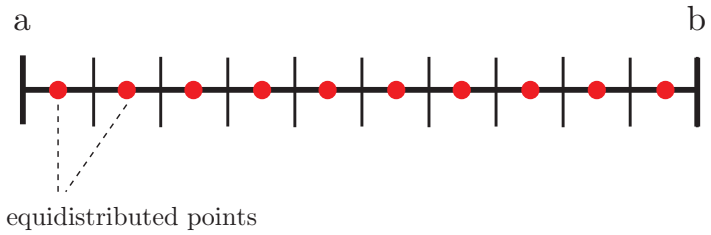


FIGURE 5.9: Example of a line segment divided into N equal parts (black divisions) and N equidistributed points (red circles).

5.6.2 Landmark Placement

Consider the ceiling of a region of the environment as a rectangular polygon with width w_i and height h_i. The number of necessary landmarks N to obtain an equidistribution in that region, such that at least one landmark is always visible by the robot is obtained by multiplying the integer value of the following divisions,

$$n_x = \text{int}\left(\frac{w_i}{m_s}\right), \tag{5.3}$$

$$n_y = \text{int}\left(\frac{h_i}{m_s}\right) \tag{5.4}$$

and

$$N = n_x n_y. \tag{5.5}$$

Without losing generality and assuming that each region r_i is polygonal, the boundaries of the environment $\mathbb{W}$ can be defined by polygonal lines parameterized by the arc length l,

$$\mathbb{W} = \begin{cases} c_1(l), & 0 < l < l_1 \\ c_2(l), & l_1 < l < l_2 \\ \vdots, & \vdots \\ c_p(l), & l_{p-1} < l < l_p \end{cases}. \tag{5.6}$$

A rectangular region r_i is defined by polygonal lines $c_1(l)$ and $c_3(l)$ with Euclidean norm $||b - a||$ and $c_2(l)$ and $c_4(l)$ with norm $||d - c||$. The distance between two

consecutive landmarks is calculated by using (5.3) and (5.4) as follows,

$$d_x = \frac{n_x}{||b-a||}, \tag{5.7}$$

$$d_y = \frac{n_y}{||d-c||}. \tag{5.8}$$

Through the rectangle rule, the equidistributed positions of the N landmarks for region r_i are obtained as follows,

$$x_{ls} = a + sd_y, \quad \forall s \in [1, 2, \dots n_x], \tag{5.9}$$

$$y_{lj} = c + jd_x, \quad \forall j \in [1, 2, \dots n_y]. \tag{5.10}$$

Finally, the set of N equidistributed landmarks L is defined as,

$$L \equiv \left\{ L_i = (x_{ls}, y_{lj}), \quad \forall s \in [1, 2, \dots n_x] \bigvee \forall j \in [1, 2, \dots n_y] \right\}. \tag{5.11}$$

In summary, the number of landmarks for each region of the environment is calculated through conditions (5.3), (5.4) and (5.5). The necessary landmarks for each region r_i of the environment depicted in fig. 5.1 is calculated through the proposed equidistribution optimization. Once the total number of required landmarks is known, $N = 32$, and for each region, $N_1 = 8$, $N_2 = 4$, $N_3 = 3$, $N_4 = 1$, $N_5 = 4$, $N_6 = 12$, conditions (5.9) and (5.10) are applied to obtain the exact location of each landmark (see fig. 5.10 to visualize the equidistribution of the N landmarks in the indoor environment).

5.6.3 Sensor Fusion - An Extended Kalman Filter Approach

The localization of the robot is obtained by combining the information provided by a camera mounted on the robot and by the encoders on the robot's wheels. The fusion between both sensors should provide a more accurate solution than any of the sensors when considered solely. A popular solution to fuse multiple data sources is the Kalman filter. This filter is widely used when sensor fusion or reducing the contribution of noisy measurements to the estimates are required. When the state model to estimate and/or the observation model are nonlinear, which is the case in this work, the Extended Kalman filter (EKF) is used.

EKF has several features that make it suitable to deal with multisensor estimation and data fusion problems. In particular, the explicit description of the process and observations allows a wide variety of different sensor models to be incorporated within

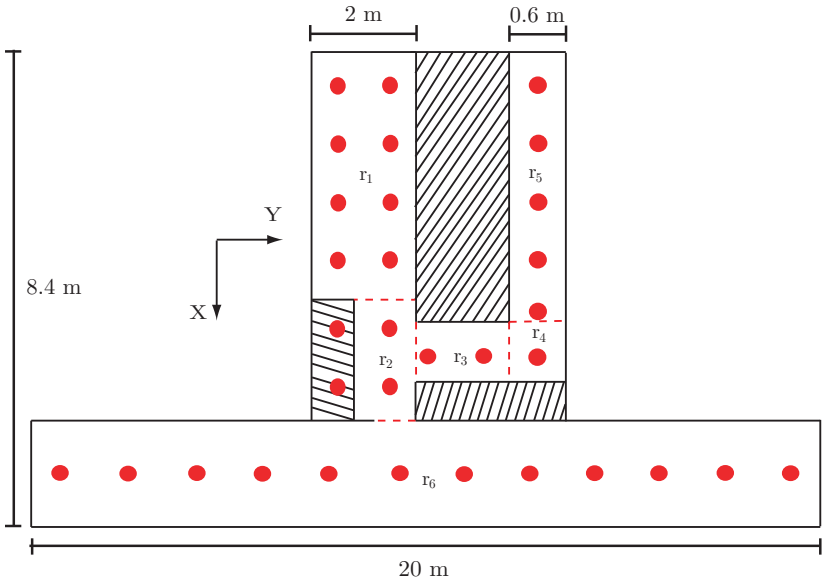


FIGURE 5.10: Red circles identify the landmarks distributed on the environment.

the basic algorithm. Furthermore, using the statistical measures of the uncertainty makes it possible to evaluate the role that each sensor plays in the overall system performance.

Several works have using Kalman filters to fuse multiple data sources in mobile navigation applications [307–309]. However, a major caveat in EKFs, is that they need to operate in the correct neighborhood to ensure that the linearized models are validated.

Particle filters could also be applied to deal with multisensor estimation and data fusion problems. However, they need a trade off between accuracy and efficiency. In fact, there is not a method to calculate the most efficient number of particles for each application. Furthermore, computational requirements are higher than in EKF. Particle filters are more suitable for multi-hypothesis estimations, where simple EKF are not suitable.

5.6.3.1 EKF Formulation

There are several methods to fuse sensorial information through an EKF (details of the most important methods can be viewed in [310]). One of the most used methods is the called direct EKF scheme (see fig. 5.11). In this scheme, the measurement signals are combined into the EKF and it can still estimate even if only one measurement signal is available. The more the measurement signals are available, the more accurate are the estimates. The disadvantage of this scheme is that it needs to know the dynamic model of the sensors, in order to find the predicted position.

The Localization System module combines the data provided by odometry, X_o[1], and by vision information, X_v, to estimate the robot's pose, $\hat{X}_r$ (see fig. 5.11). The

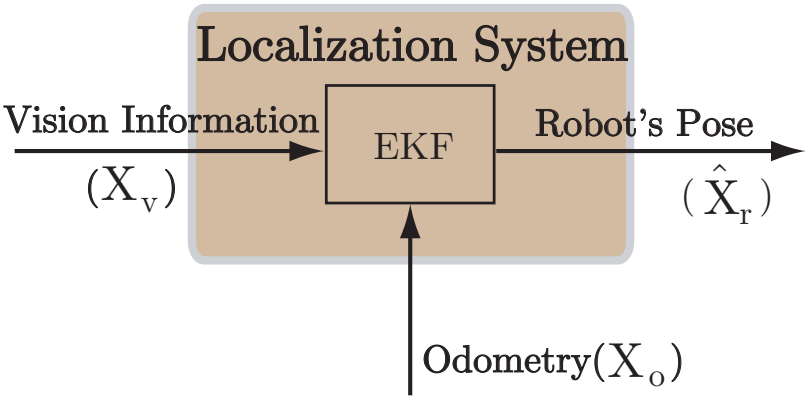


FIGURE 5.11: Localization System module composed by a direct EKF. The robot's pose $\hat{X}_r$ is estimated by combining two data sources: vision information and odometry.

vision information data, $X_v = [x_v, y_v, \phi_v]$, provides the robot's pose obtained from the detected landmarks. The odometry data, $X_o = [x_o, y_o, \phi_o]$, is obtained through the wheels encoders and provides the robot's pose according to the wheels rotation.

The EKF requires a state space model of the dynamical systems, f, describing the time evolution of the robot's pose, $X_r = [x_r, y_r, \phi]$, at each instant of time k as follows,

$$x_{rk} = x_{rk-1} + v_k \cos(\phi_{rk}) d_k, \tag{5.12}$$

$$y_{rk} = y_{rk-1} + v_k \sin(\phi_{rk}) d_k \tag{5.13}$$

$$\phi_k = \phi_{rk-1} \omega_k d_k, \tag{5.14}$$

where v_k and ϕ_k are respectively the linear and angular velocity of the robot at instant of time k. d_k is the discretization step.

The nonlinear system describing the robot's pose dynamics can be expressed in a more suitable form as follows,

$$X_{rk} = f(X_{rk-1}, U_{k-1}) + W_k, \tag{5.15}$$

$$Y_{rk} = h(X_{rk}) + Z_k, \tag{5.16}$$

where function f is used to compute the predicted state from the previous estimate and similarly, function h computes the predicted measurement from the predicted state and $U_{k-1} = [v_k, \omega_k]^T$ is the control variable. Variables W_k and Z_k represent the process

[1]Upper case letters represent non-scalar variables.

and measurement noise, respectively. They are assumed white and independent of each other and with normal probability distribution.

The EKF linearizes the nonlinear system to be estimated (5.15)-(5.16) and obtains estimates of the state space model as follows,

$$X_{\mathrm{r}k} \approx \hat{X}_{\mathrm{r}k}^{-} + F_k\,(X_{\mathrm{r}k-1} - \hat{X}_{\mathrm{r}k-1}) + L_k\,W_{k-1}, \tag{5.17}$$

$$Y_{\mathrm{r}k} \approx \hat{Y}_{\mathrm{r}k}^{-} + H_k\,(X_{\mathrm{r}k-1} - \hat{X}_{\mathrm{r}k-1}) + M_k\,Z_{k-1}, \tag{5.18}$$

where F_k and H_k are the Jacobian matrices of f and h relative to $X_{\mathrm{r}k}$. L_k and M_k are the Jacobian matrices of f and h relative to W_{k-1} and Z_{k-1}, respectively,

$$F_k = \left.\frac{\partial f}{\partial X_{\mathrm{r}k}}\right|_{\hat{X}_{\mathrm{r}k-1},U_{k-1}} = \begin{bmatrix} 1 & 0 & v_{k-1}\sin(\phi_{k-1})d_k \\ 0 & 1 & -v_{k-1}\cos(\phi_{k-1})d_k \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.19}$$

$$H_k = \left.\frac{\partial h}{\partial X_{\mathrm{r}k}}\right|_{\hat{X}_{\mathrm{r}k}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.20}$$

$$L_k = \left.\frac{\partial f}{\partial W_k}\right|_{\hat{X}_{\mathrm{r}}k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.21}$$

$$M_k = \left.\frac{\partial f}{\partial Z_k}\right|_{\hat{X}_{\mathrm{r}k}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.22}$$

The EKF time update equations are defined as follows,

$$\hat{X}_{\mathrm{r}k}^{-} = f\,(x_{\mathrm{r}k-1}, y_{\mathrm{r}k-1}, \phi_{k-1}, U_{k-1})\,, \tag{5.23}$$

$$P_k^{-} = F_k\,P_{k-1}\,F_k^T + Q, \tag{5.24}$$

where P_k^{-} is the *a priori* error covariance, P_k is the *a posteriori* error covariance, and $\hat{\mathbf{X}}_k^{-}$ is the state estimated *a priori*. The EKF measurement update equations are given as follows,

$$K_k = P_k^{-}\,H_k^T(H_k\,P_k^{-}\,H_k^T + R)^{-1} \tag{5.25}$$

$$N_k = S_k - Y_{\mathrm{r}k} \tag{5.26}$$

$$\hat{X}_{\mathrm{r}k} = \hat{X}_{\mathrm{r}k}^{-} + K_k\,N_k \tag{5.27}$$

$$P_k = (I - K_k\,H_k)\,P_k^{-} \tag{5.28}$$

where K_k is the Kalman gain, N_k is the innovation, $\hat{X}_k$ represents the state estimated *a posteriori* and S_k represents the input of the EKF, which is defined by the robot's pose obtained from odometry and vision information,

$$S_k = \begin{bmatrix} x_{\mathrm{o}} \\ y_{\mathrm{o}} \\ \phi_{\mathrm{o}} \\ x_{\mathrm{v}} \\ y_{\mathrm{v}} \\ \phi_{\mathrm{v}} \end{bmatrix}. \tag{5.29}$$

5.6.3.2 Measurement and Process Matrices

Matrices Q and R are used to specify the weight that the process and the sensor measurements contribute to the estimated state. Matrix Q defines how accurate is the nonlinear system (5.12)-(5.14). The difference between the real dynamics and the nonlinear system is called the process error, $\sigma_{\mathbf{p}}$. The higher the unmodeled nonlinearities in the real dynamics, the higher the process error. This error is used to build the matrix Q as follows,

$$Q = \begin{bmatrix} \sigma_p & 0 & 0 \\ 0 & \sigma_p & 0 \\ 0 & 0 & \sigma_p \end{bmatrix}. \tag{5.30}$$

Matrix R specifies the uncertainty of the input sensors data (odometry and vision system). The uncertainty is measured relative to coordinates x_{r} and y_{r} of the robot's position, and to the robot's heading direction, $\sigma_{\mathbf{o}} = [\sigma_{ox}, \sigma_{oy}, \sigma_{o\phi}]$ and $\sigma_{\mathbf{v}} = [\sigma_{vx}, \sigma_{vy}, \sigma_{v\phi}]$ for odometry and vision system, respectively. Thus, matrix R is built as follows,

$$R = \begin{bmatrix} \sigma_{ox} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{oy} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{o\phi} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{vx} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{vy} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{v\phi} \end{bmatrix}. \tag{5.31}$$

Matrix R defines the weight that odometry and vision data contribute to the EKF estimates. If the odometry uncertainty, $\sigma_{\mathbf{o}}$, is higher than the vision uncertainty, $\sigma_{\mathbf{v}}$, then odometry should contribute more than vision to the EKF estimates. On the

other hand, if the vision uncertainty is higher than the odometry uncertainty, the EKF estimates should rely more on the vision information than on odometry.

If the unmodeled nonlinearities existing in the system are neglected, then the process noise, $\sigma_{\mathbf{p}}$, is smaller than the sensors uncertainty, $\sigma_{\mathbf{p}} < \{\sigma_{\mathbf{v}}, \sigma_{\mathbf{o}}\}$. Consequently, the EKF strongly relies on the nonlinear equations and gives less importance to the measurements provided by sensors. On the other hand, if the unmodeled nonlinearities affect considerably the system, then the process noise must be larger than the uncertainty of the sensors, $\sigma_{\mathbf{p}} > \{\sigma_{\mathbf{v}}, \sigma_{\mathbf{o}}\}$. Thus, the EKF strongly relies on the sensors measurements. Both matrices R and Q are diagonal, since it is assumed that the errors between odometry and vision are independent from each other.

5.6.3.3 Odometry and Vision Information Uncertainty

The odometry uncertainty, $\sigma_{\mathbf{o}} = [\sigma_{ox}, \sigma_{oy}, \sigma_{o\phi}]$, and the vision uncertainty, $\sigma_{\mathbf{v}} = [\sigma_{vx}, \sigma_{vy}, \sigma_{v\phi}]$, are obtained from field tests performed with the Pioneer 3-DX. The tests include displacement of the robot along the x-axis and y-axis, and rotation over the z-axis.

Odometry Uncertainty

Table 5.2 summarizes the results of the field tests performed to calculate the odometry uncertainty. For each test, 10 trials are performed. At the end of each trial, the robot's pose is compared to the ground truth and the average result is considered. The ground truth is calculated by comparing the robot's pose at the end of each trial with the allocentric reference frame.

In the first experiment, the robot moves 4 m along one axis. The average error is 0.1 m with a standard deviation of 0.04 m. The second test consists on verifying the uncertainty of the robot's heading direction when the robot rotates around the z-axis and does not move along one of the axes. The error on the robot's heading direction is approximately 0.22 rad with a standard deviation of 0.07 rad. However, this value increases to approximately 0.7 rad if the robot rotates 2 turns around the z-axis. Note that the uncertainty on the robot's pose when the robot moves along x-axis or y-axis is similar. The last two tests are performed to verify the uncertainty when combining movement along the x-axis (or y-axis) and rotations over the z-axis. Initially, the robot rotates 1 or 2 turns according to the test and then moves 1 m. For 1 rotation, the error is approximately 0.29 m and the standard deviation is 0.08 m. For 2 rotations, the error increases to 0.4 m with a standard deviation of 0.1 m. Other more complex tests could be performed, but these will lead to larger odometry errors. In fact, combining linear and angular movement leads to catastrophic localization errors after a few meters

of displacement. The odometry uncertainty increases as the robot moves forward or

TABLE 5.2: Results of the experimental field tests to verify the odometry uncertainty.

Experiment	Average (m) x and y axes	Std. dev. (m) x and y axes)	Average (rad) orientation	Std. dev. (rad) orientation)
4 m displacement along one axis	0.1	0.04	0.08	0.02
1 rotation over z-axis	0.01	0.006	0.22	0.07
2 rotations over z-axis	0.015	0.009	0.7	0.18
1 Rotation and 1 m displacement	0.29	0.08	0.26	0.1
2 Rotations and 1 m displacement	0.4	0.1	0.75	0.2

rotates over the z-axis. Thus, for the EKF purposes, odometry should contribute more to the estimates during short distances and small rotations.

Both the average and the standard deviation are important to define the uncertainty of each sensor. One solution is to define the data uncertainty for the EKF as the sum of the average and the standard deviation. Thus, for short distances and small rotations the odometry uncertainty can be given as $\sigma_{\mathbf{o}} = [0.14, 0.14, 0.1]$.

Vision Information Uncertainty

There are typical errors that must be considered when vision is used to detect landmarks. First, when a landmark is detected, its position is acquired with a certain level of noise. Second, there are false positive detections, where the camera detects a landmark that it is not within the range of detection. Finally, there are false negative detections, in which the camera does not detect landmarks that are in the range of detection.

To analyze the uncertainty of the vision system, $\sigma_{\mathbf{v}} = [\sigma_{vx}, \sigma_{vy}, \sigma_{v\phi}]$, the robot performs a sequence of single missions. When the robot reaches a goal location, it rotates towards the next goal location and moves to it. This sequence is repeated 10 times. The error values $(e_{vx}, e_{vy}, e_{v\phi})$ are the difference between the ground truth and the values provided by the camera. The error on coordinates x and y is depicted in fig. 5.12 (a). Their average error is approximately 0.07 m and the standard deviation is 0.23 m. Fig. 5.12 (b) shows the error relative to the robot's heading direction. The average error is approximately 0.01 rad and the standard deviation is 0.09 rad.

The vision uncertainty is constant independently of the distance covered by the robot, or by the number of rotations that the robot performs during a mission. Similarly to odometry, the value of the vision uncertainty is calculated as the sum of its average and standard deviation. Thus, the vision uncertainty can be defined as $\sigma_{\mathbf{o}} = [0.3, 0.3, 0.1]$.

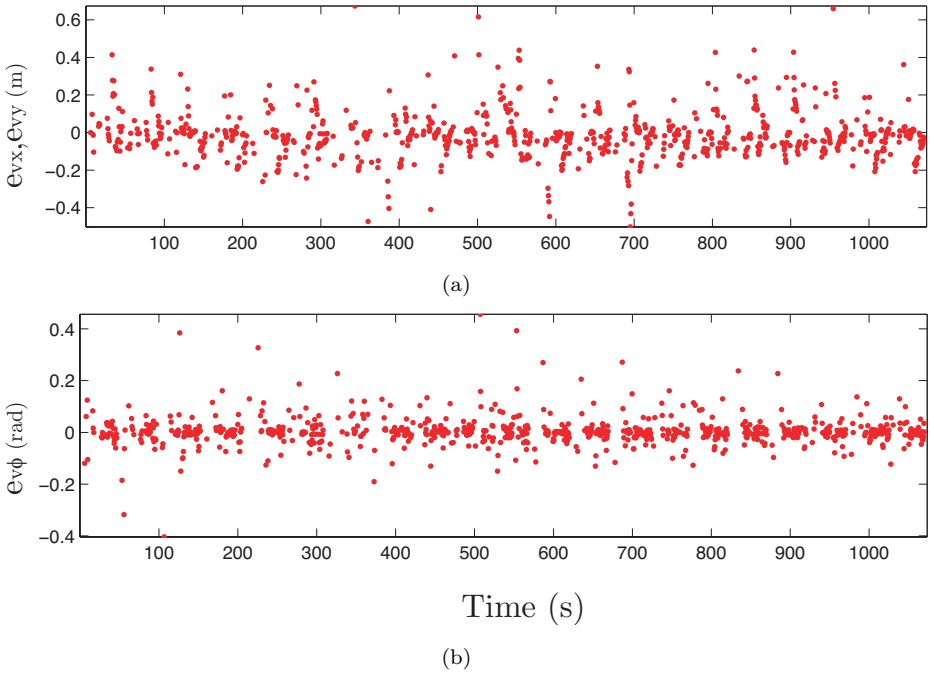


FIGURE 5.12: Error of the robot's pose calculated by the camera when compared to the ground truth. (a) Coordinates x and y of the robot's pose. (b) Orientation of the robot.

The uncertainty tests show that for short displacements, (< 4 m), and without rotations, the odometry uncertainty is lower than the vision uncertainty. However, odometry uncertainty grows unbounded, up to a certain point that exceeds the vision uncertainty. At this point, odometry is unreliable and vision becomes the most reliable sensor to estimate the robot's pose. A reset operation is necessary to change the weights that vision and odometry contribute to the EKF estimates. Thus, two heuristic conditions are defined to trigger the reset operation,

- the robot moves more than 4 m;
- the robot rotates more than $\frac{\sqrt{2}}{2}$ rad.

When one of these conditions is verified, the reset operation is triggered and matrix R is changed. Even though the reset conditions were not exhaustive studied, they revealed to be important conditions to change the sensors weights on the EKF formulation.

5.6.3.4 Reset Operation

The reset operation is performed by updating the matrix R. This operation is usual when multiple sensors have to be considered. For instance, there are situations in which the most reliable sensor becomes inaccurate, and the weight of its contribution

to the estimates should be decreased. A general solution [311, 312] is to use fuzzy logic to decide which is the most reliable sensor and thus change the matrix R accordingly.

Two R matrices were adopted, one for the normal operation, R_n, and another for the reset operation, R_r. In the normal operation, odometry provides more accurate estimates of the robot's pose than vision. Thus, $R = R_n$. The reset operation is triggered when odometry becomes unreliable, and this occurs when one of the two reset conditions is verified. During the reset operation, $R = R_r$.

Matrix R_n is built based on the tests accomplished to verify the odometry and vision uncertainty, as depicted in (5.31),

$$R_n = \begin{bmatrix} 0.14 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.14 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}. \tag{5.32}$$

In the reset operation, odometry has a very weak contribution to the robot's pose estimates. A sufficient value for the odometry uncertainty during the reset operation is $\sigma_{\mathbf{o}} = [0.5, 0.5, 0.5]$. These values were empirically validated through field experiments, but other values could be applied as well. Thus, matrix R_r is defined as,

$$R_r = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}. \tag{5.33}$$

Once matrix R is defined, it is required to define matrix Q. The estimation of the process error is assumed constant, *i.e.*, the unmodeled nonlinearities presented in the system do not change as the robot performs the missions. Thus, the process error is empirically defined as $\sigma_{\mathbf{p}} = [0.5, 0.5, 0.5]$ and matrix Q is defined as follows,

$$Q = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}. \tag{5.34}$$

Field experiments show that the EKF estimates with these assumptions are sufficient accurate to keep the robot well localized, *i.e.*, its uncertainty when compared to the ground truth are at most equal to the maximum expected error for the used indoor environment 0.40 m.

5.7 Dependability of the Localization System

Dependability can measure the service quality provided by a system, so that the user can have justified confidence on it. Dependability involves physical safety and operational robustness, such as reliability and integrity [296, 313]. Some of these factors can overlap, as their respective measures carry a certain degree of subjectivity.

Monte Carlo tests might be used to assess the performance of the localization system by analyzing its accuracy, precision and convergence time. Accuracy gives the closeness of the estimates to the ground truth, and precision is related to the repeatability of the estimates. Integrity assesses the ability of the localization system to recover from unexpected situations, *e.g.*, (i) collisions, (ii) landmarks failures and (iii) robot kidnapping.

5.7.1 Accuracy and Precision

The accuracy and precision of the localization system are obtained from a set of 20 experiments. In each experiment, the robot is stopped at random locations and the estimates of its pose are compared to the ground truth. Table 5.3 summarizes the results of these experiments. Errors e_x, e_y, e_ϕ stand for the error on x, y coordinates and orientation of the robot, respectively. The average error of the coordinates x and y is approximately 0.16 m and its standard deviation is 0.1 m.

Table 5.3: Accuracy and precision of the Localization System.

Error	Accuracy	Precision
e_x, e_y (m)	0.16	0.1
e_ϕ (rad)	0.05	0.01

Even though the accuracy of the localization system is sufficient to allow the navigation of the robot through doors with a maximum width of approximately 0.86 m, its precision relative to coordinates x and y is poor. In cluttered environments with small free areas, poor precision can increase the number of mission failures. In terms of the robot's orientation, the obtained values are sufficient to allow the navigation of the robot.

Nonetheless, these values are similar to current UWB standalone indoor localization systems [193, 194] and to systems combining odometry, laser range measurements, and *a priori* known maps [314].

5.7.2 Convergence Time

The convergence rate of the robot's localization is obtained from a set of 20 field experiments. The robot is stopped and does not move during the experiments. Furthermore, it does not know its initial location and the EKF is initialized at a random location for each experiment. When compared to the ground truth, the random initializations of the EKF have an error of approximately 4 m in coordinates x, y and 3 rad relative to orientation.

It is considered that the localization system has converged, when the EKF estimates are approximately equal to the values obtained in table 5.3. Fig. 5.13 shows the precision of the robot's pose, namely in coordinates x, y and in orientation, ϕ, along the time. In terms of coordinates x, y, it can be seen in fig 5.13 (a)-(b) that the robot's pose converges to approximately 0.16 m in 5 s. The orientation ϕ takes approximately 1 s to converge to 0.05 rad.

5.7.3 Integrity

In order to evaluate the ability of the robot to recover from unexpected situations, several experiments are conducted to expose the robot to the following failures: (i) collisions deliberately caused with the robot. Despite the navigation and security systems that prevent the robot to collide with obstacles, some people can intentional or involuntary collide with the robot. The localization system must be able to deal with those situations, in such a way that the robot is never lost. (ii) Variations on the percentage of detected landmarks. Along the day, the number of detected landmarks changes as the lighting conditions vary. Also, the distance that the robot needs to travel to detect a landmark increases as the number of detected landmarks decreases. (iii) How do the localization system deals with the kidnapped problem [185, 186].

5.7.3.1 Collisions

Collisions with obstacles may occur when the robot moves in populated environments, as people may inadvertently bump the robot. To verify the ability of the localization system to keep providing estimates of the robot's pose after collision, the robot performs a sequence of missions lasting 500 s and covering 100 m. During these missions,

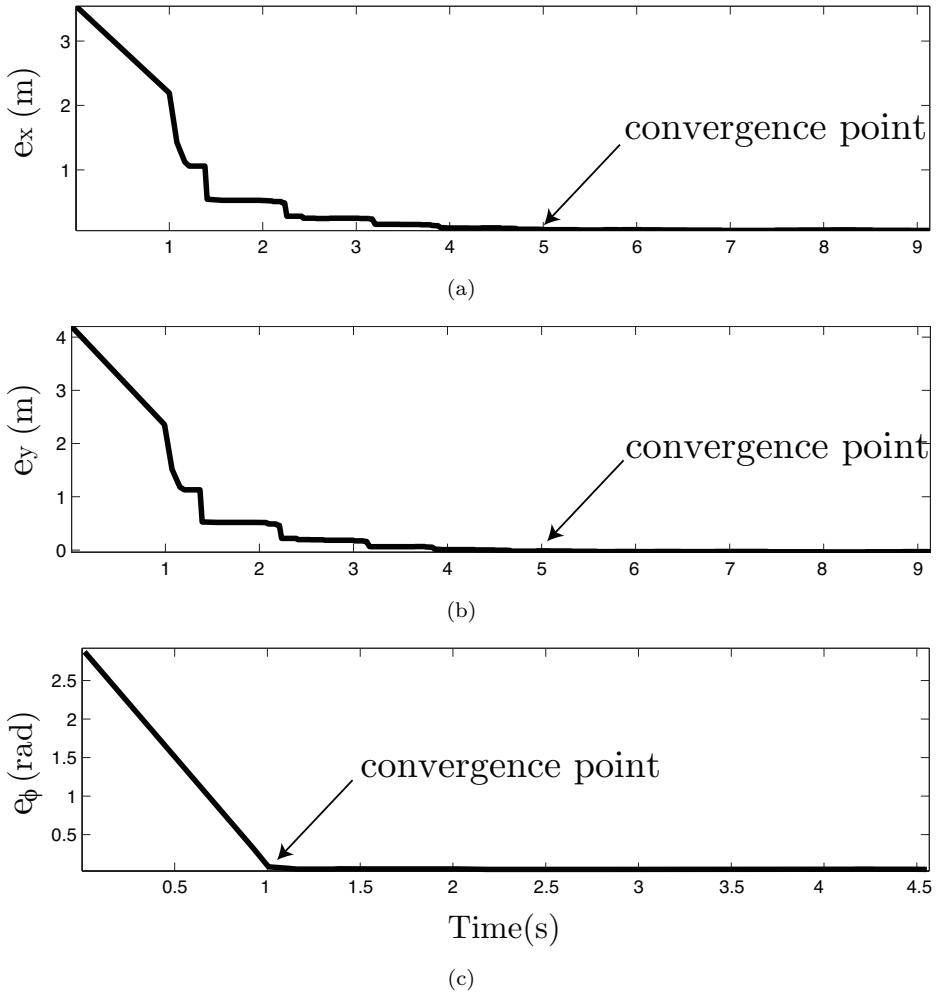


FIGURE 5.13: Precision *vs* time: convergence of the localization system. Blue line depicts the average error of the 20 experiments.

some people intentionally collide with the robot by bumping its front and forcing it to rotate and find a new trajectory.

Fig. 5.14 depicts the robot's linear velocity during this experiment. The continuous blue line in fig. 5.14 (a) illustrates the minimum distance between the robot and obstacles measured by the laser. Whenever this distance lies below 0.25 m (identified by the dashed red line), a collision occurs. Thirteen collisions are verified during the missions (green circles). Fig. 5.14 (b) illustrates the robot's linear velocity (continuous blue line) and vertical dashed green lines depict the instants of time where collisions occur. Despite the collisions, the Timing Control module successfully generates the suitable linear velocity, as the robot completes the missions in due time.

Fig. 5.15 illustrates the estimates of the robot's pose (blue line) $(\hat{x}_{\mathrm{r}}, \hat{y}_{\mathrm{r}}, \hat{\phi})$ and green dashed lines depict situations where collisions occur. The data provided by the camera

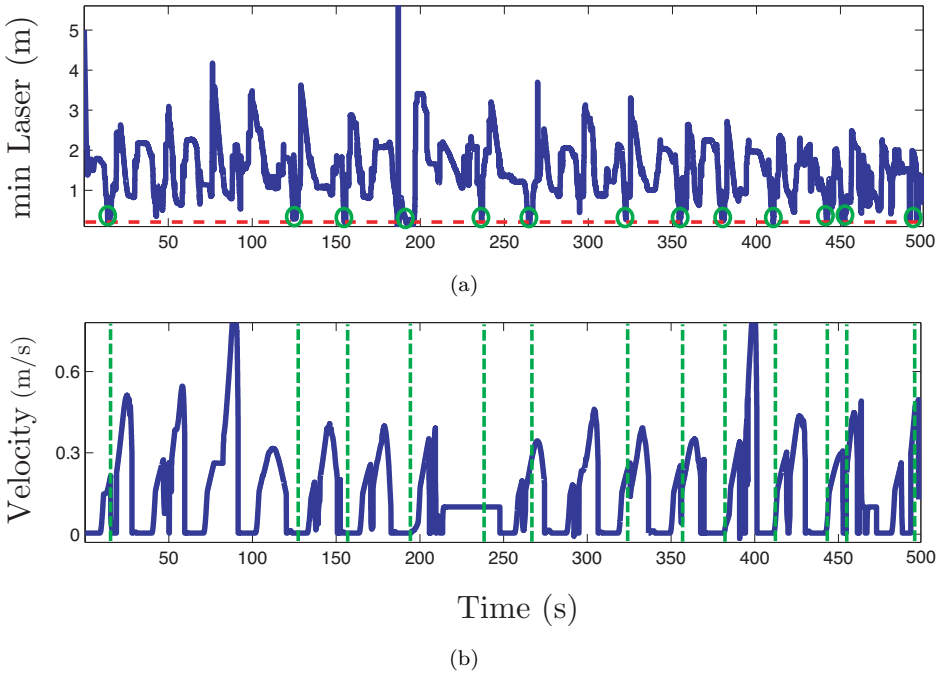


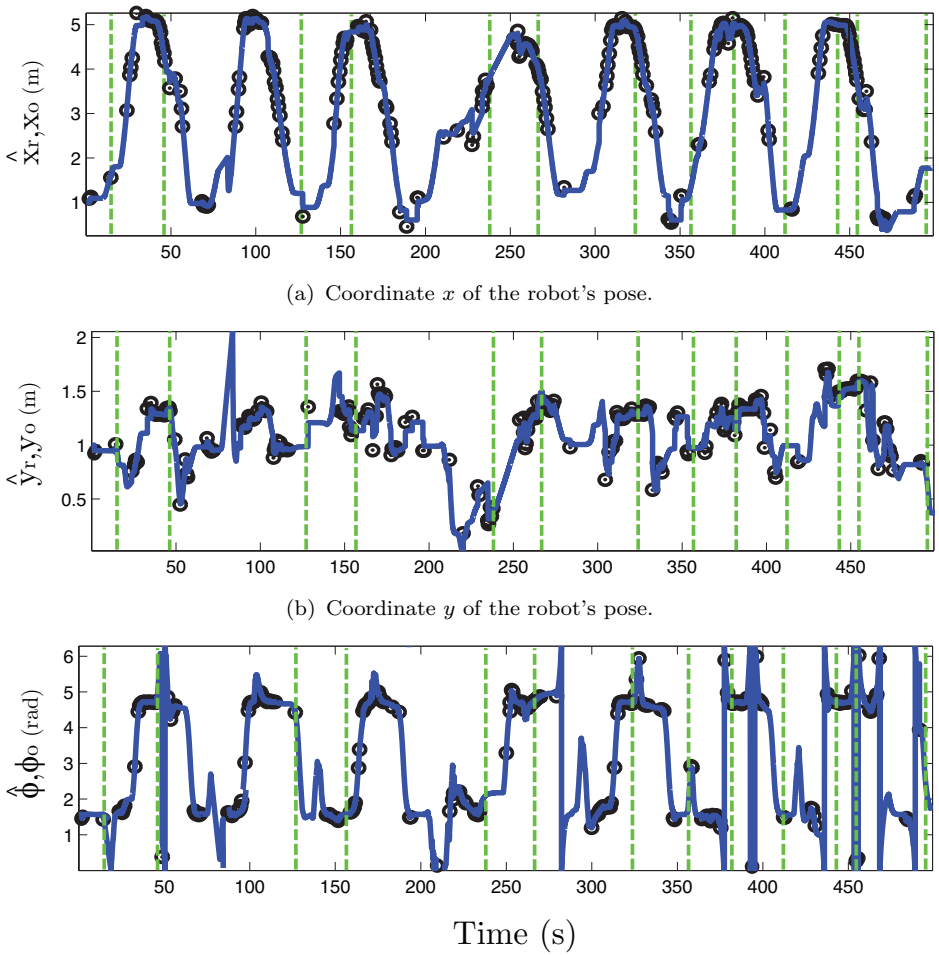
FIGURE 5.14: (a) Minimum distance measured by the laser mounted on the robot (blue line), red dashed line identifies 0.25 m and green circles represent situations in which collisions occurred. (b) Reference velocity, generated by the Timing Control module (blue line) and vertical dashed green lines indicating the collisions.

mounted on the robot about its pose (black circles) (x_o, y_o, ϕ_o) continues to be correctly acquired after the collisions, and the degradation of the localization accuracy is not relevant. Although the robot has to rotate and find a new trajectory to follow after a collision, the localization system is able to keep the robot within an adequate localization uncertainty for navigation.

5.7.3.2 Landmark Fails

During missions, it is expected that some landmarks are not detected, *e.g.*, due to varying lighting conditions. The effect of non-detected landmarks in the localization uncertainty is verified by modifying the available landmarks in the experiments. For each experiment, a percentage of landmarks is wittingly removed, namely, using 100%, 75%, 50%, 25% and 0% of the full set of landmarks. In each experiment, the robot performs a sequence of single missions for approximately 300 m. When the robot reaches a goal location, it immediately assumes a new goal location and moves towards to it.

Fig. 5.16 shows the uncertainty of the localization system when the percentage of detected landmarks changes. The percentage of available landmarks has an impact on the localization uncertainty, *i.e.*, the higher the percentage of available landmarks, the



(a) Coordinate x of the robot's pose.

(b) Coordinate y of the robot's pose.

(c) Orientation ϕ of the robot's pose.

FIGURE 5.15: Estimates of the robot's pose $(\hat{x}_r, \hat{y}_r, \hat{\phi})$ (blue line), data of the robot's pose acquired by the camera (black circles) and instants of time in which collisions occurred (green dashed lines).

lower the average error (green circles). The maximum and minimum numbers (black dotted lines) and the percentile (blue rectangles) show that when fewer landmarks are available, the estimates are more inaccurate. In normal conditions, in which 100% of the landmarks are available for detection, the results are similar to the ones presented in table 5.3.

The robot successfully completes the sequence of missions even when the number of available landmarks is reduced to $75\%, 50\%$, and 25%, albeit with larger uncertainty values. When no landmarks are available (0%) the robot is unable to reset its position based on vision information and therefore unable to complete the sequence of missions. In fact, the robot is lost after moving a few meters.

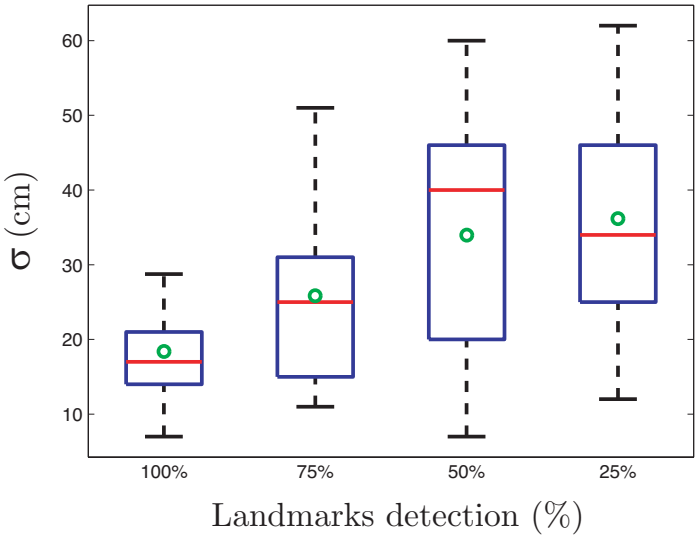


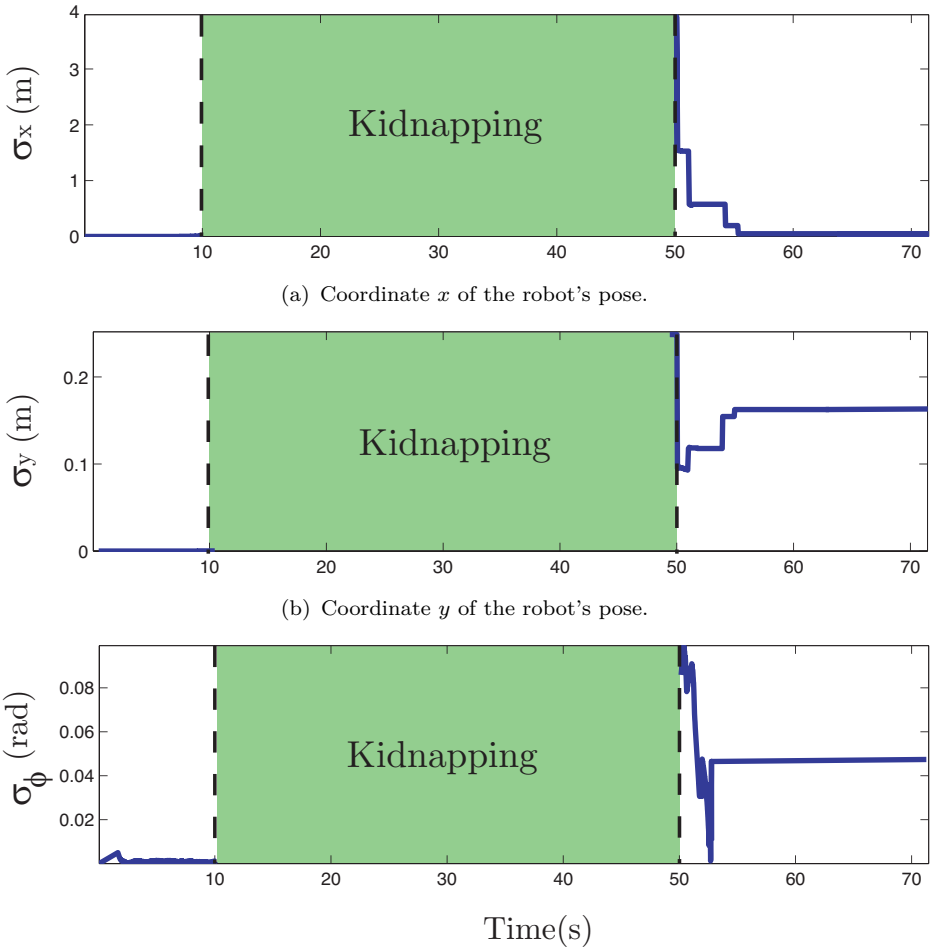
FIGURE 5.16: Uncertainty of the localization system when the percentage of detected landmarks changes. Green circles represent the average error, red lines represent the median, black dotted lines represent the minimum and maximum values and blue rectangles represent the $25\% - 75\%$ percentile.

5.7.3.3 Robot Kidnapping

The kidnapped robot tests assess the ability of the Localization System to recover from catastrophic localization failures.

In these tests, the robot initiates its mission in a known position and remains stopped for 10 s to ensure the convergence of the robot's localization (note that the convergence time is approximately 5 s). Then, the robot is moved to another position and has no vision information during a period of time sufficient for the robot to get lost.

This test is repeated for 10 runs and the average uncertainty between the robot's true position and the estimated robot's position after the kidnapping is 4 m. Fig. 5.17 shows the average uncertainty relative to coordinates x, y and orientation ϕ in the kidnapped test. Green area represents the average interval of time in which the robot is being kidnapped. When vision information is available again, coordinates x, y of the robot's position converge to the expected uncertainty (0.16 m according to table 5.3) in approximately 5 s (see fig. 5.17 (b), which is zoomed during the interval time of convergence, $50 \text{ s} < t < 72 \text{ s}$). Relative to the orientation, it converges to the expected uncertainty (0.05 rad according to table 5.3) in approximately 1 s (see fig. 5.17 (c)). These values suggest that the convergence of the robot's pose given by the Localization System is consistent to those obtained in table 5.3, even after a catastrophic failure.



(a) Coordinate x of the robot's pose.

(b) Coordinate y of the robot's pose.

(c) Orientation ϕ of the robot's pose.

FIGURE 5.17: Uncertainty *vs* time: convergence of the localization system when the robot is kidnapped during an interval of time (green area).

5.8 External module

The External module represents the interface over which the user defines the missions for the robot (see fig. 5.18). For instance, the mission of the robot might be initiated via the human interface, and the required information should be provided to the robot. The user, which might represent a nurse or an auxiliary, defines the destination of a single mission, P_g, or a set of sequential destinations, in case of multiple missions. For each mission, a correspondent time constraint MT must be specified. In cases where a time constraint is not assigned by the user, the robot can reasonable calculate a time constraint for the mission.

In this work, the map is provided to the robot through a topological representation in the form of a transition function M. Nevertheless, the specification of the map could be done by CAD files [58], or by building the map through the robot's onboard laser by moving the robot along the environment.

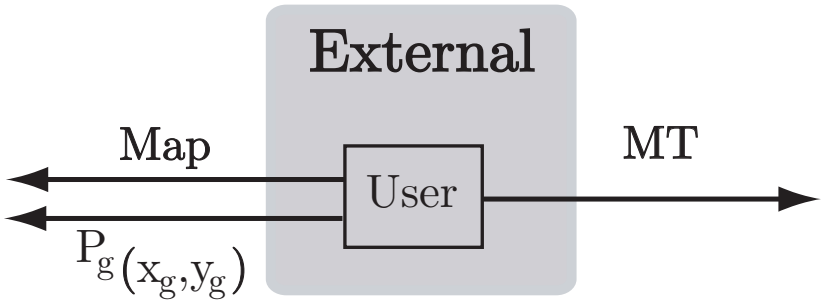


FIGURE 5.18: Schematic of the External module that represents the interface between the user and the robot.

Chapter 6

Experiments

This chapter presents a set of simulations and real experiments to illustrate the ability of the architecture to guide the robot while satisfying a time constraint. Furthermore, the chapter discusses a dependability analysis about the robustness of the proposed architecture, considering the reliability, integrity and safety of the robot. It is also expected that the stability indicator based on the Contraction Theory identifies the failed and successful missions.

Even though the contraction analysis suggests that the whole system is dependable, real experiments allow a realistic assessment. However, there are multiple reasons why simulations are useful to develop and test a control architecture. These include, (1) reduction of the time to develop the robot control code; (2) enabling real-time testing of complex control algorithms; (3) avoiding unnecessary damage to the robot equipment when testing new control strategies or stability solutions; (4) studying the robot behavior in complex systems without having to build them.

The real experiments are conducted in a typical university indoor environment, but the obtained results can be considered to hospital environments. In both environments, the robot should deal with people, whose behavior is unpredictable, avoid static obstacles and navigate through narrow passages.

Both simulations and real experiments include long-term missions, which allow verifying if the proposed architecture is able to drive a mobile robot for large periods of time in realistic environments. These long-term missions consist of multiple single missions performed sequentially. When the robot reaches a goal location, the current mission is completed, a new one is selected and the robot starts moving towards its goal location. Conducting long-term experiments allows a more meaningful analysis of the architecture, and hence, a truer representation of the localization performance accuracy. Delivery autonomous robots must cover distances in the order of kilometers and maintain an accurate information about their location.

The robot must know the map of the environment before performing any mission. Furthermore, in the beginning of each mission, the goal location where the mission ends, P_g, and its time constraint, MT, are provided to the robot. The time constraint varies in each mission, and is empirically set to allow the robot to complete the respective mission using an average velocity of approximately 0.4 m/s. Before of each mission, the robot waits a time interval, used for rotating towards the goal location. In the following simulations and experiments, this time interval is defined as $t_{\text{init}} = 8$ s. In the case that the robot is unable to complete its mission within the time constraint, *i.e.*, the robot can not reach the goal location even moving at its maximum velocity, $v_{\max} = 0.8$ m/s, the rescue behavior must be activated. The condition that activates this behavior and deactivates the others is given in (3.69). A mission is successful completed when the robot reaches the neighborhood of a goal location, *i.e.*, the remaining distance is less than $\epsilon = 0.4$ m. This is a suitable distance that allows the robot to stop safely and close to the goal location, despite the error no localization. As an emergency condition, the robot stops immediately when an obstacle is detected at a distance less than 25 cm.

Table 6.1 depicts the goal of the simulations and experiments included in this chapter. The two simulations aim at showing the behavior of the robot in a simulated hospital environment. In the first four real experiments, the robot has to deal with typical situations of dynamic and cluttered environments, such as static and dynamic obstacles. In experiments 5 and 6, a person intentionally obstructs the trajectory of the robot. In experiments 7, 8 and 9, the robot faces catastrophic localization failures, since it is kidnapped during its missions. In experiment 10, the robot navigates through a long narrow passage and in the last two experiments, the robot performs long-term missions.

TABLE 6.1: Summary of the simulations and experiments.

Identification	Main goals
Simulation 1	Robot has to handle typical hospital obstacles in 3 missions
Simulation 2	Robot has to perform multiple single missions during a working day of approximately 9.5 hours
Experiment 1	Single mission without obstacles
Experiment 2	Single mission with static obstacles
Experiment 3	Single mission with narrow passages, static and dynamic obstacles
Experiment 4	Robot has to move through a door while facing the same kind of obstacles of experiment 3
Experiment 5	Person prevents the robot to complete its mission with success
Experiment 6	Person obstructs the robot, but it succeeds to complete the mission
Experiment 7	Robot performs two single missions and is kidnapped during the second mission
Experiment 8	Robot is kidnapped during a mission, in which it has to move through a door
Experiment 9	Robot performs two single missions and does not recover from a kidnapping
Experiment 10	Robot must navigate through a long narrow passage
Experiment 11	Robot performs a long-term mission consisting on sequential single missions
Experiment 12	Robot performs a long-term mission whose locations are randomly generated

6.1 Simulation Experiments

The environment used for the simulations is the hospital environment shown in fig. 5.6. The obstacles that the robot will face during its missions are the ones illustrated in fig. 5.7. Pedestrians are simulated with a velocity of 1.2 m/s [298], when no other pedestrian or obstacle is in the surroundings. Otherwise, the velocity of the pedestrian decreases to 0.8 m/s [299].

6.1.1 Simulation 1

This simulation shows the behavior of the robot when it has to perform three missions in a hospital environment. These missions are sufficient to show the robot handling typical hospital situations, as avoiding static and dynamic obstacles or moving through

narrow passages. It is expected that the robot completes its missions within due time, 80 s for the first two missions and 120 s for the last one. Furthermore, the stability indicator should identify the success of the three missions.

Fig. 6.1 shows situations that the robot faces during these three missions. Snapshots A, B and C concern the first mission, D and E the second mission and F, G and H the third mission. Snapshots A, D, F, G and H show the robot avoiding people and other static obstacles. Snapshots B, C and F illustrate the robot moving in narrow passages, created by obstacles or doors. Snapshot E shows the robot reaching the goal location of the second mission, identified by the red circle.

FIGURE 6.1: Different situations faced by the robot during simulation 1. The robot avoids people (snapshots A, D, F, G and H), static obstacles (snapshots B and D) and navigates through narrow passages (snapshots B, C, D and F).

Fig. 6.2 (a) shows the solutions m (continuous blue line) and n (red dashed line), generated by the Stuart-Landau oscillator. Note that solution m is directly used to control the robot's linear velocity, v, according to (3.41). Solution n is required to enable the oscillator to undergo periodic motion. In three situations, identified by the green ellipses, the amplitude of solution m is reduced through the velocity decreasing condition (3.61), in order to ensure a safe circumnavigation of obstacles. The second and third situations in which the solution m decreases can be viewed in snapshots F and G, respectively. After the obstacle circumnavigation, the robot's linear velocity increases to compensate for the delay caused by the obstacles. Fig. 6.2 (b) illustrates the remaining distance between the robot and the goal location during the three missions, D (black continuous line). The red dashed line stands for the threshold ($\epsilon = 0.4$ m)

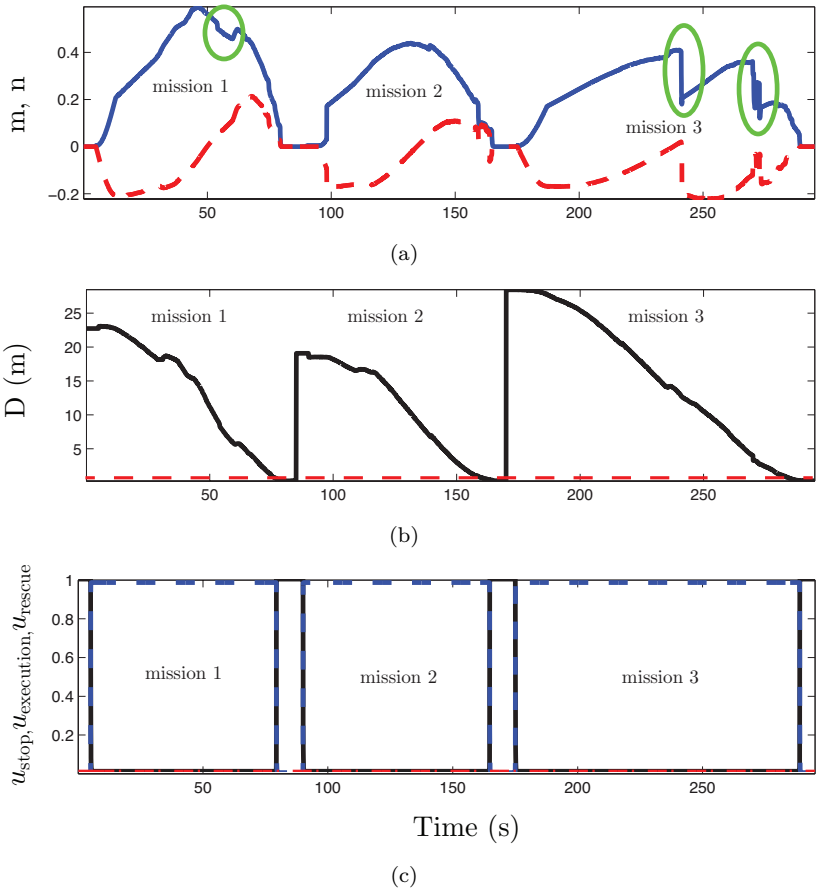


FIGURE 6.2: (a) Solutions m (blue continuous line) and n (red dashed line) generated by the Stuart-Landau oscillator. (b) Remaining distance between the robot and the goal location, D (black continuous line), during the 3 missions. Red dashed line identifies the distance between the robot and the goal location over which the mission is considered finished. (c) Variables responsible for the robot's motor behavior. When u_{stop} (black continuous line) is activated, the robot is stopped; when $u_{\text{execution}}$ (blue dashed line) is activated, the robot is performing the mission; u_{rescue} (dashed-dotted line) was never activated, since the robot was able to complete its missions within their respective time constraints.

that identifies when the robot reaches the goal location. In the three missions, the robot reaches the goal location, as the remaining distance is lower than ϵ after the time constraint has been elapsed, $D < \epsilon$. In fact, the robot takes 76 s, 75 s and 113 s to reach, respectively, the goal location of the first, second and third mission. Thus, it completes the three missions within due time. The time the robot takes to complete the mission could be closer to the respective time constraint if ϵ was smaller. However, this would require a more accurate localization system. The total distance covered by the robot is approximately 80 m. Fig. 6.2 (c) depicts the variables responsible for the robot's motor behavior. In the beginning of the first mission, variable u_{stop} (black continuous line) is activated and variables $u_{\text{execution}}$ (blue dashed line) and u_{rescue} (red

dashed-dotted line) are deactivated. Consequently, the robot is stopped. When t_{init} (waiting time in the beginning of each mission) has been elapsed, the variable $u_{\text{execution}}$ is activated and u_{stop} is deactivated. Thus, the robot starts its movement towards the goal location. When the robot reaches the goal location, variable u_{stop} is activated again and $u_{\text{execution}}$ is deactivated. Accordingly, the robot stops moving. The same procedure is repeated for the other 2 missions. As the robot reaches the goal locations of the 3 missions within the time constraint, the variable u_{rescue} is never activated.

Fig. 6.3 shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$. Condition (4.5), $\|D(f_{\text{supervisor}})\| << 1$, holds for the three missions. This means that the robot reaches the three goal locations within their respective time constraints, MT. As expected, the stability indicator is sufficient to identify the successful missions. Despite the several static and dynamic obstacles detected by the robot during the three missions, the robot is able to successfully complete them.

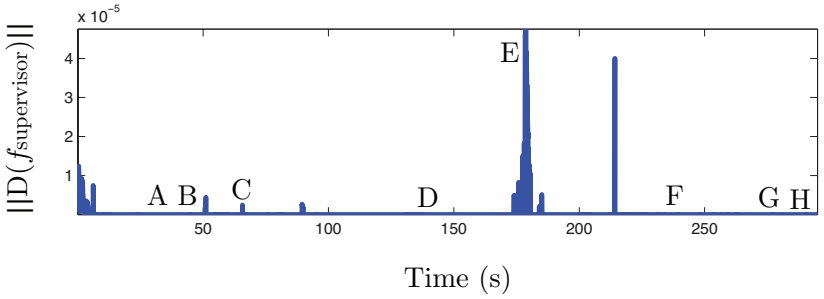


FIGURE 6.3: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the three missions of simulation 1. Capital letters identify the snapshots illustrated in fig. 6.1.

Fig. 6.4 depicts the trajectory followed by the robot. The ground truth, representing the real trajectory that the robot covers, is depicted by the blue line. The position of the robot obtained by the robot's vision system is depicted by the green circles. Red line stands for the trajectory calculated by odometry. The trajectory estimated by the EKF is depicted through the yellow line. Black circles represent the goal locations of the 3 missions. The robot starts its missions at location $(2, 2)$, identified by the red cross. The goal location of the first mission is $(14, 4)$. When the robot reaches this location, a new mission is defined, whose goal location is $(1, 8)$. Once this mission is completed, the next one, whose goal location is $(13, 15.5)$, is started. Black arrows show the direction of the trajectory followed by the robot. Numbers stand for the identification number of the missions.

Fig. 6.5 illustrates the trajectory estimated by the EKF followed by the robot. The closer the points, the slower the robot moves. Note that mission 2 is shorter than mission 1 (see fig. 6.2), but the time constraint is the same. Consequently, during

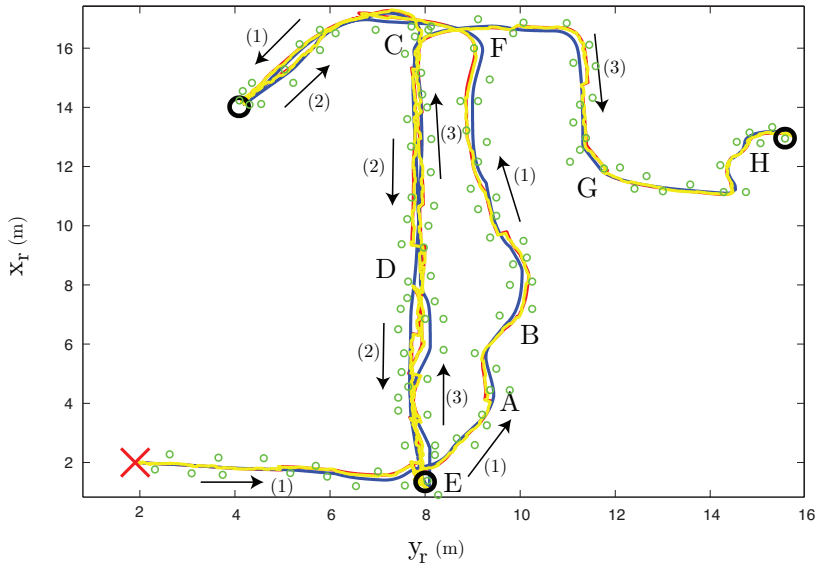


FIGURE 6.4: Trajectory followed by the robot during simulation 1. Yellow line represents the estimated trajectory through the EKF. Green circles represent the robot's position obtained through the robot's vision system. Red line represents the trajectory calculated by odometry and blue line represents the ground truth. Red cross represents the initial position of the robot in simulation 1. Black circles stand for the goal locations. Black arrows indicate the direction followed by the robot during the respective mission. Capital letters identify snapshots illustrated in fig. 6.1.

mission 2, the robot follows at a lower velocity than in mission 1. Thus, the points are closer.

Fig.6.6 shows the error between the ground truth of the robot's position and the one estimated by the EKF. During the sequence of missions, the maximum error is approximately 0.40 m. This error is small enough to allow the robot to complete its missions with success. Green dashed circles illustrate some instants of time in which a reset operation to the Localization System is performed. Note that in the reset operation, the robot's pose provided by the vision system has a larger weight than odometry to the estimates of the EKF. Furthermore, odometry data is reset with the vision data.

6.1.2 Simulation 2

In this simulation, the robot performs an interrupt sequence of missions during a real working day of approximately 9.5 hours. This sequence consists of 9 single missions. When the robot completes one mission, a new one is defined and the robot starts moving towards the goal location of the new mission. The robot repeats 400 times the sequence of 9 single missions, covering approximately 8400 m.

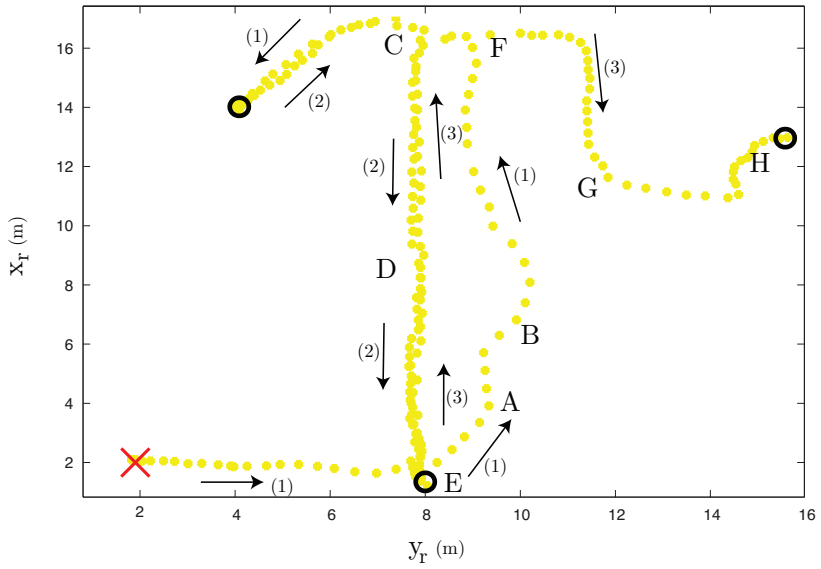


FIGURE 6.5: Estimates given by the EKF of the trajectory followed by the robot in simulation 1.

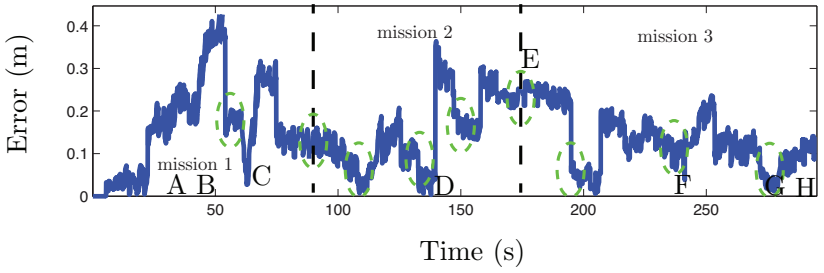


FIGURE 6.6: Error between the ground truth of the robot's position and the one estimated by the EKF. Capital letters identify the snapshots illustrated in fig. 6.1.

Fig. 6.7 illustrates the trajectory covered by the robot estimated through the EKF. The robot starts the sequence of missions at location $(14, 4)$, identified by the red cross. When the robot reaches the goal location of the first mission, located at $(1, 8)$, it assumes a new mission and starts moving towards the goal location of mission 2, located at $(13, 16)$. The robot performs the remaining missions until it completes mission 9. When mission 9 is completed, the robot starts moving towards the goal location of mission 1, and thus repeating the sequence. In this simulation, the robot faces similar obstacles identified in simulation 1 (see fig. 6.1). It is expected that the robot completes all missions within their respective time constraints, such that the schedule of the delivery missions is completed without delay.

Fig. 6.8 (a) shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the sequence of missions. Its upper bound does not verify condition (4.5) in two missions, which means that the robot is unable to complete them within due time. Fig. 6.8 (b) illustrates the

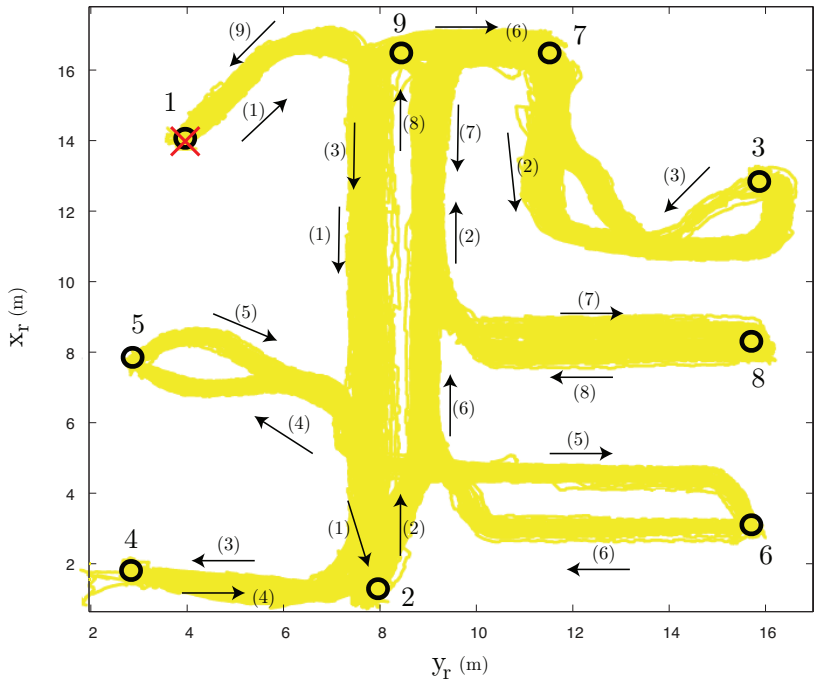


FIGURE 6.7: Trajectory followed by the robot during simulation 2. Red cross represents the initial position of the robot. Black circles stand for the goal locations. Numbers represent the sequence of missions that the robot has to perform. Black arrows indicate the direction followed by the robot during the respective mission.

sequence of f_i modules, $\prod_{i=1}^{17} \|D(f_i)\|$ (condition (4.9)), that represents the stability of the architecture without the time constraint. Its upper bound is less than 1, meaning that the architecture is able to guide the robot towards its goal locations. Consequently, the parameters illustrated in table 4.1 were fulfilled.

These two missions are failed, because a configuration of obstacles created by people forces the robot to reduce its velocity and prevents it to follow its trajectory. Fig. 6.9 illustrates the trajectory covered by the robot estimated through the EKF during the two failed missions. The first failed mission is started in goal location 2 and the robot follows towards goal location 3. The mission fails when the robot is close to the goal location 3 (green dashed circle). A person obstructs the trajectory of the robot and it is unable to complete the mission in due time. In the second failed mission, the robot starts from goal location 3 and must reach goal location 4. However, the robot is obstructed during a large period of time by a person (black dashed circle). When the path is clear, the mission is already failed and the robot follows with the rescue behavior, at 0.1 m/s. Note that the blue points are very close after the black dashed circle, indicating the low velocity of the robot. During the two intervals of time in which the rescue behavior is active, the robot covers 23 m in 218 s. This period of time is the amount of time wasted by the robot to complete the sequence of missions.

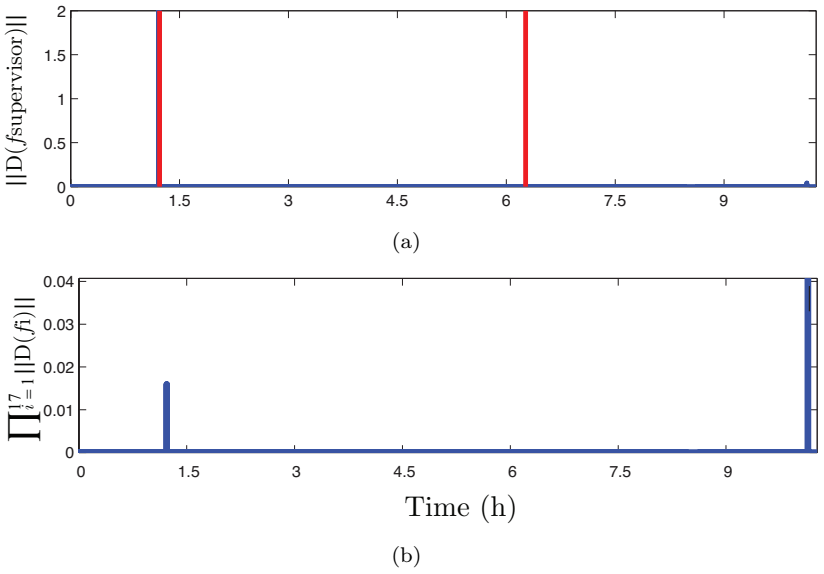


FIGURE 6.8: (a) Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the three missions of simulation 2. Graphic is bounded to 2, but for failed missions $\|D(f_{\text{supervisor}})\|$ is much higher. (b) Evolution of the sequence of f_i Jacobians without the time constraint as in (4.9), $\prod_{i=1}^{17} \|D(f_i)\|$.

However, it could be reduced, if during the rescue behavior, the robot had follow at a higher constant velocity.

Fig. 6.10 shows the robot's linear velocity during simulation 2. The robot moves at different velocities, according to the mission being executed. In some missions, the robot reaches its maximum velocity, 0.8 m/s, in order to compensate for delays. Green circles show the periods in which the robot is in the rescue behavior and follows at a constant velocity of 0.1 m/s. The red dashed line indicates the average velocity followed by the robot during the sequence of missions, 0.24 m/s.

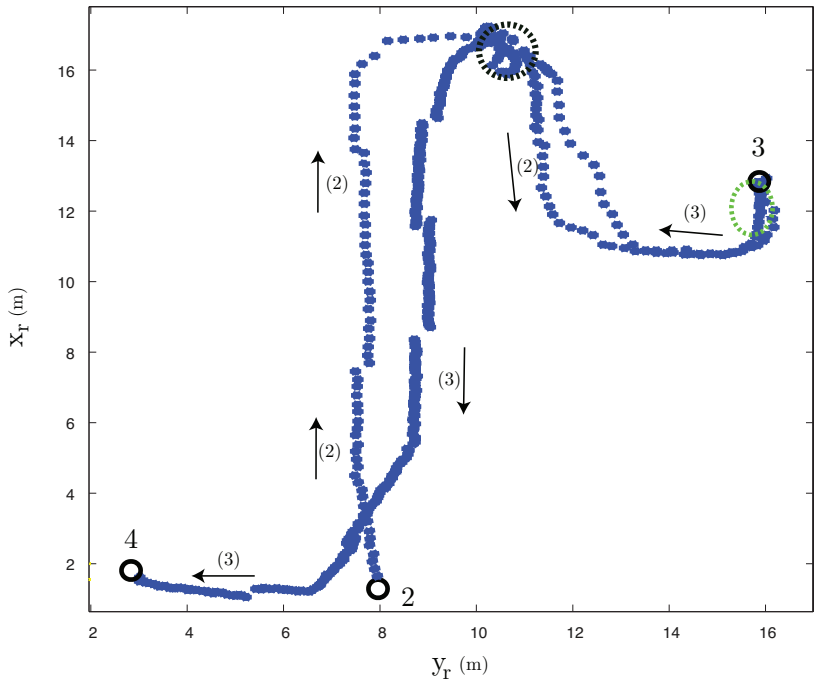


FIGURE 6.9: Trajectory followed by the robot during the two failed missions of simulation 2. Black circles stand for the goal locations. Numbers represent the sequence of missions that the robot performs.

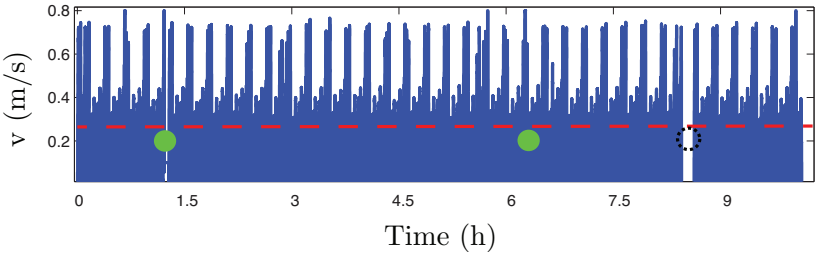


FIGURE 6.10: Robot's linear velocity, v, during simulation 2. Green circles represent periods of time when the robot is in the rescue behavior. Black dashed circle represents a period of time in which the robot is stopped due to an obstacle configuration. Red dashed line depicts the average robot's linear velocity, 0.24 m/s.

6.2 Real Experiments

Several single missions are performed in a cluttered and uncertainty environment to verify the ability of the robot to handle static and dynamic obstacles (people), to navigate in narrow passages, to pass through doors and the ability of the robot's localization system to recover from catastrophic failures. Furthermore, it is verified if the disturbances of the environment affect the stability of the global system. Long-term missions are performed to verify the ability of the architecture to guide the mobile robot with an accurate localization.

6.2.1 Experiment 1

In the first experiment performed on the real environment, the robot has to complete a single mission without unexpected obstacles. The initial distance between the robot and the goal location is 4 m, and the robot has 10 s to complete the mission. It is expected that the robot follows a straight line to complete the mission. Fig. 6.11 shows some snapshots of this mission. For visual purposes, a blue square on the floor identifies the goal location.

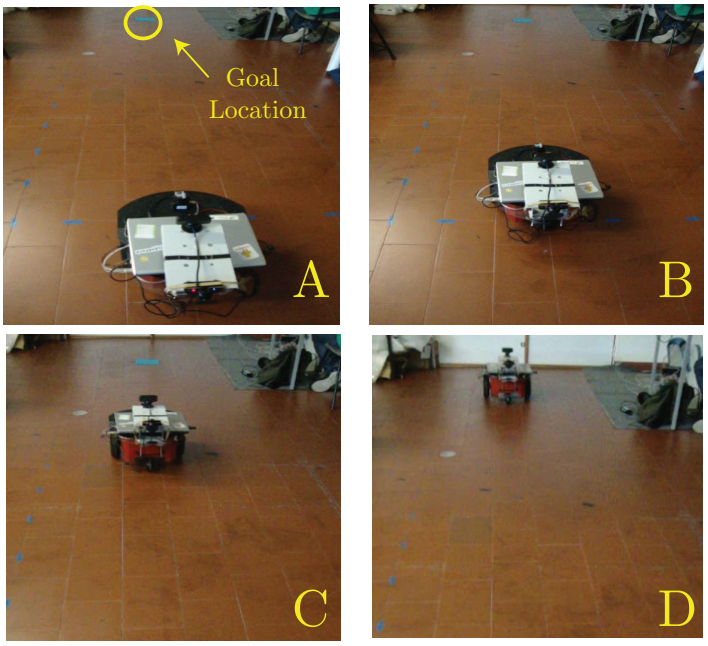


FIGURE 6.11: Snapshots of the robot performing the mission of experiment 1.

Fig. 6.12 (a) shows the solutions m (blue continuous line) and n (red dashed line), and the amplitude, A (black dashed-dotted line), of the Stuart-Landau oscillator. During the first 8 s, when $t < t_{\text{init}}$, the robot does not move and only rotates towards the goal location. The amplitude A is 0 and consequently $m = 0$. After this interval of time, the robot moves to the goal location and reaches a maximum velocity of approximately 0.7 m/s. Note that during the interval of time, $8 < t < 9$ s, the robot's linear velocity increases quickly. This is a consequence of the angular frequency of the oscillator defined in (3.43) and selected in (3.58). Fig. 6.12 (b) illustrates the distance between the robot and the goal location (black continuous line). The red dashed line shows the distance ($\epsilon = 0.4$ m) in which it is assumed the robot has reached the goal location. As expected, the robot reaches the goal location within the time constraint, $MT = 10$ s (note that 8 s are wasted in the beginning of the mission). In fact, the

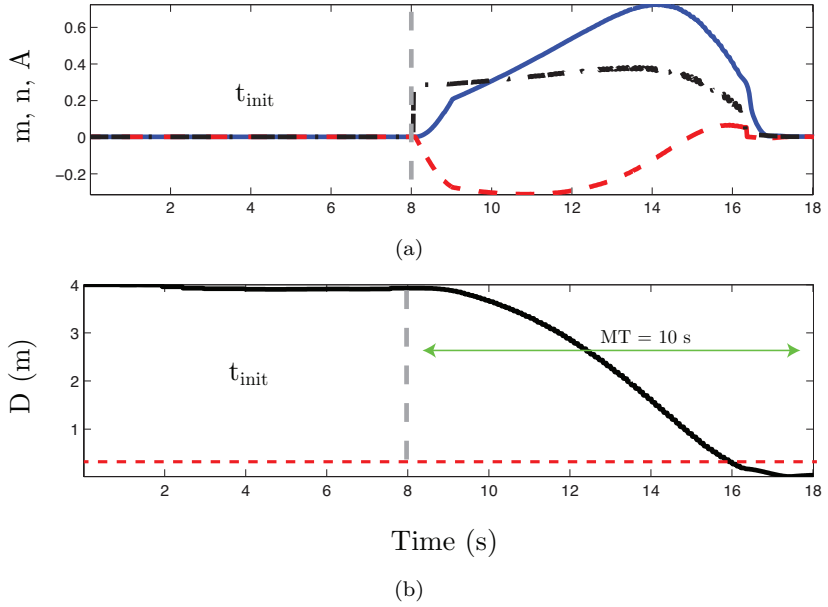


FIGURE 6.12: (a) Solutions of the Stuart-Landau oscillator m (blue continuous line), n (red dashed line) and the amplitude of the oscillator, A (black dashed-dotted line). (b) Distance between the robot and the goal location (black continuous line) and distance in which it is assumed that the robot reaches the goal location (red dashed line), $D < 0.4$ m.

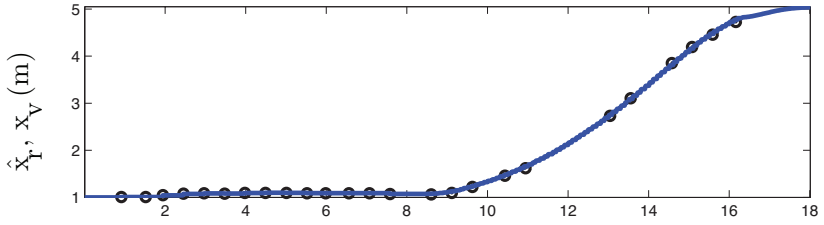
robot takes 8 s to reach the goal location. Despite the mission is considered completed, the robot continues moving due to inertial forces.

Fig. 6.13 illustrates the estimates of the robot's pose, $(\hat{x}_\mathrm{r}, \hat{y}_\mathrm{r}, \hat{\phi})$ (blue continuous line) estimated by the EKF and the robot's pose provided by the camera mounted on top of the robot $(x_\mathrm{v}, y_\mathrm{v}, \phi_\mathrm{v})$ (black circles). The robot starts the mission at location $(1, 1)$ and must reach the goal location at $(5, 1)$. As the robot follows a straight trajectory, its orientation is approximately constant during the mission, $\hat{\phi} \approx \frac{\pi}{2}$ rad. Vision information is only available when a landmark is detected by the robot's vision system.

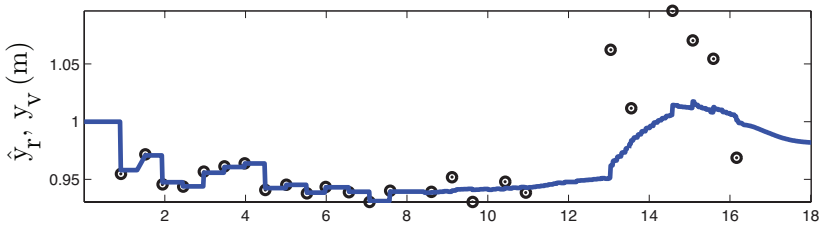
Fig. 6.14 shows the evolution of the bound of $\|D(f_\mathrm{supervisor})\|$. As expected, the stability indicator is well below 1, in agreement with (4.5). This implies that the global system is contracting and converges to the unique fixed point within MT. In practical terms, this condition suggests that the robot successfully completes this mission.

6.2.2 Experiment 2

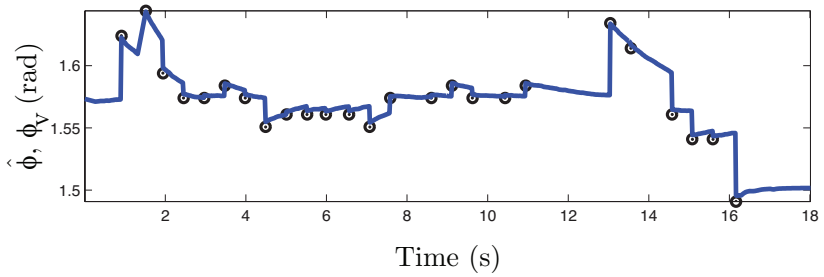
In this experiment, the initial and goal locations of the robot are the same of the previous experiment. However, the robot has to deal with unexpected static obstacles located in the environment. The robot should detect and circumnavigate them to reach the goal location. Consequently, the time constraint for this mission is increased



(a) Coordinate x of the robot's pose.



(b) Coordinate y of the robot's pose.



(c) Orientation ϕ of the robot's pose.

FIGURE 6.13: Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles).

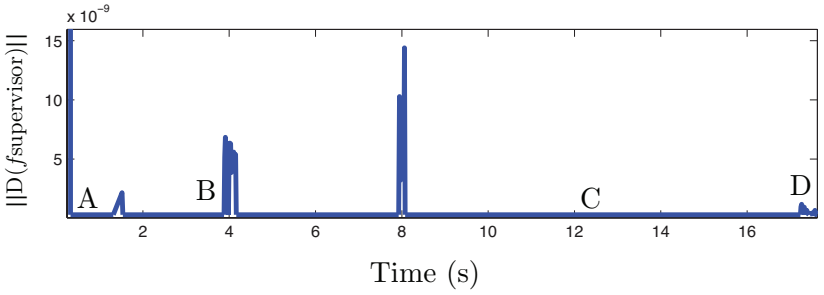


FIGURE 6.14: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 1. Capital letters identify the snapshots illustrated in fig. 6.11

.

to 20 s. It is expected that the robot avoids the obstacles and reaches the goal location within the time constraint. Fig. 6.15 shows some snapshots of this experiment. Panels B and C show the robot avoiding obstacles. In panel D, the robot is reaching the goal location.

Fig. 6.16 illustrates the trajectory followed by the robot along regions $\{r_1, r_2\}$. Red

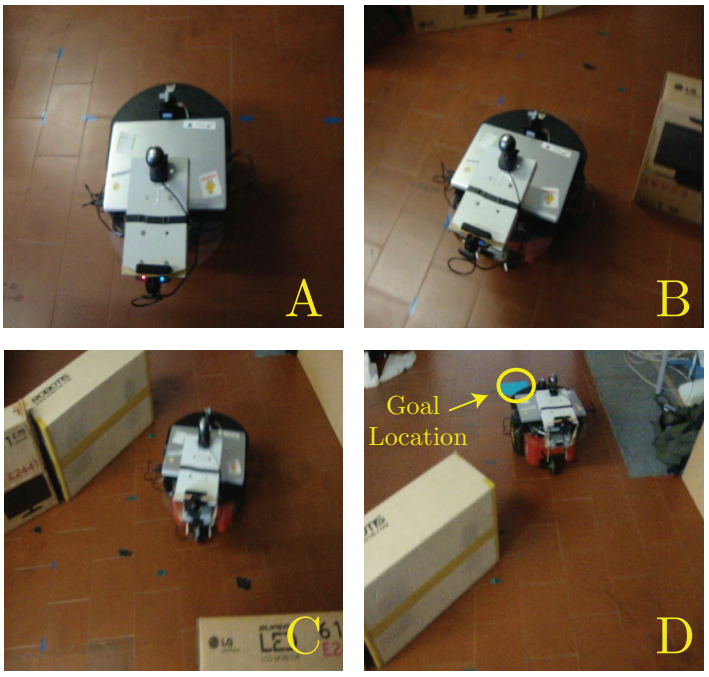


FIGURE 6.15: Snapshots of the robot performing the mission of experiment 2. The robot avoids unexpected static obstacles located in the environment.

circles show the estimates of the robot's position given by the EKF. Black circles represent the robot's position provided by the vision system. When no vision information is available, due to lighting conditions, only the odometry is used to estimate the robot's pose. As expected, the robot circumnavigates the obstacles and reaches the goal location (green cross).

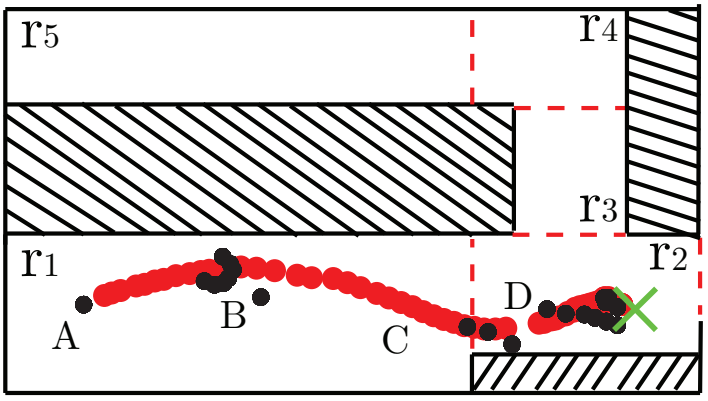


FIGURE 6.16: Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.15 (red circles). Black circles show the robot's position provided by the vision system. Green cross represents the goal location and red dashed lines stand for the critical lines dividing the regions. r_i represents the regions of the environment. Capital letters represent the snapshots shown in fig. 6.15.

Fig. 6.17 (a) illustrates the solution m (blue continuous line) and the amplitude A (black dashed-dotted line) of the Stuart-Landau oscillator. As a consequence of the detected obstacles, the robot has to change its trajectory and cover a larger distance than expected. Thus, to compensate this delay, the amplitude of the oscillator, A, increases and the robot's linear velocity increases as well. The green ellipse shows the interval of time in which the robot's linear velocity increases from 0.2 to 0.4 m/s. When the robot recovers from the delay, its velocity decreases until stopping. Fig. 6.17 (b) shows the

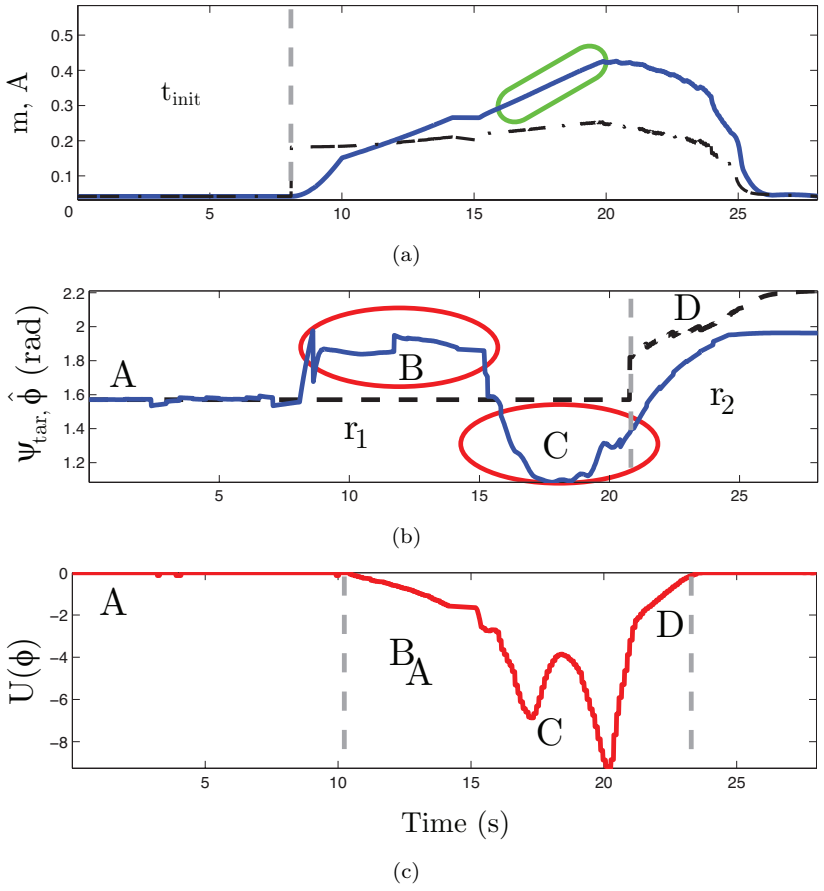


FIGURE 6.17: (a) Solution m (blue continuous line) and amplitude of the Stuart-Landau oscillator, A (black dashed-dotted line). (b) Direction that the robot should follow to reach the goal location, ψ_{tar} (black dashed line), and robot's heading direction, $\hat{\phi}$ (blue continuous line). Red circles show the periods of time in which the robot is avoiding unexpected obstacles and the dashed grey line indicates the transition between regions of the environment. (c) Potential function indicating the presence of obstacles, $U(\phi)$.

direction that the robot should follow, ψ_{tar} (dashed black line), and the robot's heading direction estimated by the EKF, $\hat{\phi}$ (blue continuous line). The red circles show the periods of time in which the robot is avoiding unexpected obstacles. Capital letters indicate the respective panel in fig. 6.15. Despite the detected obstacles in region r_1, the robot must follow $\psi_{\text{tar}} = \frac{\pi}{2}$ rad. After the obstacles circumnavigation, the robot

enters in region r_2, where the goal location is, and ψ_{tar} changes, in order to orientate the robot towards the goal location. As expected, the robot's heading direction follows ψ_{tar} and the robot completes the mission within $MT = 20$ s. Fig. 6.17 (c) shows the potential function, $U(\phi)$, that indicates the presence of obstacles in the surroundings of the robot. The detection of obstacles occurs in the interval $10 < t < 23$ s (bounded by the dashed grey lines), when $U(\phi) < 0$.

Fig. 6.18 shows that the stability indicator holds during the mission. This means that the architecture remains stable and the robot reaches the goal location, P_{g}, within the specified time constraint, $MT = 20$ s. Even though the circumnavigation of the unexpected static obstacles illustrated in panels B and C in fig. 6.15, the robot is able to successfully complete the mission.

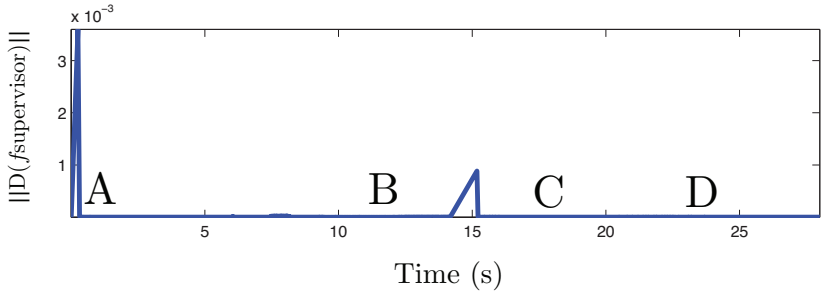


FIGURE 6.18: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 2. Capital letters identify the snapshots illustrated in fig. 6.15.

6.2.3 Experiment 3

In this experiment, the robot must cope with dynamic obstacles (*e.g.* people) (panels A and B in fig. 6.19), unexpected static obstacles (panel A) and narrow passages (snapshots C and D), while completing the mission. The robot starts its mission in region r_1 and the goal location is located in region r_5 (see fig. 6.20). The time constraint, $MT = 40$ s, is sufficient for the robot to complete the mission. The mission evolves along regions $\{r_1, r_2, r_3, r_4, r_5\}$ and people are already familiar with the behavior of the robot.

Fig. 6.20 illustrates the trajectory followed by the robot. The robot successfully reaches the goal location in region r_5 identified by the green cross. The green circle shows an example of the robot's pose reset. The estimates of the EKF in the reset situation rely much more in the robot's vision system data, rather than in the odometry. Consequently, the EKF estimates converge quickly to the robot's pose provided by the vision system. This is achieved by setting $R = R_r$ (see section 5.6.3.4).

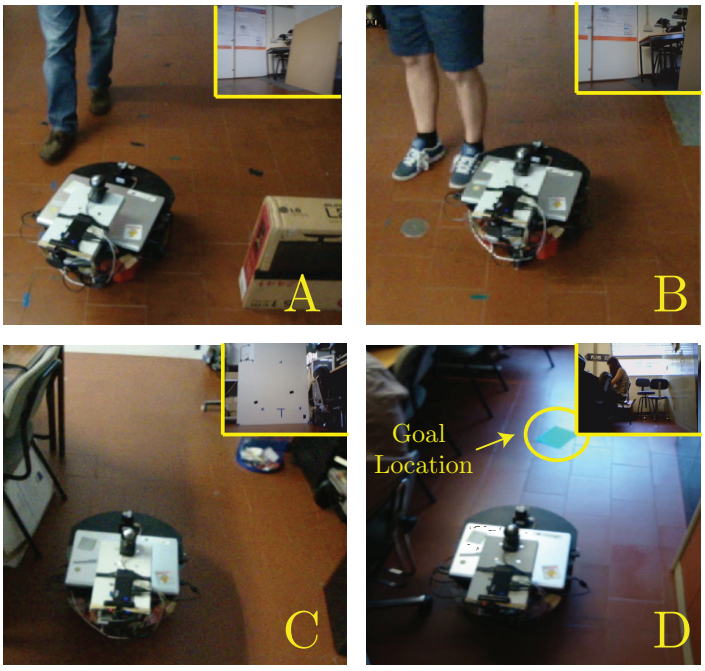


FIGURE 6.19: Snapshots of the robot performing the mission of experiment 3. The robot avoids unexpected static and dynamic obstacles while navigating in narrow corridors. The upper right corners show snapshots captured by the robot of its front.

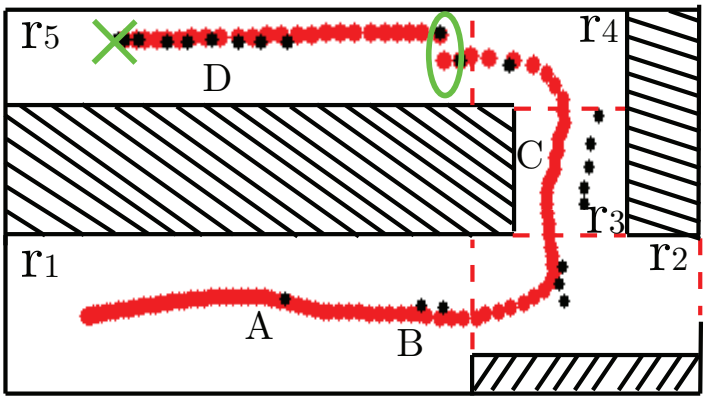


FIGURE 6.20: Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.19 (red circles). Black circles show the robot's position provided by the vision system. Green cross represents the goal location and red dashed lines stand for the critical lines dividing the regions. Capital letters represent the snapshots shown in fig. 6.19.

Fig. 6.21 (a) shows the robot's linear velocity, v (red continuous line), and the solution m of the Stuart-Landau oscillator (blue dashed line). The green dashed circles depict the instants of time in which obstacles force the robot to decrease its velocity to ensure a safe circumnavigation. Capital letters indicate the panels in fig. 6.19. After the obstacle circumnavigation, the robot's linear velocity increases to compensate for

the delay. Fig. 6.21 (b) shows the distance between the robot and the goal location

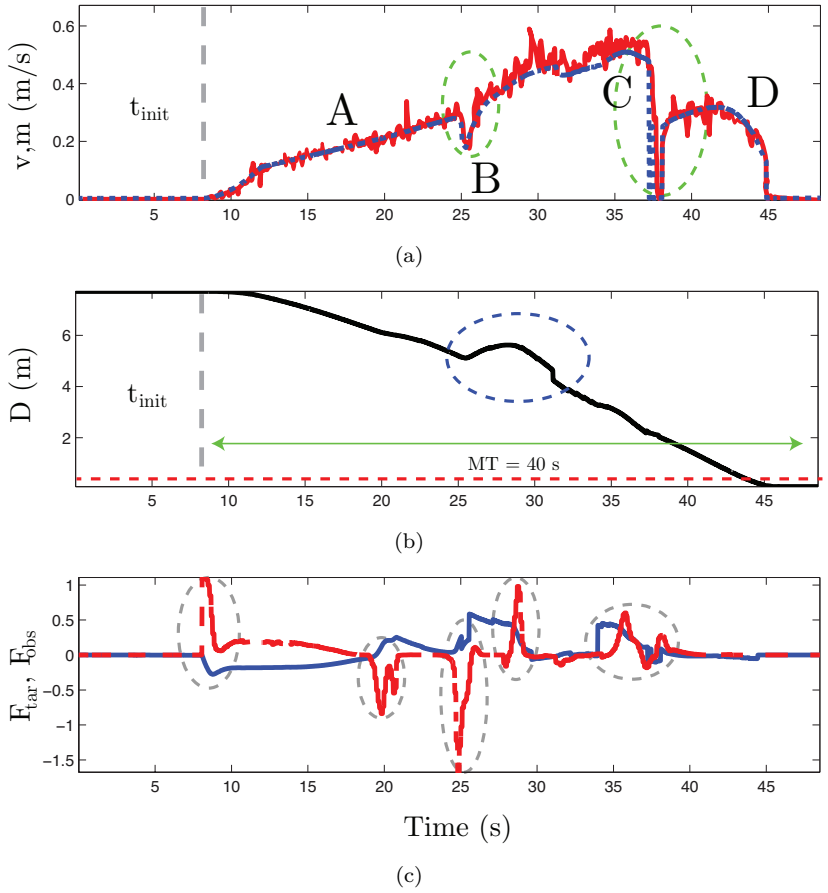


FIGURE 6.21: (a) Velocity followed by the robot, v (red continuous line), and solution m of the Stuart-Landau oscillator (blue continuous line). The green dashed circles depict the instants of time in which obstacles force the robot to decrease its velocity to ensure a safe circumnavigation. Capital letters indicate the panels in fig. 6.19. (b) Distance between the robot and the goal location (black continuous line) and distance in which it is assumed that the robot reaches the goal location (red dashed line), $D < 0.4$ m. The blue dashed circle shows an example of the increasing of the distance caused by the obstacle circumnavigation shown in panel B of fig. 6.19. (c) Target orientation F_{tar} (blue continuous line) and obstacle avoidance F_{obs} (red dashed line) contributions.

(black continuous line). As expected, the robot completes the mission within the time constraint, $MT = 40$ s. In fact, the robot reaches the goal location in 36 s. The blue dashed circle shows a period of time in which the robot has to cover a larger distance than expected, because of an obstacle circumnavigation. Nevertheless, the robot's linear velocity increases to compensate this delay and the robot completes with success its mission. Fig. 6.21 (c) shows the target orientation F_{tar} (blue continuous line) and the obstacle avoidance F_{obs} (red dashed line) contributions. It is noticeable that when obstacles are detected (identified by the grey dashed circles), the absolute value of the obstacle avoidance contribution is larger than the absolute value of the

target orientation contribution, $\|F_{\text{obs}}\| > \|F_{\text{tar}}\|$. This prevents the robot to collide with obstacles when following the direction to the goal location.

Fig. 6.22 shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$. Clearly, it is well

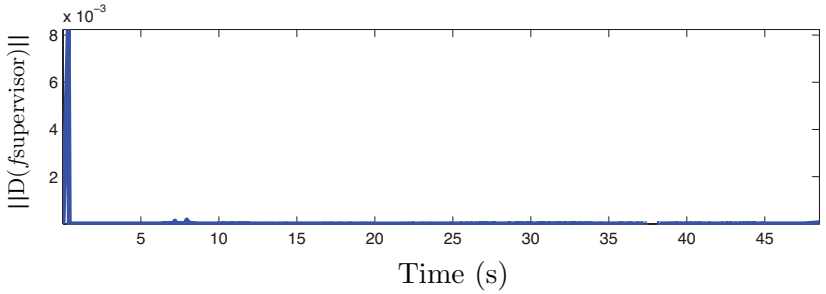


FIGURE 6.22: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 3.

below 1, in agreement with (4.5). This suggests that the global system is contracting and converges to the unique fixed point within MT. This correlates with the mission success, as expected.

6.2.4 Experiment 4

In this experiment, the robot has to move from the laboratory (region r_1) to the goal location in the corridor (region r_6) (see fig. 6.25). In order to complete the mission, the robot has to pass through a door that separates the laboratory from the corridor, while facing the same kind of obstacles detected in experiment 3. Nevertheless, in this experiment, the robot deals with people moving in the corridor who have never interacted with the mobile robot. Despite these disturbances, the robot should complete its mission within $MT = 45$ s. Fig. 6.23 shows snapshots of the robot performing this mission. The robot detects unexpected static obstacles (panel A), passes through a door (panel B) and detects dynamic obstacles in the corridor (panels C and D).

Fig. 6.24 shows snapshots captured by a camera mounted on the robot's front, showing the surroundings of the robot. It is noticeable the people that the robot faces during this mission. Some people in the corridor interact with this robot for the first time and do not know its behavior in cases where obstacles have to be avoided. While some people act cooperatively and avoid staying in front of the robot, other stop in its trajectory to verify if the robot is able to avoid them (see panel 6).

Fig. 6.25 shows the trajectory of the robot estimated by the EKF for this experiment. The robot starts its mission in region r_1, crosses region r_2 and completes its mission in region r_6, where the goal location is located (green cross). In the corridor (region r_6),

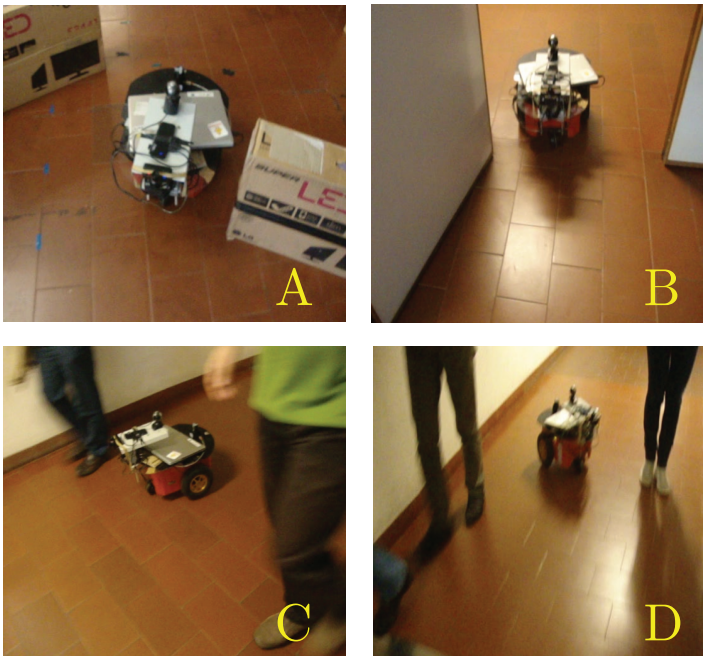


FIGURE 6.23: Snapshots of the robot performing the mission of experiment 4. The robot avoids unexpected static and dynamic obstacles while navigating in narrow corridors.

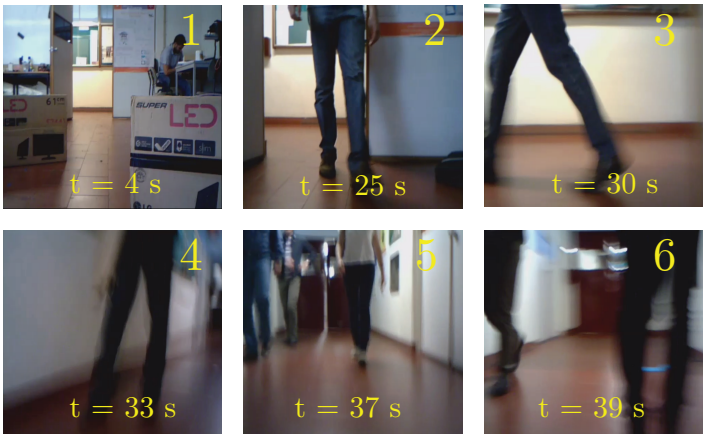


FIGURE 6.24: Snapshots of the surroundings of the robot when it performs its mission in experiment 4. Numbers denote the images captured by the robot's camera.

the red circles are further apart from each other, meaning the robot's linear velocity is higher. The detected people force the robot to cover a larger distance and to reduce its velocity. This delay is compensated for by increasing the robot's linear velocity, v.

Fig. 6.26 (a) shows the robot's linear velocity, v (red continuous line) and the solution m generated by the Stuart-Landau oscillator (blue dashed line). Green dashed lines indicate instants of time in which obstacles are detected and the solution m is decreased to ensure a safe circumnavigation, according to (3.61). Consequently, the

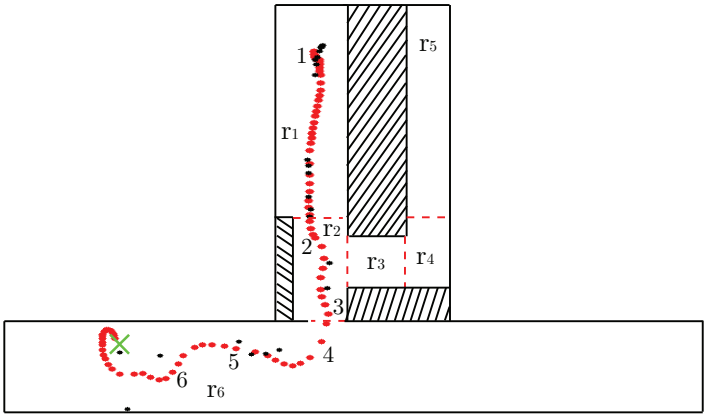


FIGURE 6.25: Estimates given by the EKF of the trajectory followed by the robot during the experiment illustrated in fig. 6.23 (red circles). Black circles show the robot's position provided by the vision system. Green cross represents the goal location and red dashed lines stand for the critical lines dividing the regions. Numbers represent the snapshots shown in fig. 6.24.

robot's linear velocity decreases accordingly. However, due to delays on the servos control, the robot's linear velocity presents a delay relatively to solution m. Fig. 6.26 (b)

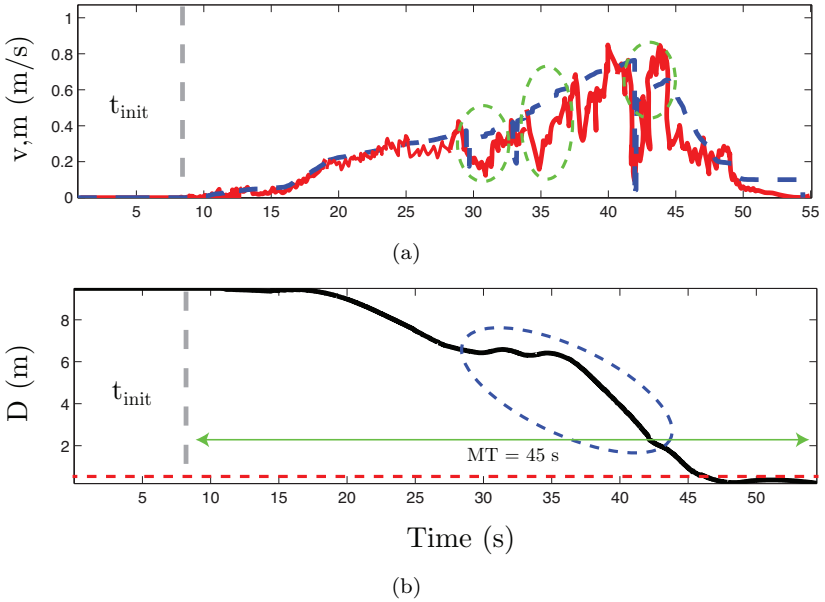


FIGURE 6.26: (a) Robot's linear velocity, v (red continuous line) and solution m generated by the Stuart-Landau oscillator (blue dashed line). (b) Distance between the robot and the goal location, D (black continuous line) and distance in which it is assumed that the robot has reached the goal location (red dashed line), $D < 0.4$ m. Blue dashed ellipse shows the increasing of the distance covered by the robot.

illustrates the distance D. Note the increasing of the distance (blue dashed ellipse) because of the new trajectory that the robot has to follow, in order to avoid the unexpected obstacles. Nevertheless, the robot completes the mission within the time

constraint, $MT = 45$ s. In fact, the robot takes 39 s to reach the goal location. Due to inertial forces, the robot continues moving after reaching the goal location.

The robot reaches the goal location within the time constraint MT, as it is visible in fig. 6.26 (b). Consequently, the upper bound of the stability indicator should be less than 1 during the mission. As expected, fig. 6.27 shows that $\|D(f_{\text{supervisor}})\| < 1$ holds during the mission, meaning that the robot completes with success the mission.

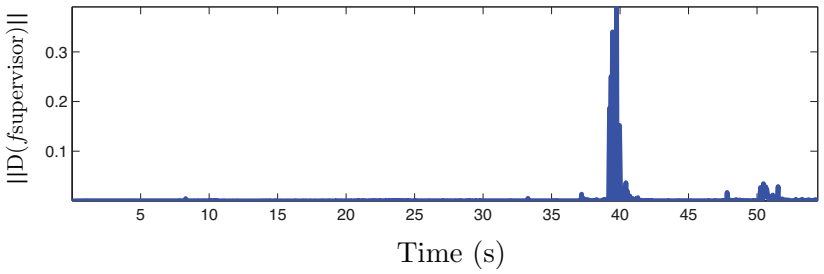


FIGURE 6.27: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the single mission of experiment 4.

6.2.5 Experiment 5

In this experiment, when the robot is completing a mission, a person obstructs the trajectory of the robot, such that it is unable to reach the goal location within the time constraint, $MT = 50$ s. It is expected that the robot activates its rescue behavior and completes the mission with a constant velocity, $v = 0.1$ m/s. Furthermore, the stability indicator must identify this mission as a failed one.

Fig. 6.28 shows snapshots of the robot performing this mission. Panels A, B and C show the person obstructing the trajectory of the robot. The person forces the robot to face the opposite direction to the one that it should follow. When the person stops perturbing the robot, it rotates towards the goal location and follows the suitable trajectory to complete the mission.

Fig. 6.29 (a) shows the robot's linear velocity, v (red continuous line) and the solution m (blue dashed line). Grey dashed line represents the instant of time in which the rescue behavior is activated. The green shadow stands for the period of time in which the person is obstructing the trajectory of the robot. When the person stops perturbing the robot, its velocity increases to the maximum value, 0.8 m/s, in order to compensate the delay caused by the person. However, during the period $38 < t < 42$ s (green dashed circle), the robot is unable to continue following at its maximum velocity as a consequence of a sharp turn. At $t \approx 42$ s, the rescue behavior is activated, as the robot is unable to complete its mission within $MT = 50$ s. Thus,

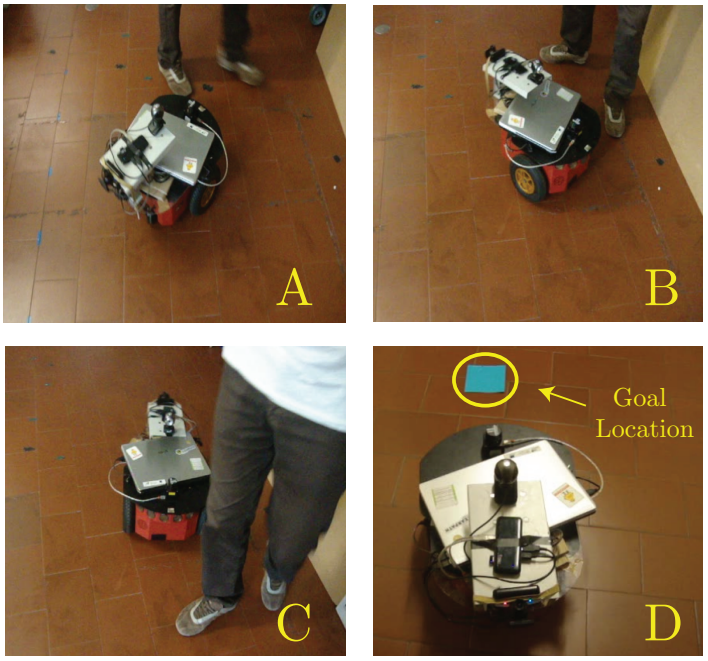


FIGURE 6.28: Snapshots of the robot performing the mission of experiment 5. The robot is performing the mission when a person blocks its trajectory and prevents the robot to follow towards the goal location. Eventually, the person stops obstructing the trajectory of the robot and it is able to move towards the goal location.

the robot covers the remaining distance at a constant velocity, 0.1 m/s. Fig. 6.29 (b) depicts the distance between the robot and the goal location, D. As expected, the robot completes the mission after the time constraint has been elapsed. Fig. 6.29 (c) shows the variables responsible for the behavior of the robot. Initially, when $t < t_{\text{init}}$, $u_{\text{stop}} = 1$ (blue dashed-dotted line), $u_{\text{execution}} = 0$ (red dashed line) and $u_{\text{rescue}} = 0$ (green continuous line). Thus, the robot does not move and rotates towards the goal location. When $t > t_{\text{init}}$, variable u_{stop} is disabled and variable $u_{\text{execution}}$ is activated. Consequently, the robot performs its timed movement. When it is not possible to complete the mission within MT, variable $u_{\text{execution}}$ is disabled and u_{rescue} is activated, such that the robot can complete the mission at a constant velocity.

Fig. 6.30 (a) shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission. As expected, at the moment the robot realizes that it is not possible to complete the mission within MT, the stability indicator is not verified, as condition (4.5) does not hold. Consequently, the contraction of the global system is not verified. The Jacobian of the C^1 feed-through map responsible for verifying the ability of the robot to successfully complete the mission, $D(f_{\text{timing}})$, returns a value sufficiently large to set $\|D(f_{\text{supervisor}})\| >> 1$ when the robot is unable to complete the mission in MT (see fig. 6.30 (b)).

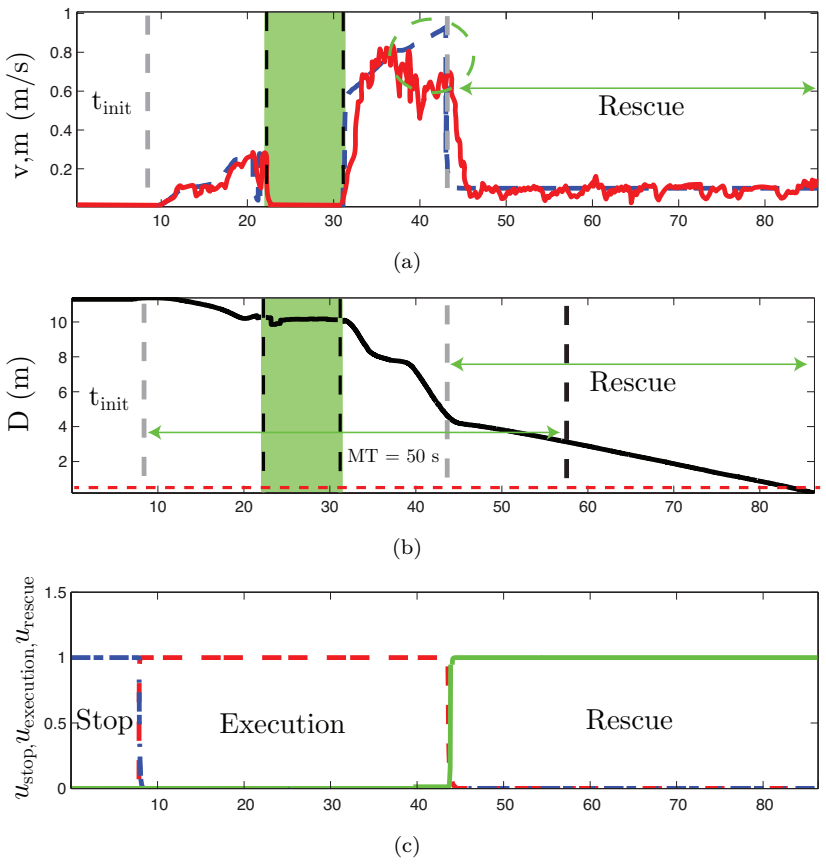


FIGURE 6.29: (a) Robot's linear velocity, v (red continuous line) and solution m (blue dashed line). Green shadow represents the period of time in which the person is obstructing the trajectory of the robot. Green dashed circle shows the period of time in which the robot is unable to follow at its maximum velocity. Grey dashed line depicts the instant of time in which the rescue behavior is activated. (b) Distance between the robot and the goal location, D (black continuous line) and distance in which it is assumed that the robot reaches the goal location (red dashed line), $D < 0.4$ m. (c) Set of variables responsible for the behavior of the robot, u_{stop} (blue dashed-dotted line), $u_{\text{execution}}$ (red dashed line), u_{rescue} (green continuous line).

6.2.6 Experiment 6

In this experiment, a person perturbs the robot during its mission. However, the perturbation caused by the person is insufficient to prevent the robot to complete successfully its mission.

Fig. 6.31 (a) depicts solution m (blue continuous line) and the amplitude of the oscillator, A (black dashed-dotted line). Green dashed circle illustrates the period of time in which the robot is unable to continue its mission due to the movement of the person. During this period, the amplitude decreases to 0, and solution m decreases accordingly to ensure a safe circumnavigation of the obstacle. After the robot has circumnavigated this obstacle, other obstacles are detected, namely at $t \approx 47$ s and $t \approx 57$ s. Fig. 6.31 (b) shows the variables responsible for the behavior of the robot.

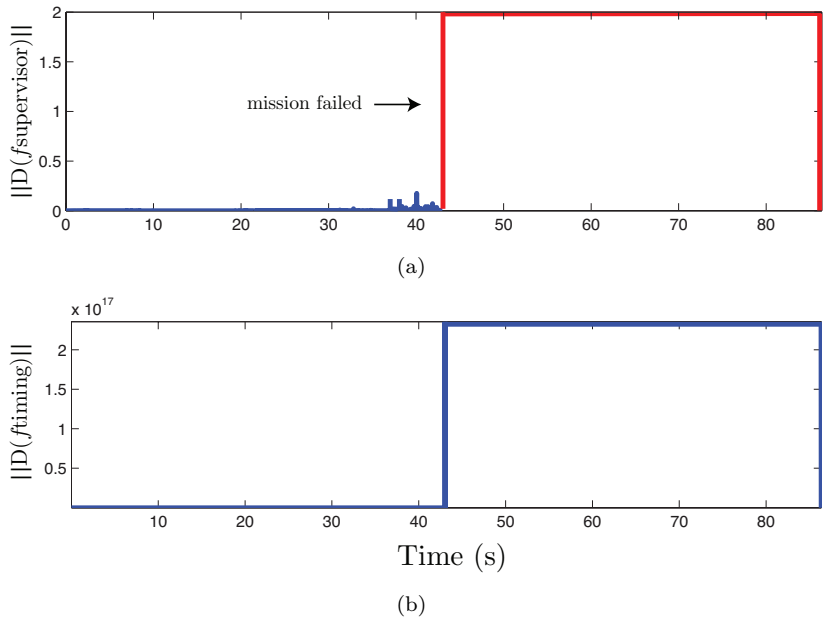


FIGURE 6.30: (a) Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along experiment 5. Values larger than 1 mean that the mission is not successfully completed. Graphic is bounded to 2, but for failed missions, the bound of $\|D(f_{\text{supervisor}})\|$ is much higher. (b) Feed-through map, $D(f_{\text{timing}})$ identifying the nonsuccess of the mission.

Variable u_{rescue} (green continuous line) is never activated, which means that the robot is able to complete the mission within the time constraint, $MT = 50$ s. Variable u_{stop} (blue dashed-dotted line) is activated when the robot starts the mission and when it reaches the goal location. When the robot is performing the timed movement, $u_{\text{execution}}$ (red dashed line) is activated.

The robot is able to complete successfully its mission. Thus, the stability indicator must identify the mission success (see fig. 6.32 (a)). Furthermore, the feed-through map f_{timing} must return 1 during the mission, since the robot is able to complete the mission within MT.

6.2.7 Experiment 7

In this experiment, the robot performs two missions. In the first one, the robot moves from regions r_1 to r_2. When this mission is completed, the robot returns to its initial position in region r_1. The robot should cover 4 m in each mission. The time constraints are $MT = 20$ s and $MT = 40$ s for the first and second missions, respectively. During the second mission, the robot's vision system is disabled and the robot is taken to another position without knowing. When its vision system is activated again, it is expected that the robot localizes itself and completes the mission with success.

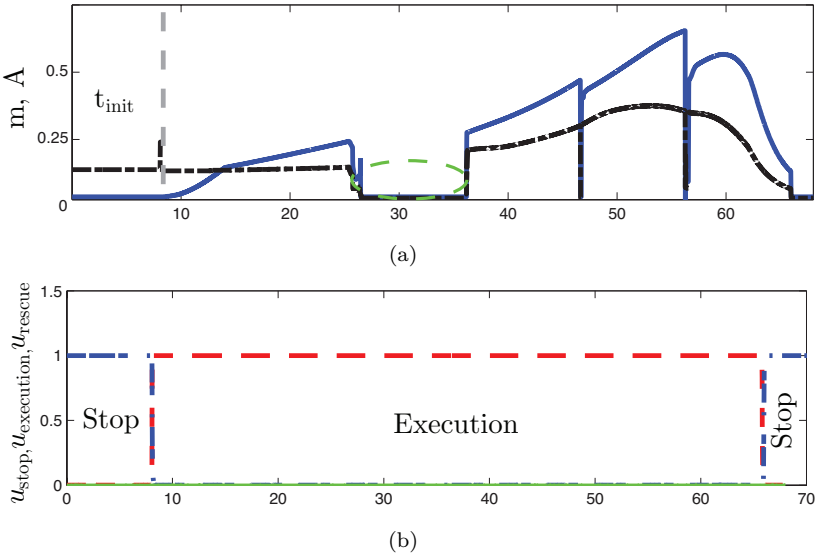


FIGURE 6.31: (a) Solution m (blue continuous line) and the amplitude of the oscillator, A (black dashed-dotted line). Green dashed circle shows the period of time in which the person is disturbing the robot. (b) Set of variables responsible for the behavior of the robot, u_{stop} (blue dashed-dotted line), $u_{\text{execution}}$ (red dashed line), u_{rescue} (green continuous line).

Fig. 6.33 shows snapshots of the robot during this experiment. Panel A shows the robot reaching the goal location (identified by a red square) of the first mission. Panel B shows the robot's vision system being disabled by occluding the camera mounted on the robot. Panel C depicts the robot being kidnapped to the position indicated by

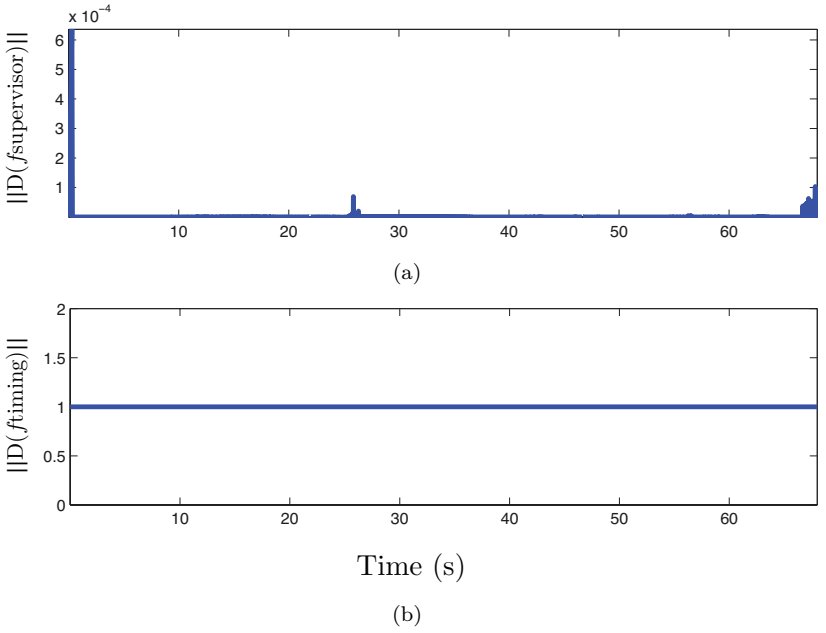


FIGURE 6.32: (a) Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along experiment 6. (b) $\|D(f_{\text{timing}})\|$ identifies if the mission is completed within the time constraint.

the yellow arrow. The last panel shows the robot completing the second mission.

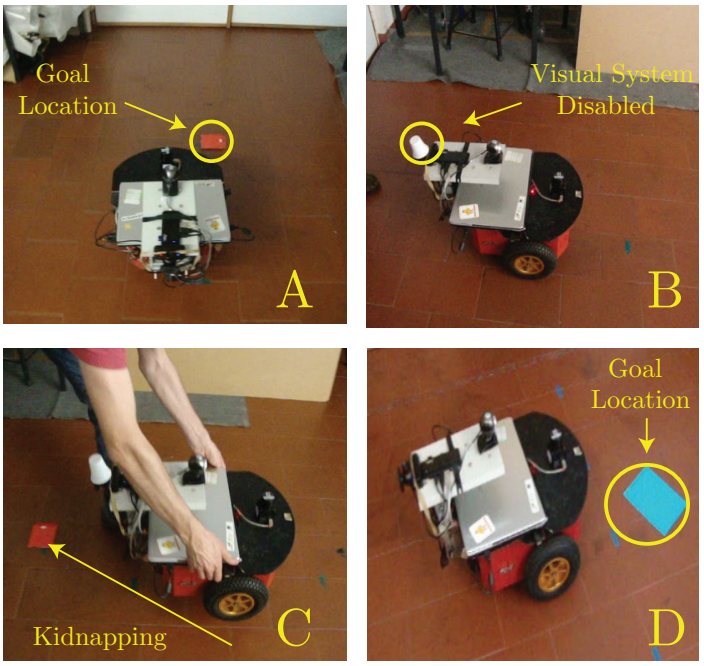
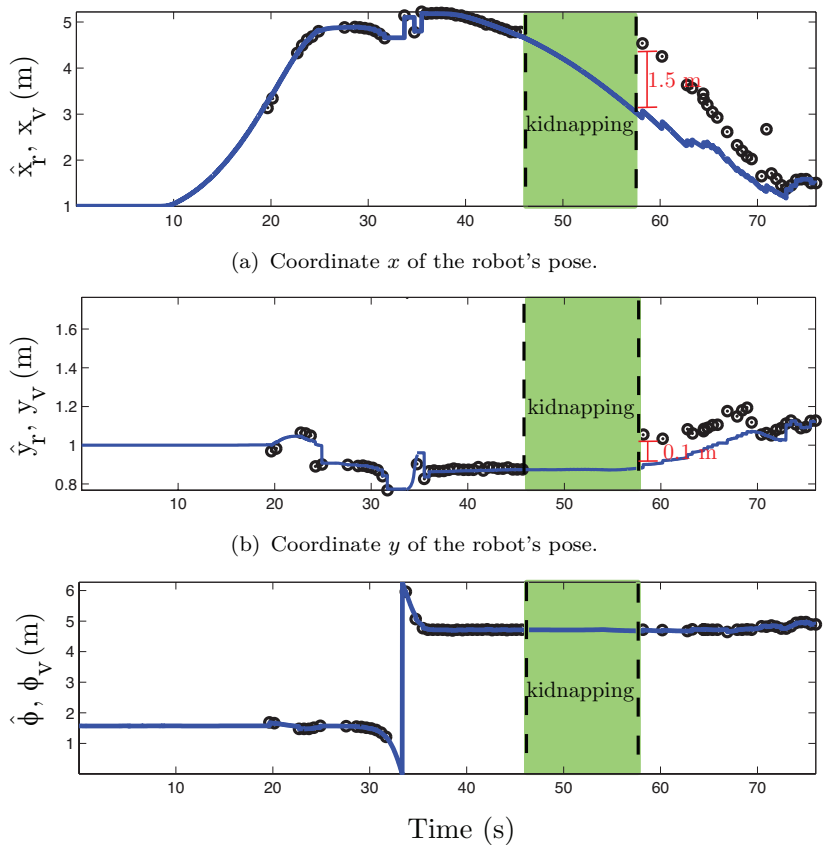


FIGURE 6.33: Snapshots of the robot performing the two missions in experiment 7. The robot is kidnapped during the second mission.

Fig. 6.34 illustrates the estimates of the robot's pose, $(\hat{x}_\mathrm{r}, \hat{y}_\mathrm{r}, \hat{\phi})$ (blue continuous line) and the robot's pose provided by the camera mounted on top of the robot $(x_\mathrm{v}, y_\mathrm{v}, \phi_\mathrm{v})$ (black circles). During the kidnapping (green areas), the robot's vision system is disabled and the robot is taken to another position without knowing. The new position is far from the goal location than the position where the robot was at the time of the kidnapping. The robot's wheels keep rotating and the EKF only relies on the odometry data during the kidnapping period. Therefore, the estimates of the robot's pose are wrong. The difference between the robot's pose estimates and the robot's pose provided by the vision system after the kidnapping is approximately 1.5 m in axis x and 0.1 m in axis y. Relatively to the robot's heading direction, there is not significant differences before and after the kidnapping. Even though the error on coordinate x of the robot's pose, the EKF estimates converge to the values provided by the vision system before the robot completing the mission.

As the robot has to cover a larger distance than expected, once the robot is moved to a position farther the one that it was before the kidnapping, it is expected that the robot's linear velocity increases to compensate the delay provoked by the kidnapping. Fig. 6.35 (a) shows the solution m (blue continuous line) and the amplitude of the oscillator, A (black dashed-dotted line). The robot completes the first mission without any disturbance. During the second mission, note the increasing of the amplitude A



(a) Coordinate x of the robot's pose.

(b) Coordinate y of the robot's pose.

(c) Orientation ϕ of the robot's pose.

FIGURE 6.34: Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). Green area shows the interval of time when the robot is being kidnapped.

after the kidnapping, and consequently the increasing of solution m. Fig. 6.35 (b) shows the distance covered by the robot during the two missions. It is expected that the robot covers approximately 8 m during the two missions. However, during the kidnapping, the robot's wheels keep rotating, increasing the distance that the robot believes it traveled. When the robot recovers from the kidnapping, it has to cover the remaining distance to reach the goal location. Thus, the robot covers approximately 10 m to complete these two missions.

Despite the kidnapping during the second mission, the robot completes with success both missions. Thus, it is expected that the stability indicator holds less than 1 during the two missions, which can be verified in fig. 6.36.

6.2.8 Experiment 8

In this experiment, the robot is again moved to another location without being noticed. However, this mission is more complicated than the previous one, since the robot has

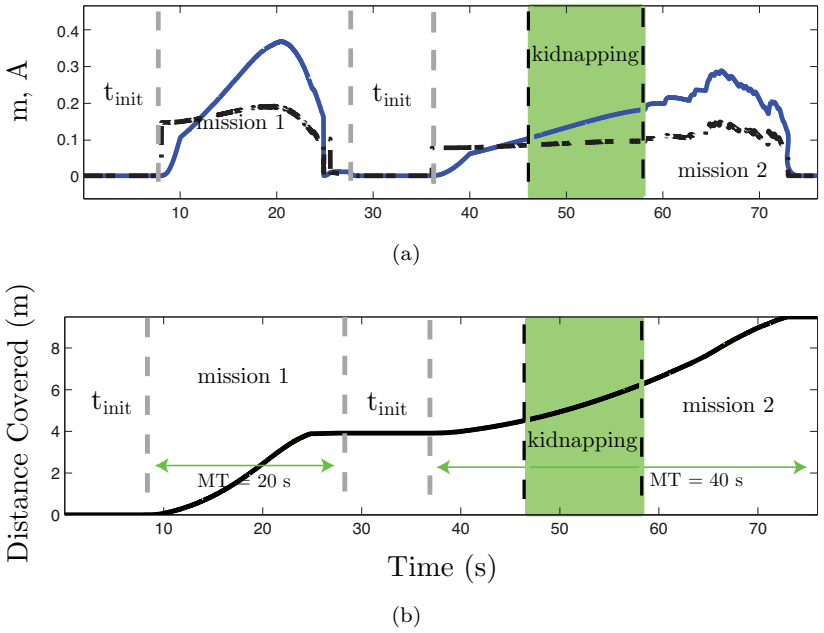


FIGURE 6.35: (a) Solutions m (blue continuous line) and amplitude of the oscillator, A (black dashed-dotted line). (b) Distance covered by the robot throughout the two missions.

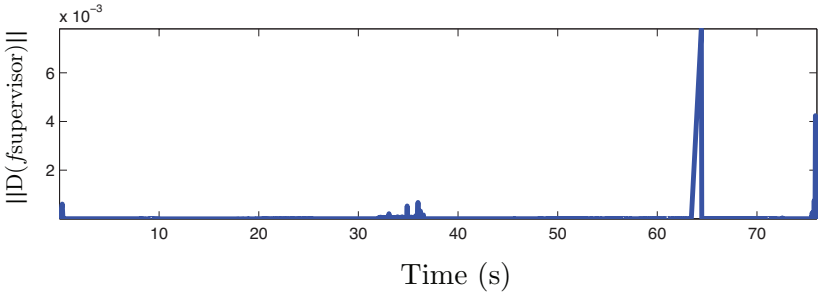


FIGURE 6.36: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 7.

to move from regions r_1 (inside the laboratory) to r_6 (corridor) (see fig. 6.25). To complete this mission, the robot should navigate through a door that separates the laboratory from the corridor. If the robot is kidnapped inside the laboratory, it must recover its pose before reaching the door. In the case it does not, it might assume that it already reached the door and turns towards the goal location. This will cause a long delay in the mission, as its true position is inside the laboratory. The time constraint for this mission is $MT = 60$ s.

Fig. 6.37 shows snapshots of the robot during this experiment. Panel A shows the robot starting its mission. Panel B illustrates the disabled robot's vision system and the robot being moved to another location without knowing. In panel C, the robot is passing through the door, in order to move to the corridor and reach the goal location.

Panel *D* shows the robot approaching the goal location.

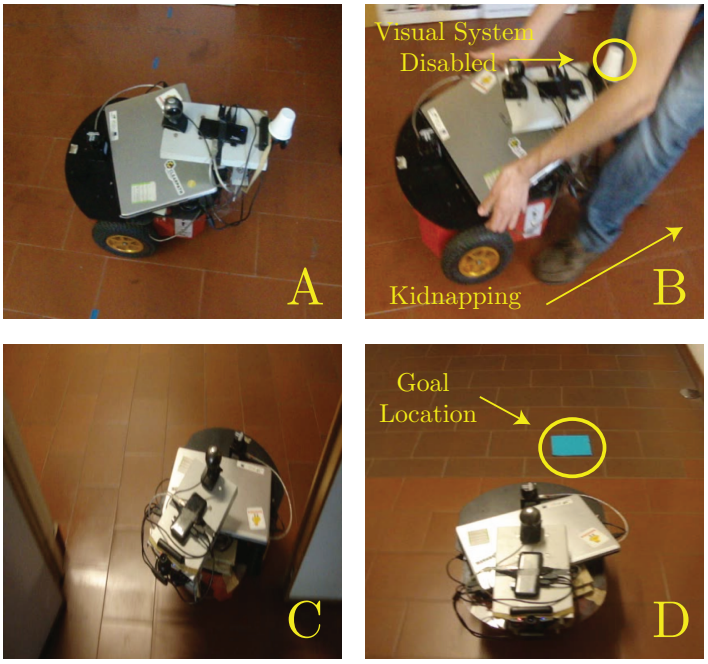


FIGURE 6.37: Snapshots of the robot performing the mission in experiment 8. During this mission, the robot is kidnapped and its vision system is disabled.

Fig. 6.38 (a) shows the robot's linear velocity, v (red continuous line). Green dashed circle shows the period of time in which the robot's pose estimated by the EKF is converging to the data provided by the robot's vision system after the kidnaping. After the period of convergence, the robot's linear velocity is increased, as the robot is moved to a position farther from the goal location than it was before the kidnapping. Fig. 6.38 (b) illustrates the distance between the robot's position and the goal location, D. Despite the kidnapping of the robot, it is noticeable that the robot is able to reach the goal location within the time constraint. The robot takes 58 s to reach the neighborhood of the goal location.

Fig. 6.39 illustrates the estimates of the robot's pose, $(\hat{x}_{\mathrm{r}}, \hat{y}_{\mathrm{r}}, \hat{\phi})$ (blue continuous line) and the robot's pose provided by the camera mounted on top of the robot $(x_{\mathrm{v}}, y_{\mathrm{v}}, \phi_{\mathrm{v}})$ (black circles). The green area represents the period of time in which the robot is moved to another location, farther from the goal location, without being noticed. During the interval of time $27 < t < 42$ s, the robot's vision system is activated, but it does not detect any landmark due to the lighting noise. After this interval of time, the robot starts detecting landmarks and approximately at $t = 48$ s, the robot's pose has converged. The convergence occurs before the robot reaches the door that separates the laboratory from the corridor. Consequently, the robot is able to rotate towards the goal location within the corridor and complete the mission with success.

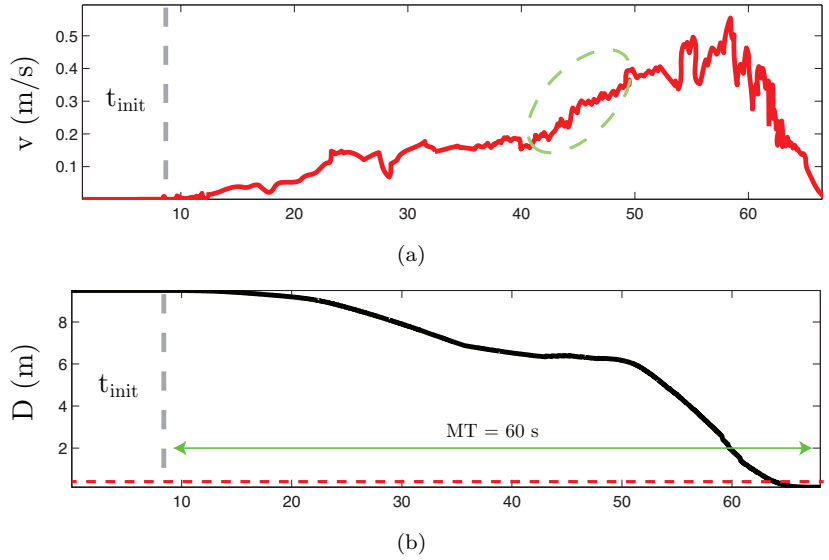


FIGURE 6.38: (a) Robot's linear velocity, v (red continuous line). (b) Distance covered by the robot during the two missions.

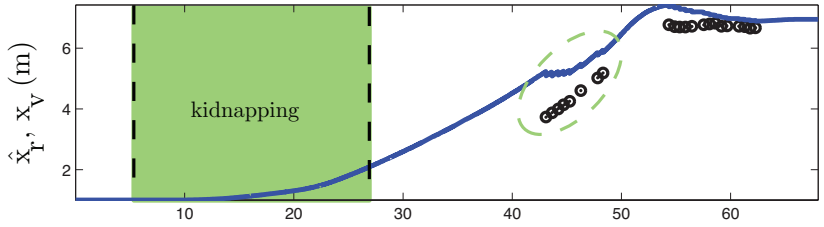
Fig. 6.40 shows the limit-cycle of the Stuart-Landau oscillator. Black arrows indicate the direction over which solutions m and n evolve. It is noticeable the increasing of solution m (green dashed ellipse), in order to compensate the delay provoked by the kidnapping.

The mission is successfully completed, despite the kidnapping. Consequently, it is expected that the stability indicator identifies the success of the mission. Fig. 6.41 shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along this mission. It is below 1 during all mission, and condition (4.5) holds, identifying the success of the mission.

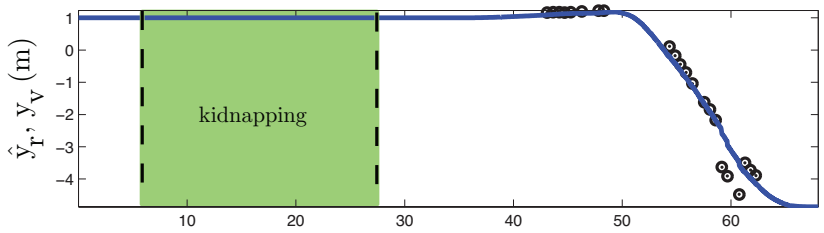
6.2.9 Experiment 9

In this experiment, the robot performs two missions and is kidnapped during the second mission. In the first mission, the robot moves from regions r_1 to r_2. The second mission consists on returning the robot to the original location in region r_1. The robot should cover 4 m in each mission. The time constraints are $MT = 20$ s and $MT = 40$ s for the first and second missions, respectively. The time constraint for the second mission is higher to allow compensating the time lost during the kidnapping.

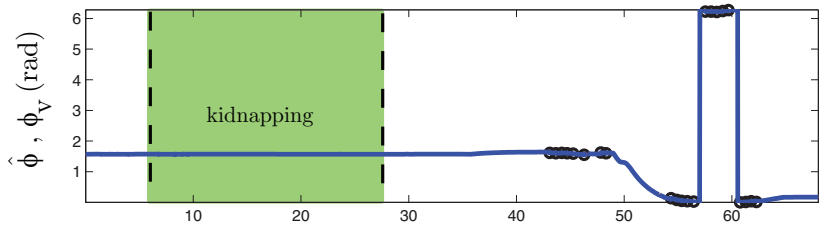
During the second mission, the robot is kidnapped, but the localization system is unable to recover the robot's pose before the robot assumes that the mission is completed. In fact, the position where the robot stops is far from the goal location. Fig. 6.42 depicts some snapshots of this experiment. Panel A shows the robot reaching the goal location of the first mission. Panel B shows the robot's vision system being



(a) Coordinate x of the robot's pose.



(b) Coordinate y of the robot's pose.



(c) Orientation ϕ of the robot's pose.

FIGURE 6.39: Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). Green area shows the interval of time when the robot is being kidnapped.

disabled. Panel *C* shows the robot being kidnapped and panel *D* depicts the position where the robot stops, which is at a distance of 2 m from the goal location.

Fig. 6.43 depicts the estimates of the robot's pose, $(\hat{x}_r, \hat{y}_r, \hat{\phi})$ (blue continuous line) and the robot's pose provided by the camera mounted on top of the robot (x_v, y_v, ϕ_v) (black circles). The kidnapping and the deactivation of the robot's vision system occurs in the interval of time $41 < t < 60$ s, represented by the green area. After the kidnapping, the robot's vision system is unable to detect any landmark, due to lighting noise. Consequently, as the EKF relies only on odometry during the period in which the robot is being kidnapped, the robot will be lost, since only the odometry data is being updated. The robot will believe that it already reached the goal location, when its true position is approximately at 2 m from the goal location.

Fig. 6.44 (a) shows the robot's linear velocity, v (red continuous line) and the solution m (blue dashed line). The green area stands for the period of time in which the robot is being kidnapped and its vision system is disabled. As the robot's vision system is unable to recover the robot's pose after the kidnapping, the EKF continues estimating the robot's pose only based on odometry. The robot is unable to detect this

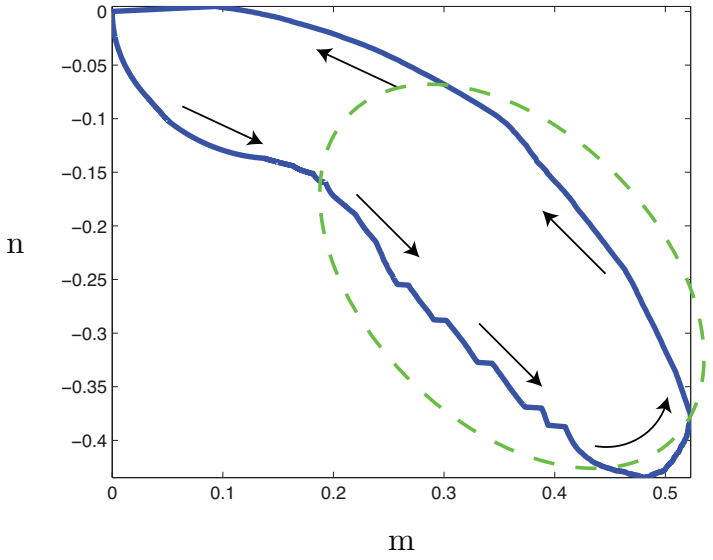


FIGURE 6.40: Limit-cycle of the Stuart-Landau oscillator when the robot is kidnapped and its amplitude is increased to compensate the provoked delay. Green dashed ellipse shows the increasing and decreasing of the solutions required to solve the kidnapping.

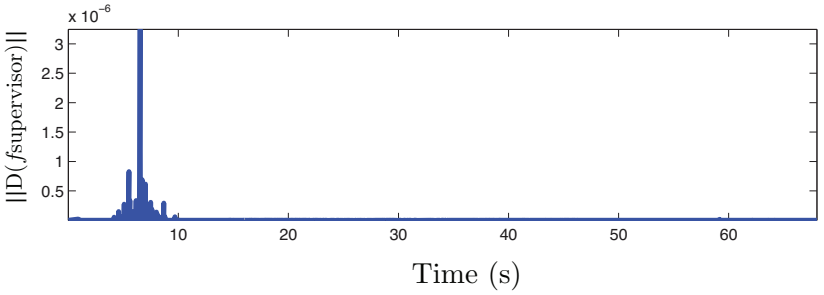


FIGURE 6.41: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 8.

catastrophic failure and stops moving before reaching the goal location. As shown in fig.6.44 (b), the robot finishes both missions in the neighborhood of both goal locations. In mission 1, both vision and odometry data are used by the EKF to estimate the robot's pose. However, in mission 2 only odometry is used. Therefore, the robot is unable to detect any kidnapping and re-localize itself.

Fig. 6.45 shows that $\|D(f_{\text{supervisor}})\|$ remains below 1 during this experiment, which means that the robot believes that the mission is successfully completed. In fact, this is true if only considering the information the robot has available to estimate its robot's pose. Moreover, it is expected that eventually, a landmark will be detected and the robot would be able to recover its pose. Nevertheless, this does not happen before mission 2 has been completed.

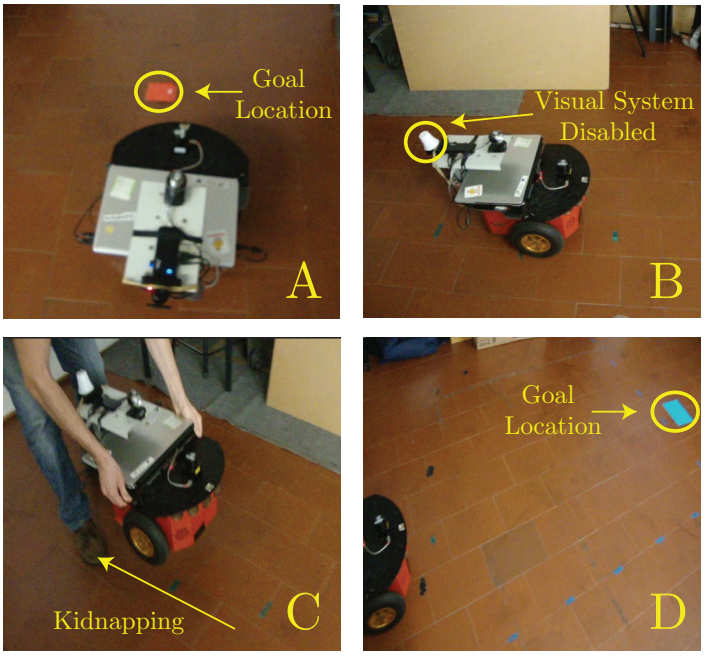


FIGURE 6.42: Snapshots of the robot performing a mission in which it is kidnapped and does not reach the goal location.

6.2.10 Experiment 10

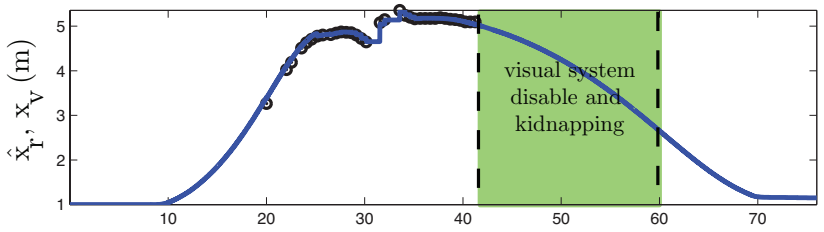
In this experiment, the robot must pass through a narrow passage and reach a goal location at the end of the passage. As a result of the narrow passage, the mechanism for reducing the robot's linear velocity (3.59) prevents the robot to follow at the required velocity to complete the mission within $MT = 20$ s. Consequently, the robot will be delayed and unable to complete successfully this mission.

Fig. 6.46 shows snapshots of this experiment. Panel A illustrates the beginning of the mission. In panels B and C, the robot is moving inside the tunnel and in panel D the robot reaches the goal location.

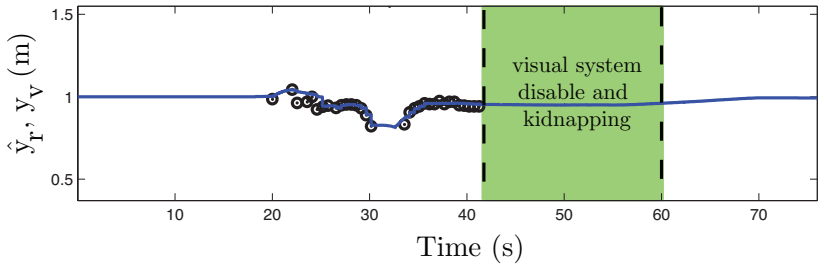
Even though the robot is able to reach the goal location, the time constraint for this mission is not verified. The robot's linear velocity is reduced, so the robot is able to safely traverse the narrow passage. Fig. 6.47 shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission. At $t \approx 22$ s, the robot is unable to complete the mission in MT. Thus, $D(f_{\text{timing}})$, returns a value sufficiently large to set $||D(f_{\text{supervisor}})|| >> 1$.

6.2.11 Experiment 11

In this experiment, the robot performs a long-term mission consisting on reaching a sequence of 7 different goal locations. The sequence is repeated 70 times, performing

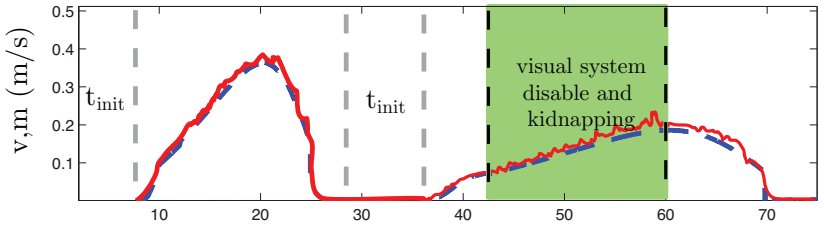


(a) Coordinate x of the robot's pose.

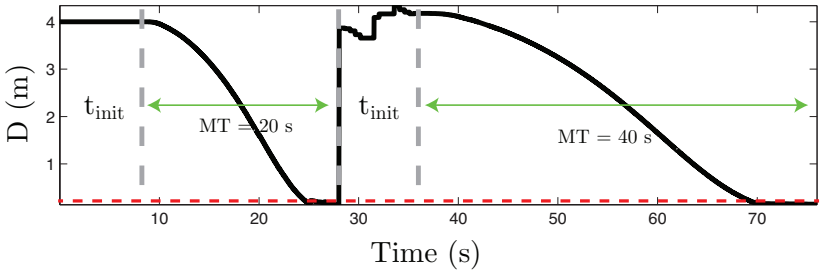


(b) Coordinate y of the robot's pose.

FIGURE 6.43: Estimates of the robot's pose (blue continuous line) through the EKF and robot's pose provided by the camera (black circles). Green area shows the interval of time when the robot is being kidnapped and its vision system is disabled.



(a)



(b)

FIGURE 6.44: (a) Robot's linear velocity, v (red continuous line) and solution m generated by the Stuart-Landau oscillator (blue dashed line). (b) Distance covered by the robot during the two missions.

490 missions. When the final goal of the sequence is reached, the robot's next goal is set as the first mission, thus repeating the cycle. These goal locations are chosen, so the robot covers a large part of the free space in the environment.

During these multiple missions, the robot faces several situations, usually found in disturbed environments, such as, static obstacles and people moving around. While

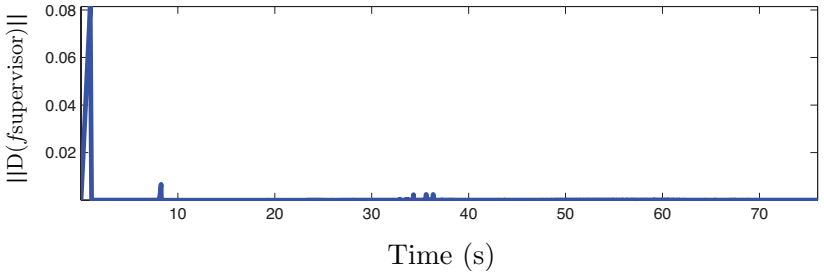


FIGURE 6.45: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the two missions of experiment 9.

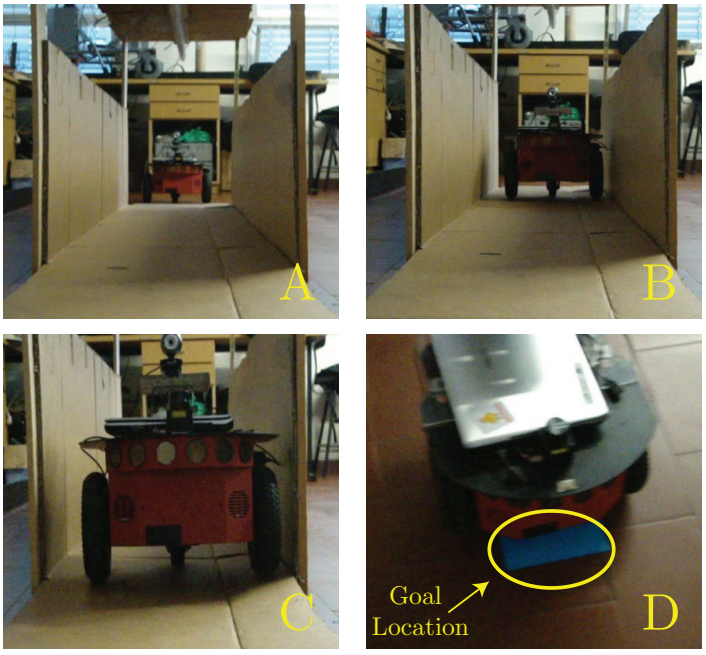


FIGURE 6.46: Snapshots of the robot moving in a narrow passage.

some people collaborate with the robot by following a suitable trajectory to avoid colliding with it, others try to obstruct the robot by standing up in front of it and complicating its trajectory.

Fig. 6.48 illustrates some situations that the robot has to face during the long-term mission. Panels 1, 2, 4, 7 and 8 show the robot moving in the corridor, while the other panels show the robot in the laboratory.

The total time spent by the robot to complete the missions is approximately 6 hours, covering a total distance of approximately 4200 m. Fig. 6.49 depicts the trajectory performed by the robot during the missions. This trajectory is estimated through the EKF. The blue circles illustrate the 7 goal locations. Note that the robot reaches the neighborhood of these goal locations. Red dashed ellipses depict 3 situations in which the error between the robot's pose estimates and the ground truth was higher than

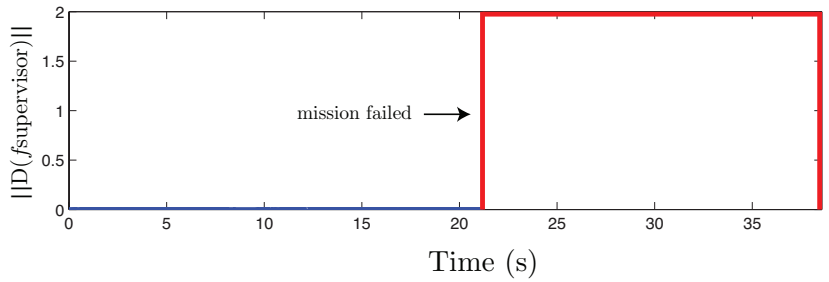


FIGURE 6.47: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the mission of experiment 10.

FIGURE 6.48: Snapshots of the robot performing the long-term mission. These images are captured by a camera mounted on top of the robot and pointing forwards.

0.4 m. Despite these errors, the robot is able to re-localize itself in such situations. These errors occur because the robot travels a large distance without detecting a landmark. The re-localization is possible because eventually, a landmark on the ceiling is detected and the reset operation of the estimates is triggered.

Fig. 6.50 shows the evolution of the bound of $\|D(f_{\text{supervisor}})\|$. Condition (4.5) holds when the mission is completed within MT and does not hold when the mission fails (see red data for failed missions). During the 6 hours of movement, the robot fails to reach the goal location within the time constraint in 17 missions. Despite the unsuccessful of the missions in terms of time constraints, the robot reaches the goal location of the missions.

The failed missions occur because of two reasons. The first one, and the one that causes more failed situations (14 in 490 missions), is the high density of obstacles in the environment. These obstacles create complex situations, in which the robot is unable to handle them. The other reason that lead to the failure of 3 missions, is the large error on the robot's vision system during these missions. This happens because the robot remains a long time without detecting a landmark. The robot's pose estimates

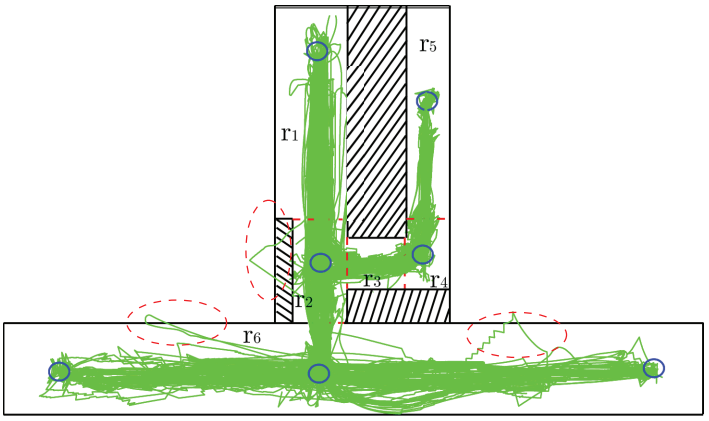


FIGURE 6.49: Estimates given by the EKF of the trajectory followed by the robot. Striped areas represent forbidden space. Red dashed lines stand for the critical lines dividing the regions. Blue circles illustrate the 7 goal locations where the respective missions end. Red dashed ellipses show situations in which the error of the robot's pose was larger than 0.4 m.

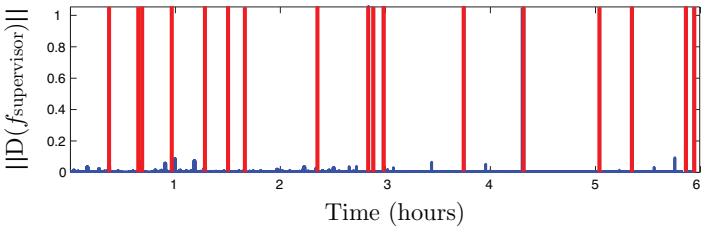


FIGURE 6.50: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the sequence of missions. Values larger than 1 mean that the mission is not successfully completed. Graphic is bounded to 2, but for failed missions the bound of $\|D(f_{\text{supervisor}})\|$ is much higher.

are being updated only through the odometry, which become unreliable after a few meters of displacement or when the robot rotates.

In terms of a dependability analysis, the architecture is evaluated in terms of integrity, reliability and safety. Regarding the integrity analysis, the localization system is able to re-localize the robot from the catastrophic failures occurred in these 3 missions, when a landmark is detected and the reset operation activated. However, the convergence of the Localization System is not sufficiently fast to allow that the robot completes the 3 missions in due time.

In terms of safety, despite the several obstacles that the robot has to face, only one collision is recorded. In fact, the robot collides with one wheel of the cleaning cart in the corridor, when the cleaning staff is operating. This occurs because the laser or sonars of the robot do not detect the cleaning cart wheels. Nevertheless, after the collision, the robot is able to continue moving towards the goal location and

successfully complete the respective mission.

During the missions, the robot fails to reach the goal location within the time constraint, MT, in 17 missions, yielding a reliability percentage of approximately 96.5% (17/490).

6.2.12 Experiment 12

At the beginning of this experiment, a set of 242 locations over the environment are randomly generated, such that no two successive locations are from the same region of the environment. Sequentially, the robot has to autonomously navigate to these locations. The time MT for each mission is also randomly generated, so the required average robot's linear velocity to complete the mission is 0.30 m/s.

Fig. 6.51 depicts the goal location for the missions in this experiment (green circles for success missions and red crosses for failed missions). During the missions, the robot faces the same difficulties found in the previous experiments. The percentage of failed missions is higher inside the laboratory, (regions r_1, r_2, r_3, r_4, r_5), than in the corridor (region r_6). The laboratory has more narrow spaces, which facilitates the creation of unlucky obstacles configurations that prevent the robot to reach the goal within the assigned MT.

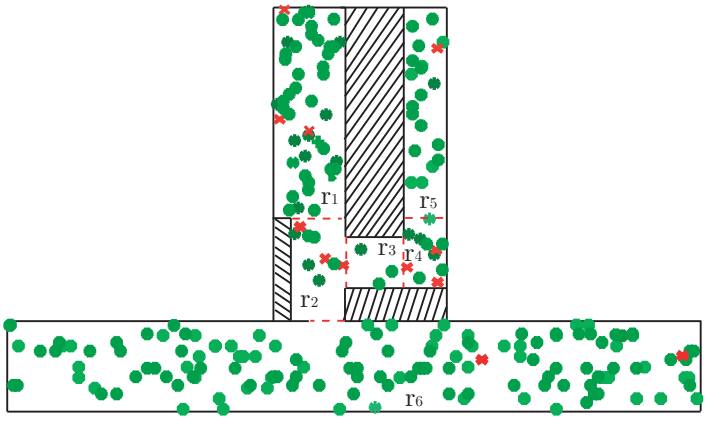


FIGURE 6.51: Goal locations P_g of the missions performed by the robot during experiment 12. Green circles represent successful missions whereas red crosses represent unsuccessfully missions. There is a total number of 230 successful missions and 12 unsuccessfully missions.

The robot performs 242 missions and covers a distance of approximately 1261 m in approximately 2 hours. Results of this experiment are summarized in table 6.2. The average MT for each mission is approximately 22 s, and the average linear velocity is 0.27 m/s. Note that this velocity is lower than the required average velocity used to calculate MT (0.30 m/s). This is a consequence of the failed missions, in which

the robot moves at a low velocity (0.1 m/s) when it verifies that the goal location is unreachable in the remaining time. The percentage of successful missions in this experiment (95%) is slightly lower than the percentage verified in experiment 11 (96.5%). This means that the robot has more difficult to complete the missions within their time constraints. The reason for the decrease in the success of the missions is that some of the random goal locations are difficult for the robot to reach them (*e.g.*, corners and locations close to obstacles).

TABLE 6.2: Results of the sequential missions performed in experiment 12.

Number of missions	Time (s)	Average Velocity (m/s)	Maximum Velocity (m/s)	Distance Covered (m)	Average MT (s)	Fails
242	7057	0.27	0.8	1261	22.2	12

Fig. 6.52 depicts $\|D(f_{\text{supervisor}})\|$. Clearly, condition (4.5) holds when the missions are successfully completed. Once again, when the robot does not complete the mission within MT, condition (4.5) is not verified.

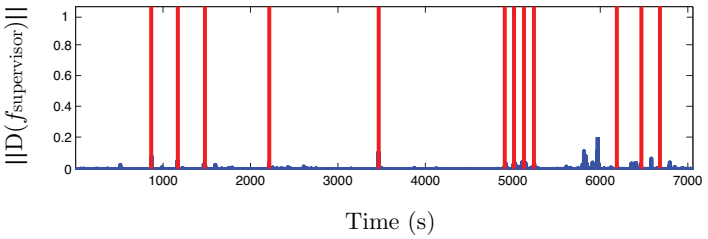


FIGURE 6.52: Evolution of the bound of $\|D(f_{\text{supervisor}})\|$ along the sequence of missions. Values larger than 1 mean that the mission is not successfully completed.

6.3 Discussion of the experiments

In this chapter, several simulations and real experiments in a dynamic and cluttered indoor environment were performed. These experiments allowed verifying the behavior of the robot while performing missions under different situations. During the missions, the robot detected static and dynamic obstacles, navigated through narrow passages and was kidnapped to other locations without being noticed.

The robot successfully avoided the detected obstacles, excepting a cleaning cart, whose wheels were not detected by the laser or sonars. During the kidnappings, the robot's pose was being updated only through the odometry data, since the robot's vision system was disabled. Consequently, the robot was lost. After the kidnappings,

the robot's localization system took approximately 5 s, after detecting at least one landmark, to recover to the true robot's pose. This convergence time could be reduced, if a larger weight in the EKF estimates was given to the robot's vision system. However, the odometry would had a negligible contribution, even for shorter distances.

In the long-term missions, the robot traveled 5460 m in approximately 8 hours and performed 732 missions. The percentage of successful missions was approximately 97% (710/732), *i.e.*, the robot failed 22 missions. The majority of these failed missions were caused by complex configurations of obstacles. The robot spent much time to handle them and it was unable to complete the missions in due time. The remaining missions were failed because of the large error on the robot's vision system. Even though these missions were not completed in due time, the robot succeeded to reach the goal location of each mission. In the simulation experiments, the robot successfully avoided typical hospital obstacles and performed delivery missions during approximately 9.30 hours.

When the mission was successfully completed, the robot reached the goal location within its respective time constraint. However, the time that the robot took to complete the mission was less than the time constraint. This was due to the distance ϵ. In general, the robot completed a mission in 88% of the time constraint of the respective mission. The standard deviation was approximately 8%. A more accurate localization system could reduce the assumed value of ϵ. Consequently, the average time to complete the mission would be closer to the time constraint of the mission. Furthermore, the standard deviation would also be reduced.

The performance condition based on the Contraction Theory identified the ability of the robot to complete with success its missions. In cases where the robot failed to complete the mission, $\|D(f_{\text{supervisor}})\|$ was greater than 1. Otherwise, the bound of $\|D(f_{\text{supervisor}})\|$ was below 1.

Chapter 7

Conclusions

This thesis addresses a robot control architecture based on nonlinear dynamical systems. The architecture is in charge of controlling a mobile robot, so that it performs delivery missions with time constraints between different locations in an indoor environment. Three modules of control compose the architecture. Motion and Local Control modules are responsible for generating an angular velocity that safely guides the robot towards the goal locations. Timing Control module generates the linear velocity that enables the robot to complete its mission within the time constraint. Each module of the architecture is parameterized under stability conditions derived through the Contraction Theory. Furthermore, a stability indicator based on the combination property of the Contraction Theory allows identifying the success of the missions as a stability problem. Several long-term experiments in an indoor environment show the ability of the architecture to guide the robot along its missions and the capability of the stability indicator to classify missions as failed or success.

This final chapter starts with a section summarizing the subjects addressed throughout the thesis. The second section overviews the achieved original contributions and includes a general discussion about the work. The final section addresses several possibilities for future research.

7.1 Addressed Subjects

Section 2.5 addresses the requirements for autonomous navigation of mobile robots in cluttered and dynamic environments. These were: (1) the need for including time constraints in missions performed by a robot. (2) The development of an architecture that drives the robot so that it follows the fastest path while avoiding obstacles. (3) Design principles for complex control architectures. (4) The need to analyze the mission

success as a stability problem and (5) lack of results in previous studies demonstrating the stability and behavior of robots when performing long-term missions.

These problems were tackled throughout this thesis as shown by the following topics:

- (1) The Timing Control that fulfills the timing constraints of the mission is detailed in section 3.4. This module includes the three-iterative algorithm that calculates the remaining distance to reach the goal location in section 3.4.2.1 and the heuristic rule that reduces the robot's linear velocity in the vicinity of obstacles in section 3.4.3.

- (2) The Motion Control capable of generating the trajectories for the robot is depicted in section 3.2. This module includes the critical areas that handle the uncertainty on the robot's position in section 3.2.5 and the dynamical system responsible for generating the local goals in section 3.2.6. Furthermore, the Local Control responsible for avoiding obstacles and driving the robot towards the goal locations is described in section 3.3.

- (3) The performance condition that identifies the mission success as a stability measure is derived in section 4.2.

- (4) The stability analysis of the proposed architecture through the Contraction Theory is described in section 4.3.

- (5) The optimal landmark distribution necessary for the Localization System is illustrated in section 5.6.1. A dependability analysis of the Localization System is verified in section 5.7. The performed experiments, including the long-term ones, are shown in chapter 6.

7.2 Summary of Contributions

The contributions of this thesis are divided in three main research directions: (1) the development of a control architecture based on nonlinear dynamical systems, responsible for generating timed movements for a mobile robot while guiding it in an indoor environment; (2) a stability analysis based on the Contraction Theory that infers conditions so that the modules of the architecture are designed on top of stability conditions. Furthermore, the combination property of the Contraction Theory is used to obtain a stability indicator that verifies the ability of the robot to successfully complete its missions; (3) realization of long-term field experiments that experimentally show

the ability of the architecture to drive the robot, and the ability of the performance condition to identify failed and successful missions as a stability problem.

(1) Control architecture based on nonlinear dynamical systems

The proposed architecture is fully designed in terms of C^1 nonlinear dynamical systems and feed-through maps. It contains three modules of control, namely, Motion, Local and Timing.

The Local Control module (chapter 3) explores the dynamical systems theory to control the robot's heading direction, and thus guide the robot along the environment. Sensory information is included into the dynamical system, in order to verify the presence of obstacles. If obstacles are detected, repulsive values change the robot's heading direction so that the robot avoids the obstacles.

1. Works that adopted the dynamical systems theory to guide mobile robots [1, 83, 85, 289], used low cost sensors, such as sonars or infrared to sense the environment. The number of sensors was limited, but the dynamical approach successfully guided the robot towards its goal locations in environments with few obstacles. This thesis contributed to show the feasibility of this dynamical approach for obstacle avoidance when the robot uses a laser range finder in complex environments with unpredictable obstacles (chapter 6).

   This local planner method based on the dynamical systems theory successfully guided the robot throughout the missions with time constraints in cluttered and dynamic environments during long-term missions. However, this local planner takes a long time to overcome some configurations of obstacles.

The Timing Control module (chapter 3) explores the Stuart-Landau oscillator to control the robot's linear velocity. This oscillator contains a Hopf bifurcation that modifies the generated solution from a stable fixed point to a limit cycle, or vice versa. The stable fixed point generates a constant or null velocity, depending on the solution offset. The limit cycle solution is adapted to accelerate or decelerate the robot's linear velocity, such that the mission is accomplished within its time constraint. In cases where the robot is delayed, the amplitude of the oscillator increases and the robot's linear velocity increases as well. When the robot needs to decrease its velocity, the amplitude is reduced and the robot slows down. This real-time adaptation of the robot's linear velocity is possible if the robot knows the remaining distance to reach the goal location.

2. The main innovation of this module regarding previous works [1, 3, 13, 85, 151] is the extension of the dynamic approach to generate and modulate the robot's linear velocity, when dealing with global and local path planning in complex environments. The difference to these works is the calculus of the remaining distance that the robot has to cover for each mission. In such works, the remaining distance was calculated through the Euclidean distance between the goal location and the current robot's position, since both locations were in the same region of the environment.

   In this work, a three-step iterative algorithm (section 3.4.2.1) considers the sequence of regions that the robot has to traverse and the current robot's position to calculate the distance the robot has to cover. Real experiments and simulations showed the feasibility of this three-step iterative algorithm to calculate the remaining distance the robot has to cover.

   It can be concluded that the robot's linear velocity generated by the Timing Control module was successfully generated and adapted in real-time, in cases where the robot was delayed.

(2) Stability analysis and mission success

Contraction Mapping Theory was used as a stability theory framework to design, specify and analyze the dynamical systems of the control architecture. The stability analysis derived conditions used as guidelines to define the parameters of the dynamical systems, so that the contraction or eventual contraction of each dynamical system is verified. Ensuring that a dynamical system is contracting or eventually contracting guarantees the exponential convergence to its unique equilibrium state, independently of its initial conditions.

3. Previous works [270–273, 275] that used Contraction Theory to prove stability of control architectures did not consider the integration of timing control, global and local navigation into the same architecture.

   This thesis contributed to verify the feasibility of the Contraction Theory to derive stability conditions for a navigation architecture able of generating timed movements, local and global path planning (chapter 4).

   It can be concluded that Contraction Theory is an important method from stability theory to provide conditions that ensure the exponential stability of the navigation architecture. This thesis provided sufficient formal and experimental

results showing that Contraction Theory is an alternative to Lyapunov stability to establish exponential stability of nonlinear systems.

The combination property of the Contraction Theory (section 4.2) provides freedom in the architecture design. The knowledge of the internal organization of the architecture is not required to prove its contraction. Furthermore, for a generic robot, for which there is no information on its structure, the contraction of the architecture is viewed as a weak performance condition that identifies the mission success.

4. This thesis showed that the combination property of the Contraction Theory allows the inclusion of additional constraints into the architecture. In this work, the mission constraint was time, but others could be added as well.

   The performance condition identified as a stability measure the ability of the robot to complete with success its missions. This shows the usefulness that Contraction Theory, in particular its combination property, provides to the stability analysis of complex real-time control architectures.

(3) Long-term experiments

Despite the great importance of stability theory to design stable architectures for mobile robots, there is no extensive research addressing the architecture stability when robots move in real environments. Works [270–273, 275] that used Contraction Theory lack in robustness when both the robot and environment include external perturbations into the global system. Besides, the obtained results were achieved in simulation. On the other hand, long-term experiments allow to verify if the architecture is able to drive the mobile robot for large periods of time.

5. This thesis yielded stability analysis results of an architecture that guides a mobile robot in long-term experiments in a typical university indoor environment. The experiments showed that the dynamical systems remained stable even when perturbations of the environment affect the architecture. Furthermore, Contraction Theory is a suitable stability theory framework to verify the stability of the architecture in real environments.

7.3 Outlook

The work described in this thesis offers several possibilities for further research. These are focused on improving the major limitations identified throughout the thesis.

A first limitation was identified on the local planner method based on the dynamical systems approach. This method presented difficulties to handle complex configuration of obstacles. The Gaussian noise added to the vector field to ensure an escape from local minima in finite time is a poor solution. If the added noise level is high, the robot presents an oscillatory behavior. If the noise level is low, the robot might take longer to avoid some complex configurations of obstacles. This might imply higher velocities to fulfill the time constraint of the mission. A future improvement is to use other C^1 local planner methods that avoid local minima and present better performance when dealing with complex configuration of obstacles.

The Localization System uses artificial landmarks distributed *a priori* along the environment, in order to estimate the robot's pose. In large environments, such as hospitals, the number of required landmarks could be about thousands, if comparing to the number of landmarks used in this work. A further research direction could be exploited to reduce the number of required landmarks. A typical solution is to use a camera with a wider field of view, so that the distance between landmarks increases and fewer landmarks are necessary to ensure that at least one landmark is always visible by the robot. Other solution could be use the natural landmarks already present in the environment, as doors, corners, walls, *etc.* Combining natural with artificial landmarks or using only natural landmarks is an alternative solution to estimate the robot's pose without the need to distribute thousands of artificial landmarks in the environment.

The robot's pose estimates provided by the Localization System were obtained through a combination of odometry and vision data. A typical solution that could improve the accuracy of the robot's pose estimates is to match the acquired sensory information with an *a priori* built map of the environment. Furthermore, using maps to localize the robot could be an important alternative for when the camera mounted on the robot is not detecting any landmark. The fusion of vision and odometry data was obtained through an Extended Kalman Filter. The inclusion of more sensors to estimate the robot's pose is possible, however, the reset operation would have to be updated. The weight that each sensor contributes to the estimates has to be calculated when sensors are added or replaced.

The stability results provided in this thesis were obtained through the Contraction Theory, which relied on smoothness assumptions, and thus on the use of C^1 dynamical systems and feed-through maps. A future improvement should provide to the stability analysis the ability to deal with non-smooth dynamics. For instance, transitions between solutions of dynamical systems can be studied through the bifurcation theory. Other solution includes converting non-smooth dynamical systems to C^1 dynamical systems. For instance, a non-smooth dynamical system can be partitioned

into several pieces, on which smoothness holds (piecewise smoothness). Using bifurcation theory [80] or piecewise smoothness [280] to analyze non-smooth dynamics allows the inclusion of new control algorithms into the architecture, which could handle for instance the navigation limitations, namely on the Local Control module.

The experiments accomplished along this work were done in a typical university indoor environment, in order to mimic the conditions that a mobile robot would face in a hospital. The university indoor has static and dynamic obstacles, as well as narrow passages and doors. However, a real hospital environment provides unexpected situations that were not covered in the performed experiments and must be solved when considering the real application. For instance, how do patients and staff interact with the robot, how do the robot should behave when it is unable to finish the mission within the time constraint, or in emergency cases, such as evacuations and fires? Shall the robot stop and give priority to medical staff and persons in hallways or shall the robot try to circumnavigate them? All these situations must be handled by the architecture of the robot. Experiments in a hospital environment could provide additional insights to the previous questions.

In fact, when running an autonomous mobile robot in real environments, safety and operability issues are of major importance. In safety terms, it is necessary that the robot detects all the unexpected obstacles that can appear in the environment. The robot's scanning area must be maximized, such that there are no blind areas where obstacles are undetected. A typical solution to maximize the scanning area is to combine multiple sensors, such as cameras, bumpers, sonars, infrared and lasers.

In operability terms, the robot must check the charge level of its batteries, in such a way that when the charge level is low, the robot must autonomously move to a charging station. When the robot is charging its batteries or performing a mission, it is unable to accomplish a new mission. Consequently, a high level of control must supervisor and assign the new delivery missions to the available robots. Furthermore, mobile robots must be as autonomous as possible from human intervention. Thus, robots should be able to control elevators and electronic doors, in order to freely navigate in indoor environments. However, in case of failures, the robot must stop, alert the failure and wait for the human intervention.

This architecture has been developed with the focus of guiding delivery mobile robots in hospital environments. However, this architecture could be applied to other robotic applications. Timed movements can be suitable for a wide variety of robots in charge or domestic tasks, such as cleaning, mowing or vacuum tasks. Robots will be controlled such that their tasks are finished within time constraints. This is an alternative to controllers that assume that sooner or later the robots will complete the

domestic tasks. In warehouses, delivery robots can perform timed deliveries. These environments are less disturbed than hospitals and therefore the generation and modulation of the trajectory followed by the robot is more simple. In shopping malls, airports or other public spaces, exhibitory or publicity robots can make timed tours.

Appendix A

A.1 World Representations

Fig. A.1 illustrates the simulated environment (see fig. 5.6) represented by an occupancy grid, a topological and a landmark-based representation. The occupancy grid divides the environment into equal-sized cells. The topological map represents the environment through a set of regions and corridors and landmark-based representation uses known landmarks.

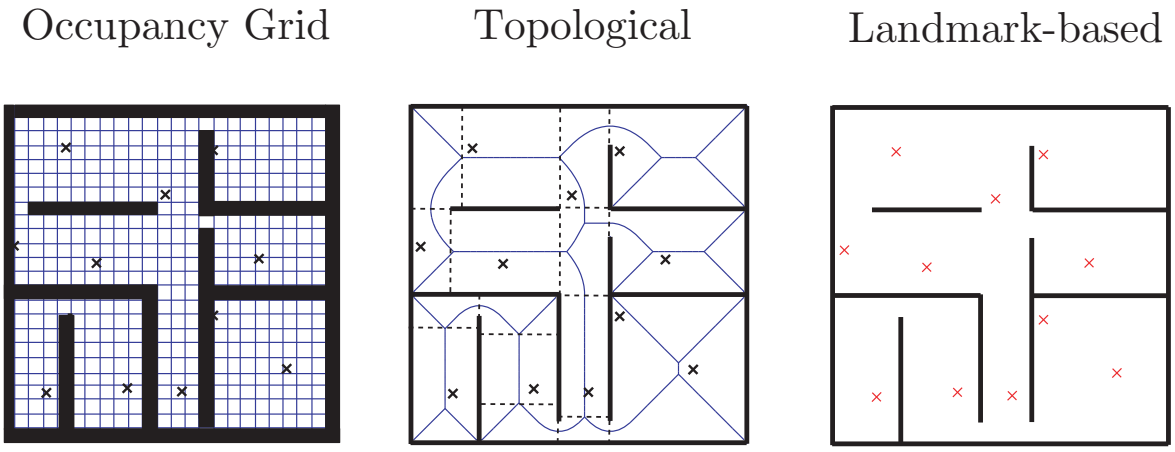


FIGURE A.1: Representation of the simulated environment using occupancy grid, topological and landmark-based representations. Crosses represent landmarks.

A.2 Orthogonal Projection

A representation of the orthogonal projection of the robot's position P_{r} onto a critical line can be viewed in fig. A.2.

The line segment $l_{i,i+1}$ that includes the points P_{1i} and P_{2i} and the line segment that includes the points P_{r} and P_{b} are perpendicular, and their slopes are the negative reciprocals of each other. Both line segments can be identified by their parametric

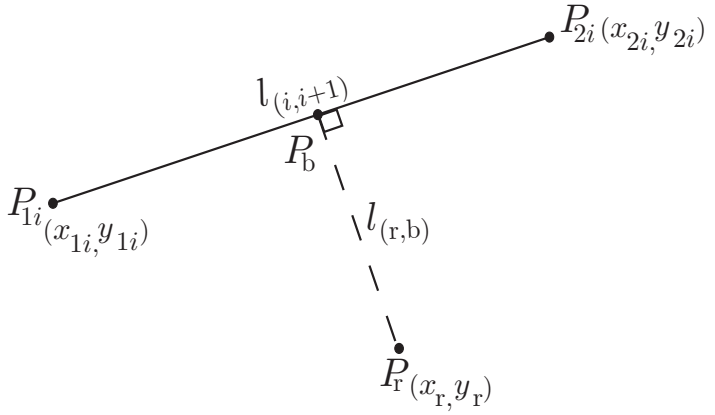


FIGURE A.2: Projection of the robot's position, P_{r}, onto the critical line, $l_{i,i+1}$. P_{1i} and P_{2i} are the extremities of the respective critical line.

representations as follows,

$$l_{i,i+1} \doteq y_1 = m_1 x_1 + b_1, \tag{A.1}$$

$$l_{\mathrm{r,b}} \doteq y_2 = m_2 x_2 + b_2. \tag{A.2}$$

Substituting m_1 and m_2 by the respective slopes and b_1 and b_2 by their intersection on y-axes, the following equations are obtained,

$$y_1 = -\left(\frac{x_{2i} - x_{1i}}{y_{2i} - y_{1i}}\right) x_1 + y_{\mathrm{r}} + \left(\frac{x_{2i} - x_{1i}}{y_{2i} - y_{1i}}\right) x_{\mathrm{r}}, \tag{A.3}$$

$$y_2 = \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right) x_2 + y_{2i} - \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right) x_{2i} \tag{A.4}$$

At point $P_{\mathrm{b}}(x_{\mathrm{b}}, y_{\mathrm{b}})$, the y_1 and y_2 intersect each other, and this point can be calculated as follows,

$$x_{\mathrm{b}} = \frac{y_{2i} - \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right) x_{2i} - y_{\mathrm{r}} - \left(\frac{x_{2i} - x_{1i}}{y_{2i} - y_{1i}}\right) x_{\mathrm{r}}}{-\left(\frac{x_{2i} - x_{1i}}{y_{2i} - y_{1i}}\right) - \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right)}, \tag{A.5}$$

$$y_{\mathrm{b}} = \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right) x_{\mathrm{b}} + y_{2i} - \left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right) x_{2i}. \tag{A.6}$$

A.3 Stuart-Landau oscillator

The topological type of the Stuart-Landau oscillator can be modified, since it contains an Hopf bifurcation, which occurs by changing the value of parameter μ. The solution bifurcates to either a limit cycle ($\mu > 0$) with amplitude A to a fixed point $(m, n) = (O_{\mathrm{m}}, 0)$, ($\mu < 0$, see Theorem 1 in section 4.4 in [80]). When $\mu = 0$, the solution

remains in the current solution (limit cycle or a fixed point). These phase portraits are shown in fig. A.3.

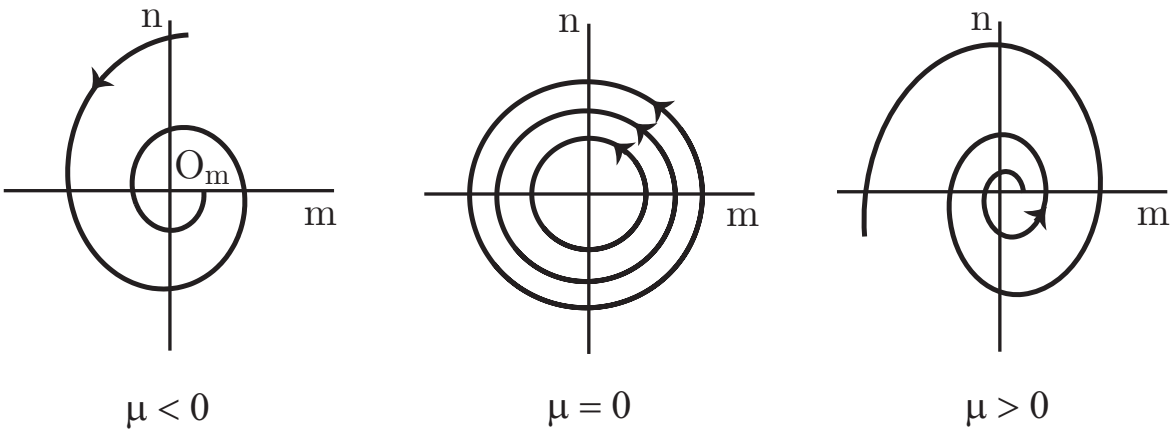


FIGURE A.3: Phase portraits of the Stuart-Landau oscillator depicted in (3.35)-(3.36).

The bifurcation diagram of the Stuart-Landau oscillator is shown in fig. A.4. The upper curve in the bifurcation diagram represents the one-parameter family of limit cycles $\Gamma_\mu = \sqrt{\mu}(\cos t, \sin t)^T$ which defines a surface in $R^2 \times R$.

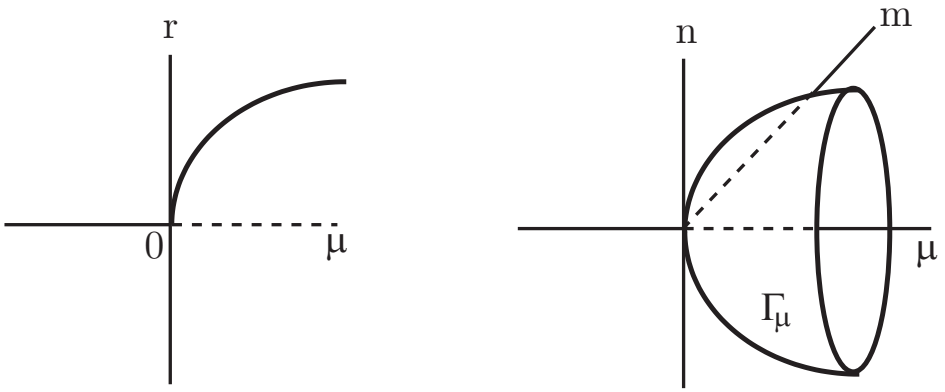


FIGURE A.4: The bifurcation diagram and the one-parameter family of limit cycles Γ_μ resulting from the Hopf bifurcation.

The control parameters of the Stuart-Landau oscillator (μ, O_m, ω and α) are used to explicitly modulate the generated solutions m and n.

The amplitude of the oscillations A is controlled by parameter μ, $A = \sqrt{\mu}$ for $\mu > 0$. Fig. A.5 depicts an example of the oscillator solutions when μ changes, and the amplitude A changes accordingly. Initially, $A = 1$ and the oscillations vary between -1 and 1. At time $t = 3$s, the amplitude changes to $A = 4$ and the oscillations vary between -4 and 4. At time $t = 6$s, the amplitude is again $A = 1$.

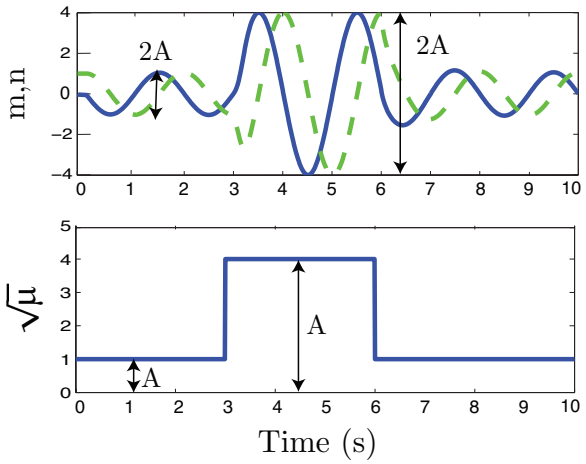
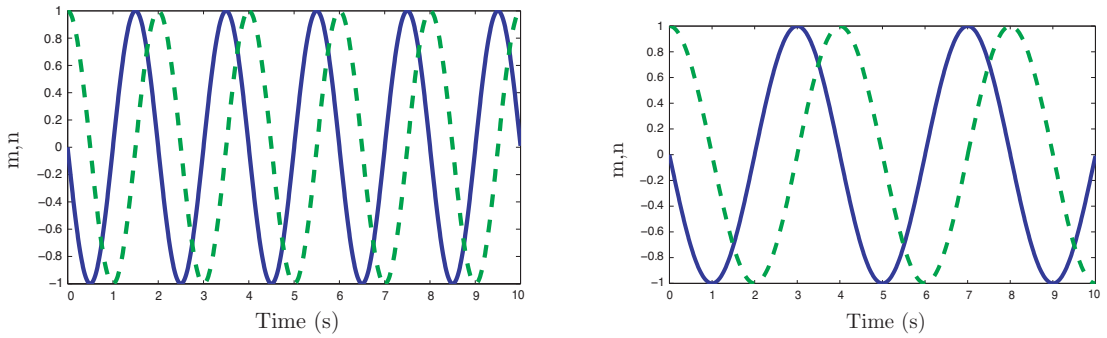


FIGURE A.5: Smooth modulation of the generated trajectory amplitude (top panel) by modifying parameter μ (bottom panel). Blue continuous line and green dashed line represent the state variable m and n, respectively.

Parameter ω specifies the oscillations frequency (rad.s^{-1}), with a periodic oscillation of $T = \frac{2\pi}{\omega}$s. Fig. A.6 shows an example of solutions (m, n) when the frequency ω is changed. In fig. A.6 (a) the frequency is $\omega = \pi$ (rad.s^{-1}), and in fig. A.6 (b), $\omega = \frac{\pi}{2}$ (rad.s^{-1}). The ω signal controls the limit cycle direction: for positive ω values, $\omega > 0$,



(a) $\omega = \pi$ (rad.s^{-1})

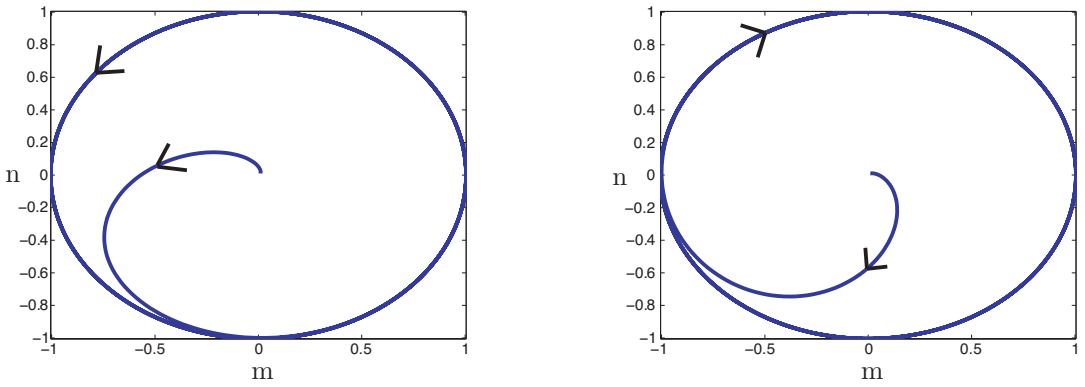
(b) $\omega = \frac{\pi}{2}$ (rad.s^{-1})

FIGURE A.6: Harmonic oscillations with different periods.

the limit cycle rotates counter-clockwise; for negative ω values, $\omega < 0$, it rotates clockwise (see fig. A.7).

Parameter O_{m} controls the offset of the solution m. Fig. A.8 illustrates O_{m} (dashed red line) that initially is 0 and $A = 1$. At time $t = 6$ s, O_{m} changes to $O_{\mathrm{m}} = -2$ and solution m (blue continuous line) converges to oscillate around O_{m}.

Control parameter, α, defines the relaxation rate, given by $\frac{1}{2\alpha\mu}$, of the generated solutions. The larger α, the faster the solutions converge to the steady state value.



(a) Counter clockwise limit-cycle. $\omega = \pi$ rad.s^{-1}

(b) Clockwise limit-cycle. $\omega = -\pi$ rad.s^{-1}

FIGURE A.7: Limit-cycles for different signals of frequency ω.

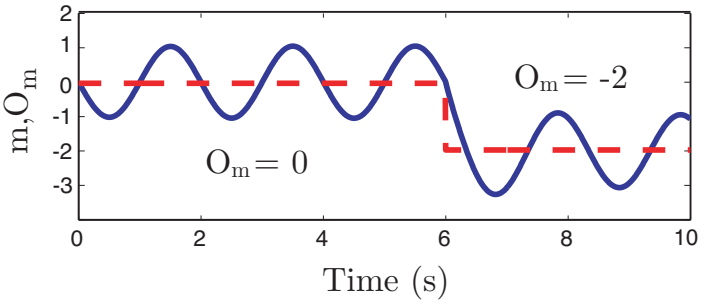


FIGURE A.8: Offset O_{m} (dashed red line) and state variable m (blue continuous line) superimposed. During the interval of time $0 < t < 6$s, $O_{\mathrm{m}} = 0$. At $t = 6$s, O_{m} is changed to -2 and the system quickly converges to oscillate around that value.

A.4 C^1 Dynamical Systems

Henceforward, it is assumed that $\mathrm{sgn}(x)$ refers a signum function of a real number x,

$$\mathrm{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{A.7}$$

A dynamical system or a feed-through map is said to be C^1 if their first derivatives exist and are continuous in $\mathbb{R}$.

Feed-through map f_2 receives the distance, u_k, between the robot's position, P_{r}, and the goal location, P_{b},

$$f_2 \triangleq e_k = \tanh\left(b\ \left(u_k - 2\tau\right)\right). \tag{A.8}$$

Its first derivative is described as follows,

$$\frac{\partial f_2}{\partial u_k} = -b\,(\tanh(b\,(2\tau - u_k))^2 - 1), \tag{A.9}$$

which is clearly continuous in $\mathbb{R}$.

Feed-through map f_3 receives parameter e_k given by f_2,

$$f_3 \triangleq L_k = \tanh(b\,e_k) l_{i,i+1} + \tanh(b\,(1-e_k)) l_{i+1,i+2}. \tag{A.10}$$

Its first derivatives are described as follows,

$$\begin{aligned} \frac{\partial f_3}{\partial e_k} &= -b\,l_{i,i+1}\tanh\Big((b\,e_k)^2 - 1\Big) + b\,l_{i+1,i+2}\tanh\Big((b\,(1-e_k)^2) - 1\Big), \\ \frac{\partial f_3}{\partial l_{i,i+1}} &= \tanh(b\,e_k), \\ \frac{\partial f_3}{\partial l_{i+1,i+2}} &= \tanh(b\,(1-e_k)), \end{aligned} \tag{A.11}$$

and clearly, they are continuous in $\mathbb{R}$.

The first derivatives of the dynamical systems f_4 relative to x_{b_k} and f_5 relative to y_{b_k} are described as follows,

$$\frac{\partial f_4}{\partial x_{\mathrm{b}_k}} = 1 + \lambda_{\mathrm{tr}} d_k, \tag{A.12}$$

$$\frac{\partial f_5}{\partial y_{\mathrm{b}_k}} = 1 + \lambda_{\mathrm{tr}} d_k. \tag{A.13}$$

Clearly, the first derivatives of f_4 and f_5 are continuous in $\mathbb{R}$.

For the analysis of feed-through map f_6, consider that $x_k = \frac{y_{\mathrm{b}_k} - y_{\mathrm{r}_k}}{x_{\mathrm{b}_k} - x_{\mathrm{r}_k}}$. Thus, f_6 can be written as,

$$f_6 \triangleq \psi_{\mathrm{tar}_k} = \arctan(x_k), \tag{A.14}$$

and its first derivative is given by,

$$\frac{\partial f_6}{\partial x_k} = \frac{1}{x_k^2 + 1}, \tag{A.15}$$

which is clearly continuous in $\mathbb{R}$.

Feed-through maps f_{7_a} and f_{7_b} are similar and for the purposes of the C^1 analysis, it is assumed that $x_k = \left(\tan\left(\frac{\Delta\theta}{2}\right) + \frac{R_{\mathrm{robot}}}{R_{\mathrm{robot}} + d_{\mathrm{l},i}}\right)$ and $y_k = \left(\tan\left(\frac{\Delta\theta_{\mathrm{s}}}{2}\right) + \frac{R_{\mathrm{robot}}}{R_{\mathrm{robot}} + d_{\mathrm{s},i}}\right)$. Their

first derivatives are given as follows,

$$\frac{\partial f_{7_a}}{\partial x_k} = \frac{1}{x_k^2+1}, \tag{A.16}$$

$$\frac{\partial f_{7_b}}{\partial y_k} = \frac{1}{y_k^2+1}, \tag{A.17}$$

which are continuous in $\mathbb{R}$.

The derivatives of the dynamical system f_8 are given as,

$$\begin{aligned}
\frac{\partial f_8}{\partial \phi_{k+1}} &= 1 + \Bigg(-\sum_{i=1}^{N_l} \frac{\lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}}\right] 2\left(\phi_k-\psi_{\text{obs}}\right)^2}{\sigma^2_{\text{obs},i_k}} \\
&+ \lambda_{\text{obs},i} \exp\left[\frac{-\left(\phi_k-\psi_{\text{obs},i}\right)^2}{2\sigma^2_{\text{obs},i_k}}\right] - \sum_{i=1}^{N_s} \frac{\lambda_{\text{sobs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i_k}}\right] (\phi_k^2-\psi_{\text{sobs}}^2)}{\sigma^2_{\text{sobs},i_k}} \\
&+ \lambda_{\text{sobs},i} \exp\left[\frac{-\left(\phi_k-\psi_{\text{sobs},i}\right)^2}{2\sigma^2_{\text{sobs},i_k}}\right] + \lambda_{\text{tar}} \cos\left(\phi_k-\psi_{\text{tar}}\right)\Bigg) d_k, \qquad \text{(A.18)} \\
\frac{\partial f_8}{\partial \sigma_{\text{obs},i_k}} &= \sum_{i=1}^{N_l} \lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}}\right] \frac{(\phi_k-\psi_{\text{obs}})^3}{\sigma^3_{\text{obs},i_k}} d_k, \qquad \text{(A.19)} \\
\frac{\partial f_8}{\partial \sigma_{\text{sobs},i_k}} &= \sum_{i=1}^{N_s} \lambda_{\text{sobs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i_k}}\right] \frac{(\phi_k-\psi_{\text{sobs}})^3}{\sigma^3_{\text{sobs},i_k}} d_k, \qquad \text{(A.20)} \\
\frac{\partial f_8}{\partial \psi_{\text{obs},i_k}} &= \sum_{i=1}^{N_l} \Bigg(\frac{\lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}}\right] 2(\phi_k-\psi_{\text{obs}})^2}{2\sigma^2_{\text{obs},i_k}} \\
&- \lambda_{\text{obs}} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})^2}{2\sigma^2_{\text{obs},i_k}}\right]\Bigg) d_k, \qquad \text{(A.21)} \\
\frac{\partial f_8}{\partial \psi_{\text{sobs},i_k}} &= \sum_{i=1}^{N_s} \Bigg(\frac{\lambda_{\text{sobs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i_k}}\right] 2(\phi_k-\psi_{\text{sobs}})^2}{2\sigma^2_{\text{sobs},i_k}} \\
&- \lambda_{\text{sobs}} \exp\left[\frac{-(\phi_k-\psi_{\text{sobs},i})^2}{2\sigma^2_{\text{sobs},i_k}}\right]\Bigg) d_k, \qquad \text{(A.22)} \\
\frac{\partial f_8}{\partial \psi_{\text{tar}}} &= -\lambda_{\text{tar}} \cos(\phi_k-\psi_{\text{tar}}) d_k. \qquad \text{(A.23)}
\end{aligned}$$

which are continuous in $\mathbb{R}$.

The derivatives of feed-through map f_9 are given as follows,

$$\frac{\partial f_9}{\partial \phi_k} = \sum_{i=1}^{N_l} -\frac{2\lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})}{2\sigma^2_{\text{obs},i_k}}\right](\phi_k - \psi_{\text{obs},i})}{2\sigma_{\text{obs},i_k}}, \tag{A.24}$$

$$\frac{\partial f_9}{\partial \psi_{\text{obs},i}} = \sum_{i=1}^{N_l} \frac{2\lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})}{2\sigma^2_{\text{obs},i_k}}\right](\phi_k - \psi_{\text{obs},i})}{2\sigma_{\text{obs},i_k}}, \tag{A.25}$$

$$\frac{\partial f_9}{\partial \sigma_{\text{obs},i_k}} = \sum_{i=1}^{N_l} \lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k - \psi_{\text{obs},i})}{2\sigma^2_{\text{obs},i_k}}\right] - \frac{2\lambda_{\text{obs},i}\sigma_{\text{obs},i_k}}{\sqrt{e}} \tag{A.26}$$

$$+ \frac{2\lambda_{\text{obs},i} \exp\left[\frac{-(\phi_k-\psi_{\text{obs},i})}{2\sigma^2_{\text{obs},i_k}}\right](\phi_k - \psi_{\text{obs},i})^2}{\sigma^2_{\text{obs},i_k}}, \tag{A.27}$$

which are continuous in $\mathbb{R}$.

The derivatives of the Stuart-Landau oscillator are given as,

$$\begin{bmatrix} \frac{\partial f_{10}}{\partial m_{k+1}} & \frac{\partial f_{10}}{\partial n_{k+1}} & \frac{\partial f_{10}}{\partial \mu_k} & \frac{\partial f_{10}}{\partial O_{\text{m}_k}} & \frac{\partial f_{10}}{\partial r_k} & \frac{\partial f_{10}}{\partial \omega_k} \\ \frac{\partial f_{11}}{\partial m_{k+1}} & \frac{\partial f_{11}}{\partial n_{k+1}} & \frac{\partial f_{11}}{\partial \mu_k} & \frac{\partial f_{11}}{\partial O_{\text{m}_k}} & \frac{\partial f_{11}}{\partial r_k} & \frac{\partial f_{11}}{\partial \omega_k} \end{bmatrix} =$$

$$\begin{bmatrix} 1+\alpha(\mu_k - r_k^2)d_k & -\omega_k d_k & -\alpha(O_{\text{m}_k} - m_k)d_k & -\alpha(-r_k^2+\mu_k)d_k & 2\alpha r_k(O_{\text{m}_k} - m_k)d_k & -n_k d_k \\ \omega_k d_k & 1+\alpha(\mu_k - r_k^2)d_k & \alpha n_k d_k & -\omega_k d_k & -2\alpha n_k r_k d_k & (m_k - O_{\text{m}_k})d_k \end{bmatrix}, \tag{A.28}$$

which are clearly continuous in $\mathbb{R}$.

The first derivatives of feed-through maps f_{12} and f_{13} relative to state variables m_k and n_k are described as follows,

$$\begin{aligned}
\frac{\partial f_{12}}{\partial m_k} &= -\frac{A_{1_k}\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(m_k-O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[b\ n_k\right]+1\right)}+\frac{A_{2_k}\ b\ \exp\left[b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2} \\
&- \frac{A_{3_k}\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[-b\ n_k\right]+1\right)}, \qquad \text{(A.29)} \\
\frac{\partial f_{12}}{\partial n_k} &= -\frac{A_{1_k}\ b\ \exp\left[b\ n_k\right]}{\left(\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)\left(\exp\left[b\ n_k\right]+1\right)\right)^2} \\
&+ \frac{A_{3_k}\ b\ \exp\left[-bn_k\right]}{\left(\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)\left(\exp\left[-b\ n_k\right]+1\right)\right)^2}, \qquad \text{(A.30)} \\
\frac{\partial f_{12}}{\partial O_{\mathrm{m}_k}} &= \frac{A_{1_k}\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(m_k-O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[b\ n_k\right]+1\right)}+\frac{A_{2_k}\ b\ \exp\left[b(m_k-O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2} \\
&+ \frac{A_{3_k}\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[-b\ n_k\right]+1\right)}, \qquad \text{(A.31)} \\
\frac{\partial f_{13}}{\partial m_k} &= -\frac{\omega_1\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(m_k-O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[b\ n_k\right]+1\right)}+\frac{\omega_2\ b\ \exp\left[b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2} \\
&- \frac{\omega_3\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[-b\ n_k\right]+1\right)}, \qquad \text{(A.32)} \\
\frac{\partial f_{13}}{\partial n_k} &= -\frac{\omega_1\ b\ \exp\left[b\ n_k\right]}{\left(\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)\left(\exp\left[b\ n_k\right]+1\right)\right)^2} \\
&+ \frac{\omega_3\ b\ \exp\left[-bn_k\right]}{\left(\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)\left(\exp\left[-b\ n_k\right]+1\right)\right)^2}, \qquad \text{(A.33)} \\
\frac{\partial f_{13}}{\partial O_{\mathrm{m}_k}} &= \frac{\omega_1\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(m_k-O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[b\ n_k\right]+1\right)}+\frac{\omega_2\ b\ \exp\left[b(m_k-O_{\mathrm{m}_k})\right]}{\left(\exp\left[b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2} \\
&+ \frac{\omega_3\ b\ \exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]}{\left(\exp\left[-b(-m_k+O_{\mathrm{m}_k})\right]+1\right)^2\left(\exp\left[-b\ n_k\right]+1\right)}, \qquad \text{(A.34)}
\end{aligned}$$

and they are continuous in $\mathbb{R}$ if A_1, A_2, A_3, ω_1, ω_2 and ω_3 are replaced without lack of generality by C^1 maps using splines [280] or other regularization methods [281, 282].

The transition between A_1 and A_2 and ω_1 and ω_2 occurs when $m = O_{\mathrm{m}}$ and $n < 0$. Assuming a sufficient positive b and substituting into (A.29), (A.30), (A.31), (A.32),

(A.33), (A.34),

$$\frac{\partial f_{12}}{\partial m_k} = \frac{b(A_{2_k} - A_{1_k})}{2^2}, \tag{A.35}$$

$$\frac{\partial f_{12}}{\partial n_k} = \frac{bA_{3_k}}{2^2}, \tag{A.36}$$

$$\frac{\partial f_{12}}{\partial O_{\mathrm{m}_k}} = \frac{b(A_{1_k} + A_{2_k})}{2^2}, \tag{A.37}$$

$$\frac{\partial f_{13}}{\partial m_k} = \frac{b(\omega_2 - \omega_1)}{2^2}, \tag{A.38}$$

$$\frac{\partial f_{13}}{\partial n_k} = \frac{b\omega_3}{2^2}, \tag{A.39}$$

$$\frac{\partial f_{13}}{\partial O_{\mathrm{m}_k}} = \frac{b(\omega_1 + \omega_2)}{2^2}. \tag{A.40}$$

This shows that these derivatives exist during the transitions between A_1 and A_2 and ω_1 and ω_2.

The transition between A_2 and A_3 and ω_2 and ω_3 occurs when $m = O_{\mathrm{m}}$ and $n > 0$. Substituting into (A.29), (A.30), (A.31), (A.32), (A.33), (A.34),

$$\frac{\partial f_{12}}{\partial m_k} = \frac{b(A_{2_k} - A_{3_k})}{2^2}, \tag{A.41}$$

$$\frac{\partial f_{12}}{\partial n_k} = -\frac{bA_{1_k}}{2^2}, \tag{A.42}$$

$$\frac{\partial f_{12}}{\partial O_{\mathrm{m}_k}} = \frac{b(A_{1_k} + A_{2_k})}{2^2}, \tag{A.43}$$

$$\frac{\partial f_{13}}{\partial m_k} = \frac{b(\omega_2 - \omega_1)}{2^2}, \tag{A.44}$$

$$\frac{\partial f_{13}}{\partial n_k} = -\frac{b\omega_1}{2^2}, \tag{A.45}$$

$$\frac{\partial f_{13}}{\partial O_{\mathrm{m}_k}} = \frac{b(\omega_2 + \omega_3)}{2^2}. \tag{A.46}$$

This shows that these derivatives exist during the transitions between A_2 and A_3 and ω_2 and ω_3.

Fig. A.9 shows the derivatives of f_{12} relative to variable m (A.29) and to variable n (A.30) for an example of a mission with $MT = 20$ s. f_{12} is initiated with A_1 and starts changing to A_2 at $t \approx 6.5$ s. At this instant of time, $\frac{\partial f_{12}}{\partial m_k} = \frac{b(A_{2_k} - A_{3_k})}{2^2}$ and $\frac{\partial f_{12}}{\partial n_k} = -\frac{bA_{1_k}}{2^2}$. At $t \approx 13.5$ s, f_{12} starts changing to A_3 and $\frac{\partial f_{12}}{\partial m_k} = \frac{b(\omega_2 - \omega_1)}{2^2}$ and $\frac{\partial f_{13}}{\partial n_k} = -\frac{b\omega_1}{2^2}$. It is noticeable that the larger b, the higher is the derivative and thus the faster is the transition between the variables. This is also valid for the feed-through map f_{13}. Fig. A.10 shows the derivative (A.31) according to the different values of b.

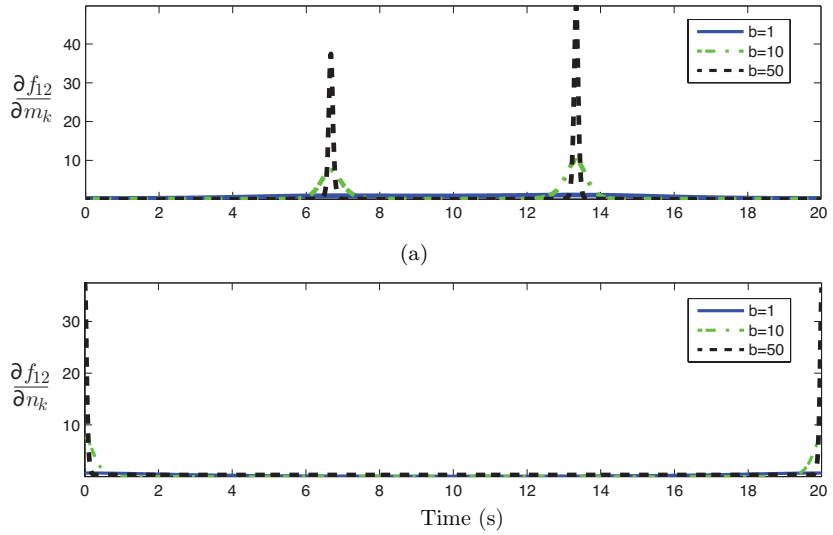


FIGURE A.9: Derivatives of the feed-through map f_{12} relative to variables m and n.

They are only shown during an interval of 20 s, but they are continuous in $\mathbb{R}$.

The first derivatives of feed-through map f_{14} are described as follows,

$$\frac{\partial f_{14}}{\partial A'_k} = \frac{1}{1+\exp\left[-bU_k(\phi_k)\right]} + \frac{d\left(\frac{1}{2}\left(La_i + S_i - \sqrt{(La_i+S_i)^2}\right)\right)^c}{1+\exp\left[bU_k(\phi_k)\right]}, \tag{A.47}$$

$$\begin{aligned}\frac{\partial f_{14}}{\partial U_k(\phi_k)} &= \frac{A_k b \exp\left[-U_k(\phi_k)b\right]}{\left(\exp(U_k(\phi_k)b)+1\right)\left(\exp(-U_k(\phi_k)b)+1\right)^2)} \\ &- \frac{A_k b \exp\left[U_k(\phi_k)b\right]\left(\frac{1}{\exp[-U_k(\phi_k)b]}+1\right)\frac{d(La_i+S_i-\sqrt{(La_i+S_i)^2}^c)}{2}}{\left(\exp\left[U_k(\phi_k)b\right]+1\right)^2},\end{aligned} \tag{A.48}$$

$$\frac{\partial f_{14}}{\partial La_i} = \frac{\partial f_{14}}{\partial S_i} = -\frac{\left(A_k\ c\ d\ \frac{La_i+S_i}{\sqrt{(La+S_i)^2}}-1\right)\left(La_i+S_i-\sqrt{(La+S_i)^2}^{(c-1)}\right)}{2\exp\left[U_k(\phi_k)\ b\right]+1}, \tag{A.49}$$

which are continuous in $\mathbb{R}$.

The derivatives of dynamical system f_{15} relative to u_{i_k}, β_i and ν are given as,

$$\frac{\partial f_{15}}{\partial u_{i_k}} = \frac{1}{\tau_\mu} + \left(\beta_i - 3u_{i_k}^2\sqrt{\beta_i^2+\eta} - \nu\sum_{a\neq i}u_{a_k}^2\right)\frac{d_k}{\tau_\mu}, \tag{A.50}$$

$$\frac{\partial f_{15}}{\partial \beta_i} = \frac{u_{i_k}}{\tau_\mu} - \frac{\beta_i u_{i_k}^3}{\sqrt{\beta_i^2+\eta}}\frac{d_k}{\tau_\mu}, \tag{A.51}$$

$$\frac{\partial f_{15}}{\partial \nu} = -\sum_{a\neq i}u_{a_k}^2 u_{i_k}\frac{d_k}{\tau_\mu}, \tag{A.52}$$

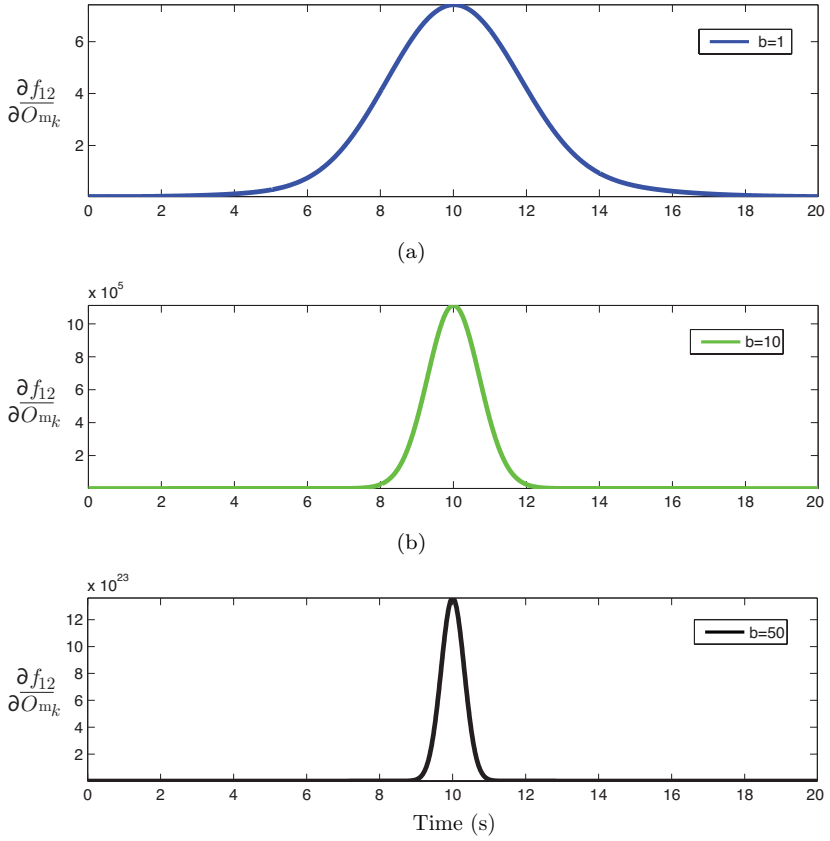


FIGURE A.10: Derivative of the feed-through map f_{12} relative to variable O_{m}.

which are continuous in $\mathbb{R}$.

The first derivatives of feed-through maps f_{16} and f_{17} are described as follows,

$$\frac{\partial f_{16}}{\partial u_{\mathrm{stop}_k}} = O_{\mathrm{s}}\, b(\tanh(b\, u_{\mathrm{stop}_k})^2 - 1), \tag{A.53}$$

$$\frac{\partial f_{16}}{\partial u_{\mathrm{execution}_k}} = O_{\mathrm{e}}\, b(\tanh(b\, u_{\mathrm{execution}_k})^2 - 1), \tag{A.54}$$

$$\frac{\partial f_{16}}{\partial u_{\mathrm{rescue}_k}} = O_{\mathrm{r}}\, b(\tanh(b\, u_{\mathrm{rescue}_k})^2 - 1), \tag{A.55}$$

$$\frac{\partial f_{17}}{\partial u_{\mathrm{stop}_k}} = \frac{\partial f_{17}}{\partial u_{\mathrm{rescue}_k}} = -\frac{A_k^2\, b(\tanh(b(u_{\mathrm{stop}_k} + u_{\mathrm{rescue}_k}))^2 - 1)}{2}, \tag{A.56}$$

$$\frac{\partial f_{17}}{\partial u_{\mathrm{execution}_k}} = A^2\, b(\tanh(b\, u_{\mathrm{rescue}_k})^2 - 1), \tag{A.57}$$

and clearly they are continuous in $\mathbb{R}$.

Appendix B

B.1 Robot Features

Pioneer 3-DX is a commercial mobile robot steered by two lateral motorized wheels and supported by a passive rear caster wheel for motion stability. Each motorized wheel has a diameter of 0.195 m, and a motor with 500-tick encoders that guarantees an accurate control of velocity. Fig. B.1 depicts a schematic of lateral and top perspectives of the robot.

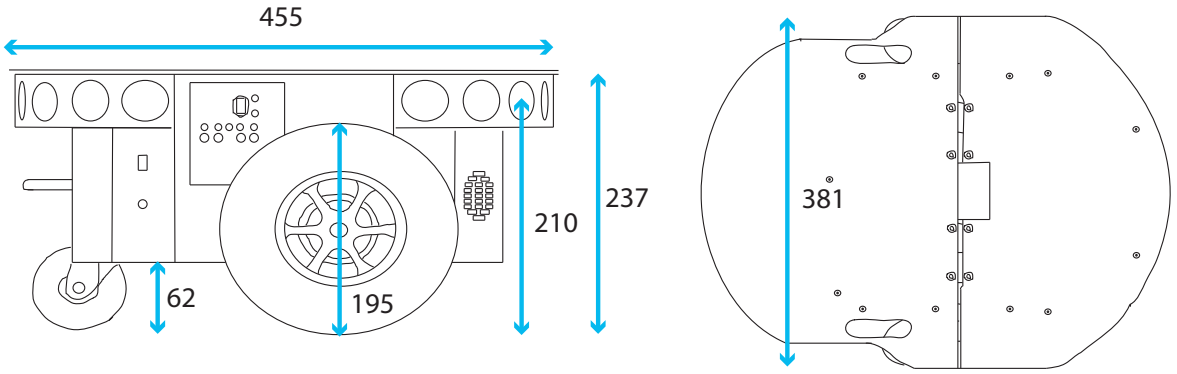


FIGURE B.1: Lateral and top perspectives of the robot. Measures are in centimeters.

The robot is supplied by 12 V, weighs approximately 9 kg and is able to carry a payload of approximately 17 kg. It can reach a maximum forward/backward speed of $v_{\max} = 0.8$ m/s and a rotation speed of 300 °/s. The robot radius is $R_{\text{robot}} = 0.1905$ m. At runtime, Pioneer 3-DX is powered by a maximum of 3 sealed batteries, each one with a capacity of 7.2 Ah, which gives the robot an operating time of 8-10 hours.

The robot is equipped with 8 forward-facing ultrasonic (sonar) sensors mounted on a ring, used to measure the environment (see fig. B.2). Each sonar has a beam width of 30°, and a range of 0.2 m to 5 m. The distance to an object is determined by the time of flight of an acoustic signal generated by the transducer and reflected by the detected object.

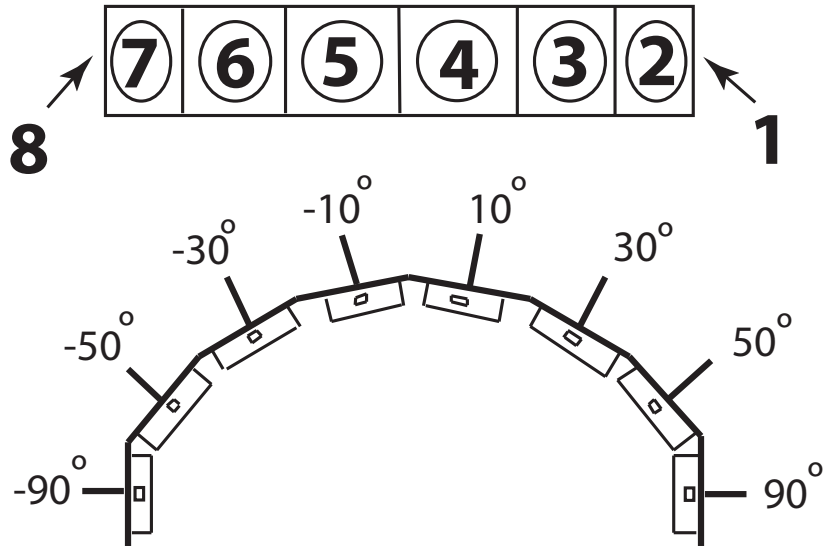


FIGURE B.2: Distribution of the 8 sonar sensors mounted in front of the robot. They are arranged at angles -90°,-50°,-30°,-10°,10°,30°,50°,90°.

The robot has an embedded controller that provides the robot's state and control information that includes battery charge data and sonar range sensing data. A single laptop with a processor operating at 2.4 GHz is mounted on the robot in order to run the ARIA framework. This framework is developed for controlling all MobileRobots platforms that includes the Pioneer 3-DX. ARIA provides a framework for communication with the robot and that includes receiving and sending data from the robot.

B.1.1 Kinematics

Robots with a two-wheel differential drive and a passive caster wheel offers a simple mechanical structure and a simple kinematic model. They allow a zero turning radius and the obstacle-free space can easily be computed by extending obstacle boundaries by the robot radius.

The kinematics model for a two-wheel differential drive robot specify the robot's linear velocity, v, and the robot's angular velocity, ϖ by setting the rotational velocity for each robot's wheel, namely, ϖ_{lw} and ϖ_{rw} for left and right wheels, respectively, as follows,

$$\varpi_{\text{lw}} = \frac{1}{R_{\text{wheel}}}\left(v - \varpi\left(\frac{D_{\text{wheel}}}{2}\right)\right), \tag{B.1}$$

$$\varpi_{\text{rw}} = \frac{1}{R_{\text{wheel}}}\left(v + \varpi\left(\frac{D_{\text{wheel}}}{2}\right)\right), \tag{B.2}$$

where $R_{\text{wheel}} = 0.0975$ m and $D_{\text{wheel}} = 0.381$ m are respectively the wheels radius and the distance between the wheels.

B.2 Sensors

The laser Hokuyo URG-04LX-UG01 was selected since its features are well-applied to our robotic application. It provides high accuracy in the range of 60 - 5600 mm, namely ±30 mm in range 60-1000 mm, and ±3% of measurement in range 1000-4095 mm. Its angular measuring is 240° with a step angle of approximately 0.36°, which gives 682 values for each scan. The laser scans the environment at each 0.1 s. Fig. B.3 (a) shows the laser Hokuyo URG-04LX-UG01.

The PsEye is a digital camera device trademarked by Playstation. It is capable of capturing standard images at 60 Hz at a pixel resolution 640x480. In addition, the camera features a two-setting adjustable fixed focus zoom lens. It is possible to manually select the field of view, which alternates between 56° for close-up framing, and 75° for long shot framing. Fig. B.3 (b) shows the PsEye camera.

(a) (b)

FIGURE B.3: (a) Laser Range Finder Hokuyo URG-04LX-UG01. (b) PsEye camera.

Bibliography

[1] C. Santos. Generating timed trajectories for an autonomous vehicle:a non-linear dynamical systems approach. *in IEEE International Conference on Robotics and Automation ICRA'04*, 4:3741 – 3746, 2004.

[2] S. Degallier, C. Santos, L. Righetti, and A. Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[3] M. Tuma, I. Iossifidis, and G. Schöner. Temporal stabilization of discrete movement in variable environments: an attractor dynamics approach. *in IEEE International Conference on Robotics and Automation ICRA'09*, pages 863 – 868, 2009.

[4] G. Schöner. Timing, clocks, and dynamical systems. *Brain and Cognition*, 48 (1):31–51, 2002.

[5] M. D. Rossetti, R. A. Felder, and A. Kumar. Simulation of robotic courier deliveries in hospital distribution services. *Health care management science*, 3 (3):201 – 213, 2000.

[6] K. Niechwiadowicz and K. Zahoor. Robot based logistics system for hospitals-survey. *IDT Workshop on Interesting Results in Computer Science and Engineering*, 3, 2008.

[7] A. Özkil, Z. Fan, S. Dawids, H. Aanæs, J. Kristensen, and K. Christensen. Service robots for hospitals: A case study of transportations tasks in a hospital. *Proc. of the IEEE International Conference on Automation and Logistics, Shenyang, China*, pages 289–294, 2009.

[8] J. Slotine and W. Li. *Applied Nonlinear Control*, volume 199. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[9] G. Schöner and C. Santos. Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. *in Proc. of the 9th Int. Symposium on Intelligent Robotic Systems (SIRS)*, 2001.

[10] K. Ogata and Y. Yanjuan. *Modern control engineering*. Prentice-Hall Englewood Cliffs, 1970.

[11] W. Lohmiller and J. Slotine. On contraction analysis for non-linear sytems. *Automatica*, 34(6):683 – 696, 1998.

[12] J. Slotine and W. Lohmiller. Modularity, evolution, and the binding problem: A view from stability theory. *Neural networks*, 14(2):137 – 145, 2001.

[13] J. Silva, C. Santos, and V. Matos. Generating trajectories with temporal constraints for an autonomous robot. *In 8th IEEE International Workshop on Safety, Security & Rescue, Bremen, Germany, July 26-30*, 2010.

[14] International federation of robotics ifr. @ONLINE [accessed 07 november 2014]. URL `http://www.ifr.org/service-robots/statistics/`.

[15] Roomba vacuum cleaning robot irobot inc. @ONLINE [accessed 22 april 2014]. URL `www.irobot.com/us/learn/home/roomba.aspx`.

[16] Scooba floor scrubbing robot from irobot inc. @ONLINE [accessed 22 april 2014]. URL `www.irobot.com/us/learn/home/scooba.aspx`.

[17] Vacuuming robot from agait technology corporation inc. @ONLINE [accessed 22 april 2014]. URL `www.agaitech.com/en/products_list.aspx?xs_class=90`.

[18] Deebot vacuuming robot from ecovacs inc. @ONLINE [accessed 22 april 2014]. URL `www.ecovacs.com/bot/Deebot-D76.html`.

[19] J. Forlizzi and C. DiSalvo. Service robots in the domestic environment: a study of the roomba vacuum in the home. *1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265, 2006.

[20] Hom-bot robot vacuum cleaner from lg corp.@ONLINE [accessed 22 april 2014]. URL `www.lg.com/us/vacuum-cleaners/lg-LrV5900-robot-vacuum`.

[21] Navibot s robot vacuum from samsung group. @ONLINE [accessed 22 april 2014]. URL www.samsung.com/au/consumer/home-appliances/vacuum-cleaner/robot-vacuum/VCR8980L4K/XSA.

[22] Ottoro from hanool robotic inc. @ONLINE [accessed 22 april 2014]. URL www.robotbg.com/robots/floor_cleaners/hanool/ottoro.

[23] Hydrobot, aerobot and duobot from intellibot robotics inc.@ONLINE [accessed 22 april 2014]. URL www.intellibotrobotics.com/products/.

[24] Lawn mowers from ambrogio robot s.p.a.@ONLINE [accessed 22 april 2014]. URL www.ambrogiorobot.com/.

[25] Indego robotic lawn mower from bosch gmbh@ONLINE [accessed 22 april 2014]. URL www.bosch-indego.com/gb/en/.

[26] Automower from husqvarna ab @ONLINE [accessed 22 april 2014]. URL www.husqvarna.com/us/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners/.

[27] Robomow rs630 from robomow@ONLINE [accessed 22 april 2014], . URL www.bosch-indego.com/gb/en/.

[28] Tango e5 from john deere inc.@ONLINE [accessed 22 april 2014]. URL www.deere.com/wps/dcom/en_INT/products/equipment/autonomous_mower/tango_e5/tango_e5.page.

[29] Miimo from honda motor co. ltd@ONLINE [accessed 22 april 2014]. URL www.honda.co.uk/garden/miimo/.

[30] Lawn mowers from belrobotics sa @ONLINE [accessed 22 april 2014]. URL www.belrobotics.com/index.php/en/products/greenmow.

[31] Golmow from selftech@ONLINE [accessed 30 october 2014]. URL http://www.selftech.pt/news/10.

[32] Robotic lawn mower lb300el from lawnbott @ONLINE [accessed 22 april 2014]. URL www.lawnbott.com/products/lb300el/.

[33] Rp-7i robot from intouch health inc.@ONLINE [accessed 22 april 2014]. URL www.intouchhealth.com/products-and-services/products/rp-vita-robot/.

[34] Ava 500 robot from irobot inc. @ONLINE [accessed 22 april 2014]. URL `www.irobot.com/us/learn/commercial/ava500.aspx`.

[35] Smart service robot furo-s from future robot co.ltd @ONLINE [accessed 22 april 2014], . URL `www.futurerobot.co.kr/en/page/product01.php`.

[36] Smart service robot furo-k from future robot co.ltd @ONLINE [accessed 22 april 2014], . URL `www.futurerobot.co.kr/en/page/product02.php#tab2`.

[37] Kompai from robotsoft@ONLINE [accessed 22 april 2014], . URL `www.robosoft.com/robotic-solutions/healthcare/kompai/kompai-rd.html`.

[38] M. Nani, P. Caleb-Solly, S. Dogramadzi, T. Fear, and H. van den Heuvel. Mobiserv: an integrated intelligent home environment for the provision of health, nutrition and mobility services to the elderly. 2010.

[39] J. González-Jiménez, C. Galindo, and J. Ruiz-Sarmiento. Technical improvements of the giraff telepresence robot based on users' evaluation. *IEEE RO-MAN*, pages 827 – 832, 2012.

[40] U. Reiser, T. Jacobs, G. Arbeiter, C. Parlitz, and K. Dautenhahn. Care-o-bot 3 vision of a robot butler. *Your virtual butler*, pages 97 – 116, 2013.

[41] Amigo robot from tech united @ONLINE [accessed 22 april 2014]. URL `www.techunited.nl/en/amigo`.

[42] M. E. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, and N. Roy. Pearl: A mobile robotic assistant for the elderly. *AAAI workshop on automation as eldercare*, 2002:85 – 91, 2002.

[43] J. Biswas and M. M.. Veloso. Localization and navigation of the cobots over long-term deployments. *International Journal of Robotics Research*, 32(14): 1679 – 1694, 2013.

[44] Santander siga from ydreams. @ONLINE [accessed 30 october 2014]. URL `http://www.ydreamsrobotics.com/projects/`.

[45] c-walker from siemens ag@ONLINE [accessed 22 april 2014]. URL `www.siemens.com/innovation/en/news/2013/e_inno_1332_2.htm`.

[46] D. Rodriguez-Losada, F. Matia, A. Jimenez, R. Galan, and G. Lacey. Guido, the robotic smartwalker for the frail visually impaired. *In 1st International*

Congress on Domotics, Robotics and Remote Assistance for All. DRT4ALL, 5: 155–169, 2006.

[47] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1), 2008.

[48] J. Evans, B. Krishnamurthy, B. Barrows, T. Skewis, and V. Lumelsky. Handling real-world motion planning: A hospital transport robot. *Control Systems, IEEE*, 12(1):15 – 19, 1992.

[49] J. M Evans. Helpmate: An autonomous mobile robot courier for hospitals. *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. Advanced Robotic Systems and the Real World*, 3: 1695–1700, 1994.

[50] M. D. Rossetti, A. Kumar, and R.A. Felder. Mobile robot simulation of clinical laboratory deliveries. *Winter Simulation Conference*, pages 1415 – 1422, 1998.

[51] M. D. Rossetti and F. Selandari. Multi-objective analysis of hospital delivery systems. *Computers & industrial engineering*, 41(3):309 – 333, 2001.

[52] B. Mutlu and J. Forlizzi. Robots in organizations: The role of workflow, social, and environmental factors in human-robot interaction. *3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 287 – 294, 2008.

[53] S. Ljungblad, J. Kotrbova, M. Jacobsson, H. Cramer, and K. Niechwiadowicz. Hospital robot at work: something alien or an intelligent colleague? *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 177 – 186, 2012.

[54] W. Fung, Y. Leung, M. Chow, Y. Liu, Y. Xu, W. Chan, and T. Law. Development of a hospital service robot for transporting task. *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, 1:628Ű–633, 2003.

[55] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *In Artificial Intelligence Laboratory Stanford University, Stamford*, 1985.

[56] F. Carreira, T. Canas, A. Silva, and C. Caldeira. i-merc: a mobile robot to deliver meals inside health services. *IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–8, 2006.

[57] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58:889 – 899.

[58] F. Capezio, F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, P. Vernazza, T. Vernazza, and R. Zaccaria. Mobile robots in hospital environments: An installation case study. *Proceedings of the 5th International Conference on Mobile Robots, Orebro, Sweden*, pages 7–9, 2011.

[59] A. Sgorbissa and R. Zaccaria. Roaming stripes: smooth reactive navigation in a partially known environment. *The 12th IEEE International Workshop on Robot and Human Interactive Communication, ROMAN*, pages 19 – 24, 2003.

[60] G. Engelberger. Helpmate, a service robot with experience. *Industrial Robot*, 25(2):101–104, 1998.

[61] Robocart from california computer research inc. @ONLINE [accessed 20 december 2013], . URL `www.robocart.com`.

[62] Tug from aethon inc. @ONLINE [accessed 20 december 2013]. URL `www.aethon.com`.

[63] Robocourier from swisslog inc. @ONLINE [accessed 20 december 2013], . URL `www.ccsrobotics.com/products/robocourier.html`.

[64] Speciminder from swisslog inc. @ONLINE [accessed 20 december 2013]. URL `www.ccsrobotics.com/products/speciminder.html`.

[65] R. Murai, T. Sakai, H. Uematsu, H. Nakajima, K. Mitani, and H. Kitano. Conveyance system using autonomous mobile robots. *In IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 54 – 59, 2009.

[66] Qc bot from vecna inc. @ONLINE [accessed 20 december 2013]. URL `www.vecna.com/on-demand-delivery`.

[67] Transcar from swisslog inc. @ONLINE [accessed 20 december 2013]. URL `www.prweb.com/releases/2013/5/prweb10661451.htm`.

[68] Y. K. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.

[69] V. Lumelsky and A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430, 1987.

[70] A. Chakravarthy and D Ghose. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(5):562–574, 1998.

[71] R Chattergy. Some heuristic for the navigation of a robot. *The International Journal of Robotics Research*, 4(1):59–66, 1985.

[72] A. Elnagar and A. Basu. Heuristics for local path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 23(2):624–634, 1993.

[73] J. Gasós and A. Martín. Mobile robot localization using fuzzy maps. *Fuzzy Logic in Artificial Intelligence Towards Intelligent Systems*, pages 624–634, 1997.

[74] G. Schöner and M. Dose. A dynamical system approach to task-level system integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, 10(4):253–267, 1992.

[75] J-O Kim, , and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3): 338–349, 1992.

[76] L. Singh, H. Stephanon, and J. Wen. Real-time robot motion control with circulatory fields. *IEEE International Conference on Robotics and Automation*, 3:2737–2742, 1996.

[77] A. Masoud, S. Masoud, and M. Bayoumi. Robot navigation using a pressure generated mechanical stress field: "the biharmonic potential approach". *IEEE International Conference on Robotics and Automation*, pages 124–129, 1994.

[78] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *In IEEE Transactions on Systems*, 19(5):1179–1187, 1989.

[79] H. Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121, March 1985.

[80] L. Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, 2003.

[81] G. Schöner. Dynamic theory of action - perception patterns: The time-before-contact paradigm. *Human Movement Science*, (3):415–439, 1994.

[82] E. Bicho and G. Schöner. The dynamic approach to autonomous robotics demonstrated on a low-level vehicle plataform. *Robotics and Autonomous Systems*, 21:23–35, 1997.

[83] E. Bicho, P. Mallet, and G. Schöner. Using attractor dynamics to control autonomous vehicle motion. *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society IECON*, 2:1176–1181, 1998.

[84] E. Bicho. Dynamic approach to behavior-based robotics design, specification, analysis, simulation and implementation. *Shaker Verlag, Aachen*, 2000.

[85] J. Silva, C. Santos, and V. Matos. Timed trajectory generation for a toy-like wheeled robot. *In 36th Annual Conference of the IEEE Industrial Electronics Society, Glendale, USA, November 07-10*, pages 1645 – 1650, 2010.

[86] J. Silva, Santos C., and Sequeira J. Navigation architecture for mobile robots with temporal stabilization of movements. *In 9th International Workshop on Robot, Motion and Control*, 2013.

[87] P. Althaus, H. Christensen, and F. Hoffman. Using the dynamical system approach to navigate in realistic real-world environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2:1023 – 1029, 2001.

[88] E. Bicho and S. Monteiro. Formation control for multiple mobile robots: a non-linear attractor dynamics approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2:2016 – 2022, 2003.

[89] B. R. Fajen and W. H. Warren. Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology: Human Perception and Performance*, 29(2):343 Ű– 362, 2003.

[90] E. Aaron, H. Sun, F. Ivancic, and D. Metaxas. A hybrid dynamical systems approach to intelligent low-level navigation. *IEEE Computer Animation*, pages 154 – 163, 2002.

[91] S. Goldenstein, M. Karavelas, D. Metaxas, L. Guibas, E. Aaron, and A. Goswami. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computer & Graphics*, 25(6):983 – 998, 2001.

[92] M. Quoy, S. Moga, and P. Gaussier. Dynamical neural networks for planning and low-level robot control. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 3(4):523–532, 2003.

[93] H. Hoffmann, P. Pastor, D. H. Park, and S. Schaal. Biologically inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. *IEEE International Conference on Robotics and Automation*, pages 2587 – 2592, 2009.

[94] I. Iossifidis and G. Schöner. Dynamical systems approach for the autonomous avoidance of obstacles and joint-limits for an redundant robot arm. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 580 – 585, 2006.

[95] L.P. Ellekilde and H. Christensen. Control of mobile manipulator using the dynamical systems approach. *IEEE International Conference on Robotics and Automation*, pages 1370 – 1376, 2009.

[96] S. M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 32(4):433 – 454, 2012.

[97] J. Borenstein and Y. Koren. The vector field histogramŰfast obstacle avoidance for mobile robots. In *Proceedings of The Ninth International Conference on the SIMULATION OF ADAPTIVE BEHAVIOR – SABŠ06*, volume 7.

[98] I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. *In IEEE International Conference on Robotics and Automation. Leuven, Belgium*, pages 1572–1577, 1998.

[99] R. Simmons. The curvature-velocity method for local obstacle avoidance. *In IEEE International Conference on Robotics and Automation*, pages 3375–3382, 1996.

[100] N. Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3:1615 – 1621, 1998.

[101] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *In IEEE Robot. Autom. Magaz.*, 1(4):23–33, 1997.

[102] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. *In IEEE International Conference on Robotics and Automation*, 1:341–346, 1999.

[103] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.

[104] B. Kluge and E. Prassler. Reflective navigation: individual behaviors and group behaviors. *IEEE International Conference on Robotics and Automation*, 4:4172–4177, 2004.

[105] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. *IEEE International Conference on Robotics and Automation (ICRA)*, 4:1610 –1616, 2007.

[106] T. Myers, L. Vlacic, T. Noel, and M. Parent. Autonomous driving in a time-varying environment. *In IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 53–58, 2005.

[107] J. Minguez and L. Montano. Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. In *IEEE Transactions on Robotics and Automation*, volume 20, pages 3855 – 3862, 2004.

[108] J. Minguez. The obstacle-restriction method for robot obstacle avoidance in difficult environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2284–2290, 2005.

[109] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180 – 197, 1997.

[110] H. A. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4):524 – 539, 2004.

[111] H. R. Beom and K. S. Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE Transactions on Systems, Man and Cybernetics*, 25(3):464 – 477, 1995.

[112] C. Ye, N. Yung, and D. Wang. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE*

Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 33(1):17 – 27, 2003.

[113] A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2):129 – 139, 1995.

[114] D. Pratihar, K. Deb, and A. Ghosh. A genetic-fuzzy approach for mobile robot navigation among moving obstacles. *International Journal of Approximate Reasoning*, 20(2):145 – 172, 1999.

[115] G. N. Marichal, L. Acosta, L. Moreno, J. A. Mendez, J. J. Rodrigo, and M. Sigut. Obstacle avoidance for a mobile robot: A neuro-fuzzy approach. *Fuzzy Sets and Systems*, 124(2):171 – 179, 2001.

[116] N. Tsourveloudis, K. Valavanis, and T. Hebert. Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic. *IEEE Transactions on Robotics and Automation*, 17(4):490 – 497, 2001.

[117] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In *Cartographica: The International Journal for Geographic Information and Geovisualization*, volume 10, pages 112 – 122, October 1973.

[118] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: a factored solution to the simultaneous localization and mapping problem. *Nat. Conf. Artif. Intell. (AAAI)*, pages 593 – 598, 2002.

[119] J. Latombe. Robot motion planning. *Kluwer Academic Publishers*, 1991.

[120] B. J. Oommen, S. S. Iyengar, V. Rao, and R. Kashyap. Robot navigation in unknown terrains using learned visibility graphs. part i: The disjoint convex obstacle case. *IEEE Journal of Robotics and Automation*, 3(6):672 – 681, 1987.

[121] B. R. Donald. Motion planning with six degrees of freedom. *Technical Report AIM-791, MIT Artificial Intelligence Laboratory*, 1984.

[122] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566 – 580, 1996.

[123] S. M. LaValle and J. Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[124] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. *IEEE International Conference on Robotics and Automation*, 2:138 – 145, 1985.

[125] B. J. Kuipers and Y. Byun. A robust qualitative method for spatial learning in unknown environments. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 1988.

[126] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. In *Artificial Intelligence*, volume 99, pages 21 – 71, 1998.

[127] K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *IEEE International Conference on Robotics and Automation (ICRA) 2011*, pages 3041 – 3047, 2011.

[128] Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building and planning based on graph partitioning. In *IEEE International Conference on Robotics and Automation*, pages 803 – 809, 2006.

[129] A. Ozkil, Z. Fan, J. Xiao, J. Kristensen, S. Dawids, K. Christensen, and H. Aanaes. Practical indoor mobile robot navigation using hybrid maps. *Proc. of the 2011 IEEE International Conference on Mechatronics, Istanbul, Turkey*, 2011.

[130] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Simultaneous localization and map building: a global topological model with local metric maps. *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii, USA*, 2001.

[131] F. Fraundorfer, C. Engels, and D. Nister. Topological, mapping, localization and navigation using image collections. *Proceedings of IEEE International Conference on Intelligent Robot and Systems (IROS)*, pages 3872 – 3877, 2007.

[132] D. Rawlinson and R. Jarvis. Topologically-directed navigation. *Robotica*, 26(2): 189 – 203, 2008.

[133] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume 1, pages 269 – 271, 1959.

[134] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics In Systems Science and Cybernetics*, volume 4, pages 100 – 107, February 1968.

[135] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning a*. In *Artificial Intelligence*, volume 155, pages 93 – 146.

[136] A. Stentz. Optimal and efficient path planning for partially-known environments. *In Proceedings IEEE International Conference on Robotics and Automation*, 4:3310–3317, IEEE, May 1994.

[137] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. In *Transactions on Robotics*, volume 21.

[138] R.C. Arkin. Behavior-based robotics. *In MIT Press, Cambridge*, 1998.

[139] R. D. Beer, H. J. Chiel, and L. S Sterling. A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6(1):169–189, 1990.

[140] M. R. Clark, G. T. Anderson, and R. D Skinner. Coupled oscillator control of autonomous mobile robots. *Autonomous Robots*, 9(2):189–198, 2000.

[141] S. Grillner and P. Wallén. Central pattern generators for locomotion, with special reference to vertebrates. *Annual review of neuroscience*, 8(1):233–261, 1985.

[142] S. Hooper. Central pattern generators. *eLS*, 2001.

[143] J. Kelso, J. Scholz, and G. Schöner. Dynamics governs switching among patterns of coordination in biological movement. *in Biological Cybernetics*, 134 (1):8 – 12, 1998.

[144] H. Haken. Synergetics, an introduction: Nonequilibrium phase transitions and self-organization in physics. 1993.

[145] G. Schöner. A dynamic theory of coordination of discrete movement. *in Biological Cybernetics*, 63:257Ű–270, 1990.

[146] G. Schöner. Dynamic theory of action - perception patterns: The “moving room” paradigm. *Biological cybernetics*, 64(6):455–462, 1991.

[147] H. Haken, J. Kelso, and H. Bunz. A theoretical model of phase transitions in human hand movements. *Biological cybernetics*, 51(5):347–356, 1985.

[148] G. Schöner and C. Santos. Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. 2001.

[149] C. Santos and M. Ferreira. Two vision-guided vehicles: Temporal coordination using nonlinear dynamical systems. *in IEEE International Conference on Robotics and Automation, ICRA 2007, April*, 2007.

[150] C. Santos and M. Ferreira. Timed trajectory generation using dynamical systems: Application to a puma arm. *In Robotics and Autonomous Systems*, 57 (2):182–193, 2009.

[151] J. Silva, C. Santos, and J. Sequeira. Timed trajectory generation combined with an extended kalman filter for a vision-based autonomous mobile robot. *Proc. of the 12th IEEE International Conference on Intelligent Autonomous Systems, Jeju Island, Korea*, 2012.

[152] J. Silva, C. Santos, and J. Sequeira. Timed trajectory generation for a vision-based autonomous mobile robot in cluttered environments. *In ICINCO*, pages 431 – 434, 2012.

[153] J. Silva, J. Sequeira, and C. Santos. A stability analysis for a dynamical robot control architecture. *Intelligent Autonomous Vehicles*, 8(1):225–230, 2013.

[154] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 958 – 963. MIT Press, 2002.

[155] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, pages 1547–1554. MIT Press, 2002.

[156] E. Gribovskaya and A. Billard. Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. *3rd ACM/IEEE International Conference on Human-Robot Interaction, HRI08*, pages 33–40, 2008.

[157] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 763–768, 2009.

[158] J. Kober and J. Peters. Imitation and reinforcement learning. *Robotics and Automation Magazine, IEEE*, 17(2):55–62, 2010.

[159] A. Ijspeert, J. Nakanishi, and S. Schall. Movement imitation with nonlinear dynamical systems in humanoid robots. *IEEE International Conference on Robotics and Automation (ICRA)*, 2:1398–1403, 2002.

[160] H. Kimura, Y. Fukuoka, and A. H. Cohen. Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *International Journal of Robotics Research*, 26(5):475–490, 2007.

[161] C. Maufroy, H. Kimura, and K. Takase. Towards a general neural controller for quadrupedal locomotion. *Neural Networks*, 21(4):667–681, 2008.

[162] L. Righetti and A. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 819 – 824, 2008.

[163] V. Matos and C. Santos. Omnidirectional locomotion in a quadruped robot: a cpg-based approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3392–3397, 2010.

[164] T. Komatsu and M. Usui. Dynamic walking and running of a bipedal robot using hybrid central pattern generator method. *IEEE International Conference on Mechatronics and Automation*, 2:987–992, 2005.

[165] S. Hyon, J. Morimoto, and G. Cheng. Hierarchical motor learning and synthesis with passivity-based controller and phase oscillator. *IEEE International Conference on Robotics and Automation*, pages 2705–2710, 2008.

[166] V. Matos and C. Santos. Central pattern generators with phase regulation for the control of humanoid locomotion. *IEEE-RAS International Conference on Humanoid Robots*, pages 134–139, 2012.

[167] S. Inagaki, H. Yuasa, T. Suzuki, and T. Arai. Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118–126, 2006.

[168] R. Guanjiao, W. Chen, C. Kolodziejski, F. Worgotter, S. Dasgupta, and P.e Manoonpong. Multiple chaotic central pattern generators for locomotion generation and leg damage compensation in a hexapod robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2756 – 2761, 2012.

[169] H. Yu, W. Guo, J. Deng, M. Li, and H. Cai. A cpg-based locomotion control architecture for hexapod robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5615 –5621, 2013.

[170] K. Inoue, S. Ma, and C. Jin. Neural oscillator network-based controller for meandering locomotion of snake-like robots. *IEEE International Conference on Robotics and Automation*, 5:5064 – 5069, 2004.

[171] A. Ijspeert, A. Crespi, and J. Cabelguen. Simulation an robotics studies of salamander locomotion: Applying neurobiological principles to the control of locomotion in robots. *NeuroInformatics*, 3:171 – 196, 2005.

[172] A. Crespi, K. Karakasiliotis, A. Guignard, and A. Ijspeert. Salamandra robotica ii: An amphibious robot to study salamander-like swimming and walking gaits. *IEEE Transactions on Robotics*, 29(2):171 – 196, 2013.

[173] S. Kotosaka and S. Schaal. Synchronized robot drumming by neural oscillator. *Journal-Robotics Society of Japan*, 19(1):116–123, 2001.

[174] S. Degallier, C. Santos, L. Righetti, and A. Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. *IEEE-RAS International Conference on Humanoid Robots*, pages 512–517, 2006.

[175] R. Ronsse, N. Vitiello, T. Lenzi, J. van den Kieboom, M. Carrozza, and A. Ijspeert. Human-robot synchrony: flexible assistance using adaptive oscillators. *IEEE Transactions on Biomedical Engineering*, 58(4):1001–1012, 2011.

[176] J. Gonzalez-Gomez, H. Zhang, E. Boemo, and J. Zhang. Locomotion capabilities of a modular robot with eight pitch-yaw-connecting modules. *9th international conference on climbing and walking robots*, 2006.

[177] A. Sprowitz, S. Pouya, S. Bonardi, J. Van den Kieboom, R. Mockel, P. Dillenbourg, A. Billard, and A. Ijspeert. Roombots: reconfigurable robots

for adaptive furniture. *IEEE Computational Intelligence Magazine*, 5(3):20–32, 2010.

[178] X. Cui, Y. Zhu, X. Zang, S. Tang, and J. Zhao. Cpg based locomotion control of pitch-yaw connecting modular self-reconfigurable robots. *IEEE international conference on Information and automation (ICIA)*, pages 1410–1415, 2010.

[179] J. Kober and J. Peters. Learning motor primitives for robotics. *IEEE International Conference on Robotics and Automation*, pages 2112–2118, 2009.

[180] F. Oubbati, M. Richter, and G. Schöner. Autonomous robot hitting task using dynamical system approach. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4042 – 4047, 2013.

[181] M. Hersch and A. Billard. Reaching with multi-referential dynamical systems. *Autonomous Robots*, 25(1-2):71–83, 2008.

[182] S. Kim, E. Gribovskaya, and A. Billard. Learning motion dynamics to catch a moving object. *IEEE-RAS International Conference on Humanoid Robots*, pages 106–111, 2010.

[183] S. Thrun, W. Burgard, and D. Fox. Probabilistic robotics. 2005.

[184] L. Zhang, R. Zapata, and P. Lépinay. *Self-adaptive Monte Carlo localization for mobile robots using range sensors*. 2009.

[185] S. Engelson and D. McDermott. Error correction in mobile robot map learning. *Int. Conference on Robotics and Automation (ICRA)*, pages 2555 – 2560, 1992.

[186] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99 – 141, 2001.

[187] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067 — 1080, 2007.

[188] Eiris infrared localization system. system manual - raanana, israel: Elpas electro-optic systems ltd. @ONLINE [accessed 22 april 2014]. URL `http://www.elpas.com/Products/Eiris-Software.aspx`.

[189] B. Ottersten, M. Viberg, P. Stoica, and A. Nehorai. Exact and large sample ml techniques for parameter estimation and detection in array processing. *Springer Berlin Heidelberg*, pages 99–Ű151, 1993.

[190] M. Philipose, K. P Fishkin, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with rfid technology. *IEEE International Conference on Robotics and Automation*, pages 1015 – 1020, 2004.

[191] S. Schneegans, P. Vorst, and A. Zell. Using rfid snapshots for mobile robot self-localization. *3rd European Conference on Mobile Robots (ECMR 2007)*, pages 241 – 246, 2007.

[192] A. Milella, D. Di Paola, G. Cicirelli, and T. DŠOrazio. Rfid tag bearing estimation for mobile robot localization. *International Conference on Advanced Robotics, ICAR 2009*, pages 1 – 6, 2009.

[193] S. Krishnan, P. Sharma, Z. Guoping, and Ong Hwee Woon. A uwb based localization system for indoor robot navigation. *IEEE International Conference on Ultra-Wideband, ICUWB 2007*, pages 77 – 82, 2007.

[194] J. Gonzalez, J. Blanco, C. Galindo, A. Ortiz de Galisteo, J. Fernandez-Madrigal, F. Moreno, and J. Martinez. Mobile robot localization based onultra-wide-band ranging: A particle filter approach. *Robotics and Autonomous Systems*, 57(5):496 – 507, 2009.

[195] A. Howard, S. Siddiqi, and G. S. Sukhatme. An experimental study of localization using wireless ethernet. *4th International Conference on Field and Service Robotics*, 2003.

[196] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach. Robotics-based location sensing using wireless ethernet. *Wireless Networks*, (11):189 – 204, 2005.

[197] C. Röhrig and F. Künemund. Mobile robot localization using wlan signal strengths. *International Journal of Computing, SPECAL ISSUE: Intelligent Data Acquisition and Advanced Computing Systems*, 2(7):73 – 83, 2007.

[198] A. N. Raghavan, H. Ananthapadmanaban, M. S. Sivamurugan, and B. Ravindran. Accurate mobile robot localization in indoor environments using bluetooth. *IEEE International Conference on Robotics and Automation, ICRA 2010*, pages 4391 – 4396, 2010.

[199] K. Lee, S. Kim, H. Park, and M. Lee. Pseudolite ultrasonic system (pus) and gyro integrated system using kalman filter. *International Conference on Control, Automation and Systems*, pages 1125–1128, 2010.

[200] S. Kim, K. Yoon, D. Lee, and M. Lee. The localization of a mobile robot using a pseudolite ultrasonic system and a dead reckoning integrated system. *International Journal of Control, Automation, and Systems*, 9(2):339–347, 2011.

[201] Y. Sakamoto, T. Ebinuma, K. Fujii, and S. Sugano. Gps-compatible indoor-positioning methods for indoor-outdoor seamless robot navigation. *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 95 – 100, 2012.

[202] D. Fontanelli, A. Danesi, F. A. Belo, P. Salaris, and A. Bicchi. Visual servoing in the large. *The International Journal of Robotics Research*, 28(6):802 –814, 2009.

[203] T. Goedeme, M. Nuttin, T. Tuytelaars, and L. Van Gool. Omnidirectional vision based topological navigation. *International Journal of Computer Vision*, 74(3):219 –236, 2007.

[204] A. Remazeilles and F. Chaumette. Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4):345 –356, 2007.

[205] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237 –267, 2002.

[206] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263 – 296, 2008.

[207] Stargazer localization system from hagisonic co, ltd. @ONLINE [accessed 22 april 2014]. URL `www.hagisonic.com/`.

[208] S. Se, Lowe D., and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364 – 375, 2005.

[209] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. *IEEE International Conference on Robotics and Automation*, 2: 1322 – 1328, 1999.

[210] E. Ivanjko, A. Kitanov, and I. Petrovic. Model based kalman filter mobile robot self-localization. *Robot Localization and Map Building*, pages 59 – 89, 2010.

[211] J. Rowekamper, C. Sprunk, G. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard. On the position accuracy of mobile robot localization based on particle filters

combined with scan matching. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3158 – 3164, 2012.

[212] D. Lee and W. Chung. Discrete-status-based localization for indoor service robots. *IEEE Transactions on Industril Electronics*, 53(5):1737–1746, 2006.

[213] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages 35 – 45, 1960.

[214] I. Cox and J. Leonard. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66.

[215] D. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. *IEEE International Conference on Robotics and Automation*, 1.

[216] N.J. Nilsson. A mobile automaton: An application of ai techniques. *Proc. of the First International Joint Conference on Artificial Intelligence (Morgan Kaufmann Publishers, San Francisco)*, pages 509–520, 1969.

[217] A.R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, 1986.

[218] J. H. Connell. Sss: A hybrid architecture applied to robot navigation. *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2719–2724, 1992.

[219] I. Horswill. Polly: A vision-based artificial agent. *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1993.

[220] R. C. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.

[221] Robert James Firby. Adaptive execution in complex dynamic worlds. Technical report, Ph.D Thesis (Yale Univ., New Haven), 1989.

[222] R. P. Bonasso. Integrating reaction plans and layered competences through synchronous control. In *Proceedings of International Joint Conferences on Artificial Intelligence*, 1991.

[223] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1992.

[224] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17(4):315–337, 1998.

[225] I. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I. Shu, and D. Apfelbaum. Claraty: Challenges and steps toward reusable robotic software. *International Journal of Advanced Robotic Systems.*, 3(1):23–30, 2006.

[226] B. Sellner, F.W Heger, L.M. Hiatt, R. Simmons, and S. Singh. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proc. of the IEEE- Special Issue on Multi-Robot Systems*, 94(7):1425–1444, 2006.

[227] J. T. Feddema, R. D. Robinett, and B. J. Driessen. Designing stable finite state machine behaviours using phase plane analysis and variable structure control. volume 2, pages 1134 – 1141, 1998.

[228] A. Goswami, B. Espiau, and A. Keramane. Limit cycles and their stability in a passive bipedal gait. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 246–251, 1996.

[229] A. Goswami, B. Thuilot, and B. Espiau. A study of the passive gait of a compass-like biped robot symmetry and chaos. In *The International Journal of Robotics Research*, volume 17, pages 1282–1301, 1998.

[230] Y. Hurmuzlu, F. Genot, and B. Brogliato. Modeling, stability and control of biped robots - a general framework. In *IEEE International Conference on Robotics and Automation*, volume 40, pages 1647–1664, 2004.

[231] P. M. Silva and J. A. T. Machado. Towards force interaction control of biped walking robots. volume 3, pages 2568 – 2573, 2004.

[232] T. Wang and C. Chevallereau. Stability analysis and time-varying walking control for an under-actuated planar biped robot. In *Robotics and Autonomous Systems*, volume 59, pages 444–456, 2011.

[233] Y. Aoustin, C. Chevallereau, and A. FormalŠsky. Numerical and experimental study of the virtual quadrupedal walking robot-semiquad. *Multibody System Dynamics*, 16(1):1 – 20, 2006.

[234] R. Grupen and J. Coelho. Acquiring state from control dynamics to learn grasping policies for robot hands. *Advanced Robotics*, 16(5):427 – 443, 2002.

[235] D. Koditschek and M. Buhler. Analysis of a simplified hopping robot. In *The International Journal of Robotics Research*, volume 10, pages 587–605, 1991.

[236] W. Schwind and D. Koditschek. Control of forward velocity for a simplified planar hopping robot. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 691–696, 1995.

[237] Z. Lu, S. Ma, B. Li, and Y. Wang. Design of a snake-like robot controller with cyclic inhibitory cpg model. pages 35 – 40, 2005.

[238] J. Ryu, N. Chong, B. You, and H. Christensen. Locomotion of snake-like robots using adaptive neural oscillators. volume 3, pages 1 – 10, 2010.

[239] O. Tchernichovski and I. Golani. A phase plane representation of rat exploratory behavior. *Journal of neuroscience methods*, 62(1):21 – 27, 1995.

[240] J. S. Il'Yashenko. Global analysis of the phase portrait for the kuramoto-sivashinsky equation. *Journal of Dynamics and Differential Equations*, 4(4):585 – 615, 1992.

[241] A. M. Lyapunov. The general problem of stability of motion. *International Journal of Contro*, 55(3):531 – 534, 1992.

[242] A. Behal, W. Dixon, D. M. Dawson, and B. Xian. *Lyapunov-based control of robotic systems*, volume 36. 2009.

[243] M. Branicky. Stability of switched and hybrid systems. volume 4, pages 3498 – 3503, 1994.

[244] Y. Ma, J. Kosecka, and S. Sastry. Vision guided navigation for a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, 15(3):521 – 536, 1999.

[245] E. Freiret, T. Bastos-Filho, M. Sarcinelli-Filho, and R. Carelli. A control architecture for mobile robots using fusion of the output of distinct controllers. pages 142 – 147, 2002.

[246] R. Carelli and E. Freire. Corridor navigation and wall-following stable control for sonar-based mobile robots. *Robotics and Autonomous Systems*, 45(3):235 – 247, 2003.

[247] J. Toibero, R. Carelliz, and B. Kuchen. Switching control of mobile robots for autonomous navigation in unknown environments. pages 1974 – 1979, 2007.

[248] A. Benzerrouk, L. Adouane, and P. Martinet. Lyapunov global stability for a reactive mobile robot navigation in presence of obstacles. *ICRA 2010 International Workshop on Robotics and Intelligent Transportation System*, 2010.

[249] F. Cuesta and A. Ollero. Fuzzy control of reactive navigation with stability analysis based on conicity and lyapunov theory. *Control engineering practice*, 12(5):625 – 638, 2004.

[250] M. Ertugrul and O. Kaynak. Neuro sliding mode control of robotic manipulators. *Mechatronics*, 10(1):239 – 532636, 2000.

[251] L. Beji, M. ElKamel, and A. Abichou. A strategy for multi-robot navigation. in decision and control and european control conference. pages 4214 – 4219, 2011.

[252] J. Yang, S. J. Chung, S. Hutchinson, D. Johnson, and M. Kise. Vision-based localization and mapping for an autonomous mower. pages 3655 — 3662, 2013.

[253] T. Das and I. Kar. Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(3):501 – 510, 2006.

[254] S. Yang, A. Zhu, G. Yuan, and M. Meng. A bioinspired neurodynamics-based approach to tracking control of mobile robots. *IEEE Transactions on Industrial Electronics*, 59(8):3211 – 3220, 2012.

[255] S. Yang and M. Meng. An efficient neural network approach to dynamic robot motion planning. *Neural Networks*, 13(2):143 – 148, 2000.

[256] H. Berti, A. Sappa, and O. Agamennoni. Autonomous robot navigation with a global and asymptotic convergence. In *International Conference on Robotics and Automation (ICRA)*, pages 2712–2717, 2007.

[257] I. Sandberg. A frequency-domain condition for the stability of feedback systems containing a single time-varying nonlinear element. *Bell System Technical Journal*, 43(4):1601 – 1608, 1964.

[258] G. Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE Transactions on Automatic Control*, 11(2):228 – 238, 1966.

[259] E. Sontag. Input to state stability: Basic concepts and results. *Nonlinear and optimal control theory, Springer Berlin Heidelberg*, pages 163 – 220, 2008.

[260] E. Fridman and U. Shaked. Input - output approach to stability and l2-gain analysis of systems with time-varying delays. *Intelligent Automation & Soft Computing*, 55(12):1041 – 1053, 2006.

[261] N. Sarkar, X. Yun, and V. Kumar. Control of mechanical systems with rolling constraints application to dynamic control of mobile robots. *The International Journal of Robotics Research*, 13(1):55 – 69, 1994.

[262] Y. Yang, C. Zhou, and J. Du. Adaptive robust fuzzy tracking control for pole balancing robots using small gain design. *Intelligent Automation & Soft Computing*, 11(2):97 – 109, 2005.

[263] J. Jouffroy and J. Slotine. Methodological remarks on contraction theory. *In 43rd IEEE Conference on Decision and Control*, 3:2537 – 2543, 2004.

[264] W. Wang and J. Slotine. On partial contraction analysis for coupled nonlinear oscillators. *Biological cybernetics*, 92(1):38 – 53, 2005.

[265] D. Angeli. A lyapunov approach to incremental stability properties. *IEEE Transanctions on Automatic Control*, 47(3):410 – 421, 2002.

[266] V. Fromion, G. Scorletti, and G. Ferreres. Nonlinear performance of a pi controlled missile: an explanation. *International Journal of Robust and Nonlinear Control*, 9:485 – 518, 1999.

[267] J. Jouffroy and T. Fossen. A tutorial on incremental stability analysis using contraction theory. *Modeling, Identification and Control*, 31(3):93 – 106, 2010.

[268] J. Jouffroy. A simple extension of contraction theory to study incremental stability properties. In *European Control Conference*, 2003.

[269] J. Slotine and W. Lohmiller. Modularity, evolution, and the binding problem: A view from stability theory. *Neural networks*, 14(2):137 – 145, 2001.

[270] B. Perk and J. Slotine. Motion primitives for robotic flight control. *arXiv preprint cs/0609140*, 2006.

[271] T. Kiant, S. Hungsun, and P. Chai. Uav flight path control using contraction-based back-stepping control. *Open Journal of Applied Sciences*, 65 (3):65 – 70, 2013.

[272] T. D. Nguyen and O. Egeland. Output tracking control of a flexible robot arm. *44th IEEE Conference on Decision and Control, CDC-ECC'05*, pages 5269 – 5274, 2005.

[273] S. Chung and J. Slotine. Cooperative robot control and concurrent synchronization of lagrangian systems. *IEEE Transactions on Robotics*, 25(3): 686 – 700, 2009.

[274] J. Sequeira, C. Santos, and J. Silva. Dynamical systems in robot control architectures: A building block perspective. *Proceedings of the 12th International Conference on Control, Automation, Robotics and Vision, ICARCV*, 2012.

[275] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328 – 373, 2013.

[276] Y. Zhao and J. Slotine. Discrete nonlinear observers for inertial navigation. *Systems & control letters*, 54(9):887 – 898, 2005.

[277] G. Torsetnes. Nonlinear control and observer design for dynamic positioning using contraction theory, 2004.

[278] A. Park, A. Mukovskiy, L. Omlor, and M. Giese. Synthesis of character behaviour by dynamic interaction of synergies learned from motion capture data. pages 9 — 16, 2008.

[279] I. Chang and S. Chung. Bio-inspired adaptive cooperative control of heterogeneous robotic networks. pages 9 — 16, 2009.

[280] S. Pring and C. Budd. The dynamics of regularized discontinuous maps with applications to impacting systems. *SIAM Journal on Applied Dynamical Systems*, 9(1):188 – 219, 2010.

[281] J. Awrejcewicza, M. Fekanb, and P. Olejnika. On continuous approximation of discontinuous systems. *Journal of Nonlinear Analysis*, 62:1317–1331, 2005.

[282] M. F. Danca and S. Codreanu. On a possible approximation of discontinuous dynamical systems. *Journal of Chaos, Solitons and Fractals*, 13:681–691, 2002.

[283] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 723 – 730, 2011.

[284] Y. Guo and Q. Zhihua. Coverage control for a mobile robot patrolling a dynamic and uncertain environment. *5th IEEE World Congress on Intelligent Control and Automation*, 6:4899 – 4903, 2004.

[285] T. Bretl and S. Hutchinson. Robust coverage by a mobile robot of a planar workspace. *IEEE International Conference on Robotics and Automation (ICRA)*, 6:4567 – 4572, 2013.

[286] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249 – 265, 1987.

[287] A. Elfes. Occupancy grids: a probabilistic framework for robot perception and navigation. *Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburhg, PA*, 1989.

[288] H. Hu and M. Brady. Dynamic global path planning with uncertainty for mobile robots in manufacturing. *IEEE Transactions on Robotics and Automation*, 13(5), 1997.

[289] E. Bicho, P. Mallet, and G. Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, (210):424–447, 2000.

[290] E. Large. Scaling the dynamical systems approach to path planning. *IEEE International Symposium on Industrial Electronics, Guimaraes, Portugal*, pages 21–26, 1997.

[291] A. Steinhage and G. Schöner. Dynamical systems for the behavioral organization of autonomous robot navigation. *In McKee G T Schenker PS, editor, Sensor Fusion and Decentralized Control in Robotic Systems: Proceedings of Spie-Intelligent Systems Manufactors, Boston*, pages 169Ű–180, 1998.

[292] B. Hasselblatt and A. Katok. *A First Course in Dynamics*. Cambridge University Press, 2003.

[293] E. Kreyszig. *Introductory functional analysis with applications*, volume 81. New York: wiley, 1989.

[294] H. Lütkepohl. *Handbook of Matrices*. John Wiley & Sons, 1996.

[295] W. Lohmiller and J. Slotine. Control system design for mechanical systems using contraction theory. *IEEE Transactions on Automatic Control*, 45(5):984 – 989, 2000.

[296] K. Rifai and J. Slotine. Compositional contraction analysis of resetting hybrid systems. *IEEE Transactions on Automatic Control*, 51(9):1536–1541, 2006.

[297] A. Ozkil, S. Dawids, Z. Fan, and T. Sorensen. Design of a robotic automation system for transportation of goods in hospitals. *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 392–397, 2007.

[298] K. Fitzpatrick, M. Brewer, and S. Turner. Another look at pedestrian walking speed. *Transportation Research Record: Journal of the Transportation Research Board*, 1982(1):21 – 29, 2006.

[299] S. Hoogendoornn and P. Bovy. Simulation of pedestrian flows by optimal control and differential games. *Optim. Control Appl. Meth*, 24:153 – 172, 2003.

[300] O. Michel. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[301] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *Proceedings of IWAR 99*, pages 85 – 94, 1999.

[302] S. Zhang, L. Xie, and M. D. Adams. Entropy based feature selection scheme for real time simultaneous localization and map building. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1175–1180, 2005.

[303] M. Beinhofer, J. Muller, and W. Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robotics and Autonomous Systems*, 2012.

[304] J. Salas and J. Gordillo. Placing artificial visual landmarks in a mobile robot workspace. *Proc. of the Ibero-American Conf. on Artificial Intelligence*, 1484: 274 – 282, 1998.

[305] P. Sala, R. Sim, and A. Shokoufandeh. Placing artificial visual landmarks in a mobile robot workspace. *IEEE Transactions on Robotics*, 22(2):334 – 349, 2006.

[306] D. Meyer-Delius, M. Beinhofer, A. Kleiner, and W. Burgard. Using artifical landmarks to reduce the ambiguity in the environment of a mobile robot. *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, pages 5173 –5178, 2011.

[307] L. Jetto, S. Longhi, and G. Venturini. Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 15(2):219–229, 1999.

[308] S. Han, Q. Zhang, and H. Noh. Kalman filtering of dgps positions for a parallel tracking application. *Transactions-American Society of Agricultural Engineers*, 45(3):553–560, 2002.

[309] A. Paul and E. Wan. Dual kalman filters for autonomous terrain aided navigation in unknown environments. *IEEE International Joint Conference on Neural Networks*, 5:2784–2789, 2005.

[310] J. Sasiadek and P. Hartana. Sensor data fusion using kalman filter. *3rd International Conference Information Fusion*, 2:19–25, 2000.

[311] P. Escamilla-Ambrosio and N. Mort. Multisensor data fusion architecture based on adaptive kalman filters and fuzzy logic performance assessment. *Proceedings of the Fifth International Conference on Information Fusion*, 2: 1542–1549, 2002.

[312] V. Subramanian, T. Burks, and W. Dixon. Sensor fusion using fuzzy logic enhanced kalman filter for autonomous vehicle guidance in citrus groves. *Transactions of the ASAE*, 52:1411–1422, 2009.

[313] B. Graf. Dependability of mobile robots in direct interaction with humans. *Advances in Human-Robot Interaction, Springer Berlin Heidelberg*, pages 223Ű–239, 2005.

[314] A. Llarena, J. Savage, A. Kuri, and B. Escalante-Ramirez. Odometry-based viterbi localization with artificial neural networks and laser range finders for mobile robots. *Journal of Intelligent & Robotic Systems*, 66(1-2):75 – 109, 2012.