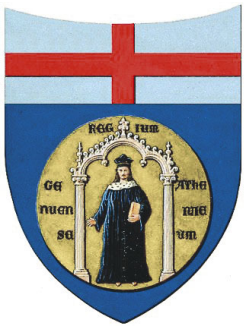
Planning and Control of Autonomous Marine Systems



Enrico Simetti

Department of Communication, Computer and System Science

University of Genova

A thesis submitted for the degree of

Dottorato di Ricerca in Ingegneria Matematica e Simulazione

XXIV Ciclo

April, 17th 2012

To my family

Acknowledgements

I would like to thank Prof. Casalino, alias *Pino*, for his guidance during all my Ph.D. studies. His unquenchable thirst for knowledge made him study all the challenges and problems we have encountered to their very essence. This lead him to write long (and sometimes heavy :-)) technical reports that have been a fundamental source for most of the work presented here.

My second thank goes to Alessio Turetta. Other than being co-author of most of the work here presented, I have seen him as my second tutor, and that gave me the possibility of having two reliable but sometimes different point of views on the different problems we have tackled. Moreover, we had the luck to share a lot of journeys for conferences all around the world and we had always a good time.

Then, I would like to thank my friends and co-workers Alessandro Sperindè (that's the correct accent, right?) and Sandro Torelli (see, I was tempted but didn't write Lorenzo Torello), with whom I shared the pleasure of writing the thousand lines of code that make up the *Workframe* and many hours of compilation during these 3 years; as if that was not sufficient, we also shared, repeatedly, the weight of the TRIDENT's arm each time we had to move it :-).

My thanks go also to Enrica Zereik, with whom I shared the impossible crusade (more like Don Quixote tilting against the windmills) to make Pino learn LaTeX and abandon MS Word for the scientific publications. Unfortunately, I can't say we have been very successful :-(.

Finally, I would like to thank all the undergraduate students that I had the pleasure to tutor and all those that followed (without sleeping) all the Automatic Control lessons that I have done during these years.

Abstract

The main objective of the research work presented in this thesis is the developing of new algorithms for the planning and control of unmanned marine systems. While unmanned surface vehicles (USVs) and autonomous underwater vehicles (AUVs) share the same medium and are often employed together to collaboratively accomplish complex goals, the surface and the underwater are quite distinct worlds with different critical issues.

USVs, depending on the context, can operate in waterways where other manned vehicles are navigating, thus the main problem is to guarantee that they are fully able to move and avoid both static and moving obstacles. The lack of proven and reliable methodologies and sensors for the obstacle avoidance is the major hindrance to the extensive use of USVs in civil environments.

Conversely, while the problem of obstacle avoidance is certainly important, AUVs have to face other primary issues. The localization problem is one aspect where multiple solutions have been proposed, each one with different cons and pros, and it is still an interesting and very active area of research. Long Base Line (LBL) systems can localize the AUV in a similar manner as the USV localizes itself with the use of the GPS. However, transponders must be moored to the sea floor (and then retrieved), or put inside buoys, that can possibly move due to the sea current. Other technologies include the Ultra Short Base Line (USBL), where an array of transponders is towed from the support ship, and the AUV position can be tracked. However, this only lets the support ship know where the vehicle is located. A third method to solve the localization problem is to use Inertial Navigation Systems (INS) and to surface from time to time to obtain a GPS fix and correct the estimation. This localization method greatly suffers from the bias and errors of the INS and is only used for gliders, which are vehicles that navigate for very long ranges without the need of support ships and thus where LBL or USBL systems cannot be used.

Due to these considerations, this work has been divided in two main parts, each one focusing on one class of vehicles. The first part is dedicated to surface vehicles and presents the results of an ongoing joint research efforts with Selex Sistemi Integrati. The main goal of the joint research program is to develop a Swarm Management Unit (SMU), a tool for the

supervision of a team of USVs performing patrolling and surveillance activities within civilian harbors. In particular, the focus of the research has been on devising an effective algorithm for the obstacle avoidance. On the basis of the knowledge of the position, heading and current speed of the vessels operating in the area, the planner finds the time optimal path for the vehicle in order to reach a particular destination. The algorithm is fast enough to be able to recompute the path every time an update of the aforementioned vessel information is available.

In the considered surveillance scenario, another important aspect is the interception and classification of unknown vessels. While the classification is not investigated, the interception, i.e. the problem of getting close to the position of the unknown vessel, is tackled from two perspectives: finding on-line which vehicle amongst all the available ones should head towards the unknown vessels and how to do so, and the off-line optimization of the positioning of the team of vehicles in order to maximize the performances of the team once a possible menace is detected. The on-line problem is shown to be closely related to the obstacle avoidance problem, as the vehicle can be seen as a moving goal that the USV needs to reach in a time-optimal fashion. The off-line problem is instead solved following two criteria with a strict priority order. First, a minimum interception distance between the unknown menace and an asset that has to be protected is guaranteed. The output of such optimization is the number of vehicles necessary to meet the requirements and their positioning. Then, any other vehicles is assigned to minimizing the interception time, which allows a prompt answer of the system even in cases where the unknown vessels are quite distant from the protected asset. Finally, the design of a prototype of cheap USVs is presented. This prototype will be used as a platform for testing the algorithms implemented inside the SMU.

The second part of the work is instead dedicated to the underwater world. The research is then framed within two European FP7 projects: CO^3AUVs and TRIDENT. The first is aimed at developing technologies and algorithms for the cooperation between AUVs, while the latter is instead focusing on the very challenging problem of underwater intervention and manipulation. The critical issue of underwater localization is discussed and two different algorithms are proposed due to its importance. Both of them exploit a LBL system, where the time of arrival of an acoustic signal is used to determine the distance w.r.t. the transponders, and then used

for computing the position of the AUV. However, the two algorithms differ for assumptions and requirements. The first one, called RT^2, is based on the knowledge of the speed of sound (SoS) profile along the water column, which is also assumed constant radially. As it is known, such anisotropy prevents the acoustic waves to propagate in a rectilinear way. However, within the validity of these assumptions, it is shown that the ray propagation is periodic, and a computationally effective method is presented for precisely calculating the horizontal range between the AUV and the transponders. Once the horizontal range is known, the position can be calculated with well-known triangulation schemes. Simulative results and a sensitivity analysis are presented to show the validity of the proposed approach. However, as obtaining the SoS profile can be a time-consuming procedure, and that the constancy in radius might not always be true, a second algorithm for the localization is presented, which does not require any a-priori knowledge of the SoS profile. Instead, the algorithm assumes a straight ray approximation, and instead of considering a particular value of the SoS, it finds the best value for it in order to minimize the localization error. Preliminary field tests are reported and show the effectiveness of such approach, at least for "short" ranges where the effects of ray bending are somewhat less crucial.

The problem of free floating manipulation is then tackled, and is one of the most challenging goals of the TRIDENT project. The goal is to control the vehicle and the arm at the same time, exploiting the high degrees of freedom, in order to reach a target position whilst fulfilling different secondary objectives. For example, the arm must stay away from its joint limits, and keep a good dexterity. The vehicle must keep the object within the visibility of the manipulation camera, stay horizontal in order to avoid excessive consumption, keep itself within given distance boundaries and so on. An effective way to satisfy all these constraints is thus presented, with the aim of surpassing the sequential schemes. The proposed algorithm is based on a dynamic programming (DP) approach, which is made run between the chain composed by the arm and the vehicle. Simulative results show that, even within the multi-sample rate nature of the system, and with noise, the system manages to accomplish the given tasks.

Finally, a software framework for real time systems is presented. The primary objective of the framework is to simplify the process of writing

the control software. This is achieved by offering higher level interfaces and libraries to the programmer, instead of letting him directly interact with the underlying Real Time Operating System (RTOS). Moreover, the framework promotes the reuse of software, as many common problems such as data logging, scheduling of tasks, creation and request of resources, etc. have already been solved and are ready to be used. This dramatically reduces the time needed to develop the control software, allowing the programmer to focus on its particular algorithm rather than wasting time with other problematics. The framework has also been designed to be portable across three different RTOS (RTAI, QNX Neutrino and RTEMS).

Index

1 Introduction **1**

1.1 On-going National and European Projects 5

1.1.1 EU FP7: TRIDENT 6

1.1.2 EU FP7: CO^{3}AUVs 8

1.1.3 Selex-SI: Swarm Management Unit 10

2 Unmanned Surface Vehicles **12**

2.1 Main Challenges and Applications of USVs 14

2.1.1 Main USVs Applications 14

2.1.2 Main Challenges and Critical Issues 15

2.1.3 Proposed Algorithms 16

2.2 Real Time Path Planning and Avoidance of Moving Vessels 19

2.2.1 Avoidance of Static Obstacles 20

2.2.2 Real Time Optimal Avoidance of Moving Obstacles 22

2.3 Interception of Detected Menaces 33

2.4 Off-line Optimization of Fleet Positioning 36

2.4.1 Preventing Menaces from Getting Too Close to the Asset . 38

2.4.2 Minimizing the Maximum Interception Time 40

2.4.3 Solution of the Off-line Optimizations 41

2.4.4 Simulative Results 41

2.4.5 Multi Asset Problem 51

2.5 Prototype Development 53

2.5.1 The General Architecture 53

2.5.2 First Prototype Design and Sensors 54

2.5.3 Sensors Equations 55

2.5.4 Preliminary Field Tests 57

2.5.5 Position Estimation Algorithms 59
2.5.6 Field Tests of the Localization Algorithms 65
2.5.7 Final Prototype Design 69

3 Autonomous Underwater Vehicles 73
3.1 Main Challenges and Applications of AUVs 73
3.1.1 Main AUVs Applications 73
3.1.2 Main Challenges and Critical Issues 75
3.1.3 Proposed Algorithms 78
3.2 RT^2: Real Time Ray Tracing for Underwater Range Evaluation . 80
3.2.1 Brief Review Sound-Ray Theory 81
3.2.2 Basic Algorithms 84
3.2.3 Simulative Results 89
3.2.4 Application to Distributed Localization 100
3.3 Acoustic-based Localization Technique for AUVs in the Absence of Information on the Speed of Sound Profile 102
3.3.1 2D Localization Problem 104
3.3.2 Introductory Problem: 2 Transponders 104
3.3.3 3 Transponders 108
3.3.4 4 Transponders 112
3.3.5 3D Localization Problem 118
3.3.6 Algorithm Validation 119

4 Intervention Autonomous Underwater Vehicles 121
4.1 Main Challenges and Applications of I-AUVs 122
4.1.1 Main I-AUVs Applications 122
4.1.2 Main Challenges and Critical Issues 122
4.1.3 Proposed Algorithms 123
4.2 A Dynamic Programming Approach to the Free Floating Manipulation Problem 125
4.2.1 System Organization 126
4.2.2 Control Objectives 128
4.2.3 Introducing System Agility 136
4.2.4 Control Architecture 138
4.2.5 Geometric-Kinematic Control Layer 139

4.2.6 Simulative Results 156

5 Real Time Software Architecture for the Control of USVs and AUVs 167

5.1 Kernel Abstraction Layer 170
5.2 Workframe Library 171
5.2.1 System Manager 173
5.2.2 Scheduler 174
5.2.3 Debug Console 174
5.2.4 Workframe Console 174
5.2.5 Watchdog 175
5.3 Network Communication Library 175
5.4 BlackBoard System 175
5.5 Mathematical Library 177
5.6 Control Library 178

6 Conclusions 180

6.1 On-going Activities @ GRAAL 183
6.2 Future Research Activities 184

References 185

List of Figures

1.1 Photo of Sentry AUV during recharge between dive missions at the Deepwater Horizon spill site. The TETHYS payload is hidden within Sentry's lateral port area, between the forward and aft dive planes (photo: D. Yoerger, WHOI) 2
1.2 The SESAMO catamaran on the Malippo stern deck (Terra Nova Bay, February 2004) 3
1.3 Different phases of the Trident's survey and manipulation 7

2.1 (**a**) red points are candidate vertexes; (**b**) resulting visibility graph 21
2.2 Example of path planning: yellow represent the current segment, green segments are the next segments already planned 21
2.3 Schematization of the intercept problem 22
2.4 Simulation of the algorithm: (**a**) the original avoidance problem (**b**) the intercept points with all the vertexes are created 25
2.5 Simulation of the algorithm: (**a**) some of the nodes built are removed for collision (**b**) the most promising node is selected 26
2.6 Simulation of the algorithm: (**a**) vehicle is at the box's vertex ($t = \bar{t}$) and the algorithm is executed once again (**b**) two new intercept points are obtained 27
2.7 Simulation of the algorithm: (**a**) goal can be reached directly (**b**) example of solution obtained 28
2.8 Simulation of the algorithm: (**a**) waypoints previously created for vehicle A only need to be checked for collision with vehicle B (**b**) intercept points with vehicle B are also added 30

2.9 Simulation of the algorithm: (**a**) collision checking is performed for the new waypoints (**b**) the final set of waypoints, some created from vehicle A's vertexes, and some from vehicle B's ones. The algorithm is then repeated as in the single vehicle case 31
2.10 Mutual avoidance: (**a**) potential collision situation, yellow path is the nominal path of the highest priority USV (**b**) collision is avoided without having the highest priority USV to change its path 32
2.11 Simulation of the algorithm: (**a**) initial trajectory (**b**) modified trajectory (**c**) vehicle is at the box's vertex ($t = \bar{t}$) (**d**) final position 32
2.12 The interception problem: **a**) without obstacles (**b**) with an obstacle 34
2.13 Schema of the intercept problem 36
2.14 Schema of the intercept problem with a required minimum distance of interception 38
2.15 Different optimized results of the asset protection problem obtained by varying d_{th}: (**a**) $d_{th} = 86m$ (**b**) $d_{th} = 96m$ (**c**) $d_{th} = 110m$ (**d**) $d_{th} = 120m$ 42
2.16 (**a**) USV with speed of 5m/s (**b**) speed 15m/s (**c**) with $W(P_m) = 1$ 42
2.17 (**a**) with 4 USVs (**b**) with 6 USVs 43
2.18 (**a**) 6 USVs with asset in $(0, 500)$ (**b**) asset in $(0, 1000)$ (**c**) asset in $(1000, 1000)$ (**d**) 9 USVs with asset in $(0, 0)$ 44
2.19 Red dot: asset; Colored squared: USV; Smaller circle: d_{th}; Bigger circle: r 45
2.20 Different computed paths for the intercepting vehicles. (a) as the menace is detected just at the minimum distance r, the blue vehicle intercepts it before the minimum required distance d_{th} (b) menace is detected far away (top left corner) and thus multiple vehicles can intercept it, and the best one is the yellow one 46
2.21 Comparison by changing the detection distance r (a) $r = 400$, $d_{worst} = 214$, $t_{worst} = 22.93$, (b) $r = 750$, $d_{worst} = 373$, $t_{worst} = 22.62$ 47
2.22 Comparison by changing the minimum required interception distance d_{th} (a) $d_{th} = 178$, $d_{worst} = 374$, $t_{worst} = 20.64$, (b) $d_{th} = 400$, $d_{worst} = 460$, $t_{worst} = 20$ 48

2.23 Comparison by changing the position of the asset (a) if the asset is inside the inner harbor area only one vehicle is necessary for the first criterion (b) when the asset is near the open sea, three vehicles are necessary to guarantee the same distance d_{th} 49
2.24 Comparison by changing the number of USVs (a) 5 USVs (b) 4 USVs 50
2.25 Comparison by changing the possible target (a) northern asset (b) southern asset 52
2.26 The Architecture of the Prototype. 54
2.27 Green: GPS position; Red: Kalman estimated position. 58
2.28 Estimates from the compass. Blue: Pitch; green: Roll. 59
2.29 Yaw measured by the compass while performing an approximated square trajectory. 60
2.30 Yaw measured by the GPS module while performing an approximated square trajectory. 60
2.31 Accelerations measured while the vehicle was not moving. 61
2.32 Estimates from the compass. Blue: Pitch; green: Roll. 61
2.33 Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing 66
2.34 Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing 67
2.35 Blue: Kalman filter yaw estimate. Red: compass yaw value. Green: GPS heading 67
2.36 Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing 68
2.37 Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing 68
2.38 Blue: Kalman filter roll angle estimate. Red: Kalman filter pitch angle estimate 69
2.39 Schematic of the dsPIC and the connectors of the control board . 70
2.40 Schematic of the voltage regulation 71
2.41 Schematic of the IDG 500 gyroscope sensor 71
2.42 Schematic of the HMC6343 compass sensor 71
2.43 Schematic of the ADXRS613 gyroscope sensor 72
2.44 Schematic of the ADXL345 accelerometer sensor 72

3.1 (a) sound velocity profile as a function of depth in the water column; (b) cylindrical coordinate system, sample ray path and corresponding angles accordingly with Snell's law 82
3.2 Sound speed profile as a function of depth and different lines $1/k$ corresponding to different couples of ray solutions. The intersection of any $1/k$ line with the sound speed profile determines the upper and lower depths within which the rays will be confined. (a) the rays are confined in an "acoustic channel" (b) the ray reaches the bottom, but is refracted at z_u (c) the ray moves freely between the top of the surface and the bottom. 83
3.3 (a) sound speed profile as a function of depth, with indication of upper extremum z_u for a given k; (b) the couple of ray paths corresponding to the given k; note that ray evolution can be expressed either as a function of depth and time (t) or of depth and range (r). 85
3.4 (a) sound speed profile as a function of depth, with indication of the chosen depths z_a, z_b and of the chosen k. (b) the couple of ray paths corresponding to the given k showing the 4 intersections of the ray with the depth z_b within one period, either in time or in range. $\Delta_{i,j}$ represents the travel time between points i, j, while $\rho_{i,j}$ represents the horizontal travel distance between the points i, j. . 85
3.5 Sound speed profile as a function of depth and corresponding integral as defined in (3.12) either in time or horizontal range 87
3.6 Sound speed profile used in Test 1.1 and Test 2.1. The profile used in the simulation is exactly known by the RT^2 algorithm. 90
3.7 Test 1.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 91
3.8 Test 2.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 92
3.9 Sound speed profile used in Test 3.1. The profile used in the simulation is exactly known by the RT^2 algorithm. 92
3.10 Test 3.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 93
3.11 Sound speed profile used in Test 1.2 and Test 2.2. The profile used by the RT^2 algorithm and the straight ray approximation is the result of a sampling of the true one. 95

3.12 Test 1.2: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 96
3.13 Test 2.2: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 96
3.14 Sound speed profile used in Test 1.3 and Test 2.3. The profile used by the RT^2 algorithm and the straight ray approximation differs from the true one with a random noise of $\pm 7 m/s$. 97
3.15 Test 1.3: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 98
3.16 Test 2.3: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation. 98
3.17 Conceptual configuration of the team localization scheme, in which localization is obtained with respect to neighboring vehicles and ultimately to the GPS linked surface vehicles 101
3.18 Different considered 2D scenarios (cyan dots are the transponders; the red square is the AUV) 105
3.19 Different positions of C_{21} (red dots) for different valus of α_{21}) . . . 107
3.20 An exemplificative scenario with $t_1 < t_2$ ($\alpha_{21} < 0$) 108
3.21 An exemplificative scenario of the 3-transponders problem with $t_1 < t_2 < t_3$ 110
3.22 Ideal scenario leading to 4 coinciding candidate solutions after the 3-transponders problem phase 115
3.23 An exemplificative scenario leading to 4 different candidate solutions after the 3-transponders problem phase 116
3.24 The employed eFOLAGA AUV 119
3.25 Localization data collected during an underwater navigation phase; the harbour scene (top) - detail of the AUV path (bottom) 120

4.1 Overall structure of the I-AUV used in the TRIDENT project . . 127
4.2 The camera centering set control objective 129
4.3 The camera distance set control objective 130
4.4 The camera height set control objective 131
4.5 The modified camera height set control objective 132
4.6 The joint limits set control objective 133
4.7 The manipulability set control objective 133

4.8 The vehicle's horizontal attitude precision control objective 134
4.9 The end effector approach precision control objective 135
4.10 The control architecture subdivided in three layers. Data symbols enclosed within a square just correspond to those acquired with rates lower than 100 Hz 138
4.11 Continuity of the evolution of the regularized kernel space when moving from $\alpha(\sigma) = 0$ (left) to $\alpha(\sigma) = \alpha^*$ (right) 143
4.12 The DP algorithm is run in every sub periods for the arm, while only in the first one for the vehicle 156
4.13 Test 1: snapshots of the movement of the I-AUV 157
4.14 Test 1: time behavior of the α functions of the set rate tasks . . . 158
4.15 Test 1: time behavior of the quantities related to the horizontal attitude task 158
4.16 Test 1: the generated $\dot{q}$ for the arm 159
4.17 Test 1: the generated v for the vehicle 159
4.18 Test 2: snapshots of the movement of the I-AUV 160
4.19 Test 2: time behavior of the α functions of the set rate tasks . . . 161
4.20 Test 2: time behavior of the α functions of the joint limits set rate task 161
4.21 Test 2: time behavior of the quantities related to the camera height task 162
4.22 Test 2: time behavior of the quantities related to the camera distance task 162
4.23 Test 2: time behavior of the quantities related to the horizontal attitude task 163
4.24 Test 2: time behavior of the quantities related to the camera centering task 163
4.25 Test 3: snapshots of the movement of the I-AUV 164
4.26 Test 3: time behavior of the α functions of the set rate tasks . . . 165
4.27 Test 3: evolution of the position of the end effector expressed in the inertial frame $\langle 0 \rangle$ 165
4.28 Test 3: evolution of the position of the vehicle expressed in the inertial frame $\langle 0 \rangle$ 166

5.1 Overall schema of the *Workframe* layers 169

5.2 Example of name resolution for a mailbox 171
5.3 The *Workframe* tasks and their interaction with the user tasks. . 172
5.4 System Manager's states and transitions between them 173
5.5 Exchange of data between a remote task and a local task. 176
5.6 Example of use of the request/execution/acknowledge mechanism: 1) the user, through the remote HCI, requests the rover to go to a new position and writes it to the blackboard; 2) the remote HCI asks the framework to execute the Path Planner task; 3) the Execution Manager forwards the request to the Path Planner task. 4) the Path Planner executes and computes a new path, which is written to the blackboard; 5) the Path Planner ends its execution and informs so the Execution Manager; 6) the remote HCI is informed of the conclusion of the Path Planner execution 7) the vehicle controller continuously read from the blackboard and now finds the new path to follow. 177

List of Tables

3.1 Absolute maximum errors and mean error using both the RT^2 algorithm and a straight ray propagation approximation. 99
3.2 Relative maximum errors and mean error using both the RT^2 algorithm and a straight ray propagation approximation. 99

Nomenclature

AUV	Autonomous Underwater Vehicle
COB	Center Of Buoyancy
COG	Center Of Gravity
COTS	Commercial Off The Shelf
COLREGS	International Regulations for Preventing Collisions at Sea
DP	Dynamic Programming
DGPS	Differential Global Position System
d.o.f.	degrees of freedom
DVL	Doppler Velocity Log
HCI	Human Computer Interface
I-AUV	Intervention Autonomous Underwater Vehicle
INS	Inertial Navigation Systems
LBL	Long Base Line
MCM	Mine Counter Measure
PTD	Partial Traveled Distance
PTT	Partial Travel Time
ROV	Remotely Operated Vehicle
RTOS	Real Time Operating System
SBAS	Satellite Based Augmentation System
SoS	Speed of Sound
SMU	Swarm Management Unit
TMS	Tether Management System
ToF	Time of Flight
USBL	Ultra Short Base Line
USV	Unmanned Surface Vehicle

Chapter 1

Introduction

On April, 20th 2010 an explosion at Deepwater Horizon semi-submersible offshore oil drilling rig killed 11 crewmen and ignited a fireball visible from 56 km away. Two days later, while all attempts to extinguish the fire failed, it sank, leaving the well gushing at the sea floor and causing the largest offshore oil spill in United States history. It is estimated that approximatively 4 million barrels (636 million liters) of oil were released into the Gulf of Mexico [1].

However, that tragedy was also the occasion for the engineers at Woods Hole Oceanographic Institute (WHOI) to give a fundamental support in order to establish if there was a deep water plume and what kind of composition it had. To characterize any resultant plume stemming from the Deepwater Horizon blowout, they performed three long-range surveys from 19 to 28 June 2010 using the National Deep Submergence Facility's autonomous underwater vehicle (AUV) Sentry (Fig. 1.1). The surveys, all conducted at depths in excess of 1000 m, operated for 64 hours to cover a linear distance of 235 km. Sampling made use of an iterative approach of in situ sensing and automated data analysis to identify select petroleum hydrocarbons and any associated oxygen anomalies. During these deployments, Sentry's mass spectrometer recorded more than 3500 discrete sample measurements, simultaneously tracking 10 independent chemical parameters in real time [2].

The Deepwater Horizon mission was certainly not the first time an AUV was used for important scientific discoveries. The same engineering group at WHOI was also involved in the discovery of high temperature submarine hydrothermal vents, using ABE (Autonomous Benthic Explorer) [3]. ABE used its sensors to locate the center of a hydrothermal plume, made high-resolution maps of



Figure 1.1: Photo of Sentry AUV during recharge between dive missions at the Deepwater Horizon spill site. The TETHYS payload is hidden within Sentry's lateral port area, between the forward and aft dive planes (photo: D. Yoerger, WHOI)

the seafloor beneath the plume center and took photographs in and around any new vent site to characterize its geologic setting and reveal the nature of any chemosynthetic ecosystems it may host. Such approach permitted to achieve a precision of the localization of venting site within 5 meters, compared to previous approaches that used a towed CTD (Conductivity, Temperature, Depth sensor) that guaranteed only a few kilometers of precision.

The same squad was also involved in the discovery of different "asphalt volcanoes", which are large underwater mounds, consisting of tar deposits with morphologies similar to volcanic lava flows, originated by significant oil seepage [4]. These examples alone justify the recent trend of growth of the AUV development, as they are a key, enabling tool for scientific breakthroughs in the oceans.

Conversely, the development of Unmanned Surface Vehicles (USVs) and their adoption in the scientific community is certainly more recent. Indeed, even if the first prototypes date back to World War II, USVs have always been mostly used and developed for military applications. Only in late nineties and with the new century that the USVs have been finally taken into consideration for scientific



Figure 1.2: The SESAMO catamaran on the Malippo stern deck (Terra Nova Bay, February 2004)

use.

USVs have several advantages compared to AUVs, including: simpler and cheaper energy supply, possibly using internal combustion engines and generators, easily providing ranges in excess of 1000 km; data communications and navigation are more straightforward; construction costs are lower; and transit speeds are higher. But USVs are limited to making near-surface observations unless they carry automatic winches to emulate a standard shipboard hydrographic station [5].

The typical mission involves water sampling, in order to obtain a system capable of 24h sampling, and to capture specific events and locations in timely manner [6]. Recents efforts in the development of USVs technology have focused on the power aspects, in order to increase the autonomy of the system, and thus enabling the vehicle to perform very long range and long time missions. Indeed, as observing the ocean is becoming increasingly important in order to anticipate climate and weather changes, the need of acquiring direct samples over a large area is one of the new important aspects of the ocean engineering. To this aim, solar powered USVs have been recently demonstrated in their effectiveness [7]. Still aiming at the longevity of the mission, also autonomous sailboats have seen developments in past years, with the clear advantage of low power consumption since only the on board electronics and the rudder need to be powered [8].

One of the most relevant use of USVs in the scientific field is certainly repre-

sented by SESAMO (Fig. 1.2) and its participation to the XIX Italian Expedition to Antarctica, where SESAMO was employed to sample the sea-surface microlayer and immediate subsurface in the Terra Nova Bay area of the Ross Sea, in order to better understand the role played by the physical and chemical interactions between the hydrosphere and atmosphere in the prediction of global-scale climatic circumstances [9].

The aforementioned examples explain the focus of the last decade over the development of technologies and new algorithms for unmanned marine vehicles. As about 70% of Earth surface is covered by oceans, it is clear to see how unmanned system can help tremendously in the environmental monitoring of such vast area. Thus, engineers from all over the world are constantly trying to improve AUV and USV technologies in order to provide lower-cost, long-range and long duration tools for biologists, oceanographers and all the other scientists involved with the oceans.

In addition to the scientific aspects, also the security of waterways and harbors is a subject for research and possible employment of unmanned systems. Ranging from mine countermeasure to simple surveillance of restricted water zones, the applications of AUV and USV are broad. The main advantage in such applications is certainly the fact that no human being are put to risk.

Building on these premises, the present work aims to improve the current state of art mostly from an algorithmic point of view, for both AUVs and USVs.

The work is organized as follows. In the successive Section 1.1 the current on-going projects in which this research is framed are presented: subsection 1.1.1 explains the TRIDENT project about underwater dexterous manipulation; subsection 1.1.2 introduces the CO^3AUVs project, which covers cooperating AUVs; while the last subsection 1.1.3 explains the joint research project between DIST and Selex-SI for the creation of a command and control center for USVs in the harbor protection scenario.

The first section of Chapter 2 introduces the main applications and on-going challenges of the surface vehicles. Since for the surface part this work focuses on the surveillance application inside an harbor, the challenging problem of obstacle avoidance w.r.t. the other moving vessels is tackled first in Section 2.2. Then, the problem of "intercepting" a detected menace is presented in Section 2.3, while

the final Section 2.4 presents an optimization technique to improve the a-priori position of the fleet of USVs inside the harbor zone to protect.

The successive Chapter 3 is instead dedicated to the world of underwater vehicles. Again, the main applications and challenges are presented at the start of the chapter. Then, due to the importance of localization in the underwater domain, two different algorithms are presented. Section 3.2 explains the so-called RT^2 algorithm, which exploits the knowledge of the Speed of Sound (SoS) profile to accurately capture the anisotropy of the underwater medium, whereas Section 3.3 presents a localization algorithm which makes no assumption about the availability of the SoS profile.

Strictly linked with the world of AUVs, Chapter 4 deals with the emerging trend of Intervention-AUVs (I-AUV), which are essentially AUVs endowed with one or more robotic arms, and are thus able to carry out intervention and manipulation missions. Their main applications and challenges are presented at the start of the chapter, highlighting how the coordination between the arm and the vehicle represents the major area for research for such type of vehicles. The problem of free floating manipulation, and the proper coordination of the overall structure is thus tackled in Section 4.2, which details the control architecture and methodology that has been developed within the TRIDENT project.

The following Chapter 5 focuses on a software framework developed to ease the effort for implementing the aforementioned algorithms on board the vehicles. The framework has been designed to be object oriented, to favor the reuse of software; and to be portable, in order to run across different operating system possibly used in different vehicles.

Finally Chapter 6 draws some final conclusion about the presented work and presents some possible future lines of development.

1.1 On-going National and European Projects

The following section will briefly present the European and National projects in which most of the research presented in this work is framed.

1.1.1 EU FP7: TRIDENT

Nowadays a significant number of field operations that are carried out with autonomous underwater vehicles (AUVs) in applications like marine rescue, marine science and offshore, need manipulative capabilities in order to perform the desired task. In such scenarios, most of the intervention operations are undertaken by manned submersibles or by Remotely Operated Vehicles (ROVs). Manned submersibles have the advantage of placing the operator in the field of operation with a direct view of the object being manipulated. Their drawbacks are the reduced time for operation (typically a few hours), the human presence in a dangerous and hostile environment, and a very high cost associated with the vessel required for overseeing the overall operation. Work class ROVs, are probably the most standard technology for deep intervention. They can be remotely operated for days without problems. Nevertheless, they still need an expensive oceanographic vessel with a heavy crane, an automatic Tether Management System (TMS) and a dynamic position system. Another issue is the cognitive fatigue of the ROV's pilot who has to take care of the umbilical and the ROV while cooperating with the operator in charge of the robotic arms. For all these reasons, very recently some researchers have started to think about the natural evolution of the intervention ROV, the Intervention Autonomous Underwater Vehicle (I-AUV). Without the need for the TMS and the dynamic position system, light I-AUVs could theoretically be operated from cheap vessels with the opportunity of reducing the cost considerably. With the fast development of battery technology, and by removing the operator from the control loop, even manipulative operations lasting for several days are possible, where the ship is only needed the first and the last day for launch and recovery operations. For operations in inland waters or near the shoreline, it is even possible to replace the oceanographic ship by an USV - which dramatically reduces the operative cost.

In particular, TRIDENT proposes a new methodology to provide multipurpose dexterous manipulation capabilities for intervention operations in unknown, unstructured and underwater environments. The TRIDENT project is built on the top of two main concepts: 1) the use of a team of heterogeneous marine robot with complementary skills, an USV and an I-AUV, to achieve light intervention capabilities without the need for expensive support ships and 2) the use of a dexterous hand mounted on a redundant robot arm to achieve multipurpose

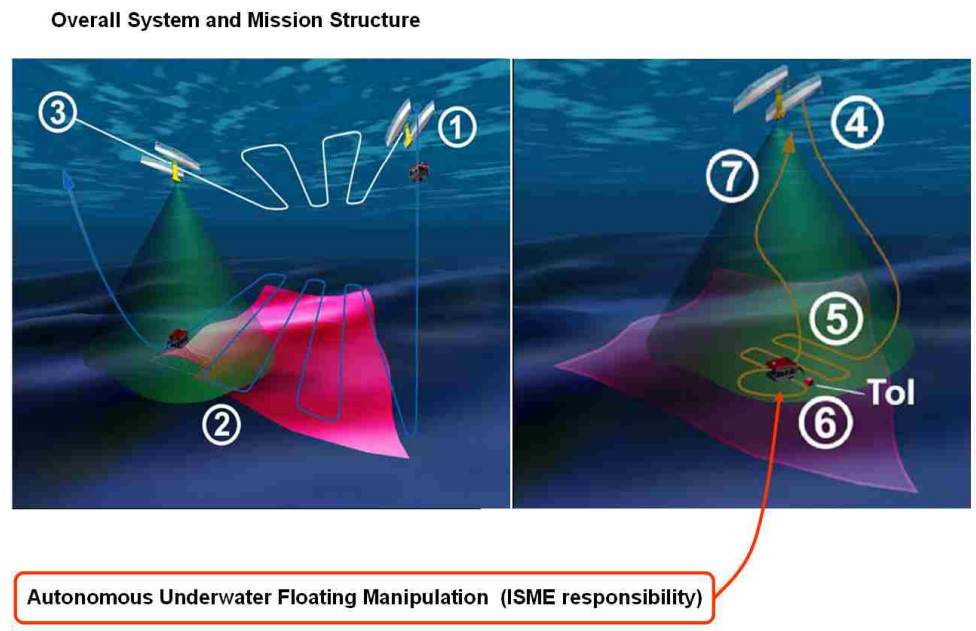


Figure 1.3: Different phases of the Trident's survey and manipulation

manipulation capabilities.

In the TRIDENT project, a multipurpose generic intervention is composed of two phases (see Fig. 1.3):

Phase I (Survey) : The USV is launched to carry the I-AUV towards the area where the survey is to be performed. Then, the I-AUV is deployed (1) and both vehicles start a coordinated path following and terrain following maneuver (2) to cover the area selected for exploration. During the survey, the USV/I-AUV team gathers navigation data so that the environment related measurements (seafloor images and multi-beam bathymetry profiles) may be geo-referenced. At the end of this phase the I-AUV surfaces (3) and contacts the end user to set-up an acoustic/optical map of the surveyed area. Using this map, the end user selects a target object (an item of interest for the mission) as well as a suitable intervention task (grasping, hooking, etc). The object is then characterized in order to allow for its automatic identification when it further appears in the field of view of the I-AUV.

Phase II (Manipulation) : Once the target has been selected, the USV/I-AUV team navigates towards a waypoint located near the target position. Then, the USV performs dynamic position maneuver (4) while keeping the

I-AUV inside the USBL cone of coverage. Then, the I-AUV performs a search pattern (5) looking for the Target of Interest (ToI). When the object appears in the robot's field of view, it is identified by the computer vision subsystem and the I-AUV switches to free floating mode using its redundant robotic arm as well as the dexterous hand to carry out the complex manipulation (6). Finally (7), with the help of the robotic arm, the I-AUV docks to the USV before recovery.

In particular, this work focuses on the point number 6 of the previous scenario, i.e. the manipulation via the use of the redundant arm.

1.1.2 EU FP7: CO^{3}AUVs

Currently, the use of single AUVs is limited to open-sea preplanned trajectories with off-line post-processing of the data gathered during the mission. State of the art AUVs are, as their name suggests, autonomous, i.e., they operate in a largely unsupervised way. But it can be argued that among all robotics fields, underwater robotics is currently the farthest from generating cognitive systems, i.e., devices that are not only able to act own their own but which also possess significant capabilities to adapt to unforeseen circumstances and requirements. There are several reasons for this. First of all, the underwater domain is a very challenging environment, where even human access is still very limited. Artificial systems cannot therefore rely on commonly found structures of "humanized" surroundings that land robots usually operate in. Second, the problem of human access makes also the risks and costs of system loss much higher for AUVs than for land robots. Third, available underwater sensors provide rather coarse and noise-ridden data in comparison to their land robotics counterparts. The whole cognitive chain from sensing over perception, modeling, planning to action has hence to be grounded under much more difficult conditions. This is actually the most crucial constraint from the cognitive systems viewpoint. Multi robot systems may help solving this problem as they allow a cooperative sensor coverage of the environment leading through mutual exchange of sensor data to better perception and modeling, which then forms the basis for better motion planning and navigation as well as overall mission execution. As simple as this strategy sounds, there are of course many significant challenges involved in turning it into reality.

Multiple AUVs are still in a very early stage of research and development.

The aim of the Co^{3}AUVs (Cooperative Cognitive Control for Autonomous Underwater Vehicles) project is to develop, implement and test advanced artificial cognitive systems for coordination and cooperative control of multiple AUVs. Several aspects are investigated within the project like the situation awareness, the deliberation and navigation, the behavioral control, which are all strictly linked with the communications issues.

The main challenges of the project can be structured in the following areas:

- cognitive control and world modeling;
- cooperative navigation and motion control for multi AUVs;
- underwater communications;
- non-conventional, geophysical-based navigation;
- behavioural control.

A particular important scenario that motivates the cooperation of multiple autonomous vehicles and poses great challenges to systems engineers, both from a theoretical and practical standpoint, is automatic monitoring and surveillance of ports and coastlines for safety and security, and commercial purposes. In this case, a possible mission scenario considers a small fleet of AUVs that execute automatically pre-specified paths above the seabed and periodically transmit variables of interest (e.g.pictures) to a base-station (and to the operator in charge). In addition, a number of AUVs stay dormant either on the seabed or at the sea surface. Upon detection of interesting patterns or events by the operator in charge, a signal is sent to a selected member of the dormant AUV to dispatch it to the spot detected so as to inspect the surrounding region in great detail. Meanwhile, the fleet of AUVs continues to execute the specified paths. Note that if this mission is performed in a harbour, e.g. harbour security, additional challenges arise. It is known, in fact, that navigation in a confined space, as a harbour is, is very difficult for AUVs due to muddy water, waves reflection, noise, etc. For this reason, additional capabilities, beyond the simple guidance algorithms actually available, are necessary for the AUVs to perceive their environment and to communicate amongst themselves.

1.1.3 Selex-SI: Swarm Management Unit

Confined/inshore waters may represent the best type of environment for the conduct of clandestine operations, especially from the sea: the features of such an environment weaken the detection and identification processes carried out by the defenders, while allowing the covert execution of illegal activities, as well as a better exploitation of surprise. The most significant illegal activities conducted in such an environment include infiltration, drugs/weapons smuggling, sabotage/disruption of infrastructures, up to terrorist acts; these activities are accomplished through the use of a variety of means of transportation, which include cigarette boats and power boats, rigid hull inflatable boats (RHIBs), fishing vessels, sailing/leisure boats, midgets, swimmers/divers, and swimmer delivery vehicles (SDV). Requirements as flexibility, rapid reaction, resilience, fast deployment, high level automation, modularity, plug & play have a key role in modern surveillance architecture definition.

To this aim Selex-SI, one of the international leading players in providing large systems for security and defense, is developing a CIP (Critical Infrastructure Protection) tactical architecture in accordance with the previous requirements. The CIP system integrates different types of sensors (e.g. radar, IR, TV, SONAR, underwater acoustic network) and classes of vehicles (Surface, Underwater, Air Unmanned Vehicles) which cooperate in a multi environmental scenario.

Indeed, exploiting a team of Unmanned Surface Vehicles (USVs) for the protection of harbors is one of the areas under major investigation. The USVs could be used to perform patrolling around the crucial waterways or to investigate possible suspect vessels, increasing the effectiveness of the security system, while at the same time decreasing the number of human beings directly exposed to threats.

A joint research project between DIST and Selex-SI has been established with the goal of creating the so-called Swarm Management Unit (SMU), a tool conceived for supervising the operations of a team of USVs performing semi-autonomous surveillance activities within civilian harbors. The SMU is intended to be an intelligent interface between the USVs and a remote human operator in charge of monitoring the activities in the harbour area.

During normal operative conditions, every team member performs patrolling surveys of the more crucial waterways by following pre-planned paths, selected

by the SMU with the intent of optimizing the area coverage. The USVs are equipped with sensors providing acoustic and/or optic images of the scene and act like additional mobile "eyes and ears" to the operator. The images taken by the sensors on-board the USVs are transmitted to the console connected with the SMU through dedicated radio-links. Even during autonomous patrolling phases, the operator is always in the condition of assuming the direct control of one or more USV. In this way the human-in-the-loop requirement imposed by current legal rules is guaranteed. Further, whenever the operator detects some presumed anomalies in some particular area, he or she could decide to specify absolute or relative motion commands for the USVs via the SMU, in order to drive one or more of them towards the point of interest for performing a deeper analysis of the suspect situation.

A different, more critical, situation arises whenever an "unknown" vessel (i.e. any unrecognized incoming ship or boat) is detected by the harbour radar system. In this case the SMU prompts the attention of the operator and automatically suggests him the most suitable USVs to be used for "intercepting" the unknown vehicle (i.e. moving as fast as possible towards it) and allow its "classification" (i.e. determining whether the unknown is "hostile" or "friend").

Once the operator selects a USV for the classification mission (be it the one suggested by the SMU or another one), the SMU immediately plans the reference path driving it to the unknown vehicle and sends it the resulting list of waypoints to follow. If the unknown vessel is classified as hostile, the harbour emergency procedure is fired, otherwise the intercepting USV comes back to its usual patrolling activity.

Chapter 2

Unmanned Surface Vehicles

The development of USVs dates back to World War II, where the first experimentation with USVs took place. Canadians developed the COMOX torpedo concept in 1944 as a pre-Normandy invasion USV designed to lay smoke during the invasion, as a substitute for aircraft. Although COMOX was not deployed, a vehicle was constructed and a successful test was completed.

Post-war applications of USVs expanded, with the US Navy using drone boats to collect radioactive water samples after atomic bomb blasts Able and Baker on Bikini Atoll in 1946. The 1950s-era US Navy Mine Defense Laboratory's project DRONE constructed and tested a remotely operated mine-sweeping boat in 1954. The development of USV for mine clearance continued, and by the 1990s, the Navy developed and tested more sophisticated USV mine sweeping systems, [10], including the R/C DYADS, the MOSS, and finally ALISS - which demonstrated a remotely operated simultaneous acoustic and magnetic influence sweep capability.

From the 1990s, USVs started to be an interest topic also for research and civil applications. MIT developed in 1993 one of the first USVs, designed for various missions. Its name was ARTEMIS and it was used as a platform capable of testing the navigation and control systems required by an USV. It was then used to collect simple bathymetry data in the Charles River in Boston, MA [11].

To continue the automated bathymetry experiments begun with ARTEMIS, a new USV was developed. The specifications of the next USV were based on a desire to create a system as versatile and useful as a small manned vessel while maintaining a small size to allow for easy deployment and survey operations. The new USV ACES (Autonomous Coastal Exploration System) was developed during 1996 and 1997 [12]. The completed USV underwent field tests off Gloucester, MA

during the summer of 1997. Upon completion of these trials it was outfitted with sensors suitable for hydrographic survey and successfully completed such a survey in Boston Harbor in December 1997 [13]. To conclude the survey of MIT's USVs development, in 2004 four USV named SCOUT(Surface Craft for Oceanographic and Undersea Testing) were built by the MIT Department of Oceanic Engineering for developing robust control software for cooperating AUVs [14].

Meanwhile in Europe, from 1998 to 2000 the German Federal Ministry of Education, Research and Technology sponsored the MESSIN project for the development and testing of the prototype USV Measuring Dolphin for high accuracy positioning and track guidance and carrying of measuring devices in shallow water [15]. During the same years, the European Union funded the project ASIMOV (Advanced System Integration for Managing the coordinated operation of robotic Ocean Vehicles) for the development, among other things, of an USV for supporting a fast direct acoustic communication link with an AUV, and, thus, between the AUV and a support vessel [16]. To this aim, the Instituto Superior Tecnico of Lisbon designed and developed Delfim, a 3.5 m catamaran, characterized by a wing-shaped central structure able to carry acoustic transducers [17]. The vessel was also used as a stand-alone unit for collecting bathymetric maps and marine data. In addition to Delfim, Lisbon IST is developing Caravela, a long-range autonomous research vessel, for testing advanced concepts in vehicle/mission control and radar based obstacle avoidance and demonstrating technologies for the marine science community.

In the last decade, the USV community continued its growth. Among others, a special mention deserves the SESAMO USV, a small catamaran-like shape prototype vehicle originally developed by the CNR-ISSIA in Genova. Its main application is the sampling of the sea surface microlayer and immediate subsurface for the study of the sea-air interaction and was successfully used during the XIX Italian Expedition to Antarctica [9] as already pointed out in the introduction chapter.

Finally, recent and important efforts have been done by the robotics group at the US Space and Naval Warfare Systems Center in San Diego, who developed an USV test-bed as versatile platform for rapid prototyping and testing of new concepts [18].

2.1 Main Challenges and Applications of USVs

2.1.1 Main USVs Applications

The following paragraphs will underline the current main applications of USVs in the scientific community and their possible use in safety and protection contexts.

Supporting Vessel

One of the application for autonomous vessels is to be the support ship for one or more AUVs that are operating underwater. The USV mission can be a simple relay of information from the AUV to the actual manned support ship or base station [19], but it can also require the participation in the actual localization of the AUV, acting as a moving buoy or beacon in a Moving Long Base Line (MLBL) localization system [20, 21].

Sampling

Sampling the water to check its quality is usually done by acquiring samples in fixed spots, just beneath the surface. This sampling method is quite simple and cheap to perform, and is mostly done manually along shores and rivers or by vessels in the sea. Such task can be easily automatized with the use of an USV, which has the advantage to be able to navigate in shallow waters, where a survey vessel certainly cannot go, and in polluted areas, with considerable savings in time and money (since operating a vessel is always costly) [22].

Protection of Harbours and Waterways

Another important application of USVs is the surveillance and protection of harbors and waterways. Indeed, ever since the attacks of September 11th, there has been an increase focus in security, and harbors do not make an exception.

In a harbour protection scenario one wants to monitor the harbour's traffic in order to reveal and eventually intercept possible intruders that can pose a threat to sensible targets inside the protected area. To this aim it is clear that the area has to be covered by some kind of sensors in order to reveal such intruders. Usually this is achieved by properly positioning radars and cameras all around the area to be protected; there are cases though where the coverage cannot be

complete, which explains why the need of using USVs, equipped with proper sensors (infrared, cameras, sonars, etc.) arise. Patrolling USVs one can properly fill the gaps in the areas not covered.

Another reason that goes for employing USVs is that they enhance the capabilities of the overall system to recognize and classify the unknown radar tracks because they can be equipped with cameras and can get close to the unclassified vehicles. Last but not least important, they can intercept possible threats.

However, to reach this ambitious goal, since the USVs need to operate in manned waterways, a particular focus must be posed on the security of the motion of such USVs. If on the one hand one wants to guarantee the security of the area against possible threats, and thus need to intercept them fast, on the other hand one needs to make sure to not harm any other friendly target moving around in the area.

Vessel Escort and Asymmetric Protection

A further protection mission that an USV can perform is the vessel escort, with particular care to the case of asymmetric menaces. These threats generally include terrorism, unconventional or guerrilla tactics or guerrilla warfare. The objective of an adversary is to create instability, and usually in these cases the attack is performed by using irregular forces, prohibited weapons and improvised devices. Moreover it may use civilian facilities, vehicles and equipment as weapons or even be in contradiction to international and humanitarian laws [23].

As vessels are very vulnerable to such kind of attacks (take as example a suicide speed boat against a bigger vessel), a fleet of USVs could be employed to increase the capability of the vessel's security system to detect such menaces in time, and even take proper action against.

2.1.2 Main Challenges and Critical Issues

Obstacle Avoidance

The current applications of USVs are still confined to military zones, where the navigation of civilian vessels is prohibited, or to scientific missions, such as the SESAMO's Antarctica expedition, in hard to access areas or with strict human supervision.

For the adoption of the USV in daily scenarios, such as their use in civilian harbors or sea shores, the important topic of obstacle avoidance has to be tackled. Despite the research efforts of the last decade, this remains an open problem, since the developed algorithms still need to be reliably tested in real harbour scenarios. Indeed, a recent survey of Caccia [24] shown a lack, at the current state-of-the-art, of a reliable methodology for obstacle avoidance. Moreover, the same survey underlines how the bottleneck is yet in the availability of effective and reliable obstacle detection sensors. On this matter, preliminary work is being carried out in the military field, trying to integrate a radar and artificial vision devices on the SSC San Diego [18].

Maritime Law and COLREGS

Another important issue that needs to be surpassed is the hole in the maritime law for what regards unmanned system. Indeed, even today, an unmanned vessel at sea may be taken by any claimer. Thus there are still no regulations for what regards the operations, approval, insurance of this kind of vehicles, which restricts their operations within areas forbidden to civil traffic, or under a strict surveillance during scientific operations.

Further aspects that limit the applicability of USVs in manned areas is the fact that most system do not comply with the COLREGS (International Regulations for Preventing Collisions at Sea, see [25]). Indeed most obstacle avoidance systems do not take such regulations into account. However, recent efforts in this direction have been undertaken by the Naval Centre in San Diego [26, 27].

2.1.3 Proposed Algorithms

After this brief summary of applications and main critical issues of USVs development, this work will focus on the use of USVs for harbor protection, within the framework of the on-going joint research project with Selex-SI called Swarm Management Unit (see Section 1.1.3), whose main goal is to supervise the operations of a team of USVs performing (semi-)autonomous surveillance activities within civilian harbors.

Previous experiments on USVs where mainly performed in open sea or in waterways in absence of other unknown moving vessels (see [28, 24] for nice surveys on the topic).

In a civilian harbour the situation is completely different and more crucial issues arise: the ship traffic is intense and not completely under control; operations of tourist or merchant ships cannot be delayed or affected anyway by the security vehicles, therefore the manoeuvres of USVs must not perturb at all the normal harbour activities. Further there is a concrete risk of collision with other vessels, with consequent risks of personal injuries and property damages. The consequences of the above safety considerations clearly impact noticeably on the required characteristics of the overall USVs-based security system. First of all USVs have to be provided with efficient path-following capabilities, as they are expected to follow the reference path with a certain accuracy. Further they need emergency sensory devices enabling a reliable detection of any incipient obstacle. Finally they need suitable techniques for guaranteeing autonomous reactive obstacle avoidance capabilities, in case an unforeseen obstacle is actually detected. A number of research activities on each one of the above topics have been registered and some interesting results have been recently proposed [29, 30, 31, 32]. Despite for some of the above issues consolidated results are not yet available [24], for the sake of the present work the considered USVs will be assumed totally compliant with all the above requirements.

Even under the hypothesis of almost-perfect USVs, the need of efficiently coordinate their operations still remains an important issue to be solved for obtaining an effective and reliable harbour security system. In this perspective, the SMU aims to represent a good answer to this demand. Among the other provided functionalities, certainly the more important role played by the SMU concerns the real-time planning of all the USVs' motions. This is done on the basis of the information, extracted from the main harbour radar system, concerning current positions, speeds and heading angles of all the vessels in the harbour area.

Despite the on-line availability of any radar track, the problem of driving unmanned mobile vehicles among moving obstacles remains a not straightforward task. The problem of safety induces the need for paths preventing any risk of collisions with a desired margin of tolerance. Further, as the environment can dynamically change in a not totally predictable way (because some of the moving vessels might suddenly change their speed or direction of motion), the motion planning procedure must be fast enough (in terms of computation time) to generate a new corrected path anytime a variation of a radar track is detected (i.e. within the period of a radar scansion, namely around 1 second).

Finally, since a prompt reaction to any detected anomaly represents a key feature for the effectiveness of a security system, reference paths generated by the motion planner must be as much as possible optimal in terms of guaranteeing minimal-time trajectories to targets.

Moving from the above requirements, the main research efforts so far spent within the SMU project have been focused on the motion planning problem. An efficient technique guaranteeing minimal-time path under some assumptions has been implemented and a simulation environment for testing and validating algorithms has been developed and is presented in Section 2.2.

Furthermore, the problem of intercepting a detected menace before it could reach a particular "asset" (i.e. a crucial site to be maintained safeguarded) is also tackled in the successive sections.

An effective solution to such a problem requires two main components. Once a menace is detected, a prompt reaction is obtained by *on-line* selecting the best USV for the interception, by taking into account all the following factors: position of the menace and its measured (by the harbour radar system) speed and heading, position of all the available "interceptors" USVs and their kinematic characteristics, current traffic conditions.

Further, it is also of fundamental importance to determine a-priori the best nominal positioning of all the interceptors, in order to increase their chances of success. To better understand this last consideration, note that a vessel is classified by the global security system as a menace in case, at a certain time, it starts exhibiting a suspect behaviour, like if it suddenly starts moving fast towards the asset. Since a vessel may become suspect when it is at any location (inside or outside the harbour), a menace can in principle "appear" for the first time in any point of the considered area.

Therefore the second component for an effective reaction is the *off-line* optimization of the nominal positions of all the interceptors, to be determined as a trade off between two different needs: protecting the asset and guaranteeing a proper area coverage against possible menaces appearing at different locations.

In the present work both the above sub-problems are tackled and solutions optimizing given indexes of performances are proposed. The on-line selection of the most suitable interceptor is first described in Section 2.3, as it helps introducing concepts and notations used in the following. Then, the off-line optimization of the position of the USVs is detailed in Section 2.4.

2.2 Real Time Path Planning and Avoidance of Moving Vessels

The problem of finding the shortest or the minimal-time path connecting two points in the presence of obstacles has been a topic of interest in robotics, machine intelligence, and computational geometry for many years (see for instance [33, 34, 35, 36, 37]). Despite the problem of motion planning around stationary obstacles is quite easy to solve and many best-path searching algorithms have been used for years (see [38] for a nice survey), in its general formulation, the dynamic case (with moving obstacle) is a very difficult problem. Even the simpler case of a point-wise robot acting on a plane where the obstacle are simpler polygons moving with constant speed without rotation has been proved to be NP-hard [39]. Nevertheless, it has also been shown that when some assumptions hold on the kinematic properties of the robot and the moving obstacle, the problem can become more tractable. In this perspective, an important result has been obtained by Fujimura and Samet in their very interesting paper [40]. In such a work the authors considered a modified version of the above 2D problem, in which: i) the maximum speed of the robot is greater than the speed of the obstacles; ii) the obstacles never collide each other nor overlap the target point. Under these two assumptions they proposed an algorithm that solves the problem of finding a minimal-time path for a point-wise robot with a complexity $O(n^2 log(n))$, being n the number of edges constituting the polygonal moving obstacles. Further they also demonstrated that whenever such a minimal-time path exist, it has to be run by the robot at its maximum speed.

The proven results of [40] have been very useful for the development of the SMU's motion planner. First of all, despite in their work Fujimura and Samet just considered obstacle moving with a constant linear speed (i.e. without changing their orientation on the plane), the case of steering obstacles can be easily coped within the considered application. Indeed the radar system always provides information on the instantaneous linear speeds of all the moving vessels. Even while a vessel makes a turn, the corresponding radar output after every scansion will be the vector of the current linear speed tangential to the curved trajectory. Since the algorithm processing time is highly smaller than the radar scansion period, a new reference path can be computed at any radar update. Of course, in case all the vessels move with a fixed speed (both in modulus and direction), the

algorithm will always produce the same optimal solution. Conversely, whenever a radar track variation occurs, the algorithm will lead to a different "corrected" reference path.

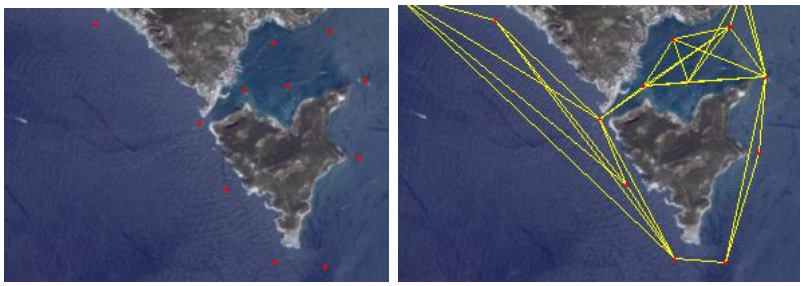
In addition in the considered domain of the harbour security problem, the two assumption in [40] reveal to be quite reasonable and can therefore be maintained without any loss of generality. Indeed the fact that a USV can move faster than any "friend" vessel is guaranteed by navigation rules which limit the speed of vessel inside the harbour area.

As it instead concerns the second assumption in [40], it also naturally holds in the scenario at hand, as two vessels obviously are not expected to collide each other. Further, since inside the harbour area vessels have to maintain a certain safety distance each other, every vessel can be conveniently enclosed in a "safety" rectangular bounding box, (properly over-dimensioned with respect to the actual size of the vessel) for which the non-collision assumption still holds. This solution leads to deal just with polygonal obstacles, thus simplifying sensibly the computation of the collision-free path. Indeed, in this way, the motion planner can easily keep into account all the moving obstacle by simply predicting their future positions through a movement of their bounding boxes computed on the basis of measured current speeds and heading angles. Finally, if the bounding box of every vessel are properly enlarged to keep into account also the dimension of the USV, the same approach of a point-wise robot suggested in [40] can be maintained.

Moving from these premises, a modified version of the algorithm proposed in [40] has been implemented by using the A* method for efficiently exploring the tree of possible alternative collision-free paths, while looking for the minimal-time one. However, before tackling the problem of avoidance of moving obstacles, the algorithms used for the avoidance of static obstacle are presented in the next subsection.

2.2.1 Avoidance of Static Obstacles

The known static obstacles in the area are for example the coast, islands, docks and so on, but also zones where the user decided to prohibit USV navigation inside. Planning a collision free path with static obstacles is well known in the robotic literature and the visibility graph approach has been used, which is a



(a) (b)

Figure 2.1: (**a**) red points are candidate vertexes; (**b**) resulting visibility graph

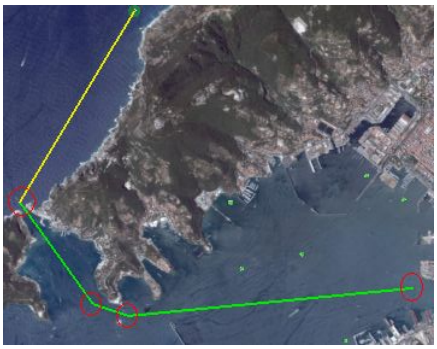


Figure 2.2: Example of path planning: yellow represent the current segment, green segments are the next segments already planned

consolidated technique in literature and has roots in [41].

After all obstacles have been made polygonal (see Fig. 2.1), the resulting graph is then solved via the well known Dijkstra's Algorithm [42]. The result of this operation is, for each destination vertex, an array that contains for each vertex the next vertex to visit in order to optimally reach the destination. This structure minimizes the amount of memory required to keep all the resulting paths available for use. Note that this operation, whose cost is $O(n^2)$ in the number of vertexes, is only done once at startup.

To find the best path between two points S and G that are not part of the visibility graph, the only operation that has to be done online is determining which points on the graph are connected to S and G, which has a cost of $O(n)$ in the visibility graph's number of vertexes.

The result of this procedure is a set of waypoints the vehicle has to follows in order to reach the goal. Fig. 2.2 shows an example of planned path.

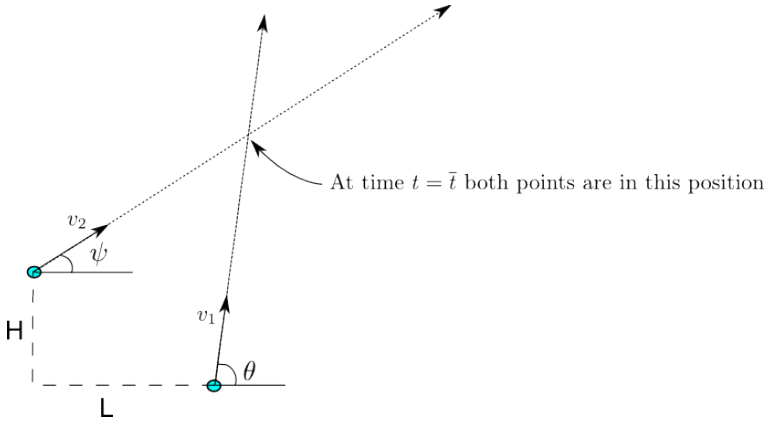


Figure 2.3: Schematization of the intercept problem

2.2.2 Real Time Optimal Avoidance of Moving Obstacles

In the following a series of increasing difficulty problems will be presented and discussed

- avoidance of one rectangular obstacle
- avoidance of multiple rectangular obstacles
- avoidance between multiple USVs and multiple obstacles

However, to better introduce the obstacle avoidance algorithm, it is necessary to start with the following auxiliary problem.

Auxiliary Problem: Interception of a Moving Point

The auxiliary problem requires that the USV intercepts a moving point in the plane, as schematized in Fig. 2.3. Let us assume that the USV moves subject to the following equations

$$\begin{cases} x_1(t) &= v_1 \cos(\theta) t \\ y_1(t) &= v_1 \sin(\theta) t \end{cases}; \tag{2.1}$$

and assuming the target is a point moving with fixed speed and a given heading, starting in a generic position $(-L, H)$

$$\begin{cases} x_2(t) &= v_2 \cos(\psi) t - L \\ y_2(t) &= v_2 \sin(\psi) t + H \end{cases}; \tag{2.2}$$

find, if any, an angle θ such that, for a certain $\bar{t}$, the USV intercepts the point, i.e. the following equalities hold

$$\begin{aligned} x_1(\bar{t}) &= x_2(\bar{t}) \\ y_1(\bar{t}) &= y_2(\bar{t}) \end{aligned} \tag{2.3}$$

By posing $x_1(t) = x_2(t)$ and solving for t, the time $\bar{t}$ of intercept can be found as:

$$\bar{t} = \frac{-L}{v_1\cos(\theta) - v_2\cos(\psi)}, \quad L \neq 0 \tag{2.4}$$

or equivalently, if $L = 0$

$$\bar{t} = \frac{H}{v_1\sin(\theta) - v_2\sin(\psi)}, \quad H \neq 0 \tag{2.5}$$

By putting $y_1(\bar{t}) = y_2(\bar{t})$ the final condition for the intercept is met leading to

$$-Lv_1\sin(\theta) - Hv_1\cos(\theta) = -Lv_2\sin(\psi) - Hv_2\cos(\psi) \tag{2.6}$$

The previous equation in the unknown θ can be solved recalling the following trigonometric property

$$a\sin(\vartheta) + b\cos(\vartheta) = \sqrt{a^2+b^2}\sin(\vartheta+\alpha) \tag{2.7}$$

where $\alpha = \arctan(\frac{b}{a})$. Exploiting (2.7) to solve (2.6), then θ can be easily calculated as follows:

$$\theta = \arcsin\left(\frac{v_2}{v_1}\sin(\psi+\phi)\right) - \phi \tag{2.8}$$

subject to

$$-1 \leq \frac{v_2}{v_1}\sin(\psi+\phi) \leq 1 \tag{2.9}$$

and where $\phi \triangleq \arctan(\frac{-H}{-L})$. Equation (2.9) is the constraint that has to be satisfied in order to be able to find an angle. As it can be seen, if $v_1 > v_2$ there will always be a solution, while in the other case there could be one or not depending on the initial conditions of problem.

The position of the intercept point is easily calculated, by substituting $t = \bar{t}$ in (2.3)

$$\begin{cases} \bar{P}_x &= v_1\cos(\theta)\bar{t} = v_2\cos(\psi)\bar{t} \\ \bar{P}_y &= v_1\sin(\theta)\bar{t} = v_2\sin(\psi)\bar{t} \end{cases} \tag{2.10}$$

It has to be noted that problem (2.3) could have a second solution

$$\theta_2 = \pi - \phi - \arcsin\left(\frac{v_2}{v_1}\sin(\psi+\phi)\right) \tag{2.11}$$

subject to same constraint (2.9). However, for the majority of cases (always if $v_1 > v_2$), this solution will lead to a $\bar{t} < 0$, thus being an unfeasible solution.

USV Obstacle Avoidance of one Rectangular Obstacle

The solution of the previous auxiliary problem is used to build the collision avoidance algorithm. Let us consider the problem of avoiding a moving obstacle (Fig. 2.4a). As mentioned before, a safety bounding box is here considered. For each of its vertexes the algorithm solves the auxiliary problem detailed in the previous section, thus it computes the angle θ, together with the time $\bar{t}$ necessary to reach the intercept position (Fig. 2.4b). An A^* search tree is thus built, where each node contains: the position of intercept $\bar{P}$, the time of intercept $\bar{t}$, the estimated time to goal $h \triangleq |G-\bar{P}|/v_1$, the total estimated time $f = \bar{t}+h$. Figure 2.5a shows this situation, where some collision checks have been done leading to the removal of some nodes. Since an A^* search is employed to find the optimal solution, from the remaining nodes the one with the lowest total estimated time is then picked (see Fig. 2.5b), and the procedure is reiterated (Fig. 2.6) until the goal point can be reached without any collisions (see Fig. 2.7a). The resulting planned path can be seen in Fig. 2.7b.

USV Obstacle Avoidance of Multiple Rectangular Obstacles

The algorithm is very easily extended to work with an arbitrary number of moving obstacles. At the start of the procedure, the intercept vertexes with the bounding box are built for every obstacles, then each path is checked for collision against all obstacles, and discarded if a collision is detected. For example, a node J related to obstacle A can be removed if the path from S to J intersects obstacle B. Since the procedure is not imposing any order of avoidance between obstacles, nothing else but discard this particular node is necessary, because the procedure is already building the nodes to avoid obstacle B in parallel. After the checks are done, the situation is similar to the case of one obstacle, the only difference being that there are more nodes to pick from this time. The most promising node is again picked and the procedure reiterated. Note that the order of avoidance (e.g.

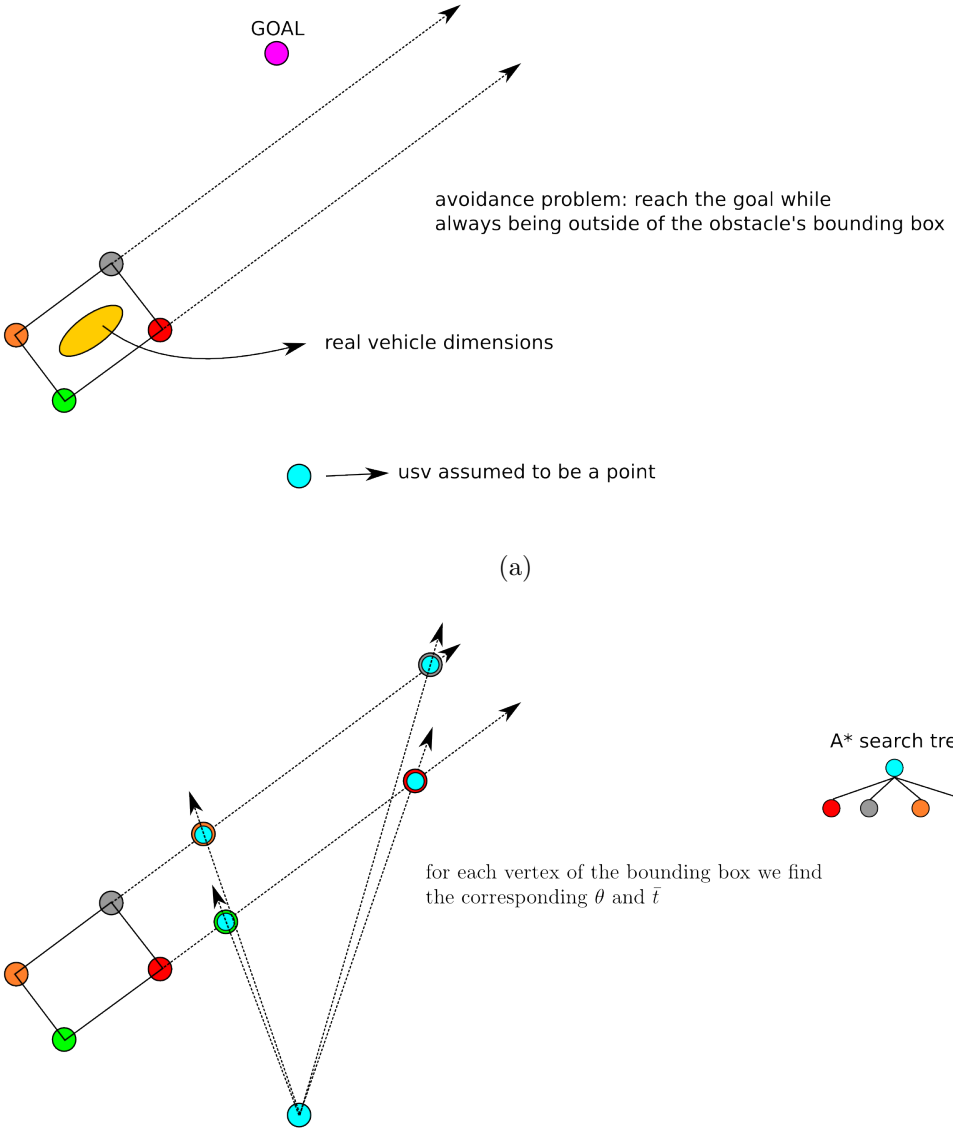
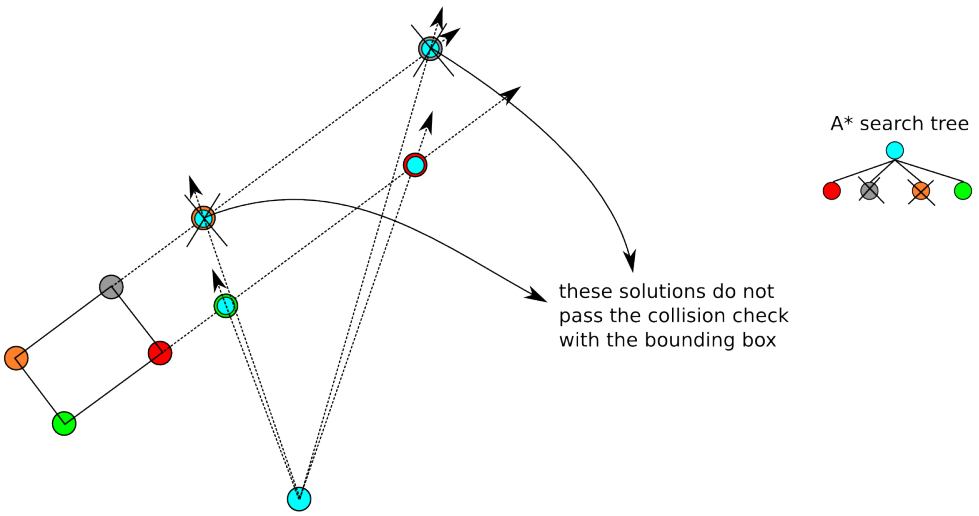
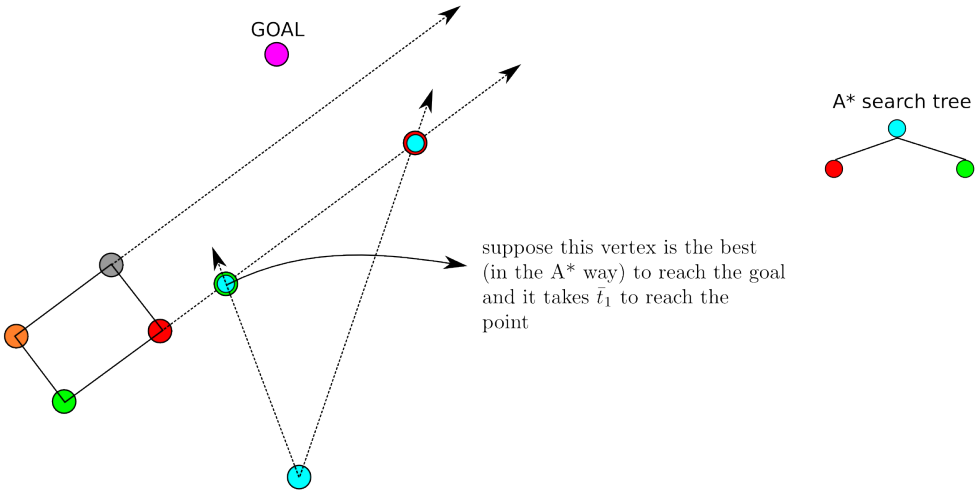


Figure 2.4: Simulation of the algorithm: (**a**) the original avoidance problem (**b**) the intercept points with all the vertexes are created

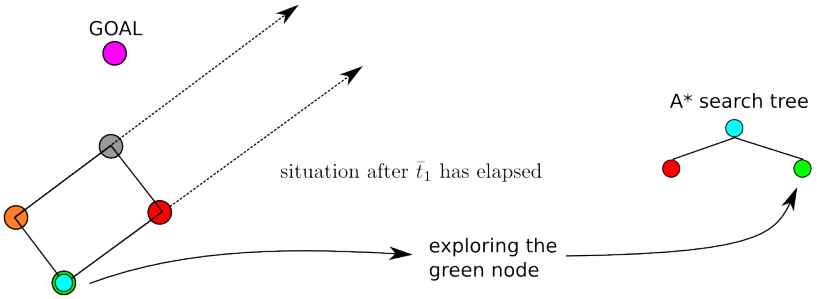


(a)

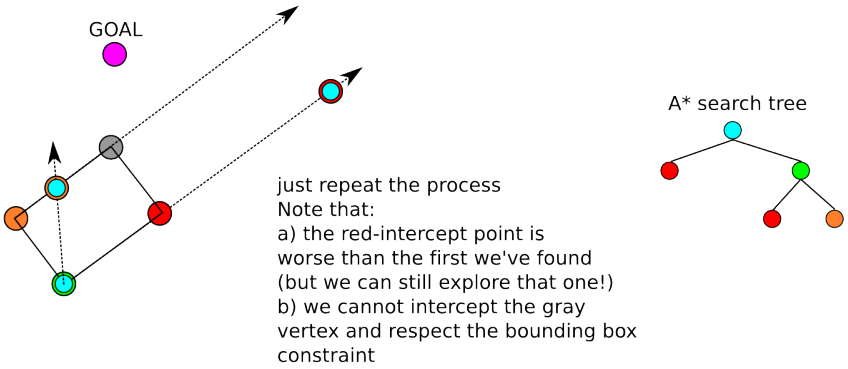


(b)

Figure 2.5: Simulation of the algorithm: (**a**) some of the nodes built are removed for collision (**b**) the most promising node is selected

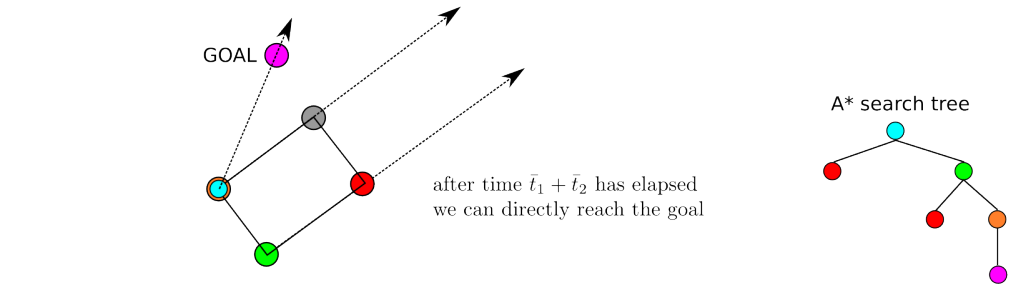


(a)

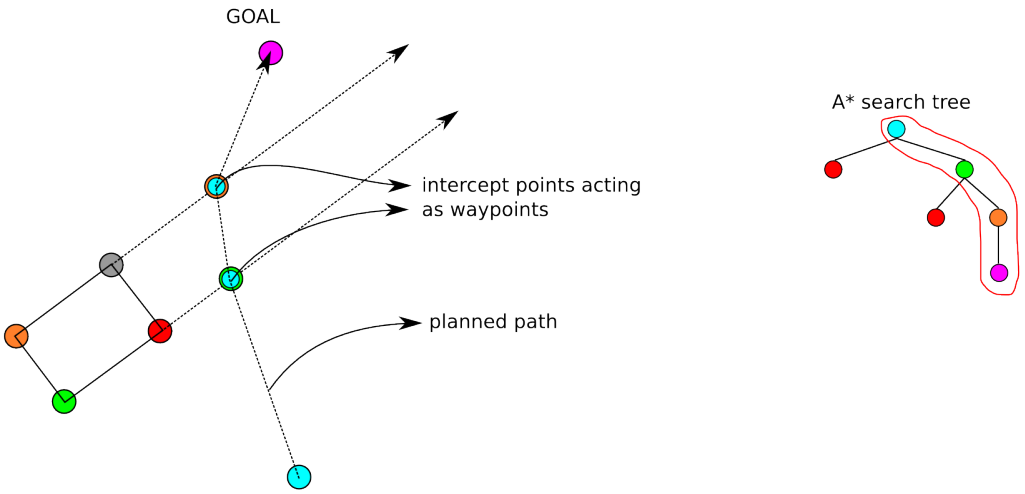


(b)

Figure 2.6: Simulation of the algorithm: (**a**) vehicle is at the box's vertex ($t = \bar{t}$) and the algorithm is executed once again (**b**) two new intercept points are obtained



(a)



(b)

Figure 2.7: Simulation of the algorithm: (**a**) goal can be reached directly (**b**) example of solution obtained

first A then B then C or C then A then B etc.) comes out automatically from exploring the best solution. An example is sketched in Fig. 2.8 and Fig. 2.9.

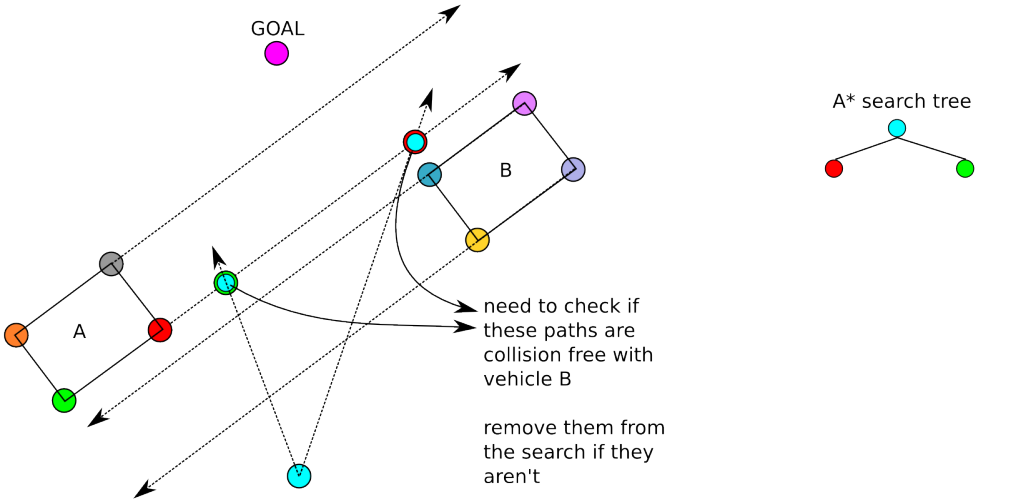
It must be recalled that the aforementioned procedure real-time runs at every radar scansion period, leading to an updated path for the USV.

Avoidance Between Multiple USVs and Multiple Obstacles

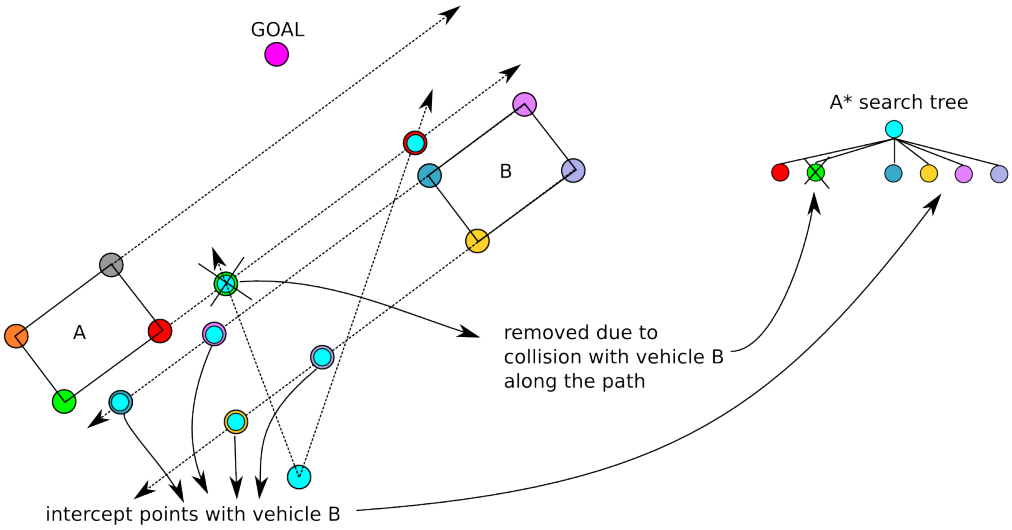
A special remark must be done for the mutual collision avoidance between USVs. In fact, in this special case each USV acts as moving obstacle for the other but, in contrast to what happened in the previous section, both paths can be changed. In order to solve any possible ambiguity a priority-based approach has been used: every USV is assigned an unique priority level on the basis of its current operation (e.g. a vehicle intercepting a threat has an higher assigned priority than one that is just patrolling); in case of intersecting reference trajectories, the higher priority USV ignores the other USV, thus it does not change its path. The lower priority USV instead treats the other USV just like any other vessel, and applies the aforementioned algorithm. Fig. 2.10 shows a scenario where three USVs are on route to collision and that the higher priority USV (USV1) does not change its path, leaving the job of avoiding collision to the others. This kind of approach achieves both avoidance and the fact that paths done by the USVs are optimized following their priority.

Implementation of the SMU's Motion Planner

Preliminary to the development of the final user-interface, a C++ software simulator has been developed in order to test the algorithms detailed in the previous section and contained in the SMU library, and has proved to be an invaluable tool for debugging the code of the SMU. In the simulator the user can see for each USV its currently planned path and the most recently followed trajectory, the current nominal area covered by the on-board sensors, and with a simple point-and-click is able to assume manual control of the USV and to generate a new reference position for it. The SMU then computes a list of waypoints for the USV to follow, based both on the visibility graph for avoiding static obstacles and on the real-time moving obstacles avoidance algorithm. Figure 2.11 shows some simulative results regarding the moving obstacle avoidance obtained with the SMU simulator.

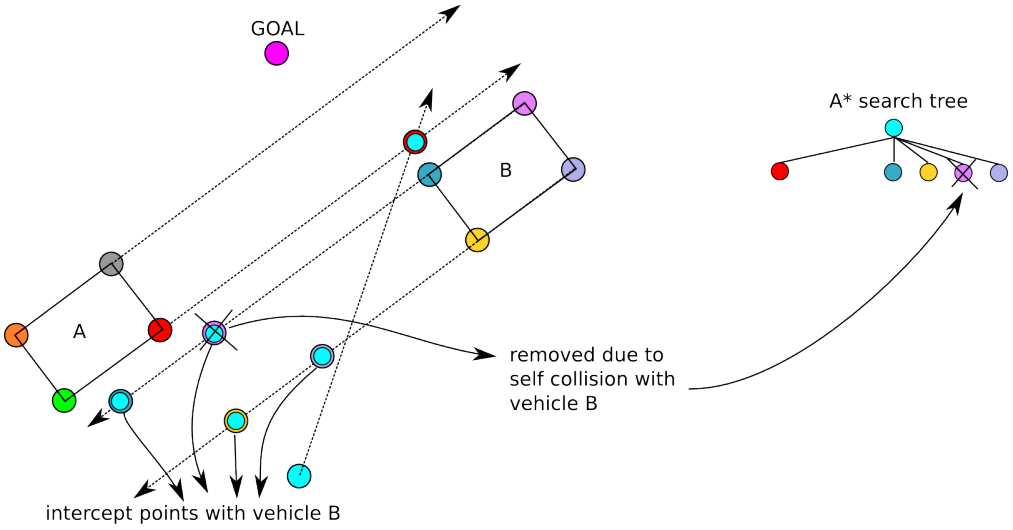


(a)

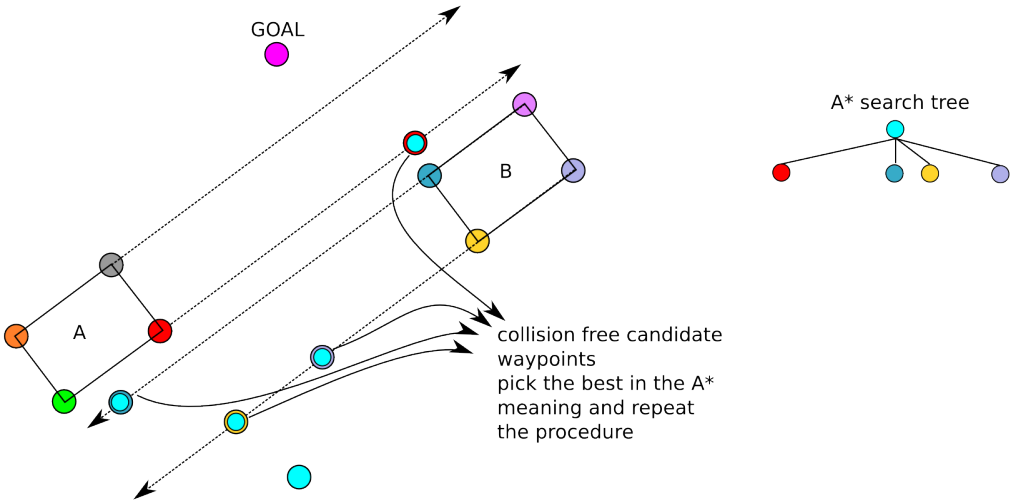


(b)

Figure 2.8: Simulation of the algorithm: (**a**) waypoints previously created for vehicle A only need to be checked for collision with vehicle B (**b**) intercept points with vehicle B are also added

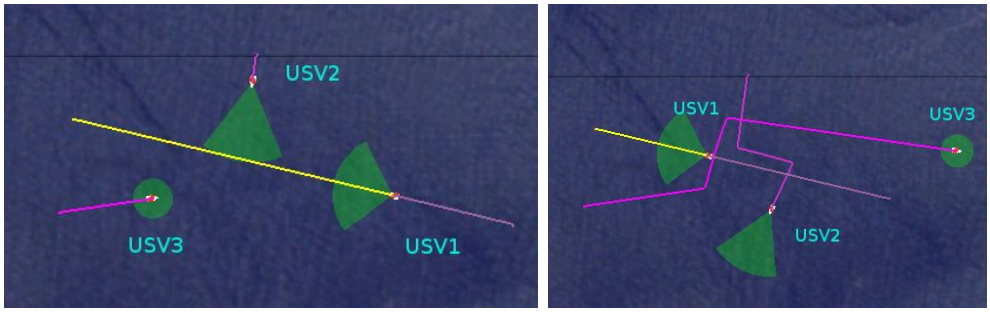


(a)



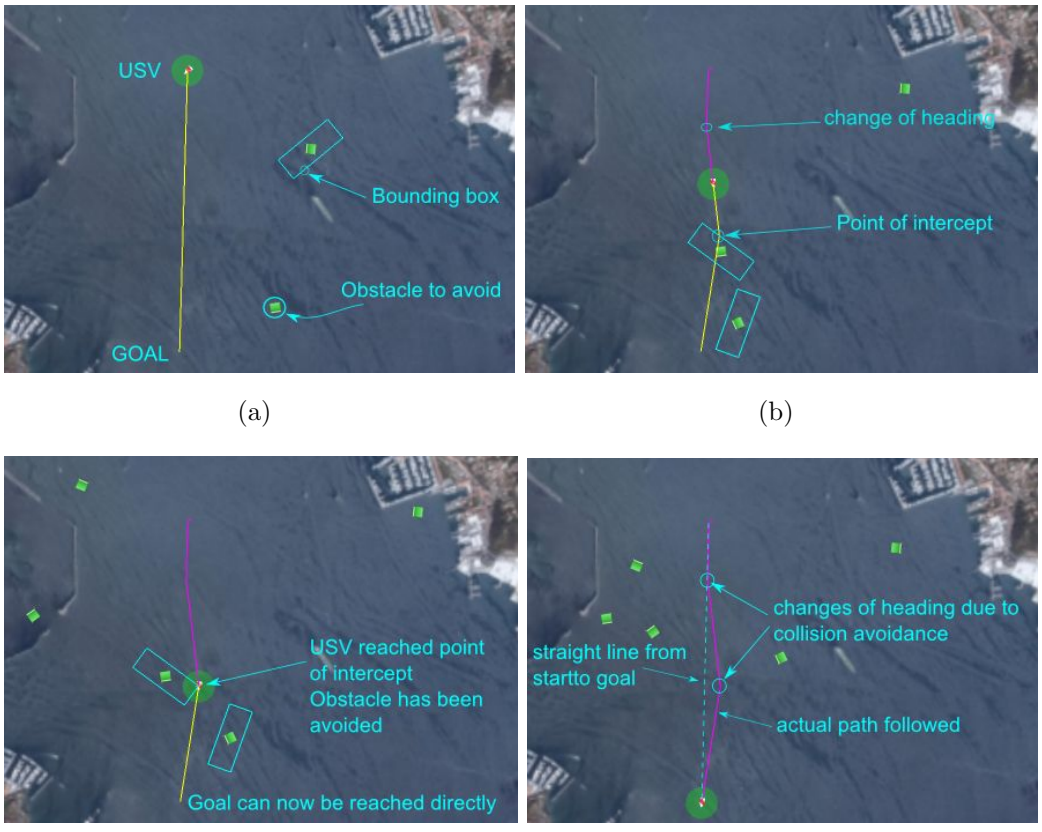
(b)

Figure 2.9: Simulation of the algorithm: (**a**) collision checking is performed for the new waypoints (**b**) the final set of waypoints, some created from vehicle A's vertexes, and some from vehicle B's ones. The algorithm is then repeated as in the single vehicle case



(a) (b)

Figure 2.10: Mutual avoidance: (**a**) potential collision situation, yellow path is the nominal path of the highest priority USV (**b**) collision is avoided without having the highest priority USV to change its path



(a) (b)

(c) (d)

Figure 2.11: Simulation of the algorithm: (**a**) initial trajectory (**b**) modified trajectory (**c**) vehicle is at the box's vertex ($t = \bar{t}$) (**d**) final position

The user can also drive all the simulated vessels in the area with the same point-and-click method; in this way, the user can manually control the threat and validate the SMU reactions.

Moreover, with a right click on the USV, a pop-up window appears detailing the current position, heading and speed of the vehicle, its fuel level and its current state (controlled by the user, patrolling, intercepting a threat, refueling and so on). Whenever an unknown vessel or a threat is detected, the simulator selects, through the SMU library calls, the best USV (in terms of intercept-time) and automatically sends it to intercept the vessel. In this way the user can actually see the performances of the SMU's suggestions.

In order to measure the effectiveness of the SMU algorithm, the USVs have been so far modeled as "ideal" vehicles (i.e. purely kinematic, without curvature constrains). As a consequence, the success rate of the performed simulations has been, as expected, a hundred percent. However tests were useful for gathering indications about real-time performances: the software was able to compute the optimal paths for 10 USVs with about 100 other vessels operating in a $10km^2$ harbor area within a few milliseconds on a standard 3GHz Intel Core Duo processor. The computational time is thus well within the boundary of the refresh time of the radar tracks (which is of about 1 second).

2.3 Interception of Detected Menaces

Once a menace is detected, the key factor for a prompt reaction is time. Therefore the best USV for the interception is the one which is predicted to reach the menace in the shortest possible time. In a hypothetic ideal scenario where only the menace and the USVs were present, the problem could be trivially solved by simple kinematic considerations based on the position and motion of the menace and on the position and speeds of the available USVs, similarly to the auxiliary interception problem shown in the previous section. However in the considered harbour scenario the situation is more crucial, because all the other vessels in the area represent fixed or moving obstacles which have to be avoided by the interceptor USV.

As a consequence, for every USV, the computation of its predicted time of interception is strictly related with the identification of its minimum-time path to reach the menace, which in turn depends on the contingent traffic situation.

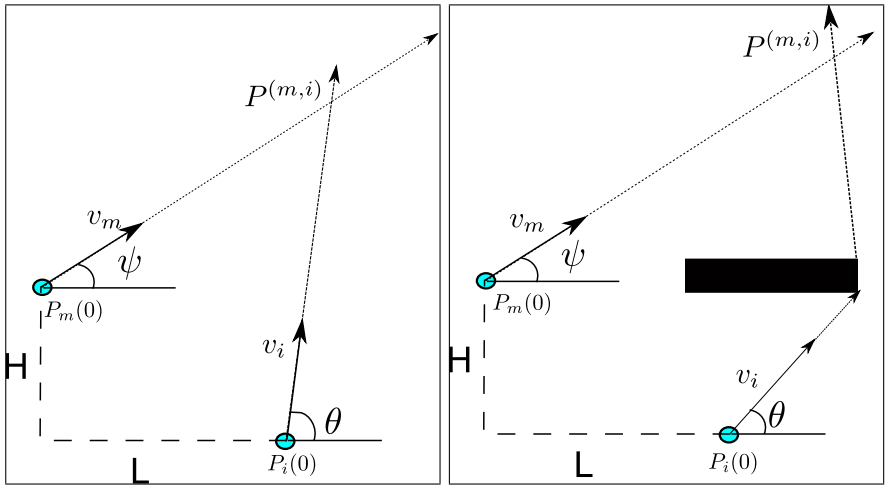


Figure 2.12: The interception problem: **a**) without obstacles (**b**) with an obstacle

To this aim the simplified problem depicted in Fig. 2.12 is first considered: given a single USV, say the i-th, moving at its maximum speed v_i, find (if any) the USV heading angle θ enabling the interception of a menace m, starting from a generic position $P_m = (-L, H)$ w.r.t. the USV and moving at a constant speed v_m with a constant heading angle ψ.

This problem is equivalent to the obstacle avoidance one. Indeed, intercepting a point-wise vessel is the same as calculating the interception point between the USV and one vertex of the bounding box of the previous section, or as simply thinking the goal as a moving one. Thus, the results of the previous section can be summed up and re-used for the interception problem.

By defining the motion of the i-th USV as:

$$P_i(t) \triangleq \begin{bmatrix} x_i(t) \\ y_i(t) \end{bmatrix} = \begin{bmatrix} v_i \cos(\theta)t \\ v_i \sin(\theta)t \end{bmatrix} \tag{2.12}$$

and the motion of the menace as

$$P_m(t) \triangleq \begin{bmatrix} x_m(t) \\ y_m(t) \end{bmatrix} = \begin{bmatrix} v_m \cos(\psi)t - L \\ v_m \sin(\psi)t + H \end{bmatrix} \tag{2.13}$$

if no obstacles are located between the menace and the USV (Fig. 2.12.a), the angle θ can be calculated as:

$$\theta = \arcsin\left(\frac{v_m}{v_i}\sin\left(\psi + \phi\right)\right) - \phi \tag{2.14}$$

subject to

$$-1 \leq \frac{v_m}{v_i} \sin(\psi + \phi) \leq 1 \tag{2.15}$$

where $\phi \triangleq \arctan(-H/-L)$.

In case solution (2.14) exists, given the angle θ, the time needed for the interception can be calculated as:

$$t^{(m,i)} = \frac{-L}{v_i cos(\theta) - v_m cos(\psi)}, \quad L \neq 0 \tag{2.16}$$

while the point of the interception is obtained as:

$$P^{(m,i)} \triangleq \begin{bmatrix} P_x^{(m,i)} \\ P_y^{(m,i)} \end{bmatrix} = \begin{bmatrix} v_i cos(\theta) t^{(m,i)} \\ v_i sin(\theta) t^{(m,i)} \end{bmatrix} \tag{2.17}$$

where the notation $(\cdot)^{(m,i)}$ means that the quantity $(\cdot)$ refers to the i-th vehicle intercepting the menace m.

The more realistic situation where at least one obstacle (be it a static or a moving one) prevents the USV from moving on a straight line is sketched in Fig. 2.12b. In such a case the motion planner first solves the set of interception problems between the USV and all the vertices v_k of the obstacle. To this aim, by considering the speed of the obstacle (be it zero or not) in lieu of the menace velocity in expressions (2.14), (2.16) and (2.17), the sets of feasible predicted interception points $\{P^{(v_k,i)}\}$ and related interception times $\{t^{(v_k,i)}\}$ can be computed as shown before.

Then from any of the resulting point $P^{(v_k,i)}$ the motion planner checks if the menace can be reached with a collision-free straight line. If this is the case, the resulting menace interception point and related time are computed. Otherwise the above procedure is re-iterated from the considered point $P^{(v_k,i)}$, by solving the set of interception problems with the remaining vertices of the obstacle.

During the process a tree of the possible alternative collision-free paths is built-up and explored by using the A^* method. At the end, the minimum-time path is always obtained (even in the presence of many obstacles), together with the resulting optimal $P^{(m,i)}$ and $t^{(m,i)}$.

By now moving back the attention to the original problem of on-line selecting the best interceptor, consider Fig. 2.13, representing a situation where a menace is discovered at point P_m and is moving towards an asset, located at point $P_a = (x_a, y_a)$. A given number of vehicles $i = 1, \ldots, N$ are located in their respective

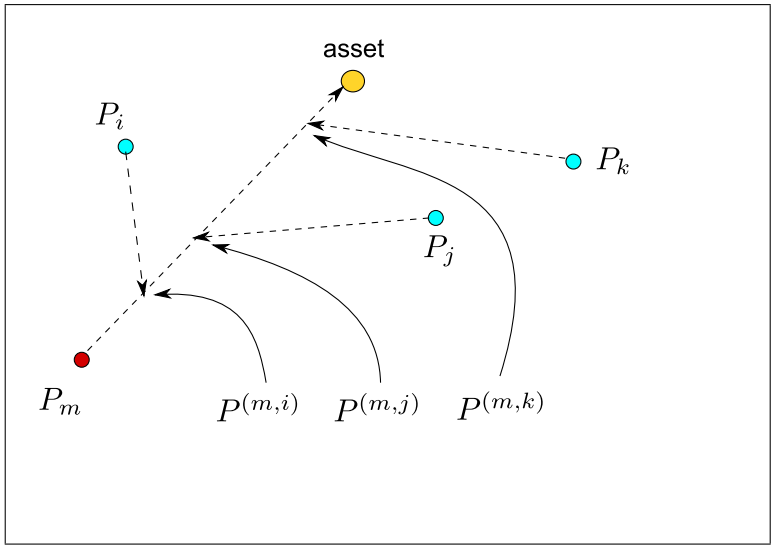


Figure 2.13: Schema of the intercept problem

positions P_i. No obstacles are sketched in the figure for clarity, but an arbitrary number of other vessels can be thought to be present in the area.

As soon as the menace is detected, for every USV, the predicted point of interception $P^{(m,i)}$ and related instant of interception $t^{(m,i)}$ is first of all calculated by the motion planner, as explained before. The problem of selecting the most suitable vehicle can then be simply stated as the following minimization problem:

$$\arg\min_i t^{(m,i)} \tag{2.18}$$

which can be easily on-line solved, given the availability of all the $t^{(m,i)}$ terms. As a result, other than indicating the best USV for the interception, the SMU outputs also the list of waypoints it has to follow for moving among the obstacles, along the minimum-time path to the menace.

2.4 Off-line Optimization of Fleet Positioning

Before considering the problem of better positioning the USVs for achieving adequate levels of protection, some preliminary considerations have to be drawn.

First of all, the statement that a menace can be detected for the first time *anywhere* in the considered area makes the problem of asset protection intractable, since the menace could be theoretically detected when it is already on the asset, making useless the presence of the USVs. In order to approach the problem via a

more meaningful formulation, it is therefore hereafter assumed that any menace is always certainly detected for the first time before it reaches a certain distance r from the asset. Such an assumption is not so unreasonable, because it can be easily assumed that in the proximities of the asset the navigation be totally forbidden to unidentified vessels; therefore any vessel trespassing the threshold distance r may be declared as a suspect vessel.

Further in the here considered off-line optimization problem, despite the actual interception time of a real menace is dependent on the traffic conditions at the time the menace is detected for the first time, no other vessels are assumed in the area, because their presence is highly time-variant and cannot be a-priori predicted in a reliable way.

Finally, as the velocity of any possible menace is not known a-priori, in the following it is assumed that the menace is always moving with a given upper-bounding speed.

Even under the above assumptions, the problem of off-line deciding the "best" nominal USVs positioning is not as straightforward as the on-line part of the interception problem, always admitting the optimal solution (2.18).

Indeed such a problem can be considered as a special instance of the so-called spatial resource-allocation problem, which has been studied for many years and has registered interesting results in particular in the field of fixed or mobile sensing networks (see, among the others, [43],[44] and references therein indicated). The goal in such cases is obviously finding out the best location of the available sensing resources in order to maximize a given index of performance, such as the area covered by the network, or the inter-node connectivity, or the confidence level of a measured quantity in the context of adaptive sampling.

Differing from the above cases, in the considered harbour security scenario it is not easy to identify a unique optimal criterion driving the optimization process. Indeed, since the positions of the USVs have a clear impact on the overall performances of the USV-based menace interception system, several *reasonable* alternative strategies can be adopted, in response to the specific index of performance one wish to optimize. For example, in order to reduce the average interception time, since a menace can appear at different possible locations, it might be convenient to spread all the available USVs in the harbour, for maximizing their resulting global area coverage. Conversely, for increasing the protection level of the considered asset, it would be more convenient to maintain all the USVs as

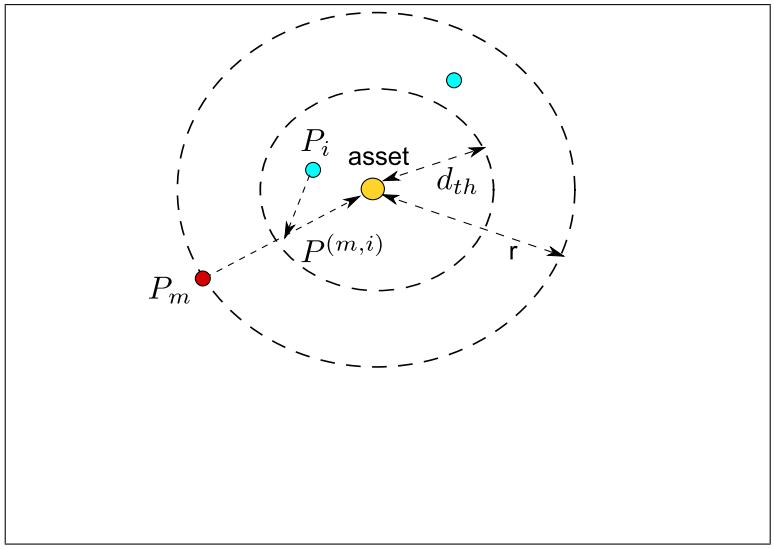


Figure 2.14: Schema of the intercept problem with a required minimum distance of interception

close as possible to the asset itself.

Moving from the above considerations, the proposed solution is based on the following two criteria. First of all the USVs must be in the condition of *always* intercepting *any* menace before it can reach a certain security distance from the asset. As shown in the following, the adoption of such a primary criterion, other than reducing the chances of the menace to harm the asset, provides, as a by-product, an indication on the minimum number of "asset-protector" USVs. All the (eventually available) other USVs can then be employed for the secondary criterion, here stated in terms of reducing the maximum interception time.

2.4.1 Preventing Menaces from Getting Too Close to the Asset

The problem of guaranteeing that *any* menace can be *always* intercepted, before it gets too close to the asset, can be translated into a worst-case-scenario optimization problem. Indeed it has to be granted that, even if the menace is detected in the closest possible position to the asset (i.e. at a distance r), the USVs are always in the condition of intercepting it on time (i.e. before the menace reaches a given security distance $d_{th} < r$, see Fig. 2.14).

In order to properly approach the formulation of the above optimization problem, let $P \triangleq \{P_1, \ldots, P_N\}$ denote the set of initial positions of a team of N USVs,

while let P_m be the initial position of a detected menace m. For any given set of initial conditions (P_m, P), the distance between the interception point related to the i-th USV and the asset can be easily computed as follows:

$$d_i\left(P_m, P\right) \triangleq \|P^{(m,i)} - P_a\| \tag{2.19}$$

It then follows that, for any given (P_m, P), the most suitable USV for the interception is selected as:

$$i^o = \arg\max_{i|P_m,P} d_i\left(P_m, P\right) \tag{2.20}$$

whose corresponding distance between the interception point and the asset is

$$D\left(P_m, P\right) \triangleq \max_{i|P_m,P} d_i\left(P_m, P\right) \tag{2.21}$$

By now solving the above problem for any possible P_m (while still maintaining fixed the set P), the worst-case-scenario, i.e. the point P_m leading to the closest to the asset interception point, can be calculated as:

$$D_w\left(P\right) \triangleq \min_{P_m|P} D\left(P_m, P\right) \tag{2.22}$$

By finally letting the optimization variable P vary, the optimal positions of the USVs are obtained as:

$$P^o \triangleq \arg\max_{P} D_w\left(P\right) \tag{2.23}$$

By combining all the above relationships, the following formulation of the original optimization problem is obtained:

$$P^o = \arg\max_{P} \left\{ \min_{P_m} \left[\max_{i} \left(\|P^{(m,i)} - P_a\| \right) \right] \right\} \tag{2.24}$$

whose corresponding distance from the interception point and the asset in the worst-case-scenario is clearly:

$$D^o = \max_{P} \left\{ \min_{P_m} \left[\max_{i} \left(\|P^{(m,i)} - P_a\| \right) \right] \right\} \tag{2.25}$$

By finally noting that, in case D^o results lower than d_{th}, it means that the considered amount of USVs is not sufficient to always guarantee the fulfillment of the security threshold distance, a simple algorithm is here proposed for determining the minimum number of USVs required for protecting the asset: solve problem (2.25) for an increasing number of vehicles k (starting from 1) until D^o is made greater than d_{th}. The resulting number k is then the minimum number of required asset-protectors, while the corresponding P^o is the vector of their optimal positioning.

2.4.2 Minimizing the Maximum Interception Time

The previous procedure gives the number k of vehicles necessary to meet the required minimum distance of interception from the asset. In case the number of available vehicles N is greater than k, the remaining $N-k$ USVs can be exploited to solve another kind of optimization problem: minimizing the maximum interception time. Indeed it is easy to see that such a criterion allows to spread out the extra vehicles in the area to safeguard, making the team more reactive against menaces detected at any distance from the asset greater than r.

To better approach the problem, it is convenient to split the set of USVs into two subsets: the first k vehicles with a fixed optimal positioning, as determined by the previous problem; and the remaining $N-k$ ones, whose set of positions $\hat{P} \triangleq \{P_{k+1}, \ldots, P_N\}$ is the subject of the here considered secondary optimization problem.

With these premises, for any given initial conditions (P_m, P), the most suitable USV for the interception is now the one with the lowest interception time, that is:

$$i^o \triangleq \arg \min_{i|P_m,P} t^{(m,i)} \tag{2.26}$$

whose corresponding interception time is

$$T(P_m, P) \triangleq \min_{i|P_m,P} t^{(m,i)} \tag{2.27}$$

Then, by again considering the menace in all the allowed positions, the worst-case-scenario can be found as:

$$T_w(P) \triangleq \max_{P_m|P} T(P_m, P) \tag{2.28}$$

Therefore the optimal positions of the extra-USVs can be found by minimizing the time of interception in the worst-case-scenario; that is:

$$\hat{P}^o \triangleq \arg\min_{\hat{P}} T_w(P) \tag{2.29}$$

Finally note that, since problem (2.28) considers all the possible P_m points, the extra USVs are spread out, the farther away from the asset, the bigger the considered area is. The following more convenient formulation of problem (2.28) can therefore be made, by introducing a proper weighting function $0 \leq W(P_m) \leq 1$ expressing the "probability of detection" of a menace in any particular point:

$$T_w(P) \triangleq \max_{P_m|P} W(P_m) T(P_m, P) \tag{2.30}$$

In this way the points at the boundaries of the considered area can have a very low weight, preventing the algorithm to spread the vehicles too much. Moreover, the points inside the circle of radius r should have a zero weight, since any menace is supposed to be already detected before the radius r.

By also considering the weighting function, the final formulation of the secondary optimization problem becomes:

$$\hat{P}^o = \arg\min_{\hat{P}} \left\{ \max_{P_m} \left[W(P_m) \min_i t^{(m,i)} \right] \right\} \tag{2.31}$$

2.4.3 Solution of the Off-line Optimizations

Both the off-line optimization problems have been solved using a Monte Carlo approach, coupled with a gradient descent algorithm. During each Monte Carlo run, the team of USVs has been randomly placed in the area, obtaining a particular set P. Then, the area has been discretized using a grid, and for each point the corresponding worst-case-scenario distances (2.22) or interception times (2.30) have been calculated. Then, the gradients of the cost indexes D_w and T_w were calculated, by varying one position of the set P at a time, and recalculating the new worst cases. This procedure was iterated until the variation was small enough, or a maximum number of iteration was reached. The final cost values and the relative positioning were saved and the best ones selected by comparing the different results of all the Monte Carlo runs.

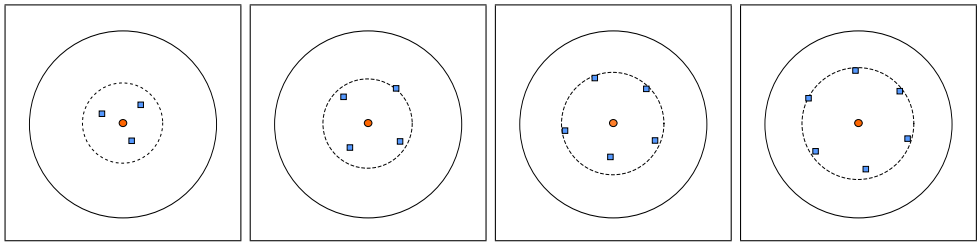
Naturally, since a numeric method has been used, all the solutions are affected by the discretization process and the numerical precision.

2.4.4 Simulative Results

Preliminary Simulations

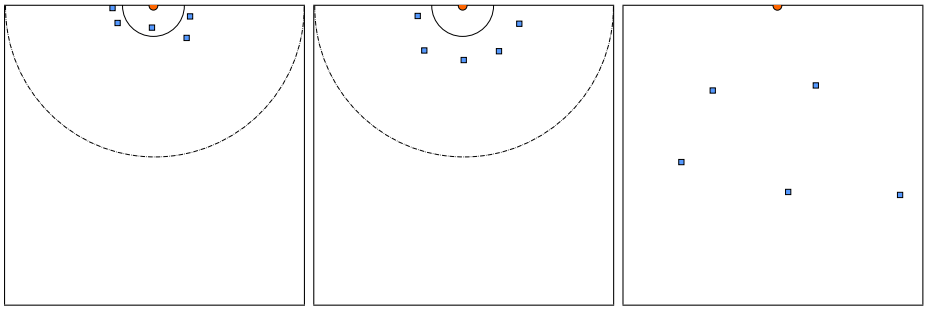
This section presents some simulative results, obtained by considering a simple squared area of $4km^2$ and solving the two off-line optimization problems. Cases with irregular borders, such as coast lines or harbour infrastructures, are presented in the successive section.

In all the following figures the orange circle represents the position of the asset, while the blue squares represent the positions of the USVs, as they result from the considered optimization problem.



(a) (b) (c) (d)

Figure 2.15: Different optimized results of the asset protection problem obtained by varying d_{th}: (**a**) $d_{th} = 86m$ (**b**) $d_{th} = 96m$ (**c**) $d_{th} = 110m$ (**d**) $d_{th} = 120m$



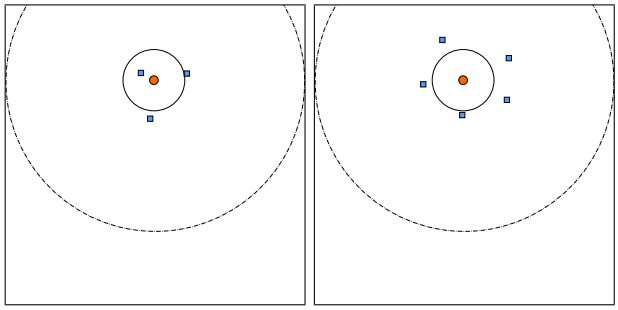
(a) (b) (c)

Figure 2.16: (**a**) USV with speed of 5m/s (**b**) speed 15m/s (**c**) with $W(P_m) = 1$

Fig. 2.15 shows simulations performed for the asset protection problem with a given forbidden region $r = 200m$ (solid circle), varying the security threshold d_{th} (dashed circle). Menaces are supposed to move at a speed of $20m/s$, while USVs are considered slower, with a velocity of $15m/s$. As expected, the minimum number of required asset protectors is strictly dependent on the dimension of d_{th}.

All the succeeding simulations tackle the secondary optimization problem, where in all the considered cases, a point-wise asset with a very low d_{th} threshold has been used. It is easy to see that the solution of the first optimization problem leads to a single asset protector located on the asset (and not depicted in figures). In this way the the positioning of the extra-USVs is emphasized.

The first simulation, depicted in Fig. 2.16a and Fig. 2.16b, shows the different results obtained with the asset placed in $(0, 1000)$, by varying the velocity of the USV. According to intuition, the slower the USVs are, the closer they need to be



(a) (b)

Figure 2.17: (**a**) with 4 USVs (**b**) with 6 USVs

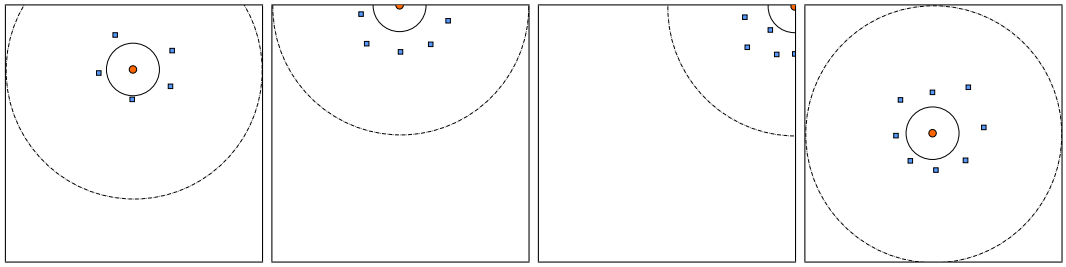
to the asset. The following weighting function is used:

$$W(P_m) = \begin{cases} 0 & d < 200; \\ (d - 200)/1000 & 200 < d < 600; \\ (1000 - d)/1000 & 600 < d < 1000; \\ 0 & d > 1000 \end{cases} \tag{2.32}$$

where $d = \|P_m - P_a\|$ is the distance of the considered point to the asset. In this and all the following simulations, the dashed circle represents $d = 1000m$ from the asset, so outside it is $W(P_m) = 0$. In the second example, shown in Fig. 2.16b and Fig. 2.16c, the effects of the weighting function $W(P_m)$ can be seen. While in Fig. 2.16b the use of the previously defined $W(P_m)$ is shown, in Fig. 2.16c the results have been obtained by setting $W(P_m) = 1$ for each point. As expected, not weighting the points makes the vehicles spread even more towards the points farthest away from the asset, as they are the ones with the highest possible interception time.

The simulation of Fig. 2.17 (again performed with the above weighting function and with speeds of $20m/s$, and $15m/s$ for menaces and USVs respectively) shows instead the effects of the number of available USVs.

Fig. 2.18 depicts the results obtained under the same conditions of the previous one by using 6 USVs, and the effects of the position of the asset with respect to the zone of interest. Indeed, if the asset is in an open sea zone, the USVs will be spread all around, while if the asset is on the margin of the zone, or even in one corner, the USVs are positioned in front of it. The last simulation (see Fig. 2.18d) shows a perfectly symmetric case, where the asset has been placed at the center of the zone and a radial $W(P_m)$ has been used as weighting function.



(a) (b) (c) (d)

Figure 2.18: (**a**) 6 USVs with asset in $(0, 500)$ (**b**) asset in $(0, 1000)$ (**c**) asset in $(1000, 1000)$ (**d**) 9 USVs with asset in $(0, 0)$

According to common intuition, the procedure places the USVs in a circle all around the asset.

Simulations in a Real Scenario

The preliminary simulations, presented in the previous section, shown results that are in accordance with common intuition in those very simple cases, without obstacles in the area and with symmetric properties of the area. This section presents the results obtained when the more realistic scenario of Fig. 2.19 is used as area of operation. In the following pictures, the red dot represents the asset to protect, the colored squares with black frame represent the position of the USVs, while the smaller colored dots instead indicate that if a menace appears in such position, the time-optimal vehicle is the one with the same color. Moreover, the smaller circumference around the asset is a graphical representation of the minimum safety distance d_{th}, while the bigger one represent the minimum distance of detection r. Figures 2.20a and 2.20b show the paths of the USVs when intercepting the candidate menace, which is represented with a colored circle. The color of the menace is the same as the best intercepting vehicle.

The first simulation is depicted in Fig. 2.21. Between the two cases, only the detection distance r has been changed. When r is increased, the position of the USVs can be further away from the asset, as r guarantees that the first detection of the menace cannot occur at ranges closer than it. Moreover, as it can be clearly seen, the USVs are chosen to guard the entry points to the inner harbour area, as these points are choke points, where a surface menace must pass through if it wants to reach the asset.



Figure 2.19: Red dot: asset; Colored squared: USV; Smaller circle: d_{th}; Bigger circle: r

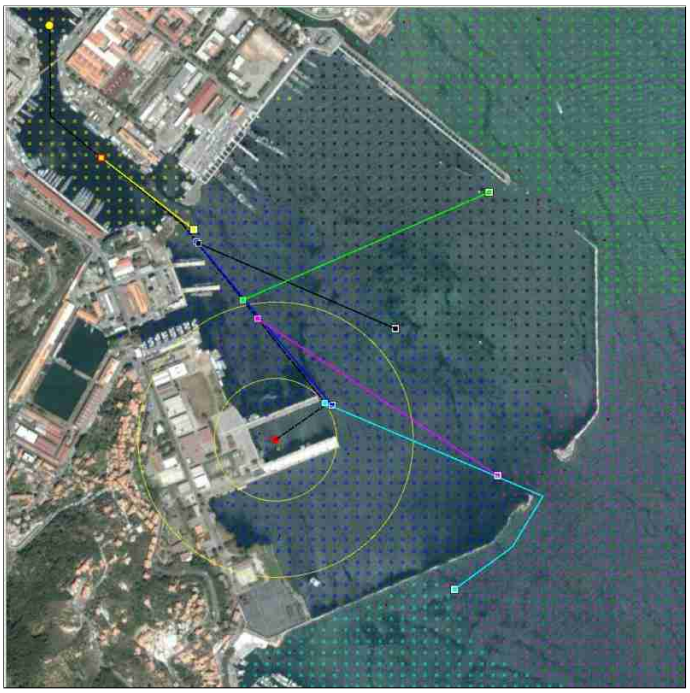
The second simulation, presented through Fig. 2.22, shows what happens if the minimum required interception distance is increased. This increase in the required performance of the system imposes that two vehicles are now necessary to satisfy the first optimization problem (the blue and black vehicles). Such change will obviously decrease the performance of the system w.r.t. the secondary optimization problem.

The successive simulation of Fig. 2.23 is instead aimed at showing the influence of the position of the asset and its location w.r.t. the obstacles in the map. As clearly seen in Fig. 2.23b, if the menaces can come from almost all directions, then the first criterion of guaranteeing the minimum interception distance becomes much more difficult to be fulfilled, and indeed the number of required vehicles increases from 1 to 3, with the same thresholds.

Finally, Fig. 2.24 shows what happens when the number of USVs used for the optimization is changed. In this case, by decreasing the number of USVs, the remaining vehicles all move closer to the asset to protect. Their position, as previously underlined, is still very close to the choke points.

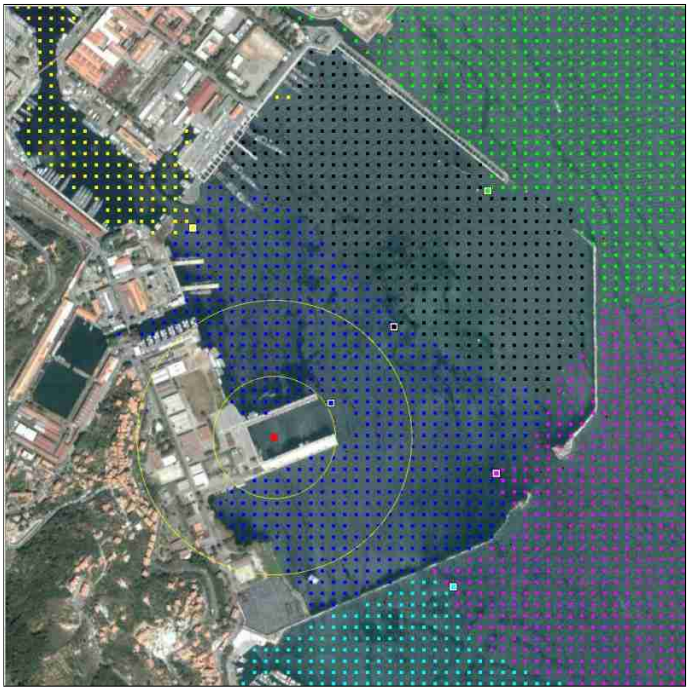


(a)

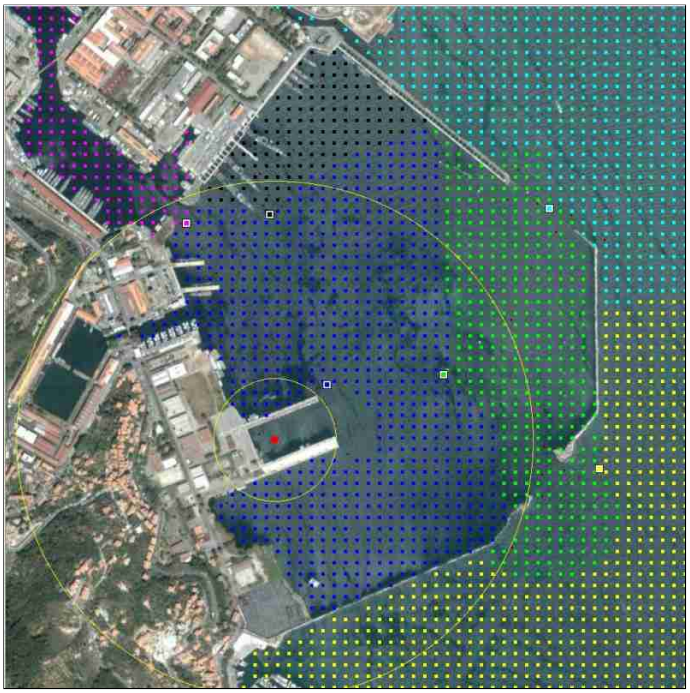


(b)

Figure 2.20: Different computed paths for the intercepting vehicles. (a) as the menace is detected just at the minimum distance r, the blue vehicle intercepts it before the minimum required distance d_{th} (b) menace is detected far away (top left corner) and thus multiple vehicles can intercept it, and the best one is the yellow one



(a)

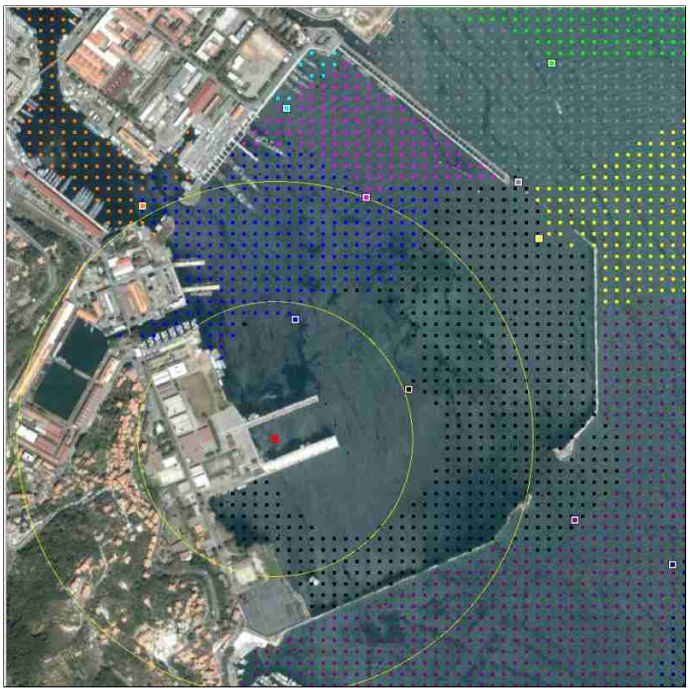


(b)

Figure 2.21: Comparison by changing the detection distance r (a) $r = 400$, $d_{worst} = 214$, $t_{worst} = 22.93$, (b) $r = 750$, $d_{worst} = 373$, $t_{worst} = 22.62$

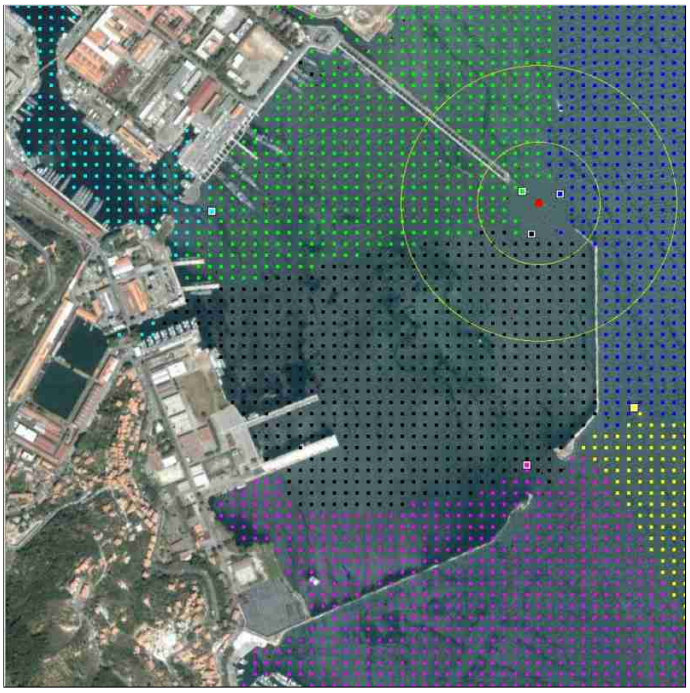


(a)

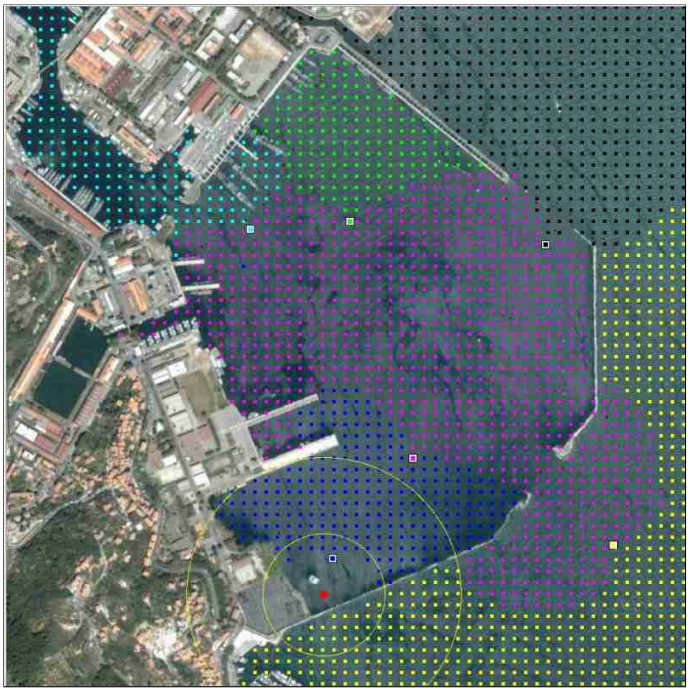


(b)

Figure 2.22: Comparison by changing the minimum required interception distance d_{th} (a) $d_{th} = 178$, $d_{worst} = 374$, $t_{worst} = 20.64$, (b) $d_{th} = 400$, $d_{worst} = 460$, $t_{worst} = 20$

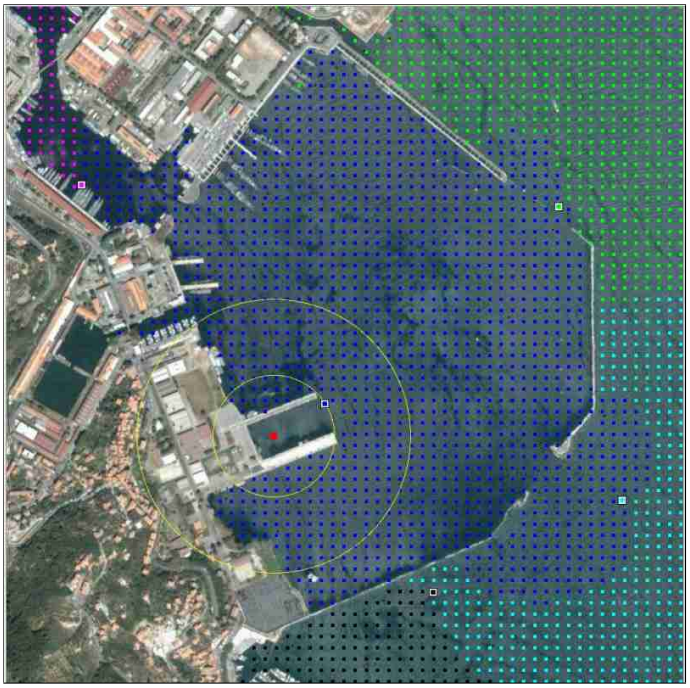


(a)

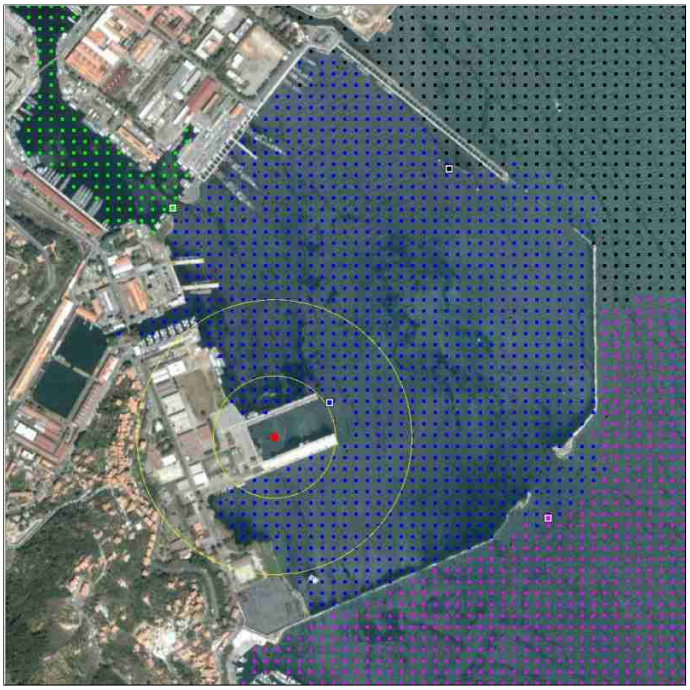


(b)

Figure 2.23: Comparison by changing the position of the asset (a) if the asset is inside the inner harbor area only one vehicle is necessary for the first criterion (b) when the asset is near the open sea, three vehicles are necessary to guarantee the same distance d_{th}



(a)



(b)

Figure 2.24: Comparison by changing the number of USVs (a) 5 USVs (b) 4 USVs

2.4.5 Multi Asset Problem

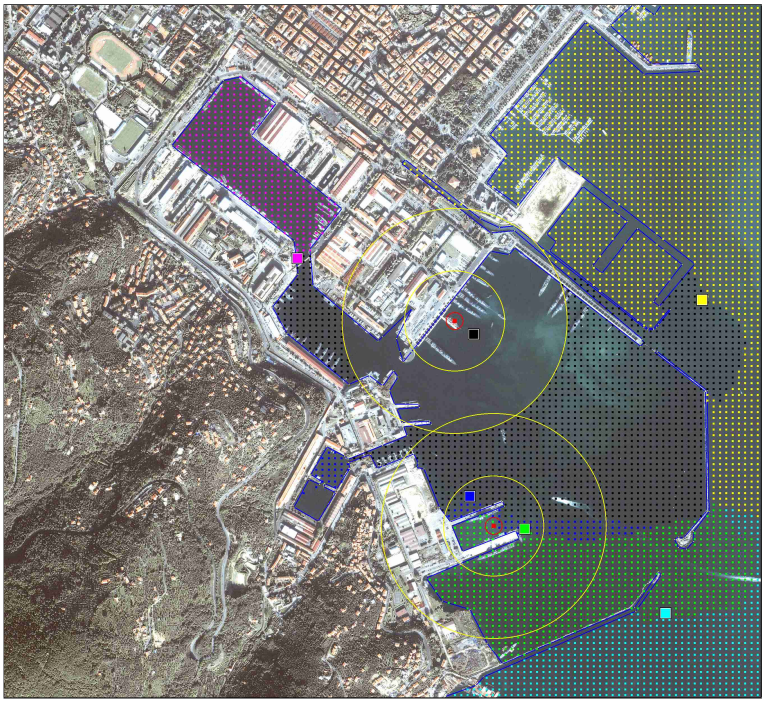
Preliminary results have been obtained also for the multi-asset scenario. In this case, (2.22) can be extended to consider more than one asset by evaluating the interception distance towards each asset $j = 1, \ldots, A$ and selecting the worst case:

$$D_w(P) \triangleq \min_j \min_{P_m|P} D_j(P_m, P) \tag{2.33}$$

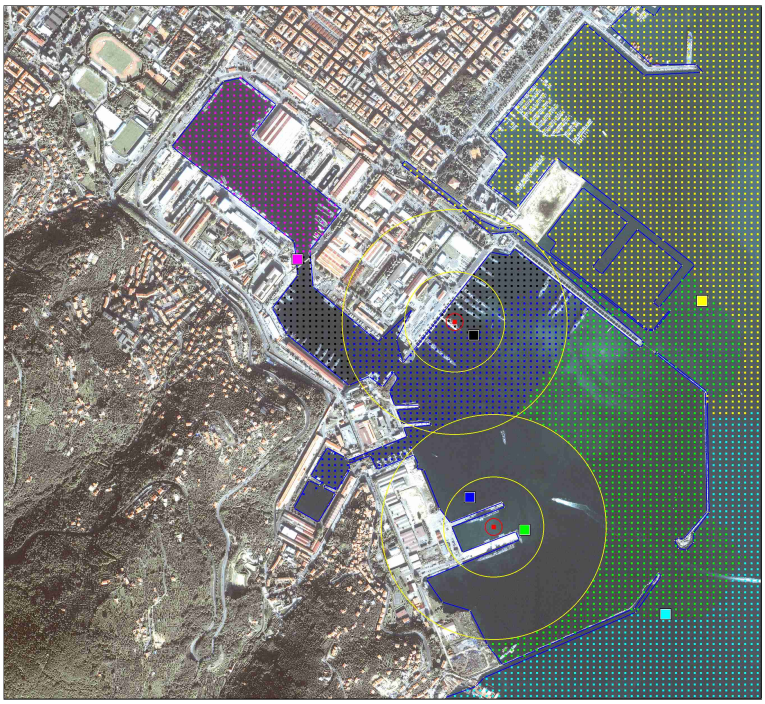
The same reasoning can be applied to the second criteria, modifying (2.30) to obtain

$$T_w(P) \triangleq \max_j \max_{P_m|P} W(P_m) T_j(P_m, P) \tag{2.34}$$

Figure 2.25 shows the optimization results obtained with two assets. In such case, the best interceptor obviously depends on which of the two assets is assumed as possible target.



(a)



(b)

Figure 2.25: Comparison by changing the possible target (a) northern asset (b) southern asset

2.5 Prototype Development

To test all the aforementioned algorithms, it has been decided to develop a small fleet of cheap USVs. These USVs are small replicas of the actual USVs, with limited capabilities. Their purpose is to test the avoidance and intercept algorithms. Thus it has been decided to buy radio controlled (RC) boats, and to modify them in order to make them autonomous. To do so, a control board with the necessary intelligence and control capability was designed and is presented in this section.

The field tests would thus use a base station, running the SMU, and a fleet of USVs. In lieu of the radar, the USVs themselves compute their global position with the aid of inertial sensors and a GPS, and send this information back to the base station once per second, thus simulating the presence of the radar.

Other than the inertial sensors and a GPS, each USVs is only equipped with the control board, the motors and a modem for communicating with the base station. The USV can only execute simple commands like a waypoint based navigation. This is however sufficient in order to test the algorithms of the SMU that have been presented in the previous sections.

Since the localization algorithms used inside the control board are actually independent of the marine environment, it has been decided to develop and test them using a RC car. This allowed to remove any risk since the car is remotely controlled, and greatly reducing the time for these preliminary field tests when compared to using a RC boat.

2.5.1 The General Architecture

The control is based on a single microcontroller, which performs all the calculations for both localization and the actual control of the motors. A tri-axis accelerometer and a tri-axis gyroscope provide inertial measurements, while a tri-axis magnetometer and a GPS module provide absolute measurement about the positioning. The USV can communicate to a base station through a radio frequency (RF) communication channel provided by a RF modem. Figure 2.26 shows a diagram of such architecture.

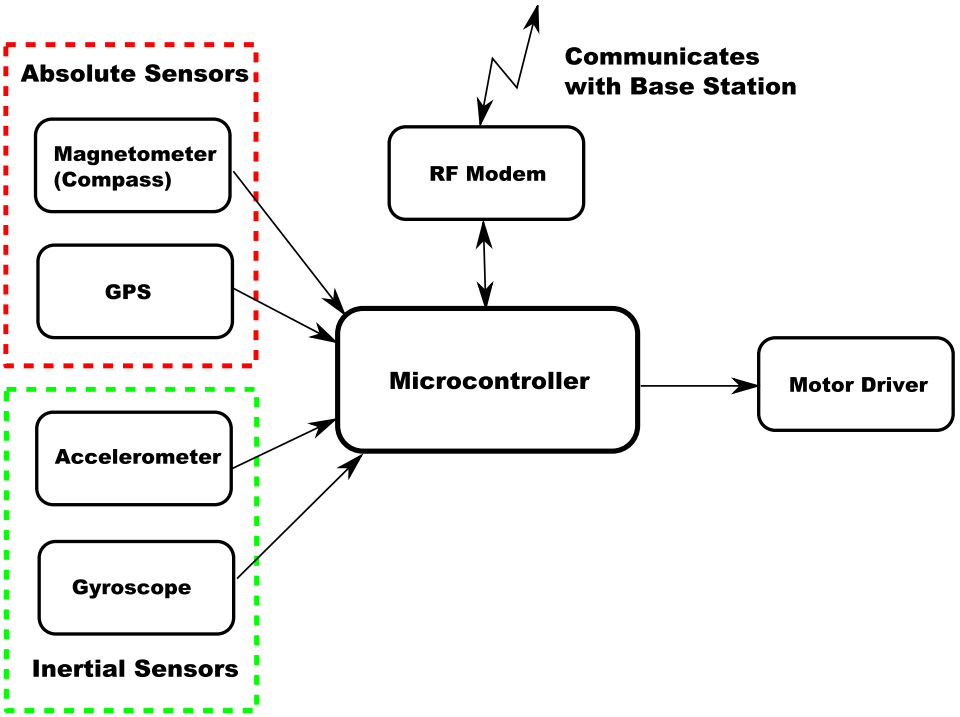


Figure 2.26: The Architecture of the Prototype.

2.5.2 First Prototype Design and Sensors

Following the aforementioned general architecture, the first prototype comprises the following components:

microcontroller: a dsPIC 30F4011, a microcontroller belonging to the motor control family produced by Microchip. It runs at a frequency of about 117.92 MHz providing enough computational power to run all the required filters and control algorithms.

accelerometer: ADXL345, a three axis digital accelerometer produced by Analog Devices. It can communicate both using the SPI or the I^2C digital buses.

gyroscope: the angular velocity information is provided by two separate ICs: the IDG-1215 provides velocity about the x and y axis, with a maximum of 67°/s. Information about the angular rate of the z-axis is instead provided by ADXR613 from Analog Devices, with a maximum measured velocity of 150°/s. Both of them provide their output as an analog signal.

magnetometer: the HMC6343 is a three-axis digital magnetometer from Honeywell, and provides measurements for Roll, Pitch and Yaw angles. It can also output the raw measurements of the magnetic field.

gps: the GPS module is the EM-406A, based on the SirfStar3 chipset.

modem: the modem is a 2.4GHz RF modem, with a baud rate of 19200bps from Digi.

2.5.3 Sensors Equations

In this section, the sensor equations are presented and these conventions are adopted:

- a barred variable will indicate the true value of that variable: so $\bar{\psi}$ indicates the true yaw angle.
- a hatted variable will indicate an estimate of the true value of the variable: so $\hat{x}$ is an estimate of the x position.
- a subscript k will indicate the time.
- a subscript j can indicate the different components of a vector.
- a superscript on the right hand side will indicate from which sensor that variable comes from, so ψ^c is the yaw measured by the compass
- the x, y plane has the x-axis pointed towards East, y axis pointed toward North, z axis exiting from the Earth.

Accelerometers

The accelerometers measure the vehicle's acceleration projected on the body frame $\langle u \rangle$

$$
\begin{aligned}
{}^{u}a_x &= {}^{u}\bar{a}_x + \eta_x && (2.35)\\
{}^{u}a_y &= {}^{u}\bar{a}_y + \eta_y && (2.36)\\
{}^{u}a_z &= {}^{u}\bar{a}_z + \eta_z && (2.37)
\end{aligned}
$$

These measures can be projected to the inertial frame $\langle 0 \rangle$ by a simple rotation

$$
{}^{0}a = {}^{0}_{u}R\left(\phi, \theta, \psi\right) {}^{u}a \tag{2.38}
$$

where ϕ, θ, ψ are the roll, pitch, yaw angles (x, y, z).

Gyroscopes

The gyroscopes measure the vehicle's angular velocity on the body frame

$$p = \omega_x = \bar{\omega}_x + \xi_x \tag{2.39}$$

$$q = \omega_y = \bar{\omega}_y + \xi_y \tag{2.40}$$

$$r = \omega_z = \bar{\omega}_z + \xi_z \tag{2.41}$$

They can be related to the rates of change of the roll, pitch, yaw angles in the following way

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.42}$$

Compass

The compass measure the orientation of the vehicle with respect to an inertial frame

$$\phi^c = \bar{\phi} + \eta_x \tag{2.43}$$

$$\theta^c = \bar{\theta} + \eta_y \tag{2.44}$$

$$\psi^c = \bar{\psi} + \eta_z \tag{2.45}$$

However, the first two components, roll and pitch, are measured with the aid of an internal accelerometer.

GPS

The GPS modules estimates the position of the vehicle in (lat,long) coordinates. It also provides, indirectly, an estimate of the velocity over ground. Moreover, for nonholonomic vehicles, it can also provide the orientation to the true north.

In order to exploit the GPS data, it has to be projected to the x, y plane. In order to do so, the following transformation was used:

$$x^g = (R_n \cos(\text{lat}_0))(\text{long} - \text{long}_0) \tag{2.46}$$

$$y^g = R_m(\text{lat} - \text{lat}_0) \tag{2.47}$$

where the radii R_m and R_n have been calculated on the basis of the reference location indicated by $(\text{lat}_0, \text{long}_0)$ as

$$R_n = \frac{a}{\sqrt{1 - e^2 \sin(\text{lat}_0)^2}} \tag{2.48}$$

$$R_m = R_n \frac{(1 - e^2)}{1 - e^2 \sin(\text{lat}_0)^2} \tag{2.49}$$

where a is the semi-major axis (equatorial axis), b is the semi-minor axis (polar radius) of the ellipsoid, and $e = \sqrt{1 - \frac{b^2}{a^2}}$. The values from the WSG84 standard have been used.

The measured orientation $\tilde{\psi}^g$ refers to the true north, which in our case coincides with the magnetic north. However, such orientation increases clockwise, starting from 0 when pointing north. Since the x, y plane has the x axis towards East, and the positive rotations are counterclockwise, the relationship between the measured heading and the heading translated in the x, y plane is the following

$$\psi^g = -\tilde{\psi}^g + \frac{\pi}{2} \tag{2.50}$$

By now knowing the angle, it is possible to project the estimated speed in its two components

$$\dot{x}^g = v^g \cos(\psi^g) \tag{2.51}$$

$$\dot{y}^g = v^g \sin(\psi^g) \tag{2.52}$$

where v^g is the estimated course over ground given by the GPS module.

2.5.4 Preliminary Field Tests

The first complete field tests were carried out in an open space area, where the GPS coverage was optimal. However, the SBAS (Satellite Based Augmentation System) signal was not properly locked, so DGPS corrections were not applied. The algorithm for the positioning estimation were based on a Kalman Filter with an holonomic model, with the GPS position only as measure.

As a first test, the vehicle was moved around following approximately a square. The vehicle sent to the base station raw sensors data at a rate of 10Hz, all the Kalman estimates and all the GPS data. The actual GPS data and Kalman estimate can be seen in Fig. 2.27.



Figure 2.27: Green: GPS position; Red: Kalman estimated position.

As it can be seen, the prediction between one GPS position and the successive one is wrong. Analyzing the raw sensor data the following problems were underlined:

1. the compass estimate of pitch and roll is almost unusable when moving;
2. the compass estimate of yaw is also pretty noisy when moving;
3. the board is subject to heavy vibrations that are captured by the accelerometer

In a simpler test, where the vehicle was commanded to move along a straight line and then turn back, the following values for the pitch and roll were measured, as shown in Fig. 2.28.

Such estimates are wrong, since the vehicle did not have such high degrees of tilting while moving. The terrain was indeed not flat, but still these numbers are way off. With such wrong estimates for pitch and roll, the projection of the accelerations was wrong as well, leading to bad estimates of the vehicle's velocity.

As for what concerns the yaw estimate, the following two figures show the difference between the heading measured by the compass (Fig. 2.29), and the

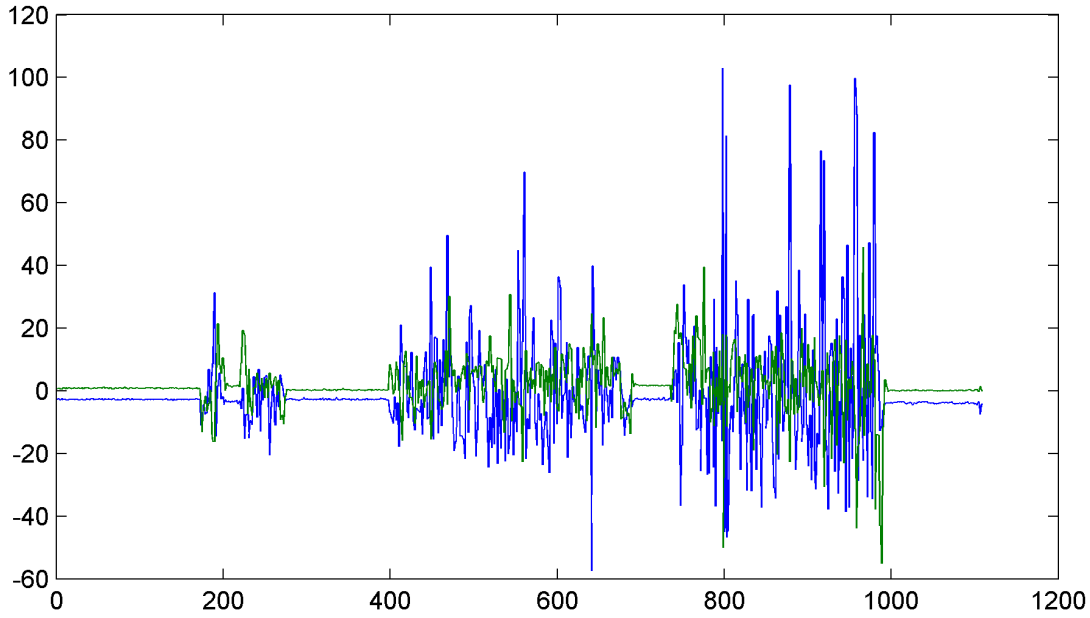


Figure 2.28: Estimates from the compass. Blue: Pitch; green: Roll.

one from the GPS module (Fig. 2.30). In this case, while still noisy, the measure from the compass is still somewhat in accordance with the measure from the GPS, which is however far more stable. The latter has problems only when the vehicle is stationary.

To confirm some of the above hypotheses, some tests have been carried out in the lab. The first series of tests consisted in the vehicle being lifted from the floor, and commanded to accelerate. The expected measured acceleration should have been zero, but due to the vibrations accelerations up to $40m/s^2$ were measured. Fig. 2.31 shows the resulting accelerations. Some backlash was present between the breadboard and the vehicle and was eliminated, reducing a bit the vibrations.

Also, the output from the compass was measured and was supposed to be constant. Fig. 2.32 shows that this was not the case, and that the compass output could not be trusted while moving, especially for the pitch and roll components. This is due to the fact that the compass, internally, exploits an accelerometer to calculate the tilt of the sensor and correct the magnetometer readings.

2.5.5 Position Estimation Algorithms

A series of tests were performed aimed at finding the most suitable filters for the estimation of linear and angular position. The final algorithms are here reported

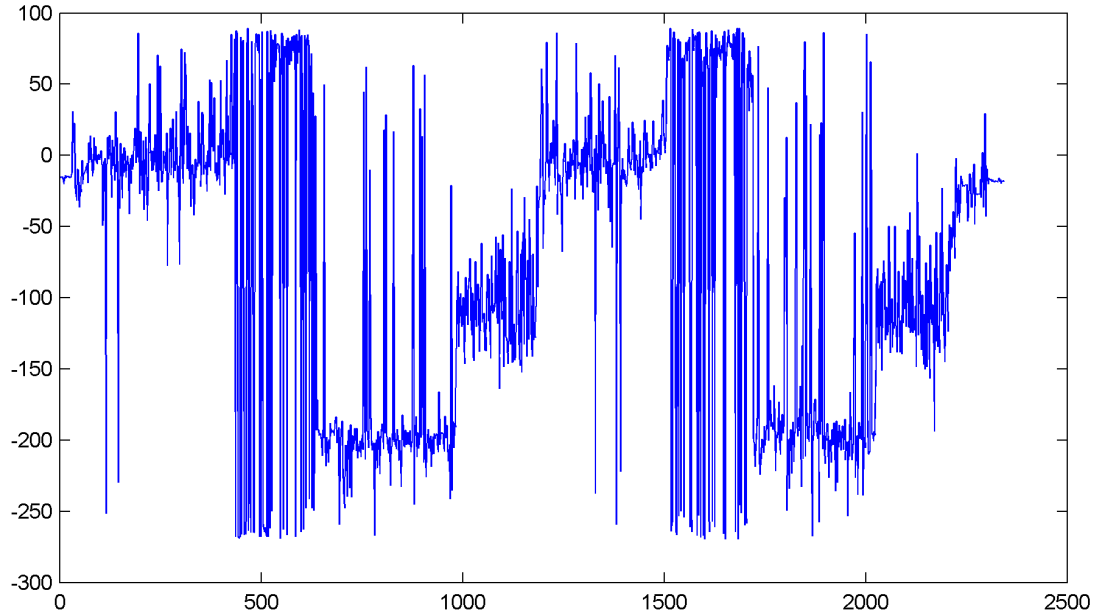


Figure 2.29: Yaw measured by the compass while performing an approximated square trajectory.

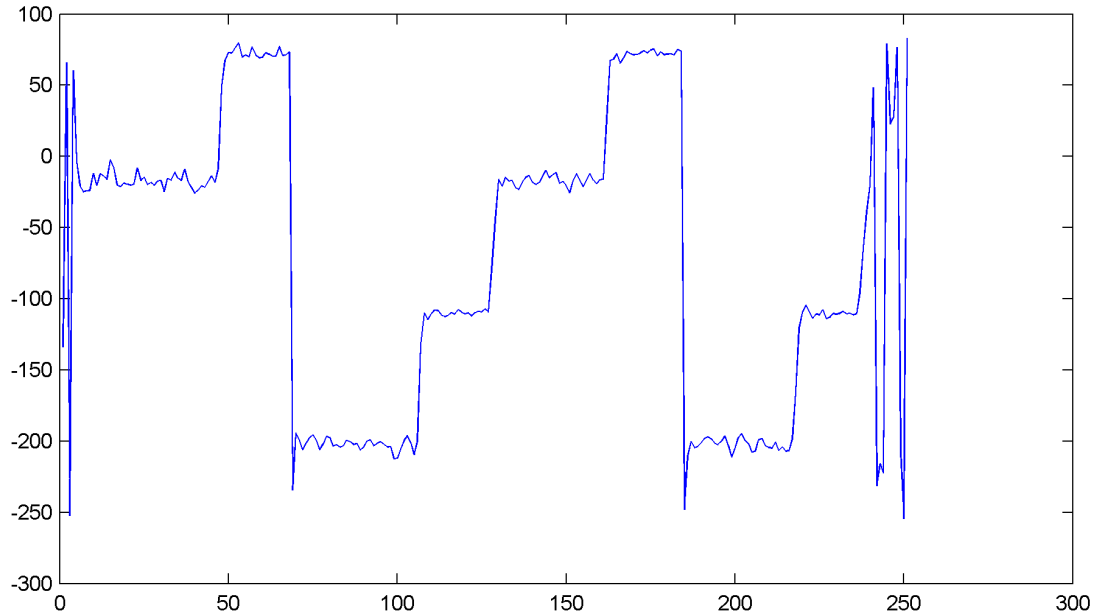


Figure 2.30: Yaw measured by the GPS module while performing an approximated square trajectory.

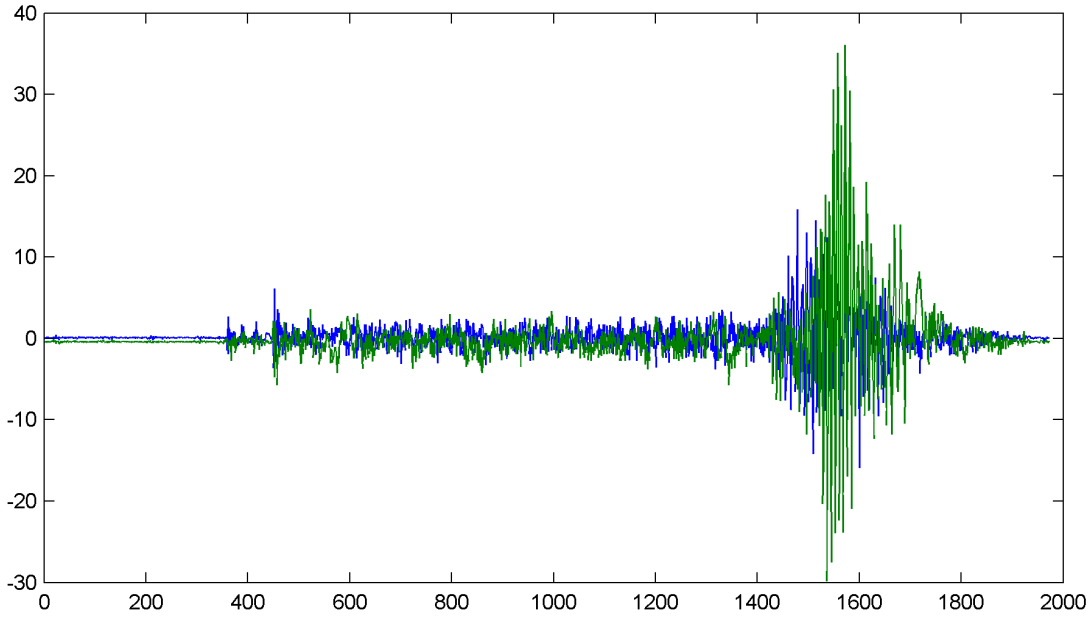


Figure 2.31: Accelerations measured while the vehicle was not moving.

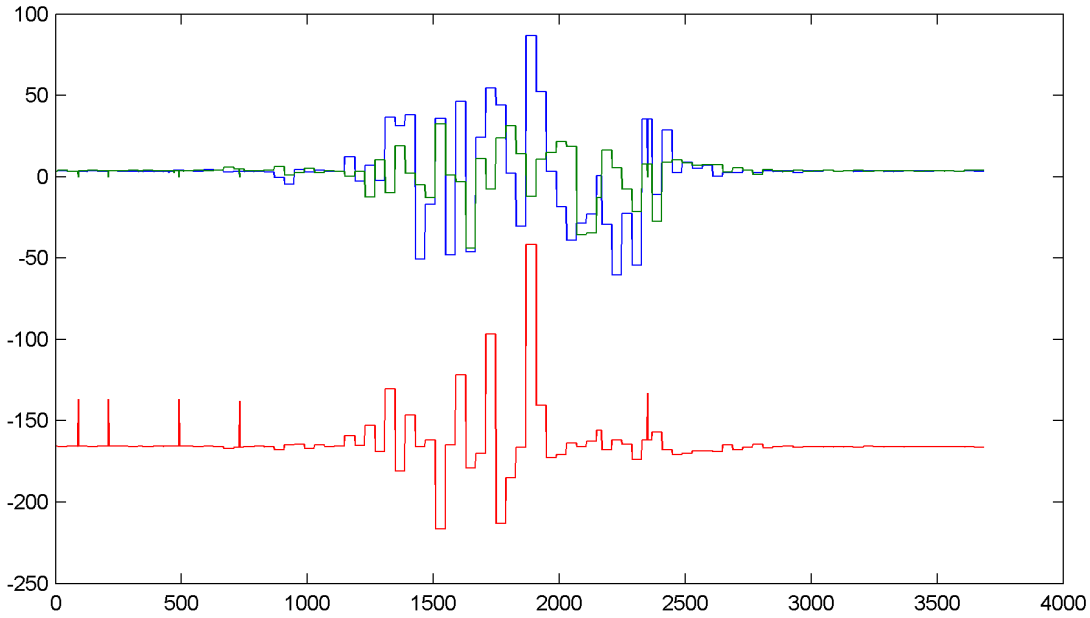


Figure 2.32: Estimates from the compass. Blue: Pitch; green: Roll.

in the following sections. Note that, due to the heavy memory limitations (only 2k bytes of memory available) and computational power of the dsPIC, only very basic filters could be implemented on the microcontroller. This explains why the angular and linear part are kept separate, while they are actually coupled.

Estimation Algorithm for the Linear part

The final estimation algorithm for the linear part exploits a non-holonomic model of the vehicle and a Kalman filter to estimate the x, y position and the vehicle's velocity along the longitudinal axis. In particular, the following model was used:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t \cos(\psi) \\ 0 & 1 & \Delta t \sin(\psi) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix} {}^{v}a_x \tag{2.53}$$

where ${}^{v}a_x$ is the acceleration along the longitudinal axis of the vehicle, after vehicle frame $\langle u \rangle$ has been rotated of the actual pitch and roll angles.

The measure equation exploits the GPS position and velocity:

$$y_k = \begin{bmatrix} x_k^g \\ y_k^g \\ v_k^g \end{bmatrix} = x_k + \eta_k \tag{2.54}$$

Estimation Algorithm for the Angular part

First, let us consider the equation linking the angular velocity of the body frame with the Euler angles rates:

$${}^{0}\omega = {}^{0}k_0\dot{\psi} + {}^{0}y_1\dot{\theta} + {}^{0}i_2\dot{\phi} = {}^{0}S\left(\phi, \theta, \psi\right) \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.55}$$

where

$${}^{0}S\left(\phi, \theta, \psi\right) = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi) & 0 \\ \sin(\psi)\cos(\theta) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \tag{2.56}$$

To obtain the Euler rates, the matrix ${}^{0}S\left(\phi, \theta, \psi\right)$ must be inverted

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = {}^{0}S^{-1}\left(\phi, \theta, \psi\right) {}^{0}\omega \tag{2.57}$$

Since the angular velocity is expressed on the body frame, it has to be projected onto the inertial frame

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = {}^{0}S^{-1}(\phi, \theta, \psi) \, {}^{0}_{b}R \, {}^{b}\omega \tag{2.58}$$

which produces the relationship (2.42), here reported again

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

By putting $x = [\phi \; \theta \; \psi]^T$, the previous equation can be seen as a nonlinear dynamic system such $\dot{x} = f(x, u)$.

In order to obtain a Kalman Filter, the accelerometer readings can be exploited as a measure of the system in the following way. In a stationary situation, the accelerometer reads the gravity vector projected onto the body frame

$${}^{b}a = -{}^{b}g = -{}^{b}_{0}R \, {}^{0}g \tag{2.59}$$

By exploiting the fact that the gravity vector is ${}^{0}g = [0 \; 0 \; G]^T$, the previous relation can be expressed as

$${}^{b}a = -{}^{b}g = -G \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.60}$$

which represent a nonlinear measure channel of the type $y = h(x)$.

In a non stationary situation, the previous equations must be extended to also consider the centripetal forces and accelerations along the body axes:

$$\begin{bmatrix} {}^{b}a_x \\ {}^{b}a_y \\ {}^{b}a_z \end{bmatrix} = \begin{bmatrix} G\sin(\theta) + qw - rv + \dot{u} \\ -G\cos(\theta)\sin(\phi) + ru - pw + \dot{v} \\ -G\cos(\theta)\cos(\phi) + pv - qu + \dot{w} \end{bmatrix} \tag{2.61}$$

where (u, v, w) are the velocities of the vehicle along the body frame axes (x, y, z). However, considering the vehicle moving only on the x, y plane, and in particular only in its longitudinal direction, neglecting lateral velocities and accelerations, the following assumptions can be done $\dot{v} = \dot{w} = v = w = 0$, thus obtaining

$$y = \begin{bmatrix} {}^b a_x \\ {}^b a_y \\ {}^b a_z \end{bmatrix} = \begin{bmatrix} G\sin(\theta) + \dot{u} \\ -G\cos(\theta)\sin(\phi) + ru \\ -G\cos(\theta)\cos(\phi) - qu \end{bmatrix} \tag{2.62}$$

Since $\dot{u}$ is unknown, it can be modeled as an external noise signal, leading to

$$y = \begin{bmatrix} {}^b a_x \\ {}^b a_y \\ {}^b a_z \end{bmatrix} = \begin{bmatrix} G\sin(\theta) \\ -G\cos(\theta)\sin(\phi) + ru \\ -G\cos(\theta)\cos(\phi) - qu \end{bmatrix} + \eta = h(x) + \eta \tag{2.63}$$

which represents a nonlinear measure channel of the type $y = h(x) + \eta$.

With all these information an Extended Kalman Filter can be employed in order to estimate the roll and pitch angles. First, both the nonlinear system and measure channel must be linearized, by computing the Jacobian of the measure channel:

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} 0 & -G\cos(\theta) & 0 \\ G\cos(\theta)\cos(\phi) & -G\sin(\theta)\sin(\phi) & 0 \\ -G\cos(\theta)\sin(\phi) & -G\sin(\theta)\cos(\phi) & 0 \end{bmatrix} \tag{2.64}$$

which indicates that the yaw angle cannot be estimated from the acceleration measurements.

Thus, the filter only estimates $x = [\phi\ \theta]^T$ and the previous equation is reduced to

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} 0 & -G\cos(\theta) \\ G\cos(\theta)\cos(\phi) & -G\sin(\theta)\sin(\phi) \\ -G\cos(\theta)\sin(\phi) & -G\sin(\theta)\cos(\phi) \end{bmatrix} \tag{2.65}$$

Proceeding with the linearizations, the Jacobian of the system can be found as

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\sin(\theta)(\cos(\phi)q - \sin(\phi)r)}{\cos(\theta)} & \frac{(\sin(\phi)q + \cos(\phi)r)}{\cos(\theta)^2} \\ -\sin(\phi)q - \cos(\phi)r & 0 \end{bmatrix} \tag{2.66}$$

With these calculations done, the EKF can be implemented. The prediction is carried out by explicitly using the nonlinear model

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \tag{2.67}$$

and the predicted estimate covariance is calculated, using the linearized model, as

$$S_{k|k-1} = F_k S_{k-1|k-1} F_k^T + Q_k \tag{2.68}$$

where Q_k was the covariance matrix of the plant noise.

Then, whenever a new measure is read by the accelerometers, the update part of the Kalman filter takes place. First the Kalman gain is calculated as

$$K_k = S_{k|k-1} H_k^T \left(H_k S_{k|k-1} H_k^T + R_k \right)^{-1} \tag{2.69}$$

where R_k is the covariance matrix of the accelerometer measure. Then, the estimate is updated

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(y_k - h(\hat{x}_{k|k-1}) g \right) \tag{2.70}$$

and the estimated covariance is also updated

$$S_{k|k} = \left(I - K_k H_k \right) S_{k|k-1} \tag{2.71}$$

2.5.6 Field Tests of the Localization Algorithms

In order to tune the parameters of the filters correctly, a series of tests were conducted and the raw data from the sensors was sent to the base station and collected. The filters were then run on the data gathered, so that multiple tests could be done with the same data. Unfortunately, due to limited bandwidth, only 1/5th of the raw data could be sent, and sometime some packets were lost in the transmission. So the algorithms implemented directly on the vehicles are expected to perform better due to more data being available.

The first test shows a simple rectangular like trajectory. The results of the estimation (red dots) are reported in Fig. 2.33 and are pretty close the GPS position (green dots). Due to some missing packets, sometime the algorithm exploits the last data available to perform the estimation. This is shown by blue dots on the picture. This phenomena will not be present when the algorithms will be run directly on the dsPIC micro controller, since they are due to the fact that the raw data has been sent to a remote base station via a wireless link.

In the second test, the vehicle is moved along a circumference with approximately a constant speed. The results of estimation are shown in Fig. 2.34, where the Kalman estimate is compared with the GPS position. Figure 2.35 shows the yaw angle estimate compared to the GPS heading and the compass yaw angle. As it is clear, the GPS heading is delayed with respect to the Kalman filter estimate and cannot be used when the vehicle is stationary (at the start of the

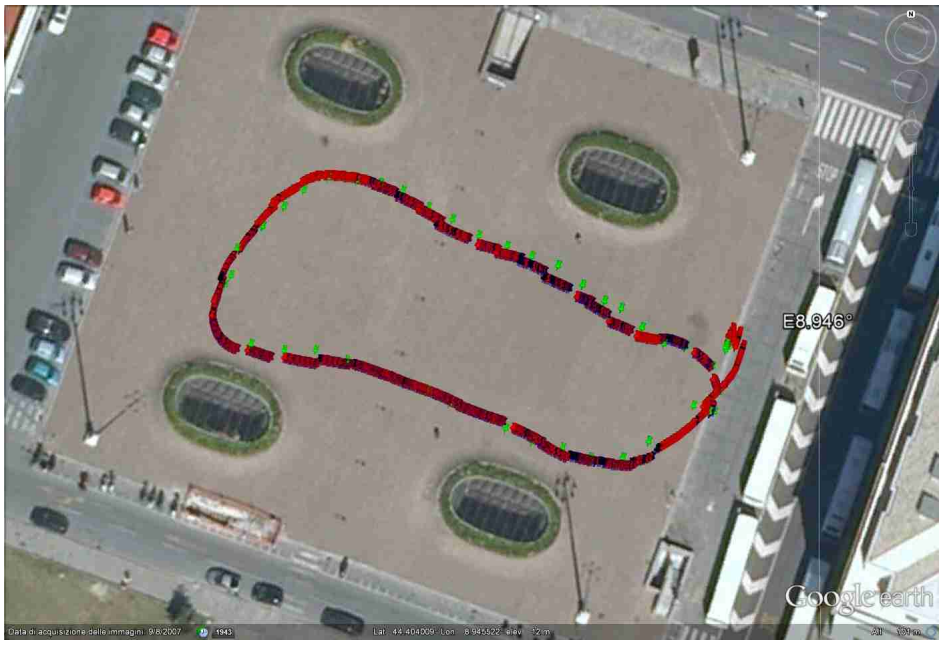


Figure 2.33: Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing

graph). Conversely, the compass reports an accurate value when the vehicle is not moving, but is noisy otherwise. This, as underlined before, occurs because the compass is using an internal accelerometer to compensate the tilt and thus is subject to the very high accelerations and vibrations that are picked up by the device when moving on the terrain.

In the successive trial, shown in Fig. 2.36, the vehicle is commanded to follow a complex trajectory, with different curves, in order to test the effects of frequent accelerations and decelerations. As it can be seen from the picture, the estimated position follows the GPS one quite well, even though it is suffering from the fact that different data packets were lost in the transmission.

The next test is instead aimed at testing the estimate of the pitch and roll angles, which are the most difficult to be estimated, since no absolute measure of these quantities is available. The vehicle is moved along a slope and the position estimates are reported in Fig. 2.37. The Kalman filter pitch and roll angles estimates are instead reported in Fig. 2.38. As it can be seen, the filter reports a quasi constant roll angle of around 30 degrees.

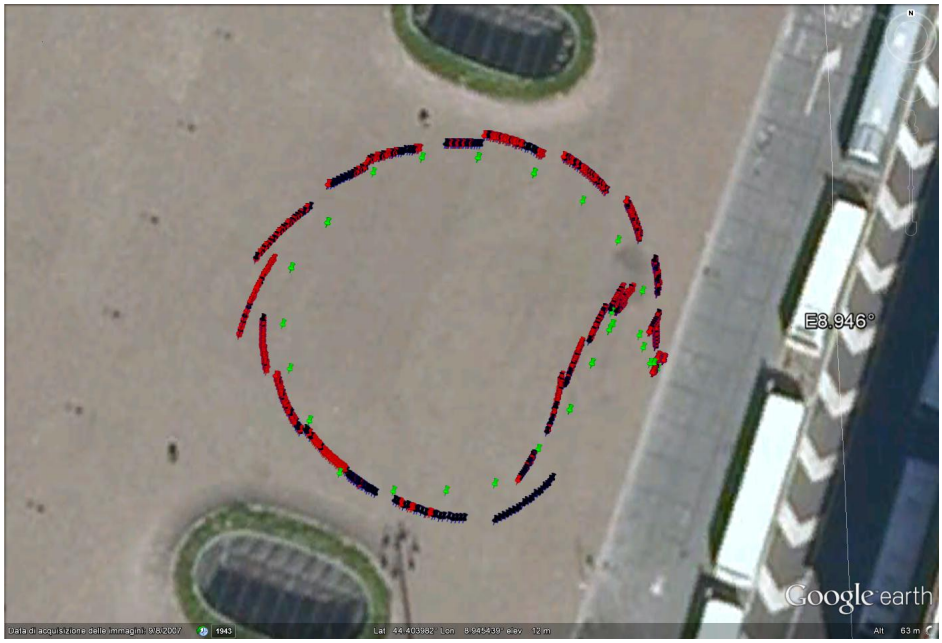


Figure 2.34: Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing

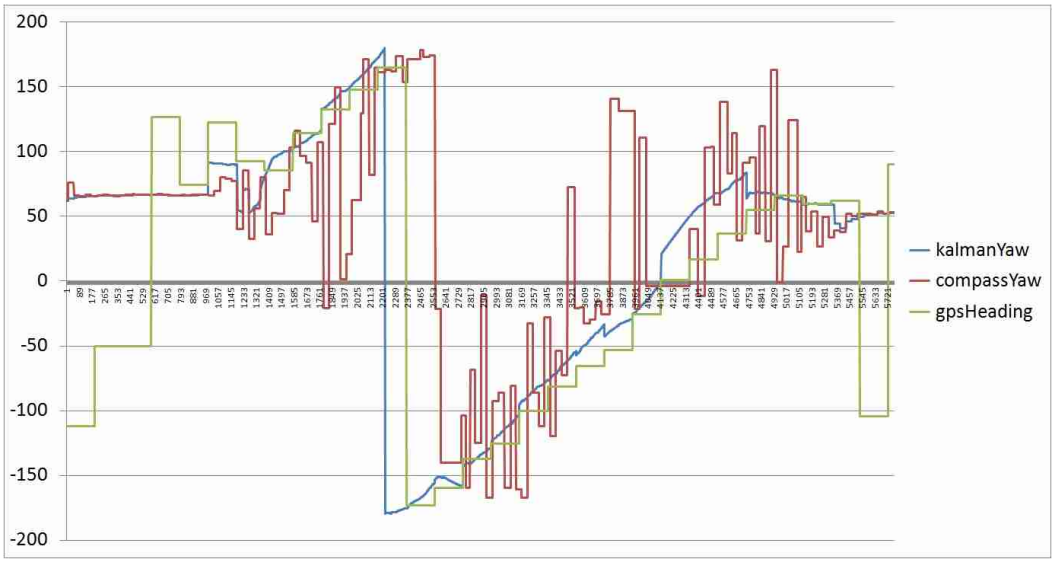


Figure 2.35: Blue: Kalman filter yaw estimate. Red: compass yaw value. Green: GPS heading

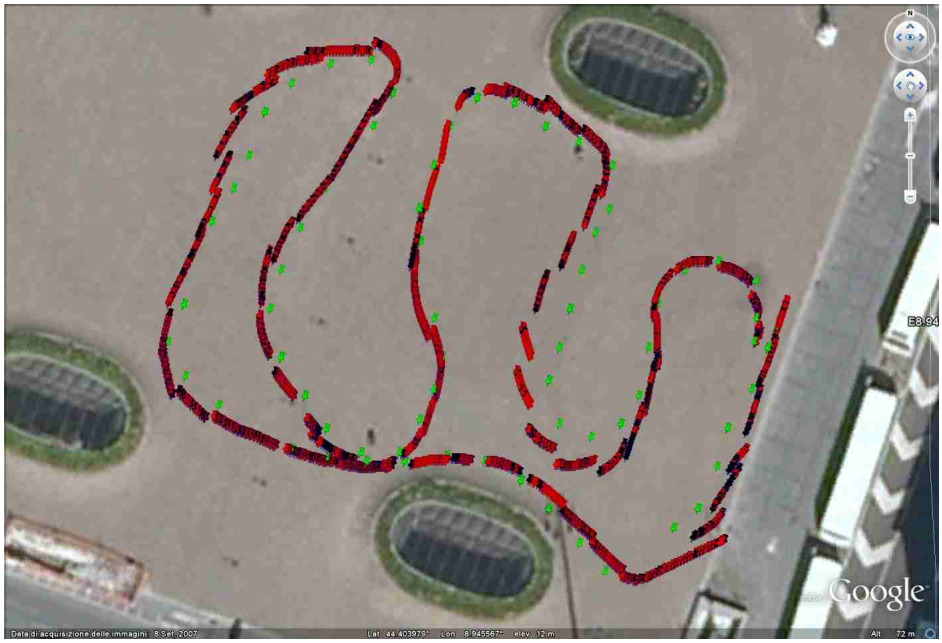


Figure 2.36: Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing



Figure 2.37: Green: GPS position; Red: Kalman estimated position. Blue: Kalman estimated position when raw data was missing

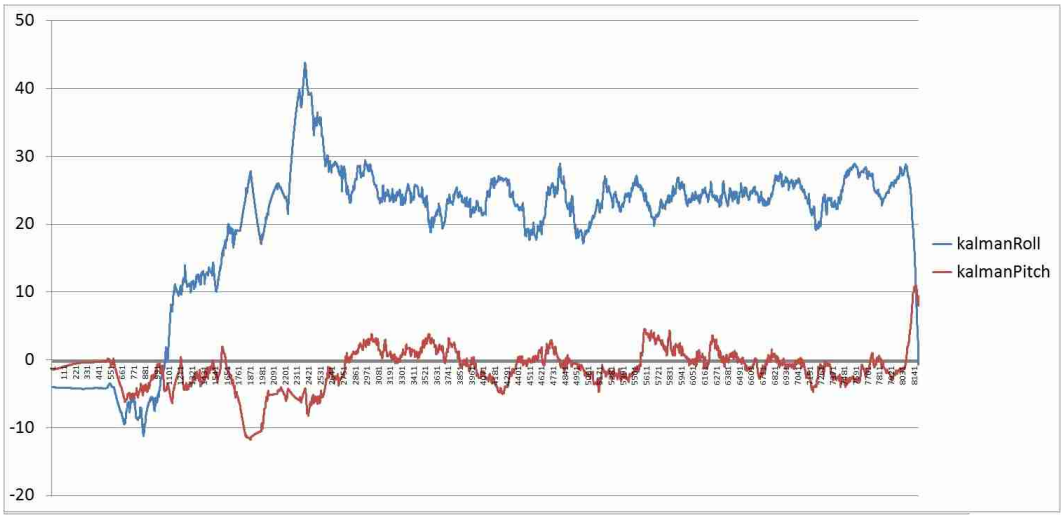


Figure 2.38: Blue: Kalman filter roll angle estimate. Red: Kalman filter pitch angle estimate

2.5.7 Final Prototype Design

After the field tests some adjustments have been made to the control board. The gyroscope IDG-1215, which provides velocity about the x and y axis, has been replaced with the IDG-500, which has a maximum value of 500°/s compared to the previous 67°/s which were insufficient. Moreover, due to the fact that the inertial sensor are very low cost and noisy, the GPS module has been replaced with the OEMStar from Novatel, which provides the GPS position with a 10Hz rate, instead of 1Hz of the previous module. Moreover, the boundaries of the estimate of the OEMStar are lower. This should greatly increase the expected performance of the localization algorithms.

Figure 2.39 shows the schematic of the dsPIC micro controller and the main connectors of the board. In the schematic are present a reset button, a calibration switch for the HMC6343 compass, the RJ11 connector which is necessary for programming the dsPIC, and a switch in order to disconnect the I^2C bus when programming the dsPIC. Figure 2.40 shows the power management part, where the battery voltage is converted to the necessary 3V and 5V for the sensors with the use of two linear regulators. Figures 2.41, 2.42, 2.43 and 2.44 show the schematics of the sensors that are used by the dsPIC.

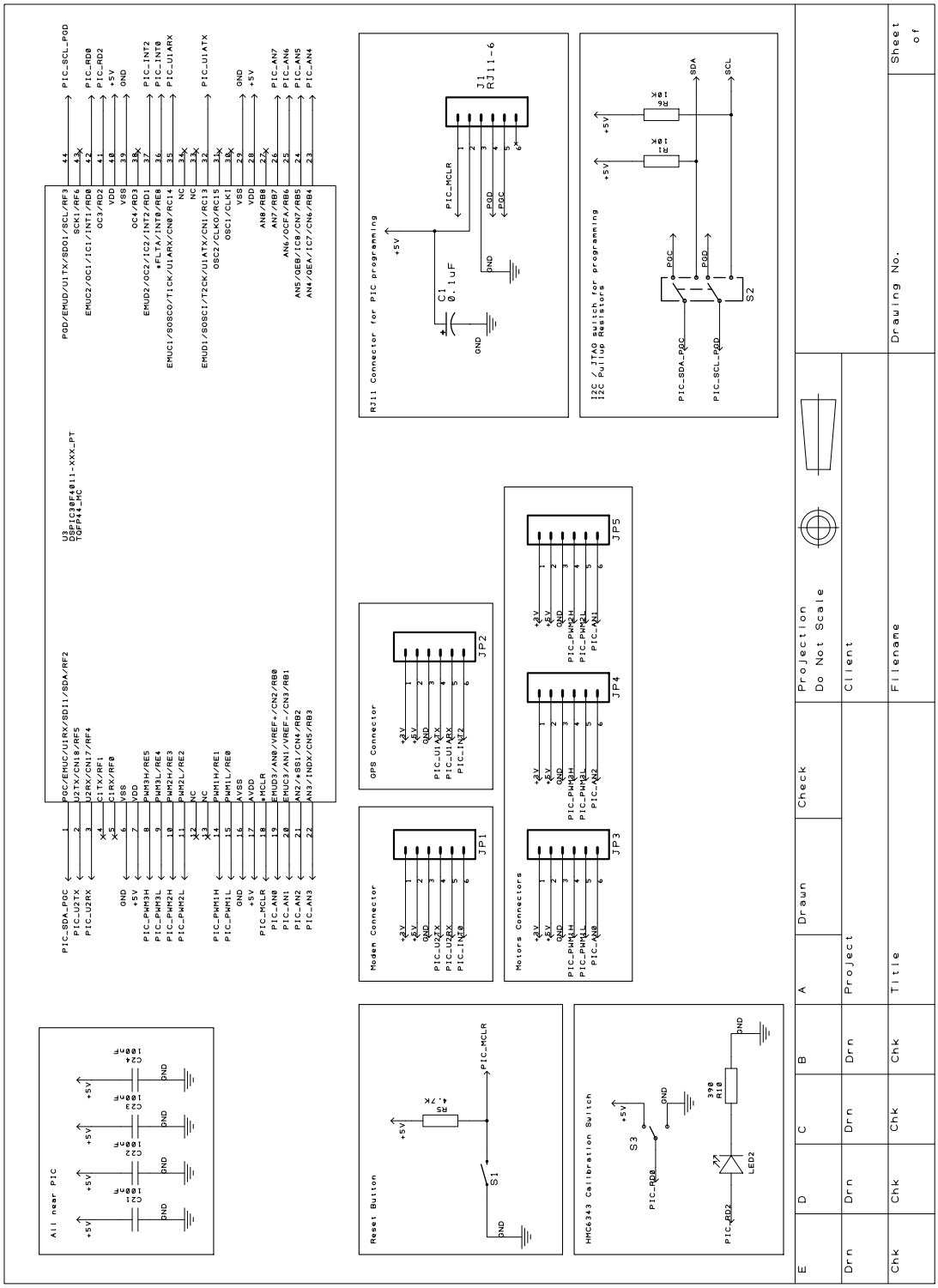


Figure 2.39: Schematic of the dsPIC and the connectors of the control board

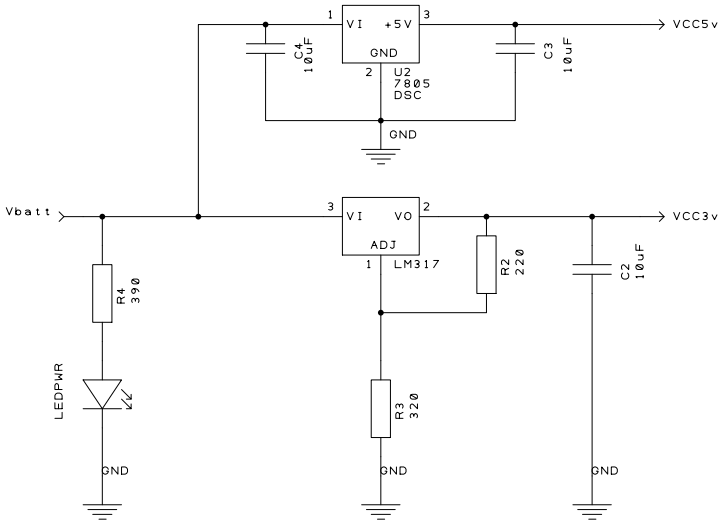


Figure 2.40: Schematic of the voltage regulation

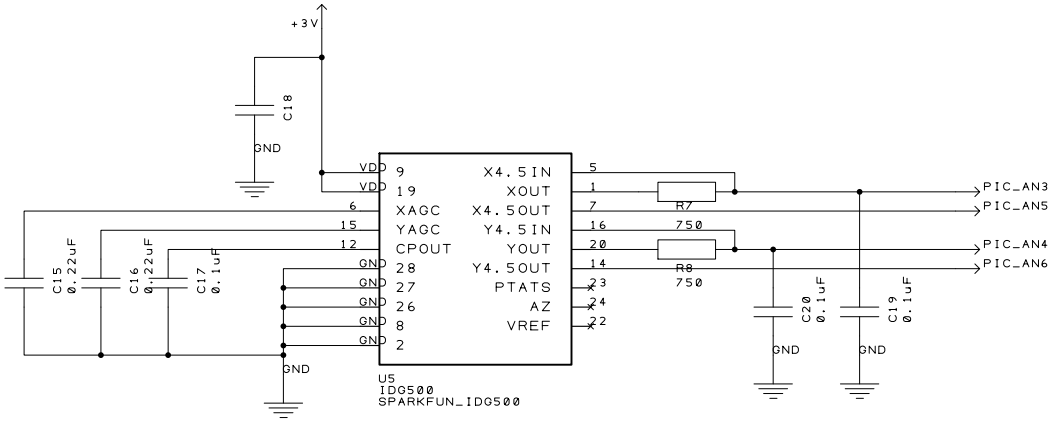


Figure 2.41: Schematic of the IDG 500 gyroscope sensor

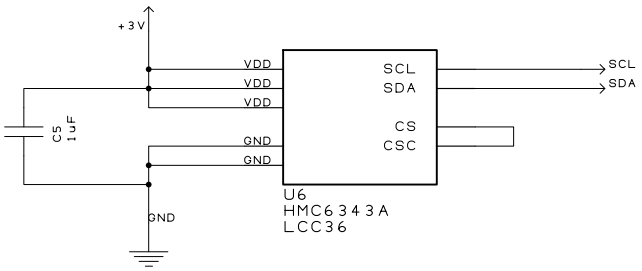


Figure 2.42: Schematic of the HMC6343 compass sensor

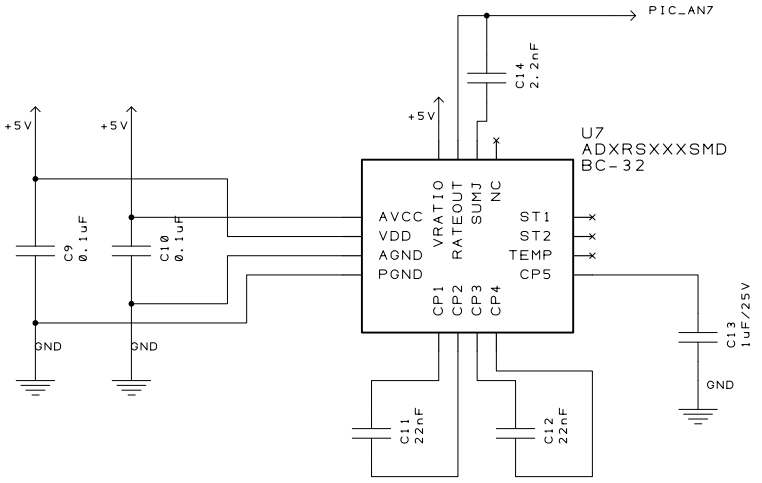


Figure 2.43: Schematic of the ADXRS613 gyroscope sensor

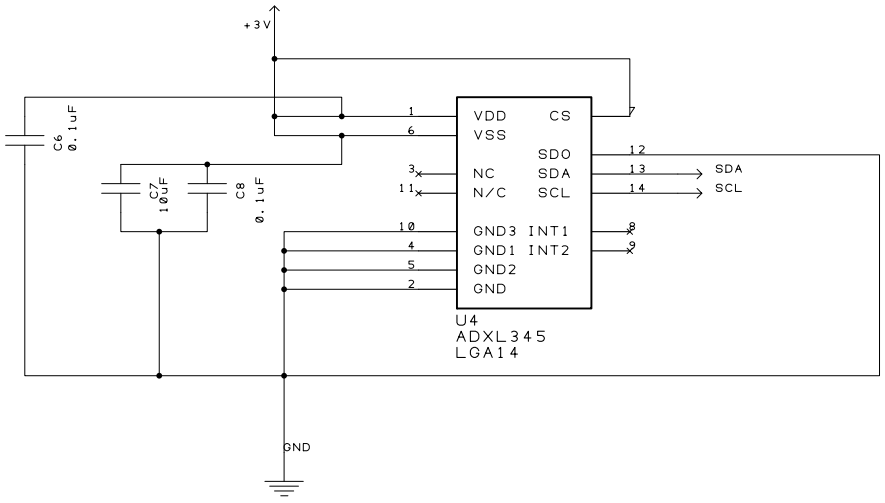


Figure 2.44: Schematic of the ADXL345 accelerometer sensor

Chapter 3

Autonomous Underwater Vehicles

Over the past few decades, the development of AUV technology has seen a steady increase, especially in the last one. This growth in the application of AUV is due to their improved performances and is fueled by the increasing number of missions made possible by the use of unmanned system. This also made new costumers aware of the potentialities of the AUVs and allowed more research funds to be available in order to further increase the capabilities of the unmanned underwater technology.

Although it started with a few players in the oceanographic, military and academic fields, today's AUV community is a robust group that includes these and several commercial players. And even if applications and technologies have reached a fair level of maturity, AUVs have not yet reached their full potential for utility and continue to be an active area of research.

3.1 Main Challenges and Applications of AUVs

3.1.1 Main AUVs Applications

Today's applications of AUVs mainly involve following pre-computed trajectories while carrying sensor payloads, trying to maximize the area covered. The typical AUV only needs to translate along segmented linear trajectories, which most of the time lie on the horizontal plane. Some exception may occur, for example in case the vehicle operates in shallow or deep water, when vertical motions are

required to follow the sea floor or to emerge in order to exchange data.

In the following paragraphs, the main applications of AUV will be detailed.

Hydrographic Survey

One of the main stimulus for the development of the AUV technology was certainly the field of oceanographic survey and mapping [45]. Indeed, the AUV mobility provides the opportunity to sample along predefined trajectories, a significant advantage over the use of moored or free drifting sensors. Manned ships with towed instruments also have this possibility, however the use of an AUV is still preferable due to its ability to do tighter turns, compared to the long turns of a ship with a long tow line. Also, and certainly not less important, the use of a manned ship implies very high costs, especially if the mission is long. On the contrary, the AUV requires less intervention by human beings, and in some applications only for the launch and recovery, greatly increasing the area covered for the same cost by a towed instrument [46].

Moreover, the AUV mobility enables it to accomplish missions that a towed or moored system cannot, like sampling in hard to access areas, as for example under the polar ice [47].

A special class of AUV, named Gliders, can operate for many days and are the ideal solution for extended surveys in the ocean. These AUV use wings and buoyancy changes in order to move while minimizing power consumption, in a similar manner to their aircraft counterparts. Their use has already been proven in long range scenarios (see for instance [48]).

Undersea Gas/Oil Production

One specific hydrographic application of AUV technology is undersea exploration and production of oil and gas. Undersea surveys to locate deposits have recently used AUVs in the North Sea and Gulf of Mexico respectively, with exceptional results in characterizing the bottom and sub-bottom [49]. The inherent mobility of AUVs in this application results in time savings of 60% or better when compared to less-maneuverable towed systems [50]. Savings increase with surveys in deeper waters, which are becoming increasing common as shallow water production fields are depleted. AUVs have also been used for undersea pipeline inspection once the field has been established. This trend in exploration and production points

to an increasing future usefulness of AUVs in this application.

Hull Inspection

A more recent application for AUV is the inspection of ship hulls. Indeed increased security about ships and harbors has made inspection of the ships and their cargo an important issue, and motivated new methods to carry out these tasks. The employment of AUV allows to keep the ship inside the water, instead of taking it outside, decreasing costs. However, the AUV now has to follow a three dimensional path, along the contours of the ship, which is quite more challenging that the usual survey applications. Successful examples can be found in both [51] and [52].

Military Applications

As most of the research in different areas, the major drivers for the development of AUV technology have been and currently are the military applications. AUVs, exploiting the coverage guaranteed by operating underwater, can carry out operations undetected. One of the major applications is the mine warfare, as the AUV can provide covert and rapid assessment of a minefield, while at the same time not risking any human beings. As proof of that, AUVs were employed in the Iraqi Operation Freedom in 2003 by the U.S. Navy in mine clearance operations in Umm Qasr Harbor [53].

Another important application related to the safety is the support to mine hunters and EOD (explosive ordinance disposal) divers, to counter terrorist mining of an harbor. Of particular interest is the capability to ship overnight small AUVs anywhere a crisis might occur and to place the appropriate sensors (sonar, optical, magnetic) in close proximity of mines without risking human lives. The limited cost of COTS AUV (compared with traditional MCM assets) allows to deploy fleets of specialized vehicles to achieve large area coverage [54].

3.1.2 Main Challenges and Critical Issues

The following paragraphs will underline the current state of AUV research, highlighting the open challenges and major critical issues for their further development.

Modular Design

A reconfigurable AUV is certainly advantageous for different reasons. As technology develops, the hardware and software tends to change and be upgraded, and that is easier and less time consuming to achieve with a modular design. Also, having one AUV capable of carrying different payloads depending on the mission at hand is cost effective, when compared to multi single-purpose vehicles or one big vehicle carrying all possible payloads.

As results of these observations, modularity is increasing in the AUV market. Examples of modular AUV include the Gavia AUV, the Bluefin 9, the REMUS AUV and the eFOLAGA AUV [55].

Localization and Navigation

As the underwater environment does not permit the GPS signal to reach the AUV, the vehicle must use alternative approaches to the localization problem. Two major approaches have arisen in the literature.

The first is based on the fact that acoustic energy propagates very well in the water. The vehicle interrogates acoustic transponders moored at known fixed positions, and on the basis of the time of flight of the acoustic signal, it computes the distance w.r.t. the transponders. After that, with a triangulation scheme, the position can be calculated. In other scenarios, an array of transponders is towed by a ship, and the array interrogates the AUV. By detecting the angle of arrival and its time, the ship can track the position of the AUV.

The second approach is based on surfacing to obtain a GPS fix. As surfacing means interrupting the mission, such fixes are very infrequent, and the AUV needs to navigate with dead-reckoning for the rest of the time. Thus, Inertial Navigation Systems (INS) are used to provide localization updates in between GPS fixes. However, as INS systems are generally subject to high bias, most of the time they are coupled with a DVL, in order to obtain an accurate estimation of the vehicle's velocity.

The first approach is certainly more accurate but requires the presence of acoustic transponders moored to the sea floor. Thus, it is indicated for mission in limited areas, whereas the second approach can be used also for oceanographic mission that span a very large area, as in the case of ocean gliders.

Communication

Another peculiar challenge of the underwater domain is certainly represented by the very limited bandwidth that is available for the communications. As the radio technology does not work underwater, AUV are limited to use acoustic communication, which has a bandwidth of few hundred bits per seconds, or to optical and laser communication, which provide higher bandwidth but for a limited range. In order to use the common radio communication, the AUV must go to the surface, interrupting the mission.

Due to these constraints, algorithms that involve cooperation between AUVs or even between the AUV and a supporting vessel must be conceived keeping in mind the limited bandwidth that characterizes the underwater domain.

Autonomy

The limitations of the bandwidth and difficulties in underwater communication place a great importance on the level of autonomy that the AUV should have. Indeed, since there are limited possibilities to obtain frequent telemetry and status updates, human intervention becomes far more difficult than in its aerial, ground or surface counterparts. Thus, an high level of autonomy becomes a key aspect in the development of AUV technology and algorithms.

However, nowadays the level of autonomy of the AUV is very limited. Indeed, the autonomy is confined to the low level control of the trajectory. The AUV usually just follows a pre-programmed set of waypoints, which is sometime changed during the mission via a command from the user, obtained at pre-defined time intervals for the communication. Thus the autonomy is confined in controlling the vehicle's speed, heading and position while going through the required waypoints.

A step towards autonomy has been carried out in the field of automatic obstacle avoidance. While obstacle avoidance is certainly not as critical and often occurring as in the case of surface vessels, it remains an important aspect even of AUVs, mostly to avoid damage to the vehicle itself. Avoidance is currently based on sensing the obstacle via the use of synthetic aperture sonars and side scan sonars, and has been demonstrated in [56, 57].

Recent efforts to improve autonomy have also been carried in order to create adaptive algorithms that behave and react to the environment, with the aim of

optimizing the mission at hand. For example, Caiti et al. [58, 59] proposed a collaborative adaptive algorithm for a team of AUV sampling an area of interest, trying to obtain more samples where the gradient of the quantity is higher, that is in the areas where the quantity of interest is rapidly changing and an higher resolution of data is required.

3.1.3 Proposed Algorithms

After having underlined what are the critical issues of the underwater domain and the main challenges of AUV development, the present work will focus on the key problem of underwater localization.

Two algorithms are presented for the underwater localization of the AUVs, by using a Long Base Line (LBL) system constituted by a set of acoustic transponders located in known fixed positions and the knowledge of the Time-of-Flights (ToFs) related to the acoustic signals exchanged by the vehicle with the transponders.

The use of ToF measurements for localizing a moving object is not a new concept. Several techniques can be found in literature [60, 61, 62], typically based on the following dual-step procedure: first, each ToF is simply transformed into the corresponding distance (range), by taking into account the speed of propagation of the exchanged signals in the considered medium; then, if a sufficient number of range estimations is available, the localization problem is solved by applying a standard trilateration-based algorithm [63].

Within terrestrial applications, the use of such a strategy is well consolidated and is normally employed, for example, by any GPS receiver, providing a quite accurate estimation of its 3D position by processing the ToFs of radio signals received by a proper set of satellites.

Conversely, in the underwater domain, the application of the above methodology is not as straightforward, as different issues are known to arise [64, 65]. Electromagnetic waves, which in air propagate with a known and constant speed, in water suffer from strong absorption effects which almost totally prevent their employment. Therefore the use of acoustic signals is the only viable solution. However the SoS in water is not a constant parameter, as it depends from the physical characteristics of the considered area. Even in a confined water column, a non constant vertical distribution of temperature and salinity originates

a depth-varying SoS profile. The resulting anisotropy of the acoustic channel causes significant distortions in the path actually followed by an acoustic signal traveling from an emitter to a receiver.

This fact is well known and indeed most recent and more sophisticated realizations of acoustic localization devices, such as USBL and LBL systems (Ultra Short & Long Base Line, [66]), can actually provide corrections to range estimates, via acoustic ray-tracing applied to measured ToF and Angle of Arrival (AoA). Despite their apparent efficiency, there is a concrete limit in the applicability of the above systems for AUVs that aim to have a good level of autonomy. Such instruments are in fact conceived for tracking the position of a mobile underwater device, while they do not allow the underwater device itself be aware of its position, which remains an information available only at a centralized level. However, for achieving a good level of autonomy (and possibly, cooperation within a team of AUVs), it is mandatory that the AUV knows its position with the maximum achievable precision.

As a consequence of the anisotropy of the acoustic channel, the accuracy achievable through the first step of the above procedure (i.e. the range evaluation on the basis of the measured ToF) is strictly related with the level of knowledge of the SoS profile in the considered water volume. A precise characterization of the SoS profile is a time-consuming (and expensive) activity, which may deserve to be performed just for some specific missions, where collected data by AUVs require high level of geo-referred accuracy. In all the other situations, the availability of an accurate SoS profile may not be assumed as a constraint for the operations.

In light of these considerations, two algorithms are therefore presented, which use two different methods for obtaining a range from the measured ToF, and also differ for the assumptions and required information.

The first algorithm presented in Section 3.2, called RT^2 (Real Time Ray Tracing), is based on the concept of LBL systems, where the role of a buoy can also be interpreted by one of the AUV of the team. The main assumptions of the algorithm are the a-priori knowledge of the SoS profile of the water column, the fact that the SoS is constant in radius and that the sea floor is also constant in depth. Under these assumptions, the propagation of acoustic waves is periodic, and an efficient way to compute the range from the emitter on the sole basis of the ToF and the depth of the emitter and receiver is presented. The validity of the algorithm is demonstrated through different simulations, showing how the proposed

algorithm captures the anisotropy of the underwater medium. Furthermore, a preliminary sensitivity analysis is also carried out, showing how, even in presence of uncertainties in the knowledge of the SoS, the algorithm still outperforms a straight ray propagation approximation by about one order of magnitude.

Due to the extreme importance of localization, even in the light of the need of autonomy, a second algorithm is then presented in Section 3.3. The framework is again a LBL system, but in this algorithm no assumption about the knowledge of SoS is made. Instead, the SoS is characterized as a constant and unknown parameter to be further estimated. Thus, a minimization problem is formulated and the best solution is obtained, in the least squared sense. The advantage of this algorithm is that no preliminary sampling of the underwater column is required, however, as only a constant value is used to model the SoS, the algorithm works well within the validity of a straight ray propagation assumption. Due to its simplicity, this algorithm has been already implemented in the eFOLAGA AUV, as part of the CO^3AUVs EU project.

3.2 RT^2: Real Time Ray Tracing for Underwater Range Evaluation

The first of the two proposed algorithms for AUV localization is called RT^2 (Real-Time Ray Tracing) and explicitly takes into account the propagation anisotropy of the underwater medium (i.e. the changes in the sound velocity profile as a function of depth).

Previous treatments of the acoustic path distortion effects can be found in [60], [64], where LBL systems are considered, but the ray correction is not performed on board the vehicle(s). In [67] the simplified solution of considering straight ray paths is proposed, by using as constant sound speed the weighted depth average of the true sound speed. This latter solution may be useful when propagation occurs mostly on the vertical, but it is shown not being accurate at oblique angles.

A novel algorithm is therefore proposed, based on a modified formulation of the sound-ray propagation laws allowing to efficiently handle the distorted and reflected acoustic ray paths and to produce accurate evaluations of distances between pairs of AUV within a team, as well as between an AUV and the buoy elements of an LBL system. Further, the implementation of the proposed RT^2

technique is very simple. It just requires the presence, on-board each vehicle, of an acoustic modem (already necessary for coordination purposes) and of a pair of look-up tables, a-priori built from the knowledge of the depth-dependent sound velocity profile. The proposed RT^2 algorithm can be easily hosted even on-board of low-cost AUVs.

Moreover, the adoption of this algorithm reduces the distributed localization problem for AUV teams to its aerial counterpart, for which efficient filtering techniques already exist [68, 69, 70]. Some of these techniques are therefore expected to be transferable to the underwater field, significantly reducing the high level of criticality generally posed by underwater medium anisotropies. Moreover, the proposed algorithm can also be applied for the distributed localization of underwater moving sensor networks (see [61] for a survey on this topic, and [71, 72] for most recent results).

3.2.1 Brief Review Sound-Ray Theory

Let us consider a cylindrical coordinate system, with the origin at the sea surface and the z axis pointing downward toward the sea bottom. It is assumed that the sound speed in water can vary only with depth and not with range from the source (*depth-dependent* sound velocity profile). With these assumptions, acoustic propagation can be represented in the plane (z, r) of depth and range, independent from the azimuth angle.

Acoustic propagation will be treated with a ray-theory approach, which is a valid approximation within the usually high (above kHz) transmission frequencies used in underwater localization and communication. Moreover, the assumption of a range independent environment allows for the use of the ray equations for a stratified media as reported in [65], pp.176. The acoustic source is assumed to be at 0 horizontal range. By letting $c(z) > 0$ denoting sound velocity, consider the associated Snell's law:

$$cos(\theta) = kc(z) \tag{3.1}$$
$$k = \frac{cos(\theta_0)}{c(z_0)} \in \left[0; \frac{1}{c(z_0)}\right]; \theta_0 \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$$

being θ_0 the angle with the horizontal axis of a ray starting at depth z_0 and θ the same one at a generic depth z, as depicted in Fig. 3.1.

In representation (3.1), a single (z_0, k) actually refers to two rays (z_0, θ_0) and $(z_0, -\theta_0)$. Also, from (3.1) the following condition can be found

$$1/k \geq c(z) \tag{3.2}$$

This equation implies that, for any given starting conditions $(z_0; k)$, the couple of resulting trajectories will time-evolve within the depth range, including z_0:

$$z \in [z_u(k), z_d(k)] \quad ; \quad k \in [0, 1/c(z_0)] \tag{3.3}$$

with $z_u(k), z_d(k)$ the abscissas of the first-encountered upper and lower intersection points that the vertical line $1/k = c(z_0)$ forms with $c(z)$, when seen as starting from z_0, as indicated in Fig. 3.2 where, for a given z_0, different values of $1/k$ are shown.

Clearly for any given value of parameter k, $z_u(k)$ must be set to zero (surface level) and $z_d(k)$ be set to z_M (bottom level) whenever the corresponding vertical segment $1/k$ results with one or both its end-points on one or both such extreme levels. Thus, depending from the shape of $c(z)$, some rays may remain confined between two definite depths included between surface and bottom, giving rise to the so called *acoustic channel*.

By now considering the kinematic equations for a generic ray, $\dot{r} = c(z)cos(\theta)$, $\dot{z} = c(z)sin(\theta)$, and constraint (3.1), the following equations can be derived

$$\begin{cases} \dot{r} = kc^2(z) \\ \dot{z} = \pm c(z)\sqrt{1 - k^2c^2(z)} \\ \theta = \pm arccos(kc(z)) \end{cases} \tag{3.4}$$

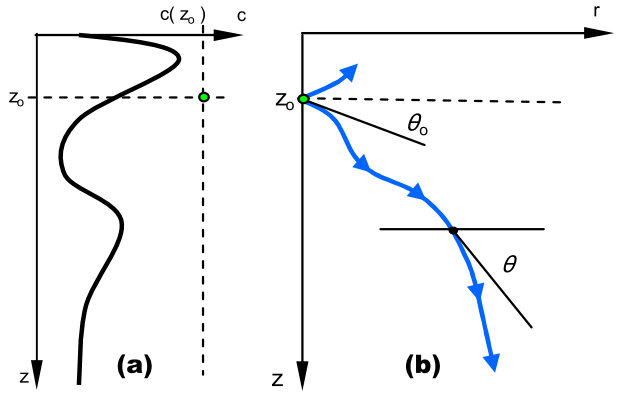


Figure 3.1: (a) sound velocity profile as a function of depth in the water column; (b) cylindrical coordinate system, sample ray path and corresponding angles accordingly with Snell's law

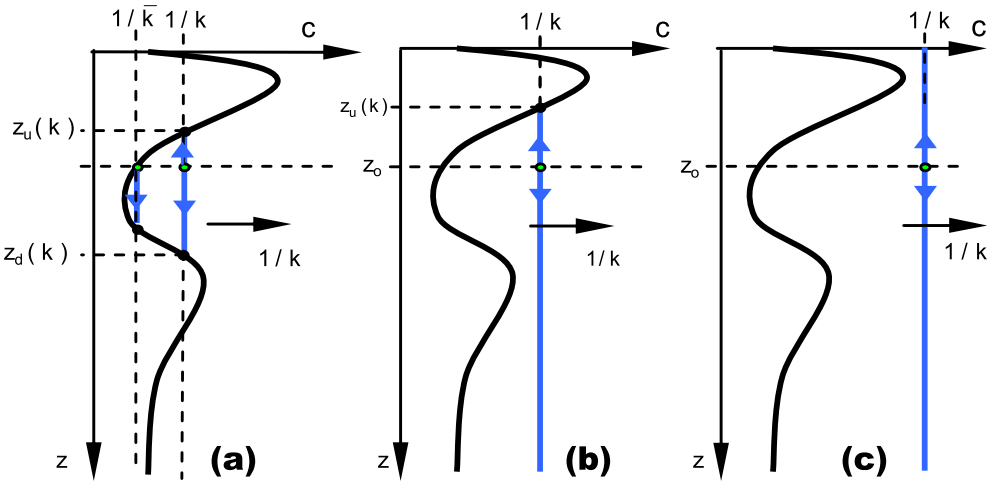


Figure 3.2: Sound speed profile as a function of depth and different lines $1/k$ corresponding to different couples of ray solutions. The intersection of any $1/k$ line with the sound speed profile determines the upper and lower depths within which the rays will be confined. (a) the rays are confined in an "acoustic channel" (b) the ray reaches the bottom, but is refracted at z_u (c) the ray moves freely between the top of the surface and the bottom.

where the "+" sign indicates the downward starting motion and the "-" sign refers to the upward starting one and where for a given $(z_0; k)$, the couple of equations for z are defined within the interval (3.3).

Moreover upon integration, each one of such equations monotonically reaches its associated interval extreme, respectively at the finite times (see equation 3.101 in [65]):

$$\begin{cases} t_d(z_0, k) \triangleq \int_{z_0}^{z_d^-} f(k, z)dz \\ t_u(z_0, k) \triangleq \int_{z_u^+}^{z_0} f(k, z)dz \end{cases} \tag{3.5}$$

with

$$f(k, z) \triangleq \frac{1}{c(z)\sqrt{1 - k^2c^2(z)}}$$

in correspondence of which, in case both the extremes do not coincide with the bottom and the surface, (as in Fig. 3.2a) it results $\dot{z} = 0$. Such occurrences correspond to the achievement of the finite time equilibria points $z_u(k), z_d(k)$. In terms of the sole kinematic equations (3.4), these equilibria can be shown to be both unstable. It follows that any perturbation will almost surely force each z motion having reached one of the extrema to reverse its direction, yielding to periodic evolutions. Note that the same result can be obtained more rigorously

including all the terms from the wave equation, in particular the dynamics for k and for $\chi = sin(\theta)/c(z)$, see for instance [65], pp.152. This is the case usually referred in physics as total reflection.

In case instead one or both of the extrema coincide with the surface and/or the bottom (Fig. 3.2b and Fig. 3.2c) surface/bottom reflections also impose a periodicity and must be accounted for. This is qualitatively depicted in Fig. 3.3, which is relevant to the case of Fig. 3.2b.

From (3.5), the time period $T(k)$ can be then easily calculated as:

$$T(k) \triangleq 2 \int_{z_u^+}^{z_d^-} f(k,z)dz \tag{3.6}$$

Further the ratio of the second and the first of (3.4) leads to:

$$\frac{dz}{dr} = \pm \frac{1}{kc(z)} \sqrt{1 - k^2 c^2(z)} \tag{3.7}$$

which establishes a *one to one* correspondence between the pair of time z motions $z(z_0, k, t)$ resulting from the second in (3.4) with the couple of distance z motions $z(z_0, k, r)$ resulting from the integration of (3.7). The representation of the pair of distance z motions on the (z, r) plane also exhibits periodic behaviours, which are qualitatively similar to those of their companion time z motions.

Moreover, by causally inverting (3.7) within a period (see also equation 3.97 in [65]) the resulting distance period $R(k)$ is:

$$R(k) \triangleq \int_{z_u^+}^{z_d^-} g(k,z)dz \; ; \; g(k,z) \triangleq \frac{kc(z)}{\sqrt{1 - k^2 c^2(z)}} \tag{3.8}$$

3.2.2 Basic Algorithms

Given a pair of depths z_a, z_b, for any assigned admissible parameter k (satisfying $1/k \geq \max\{c(z_a), c(z_b)\}$) the data sets $T_{ab}(k)$ and $R_{ab}(k)$ can be defined, as respectively constituted by all the possible travel times and their one to one associated horizontal travel distances connecting the depths z_a, z_b.

By now noting that the horizontal travel distance (or equivalently the travel time) of any ray connecting the depths z_a, z_b is constituted by an integer number of horizontal distance periods (time periods) plus one out of four possibilities of connection (see Fig. 3.4), the following expressions for the sets $T_{ab}(k)$ and $R_{ab}(k)$

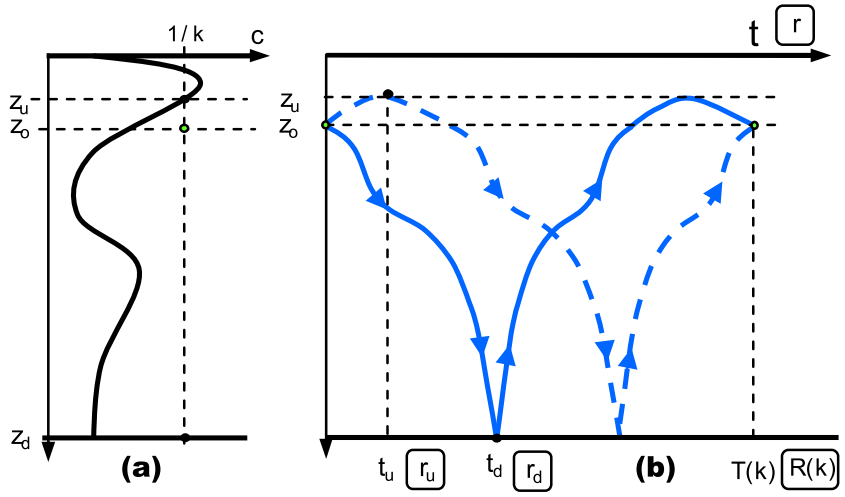


Figure 3.3: (a) sound speed profile as a function of depth, with indication of upper extremum z_u for a given k; (b) the couple of ray paths corresponding to the given k; note that ray evolution can be expressed either as a function of depth and time (t) or of depth and range (r).

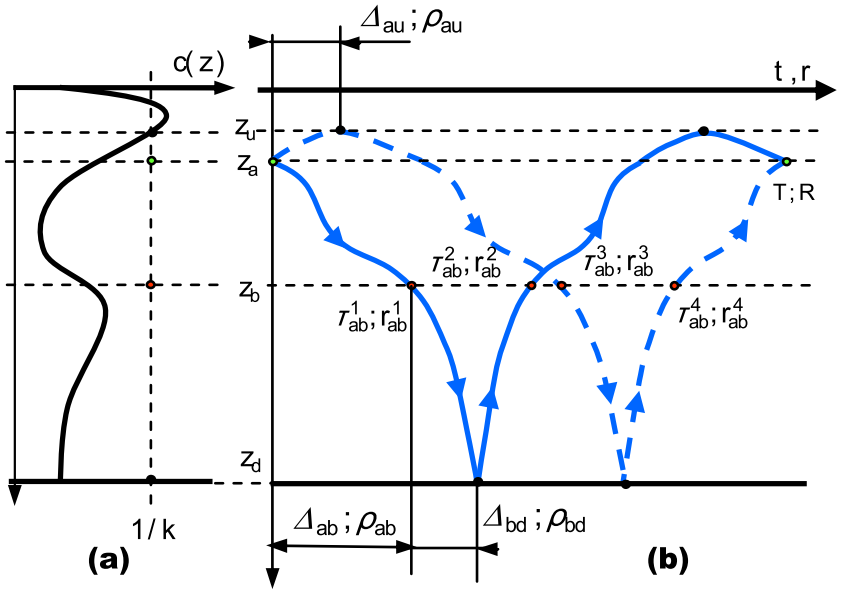


Figure 3.4: (a) sound speed profile as a function of depth, with indication of the chosen depths z_a, z_b and of the chosen k. (b) the couple of ray paths corresponding to the given k showing the 4 intersections of the ray with the depth z_b within one period, either in time or in range. $\Delta_{i,j}$ represents the travel time between points i, j, while $\rho_{i,j}$ represents the horizontal travel distance between the points i, j.

can be stated:

$$\begin{cases} T_{ab}(k) = \bar{T}_{ab}(k) + nT(k) \\ R_{ab}(k) = \bar{R}_{ab}(k) + nR(k) \end{cases} ;$$
$$\begin{cases} \bar{T}_{ab}(k) \triangleq \left\{\tau^j_{ab}(k); j = 1, \ldots, 4\right\} \\ \bar{R}_{ab}(k) \triangleq \left\{r^j_{ab}(k); j = 1, \ldots, 4\right\} \end{cases} ; \tag{3.9}$$

where the finite sets $\bar{T}_{ab}(k)$ and $\bar{R}_{ab}(k)$, respectively, admit the following representations:

$$\bar{T}_{ab}(k) = \left\{ \begin{array}{lcl} \tau^1_{a,b} & = & \Delta_{a,b} \\ \tau^2_{a,b} & = & \Delta_{a,b} + 2\Delta_{b,d} \\ \tau^3_{a,b} & = & \Delta_{a,b} + 2\Delta_{a,u} \\ \tau^4_{a,b} & = & \Delta_{a,b} + 2\Delta_{a,u} + 2\Delta_{b,d} \end{array} \right\}_{(k)} \tag{3.10}$$

$$\bar{R}_{ab}(k) = \left\{ \begin{array}{lcl} r^1_{a,b} & = & \rho_{a,b} \\ r^2_{a,b} & = & \rho_{a,b} + 2\rho_{b,d} \\ r^3_{a,b} & = & \rho_{a,b} + 2\rho_{a,u} \\ r^4_{a,b} & = & \rho_{a,b} + 2\rho_{a,u} + 2\rho_{b,d} \end{array} \right\}_{(k)} \tag{3.11}$$

From (3.9), (3.10), and (3.11) it is easy to see that, in correspondence of any given triple (z_a, z_b, k) any element of the resulting $T_{ab}(k)$, $R_{ab}(k)$ sets can be generated by knowing a finite number of parameters: the period $T(k)$ or $R(k)$ plus the finite sets $\bar{T}_{ab}(k)$, $\bar{R}_{ab}(k)$. As a consequence a pair of look-up tables, termed as *partial travel time (PTT)* and *partial traveled distance (PTD)* containing data for on-line evaluating such parameters can be constructed as follows.

For any $1/k$ (made gradually increasing via suitable step increments) detect its associated segments where $1/k \geq c(z)$. In correspondence of each one of them, evaluate the following integral functions (by arbitrarily considering either the "+" or the "-" forms)

$$\left\{ \begin{array}{lcl} \Delta_{\bar{z}}(k, z) & = & \pm \int_{\bar{z}}^{z} f(k, z)dz \\ \rho_{\bar{z}}(k, z) & = & \pm \int_{\bar{z}}^{z} g(k, z)dz \end{array} \right. ; z \in (z_u, z_d) \tag{3.12}$$

where $\bar{z}$ is the abscissa of any absolute minima of within the associated segment (z_u, z_d) (Fig. 3.5).

Once both PTT and PTD tables have been filled with terms $\Delta_{\bar{z}}(k, z)$ and $\rho_{\bar{z}}(k, z)$, for any given triple (z_a, z_b, k), the terms appearing in (3.10), (3.11) can be directly evaluated via a finite number of simple look-up operations:

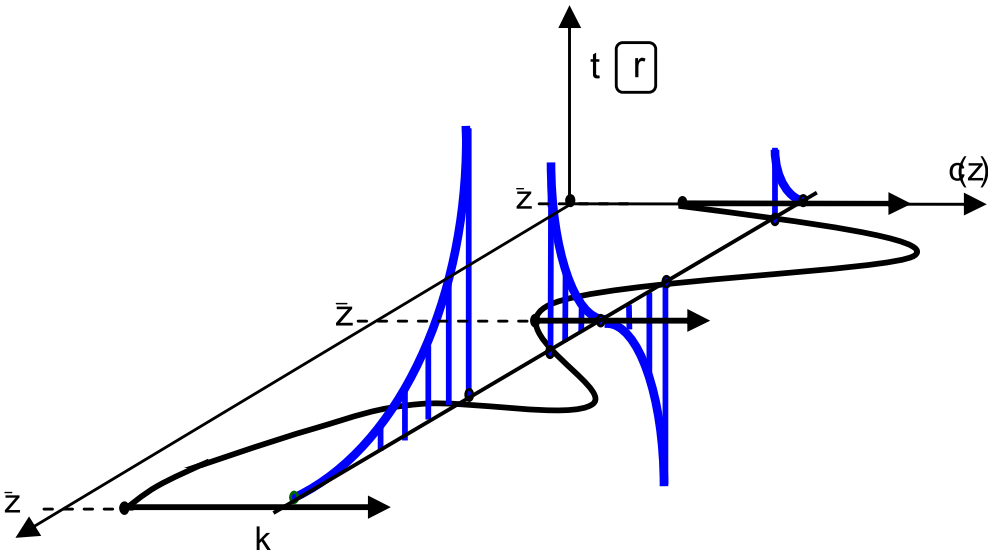


Figure 3.5: Sound speed profile as a function of depth and corresponding integral as defined in (3.12) either in time or horizontal range

$$\begin{cases} \Delta_{a,b}(k) &= |\Delta_{\bar{z}}(k, z_a) - \Delta_{\bar{z}}(k, z_b)| \\ \Delta_{b,d}(k) &= |\Delta_{\bar{z}}(k, z_b) - \Delta_{\bar{z}}(k, z_d)| \\ \Delta_{a,u}(k) &= |\Delta_{\bar{z}}(k, z_a) - \Delta_{\bar{z}}(k, z_u)| \end{cases} \tag{3.13}$$

$$T(k) = 2\,|\Delta_{\bar{z}}(k, z_u) - \Delta_{\bar{z}}(k, z_d)| \tag{3.14}$$

$$\begin{cases} \rho_{a,b}(k) &= |\rho_{\bar{z}}(k, z_a) - \rho_{\bar{z}}(k, z_b)| \\ \rho_{b,d}(k) &= |\rho_{\bar{z}}(k, z_b) - \rho_{\bar{z}}(k, z_d)| \\ \rho_{a,u}(k) &= |\rho_{\bar{z}}(k, z_a) - \rho_{\bar{z}}(k, z_u)| \end{cases} \tag{3.15}$$

$$R(k) = 2\,|\rho_{\bar{z}}(k, z_u) - \rho_{\bar{z}}(k, z_d)| \tag{3.16}$$

from which $\bar{T}_{ab}(k)$, $\bar{R}_{ab}(k)$, as well as also $T_{ab}(k)$, $R_{ab}(k)$, follow immediately.

The PTT and PTD look-up tables, whose computation has just been described, can now be considered as a unique basic functional block that, when receiving in input the triple (z_a, z_b, k), it immediately provides as output the data vector $V_{ab}(k) \triangleq \left[T, \bar{T}_{ab} | R, \bar{R}_{ab}\right](k)$.

Horizontal Range Evaluation

By composing the processing elements introduced in the previous section, the complete implementation of the Real-Time Ray-Tracing (RT^2) algorithm for horizontal range evaluation can now be described.

It is assumed that an underwater vehicle B is located at z_b depth, known since it is provided by its on-board depth-meter. Vehicle B receives a signal from another vehicle A transmitting from depth level z_a. Such signal is further assumed encoding the time instant t_a of its transmission, plus its departing depth z_a. The vehicle clocks are assumed to be synchronized. It is finally assumed that vehicle B is able to discriminate the first arrival of the signal from vehicle A, and to separate it from the subsequent, lower intensity, replicas due to multi-path. Note that the assumptions just made are similar to those routinely used in quite a number of existing commercial or prototypal equipments. In addition to the previous assumptions, it is further required to the vehicles to have the pair of look-up tables PTT and PTD (a-priori off-line evaluated from the assumed knowledge of the sound profile $c(z)$).

Under the above assumptions, vehicle B can evaluate (via trivial difference) the travel time τ spent by the sound ray that, as the *first one* received among many possible others, has reached B starting from A. From the computed travel time, the following processing mechanism provides a minimal set of possible candidate horizontal distances from vehicle A.

The PTT table is explored for all values of k. For a given k, the condition $1/k > \max\{c(z_a), c(z_b)\}$ is evaluated, in order to consider only the admissible values of k. If such condition holds, the integer number of periods n is calculated, as the integer division between τ and $T(k)$. The remainder of the division, say τ_r is then compared to the partial travel times described in (3.13), i.e. it is checked if $\tau \in \bar{T}_{ab}(k)$. If a positive match is found (say against the $\tau^j_{ab}(k)$), the value k is considered admissible and is placed in a list of candidate solution. For such value, the corresponding range r is computed as

$$r = nR(k) + \rho^j_{ab}(k)$$

Note that this comparison is performed with a given threshold, because due to the discrete nature of the algorithm, the comparison would never hold true otherwise. Thus, the actual time

$$\tilde{\tau} = nT(k) + \tau^j_{ab}(k)$$

is slightly different from the received time τ.

Such procedure is performed for all the values of k and may result in multiple solutions $(k_c, \tilde{\tau}_c, r_c)$, where c stands for candidate.

In order to reduce the number of candidates solutions, a second check is performed. In this case, the goal is to keep only the "first-arrival" rays. Let us consider one of the previously found solutions $(k_c, \tilde{\tau}_c, r_c)$. This time let us explore the PTD table, considering the range r_c. The same procedure as before is applied, but this time the division is performed between r_c and $R(k)$. If a match is found (say against the $\rho^h_{ab}(k)$), the *time* corresponding to such match is computed, as

$$\tau_v = nT(k) + \tau^h_{ab}(k)$$

where v stands for "virtual".

Two scenario may results:

1. $\tau_v \leq \tilde{\tau}_c$ which implies that the considered "virtual" ray would have been received before the "real" one, if the vehicle were at the considered horizontal distance r_c;

2. $\tau_v > \tilde{\tau}_c$ which implies that the considered "virtual" ray would have arrived after than the "real" one.

In the first case, the considered candidate solution $(k_c, \tilde{\tau}_c, r_c)$ is then discarded, since, if the vehicle was indeed at the supposed range r_c, a ray with parameter k would have been received before the considered ray k_c, thus going against the assumption of "first arrival" ray.

3.2.3 Simulative Results

In this section, some preliminary simulation results are presented. The results have been obtained using a ray tracing simulator to compute the true ray trajectory, and the time τ so obtained has been used as input to the RT^2 algorithm, as if it was received from the vehicle's modem. The localization results have been compared to the usual assumption of straight ray propagation, using the average speed along the water column. In the simulations, the receiver has been positioned at different radial distances from the emitter (with a step size of $1m$), in such way to see the different performances of the algorithm.

Simulations under Perfect Knowledge of the Environment

In these cases, the RT^2 algorithm and the straight ray approximation use the exact sound speed profile used in the simulator.

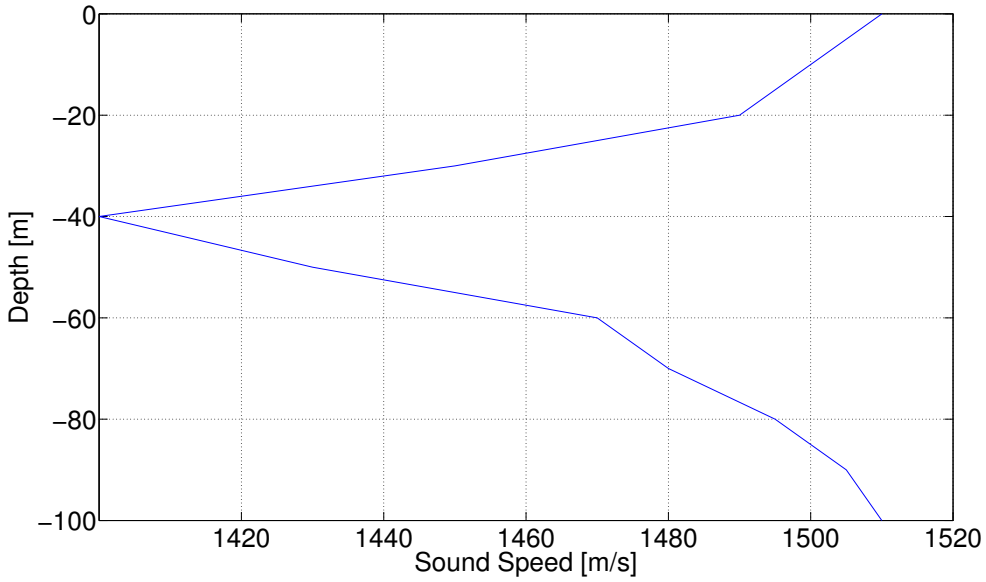


Figure 3.6: Sound speed profile used in Test 1.1 and Test 2.1. The profile used in the simulation is exactly known by the RT^2 algorithm.

Test 1.1

The first test refers to the sound speed profile shown in Fig. 3.6 and the results are presented in Fig. 3.7. In this case, the emitter is located at a depth of $z_a = 60m$, while the receiver is positioned at a depth of $z_b = 90m$.

As can be seen, except for a few peaks/outliers that will be discussed in a separate section, the error of the RT^2 algorithm is practically zero, compared to about $20m$ of the straight ray approximation. The big jumps in the error of the straight ray approximation are due to the jumps which are naturally occurring in the real simulated ToF and to the fact that the straight ray approximation cannot actually account for them. Such natural discontinuities in the time occur in correspondence of changes in the type of solutions (or solution "family"). For example, rays that propagate between the given z_a and z_b without reflection on the bottom or on the surface, are characterized by a maximum range they can reach. After such range, the first ray is one that has either reflected on the surface or on the sea floor. In either case, the path of the ray has changed in a discontinuous way, and so does also the time needed to reach the corresponding range at depth level z_b.

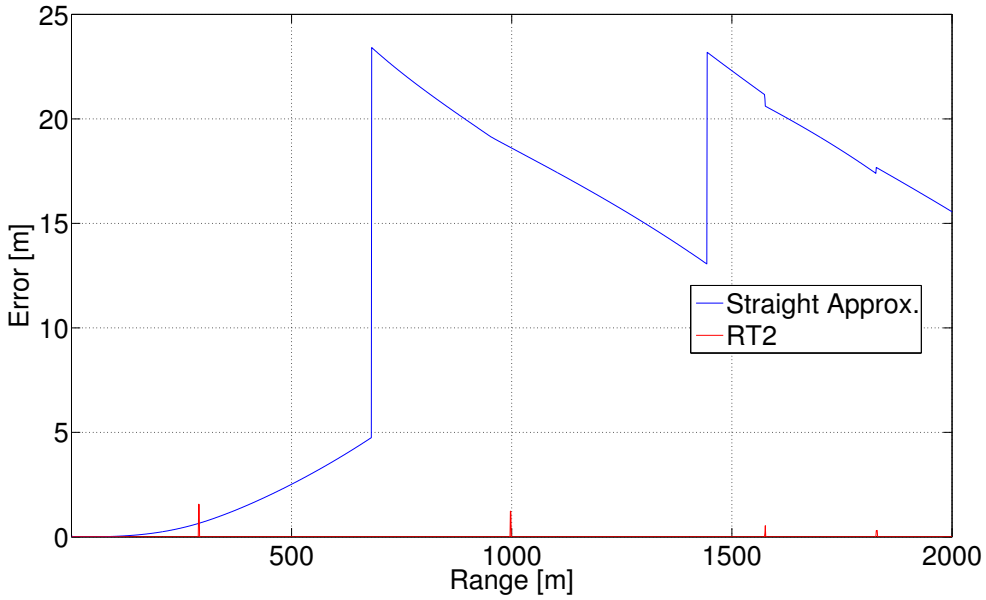


Figure 3.7: Test 1.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

Test 2.1

In the second test, the same profile is used, however the position of the emitter is changed to $z_a = 40m$ and the receiver's one to $z_b = 60m$. Thus, the pair is now inside the so called *acoustic channel*. The results of this simulation are shown in Fig. 3.8. Again, neglecting the outlier, the RT^2 algorithm yields excellent results, as compared to the high errors committed by a straight ray approximation, where, due to the higher nonlinear nature of the acoustic channel, the error increases to about $50m$.

Test 3.1

In this test, the profile is changed to the one of Fig. 3.9, which is even more complicated. The emitter is located at $z_a = 30m$ and the receiver at $z_b = 90m$. Thus in this case, the speed profile prevents rays between $\theta = 0\,^\circ$ and $\theta = arccos(c(z_a)/c(60)) = 9.36\,^\circ$ to reach the depth z_b.

Figure 3.10 shows the results of this simulation. Again, except two outliers, the RT^2 algorithm captures the ray propagation very well, and is able to keep the error at basically zero in all the simulated range.

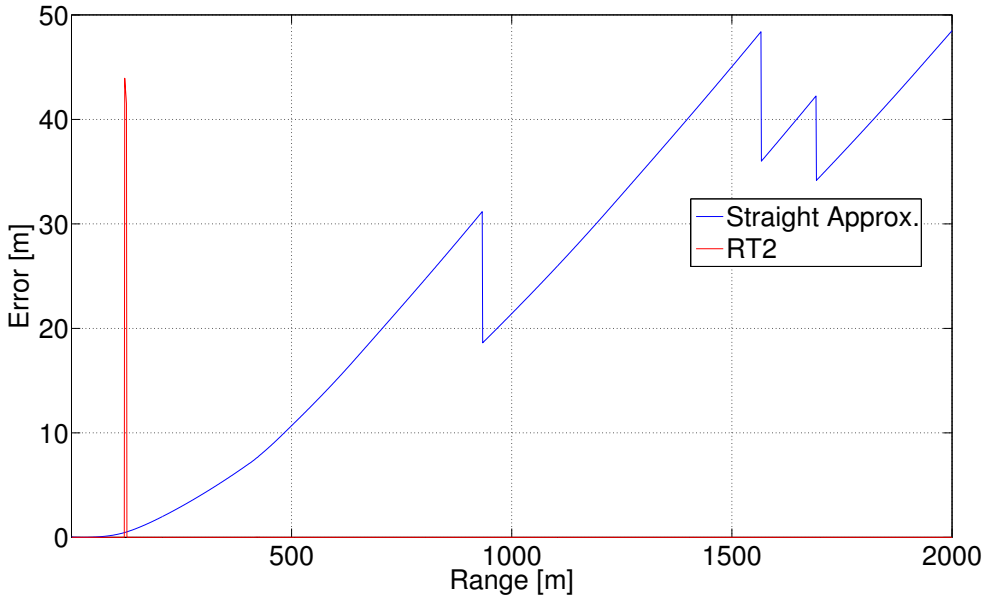


Figure 3.8: Test 2.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

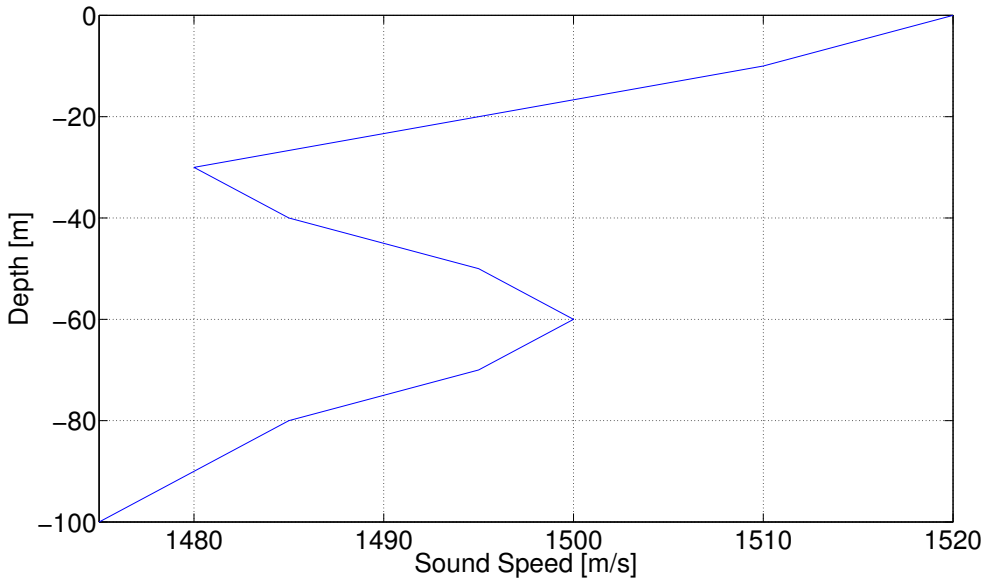


Figure 3.9: Sound speed profile used in Test 3.1. The profile used in the simulation is exactly known by the RT^2 algorithm.

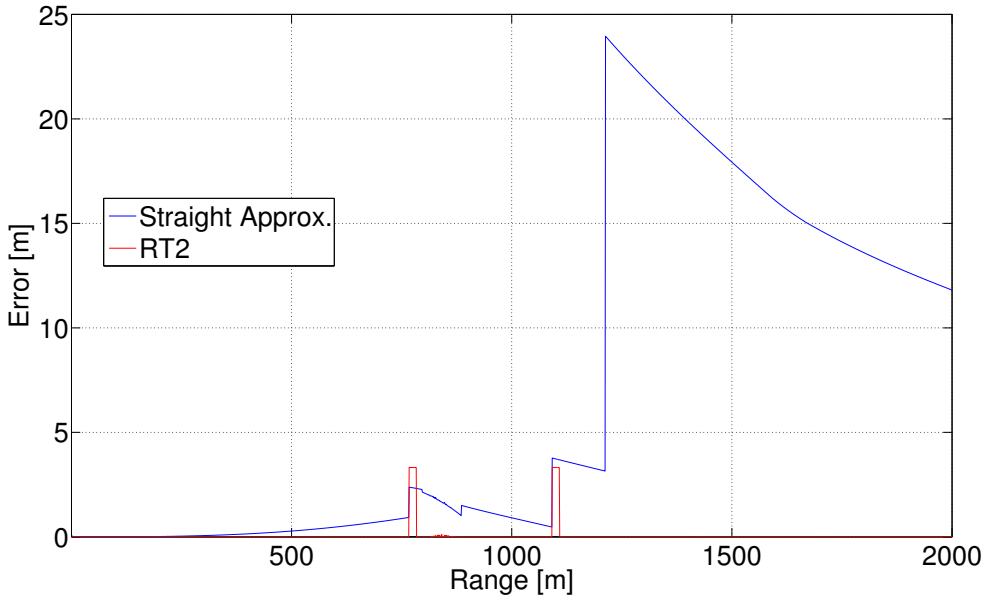


Figure 3.10: Test 3.1: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

Outlier Analysis

When the RT^2 algorithm has been presented in section 3.2.1 no mention has been made about the necessary discretizations to be used. Indeed there are a few of them, namely:

- the integrals leading to the PTT and PTD tables are calculated with a fixed dz;
- their values are stored in the PTT and PTD tables with a fixed step not necessarily coinciding with dz;
- the value of k is also discretized.

For instance, each integral is performed with a step of $dz = 0.00001m$, the values are instead stored in the table only at each $0.1m$, otherwise the table would be too huge to be memorized. However, such discretization is not critical.

More important is instead the discretization performed over k. In these simulations, around 30000 values of k have been used.

In section 3.2.2, when performing the check $\tau \in \bar{T}_{ab}(k)$, it is easy to note that, since k has been discretized, the check will rarely hold true exactly. Thus, a threshold of some sort has to be introduced. With such threshold, multiple values

of k may now pass this extended check. As a consequence, in order to limit the number of solutions, a interpolation is made between the solutions belonging to the same "family", i.e. direct rays, rays reflected from surface, reflected from bottom, or both (see Fig. 3.4 for a representation of these 4 family of solutions).

However, such interpolation may fail in two critical cases:

- inside an acoustic channel, when the receiver is close to the bottom $z_d(k)$ (or the top $z_u(k)$) of the channel;
- at a range very close to a change of the "family" of the solution.

The big spike in Fig 3.8 is actually an instance of the first, whereas for example the spikes in Fig. 3.10 occurs near a change of solution "family". However, since these errors occur for very few values of τ, their effect can be subsequently excluded by the triangulation scheme to be contemporarily used for localization. Indeed, the presence of multiple buoys or vehicles allows, considering that the range cannot actually have any abrupt jumps, to automatically discard the buoy/vehicle range with such error from the triangulation algorithm itself.

Finally, a better interpolation for the first case is currently under development and will hopefully better mitigate that problem.

Sensitivity Analysis

In order to validate the sensitivity of the algorithm to errors in the sound speed profile, more simulations have been carried out. In all the following cases, the speed profile used by the ray tracer simulator, in order to compute the true ToF τ is different from the sound speed profile used by the RT^2 algorithm and the straight ray approximation. These simulations are important as in practical situations the speed profile will be sampled, and known with a certain degree of error.

Test 1.2

In this test, the sound speed profile used are shown in Fig. 3.11. The profile used is the result of sampling the true one every $20m$. The emitter is located at a depth of $z_a = 60m$, while the receiver is positioned at a depth of $z_b = 90m$, thus outside the acoustic channel.

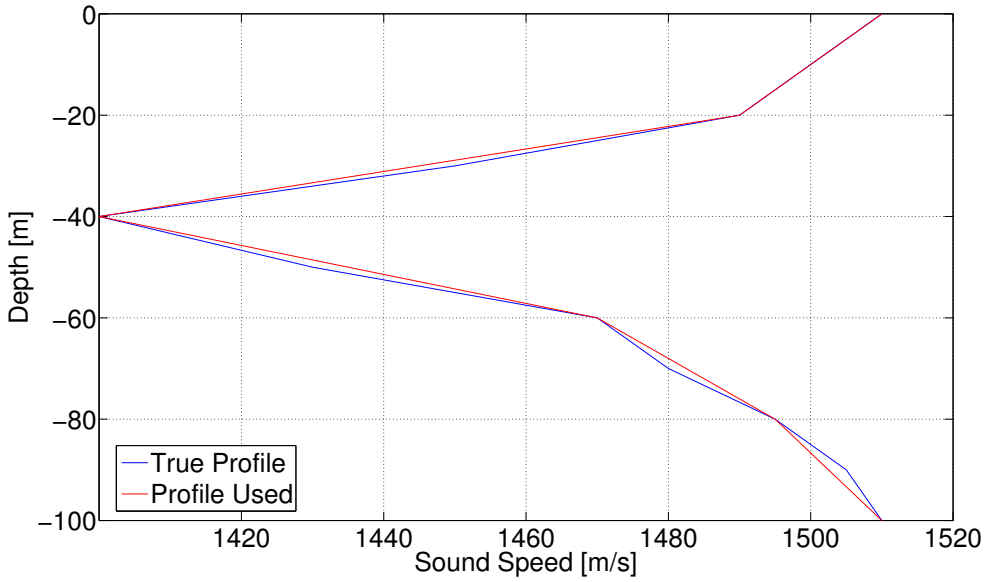


Figure 3.11: Sound speed profile used in Test 1.2 and Test 2.2. The profile used by the RT^2 algorithm and the straight ray approximation is the result of a sampling of the true one.

The result of this test is shown in Fig. 3.12. As can be seen, there is now a zone where the RT^2 algorithm performs worse than a straight ray approximation. This is due to the fact that since the sound speed profile is not perfectly known, the time discontinuities occur at a different range, thus the RT^2 picks the wrong solution for a while. This is also shown by the fact that, when the discontinuity actually occurs, the error becomes very low again.

Test 2.2

This test exploits the same sampled sound speed profile, though this time the emitter and receiver are inside the acoustic channel, at a depth of $40m$ and $60m$ respectively.

The plot of the errors is reported in Fig. 3.13. The results are very similar to the previous simulation. The zones where the RT^2 error is above zero are those just before a time discontinuity, i.e. a change in the type of solution.

Test 1.3

The last two sensitivity tests use the profiles as shown in Fig. 3.14. In this case, rather than being sampled, the sound speed profile is injected with a random error of $\pm 7m/s$.

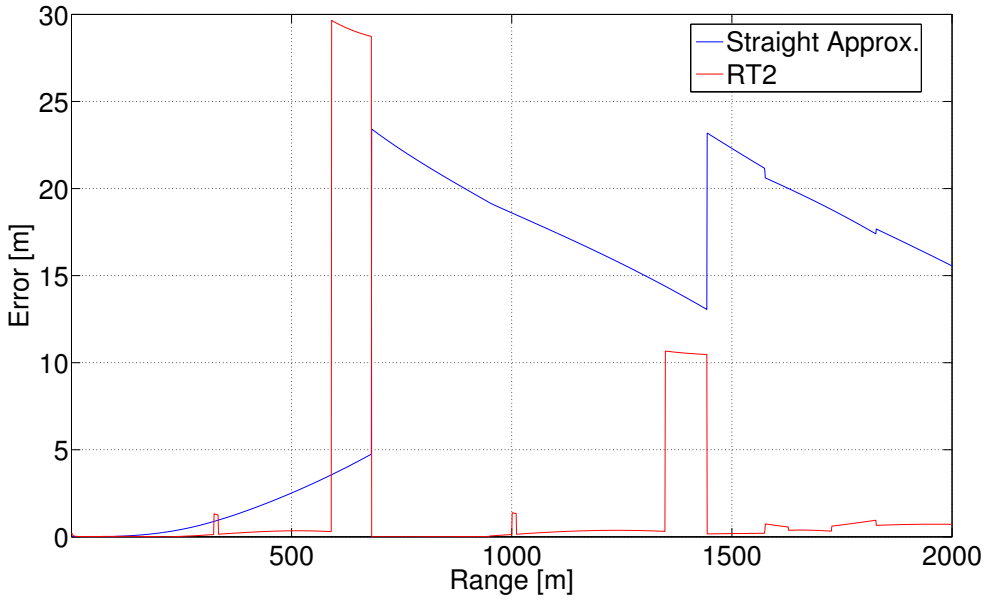


Figure 3.12: Test 1.2: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

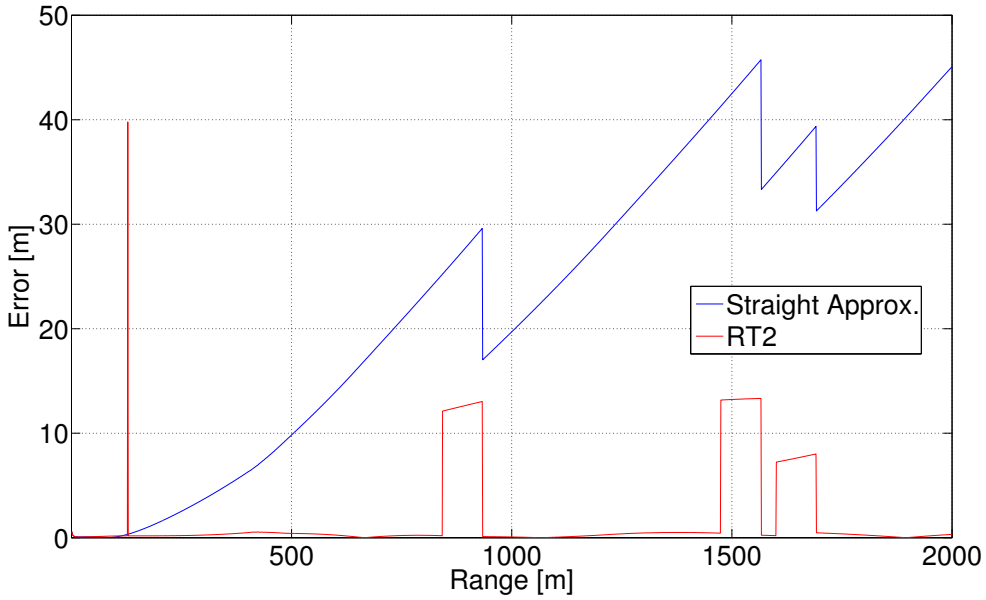


Figure 3.13: Test 2.2: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

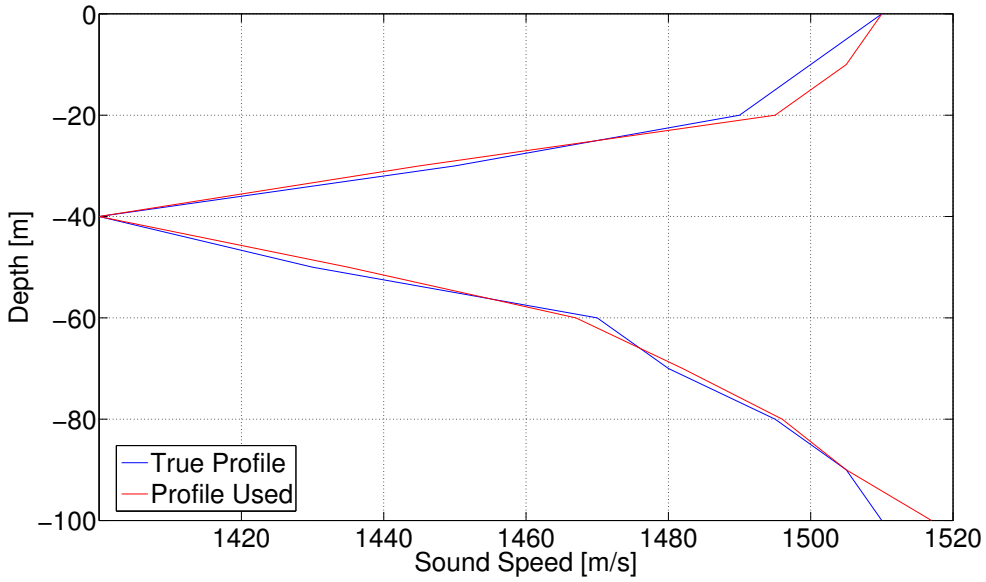


Figure 3.14: Sound speed profile used in Test 1.3 and Test 2.3. The profile used by the RT^2 algorithm and the straight ray approximation differs from the true one with a random noise of $\pm 7 m/s$.

The first of these two last tests is performed again outside the channel, as in Test 1.1 and 1.2. The results are similar as in the previous cases (Fig. 3.15), the main difference being the fact that the mean error is slightly increased due to the higher uncertainty in the sound speed profile.

Test 2.3

The last test is again performed using the profiles of Fig. 3.14, and by placing emitter and receiver inside the acoustic channel. In this case, even with the uncertainty of the speed profile, the proposed algorithm performs very well in all the considered range (Fig. 3.16).

Summary of Results

All the previously shown simulations are concisely summarized in Table 3.1, which reports, for each scenario, the maximum and average errors obtained by using both the proposed RT^2 technique and the straight ray approximation.

Moreover, the same results are reported in Table 3.2 in terms of their relative value, i.e. the error committed is compared to the actual distance of the receiver from the emitter.

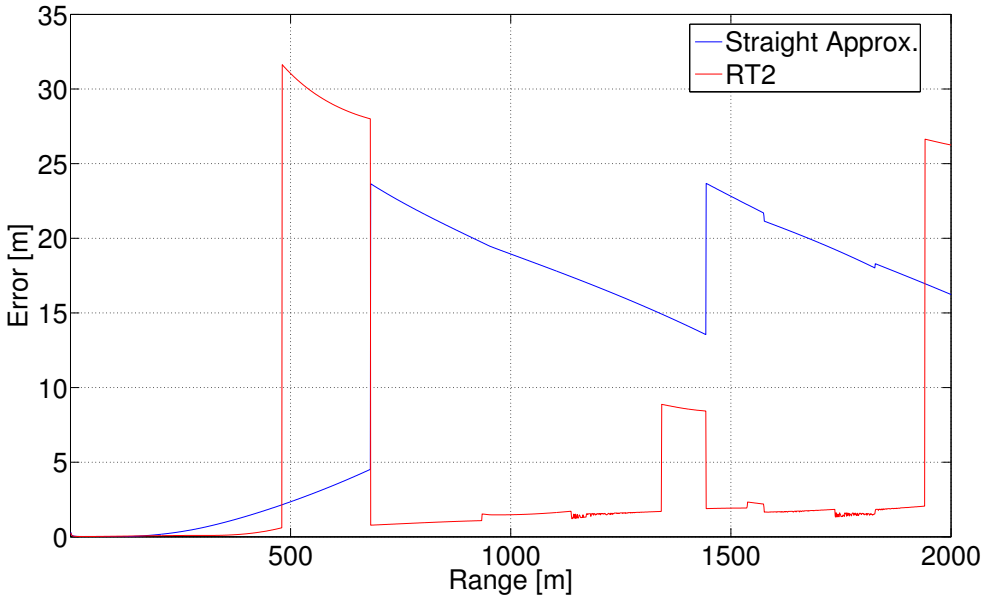


Figure 3.15: Test 1.3: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

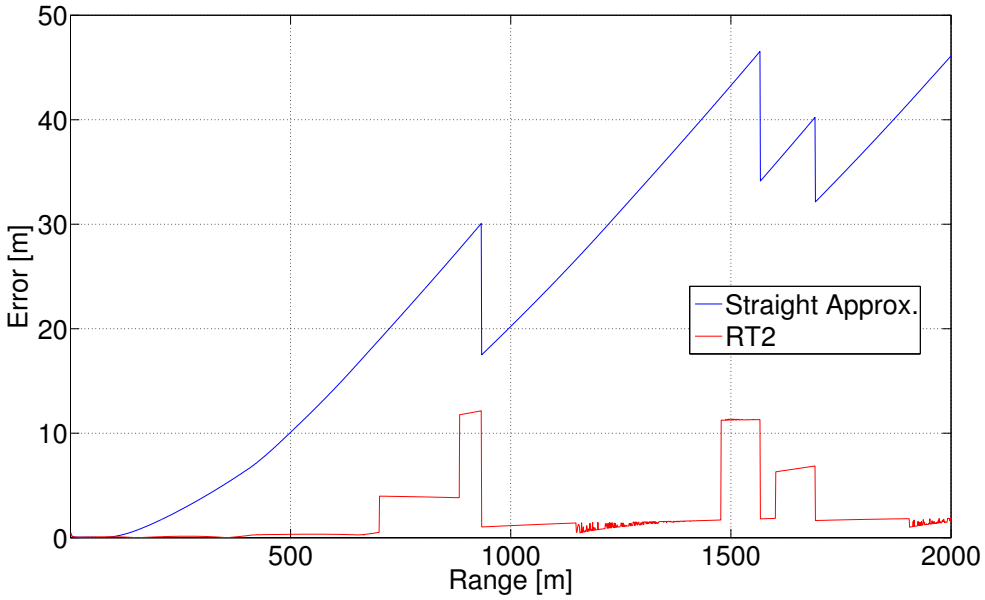


Figure 3.16: Test 2.3: plot of the absolute error of the proposed RT^2 algorithm compared with a straight ray approximation.

Table 3.1: Absolute maximum errors and mean error using both the RT^2 algorithm and a straight ray propagation approximation.

	RT^2		Straight Ray Approx.	
Test #	Max. Err	Avg. Err	Max. Err	Avg. Err
1.1	1.57m	0.004m	23.4m	12.7m
2.1	43.9m	0.1m	48.5m	24.7m
3.1	3.32m	0.06m	23.9m	7m
1.2	29.7m	2.1m	23.6m	12.9m
2.2	39.8m	1.7m	45.7m	22.9m
1.3	31.6m	5.1m	23.7m	12.9m
2.3	12.2m	2.2m	46.5m	23.5m

Table 3.2: Relative maximum errors and mean error using both the RT^2 algorithm and a straight ray propagation approximation.

	RT^2		Straight Ray Approx.	
Test #	Max. Err	Avg. Err	Max. Err	Avg. Err
1.1	0.55%	0.0008%	4.7%	1.2%
2.1	36.3%	0.09%	7.8%	2.2%
3.1	0.4%	0.007%	6%	0.5%
1.2	23.4%	0.3%	26.5%	1.2%
2.2	55%	0.23%	60%	2%
1.3	26.4%	0.7%	30.4%	1.2%
2.3	41.3%	0.23%	46.4%	2.1%

As it can be seen, whenever the speed profile is perfectly known (Tests 1.1, 2.1 and 3.1), the RT^2 provides excellent precision, of about a few centimeters, except for the presence of a few outliers, as previously discussed. When the sound speed is sampled (Tests 1.2 and 2.2) or known with error (Tests 1.3 and 2.3) the RT^2 mean error is increased to a few meters. However a few considerations must be made:

- in average, there is still one order of magnitude of increase of precision;
- the areas of greater error are limited in range.

In a common scenario where the localization is performed via triangulation with four acoustic buoys, or in case of presence of multiple vehicles, it is unlikely that those errors occur at the same time for more than one buoy or vehicle. Since the AUV moves continuously in space, a jump in the range from one buoy or vehicle can be detected as outlier and the localization can still be performed with the remaining agents, whose time of flight can still exploit the RT^2 algorithm for a great increase in the accuracy of the range calculation.

3.2.4 Application to Distributed Localization

The framework underlying the problem of AUVs group localization is represented by a set of more than one AUV, each one equipped with its on-board acoustic modem. Figure 3.17 depicts such scenario, where the presence at surface level of one or more USVs (or elements of an LBL system), each one endowed with GPS absolute localization capabilities, has been also assumed.

Let us assume that any generic j-th vehicle of the group (via modem and while moving) intermittently broadcast a pulse immediately followed by a data stream containing (for a total of *eight* numbers) the following information:

- its identification code j;
- the pulse emission instant $\bar{t}_i^j; i = 1, 2, \ldots;$
- its depth $\bar{z}_i^j$ at the pulse emission instant $\bar{t}_i^j$;
- the *prediction* $\hat{\bar{\chi}}_{i/i-1}^j$ location $\bar{\chi}_i^j$ at instant $\bar{t}_i^j$;
- the variance $\bar{p}_{i/i-1}^j$ of $\hat{\bar{\chi}}_{i/i-1}^j$.

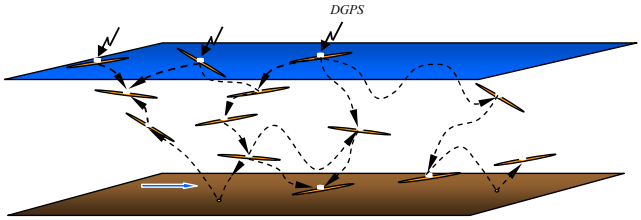


Figure 3.17: Conceptual configuration of the team localization scheme, in which localization is obtained with respect to neighboring vehicles and ultimately to the GPS linked surface vehicles

It is meanwhile required that any other generic k-th vehicle differing from the surface ones must also be able to translate its previous *prediction* $\hat{\chi}^k_{i/i-1}$ of its horizontal location χ^k_i (relevant to the pulse reception instant $t^j_i > \bar{t}^j_i$ and generally obtained via time integration of its own odometric data) into the upgraded *estimate* $\hat{\chi}^k_{i/i}$ of the same location at same time.

Note that only the production, by part of a vehicle, of prediction and estimates relevant to its *horizontal positions* is actually required. This because the shared knowledge of depths projects the problem into the horizontal plane. Moreover, the possibility offered by the RT^2 technique of directly measuring the horizontal distances between AUVs even in presence of non negligible ray distortions and reflections, reduces the problem to be of the same type of those generally considered within (planar) in air and range based distributed localization applications. As a consequence, most part of the already existing in air techniques for such problem is expected to be adoptable also for the underwater environment, without any sensible further modification. More precisely, it is deemed here that among the most appreciable advantages that could emerge from the adoption of the here proposed RT^2 method, the following should actually reveal a particular significance.

In terrestrial applications, the Gaussianity of the unavoidable noise superimposed to any distance evaluation is recognized to be a very reasonable assumption. This is actually a basic support to the use of parametric filtering methods (like Kalman Filters, Extended Kalman Filters, or even Iterated Kalman Filters, see for instance [68, 69, 70]). Then, accordingly with the RT^2 possibility of direct measuring of underwater distances, most probably the above comment could be extended also to the underwater environment. This in the sense that assuming the horizontal distance measurements superimposed by Gaussian noise may now reveal less critical than when roughly used without taking into account sound

wave distorting and reflecting effects. As a matter of fact, the need of avoiding the criticality posed by Gaussian assumptions, when in presence of ray propagation anisotropies and reflections, might be one of the basic motivation underlying the recent proposal of use of much more sophisticated non parametric filtering techniques [73] for attempting to manage underwater anisotropies. Though, the here proposed RT^2 method does not at all prevent the use of non parametric filters, in case deemed necessary (for instance as an alternative way for producing outliers removals).

Within underwater distributed group localization, one approximate practical possibility for avoiding to fully face the problem of sound ray distortions and reflections (even those induced by possible acoustic channels) could be that of trying to act in such a way to keep (horizontally and vertically) each vehicle sufficiently close to its neighbors. Instead, with the use of the proposed techniques, such type of structuring of the AUVs group should not be anymore necessary, since the formulation of the proposed technique does not actually make any difference between approximately straight, distorted, or even reflected rays along any path length. The net consequence of this should be therefore to allow even small groups of AUVs effectively working with more sparse, thus more volumetric extended, configurations.

3.3 Acoustic-based Localization Technique for AUVs in the Absence of Information on the Speed of Sound Profile

The second of the two proposed algorithms for AUV localization does not take into account the propagation anisotropy of the underwater medium. Indeed, since a precise characterization of the SoS profile is a time-consuming (and expensive) activity, in many cases the availability of an accurate SoS profile cannot be assumed as a constraint for the operations.

Luckily, within short-range operations executed in not too-deep water, the effects of acoustic-ray bending are less crucial. Therefore a common practice adopted for solving the LBL localization problem in such operative scenarios is considering the SoS profile as a constant function, described by a scalar constant term V. Such an approximation, which at a first sight may seem too strong, is

actually not too unreasonable, provided a good-enough estimation of V term is available. Moving from this consideration, several previous works on the topic proposed interesting solutions to the range evaluation problem, by presenting good experimental data collected with different equipments. Some of them employed, as V, the weighted average of some values of sound speeds, preliminarily registered at different depths. Other works proposed the execution of a "calibration" phase (before the start of the real mission) during which V was initially estimated, by moving the AUV on the surface while comparing data obtained via the acoustic-based localization procedure with those coming from GPS.

In the present work a quite different approach to the underwater localization problem is described, aiming to avoid the need of a calibration phase, even in the total absence of information about the SoS profile of the considered area. Instead of following the above indicated dual-step procedure, the proposed solution is obtained by considering V as an additional unknown parameter within the estimation procedure. More specifically, whenever three or more transponders are employed, for any set of received ToFs, a cost function (dependent on the AUV coordinates and V) is computed and then minimized via a standard gradient-descent-like technique. The optimization process exhibits very good properties in terms of speed of convergence, thanks to the simple structure of the considered cost function. After few steps, the estimation of the AUV current 3D position is available, together with the corresponding value V.

This last term has not to be considered as a reliable estimation of the real SoS (as it is known to be a non-constant function). This would be the case only if the considered environment was characterized by an isotropic acoustic channel, with signals propagating along perfectly straight paths. Otherwise (as it normally happens) V just represents the best (in a least squared sense) scalar factor that, multiplied by the registered ToFs, gives the best estimation of the ranges between the computed AUV position and the transponders locations. Notwithstanding its physical meaning, the computed V may play an interesting role in the developed procedure, as it can provide indications on the level of confidence of the produced localization data. Indeed, whenever the current value of V results to be too distant from the previously computed ones, it means that the localization procedure originated unreliable data ("outliers") which should be discarded (possibly because one or more ToFs were not available or the measurements were too affected by noise).

For better introducing the proposed algorithm, the simpler 2D localization problem is first considered and introduced in details in Section 3.3.1, and is tackled by varying the number of available transponders in Sections 3.3.2, 3.3.3 and 3.3.4. The more general 3D case is successively dealt within Section 3.3.5. Finally, results obtained by both simulative campaigns and field-trials are presented in Section 3.3.6.

3.3.1 2D Localization Problem

The 2D localization problem can be formulated as follows: the AUV is positioned on a given plane, at an unknown point $X = [x, y]^T$; a set of N transponders are placed on the same plane, at fixed points $T_i = [x_i, y_i]^T$; $i = 1, \dots, N$; the AUV knows a-priori the transponders' positions and has to localize itself (i.e. finding out the best estimation of X) by measuring the registered ToFs, t_i; $i = 1, \dots, N$ w.r.t. the transponders. As in the considered problem the vehicle does not have any information about the speed of sound profile, an additional unknown parameter, the scalar term V, is introduced in the computations, leading to the following set of equations in the unknowns X and V:

$$\|X - T_i\| = V t_i \quad ; \quad i = 1, \dots, N \tag{3.17}$$

The existence and the number of solutions of (3.17) (other than clearly depending on the number of available transponders) are strictly related to the level of noise affecting the ToFs measurements. Within an ideal noiseless environment, the measured ToFs would always originate at least one valid solution (the one corresponding to the real position of the AUV), with a level of accuracy just dependent on the validity of the approximation of the SoS profile with a constant function V. Conversely if at least one of the measured ToFs is too corrupted by noise, no valid solutions might be determined. In the following sub-sections different cases will be analyzed, by considering a different number of available transponders.

3.3.2 Introductory Problem: 2 Transponders

In the first scenario, depicted in Fig. 3.18a, two transponders, T_1 and T_2 at a distance l_{21} are considered. In such a case, from (3.17), the following system of

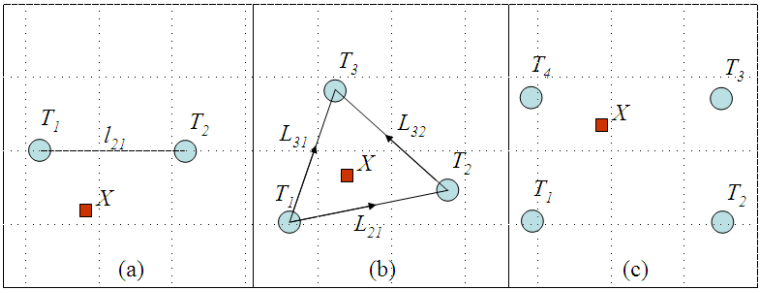


Figure 3.18: Different considered 2D scenarios (cyan dots are the transponders; the red square is the AUV)

equations can be formulated:

$$x^2 + y^2 - 2xx_1 - 2yy_1 + x_1^2 + y_1^2 = V^2 t_1^2 \tag{3.18}$$

$$x^2 + y^2 - 2xx_2 - 2yy_2 + x_2^2 + y_2^2 = V^2 t_2^2 \tag{3.19}$$

clearly stating that X must lay on one of the intersecting points (if any) of the circumferences centered in T_1 and T_2 with radii, Vt_1 and Vt_2, respectively.

By neglecting the trivial case where one of the measured ToF is null (thus meaning that the AUV is exactly located at the corresponding transponder's position), system (3.18) cannot be solved in the unknown $[x, y]^T$ because the V parameter is not known, and hence both the radii of the circumferences are not determined: different values of V lead to different estimations of X.

The set of points $[x, y]^T$ solving (3.18), by varying V, describes a certain locus whose equation can be made explicit as follows: under the assumption of non-null measured ToFs, by combining the two equations (3.18), the term V^2 can be eliminated; thus obtaining:

$$\left(x^2 + y^2\right)\left(1 - \frac{t_2^2}{t_1^2}\right) - 2x\left(x_2 - \frac{t_2^2}{t_1^2}x_1\right) - 2y\left(y_2 - \frac{t_2^2}{t_1^2}y_1\right) + \\ + \left(x_2^2 + y_2^2\right) - \frac{t_2^2}{t_1^2}\left(x_1^2 + y_1^2\right) = 0 \tag{3.20}$$

Now in case $t_1 = t_2$ equation (3.20) becomes:

$$x\left(x_2 - x_1\right) - y\left(y_2 - y_1\right) + \left(\frac{x_2^2 + y_2^2}{2}\right) - \left(\frac{x_1^2 + y_1^2}{2}\right) = 0 \tag{3.21}$$

which represents the perpendicular bisector of the segment $T_2 - T_1$. A fact, this last, consistent with the intuition: indeed, if the two measured ToFs are identical,

notwithstanding the value of V, the AUV must necessarily have the same distance w.r.t. to every transponder (thus staying on the perpendicular bisector of the segment joining them). For any point of line (3.21) the corresponding value of V can be easily obtained by applying (3.17). So doing, it is easy to see that the minimum admissible value of V, allowing a solution X of (3.18) is:

$$V_M = \frac{l_{21}}{t_1 + t_2} \tag{3.22}$$

leading to a unique solution X located on the median point M of the segment $T_2 - T_1$. Any value of V greater than V_M is licit and originates two solutions laying on line (3.21) and located on points symmetric w.r.t. M. As a consequence if $t_1 = t_2$:

$$V \geq \frac{l_{21}}{t_1 + t_2} \tag{3.23}$$

If instead $t_1 \neq t_2$, equation (3.20) represents a circumference, that is a closed curve. Thus meaning that solutions of (3.18) do exist only for a bounded range of V. To better understand this consideration it is convenient first analyzing the expressions of the center C_{21} and radius R_{21} of this circumference. To this aim, by introducing the following scalar term:

$$\alpha_{21} \triangleq \frac{t_1^2}{t_1^2 - t_2^2} \tag{3.24}$$

C_{21} and R_{21}, are obtainable, after some manipulation, as:

$$C_{21} = T_1 + \alpha_{21}(T_2 - T_1) \tag{3.25}$$

$$R_{21} = l_{21} + \sqrt{\alpha_{21}(\alpha_{21} - 1)} \tag{3.26}$$

By now noting from definition (3.22) that, with non null t_1 and t_2, the following properties hold:

$$\begin{aligned} t_1 > t_2 &\Rightarrow \alpha_{21} > 1 \\ t_1 < t_2 &\Rightarrow \alpha_{21} < 0 \end{aligned} \tag{3.27}$$

it becomes clear from equation (3.23) that the position of C_{21} is always on the line joining the transponders, outside the segment $T_2 - T_1$ (as $\alpha_{21} > 1 \notin [0, 1]$), on the side of the transponder which originated the smaller ToF; i.e. closer to T_1 if $t_1 < t_2$ (as $\alpha_{21} < 0$) and closer to T_2 if $t_1 > t_2$ (as $\alpha_{21} > 1$). Different placements of C_{21} for different values of $\alpha_{21} > 1$ are reported in Fig. 3.19.

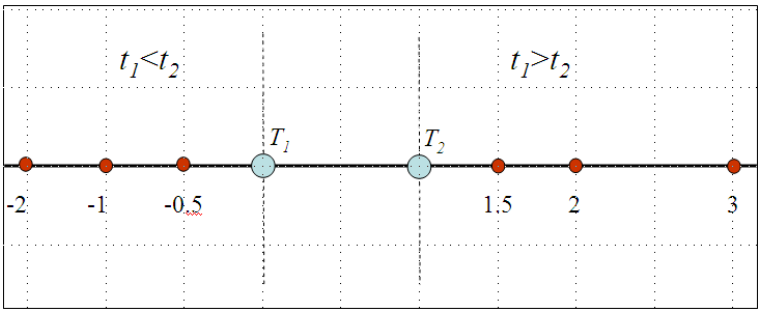


Figure 3.19: Different positions of C_{21} (red dots) for different valus of α_{21})

In order to obtain the bounding admissible values of V it is convenient to make explicit reference to Fig. 3.20, where a case with $t_1 < t_2$, has been considered, without loss of generality. As in the case $t_1 = t_2$, the minimum admissible value for V is the one corresponding to the point where circumference (3.20) intersects the segment joining the transponders; i.e A. If $V = V_P$ (a slightly greater value than V_A), two different X solutions P_1 and P_2 are found, belonging to circumference (3.20) and located at the opposite sides of the line of the transponders. By increasing further V different pairs of solutions are generated (still symmetric w.r.t. to the line) until the other intersecting point, B, between the circumference and the line is reached. Values of V greater than V_B do not allow any solution X of (3.18), because in such a case the two circumferences (3.18) will be one inside the other, with no intersections. By still referring to Fig. 3.20, from (3.25) and (3.26) it is easy to obtain the expression of points A and B:

$$A = T_1 + \frac{t_1}{t_1 + t_2}(T_2 - T_1) \quad B = T_1 + \frac{t_1}{t_1 - t_2}(T_2 - T_1) \tag{3.28}$$

whose corresponding values of V, are:

$$V_A = \frac{l_{21}}{t_2 + t_1} \quad V_B = \frac{l_{21}}{t_2 - t_1} \tag{3.29}$$

By generalizing the above considerations (taking into account also the case $t_1 > t_2$) it follows that if $t_1 \neq t_2$:

$$\frac{l_{21}}{t_2 + t_1} \leq V \leq \frac{l_{21}}{|t_2 - t_1|} \tag{3.30}$$

As a last remark of the present subsection, it is worth noting how both the distance of C_{21} from the segment and the size of R_{21} increases with the modulus of α_{21}, which in turn depends on the ratio between the two measured ToFs. By

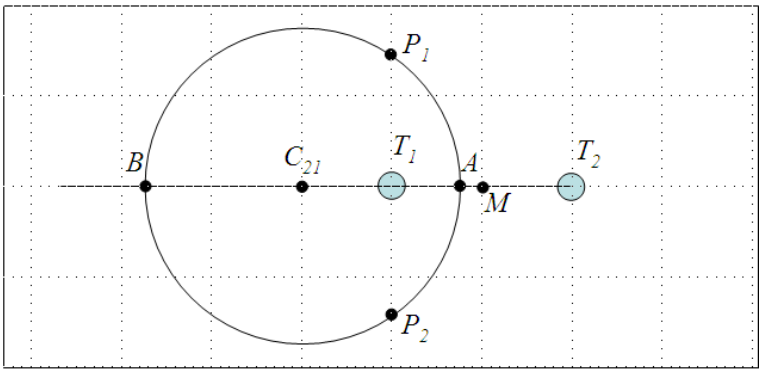


Figure 3.20: An exemplificative scenario with $t_1 < t_2$ ($\alpha_{21} < 0$)

again referring to the case of Fig. 3.20, if $t_1 << t_2$, the modulus of α_{21} is relatively small and so points A and B are not too far from T_1. If instead t_1 is slightly lower than t_2, the modulus of α_{21} is very big, A is close to M and B is very far from T_1. Making the ratio of the ToFs tending to unity, the modulus of α_{21} grows to infinity and the same holds for R_{21}. As a consequence, the case $t_1 = t_2$ can be considered as an extreme situation where the infinite-radius circumference (3.20) degenerates in the line (3.21).

In any case, as reasonably expectable, in the absence of knowledge about V, notwithstanding with the level of noise, two ToFs are not enough to univocally determine the position of the AUV on the considered plane.

3.3.3 3 Transponders

In the second scenario three transponders, T_1, T_2, T_3, defining a triangle on the plane, are considered (see Fig. 3.18b). The set of vectors describing the segments joining each pair of transponders can be defined as:

$$L_{ij} \triangleq T_i - T_j \quad ; \quad i, j = 1, 2, 3 \quad ; \quad i \neq j \tag{3.31}$$

whose lengths are:

$$l_{ij} \triangleq \|T_i - T_j\| \quad ; \quad i, j = 1, 2, 3 \quad ; \quad i \neq j \tag{3.32}$$

By following the same line of reasoning adopted in the previous subsection, the pair of ToFs (t_1, t_2) leads to the same circumference (3.20) (eventually degenerating into the line (3.21)). In this case, however, the availability of a third ToF, t_3, allow to consider other two loci of points, obtained by considering the pairs (t_1, t_3) and (t_2, t_3). To this aim, by introducing two additional scalar terms:

$$\alpha_{31} \triangleq \frac{t_1^2}{t_1^2 - t_3^2} \quad ; \quad \alpha_{32} \triangleq \frac{t_2^2}{t_2^2 - t_3^2} \tag{3.33}$$

and by again assuming that none of the measured ToFs is null, it is an easy matter to understand that the additional loci are again circumferences (possibly degenerating into lines), having centers and radii as follows:

$$C_{31} = T_1 + \alpha_{31}\,(T_3 - T_1) \qquad C_{32} = T_2 + \alpha_{32}\,(T_3 - T_2) \tag{3.34}$$

$$R_{31} = l_{31} + \sqrt{\alpha_{31}\,(\alpha_{31} - 1)} \qquad R_{32} = l_{32} + \sqrt{\alpha_{32}\,(\alpha_{32} - 1)} \tag{3.35}$$

Before describing in details the way of using the three circumferences for solving the original localization problems, it is convenient to show two interesting properties relating them.

The first property states that if two among the three circumferences have intersecting point(s), the third one must necessarily intersect both the other two in the same point(s). This property can be easily demonstrated as follows. First assume that two circumferences, say those related to pairs (t_i, t_j) and (t_i, t_k) intersect in points P and Q and then consider just one of them, say P (the same reasoning indifferently holds for Q). As P belongs to the first circumference, from (3.17) it means that a certain value of V, namely V_1, exists such that:

$$\|P - T_i\| = V_1 t_i \quad ; \quad \|P - T_j\| = V_1 t_j \tag{3.36}$$

At the same time, as P belongs also to the second circumference, it also means that another value of V, namely V_2, exists such that:

$$\|P - T_i\| = V_2 t_i \quad ; \quad \|P - T_k\| = V_2 t_k \tag{3.37}$$

By now comparing the first expressions of (3.36) and (3.37) it follows that necessarily must be $V_1 = V_2$ and, as a consequence, the second of (3.36) and (3.37) assert that P belongs also to the third circumference, the one related to the pair (t_j, t_k).

The second property which is worth showing relates the centers of the circumferences and states that they are always collinear. In case the three circumferences have intersecting point(s), the second property is evidently a direct consequence of the first one, as circumferences having common intersecting point(s) must necessarily have collinear centers. However the second property has a more general

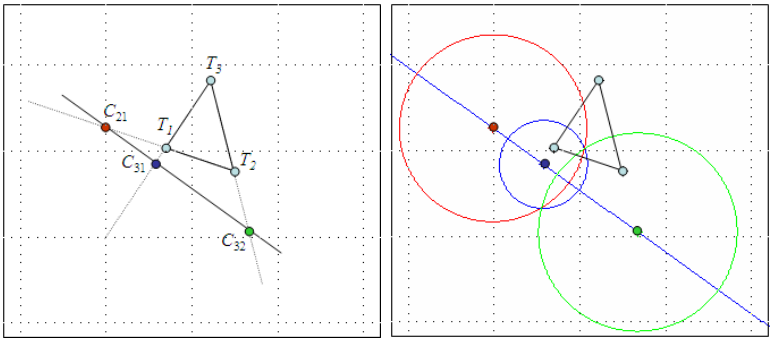


Figure 3.21: An exemplificative scenario of the 3-transponders problem with $t_1 < t_2 < t_3$

validity, notwithstanding the existence of intersecting point(s) among the circumferences. In order to demonstrate this last consideration in a clearer way, it can be convenient to reformulate the second properties by making explicit reference to an exemplificative case. To this aim, the following assumption can be adopted (without loss of generality):

$$t_1 < t_2 < t_3 \tag{3.38}$$

A scenario fulfilling (3.38) is depicted in Fig. 3.21. With reference to such a figure the second properties can be hence translated into the following statement:

$$(C_{32} - C_{21}) = k\,(C_{31} - C_{21}) \quad ; \quad k \in \mathbb{R} \tag{3.39}$$

whose correctness can be demonstrated via direct substitution of (3.34) into (3.37), and by additionally taking into account that, from definitions (3.24) and (3.33), it follows that:

$$\alpha_{32} = \alpha_{31}\frac{1 - \alpha_{21}}{\alpha_{31} - \alpha_{21}} \tag{3.40}$$

and by also considering that, from (3.31), it follows that:

$$T_3 - T_2 = L_{31} - L_{21} \quad ; \quad T_2 - T_1 = L_{21} \tag{3.41}$$

After manipulating the involved terms it turns out that (3.39) is always true and further that

$$k = \frac{1 - \alpha_{21}}{\alpha_{31} - \alpha_{21}} \tag{3.42}$$

In the light of the above properties, the analysis of the original localization problem results now to be quite simplified.

The first property states which are the only possible candidate solutions: i.e. the intersecting point(s), if any, of the three circumferences. Indeed they are the only points in the plane satisfying all the three equations (3.17) with the same value of V. Further it also suggests a convenient way for computing them: i.e. by just considering two circumferences instead of three.

The second property allows to assert that if two intersecting points are found, one is certainly outside the triangle of the transponders. This is a consequence of the following two additional considerations:

1. every center is external to the segment joining the corresponding transponders, thus implying that the whole line joining the centers lays outside the triangle; and

2. the two intersecting points are symmetric with respect to the line joining the centers, therefore one of them is certainly in the half-plane not containing the triangle.

This last aspect has a significant importance from a practical point of view. If the transponders are positioned in a way that all the AUV operations are guaranteed to be carried out inside the area identified by them, the presence of a candidate solution external to the triangle, allows to solve univocally the localization problem.

Moving from the above considerations, the following procedure can be outlined for solving the 2D localization problem with three transponders. The first step consists in computing the circumferences related to two out of the three possible pairs of ToFs. The second step is finding out (if any) the intersecting point(s) among them. If no points are found, it means that the set of collected ToFs is too affected by noise and the procedure cannot find a valid solution. If only one intersecting point is found, it is certainly external to the triangle (as it belongs to the line joining the centers) and has to be discarded (again because of the noise), under the assumption that the AUV is granted to be inside the triangle. Finally, if two intersecting points are found, the one on the half-plane not containing the triangle has to be discarded, and the only valid solution is the other one.

3.3.4 4 Transponders

As seen in the previous subsection, 3 transponders are in principle enough to univocally solve the 2D localization problem. However the unavoidable noise which is experienced during normal field activities may originate wrongly measured ToFs or even the loss of some of them. To make the system more robust to noise, it is a common practice to operate with a redundant setting, where more than three transponders are employed. As a representative case of a redundant scenario, the case with four transponders disposed in a rectangular configuration is here considered (see Fig. 3.18c).

In this case, a possible approach to the localization problem is considering the four possible triples of ToFs and solving, for each one of them, the corresponding 3-transponders sub-problem, as explained in the previous subsection. For every triple it can be demonstrated that, also in this case, only one valid solution (i.e. internal to the rectangle) can at most be generated; indeed, if within a 3-transponder sub-problem, two intersecting points are found, the one in the half-plane not containing the considered triangle always results to be also outside the whole rectangle.

Once the four sub-problems are solved, up to four candidate solutions are identified. At this point different situations may arise, depending on the number of candidates and on their values. The luckiest case occurs if the four candidates coincide, thus clearly implying that their corresponding X point represents the final estimation. On the contrary, the worst situation happens when the set of candidate solutions is empty: in this case the localization procedure terminates with a failure and no estimation is produced. Apart the above two extreme conditions, the only other one straightforward situation occurs when only one triple of ToFs could generate a valid solution. In such a case, it means that the forth ToF (the one not belonging to the triple achieving the valid solution) is probably too affected by noise; indeed the three triples containing it could not originate a valid solution. If this happens, such a ToF has to be considered as unreliable and has therefore to be discarded by the localization procedure. Therefore the resulting final estimation X is clearly the sole available candidate.

By neglecting the above three simple cases, when two or more different candidate solutions are found by solving the four sub-problems, a certain strategy has to be identified for selecting the final estimation of X. At this level different

policies may be applied, like averaging among the candidates by weighting them according to some criteria, or finding out the geometric middle point of the area by them delimited.

Among the possible choices, the proposed strategy aims to find out the best solution in a least-squared sense. More specifically by defining the following error function:

$$E(X,V) = \begin{bmatrix} \|X - T_1\| - Vt_1 \\ \|X - T_2\| - Vt_2 \\ \|X - T_3\| - Vt_3 \\ \|X - T_4\| - Vt_4 \end{bmatrix} \tag{3.43}$$

the searched solution $[X,V]^T$ is the one solving the following optimization problem:

$$\min_{X,V} \|E(X,V)\|^2 \tag{3.44}$$

As it is well known, a certain $[X^*,V^*]^T$ vector can be a minimum of the cost function $\|E\|^2$, only if it satisfies the following orthogonal condition [68, 69]:

$$J\left(X^*,V^*\right)^T E\left(X^*,V^*\right) = 0 \tag{3.45}$$

being J the Jacobian matrix of E:

$$J \triangleq [J_X \; J_V] \tag{3.46}$$

where:

$$J_X \triangleq \frac{\partial E}{\partial X} = \begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \\ u_4^T \end{bmatrix} \quad J_V \triangleq \frac{\partial E}{\partial V} = \begin{bmatrix} -t_1 \\ -t_2 \\ -t_3 \\ -t_4 \end{bmatrix} \tag{3.47}$$

and where every term $u_i \in \mathbb{R}^2$, defined as follows:

$$u_1 \triangleq \frac{X - T_i}{\|X - T_i\|} \quad ; \quad i = 1,\ldots,4 \tag{3.48}$$

is the unit vector along the direction joining X with T_i.

Before describing the adopted method for solving the optimization problem (3.44), it is worth showing that the Jacobian matrix J it is always full rank inside the rectangle delimited by the transponders. To this aim, first note how from

(3.47) it follows that J is dependent only on X and not on V . Then in order to demonstrate the above property, it is sufficient to show that the null space of $J(X)$ is the empty space, whenever X is inside the rectangle. To show this last statement, assume a certain vector $Q \in \mathbb{R}^3$ exist such that

$$JQ = 0 \tag{3.49}$$

By now considering the vector along the Q direction such that its first two components form a unit vector $q \in \mathbb{R}^2$ and the third one δ is positive, from (3.49) and definitions (3.46) and (3.47) it follows that:

$$u_i^T q = \delta t_i \quad ; \quad i = 1, \ldots, 4 \tag{3.50}$$

Now, as any left-side term is the inner product of two unit vectors, its value corresponds to the cosine of the angle between them. Therefore, by calling β_i the angle between u_i and q, and by recalling that every ToFs is always greater than zero, equations (3.50) allow to state the following conditions:

$$\cos(\beta_i) > 0 \Rightarrow \beta_i \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \quad ; \quad i = 1, \ldots, 4 \tag{3.51}$$

By now considering the line orthogonal to q, it clearly divides the plane into two half-planes, one of which containing q. Conditions (3.51) state that all the four u_i vectors must lay in the same half-plane than q. By now remembering that any u_i vector is directed along the joining line between the considered X and the corresponding transponder, the whole set of u_i can be on a single half-plane if and only if X is outside the considered rectangle.

The full rankness of J is a very important property for the considered optimization problem. It allows to employ an iterative method based on a gradient-descent technique, without caring about possible singular points, as they do not actually exist.

More specifically, for any set of measured ToFs, the proposed procedure starts from a certain estimation vector $[X_0, V_0]^T$. Then, by introducing a suitable scalar factor $\gamma > 0$ governing the speed of descent, the following iterative equation:

$$\begin{bmatrix} X_{i+1} \\ V_{i+1} \end{bmatrix} = \begin{bmatrix} X_i \\ V_i \end{bmatrix} - \gamma \left(J^T J\right)^{-1} J^T E \tag{3.52}$$

is used. As it can be easily demonstrated (for example by applying the Lyapunov's criterion to the cost function $\|E\|^2$), equation (3.52) evolves through a sequence

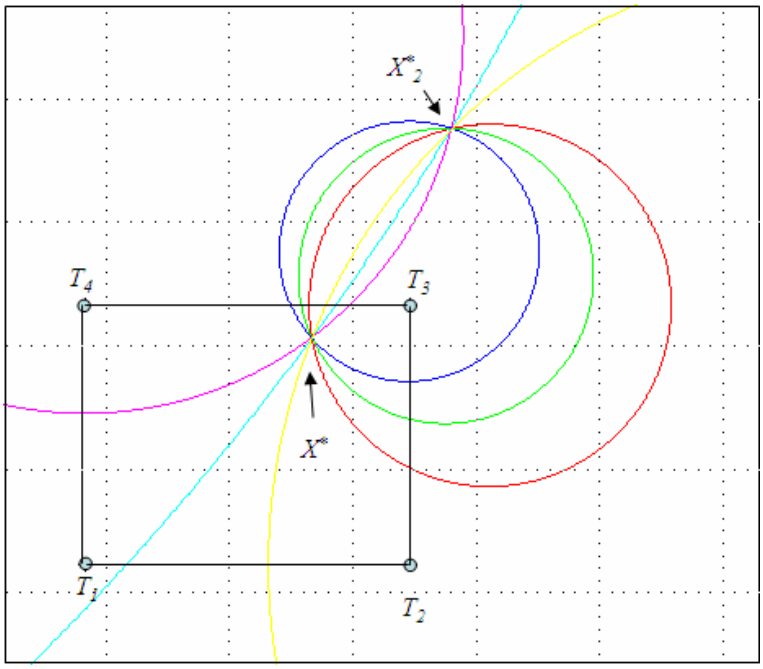


Figure 3.22: Ideal scenario leading to 4 coinciding candidate solutions after the 3-transponders problem phase

of vectors $[X, V]^T$, leading to continuously decreasing values of $\|E\|^2$. As the matrix $\left(J^T J\right)$ is always non-singular, the only terminating condition is that a vector $[X^*, V^*]^T$, satisfying the orthogonal condition (3.45), is eventually reached.

Before talking about the selections of the initial estimation vector $[X_0, V_0]^T$ and of the value of the scalar factor γ it is worth showing that the cost function $\|E\|^2$ may exhibit more than one minimum.

As an example, consider the ideal case when all the four candidate solutions, after the preliminary 3-transponders sub-problems phase, coincide in a certain point X^*. If this happens, it means that the cost function is clearly nullified for such X^* value and its corresponding V^*.

A scenario representing this situation is depicted in Fig. 3.22, where the set of six circumferences (obtained by considering any possible pair of ToFs) is reported. As expected, all the circumferences intersect in the common point X^*. However, as it can be noted by the picture, in the considered example, another point X_2^* (although external to the rectangle) exists for which the cost function is equivalently zeroed.

A similar situation occurs also if no solutions zeroing the costs can be found. In Fig. 3.23 is reported a scenario analogous to the one of Fig. 3.22, with the

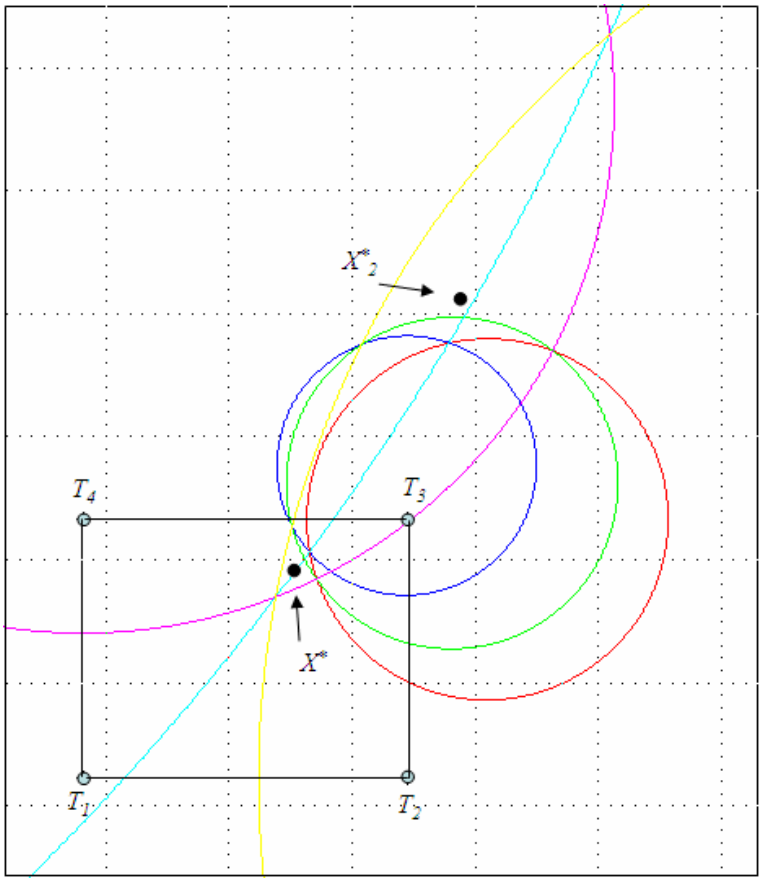


Figure 3.23: An exemplificative scenario leading to 4 different candidate solutions after the 3-transponders problem phase

only difference that one of the ToFs has been slightly perturbed on purpose for obtaining four different candidate solutions inside the rectangle (identifiable in the figure as the points where 3 circumferences intersect). In such a case, the cost function exhibits two non-zero local minima: one in correspondence of $X = X^*$, the other for $X = X^*_2$. The former is inside the region delimited by the four valid candidates; the latter is inside the region identified by the four solutions discarded during the preliminary phase.

The presence of more than one vector $[X^*, V^*]^T$ locally minimizing the cost function, clearly induces a threefold risk:

1. no "valid" $[X^*, V^*]^T$ (i.e. such that X^* is inside the rectangle) may exist;
2. more than one valid $[X^*, V^*]^T$ may exist; or
3. by iterating equation (3.52), the vector $[X, V]^T$ might be attracted by a

"wrong" $[X^*, V^*]^T$ (i.e. such that X^* is outside the rectangle).

As it concerns the first two possibilities (although a solid formal proof is still partially not available), several data so far collected through simulations and experimental trials seem to confirm that, whenever two or more valid candidates are found during the preliminary phase, a single local minimum of $\|E\|^2$ inside the rectangle does actually always exist.

As it concerns the third risk, it is clear that the selection of the initial estimation $[X_0, V_0]^T$, as well as of the value of the scalar factor γ becomes more crucial. To this respect, however, it can be noted that the choice $\gamma = 1$ provides the significant benefit of making the estimation procedure not dependent on the initial selection of V_0. In order to demonstrate the above statement, first note how, by calling D the vertical collection of the distances between a considered point X and the transponders; i.e.

$$D \triangleq \begin{bmatrix} \|X - T_1\| \\ \|X - T_2\| \\ \|X - T_3\| \\ \|X - T_4\| \end{bmatrix} \tag{3.53}$$

the iterative equation (3.52) can be reformulated as follows:

$$\begin{bmatrix} X_{i+1} \\ V_{i+1} \end{bmatrix} = \begin{bmatrix} X_i \\ V_i \end{bmatrix} - \gamma \left(J^T J\right)^{-1} J^T D - \gamma \left(J^T J\right)^{-1} J^T V_i J_V \tag{3.54}$$

Now in case $\gamma = 1$ the last term of (3.54) results to be equal to $[0\ 0\ -V_i]^T$ as a consequence relationship (3.54) becomes:

$$\begin{bmatrix} X_{i+1} \\ V_{i+1} \end{bmatrix} = \begin{bmatrix} X_i \\ 0 \end{bmatrix} - \gamma \left(J^T J\right)^{-1} J^T D \tag{3.55}$$

which shows the independence of the iterative procedure on the term V_i, and hence also on the initial selection V_0. Such a remark is quite important, as it makes the overall estimation procedure consistent with the original assumption of a null a-priori knowledge on the SoS profile.

As it finally concerns the selection of the initial point in the plane, data so far collected show that, when two or more valid candidate solutions are available, if the middle point of the rectangle identified by the transponders is selected as X_0, the procedure always converges (and after very few steps!) to the local minimum whose corresponding X^* is inside the rectangle.

3.3.5 3D Localization Problem

In this section the more general 3D localization problem is considered: The AUV is no more assumed to stay on a plane, thus its unknown position is now described by the triple $X = [x, y, z]$. Analogously the point where transponders are deployed are here identified by triples $T_i = [x_i, y_i, z_i]^T$; $i = 1, \ldots, N$. Under the reasonable assumption that the AUV knows its depth z (as it can be measured by an on-board depth meter), the results obtained in the previous sections can be just slightly modified for being adapted to the 3D scenario.

The main significant difference is registered in the basic 2-transponders subproblem, and concerns the locus of points described by the X solutions for different values of V. As it can be intuitively argued, what was a circumference on the plane, in the 3D context becomes a sphere. The above statement can be easily shown, by following the same steps as done in Section 3.3.2; further, by again posing definition (3.24), it also turns out that the center and radius of such a sphere are still given by (3.25) and (3.26) (where now every T_i term belongs to $\mathbb{R}^3$).

In order to maximally exploit the results obtained in the previous subsection, the basic idea is transforming the 3D problem into the corresponding 2D one, obtained by considering the horizontal plane where the AUV is known to be.

In this light, instead of using the equation of the sphere, it is more convenient to consider the circumference obtained by intersecting the sphere with the AUV's horizontal plane. Finding out the descriptive parameters of such a circumference is a very simple task. Indeed given a sphere with center $C_s = [x_s, y_s, z_s]$ and radius R_s, it is well known that the 2D center C_c and radius R_c of the intersecting circumference with an horizontal plane at depth z are given by:

$$C_c = \begin{bmatrix} x_s \\ y_s \end{bmatrix} \quad ; \quad R_c = \sqrt{R_s^2 - (z - z_s)^2} \tag{3.56}$$

By applying (3.56) to any possible pair of available ToFs, and by considering the resulting set of circumferences, all the considerations stated in Section 3.3.3, can be easily adapted to the 3D scenario, making it possible to solve the 3-transponders localization problem as shown before. When instead more than 3 transponders are available, a further slight modification has to be introduced: the



Figure 3.24: The employed eFOLAGA AUV

u_i term defined in (3.48), now assumes the form:

$$u_i \triangleq \left[\frac{x - x_i}{\|X - T_i\|} \quad \frac{y - y_i}{\|X - T_i\|} \right]^T \quad ; \quad i = 1, \ldots, N \tag{3.57}$$

which is no more a unit vector (as the modulus of the vector joining X and T_i now depends also on the difference $z - z_i$); rather it represents its projection onto the considered horizontal plane. Apart this minor difference, all the considerations outlined in Section 3.3.4 still hold here. Therefore also in the 3D case, the same iterative procedure can be applied, whenever two or more different candidate solutions are obtained after the preliminary 3-transponders sub-problems phase.

3.3.6 Algorithm Validation

The presented localization strategy has been initially validated through an intense simulative campaign on artificial data, properly built for testing the performance of the algorithm in many different conditions.

Successively, the algorithm has been tested on a set of real ToFs data collected by an eFOLAGA [55], the AUV developed by our research team and commercialized by Graal Tech s.r.l., while interrogating four Hydroid transponders (gently made available by the NATO Undersea Research Center).

Once validated through simulations, the algorithm has been implemented on a real eFOLAGA (see Fig. 3.24) and several field test have been performed in the Gulf of La Spezia, Italy, as part of the activities of the EU-supported research project CO^{3}AUV (see 1.1.2), in which our research group is currently involved.



Figure 3.25: Localization data collected during an underwater navigation phase; the harbour scene (top) - detail of the AUV path (bottom)

Preliminary tests were executed with the vehicle on surface, for using the GPS as ground-truth. Collected results show a very good level of accuracy, with an average achieved precision within the range of resolution of the GPS receiver on-board the AUV. Successive trials were then carried out with the vehicle navigating at various depths in different shallow-water areas. In Fig. 3.25 the output of the localization procedure computed by the AUV during an underwater navigation phase is shown with two level of zoom. The yellow stars represent the transponders; the green (yellow) markers represents the estimated AUV positions computed when 4 (3.20) ToFs were available.

Chapter 4

Intervention Autonomous Underwater Vehicles

Nowadays, many important applications of underwater robotics in different fields such as marine salvage, marine science and offshore industry, do not only need exploration capabilities, but also intervention skills. While traditionally intervention tasks were constrained to ROVs provided with manipulators, very recently the introduction of the so-called I-AUV (i.e. Intervention AUV) opened the door to perform intervention tasks autonomously. Without the need for the umbilical cable, I-AUVs can be operated from ships of opportunity, not needing dynamic positioning capabilities and hence reducing drastically the operational cost. I-AUVs have the potential to become an enabling technology opening the door to new exciting applications. However, this new technology poses new challenging problems to the scientific community like the joint control of the vehicle-manipulator system or, from the systems engineering point of view, how to design and implement the whole system architecture.

In the field of I-AUV, the earliest works date back to 1995 and were carried out by Wang [74] with the OTTER I-AUV, with the aim of accomplishing object retrieval in an underwater scenario. Successively, the pioneering project that developed the basic technologies for cooperative arm operation in an underwater scenario was AMADEUS [75].

As for what concerns the vehicles, one of the first I-AUV that was developed is ALIVE [76], while SAUVIM [77] (equipped with one AMADEUS arm) and RAUVIM [78] are certainly the most advanced prototypes in the field.

4.1 Main Challenges and Applications of I-AUVs

4.1.1 Main I-AUVs Applications

The applications for I-AUVs are basically the same of "work class" ROVs, among which some are derived from the offshore industry such as the deployment or recovery of acoustic transponders, actuation of valves, cable cutting and debris removal. Other applications are instead of general interest, like the recovery of airplane "black box" and recovery of lost objects (e.g. objects from ship wreck / assistance in ship wreck salvage). Finally, other applications are derived from the oceanic, biological or geological fields and require the collection of samples from sea bottom (a specific rock, coral, etc.).

4.1.2 Main Challenges and Critical Issues

While all the challenges presented in Section 3.1.2 for the generic AUVs are still valid, the presence of the arm and its interaction with the vehicle introduce the problem of the coordination of these two structures within a manipulation scenario.

The major problem that the coordination poses is how to effectively exploit all the extra d.o.f. that such composed structure offers. Indeed, even nowadays, the trend is to use the vehicle and the arm in consecutive steps, i.e. first the vehicle reaches the zone for the intervention, then the arm alone operates and accomplishes the intervention job, while the vehicle essentially keeps its position. This is the approach used both in SAUVIM and RAUVI.

However, the whole structure usually needs to accomplish different tasks, as for example avoiding obstacles, keeping a good manipulability measure for the arm and avoiding its joint limits, possibly keeping a certain distance from the object to be manipulated and so on. These kind of tasks can be assigned to the arm or the vehicle separately. For example, it is natural to assign the "distance to the object task" to the vehicle, while the "manipulability measure" and "joint limits" tasks can be assigned to the arm. This is what is done in SAUVIM, where a task priority framework has been used to manage multiple tasks at the arm level in order to automatically correct the required task according to the priority of the situation. For example, when approaching a singularity, in order to prevent unwanted motions and system instabilities, the priority of the control

system becomes maintaining the distance from the singularity over a predefined threshold, rather than following the input task.

However, the approach proposed in SAUVIM still does not exploit the d.o.f. of the vehicle to accomplish such tasks. Thus one of the main challenges in the development of I-AUVs, at least from the control point of view, is to surpass these sequential approaches, providing instead a fully coordinated algorithm that is able to execute concurrently multiple tasks, exploiting both the vehicle and the arm d.o.f..

4.1.3 Proposed Algorithms

Based on previous works of the research team [79, 80, 81], the problem of suitably coordinating the arm and the vehicle for the approach phase prior to the grasp is developed in Section 4.2. The main goal of the research effort, within the scope of the TRIDENT project (see Section 1.1.1), is to create an efficient coordination algorithm that can surpass the previous sequential approaches, where the arm and the vehicle are moved only one at a time, following a pre-programmed sequence.

In order to overcome this approach, an efficient way to exploit all the d.o.f. offered by the vehicle and the arm is presented. The algorithm is based on the idea that multiple tasks can be executed concurrently, once a priority has been established between them. Each task is executed inside the kernel space of the higher priority tasks, in order to not induce any side effect on them. In addition to the classic equality tasks, the proposed approach also introduces inequality constraints, and illustrates an efficient way to deal with them in the task priority based framework.

Moreover, in order to exploits the d.o.f. of the whole structure, a dynamic programming approach has been exploited. The arm and the vehicle are thus seen as two blocks along which the dynamic programming is executed. In this way, the control law of the arm becomes parametrized by the vehicle's velocity vector, and thus the vehicle is able, indirectly, to contribute to tasks that are strictly related to the arm like the joint limits task. Moreover, the adoption of the dynamic programming, run along the arm-vehicle chain, naturally allows to use the vehicle only when strictly necessary, automatically establishing a kind of sub-system prioritization. This last fact becomes important when taking into consideration energy saving aspects, since moving the vehicle is obviously more

power demanding than moving the arm alone.

The task priority based approach is certainly not new, and dates back to the set of seminal works [82], [83], [84], [85], where a prioritized rate task is imposed to be satisfied within the null space of all its preceding higher priority ones, and has been applied in different scenarios, ranging from a single manipulator to mobile manipulators and humanoids. A systematic framework for managing a generic multiplicity of prioritized rate tasks was later proposed and developed in [86].

In the very recent generalizing work [87], it is claimed that, within the various application contexts of the task priority based control, the aspects related with the unavoidable existence of inequality constraints have never been systematically tackled while preserving their true meaning, since they have been very often substituted by more restrictive equality objectives, arbitrarily chosen within the inequality ranges, causing an unnecessarily over-constraining of the robotic structure to be controlled.

To this respect, the above criticisms within [87] are certainly correct for the cases when inequality objectives are translated into equality ones. However, they may instead become partially questionable whenever extended, as just done still in [87], to the cases where the potential fields translating the inequality objectives are located at the highest priorities, as done in [88]; or in alternative, when they are real-time changed into highest priority ones upon the event of some inequality constraint violation, and vice versa, as just done (smoothly) in [89], [80] and [90].

In fact, provided that the potential fields translating the inequalities are chosen as *smooth* penalty functions with a null support extended within the inequality ranges, the two mentioned cases obviously result coinciding. Then, the criticisms provided in [87] substantially cease to exist, since translations into restrictive equalities are not anymore performed, while the concerns for possible chattering at the frontiers of the null supports actually disappear, provided that an adequate level of smoothness is guaranteed.

The above consideration actually constitutes the core idea for the development, within the proposed algorithm in Section 4.2, of an alternative systematic procedure for managing prioritized rate-tasks encompassing inequality objectives and meanwhile resulting computationally simpler, since based on the solution of a sequence of linearly constrained least square problems (on the contrary, the approach in [87] requires the solution of a linear inequality system for each inequality objective, computationally more expensive).

Finally, regarding subsystem prioritization, for the simple classical case of a single equality objective to be achieved, such type of prioritization can be introduced by applying suitable weights to the system velocity components needing to be slow. This approach leads to the use of the so-called weighted pseudo-inverse of the corresponding Jacobian matrix. However, the use of such technique cannot in general be easily extended to the case of multi-prioritized objectives.

On the contrary, the proposed approach, relying on dynamic programming techniques, introduces the required subsystem prioritization in an automatic way.

4.2 A Dynamic Programming Approach to the Free Floating Manipulation Problem

As presented in Section 1.1.1, the TRIDENT project is composed by two distinct phases, one regarding the survey of the area, and one regarding the manipulation of objects on the sea floor. It is within this last phase that our research group is involved.

Indeed, after the object of interest has been located, the I-AUV will start the approach phase towards it. When sufficiently close, and with the camera having the object on its field of view, the control will be switched from the navigation control law to the coordinated free floating manipulation one, which is the subject of this research.

Such control law is mainly responsible for guaranteeing the maximum of dexterity during the execution of intervention tasks, like object grasping, object manipulation and possibly object transportation.

Typically, such controller were designed in order to make a sequential use of the vehicle and the hand-arm system (i.e. the vehicle moves while the arm is resting on it and vice versa); thus leading to sequential coordination among the two which, though generally acceptable within applications using vehicles that are heavy compared to the mounted arm, in TRIDENT's light-weight case, this might severely limit the level of dexterity that such more agile mechanical interconnections would actually allow.

Instead, the proposed control algorithm explicitly exploits the extra d.o.f. that the floating vehicle actually adds to the system, and actually optimize the intervention performances within a set of situations that are certainly much larger

than those allowed by the sole hand-arm system, when supposed operating on an approximately fixed base.

The here proposed control architecture is based on very recent results of our research group [79, 80, 81]. Provided that the overall control problem is formerly interpreted as a cooperative distributed control problem to be defined between the vehicle and the arm-hand system, the results clearly show how the overall structure can immediately exhibit a "self-organizing behaviour". Hence, the task execution can be guaranteed within the required performance if a moderate real-time data exchange is allowed between the two isolated controllers of the two composing agents.

To better understand the algorithm, first Section 4.2.1 introduces all the sensors and quantities of interest within the TRIDENT's system. Then Section 4.2.2 outlines all the control objectives to be achieved cooperatively by the arm-vehicle system. Section 4.2.3 illustrates how the proposed solution will increment the "agility" of the system, surpassing the previous sequential approaches. The successive Section 4.2.4 presents the overall control architecture. Section 4.2.5 details the proposed algorithm, starting from a centralized version and concluding with the distributed Dynamic Programming based one. Finally, Section 4.2.6 presents some preliminary simulations that show the good level of coordination achieved with the proposed algorithm, also in case of vehicle velocity disturbances.

4.2.1 System Organization

This section briefly reports how the system and sensor information is organized within the TRIDENT's system. Moreover, most of the interesting quantities from the control point of view are also introduced.

The system is composed by a fully actuated vehicle, capable of moving along all the 6 Cartesian directions, and a 7 d.o.f. arm, installed in the vehicle's lower part. The arm is also equipped with a torque-force sensor located at its wrist, which is then connected to a three fingered sensorized end effector. The overall structure along with some geometric quantities are depicted in Fig. 4.1.

Without going into the mechanical details of the vehicle and the arm, some important considerations are here reported:

- the mass and inertia of the vehicle are approximately ten times larger than those of the arm (where the arm inertia is intended as its maximum value

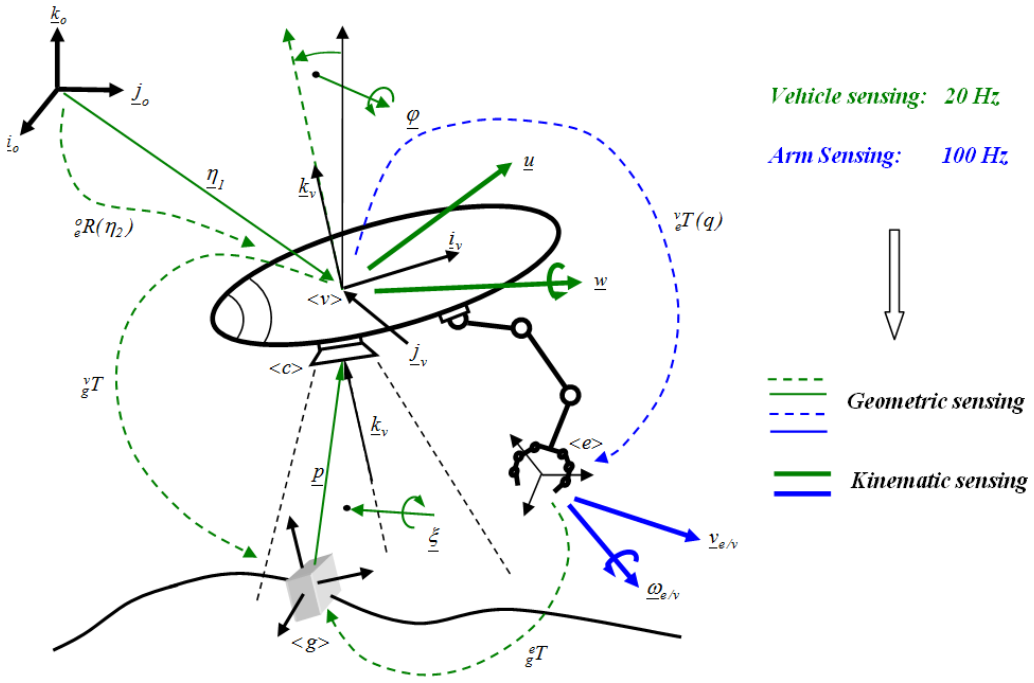


Figure 4.1: Overall structure of the I-AUV used in the TRIDENT project

attained when completely extended);

- the vehicle is neutral, with its center of gravity (COG) vertically aligned under its center of buoyancy (COB) when the vehicle has an horizontal attitude.

System Geometry and Geometric Sensing

In the lower part of the vehicle a stereo vision system is mounted, capable of estimating the position and orientation of a goal frame $\langle g \rangle$ attached to an object to be grasped w.r.t. the vehicle frame $\langle v \rangle$, i.e. the transformation matrix ${}^{v}_{g}T$. Such information is computed by the vision system at a frame rate of $f_c = 20\,\text{Hz}$.

The vehicle orientation w.r.t. the inertial frame $\langle 0 \rangle$ is obtained via the use of an inclinometer and the integration of the vehicle's angular velocities as provided by an on-board gyroscope. Following the Fossen's notations [91], the vector $\eta_2 \triangleq [\psi, \theta, \phi]^T$ is then estimated at a rate of 20 Hz. Thus in turn allowing a 20 Hz rated measures of the associated orientation matrix ${}^{v}_{0}R\,(\eta_2)$ of the vehicle fixed frame $\langle v \rangle$ with respect to an assigned inertial frame $\langle 0 \rangle$.

Finally, the arm's joint position are computed by the arm controller on the basis of the joint encoders, thus outputting the vector of joint positions q at a rate of 100 Hz. With the geometric knowledge of the arm the transformation matrix ${}^{v}_{e}T$ is then calculated.

Kinematic Geometry and Kinematic Sensing

As it concerns the measurement of kinematic variables, the vehicle is equipped with a gyro, providing the angular velocity ω projected in the frame $\langle v \rangle$ (i.e. it provides $v_2 \triangleq {}^{v}w$ according to Fossen's notations).

A DVL provides measurements of the linear velocity, still projected in the vehicle frame (i.e. it provides $v_1 \triangleq {}^{v}u$ according to Fossen's notations). Both these quantities are available at a rate of 20 Hz.

Moreover, via measurements of the joint velocity vector $\dot{q}$, it is possible to compute the linear and angular velocities for the arm's end-effector with respect to the vehicle frame and projected onto it, via use of the well known kinematic relationship between them, i.e.

$$s = \begin{bmatrix} {}^{v}w_{e/v} \\ {}^{v}v_{e/v} \end{bmatrix} = {}^{v}J_{e/v}(q)\dot{q} = J(q)\dot{q} \tag{4.1}$$

As the joint positions are updated with a rate of 100 Hz, such relationship can be updated with the same rate as well.

4.2.2 Control Objectives

Control objective generally means the goal of making a system achieving and then maintaining during time a certain condition. Typically the condition characterizing a control objective corresponds to the zeroing of an error quantity (for instance a trajectory tracking error or a final positioning one) but can also correspond to ultimately having some system variables staying within specified bounds, generally related with the system safety and/or its operational capabilities.

Control objective of the first type may be denoted as precision (control) objectives, while those of the second type may instead be denoted as set (control) objectives.

Set control objectives might be in turn classified in different ways, as for instance safety, enabling, operational-improving, in dependence of the specific reasons they have been introduced.

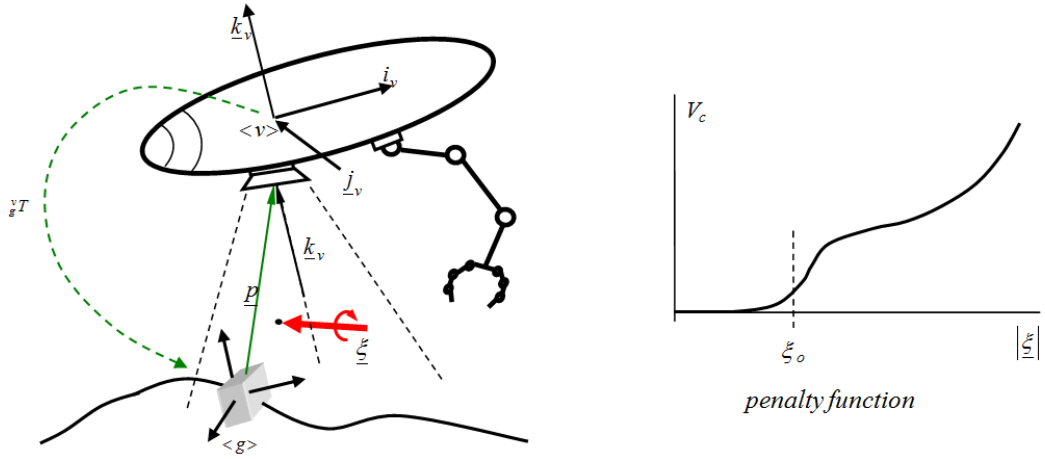


Figure 4.2: The camera centering set control objective

A system to be controlled must generally satisfy to more than one set control objective, which obviously should not be conflicting one with the other. This in the sense that the set of ultimate system configurations associated to the fulfillment of each one of them should admit a common non void intersection, inside which the precision objectives can consequently be accomplished.

In the following, the various control objectives to be achieved by the system will be presented, subdivided into the two groups of set and precision objectives respectively.

Vehicle Set Objectives

Camera Centering Once the object to be grasped has for the first time entered the camera visual cone, its position-attitude ^v_gT can be evaluated by the stereo vision system, and the grasping phase can begin. Naturally, the object must be kept inside the visual cone of the camera until the end of the grasping operation and possibly centered.

Let us define p as the distance vector pointing from the origin of frame $\langle g \rangle$ toward the origin of the camera frame $\langle c \rangle$, evaluated with components on $\langle v \rangle$. Then the rotation vector ξ formed by vector p w.r.t. the vector $^v k_c = {}^v k_v = [0\,0\,1]^T$ of the camera frame $\langle c \rangle$ can be computed, and its norm $\|\xi\|$ represents a measure of the centering of the object within the visual cone.

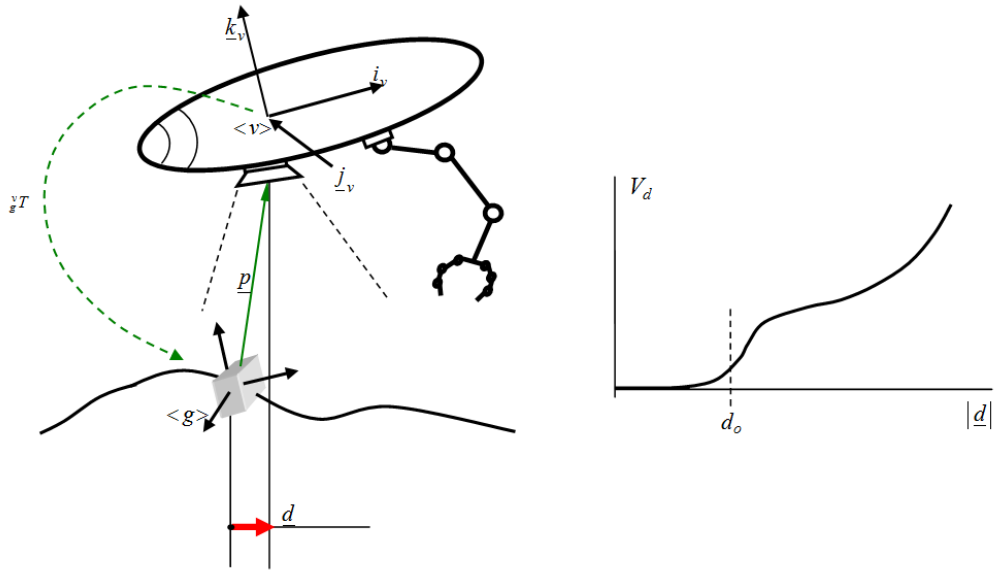


Figure 4.3: The camera distance set control objective

Thus, a control set objective of the form

$$\|\xi\| < \xi_0 \tag{4.2}$$

can be defined, and its fulfillment would guarantee keeping the object within the camera visual cone, provided that the threshold ξ_0 is set lower than the aperture semi angle of the vision system.

Such task is depicted in Fig. 4.2. Note that the above control objective is defined as a set one, since for the camera operability even a gross centering inside the visual cone is actually sufficient. Just due to this a penalty function of the type indicate in figure can be adopted for providing a measure of the objective achievement in its set sense.

It is also trivial to see that such set objective is only affected by the vehicle's control variables.

Camera Distance A further constraint, related with both arm manoeuvrability and camera operability, is to have the camera vertically aligned with the object to be grasped, as indicated in the Fig. 4.3.

A requirement, this last, that can be easily translated into an associated set objective by simply considering the horizontal vector d, now intended with

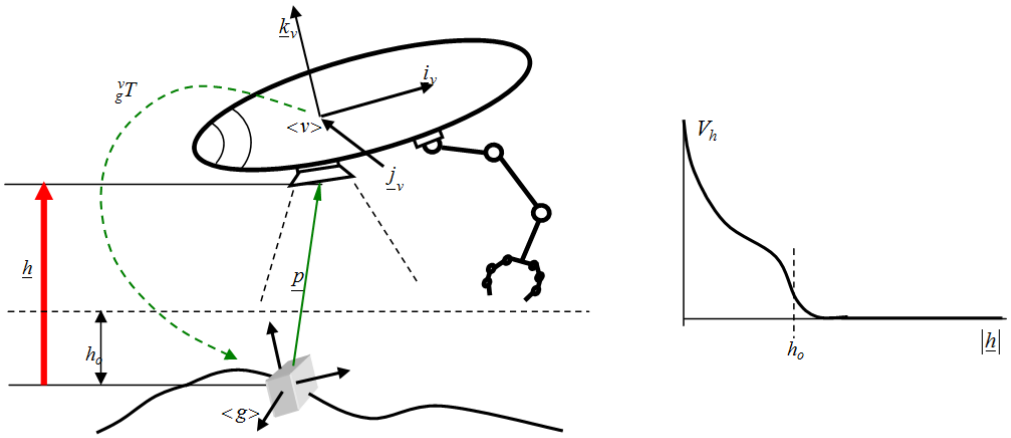


Figure 4.4: The camera height set control objective

components on the inertial frame $\langle 0 \rangle$, and then requiring

$$\|d\| < d_0 \tag{4.3}$$

In particular note how vector d can be evaluated from vector p, having its components on $\langle v \rangle$, via the simple relationship

$$d = \left(I - {}^0k_0 {}^0k_0^T\right) R(\eta_2) p \tag{4.4}$$

with $k_0 = [0\,0\,1]^T$. Also such set objective refers to the vehicle variables only and its achievement can be measured by a penalty function as in the associated figure.

Camera Height Similarly as before, in order to guarantee a good camera operability, it is needed that the camera is vertically located above the object to be grasped at a given minimum distance. This requirement can be easily translated into an associated set objective by simply considering the vertical vector h, it also intended with components on the inertial frame $\langle 0 \rangle$, and then requiring (as depicted in Fig. 4.4)

$$\|h\| > h_0 \tag{4.5}$$

In particular vector h can be evaluated from vector p, via the relationship (complementary to (4.4))

$$h = \left({}^0k_0 {}^0k_0^T\right) R(\eta_2) p \tag{4.6}$$

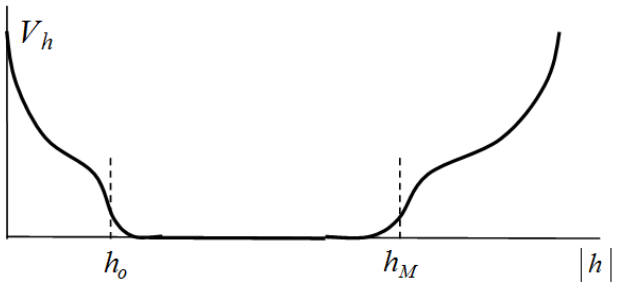


Figure 4.5: The modified camera height set control objective

Also such set objective refers to the vehicle variables only and its achievement can be now measured by the penalty function indicated in Fig. 4.4. However, since the camera must also be within a maximum height from the object, (4.5) can be modified as follows

$$h_0 < \|h\| < h_M \tag{4.7}$$

where a maximum height h_M below which the camera must ultimately stay has been added and in this case the relevant penalty function accordingly modifies into the one indicated in Fig 4.5.

Arm Set Objectives

Joint Limits The joints of the arm are constrained to remain within specific physical bounds. In order to avoid the risks of damages possibly occurring in case of collisions with such physical bounds, typically their limit values are artificially restricted to be lower than the physical ones, to be never trespassed, thus leading to the following group of simple set control objectives

$$Q_{im} < q_{im} < q < q_{iM} < Q_{iM} \; ; \; i = 1, \ldots, 7 \tag{4.8}$$

where Q_{im} and Q_{iM} represent the physical lower and upper bounds, while q_{im} and q_{iM} the restricted control bounds. The associated functional is show in Fig. 4.6.

Manipulability Another important requirement for the proper functioning of the arm is to maintain a high level of dexterity. This can be achieved by maintaining the so-called manipulability measure of the arm, defined as

$$\mu \triangleq \det\left(J(q)J^T(q)\right) \geq 0 \tag{4.9}$$

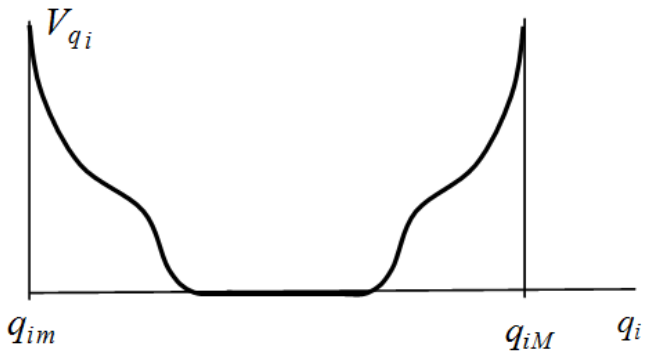


Figure 4.6: The joint limits set control objective

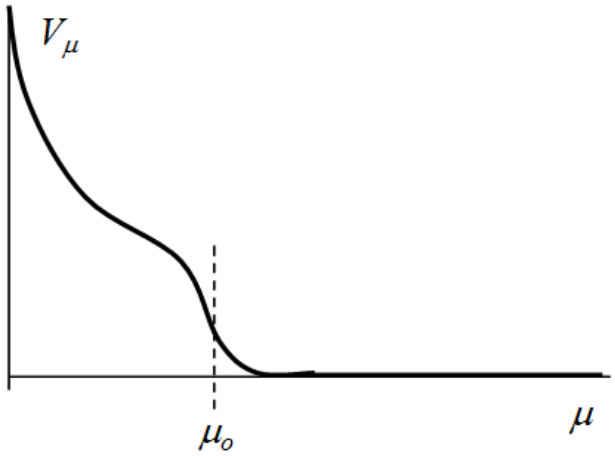


Figure 4.7: The manipulability set control objective

above a certain threshold μ_0. Thus, the set control objective aimed at keeping a good dexterity of the arm is simply defined as

$$\mu \geq \mu_0 > 0 \tag{4.10}$$

and its functional cost is represented in Fig. 4.7.

Vehicle Precision Objectives

Horizontal Attitude Since the natural posture of the available vehicle once per-se considered is the horizontal one, then reasonable engineering judgments, mainly intended to avoid energy consumption in case of a prolonged staying with non-horizontal attitudes, suggest to require the overall system to conduct its

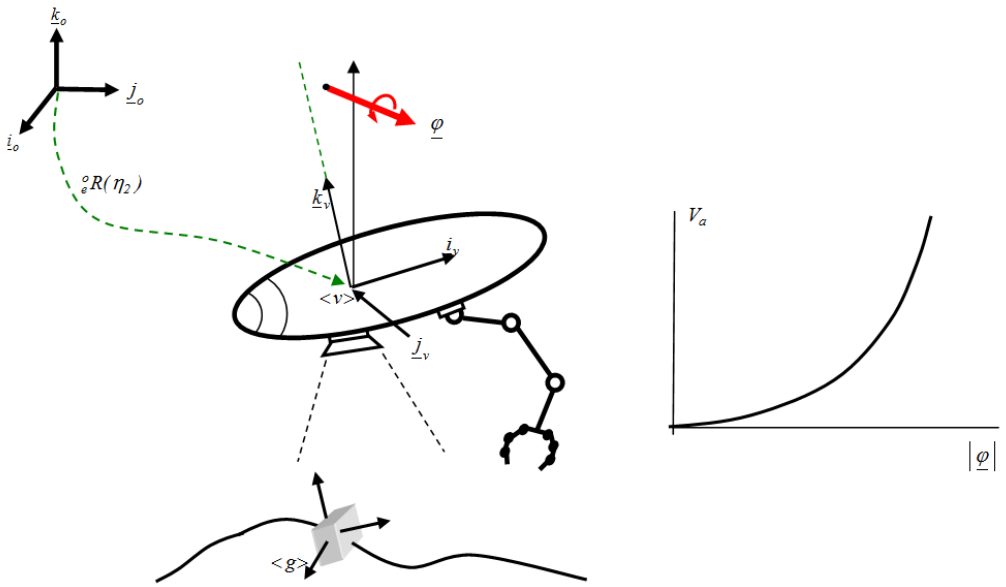


Figure 4.8: The vehicle's horizontal attitude precision control objective

floating manipulation activities while maintaining the vehicle horizontal (despite the slight perturbations that could be induced by the arm motions). This also contemporarily further improves the operability conditions of the endowed vision camera.

Such control objective is a precision one and simply corresponds to require

$$\|\varphi\| \to 0 \tag{4.11}$$

where φ is the rotation vector, with components on vehicle frame $\langle v \rangle$, that its unit vector k_v forms with the vertical unit vector k_0 of the absolute frame $\langle 0 \rangle$.

Note how such rotation vector, since always orthogonal to ${}^v k_v$, necessarily exhibits a permanent null value in correspondence of its third component (i.e. it lies on the x-y plane of $\langle v \rangle$). Moreover, since it is also orthogonal to ${}^v k_0$, this implies that also it is always horizontal in the absolute space (see the associated Fig. 4.8).

Finally, note how such task is affected only by the vehicle's control variables.

Arm-Vehicle Precision Objectives

End Effector Approach Once all the above objectives have been met, the remaining d.o.f. can be exploited for the final task, which is the most important

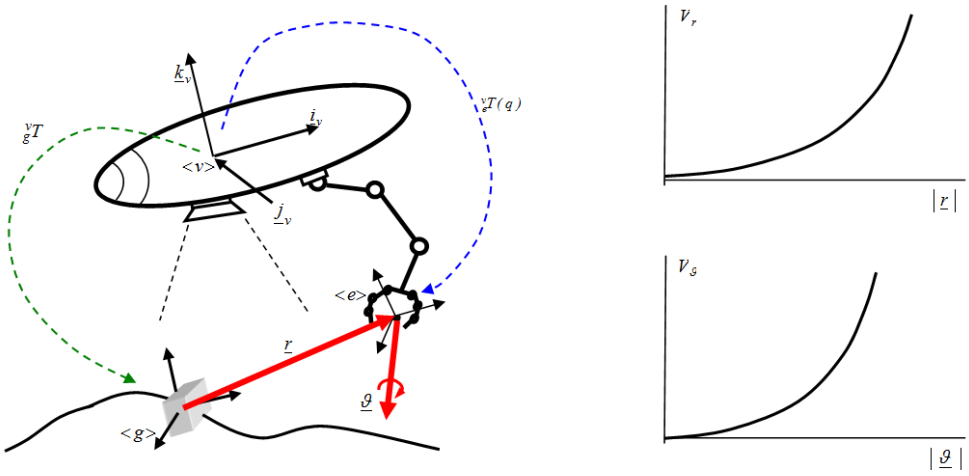


Figure 4.9: The end effector approach precision control objective

from the mission point of view, i.e. the end effector approach to the object. Indeed, this approach phase, with the fingers open, enables the fundamental task of grasping the object to be recovered from the sea floor.

The control objective associated to the approach phase can be easily described as the one of having the transformation matrix ${}^e_gT$, of the goal frame $\langle g \rangle$ with respect to the end-effector one $\langle e \rangle$, asymptotically tending to the identity, that is the following precision control objective

$$ {}^e_gT = {}^v_eT^{-1}\,{}^v_gT \rightarrow I \tag{4.12} $$

which can in turn be translated into the following equivalent couple of precision ones

$$ \|r\| \rightarrow 0 \tag{4.13} $$

$$ \|\theta\| \rightarrow 0 \tag{4.14} $$

where r and θ respectively represent the distance vector of the origin of frame $\langle g \rangle$ from that of $\langle e \rangle$ and the rotation vector of $\langle g \rangle$ with respect to $\langle e \rangle$, both represented with components on the vehicle frame $\langle v \rangle$ (see the associated Fig. 4.9 and note how the associated penalty function are zeroed only in correspondence of the zeroing of the corresponding errors).

4.2.3 Introducing System Agility

As it has been already remarked at the beginning of the present section, the group of set and precision objectives assigned to the system, or to parts of it, must always satisfy to a number of topological conditions, which are in turn directly influencing the operability of the overall system itself, namely the following:

1. each set objective admits a non-void continuous set of system configurations satisfying the set objective itself (set objective zone);
2. the intersection of all set objective zones must be it also non-void continuous (set objective non-conflicting zone);
3. each precision objective admits a non-void continuous set of system configurations where the precision objective can be eventually achieved (precision objective zone);
4. the intersection of all precision objective zones must be it also non-void continuous (precision objective non-conflicting zone);
5. then the intersection of the zones at points ii), iv) must they also admit a non void continuous intersection (non-conflicting zone).

Then, provided the topological conditions are satisfied, then following other operative conditions must also be satisfied:

1. any set objective must be achieved prior of any precision objective involving (even partially) the subsystem (or possibly the whole system) variables to which the set objective itself refers.

As it can be realized the fulfillment of the operative conditions serves for guaranteeing the system to be in the right conditions for in turn efficiently achieving the precision objectives.

The fulfillment of the topological conditions instead serves for guaranteeing that all set objectives can be achieved, and in turn all precision ones. Guaranteeing the fulfillment of the topological conditions is generally a question regarding both the overall system design and its required capabilities, as well as the definition itself of the types and number (generally the strictly necessary ones) of set objectives. Such process is generally difficult to systematize, and typically

different inspective iterations during the system design process can ultimately give the required answer. As a matter of fact, for the TRIDENT system, it is not difficult to verify by inspection how the previously introduced set and precision objectives actually satisfy the above introduced topological conditions.

As regards instead the operative conditions, at a first glance the need for their fulfillment seems hinting the idea of realizing an overall control policy substantially subdivided into two gross time-sequenced phases: the first one, where specifically dedicated motions sequences are activated for driving the system configuration toward the set objective non-conflicting zone (with such motions to be then ceased when the zone is eventually reached), followed by the second one, where other dedicated sequential motions come into play, for driving the system configurations toward the overall non-conflicting zone. Moreover, inside each phase there might even be other sequenced sub-phases allowing the relevant control objectives to be sequentially achieved.

Once the system configuration is ultimately within the set objectives non-conflicting zone, each set objective is still suitably monitored. This allows to be always ready to switch to the first phase, in case some of them may result, for some reason, violated.

All this can be certainly implemented and a classical approach generally assumes an off-line planning phase preceding all the motion phases, typically yielding the mentioned sequential execution of separate motion tasks to be successively executed. However, this generally leads to too-time consuming manoeuvres (especially in the unfortunate cases where for some reasons some of them have to be repeated), to a substantial lack of autonomy and, consequently, also to a scarce system "agility", where this term is here intended as the contemporary presence of both autonomy and dexterity.

In order to achieve the above introduced agility concept, the proposed control architecture is based on the general idea of attempting to exploit, at each time instant, any existing motion redundancy for coherently and concurrently driving the system toward the achievement of as many as possible set objectives, while contemporarily also best progressing toward as many as possible precision objectives, every time that residual motion redundancies are left available at the current time.

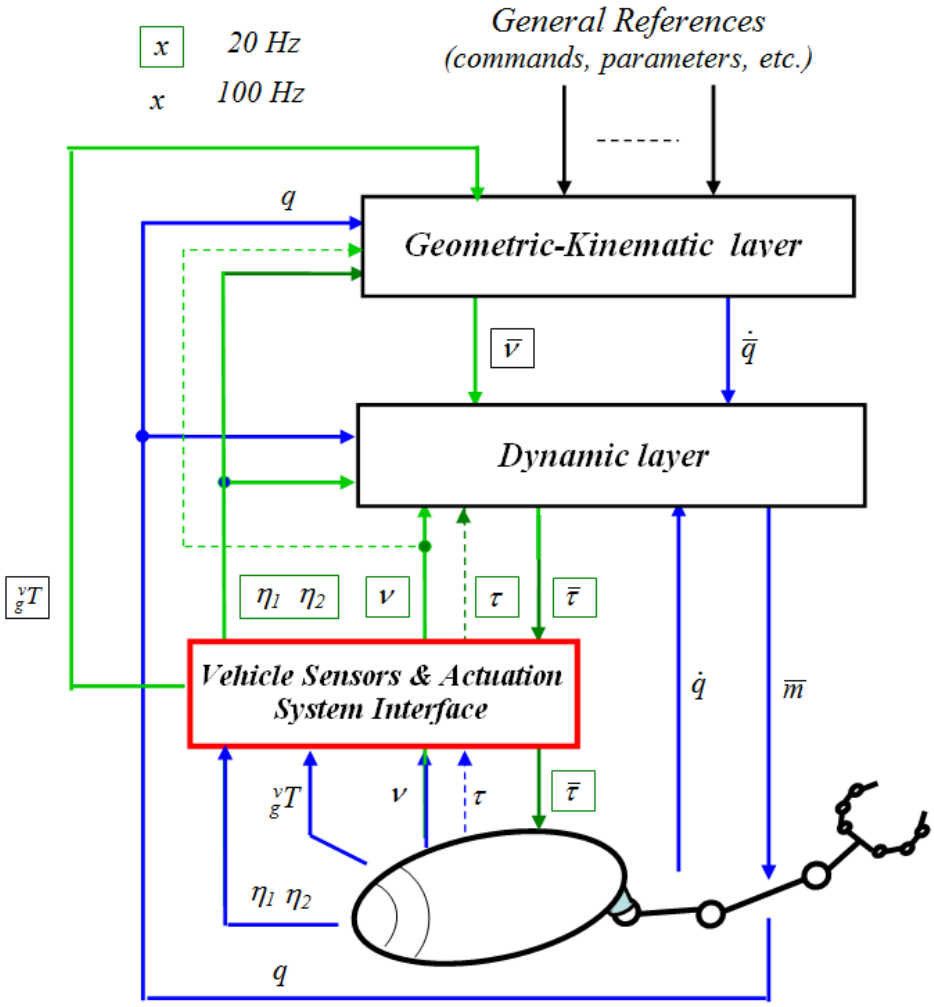


Figure 4.10: The control architecture subdivided in three layers. Data symbols enclosed within a square just correspond to those acquired with rates lower than 100 Hz

4.2.4 Control Architecture

After having listed which are the control objectives that the overall structure will try to achieve, the general control architecture is here briefly presented, as shown in Fig. 4.10.

By analyzing it along a top-down direction, the following layers are found:

Geometric-kinematic control layer It real-time receives in input the sensed geometric data described in Section 4.2.1 corresponding to: the vehicle yaw, pitch, roll attitude vector η_2; its depth z; possibly the vehicle absolute position vector; the relative position-attitude transformation matrix ${}^{v}_{g}T$ of

the object; possibly also the vehicle height h (in case an altimeter is also available); the arm joint vector q, directly related with the arm posture with respect to the vehicle.

Based on the real-time acquired data, it performs the processing required for suitably outputting the reference velocity signals $\bar{v}$, $\dot{\bar{q}}$, to be tracked by the vehicle and the arm, respectively.

Dynamic control layer It receives in input substantially the same real time sensed geometric data of the upper-lying geometric-kinematic layer, plus all the real time sensed kinematic data described in Section 4.2.1. Moreover the overall reference signals $\bar{v}$, $\dot{\bar{q}}$ provided by the upper-lying geometric-kinematic layer, are also input to the dynamic layer.

Based on its real-time acquired data it performs the real-time processing required for real-time suitably outputting the resultant force-torque vector $\tau \triangleq [\tau_1, \tau_2]$ (of resultant force τ_1 and torque τ_2, both with components on $\langle v \rangle$) to be applied in correspondence of $\langle v \rangle$ by the coordinated action of the thrusters, plus the joint torque vector m to be applied at the arm joints.

Actuation and physical layer In the Fig. 4.10 the lowest layer is finally represented by the actual physical vehicle-arm system, providing all variables sensed by the Geometric and Kinematic sensor systems discussed within Sections 4.2.1 and 4.2.1 respectively. Moreover, at this logical level, there is also the so-called thrusters allocator, whose role is transforming the commanded overall force-torque τ for the vehicle into an equivalent set of thrusters actions.

In this work, only the first level is actually tackled in the next section. Details of the dynamic control layer can be found in [92].

4.2.5 Geometric-Kinematic Control Layer

Before going into the specific details of the present section some preliminary and general considerations regarding the set of variables and relationships to be gathered within the here considered layer are actually necessary, as hereafter indicated.

General Structure of the Jacobians

Let us consider a generic objective, characterized by the following variables

$$\kappa \triangleq {}^{v}\boldsymbol{\kappa} \tag{4.15}$$

$$\sigma \triangleq \|\boldsymbol{\kappa}\| \tag{4.16}$$

$$\boldsymbol{n} = \boldsymbol{\kappa}/\sigma \tag{4.17}$$

$$n \triangleq {}^{v}\boldsymbol{n} \tag{4.18}$$

$$V(\sigma) \triangleq \alpha(\sigma)\tilde{V}(\sigma) \tag{4.19}$$

where $\boldsymbol{\kappa}$ is the error vector, σ is its norm, $\boldsymbol{n}$ its versor and $V(\sigma)$ a cost function, characterized by a sigmoid window $\alpha(\sigma)$ and a quadratic function $\tilde{V}(\sigma)$.

The geometric time derivative of such vector depends on the chosen point of view. Without going into the mathematical details of the derivation of the formulae (which can be found in [92]), if the vehicle frame is selected, then

$${}^{v}(D_v\boldsymbol{\kappa}) = \chi_v = H_v y \tag{4.20}$$

with $y \triangleq \mathrm{col}(\dot{q}, \boldsymbol{v})$ and

$$H_v = nn^T J_v + K_v \tag{4.21}$$

Similar equations can be obtained if the observer is placed on the inertial frame $\langle 0 \rangle$:

$${}^{v}(D_0\boldsymbol{\kappa}) = \chi_0 = H_0 y \tag{4.22}$$

and

$$H_0 = nn^T J_0 + K_0 \tag{4.23}$$

Since the position of the observer can be either on the vehicle frame or the inertial frame, in the following these dummy notations will be used

$${}^{v}(D\boldsymbol{\kappa}) = \chi = Hy \tag{4.24}$$

$$H = nn^T J + K \tag{4.25}$$

Finally, the present subsection can be concluded by adding to the above Jacobian relationships the one regarding the rate of change of the error $\dot{\sigma}$ associated to an error vector $\boldsymbol{\kappa}$ (i.e. to its amplitude). For such rate it is immediate to see how for $\boldsymbol{\kappa} \neq 0$ the following row-Jacobian relationship (still in dummy notations) is found

$$\dot{\sigma} = n^T H y = n^T J y \tag{4.26}$$

The above row-Jacobian relationship is obviously totally independent from the observing frame since σ is a scalar quantity. Moreover note in the above how the second equality follows directly from the orthogonality of n with respect to the orthogonal projected operator N.

Error Rates Structure

In correspondence of a generic error σ it is obviously desired that the corresponding penalty function $V(\sigma)$ is eventually zeroed, thus meaning that corresponding control objective (set or precision) is ultimately achieved. Such need can be kinematically expressed, for a generic penalty function, as the following time behaviour

$$\dot{\sigma} \quad \rightarrow \quad \dot{V} \triangleq \frac{dV}{d\sigma}\dot{\sigma} < 0 \quad \text{wherever} \quad \frac{dV}{d\sigma} \neq 0 \tag{4.27}$$

which means the desire of applying to the error σ corresponding to a generic control objective a variation rate guaranteeing a continuous decrease of the associate penalty function toward its zeroing. With the above idea in mind it is clear that an obvious choice for a reference error rate would be

$$\dot{\sigma} = -\lambda \frac{dV}{d\sigma} \quad ; \quad \lambda > 0 \tag{4.28}$$

However, in case of penalty functions $V(\sigma)$ of the form $V(\sigma) = \alpha(\sigma)\tilde{V}(\sigma)$ (with $\tilde{V}(\sigma)$ quadratic, and $\alpha(\sigma)$ a window-sigmoid) an alternative simpler choice is however possible

$$\dot{\sigma} = \alpha(\sigma)\dot{\tilde{\sigma}} \quad ; \quad \dot{\tilde{\sigma}} \triangleq -\lambda \frac{d\tilde{V}}{d\sigma} \quad ; \quad \lambda > 0 \tag{4.29}$$

which still guarantees the fulfillment of the condition (4.27).

Centralized Rate-Tasks with Priorities

Managing the Highest Priority Task As explained in Section 4.2.3, generally the set-control objectives should be achieved before any other precision objective, in such a way to guarantee, other than the fulfillment of all safety pre-conditions, the precision objectives themselves to be ultimately in the best conditions for being effectively accomplished. And this reconfirms the reasons the group of set objectives should be considered with higher priority than the precision ones.

With this general idea in mind, now assume that within the group of the established set objectives there is one that in turn has to be achieved with the highest priority. Then in order to respect such directive, the following set-rate-task can be associated to such primary set objective

$$S(\dot{\tilde{\sigma}}) \triangleq \left\{ y = \arg\min_{y} \|\dot{\tilde{\sigma}} - \alpha(\sigma) n^T J y\|^2 \right\} \iff y = G^{\#}\dot{\tilde{\sigma}} + (I - G^{\#}G)z \;\; ; \;\; \forall z \tag{4.30}$$

where $G = \alpha(\sigma) n^T J$ and $Ker(G) = (I - G^{\#}G)$.

Observe how, when $\alpha(\sigma) = 1$ (i.e. when the set objective is sufficiently far to be achieved), $\dot{\sigma}$ can be made equal to the reference rate $\dot{\tilde{\sigma}}$, which guarantees the penalty function $V(\sigma)$ to be decreasing as fast as desired, at the current sampling instant. And in these conditions it is also immediate to verify how the dimension of $S(\dot{\tilde{\sigma}})$ (which is obviously the same of $Ker(G)$) results into that of the unconditioned y, just decreased by one (thus in TRIDENT case 6 (for the vehicle) + 7 (for the arm) -1 = 12).

When instead $\alpha(\sigma) = 0$, i.e. when $V(\sigma) = 0$ and the set task has been fulfilled, then also $G(\alpha) = 0$ and the embedded regularization makes $G^{\#}(\alpha) = 0$. This allows y to be arbitrary, since in this case $y = z$. Moreover, note how in this case the dimension of $Ker(G)$ is not decreased by one.

The possibility of enlarging as much as possible the arbitrariness of the system velocity vector y within the achievement of a set objective (for the time being the first considered highest priority one) actually constitutes the main motivation for the insertion of the window-sigmoid $\alpha(\sigma)$ multiplying $n^T J$, since otherwise such enlargement would not occur, because $\dot{\sigma}$ would remain forced to assign to the reference rate $\dot{\tilde{\sigma}}$, even if not anymore necessary once the set objective is achieved.

Thus, in force of the above reasoning, note how the dimension of linear manifold $S(\dot{\tilde{\sigma}})$ consequently increases by one once a *complete* transition of $\alpha(\sigma)$ from the unitary value to the zero one occurs (and obviously the contrary in case of an opposite complete transition).

When instead $\alpha(\sigma)$ is in the transition zone (i.e. when $0 < \alpha(\sigma) < 1$), note how for monotonically decreasing $\alpha(\sigma)$ leaving from the unitary value, certainly there will be a threshold value (let us call it α^*) immediately below which the embedded regularization comes into play.

In these conditions (i.e. for $0 < \alpha(\sigma) < \alpha^*$), the above expression $(I - G^{\#}G)z$ (with $G^{\#}$ now evaluated in regularized form) cannot actually anymore represent

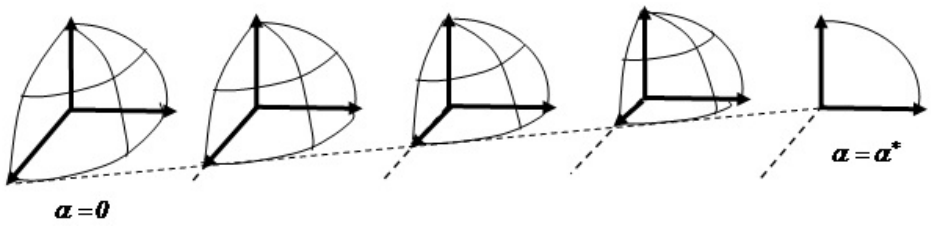


Figure 4.11: Continuity of the evolution of the regularized kernel space when moving from $\alpha(\sigma) = 0$ (left) to $\alpha(\sigma) = \alpha^*$ (right)

$Ker(G)$. Indeed, the regularization makes $G(I - G^{\#}G)z \neq 0$ for some arbitrary z. Moreover, the first term $G^{\#}\dot{\bar{\sigma}}$ does not anymore provide the minimum norm solution.

Theoretically, the minimum norm solution and $Ker(G)$ are still well defined even in these conditions, if one can accept that some elements of $G^{\#}$ tend to infinity as $\alpha \to 0$, and also that the minimum norm solution $G^{\#}\dot{\bar{\sigma}} \to \infty$.

The regularization embedded in the pseudo inverse of G serves just for the purpose of avoiding this phenomena. Indeed, for $\alpha \to 0$, the regularized minimum norm solution $G^{\#}\dot{\bar{\sigma}}$ smoothly tends to zero instead than to infinite, by passing through a finite maximum value whose entity can be controlled to remain below a suitably assigned threshold.

Moreover, in presence of regularization, a sort of "continuity" in the evolution of the "regularized kernel space" (or "approximated kernel space") $(I - G^{\#}G)z$ can be devised, when passing from $Ker(G)$ when $\alpha(\sigma) = 1$ to again $Ker(G)$ when $\alpha(\sigma) = 0$ (now increased by one dimension).

In order to figure out this idea of "continuity", without going into mathematical details involving the well known Singular Value Decomposition-SVD algorithmic procedure for matrices (see for instance [93]), refer to what is depicted in Fig. 4.11, where the axes in the leftmost drawing are intended to exactly represent $Ker(G)$ when $\alpha(\sigma) = 0$, while those in the rightmost one are intended to dually represent $Ker(G)$ when $\alpha(\sigma) = \alpha^*$; thus with a reduced by one dimensionality.

In between of such extreme cases, the reported set of axes are used for indicating the sequence of approximated kernel spaces produced by $(I - G^{\#}G)z$ for the intermediate values $0 < \alpha(\sigma) < \alpha^*$. Then what is depicted shows that, though each one of the approximating kernel spaces remains with the same dimensionality of when $\alpha(\sigma) = 0$, it happens that along the directions which must

eventually disappear, the regularization imposes that any projection along them is automatically attenuated by a contracting coefficient. This coefficient continuously decrease from 1 to 0, as $\alpha(\sigma)$ is continuously moved from 0 to α^*. This means that the liberty of choice within the intermediate spaces is also gradually attenuated along such directions.

Moreover, within the sub interval $0 < \alpha(\sigma) < \alpha^*$, the cost function in (4.30) is only approximately minimized to zero, even if with continuity with respect to the extremes. This is the price to be paid to allow a smooth transition between the two extreme optimal solutions.

In light of the above considerations, throughout the following part of this Section, all the approximations introduced by the regularizations in the sub interval $0 < \alpha(\sigma) < \alpha^*$ will be considered as practically negligible.

A few remarks about the structure of the minimization of (4.30) are still due as follows.

First of all observe that the eventual coming into play of the embedded regularization procedure has been implicitly interpreted as only due to $\alpha(\sigma)$, whenever approaching the zero value, while instead it might also come into play in case the row factor $n^T J$ were independently approaching zero. However, since this would imply the unit vector n actually approaching the condition $n \in Ker(J^T)$, it has been here reasonably deemed that with 13 degree of mobility actually available in the TRIDENT case, matrix J should most probably be always full row rank 3. Nevertheless, in case of a seldom occurrence of matrix J for some reason loosing its full row rankness, it is clear that the embedded regularization will continue to make its duty. Further it was also implicitly assumed that the unit vector n is always well defined, which is actually not anymore the case whenever $\alpha \to 0$. However, since for set objectives such eventuality is always preceded, and then masked, by the occurrence $\alpha(\sigma) \to 0$, the problem does not actually emerge.

In light of all these consideration, it is now clear that the allowed arbitrariness in the choice of y can actually be used for also attempting to drive the system toward the achievement of (at least some of) the remaining lower priority objectives.

And to this respect the dimension of $Ker(G)$ naturally results to be an important issue, because greater it is, greater also becomes the possibilities left to the system of driving itself also toward the achievement of the others, not yet

considered, lower priority objectives. Thus once more explaining the fundamental reasons underlying the adoption of cost functional (4.30), which can be now summarized as follows:

- the cost functional argument only account for the error-rate $\dot{\sigma}$ (and not for the associated error-vector rate χ) thus resulting into a scalar argument, which is consequently prone to allow the maximal dimensionality to the associated linear manifold of solutions $S(\dot{\bar{\sigma}})$; and
- the use of the associated window-sigmoid function $\alpha(\sigma)$ as modulating factor of the error-rate $\dot{\sigma}$ further allows to increase (by one) the dimensionality of $S(\dot{\bar{\sigma}})$, when the highest priority set objective is ultimately achieved.

Managing Lower Priority Tasks The arbitrariness within $S(\dot{\bar{\sigma}})$ can now be used for in turn attempting to also achieve the next control objective, exhibiting the highest lower priority and characterized by penalty function $V_1(\sigma_1)$. Then, assuming this last to be also a set objective, the associated set-rate task is

$$S_1(\dot{\bar{\sigma}}_1, \dot{\bar{\sigma}}) \triangleq \left\{ y = \arg \min_{y \in S(\dot{\bar{\sigma}})} \|\dot{\bar{\sigma}}_1 - \alpha_1(\sigma_1) n_1^T J_1 y\|^2 \right\} \tag{4.31}$$

Since the above now constitutes a constrained minimization problem (since $y \in S(\dot{\bar{\sigma}})$), finding its solution preliminary requires to represent y inside $S(\dot{\bar{\sigma}})$ via the previously obtained solution of problem (4.30). By doing so, the following equivalent form of (4.31) is obtained

$$S_1(\dot{\bar{\sigma}}_1, \dot{\bar{\sigma}}) \triangleq \left\{ y = \bar{\rho} + Qz \;\text{ s.t. }\; z = \arg \min_{z} \|\gamma_1 - \hat{G}_1 z\|^2 \right\} \tag{4.32}$$

where for notation compactness the following symbols have been introduced:

$$G_1 \triangleq \alpha_1(\sigma_1) n_1^T J_1 \tag{4.33}$$

$$y = G^{\#}\dot{\bar{\sigma}} + (I - G^{\#}G)z \triangleq \bar{\rho} + Qz \tag{4.34}$$

$$\gamma_1 \triangleq \dot{\bar{\sigma}}_1 - G_1 \bar{\rho} \tag{4.35}$$

$$\hat{G}_1 \triangleq G_1(I - G^{\#}G) \tag{4.36}$$

leading to the following linear manifold of solutions for z

$$Z_1(\dot{\bar{\sigma}}_1, \dot{\bar{\sigma}}) \iff z = \hat{G}_1^{\#}\gamma_1 + (I - \hat{G}_1^{\#}\hat{G}_1)z_1 \;\;;\;\; \forall z_1 \tag{4.37}$$

and for y

$$S_1(\dot{\bar{\sigma}}_1, \dot{\bar{\sigma}}) \iff y = (\bar{\rho} + Q\hat{G}_1^{\#}\gamma_1) + Q(I - \hat{G}_1^{\#}\hat{G}_1)z_1 \triangleq \bar{\rho}_1 + Q_1 z_1 \;\; ; \;\; \forall z_1 \quad (4.38)$$

It is at this point clear how the above process can be further iterated by exploring, in a descending priority order, all successive set objectives, while providing, for each one of them, the corresponding solution set. This can be iterated until the highest priority precision objective is encountered, which will be treated in a successive section.

Before moving to the treatment of the precision objectives, a remark about the dimensionality of the solution space of a generic i-th set-rate task (coinciding with the dimensionality of $Ker(\hat{G}_i)$) within the list should be made. By letting d_i be the dimensionality of $Ker(\hat{G}_i)$ and keeping into account that the arguments in the quadratic costs to be orderly minimized are all of scalar type, it is not difficult to verify that

$$d_i = \lceil \alpha(\sigma_i) \rceil (d_{i-1} - 1) + \lceil 1 - \alpha(\sigma_i) \rceil (d_{i-1}) \quad ; \quad i = 0, 1, \ldots \quad ; \quad d_{-1} = m \quad (4.39)$$

where $m = \dim(y)$ and where the highest priority set task has been indexed with $i = 0$. Thus, as expected and accordingly with (4.39), the dimensionality of the arbitrariness space of a generic i-th set-rate task can shift between that of the previous set-rate task reduced by one, when the associated i-th set objective is not yet achieved, and the same of previous set-rate task in the opposite case.

As it can be deduced from the backward iteration of (4.32 - 4.38) along the list of descending priority set-rate tasks, such process can be algorithmically efficiently organized as follows. By initializing the variables

$$\bar{\rho}_{-1} = 0 \quad ; \quad \hat{G}_{-1} = 0 \quad ; \quad Q_{-1} = I \quad (4.40)$$

then for $i = 0, 1, \ldots$

$$\begin{aligned}
\gamma_i &\triangleq \dot{\bar{\sigma}}_i - G_i \bar{\rho}_{i-1} && (4.41)\\
\hat{G}_i &\triangleq G_i Q_{i-1} && (4.42)\\
\bar{\rho}_i &= \bar{\rho}_{i-1} + Q_{i-1}\hat{G}_i^{\#}\gamma_i && (4.43)\\
&= \left(I - Q_{i-1}\hat{G}_i^{\#}G_i\right)\bar{\rho}_{i-1} + Q_{i-1}\hat{G}_i^{\#}\dot{\bar{\sigma}}_i && (4.44)\\
Q_i &= Q_{i-1}(I - \hat{G}_i^{\#}\hat{G}_i) && (4.45)
\end{aligned}$$

which leads to the final formula for computing the velocity vector y

$$y = \bar{\rho}_i + Q_i z_i \quad ; \quad \forall z_i \tag{4.46}$$

At this point, the management of the remaining lower-priority precision objectives can now be considered.

To this aim, let k be the index of the first backward encountered precision objective and consider that for such objective $\alpha_k(\sigma_k) = 1$. It follows that $V_k(\sigma_k) = \tilde{V}_k(\sigma_k) = \frac{1}{2}\alpha_k^2$ and $\dot{\sigma}_k = \dot{\tilde{\sigma}}_k \triangleq -\lambda\sigma_k$, and that the associated precision-rate task should consequently be written as

$$S_k(\dot{\tilde{\sigma}}_k, \ldots, \dot{\tilde{\sigma}}_0) \triangleq \left\{ y = \arg \min_{y \in S_{k-1}} \|\dot{\tilde{\sigma}}_k - n_k^T J_k y\|^2 \right\} \tag{4.47}$$

Problem (4.47) remains structurally similar to those for the set-rate task, even if it is unfortunately suffering of the drawback that in the vicinity of the precision task achievement (i.e. in the vicinity of $\sigma_k = 0$) the unit vector n_k of the associate error vector κ_k actually tends to become ill-defined. This implies that, if no measures are taken, some uncontrolled unwanted phenomena (i.e. most probably chattering around $\sigma_k = 0$) may unfortunately take place.

In order to face this aspect, let us assume $\sigma_k > 0$, allowing the unit vector n_k to be defined, at least from the formal point of view. Then, under this assumption, let us first rewrite the norm term in (4.47), equivalently as

$$\left\|\dot{\tilde{\sigma}}_k - n_k^T J_k y\right\|^2 = \left\|n_k^T n_k \dot{\tilde{\sigma}}_k - n_k^T J_k y\right\|^2 = \left\|n_k^T \left(\bar{\chi}_k - J_k y\right)\right\|^2 \tag{4.48}$$

recalling that $n_k^T n_k = 1$ since it is a unit vector, and $\bar{\chi}_k \triangleq n_k \dot{\tilde{\sigma}}_k$.

By looking at the r.h.s. of the above equation, to the aim of minimizing the given quadratic form, at least in the vicinities of $\sigma_k = 0$, the following problem, in lieu of the original one, could be resolved

$$S_k(\dot{\tilde{\sigma}}_k, \ldots, \dot{\tilde{\sigma}}_0) \triangleq \left\{ y = \arg \min_{y \in S_{k-1}} \|\bar{\chi}_k - J_k y\|^2 \right\} \tag{4.49}$$

where the square of the projection on n_k of vector $\bar{\chi}_k - J_k y$ has been substituted with the squared norm of the vector itself. This allows to remove the explicit appearance of the unit vector n_k.

Obviously, the two problems are generally not equivalent, if not in the case when the second one actually allows for exact solutions. This condition should be satisfied at least when:

1. all set objectives have been eventually achieved, that is once the system has eventually entered the assumed existing non-conflicting zone and the system velocity vector y has therefore recovered its full mobility dimension m;

2. m equates or is larger than the dimension of $\bar{\chi}_k$ (as it is in this case case where $m = 13$ against 3 as the dimension of $\bar{\chi}_k$); and

3. the Jacobian J_k is guaranteed remaining full-row rank within the entire non-conflicting zone (as it will be shortly proven for our case).

This consequently explains the rationale underlying the suggested use of (4.49).

As regard the fulfillment of condition 3) within the non-conflicting zone, remind that in this case the whole group of set objectives also includes the one regarding the arm manipulability recovering. This implies the non singularity of the arm posture once all set objectives will be achieved. Thus, by joining this fact with the natural non-singularity exhibited by the vehicle in the same conditions (i.e. a 6 d.o.f. fully actuated rigid body), it then follows that the whole system will consequently be fully non-singular within the whole non-conflicting zone.

The translation of the above suggested procedure for managing the precision rate-task is analogous to that for the management of set rate-tasks, upon the straightforward substitutions $\dot{\bar{\sigma}}_i \leftarrow \bar{\chi}_i$ and $G_i \leftarrow J_i$, leading to for $i = k, k+1, \ldots$

$$\gamma_i \triangleq \bar{\chi}_i - J_i \bar{\rho}_{i-1} \tag{4.50}$$

$$\hat{G}_i \triangleq J_i Q_{i-1} \tag{4.51}$$

$$\bar{\rho}_i = \bar{\rho}_{i-1} + Q_{i-1} G_i^{\#} \gamma_i \tag{4.52}$$

$$= \left(I - Q_{i-1} G_i^{\#} J_i\right) \bar{\rho}_{i-1} + Q_{i-1} G_i^{\#} \bar{\chi}_i \tag{4.53}$$

$$Q_i = Q_{i-1}(I - \hat{G}_i^{\#} \hat{G}_i) \tag{4.54}$$

Distributed Rate-Tasks with Priorities

The distributed rate-task with priorities kinematic control methodology arises from the employment of a Dynamic Programming (DP) techniques, once the overall system has been suitably subdivided into composing sub-chains and, to the aim of finding the searched kinematic control law, the DP procedure is then made suitably running (backward and forward) along such composing sub-chains.

The employment of such technique will actually deserve many advantages with respect to the centralized approach, also including the capability of facing multi-rate sampling situations in a very easy manner.

In order to better introduce the approach, let us assume a common sampling rate situation and consequently delay the extension to the multi-rate case to a slightly later discussion.

As anticipated, the DP based approach develops by first of all subdividing the overall kinematic chain (in the TRIDENT case the vehicle plus the arm) into composing sub-chains. Then a "direction of exploration" of the resulting sequence of sub-chains is chosen, which in the TRIDENT case is the arm-vehicle direction.

By following the chosen exploration path, a sequence of prioritized rate tasks is executed, obeying to the following basic rules:

- the velocities of all the chains underlying the starting one are considered as given parameters at the current sampling instant (thus, in our case, these parameters are just the vehicle linear and angular velocity vector); and
- as a consequence of the previous point, only the cost functions directly involving the starting sub-chain velocities (i.e. in our case the arm joint velocity vector $\dot{q}$) must be accounted.

Thus, by applying the above rules, the following table of backward sequential minimizations needs to be performed at arm level:

SET-RATE JOINT LIMITS TASK

$$S_{a0}(\dot{\tilde{q}}) \triangleq \left\{ \dot{q} = \arg\min_{\dot{q}} \left(\sum_{i=1}^{7} \left\| \dot{\tilde{q}}_i - \alpha_{q_i} \dot{q}_i \right\|^2 \right) \right\}$$

SET-RATE MANIPULABILITY TASK

$$S_{a1}(\dot{\bar{\sigma}}_\mu, \dot{\tilde{q}}) \triangleq \left\{ \dot{q} = \arg\min_{\dot{q} \in S_{a0}} \left\| \dot{\bar{\sigma}}_\mu - \alpha_\mu J_\mu \dot{q} \right\|^2 \right\}$$

PRECISION-RATE END EFFECTOR APPROACH r TASK

$$S_{a2}(\chi_r, \dot{\bar{\sigma}}_\mu, \dot{\tilde{q}}, v) \triangleq \left\{ \dot{q} = \arg\min_{\dot{q} \in S_{a1}} \left\| \chi_r - J_r y \right\|^2 / v \right\} \tag{4.55}$$

PRECISION-RATE END EFFECTOR APPROACH ϑ TASK

$$S_{a3}(\chi_\vartheta, \chi_r, \dot{\bar{\sigma}}_\mu, \dot{\tilde{q}}, v) \triangleq \left\{ \dot{q} = \arg\min_{\dot{q} \in S_{a2}} \left\| \chi_\vartheta - J_\vartheta y \right\|^2 / v \right\}$$

ARM CONTROL LAW

$$\dot{q}(\chi_\vartheta, \chi_r, \dot{\bar{\sigma}}_\mu, \dot{\tilde{q}}, v) \triangleq \dot{q}(\bar{a}, v) = \arg\min_{\dot{q} \in S_{a3}} \|\dot{q}\|^2$$

In the above note how the joint limit and manipulability set rate tasks, since involving $\dot{q}$ only, do not need of any conditioning with respect to v. The end-effector precision rate tasks instead require the explicit conditioning with respect to v, since their arguments explicitly depends from it, via the whole system velocity vector y. Moreover all the vehicle rate tasks (camera centering, camera height, camera distance, horizontal attitude) are not appearing since they also are not influenced by $\dot{q}$.

Finally note how the choice for $\dot{q}$ within the residual arbitrariness space ultimately left at arm level results into a control law (i.e not yet a control action), that is in a function of the yet-to-be-specified vector v. Such control law is however linear in all its arguments, since coming from the minimization of quadratic functions constrained on linear manifolds.

By then proceeding backward along the overall chain, let us now consider the vehicle. At such level, a sequence of prioritized rate task, to be now minimized with respect to the vehicle velocity vector v, is executed, obeying to:

1. the backward sequence of minimizations has to be performed within the conditioning provided by the previously devised (parameterized by v) arm kinematic control law; and

2. as a consequence, only the rate-tasks directly depending from v, or indirectly depending from v via the arm control law $\dot{q}(\bar{a}, v)$, have to be accounted.

Thus, by applying such rules the following table of sequential minimizations is obtained at Vehicle Level:

SET-RATE HORIZONTAL ATTITUDE TASK

$$S_{v0}(\dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_v \left\| \dot{\bar{\sigma}}_\varphi - \alpha_\varphi n_\varphi^T J_\varphi v \right\|^2 \right\}$$

SET-RATE JOINT LIMITS TASK

$$S_{v1}(\dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v0}} \left(\sum_{i=1}^{7} \left\| \dot{\tilde{q}}_i - \alpha_{q_i} \dot{q}_i \right\|^2 \right) / \dot{q}(\bar{a}, v) \right\}$$

SET-RATE CAMERA CENTERING TASK

$$S_{v2}(\dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v1}} \left\| \dot{\bar{\sigma}}_\xi - \alpha_\xi n_\xi^T J_\xi v \right\|^2 \right\}$$

SET-RATE CAMERA DISTANCE TASK

$$S_{v3}(\dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v2}} \left\| \dot{\bar{\sigma}}_d - \alpha_d n_d^T J_d v \right\|^2 \right\}$$

SET-RATE CAMERA HEIGHT TASK

$$S_{v4}(\dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v3}} \left\| \dot{\bar{\sigma}}_h - \alpha_h n_h^T J_h v \right\|^2 \right\}$$

SET-RATE MANIPULABILITY TASK

$$S_{v5}(\bar{a}, \dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v4}} \left\| \dot{\bar{\sigma}}_\mu - \alpha_\mu J_\mu \dot{q} \right\|^2 / \dot{q}(\bar{a}, v) \right\}$$

PRECISION-RATE END EFFECTOR APPROACH r TASK

$$S_{v6}(\bar{a}, \dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v5}} \left\| \chi_r - J_r y \right\|^2 / \dot{q}(\bar{a}, v) \right\}$$

PRECISION-RATE END EFFECTOR APPROACH ϑ TASK

$$S_{v7}(\bar{a}, \dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v6}} \left\| \chi_\vartheta - J_\vartheta y \right\|^2 / \dot{q}(\bar{a}, v) \right\}$$

PRECISION-RATE HORIZONTAL ATTITUDE TASK

$$S_{v8}(\dot{\bar{\varphi}}, \bar{a}, \dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq \left\{ v = \arg\min_{v \in S_{v7}} \left\| \dot{\bar{\varphi}} - J_\varphi v_2 \right\|^2 \right\}$$

VEHICLE CONTROL LAW

$$v(\dot{\bar{\varphi}}, \bar{a}, \dot{\bar{\sigma}}_d, \dot{\bar{\sigma}}_h, \dot{\bar{\sigma}}_\xi, \dot{\tilde{q}}, \dot{\bar{\sigma}}_\varphi) \triangleq v(\bar{b}, \bar{a}) \triangleq \left\{ v = \arg\min_{v \in S_{v8}} \left\| v \right\|^2 \right\} \tag{4.56}$$

At the end of the minimization sequence, provided the last remained linear manifold S_{v8} is not of null dimension, the procedure selects, inside it, the element $v(\bar{b}, \bar{a})$ exhibiting the minimal norm. As a consequence of this, note how $v(\bar{b}, \bar{a})$ results into a numerically valued control action (i.e. not anymore a control law).

A few notes about the aforementioned sequence of minimizations must be certainly made.

First, note how the Horizontal Attitude task is present twice, as a set rate task at the top of the chain of minimizations for the vehicle, and then at its bottom, as a precision task. This actually reflects the fact that the attitude error must be kept below a certain threshold with a high priority. However, by using a scalar cost, only one d.o.f. is used to ensure that this error is kept small. Then, at the bottom of the chain, if any d.o.f. are still available, the precision task tries to nullify this error. With only a precision task, this would have been impossible to achieve, unless two d.o.f. were always dedicated to the task.

Moreover, the precision Horizontal Attitude task appears after the End Effector Approach tasks, which require 6 d.o.f. to be executed. One may wonder if the precision Horizontal Attitude task ever gets executed, since the End Effector Approach precision ones are of the same dimension of all the available vehicle's d.o.f.. However, since the End Effector Approach tasks are also present at the arm level, their actual dimension at the vehicle level is generally less than 6. Thus allowing the possibility to have some available d.o.f. for the final precision Horizontal Attitude task.

Finally note how certain tasks, like the Joint Limits task and the Manipulability task are also present at the vehicle level. This is due to the choice of running the DP approach along the arm-vehicle chain in this order. This makes the arm control law parametrized by v, thus the vehicle, indirectly, can affect such tasks with its motion. For example, if the arm is too stretched and has a low manipulability, the vehicle can move closer to the target, allowing the arm to move away from its singular posture and gain dexterity.

The above described backward sequential minimization of the two sequences of costs, constitutes the so called Backward Phase of the proposed Dynamic Programming based procedure, to be followed by the associated so-called Forward Phase simply corresponding to computing the actual value $\bar{v} = v(\bar{b}, \bar{a})$ to be substituted into the arm level control law $\dot{q}(\bar{a}, v)$, thus leading to the final value for $\dot{\bar{q}}$. Then these two values, forming the overall system reference velocity vector $\bar{y}$, are sent to the underlying Dynamic Layer as current reference values to be tracked.

Algorithmic Structure The algorithmic structure underlying the implementation of the DP procedure, will be now presented in general terms starting from

the arm.

Consider the minimization regarding a generic rate-task located at arm level

$$S_i(\dot{\bar{\sigma}}_i, \ldots, \dot{\bar{\sigma}}, v) \triangleq \left\{ y = \arg \min_{y \in S_{i-1}} \left\| \left[\dot{\bar{\sigma}}_i - \alpha_i(\sigma_i) n_i^T J_i^v v \right] - \alpha_i(\sigma_i) n_i^T J_i^a \dot{q} \right\|^2 \right\} \quad (4.57)$$

where the overall contribution Jy has been split into $J_i^v v$ and $J_i^a \dot{q}$, i.e. the contribution of the vehicle and the arm. Also, for notation compactness, let $G_i^v(\sigma_i) \triangleq \alpha_i(\sigma_i) n_i^T J_i^v$ and $G_i^a(\sigma_i) \triangleq \alpha_i(\sigma_i) n_i^T J_i^a$.

Then the algorithm results for $i = 0, 1, \ldots$ equivalent to (4.41), with the substitutions $\dot{\bar{\sigma}}_i \leftarrow (\dot{\bar{\sigma}}_i - G_i^v v)$ and $G_i \leftarrow G_i^a$:

$$\hat{G}_i^a \triangleq G_i^a Q_{i-1} \quad (4.58)$$

$$\bar{\rho}_i = \left(I - Q_{i-1} (\hat{G}_i^a)^{\#} \hat{G}_i^a \right) \bar{\rho}_{i-1} + Q_{i-1} (\hat{G}_i^a)^{\#} (\dot{\bar{\sigma}}_i - G_i^v v) \quad (4.59)$$

$$Q_i = Q_{i-1} \left(I - (\hat{G}_i^a)^{\#} \hat{G}_i^a \right) \quad (4.60)$$

The above formulation can be rewritten, in order to have an explicit appearance of the parameter v in the following way

$$\hat{G}_i^a \triangleq G_i^a Q_{i-1} \quad (4.61)$$

$$P_i = \left(I - Q_{i-1} (\hat{G}_i^a)^{\#} \hat{G}_i^a \right) P_{i-1} + Q_{i-1} (\hat{G}_i^a)^{\#} G_i^v \quad (4.62)$$

$$\bar{\rho}_i = \left(I - Q_{i-1} (\hat{G}_i^a)^{\#} \hat{G}_i^a \right) \bar{\rho}_{i-1} + Q_{i-1} (\hat{G}_i^a)^{\#} \dot{\bar{\sigma}}_i \quad (4.63)$$

$$Q_i = Q_{i-1} \left(I - (\hat{G}_i^a)^{\#} \hat{G}_i^a \right) \quad (4.64)$$

leading to the following arm control law

$$\dot{q} = \bar{\rho}_i + P_i v + Q_i z_i \quad ; \quad \forall z_i \quad (4.65)$$

The treatment of the precision tasks is exactly the same, having made the substitutions $\dot{\bar{\sigma}}_i \leftarrow \bar{\chi}_i$ and $G_i^a \leftarrow J_{1,i}^a$, $G_i^v \leftarrow J_i^v$.

Thus, the final control law of the arm, containing all the set and precision tasks, is

$$\dot{\bar{q}} = \dot{q}(\bar{a}, v) = \bar{\rho}^a + P v \quad (4.66)$$

where $\bar{\rho}^a$ and P are the final values attained by the vectors at the end of the above algorithm, and where $\bar{a}$ embeds all the reference rates $\dot{\bar{\sigma}}_i, \bar{\chi}_i$ inputted at arm level.

Also note how the arm control law results without any further arbitrariness (i.e. no appearance of z_i) in the components of $\dot{q}$, since in case a residual arbitrariness space were instead left, the final minimum norm choice inside it makes it disappearing.

Let us now move to the vehicle level, and let us consider the minimization regarding a generic rate-task

$$S_i(\dot{\bar{\sigma}}_i, \ldots, \dot{\bar{\sigma}}, v) \triangleq \left\{ y = \arg\min_{v \in S_{i-1}} \left\| \left[\dot{\bar{\sigma}}_i - \alpha_i(\sigma_i) n_i^T J_i^v v \right] - \alpha_i(\sigma_i) n_i^T J_i^a \dot{q} \right\|^2 / \dot{q}(\bar{a}, v) \right\} \tag{4.67}$$

By substituting the arm control law (4.66) in lieu of $\dot{q}$, the following equivalent representation is obtained

$$S_i(\dot{\bar{\sigma}}_i, \ldots, \dot{\bar{\sigma}}, v) \triangleq \left\{ y = \arg\min_{v \in S_{i-1}} \left\| (\dot{\bar{\sigma}}_i - G_i^a \bar{\rho}^a) - (G_i^v + G_i^a P)\, v \right\|^2 / \dot{q}(\bar{a}, v) \right\} \tag{4.68}$$

Let us call $F_i(\sigma_i) \triangleq (G_i^v + G_i^a P)$. Then (4.68) is structurally the same as (4.57), and thus the following set of equations can be found, for $i = 0, 1, \ldots$

$$\hat{F}_i \triangleq F_i Q_{i-1} \tag{4.69}$$

$$\bar{\rho}_i = \left(I - Q_{i-1} \hat{F}_i^{\#} \hat{F}_i \right) \bar{\rho}_{i-1} + Q_{i-1} \hat{F}_i^{\#} \left(\dot{\bar{\sigma}}_i - G_i^a \bar{\rho}^a \right) \tag{4.70}$$

$$Q_i = Q_{i-1} \left(I - \hat{F}_i^{\#} \hat{F}_i \right) \tag{4.71}$$

and the same structure holds for the precision tasks, with $F_i(\sigma_i) \triangleq (J_i^v + J_{1i}^a P)$ and substituting $(\dot{\bar{\sigma}}_i - G_i^a \bar{\rho}^a) \leftarrow (\bar{\chi}_o - J_i^a \bar{\rho}^a)$, leading to a control law of the type

$$v = \bar{\rho}_i + Q_i z_i \; ; \; \forall z_i \tag{4.72}$$

which can also be written as

$$\bar{v} = v(\bar{b}, \bar{a}) = \bar{\rho}^v \tag{4.73}$$

where $\bar{b}$ now embeds all the reference rates at vehicle level. This is actually a control action, than can finally be substituted into (4.66) in order to compute the control action for the arm.

Compensation of Vehicle Velocity Disturbances Following the results of the above section, the implementation of the successive forward phase of the Dynamic Programming actually reduces to the following simple substitution

$$\bar{y} = y(\bar{a}, \bar{b}) = \begin{cases} \dot{\bar{q}} & = \dot{q}\left[\bar{a}, v(\bar{b}, \bar{a})\right] & = \bar{\rho}^a + P\bar{\rho}^v \\ \bar{v} & = v(\bar{b}, \bar{a}) & = \bar{\rho}^v \end{cases} \tag{4.74}$$

Note how the choice of explicitly performing the above substitution is actually not the most convenient one, since it prevents to exploit another important peculiar property exhibited by the Dynamic Programming approach, namely the more efficient management of non-negligible vehicle velocity disturbances and/or inaccuracies that could be exhibited at vehicle level, that otherwise would completely disappear.

More precisely, note how the arm-level control law actually is the best one to be employed in correspondence of *any* velocity v exhibited by the underlying vehicle (and the successive definition of a suitable $\bar{v}$ for the vehicle just serves for further improving what the arm is already per-se performing at best).

Then, by explicitly taking into account this property, it can be immediately seen that in case the actual vehicle velocity v were made accessible to the Arm Kinematic Controller, this last should preferably use the actual v as one of its input, in lieu of the computed reference one $\bar{v}$, thus automatically compensating for possible vehicle velocity tracking inaccuracies and/or velocity disturbances currently occurring at vehicle level.

Multi-Rate Sampling

In case of multi-rate sampling, as it actually is for TRIDENT, where the arm is controlled at 100 Hz and the vehicle at 20 Hz, it is sufficient to consider the larger period of 0.05s subdivided into 5 sub-periods of 0.01s, and then notice how, during the first sub-period, the Dynamic Programming procedure will have to run entirely (i.e. backward running from the arm till the vehicle, and then forward). Thus, the process will produce, other than the joint velocity reference $\dot{\bar{q}}$ for such first sub-period, also the vehicle reference velocity $\bar{v}$, that instead will be maintained valid also for all other four. During each one of the remaining sub-periods, the kinematic controller will therefore limit itself in running the Dynamic Programming part relevant to the arm level only, while using $\bar{v}$ (or the actual vehicle velocity v if available) as a constant parameter. All this as it appears schematically indicated in Fig. 4.12.

Let us conclude this remark by further noting how the capability of handling the case of multi-rate sampling as above indicated, it is also a peculiar property exhibited by the Dynamic Programming approach, not easily identifiable within the standard one.

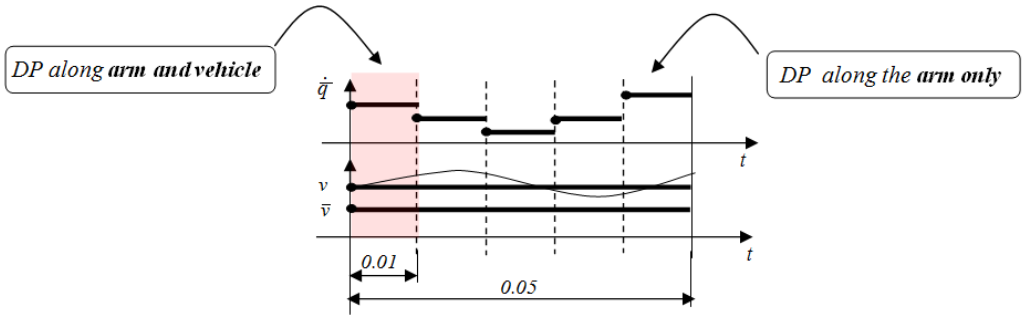


Figure 4.12: The DP algorithm is run in every sub periods for the arm, while only in the first one for the vehicle

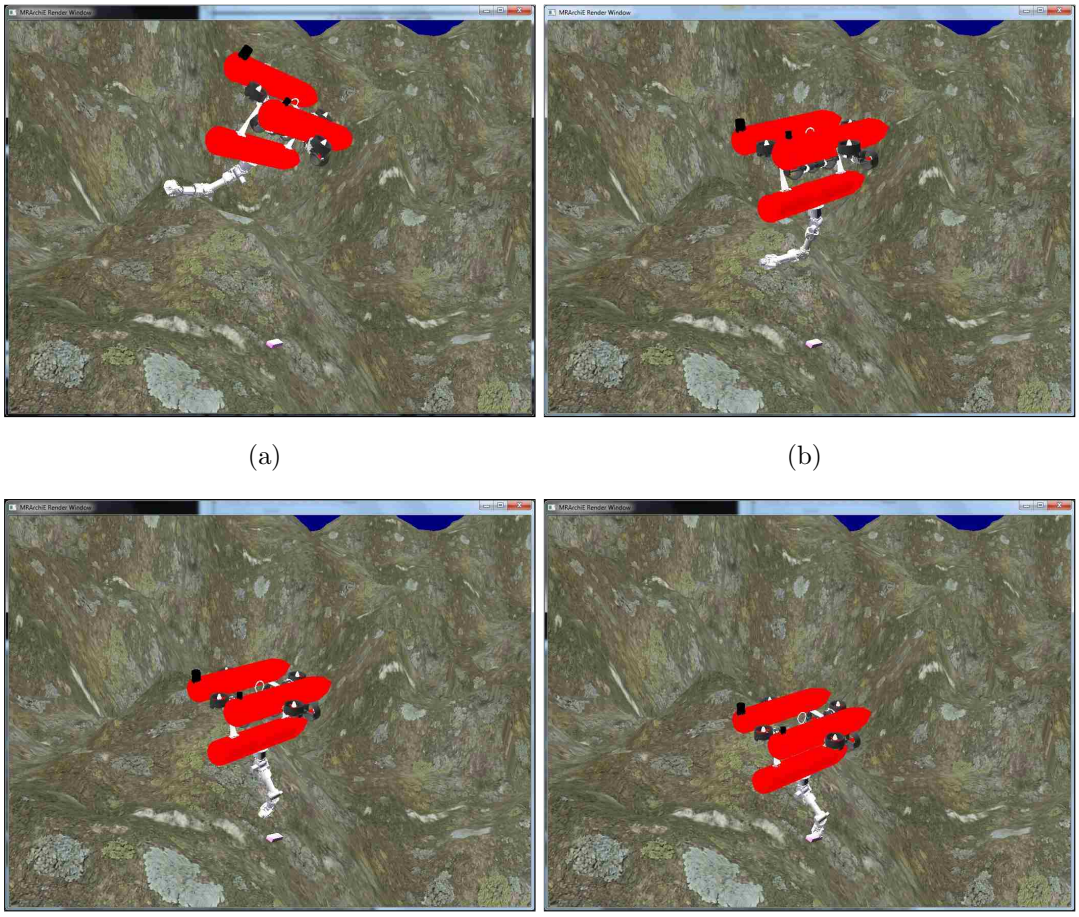
4.2.6 Simulative Results

In this section, three tests are presented in order to show the effectiveness of the proposed coordination approach.

Test 1

In this first test, the object is placed in $[0\ 0\ -3]$ with a rotation of $[1\ 0.5\ 0.2]$ radians. The vehicle starts at $[0\ 0\ 0]$ with an initial rotation of $[\pi/6\ \pi/8\ \pi/4]$. Figure 4.13 shows the movement of the I-AUV, with frames captured each 10s.

Figure 4.14 shows the behavior of the different α of the set rate tasks. In this example, the camera centering and the horizontal attitude are the two tasks that are active with their respective $\alpha = 1$. Thus, the vehicle reduces its pitch and roll angles as first objective, as can be seen in Fig. 4.15. Then, it moves horizontally in order to center the camera in its vision cone. As the manipulator stretches in order to reach the object and its manipulability decreases, the manipulability tasks becomes active and the vehicle naturally decreases its depth in order to let the arm recover its dexterity. Figure 4.16 show the generated $\dot{q}$, while Fig. 4.17 show the requested $\bar{v}$ (which coincides with v in these simulations). The chattering that can be seen in $\dot{q}$ is due to the different sample rates used for the arm and the vehicle. This is especially noticeable whenever the vehicle has great accelerations, since the arm needs to take into account the movement of the vehicle.



(a) (b)

(c) (d)

Figure 4.13: Test 1: snapshots of the movement of the I-AUV

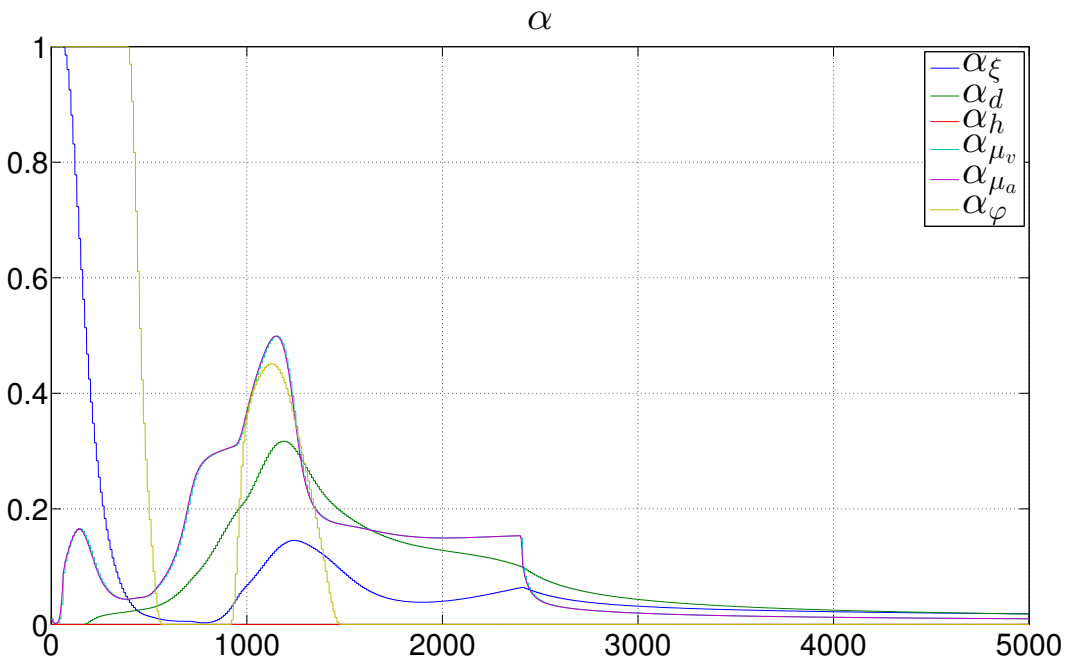


Figure 4.14: Test 1: time behavior of the α functions of the set rate tasks

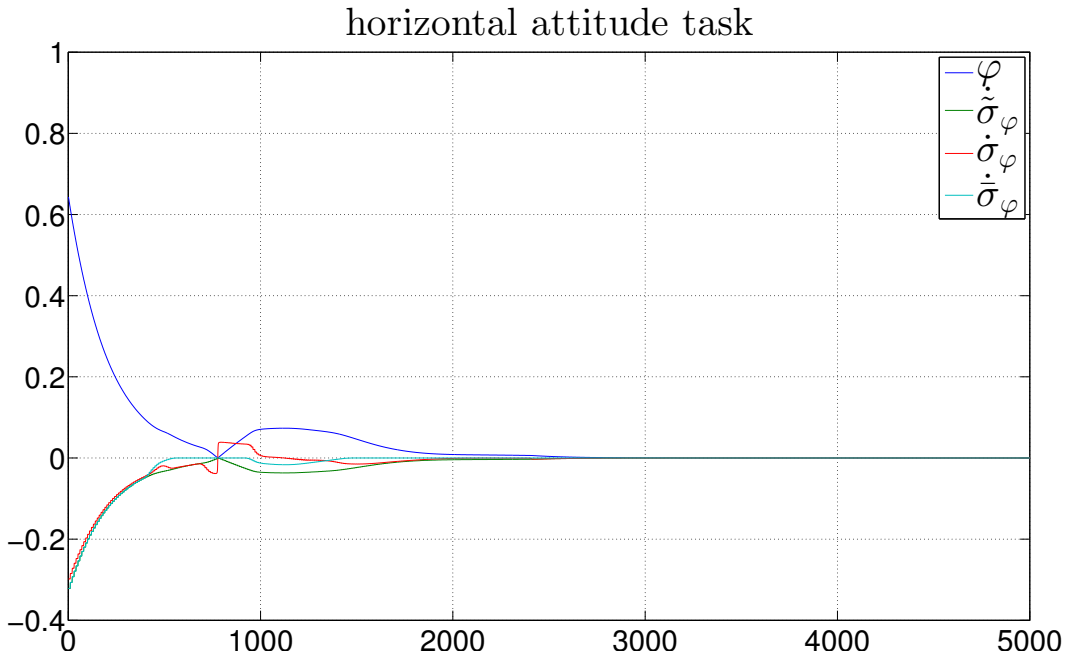


Figure 4.15: Test 1: time behavior of the quantities related to the horizontal attitude task

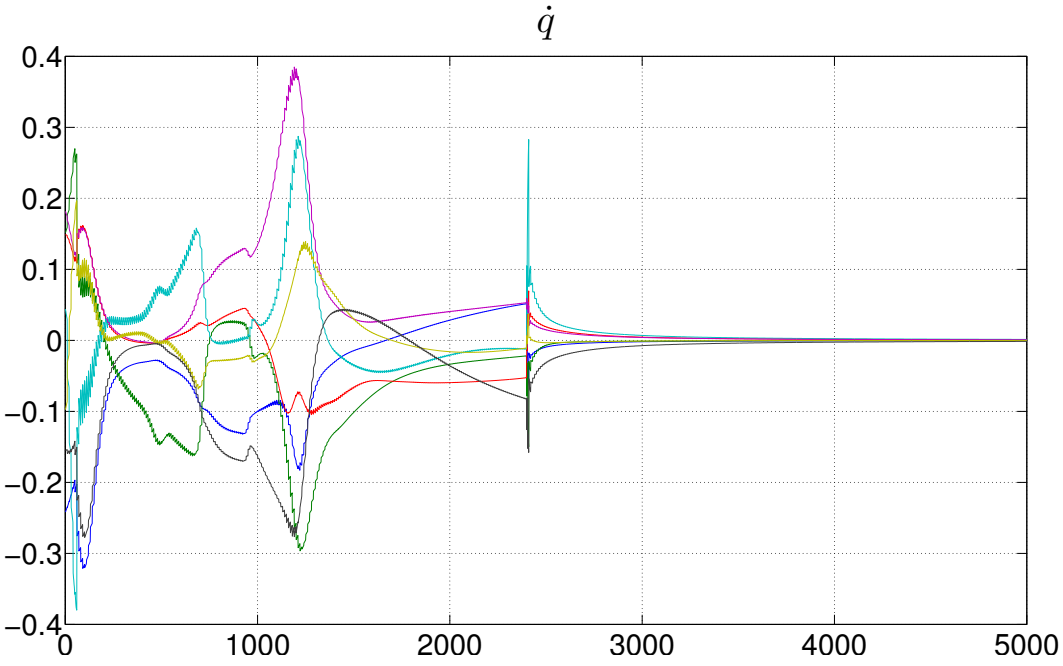


Figure 4.16: Test 1: the generated $\dot{q}$ for the arm

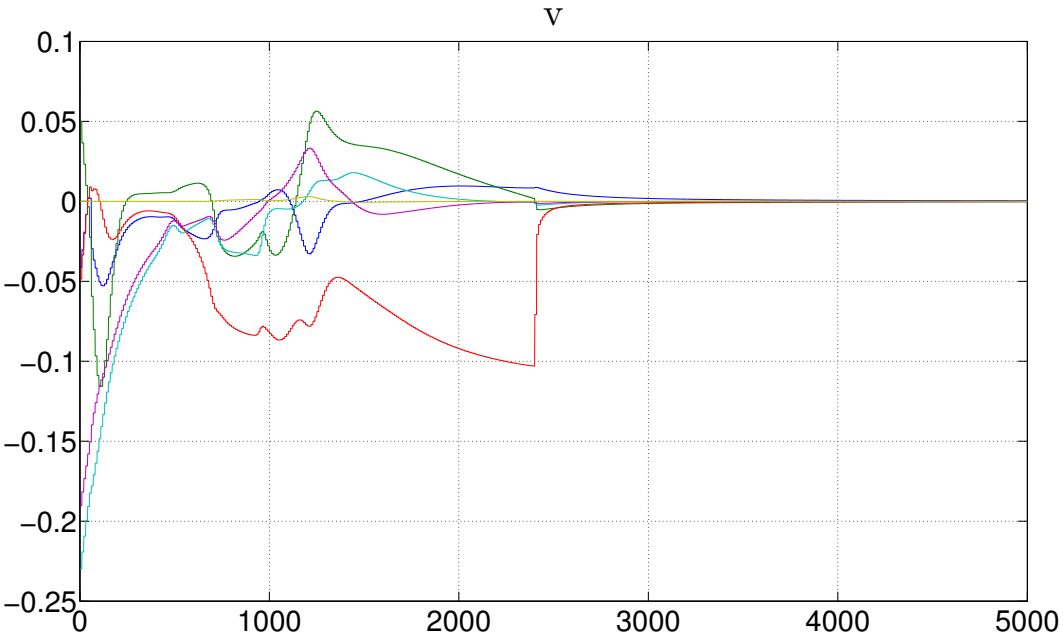
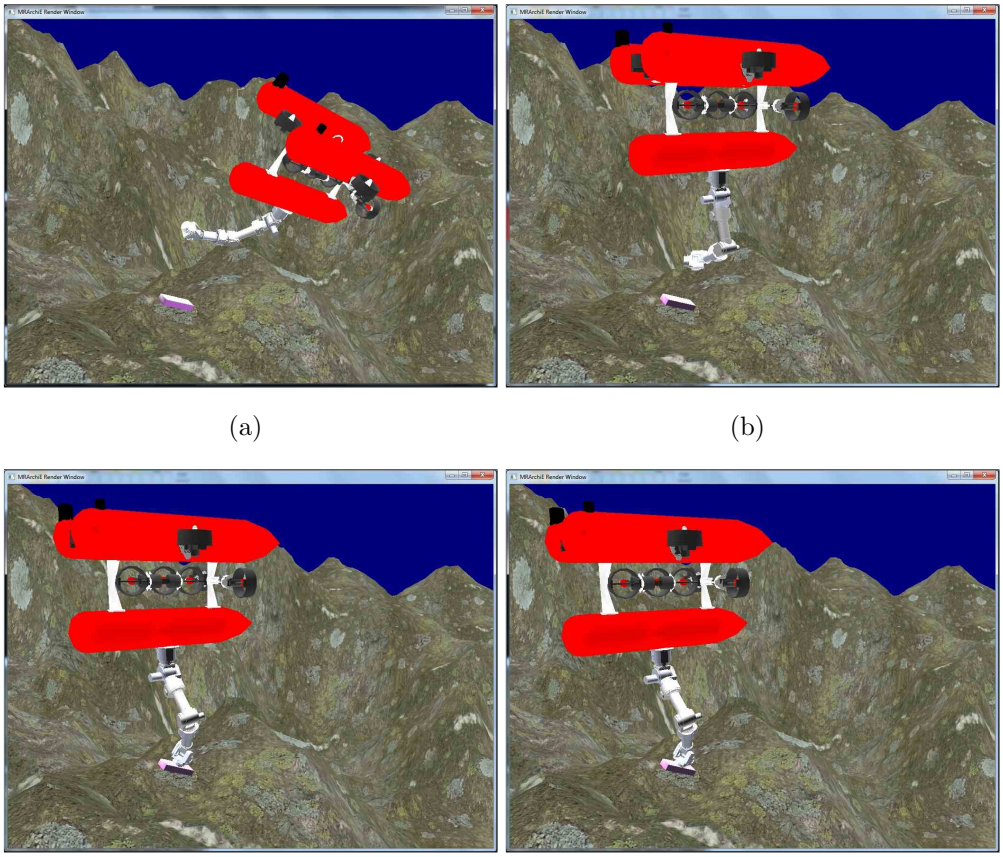


Figure 4.17: Test 1: the generated v for the vehicle



(a) (b)

(c) (d)

Figure 4.18: Test 2: snapshots of the movement of the I-AUV

Test 2

In the successive test, the object is placed in $[0\ 0\ -3]$ with a rotation of $[1\ 0.5\ 0.2]$ radians. The vehicle starts at $[1\ 1\ -2]$ with an initial rotation of $[\pi/6\ \pi/8\ \pi/4]$. Figure 4.18 shows the movement of the I-AUV, with frames captured each 10s.

In this case, the vehicle starts too close to the object along the z axis and too far away along x and y. As it can be seen from the α values shown in Fig. 4.19, both the camera height task and the camera distance task are active. Moreover, the camera centering and the horizontal attitude are also active with $\alpha = 1$. Figures 4.21, 4.22 4.23 and 4.24 show how the system manages to accomplish all the required tasks, by bringing all the set rate tasks within their respective boundaries. Moreover, Fig. 4.20 shows how the system also approaches some joint limits, but successfully manages to keep them within the predefined thresholds.

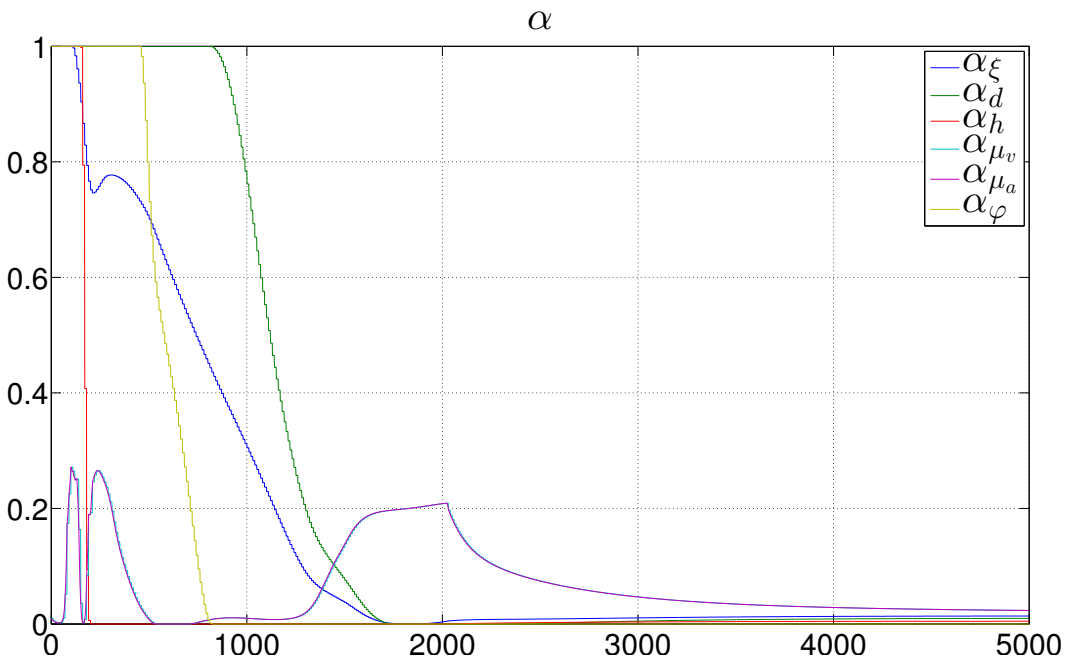


Figure 4.19: Test 2: time behavior of the α functions of the set rate tasks

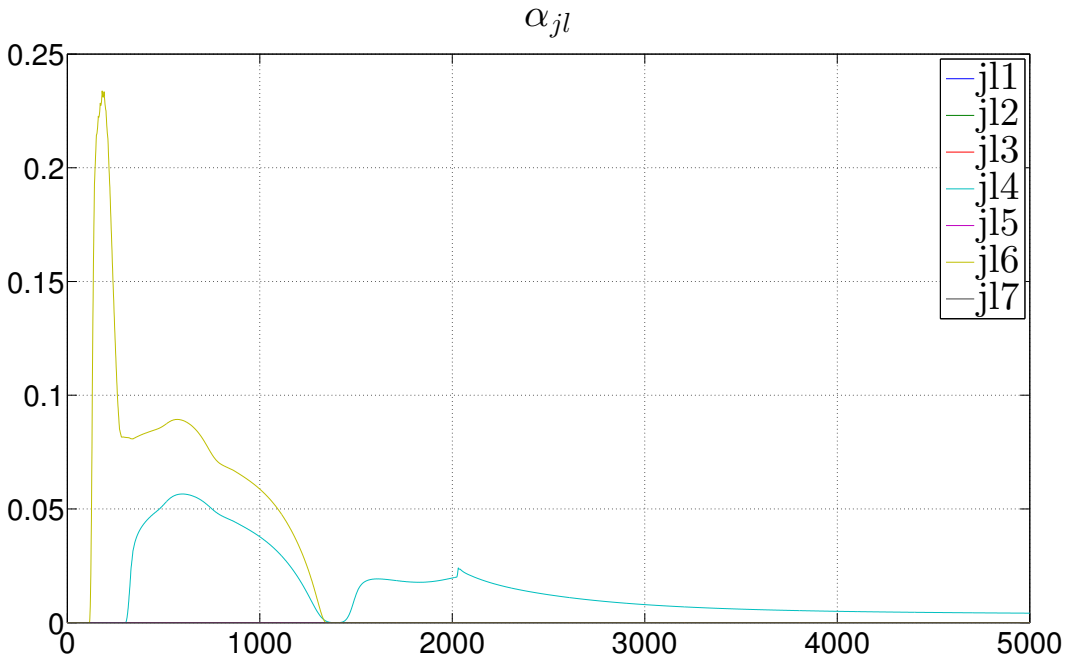


Figure 4.20: Test 2: time behavior of the α functions of the joint limits set rate task

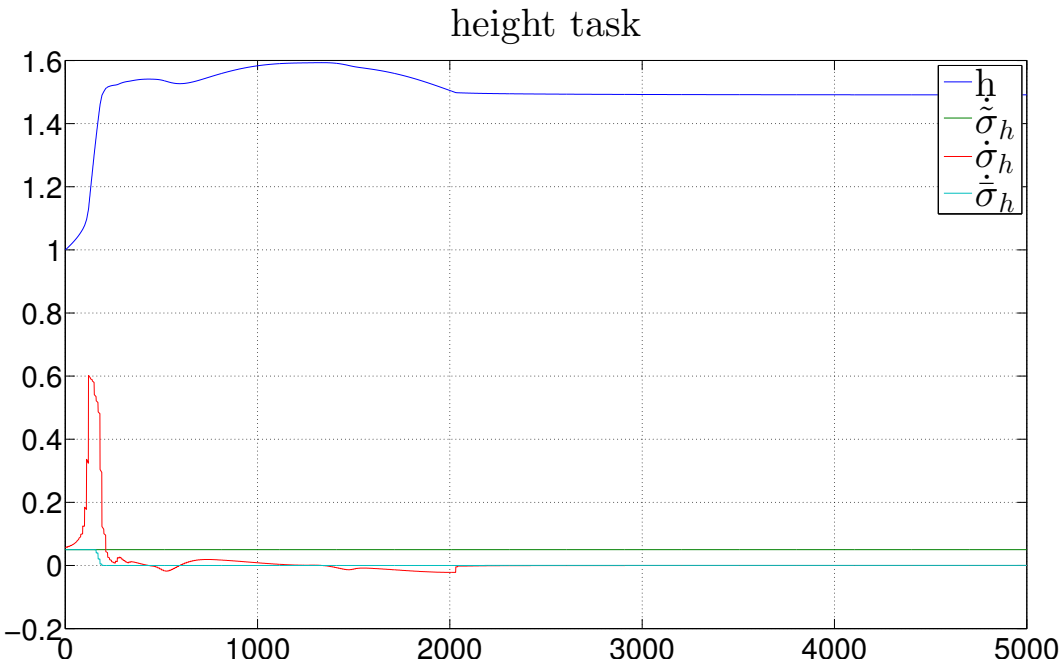


Figure 4.21: Test 2: time behavior of the quantities related to the camera height task

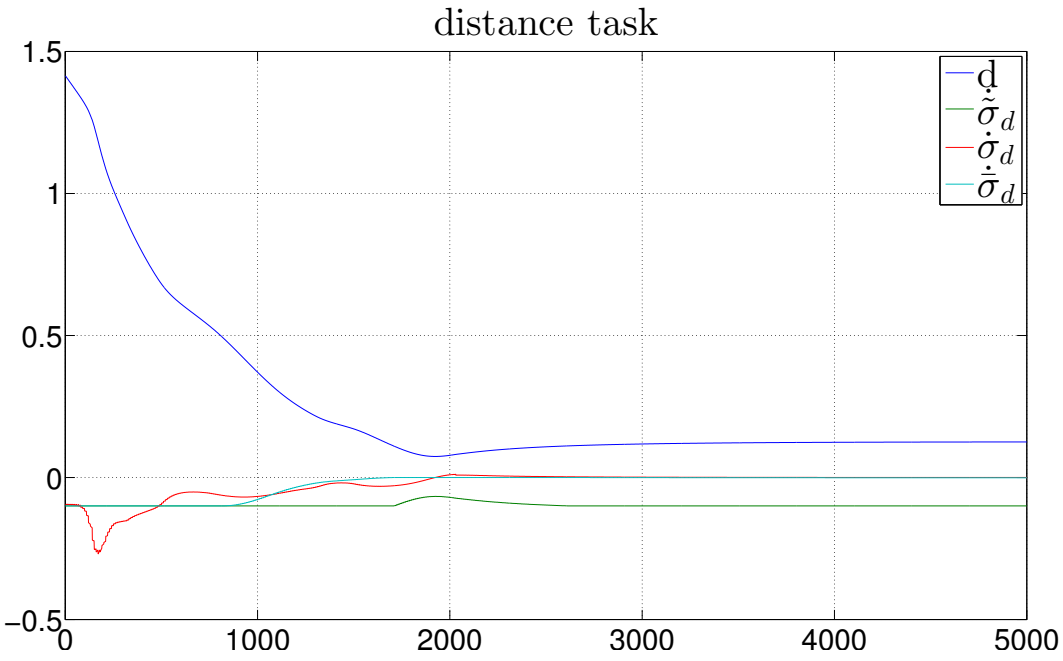


Figure 4.22: Test 2: time behavior of the quantities related to the camera distance task

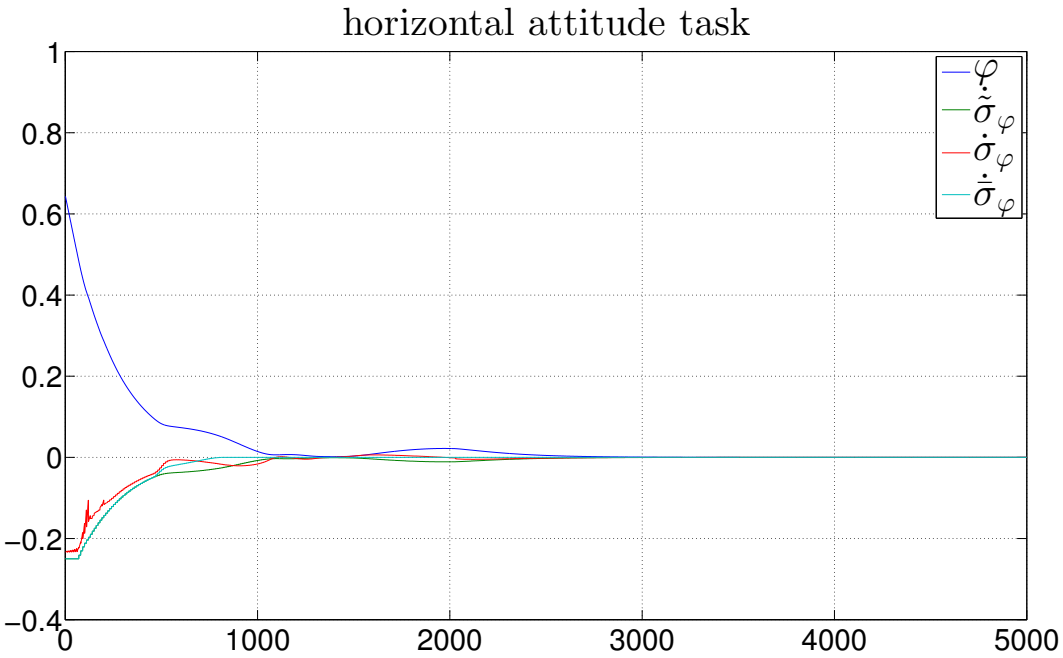


Figure 4.23: Test 2: time behavior of the quantities related to the horizontal attitude task

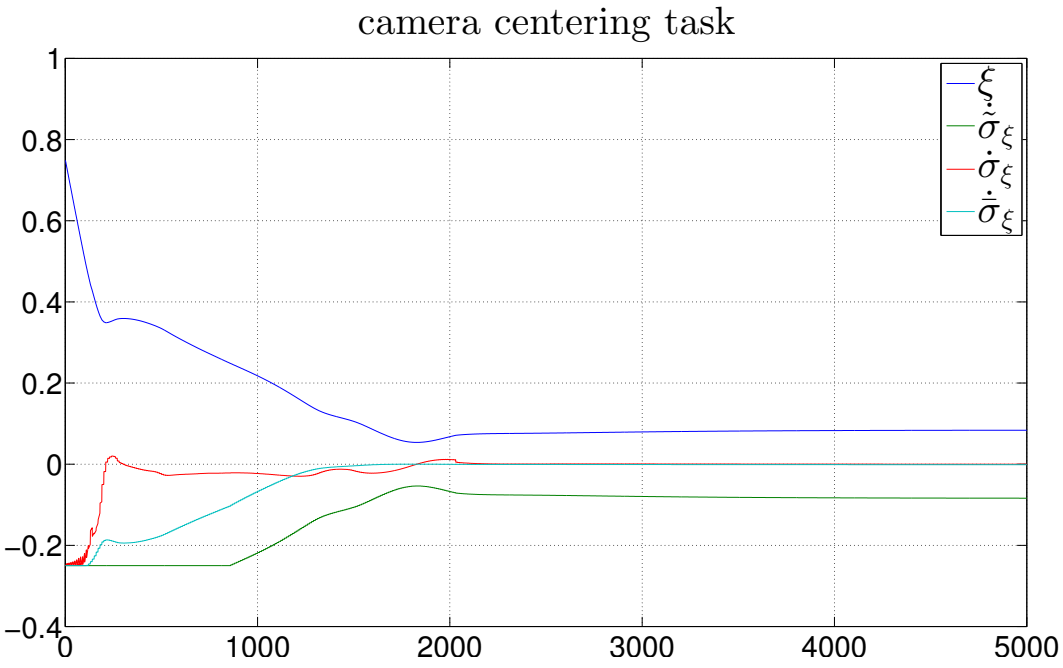
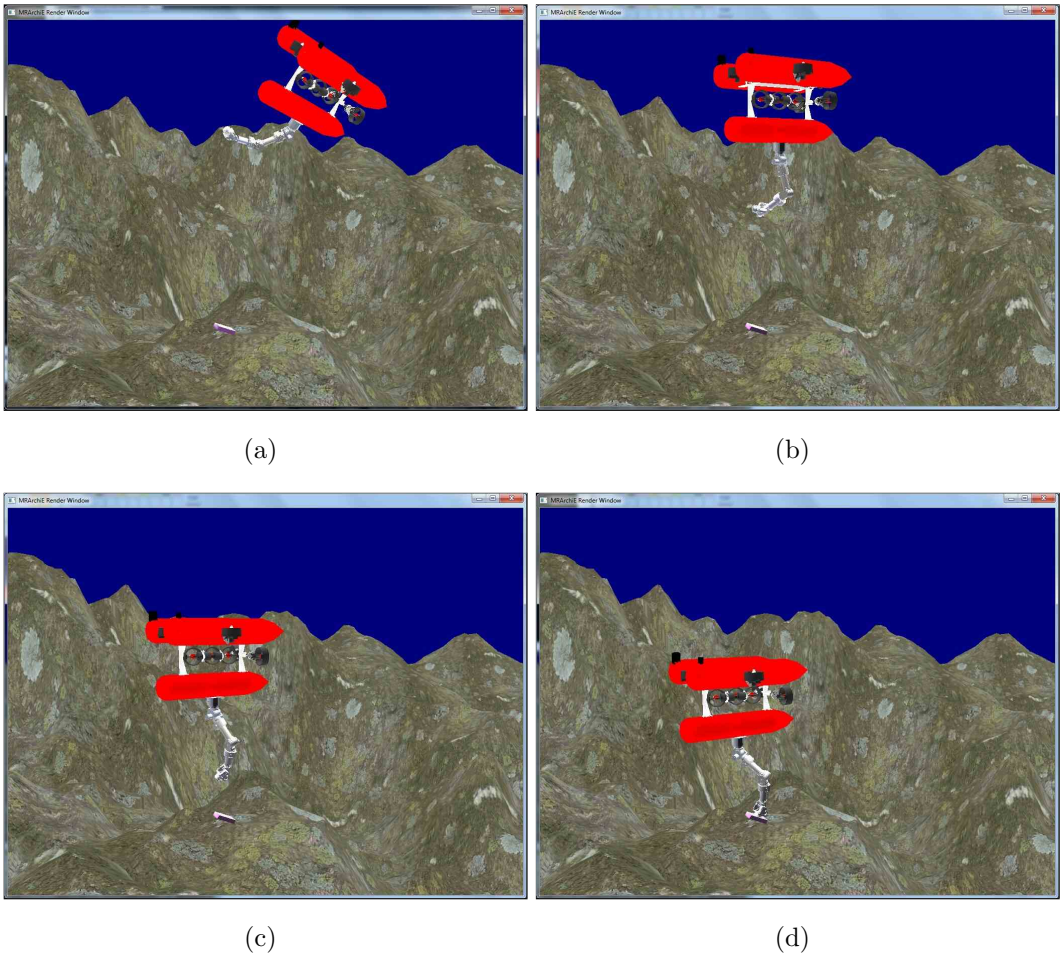


Figure 4.24: Test 2: time behavior of the quantities related to the camera centering task



(a) (b)

(c) (d)

Figure 4.25: Test 3: snapshots of the movement of the I-AUV

Test 3

In the last test, the object is placed in $[0\ 0\ -3]$ with a rotation of $[1\ 0.5\ 0.2]$ radians. The vehicle starts at $[1\ 1\ 0]$ with an initial rotation of $[\pi/6\ \pi/8\ \pi/4]$. A sinusoidal noise of 0.1Hz frequency is added to the vehicle velocity along the x axis of the inertial frame. Figure 4.25 shows the movement of the I-AUV, with frames captured each 10s.

Figure 4.26 shows the time behavior of the α functions, Fig. 4.27 shows the end-effector position and Fig. 4.28 shows the vehicle position. These last two figures show that the noise is rejected as soon as the system is close to the final position and the saturation effects are over. Indeed, while the vehicle still has oscillations in its x and y position, the end-effector's position is constant (see after instant 2500).

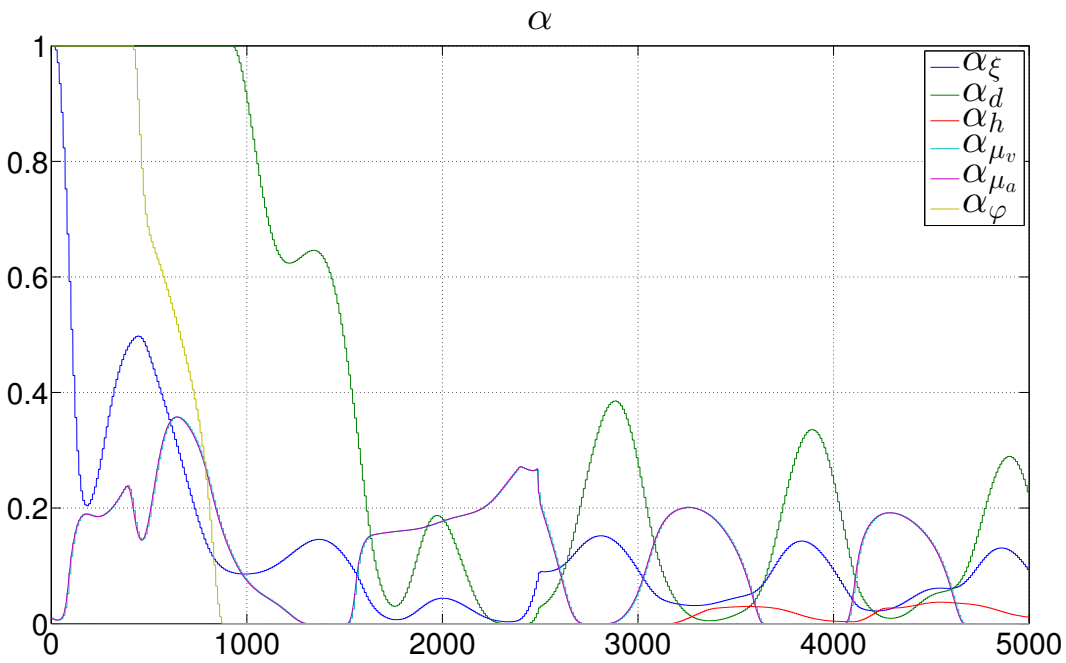


Figure 4.26: Test 3: time behavior of the α functions of the set rate tasks

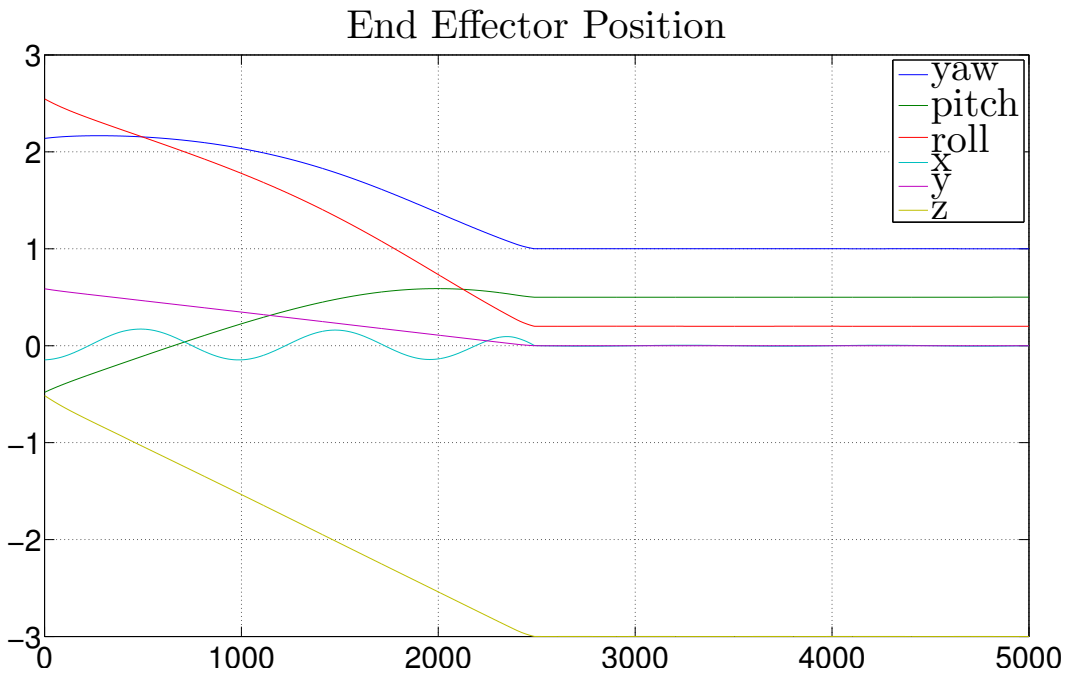


Figure 4.27: Test 3: evolution of the position of the end effector expressed in the inertial frame $\langle 0 \rangle$

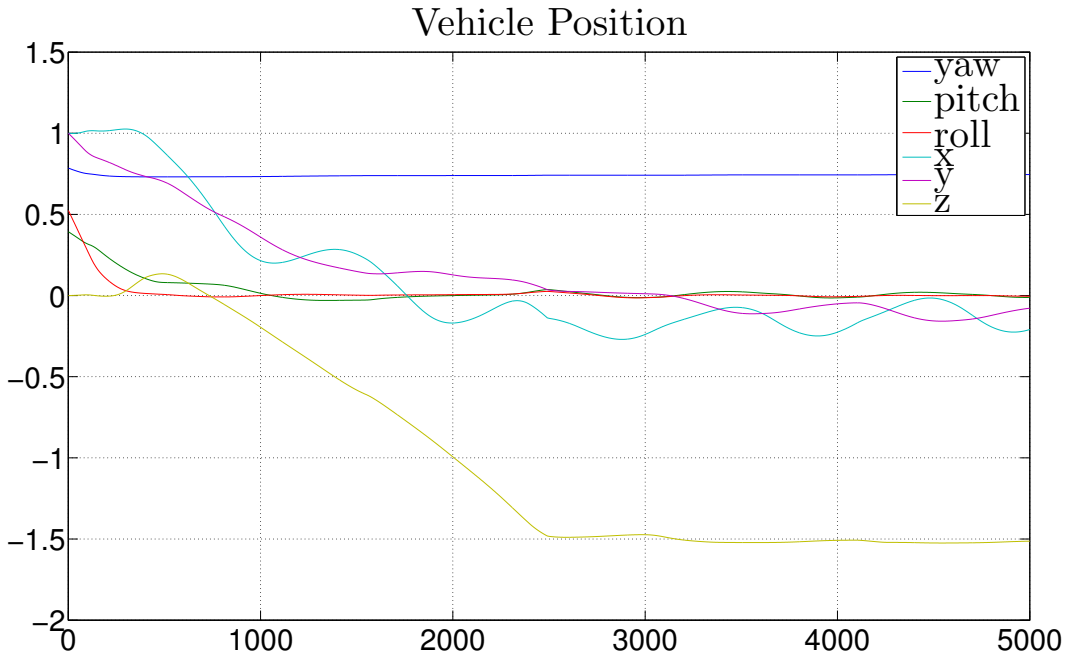


Figure 4.28: Test 3: evolution of the position of the vehicle expressed in the inertial frame $\langle 0 \rangle$

Chapter 5

Real Time Software Architecture for the Control of USVs and AUVs

Quite often, people working and doing researches with robots need to accomplish complex tasks, but seldom they have the possibility of implementing them with the bare services of a RTOS without spending a lot of time debugging the software and its interaction with the operating system. This delay created by having to program the direct interaction with the RTOS could have been spent instead in debugging, testing and improve the algorithms or tasks that the robot has to do, which is the real goal of the people working with robots in the first place.

This is the first and foremost reason for creating a framework that offers higher level services to the programmer: to ease the programming and render it less error prone, and consequently to lower the programming and debugging time, enabling the final user to be able to spend more time on his task at hand. Even experienced programmers that routinely deal with RTOS have a great advantage by exploiting the framework's capabilities, as many common problems are already solved and need not to be done each and every time a new project is undertaken. Having to deal with established Application Programming Interfaces (APIs) is certainly a clear advantage rather than reimplementing them each time (with the possibility of introducing new errors to be debugged).

However, the creation of higher level interfaces can sometime have the drawback of forcing the programmer into building its applications following strict

guidelines: for example, component-based approaches force every task to be implemented as a component; while this approach certainly has advantages in specific projects, in others being forced to follow it can even increase the complexity of the application, if compared to the freedom of choice given by dealing directly to the operating system and writing the software in the most appropriate way.

Thus, a key factor that has to be analyzed while creating a framework is how to deal with the complexity and ease it offering higher level interfaces and services, while at the same time keeping the broadest range of possibilities and choices available to the programmer.

To make matters even more complex, the required level of interaction is pretty high and thus limiting the level of abstraction that the framework should reach.

Another relevant issue is the possibility of reusing or testing the same software under different hardware or even RTOS. The portability of the code can then be an important goal to pursue, as exploiting previous implementations of certain algorithms can save time when dealing with a new project that focuses on other issues. Obviously, such goal cannot be fulfilled if the algorithms are interacting directly with the RTOS, but instead the applications need to interface with an appropriate abstraction layer where all the RTOS-specific calls are contained. In this way, only that part would need to be changed when "porting" the software to another operating system.

All the aforementioned problematics have been taken under consideration and have led the design and development of the *Workframe*. To meet those objectives, the framework has been divided into different functional layers (as seen in Fig. 5.1):

1. KAL: Kernel Abstraction Layer, taking care of the lowest level of interaction with the RTOS and providing portability of all the code;
2. WF: Workframe, handling the scheduling, system resources and execution;
3. NET: Network Communication, which controls the communications with remote programs;
4. BBS: BlackBoard System, a data-unaware information exchange system between tasks;
5. CMAT: Mathematical Library, providing basic matrix and vector operations common in robotics tasks;

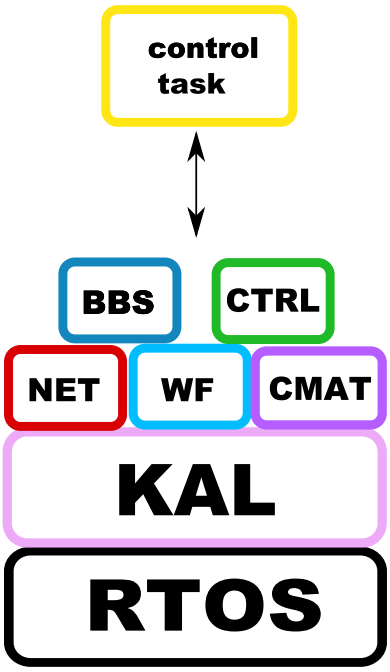


Figure 5.1: Overall schema of the *Workframe* layers

6. CTRL: Control Library, providing a hierarchy of controllers and interfaces for the control of robotic arms and vehicles.

All the libraries have been implemented in C++, following the object oriented paradigm, in contrast to the RTOS level APIs which are C functions. This approach permits to achieve data hiding, improves reusability and simplifies the interfaces between the framework and the final user. This has rendered the code easy to read and to use, since for every resource and functionality a dedicated class with a specific interface has been created, without exposing the internal functioning. Moreover, in order to enhance the distinction between the different abstraction levels, each one of them has been included in an homonym namespace, in order to separate the resolution namespaces and to clearly signal, in every part of the code, the belonging of each object and method used.

In the following, the main concepts and key aspects of each layer is detailed in its own section.

5.1 Kernel Abstraction Layer

This layer is the base of all *Workframe* and it is the only one directly interfacing with the underlying operating system. The goals of this layer are:

1. to encapsulate the RTOS resources APIs in easier to use classes;
2. to provide objects to access to I/O devices;
3. to handle the creation of tasks;
4. to deal with the scheduling of the system;
5. to guarantee the portability of the higher levels.

This last point affects all of the above ones, as in order to guarantee the portability, all the classes have to provide unique interfaces independently of the underlying operating system. During the design phase, four main RTOS have been considered: RTAI, QNX Neutrino, VxWorks and RTEMS. Thus, only the subset of resources and services provided by all these operating systems is provided by the KAL, as otherwise some calls could not have been implemented under one of the other target systems. Note however that, if the user needs to exploit some specific calls of the RTOS, he is free to do so. Nevertheless, he will have to manually do the porting of that piece of code if he needs to reuse his application under another RTOS.

Since this layer closely interacts with RTOS, a special version has to be made for each supported operating system. The programmer, when building his application for a specified operating system, has to link the appropriate version of the KAL library written for that particular RTOS.

Given the above portability considerations, the following Inter-Process Communication (IPC) resources have been implemented:

- Mailbox/Queue;
- Shared Memory;
- Semaphore (binary and counting);
- Remote procedure call (RPC);
- Socket;

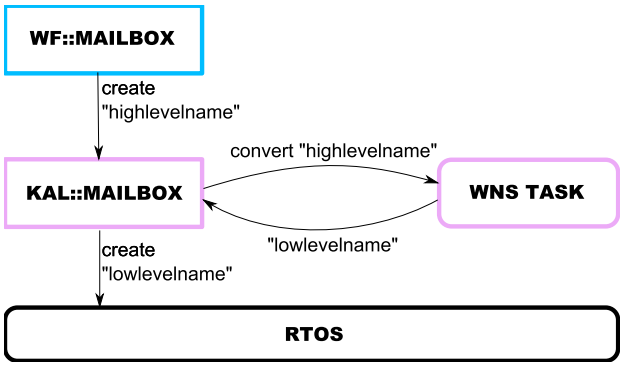


Figure 5.2: Example of name resolution for a mailbox

- Signals.

The main challenge when dealing with the resources and their implementation with portability in mind is surely the problem of abstracting the identifiers. Since each RTOS identifies resources in a different way (be it a number or strings of different length), throughout the framework such names are not allowed. A new standardized method has been used (a name string) and all the calls at KAL level are parametrized in order to take this kind of framework-level identifier only, without ever exposing the actual underlying identifier used by the RTOS.

However, such solution requires the KAL classes to be able to translate an higher level identifier (framework level) to a lower level one (RTOS level). In order to do so, a system task called *Workframe Name Server* has been introduced, whose job is to keep name resolution tables between the framework identifiers and RTOS ones, much like a domain name server translates strings into Internet addresses (hence the name); see Fig. 5.2 for an example of name resolution.

Note that such resolution is mainly performed during the initialization phase of the system and does not induce overhead during the running phase, because the results are cached locally inside the objects to avoid multiple resolutions of the same names.

5.2 Workframe Library

The most important part of the framework is the layer called *Workframe* which gives the name to the framework itself. This layer contains the core functionalities of system management, which includes the handling of the system state, the

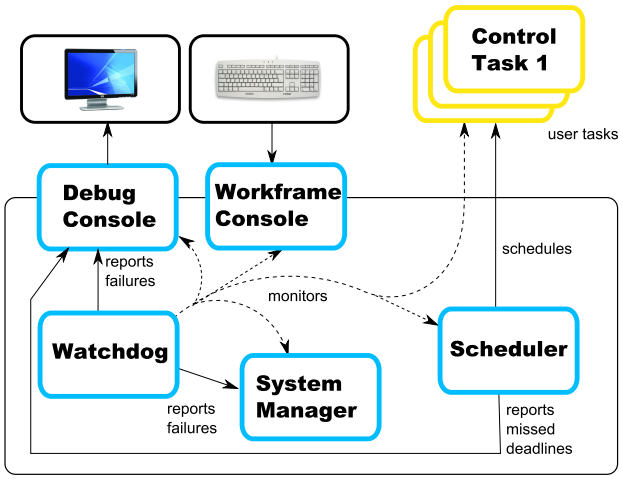


Figure 5.3: The *Workframe* tasks and their interaction with the user tasks.

proper scheduling of all tasks and control over the deadlines, the creation and request of resources and the general task functioning.

At this level, all the resources previously mentioned are available, with the addition of methods to "create" or "request" them, that enforce special policies (i.e. no creation of resources during the running phase); however, more complex resources, built upon one or more basic ones, are also available: for example, a circular buffer class based on a shared memory object is present inside this level.

In order to offer all the services, different system tasks are required and are loaded at the start of the system:

- System Manager Task;
- Scheduler Task;
- Debug Console Task;
- Workframe Console Task;
- Watchdog Task.

Fig. 5.3 depicts the overall set of tasks found at this level and their interaction with the user control tasks. In the following, each system task is briefly explained and their main jobs are detailed.

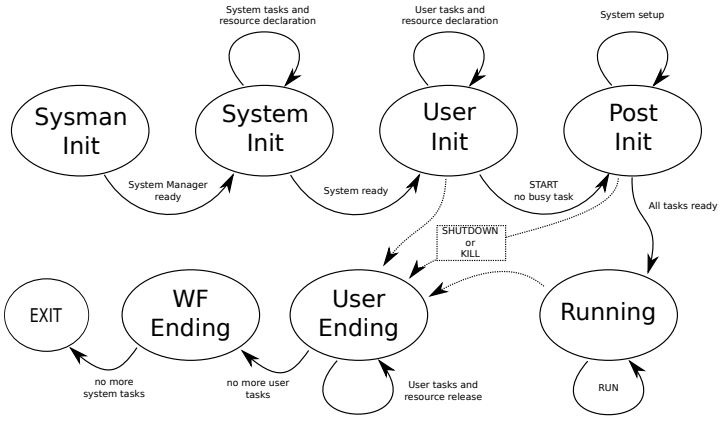


Figure 5.4: System Manager's states and transitions between them

5.2.1 System Manager

This is the core task of the system. It is this task that controls the evolution of the system state, and the unique handler of all the requests for resources or tasks. Indeed, many of the *Workframe* classes have inside them an hidden communication with the System Manager, in order to ask for permissions.

Its jobs can be summarized as follows:

- to supervise the creation and destruction of all the tasks;
- to handle the allocation and deallocation of resources requested by tasks for their functioning;
- to regulate the functioning of the system in the different states.

To those aims, the System Manager keeps tracks of the state of tasks and resources through two tables:

- a task table to store information about tasks currently loaded in the system;
- a resource table to keep track of allocated resources and which task is using them.

The System Manager is implemented as a finite state machine, so its behaviour is strictly conditioned by the current system state. A representation of all the states can be seen in Fig. 5.4.

5.2.2 Scheduler

The Scheduler is a synchronous task whose job is to release the periodic tasks' synchronization semaphores at the user's desired rate; to this aim, it keeps a scheduling table with the information on after how many cycles it has to release each particular semaphore. The table is created and filled during the *User Init* phase, where all the task are declared; the System Manager provides automatically for the creation of the synchronization semaphore for each periodic task.

During the *Running* state, the scheduler does its job of releasing the user tasks at the desired frequency. However, while doing so, it also provides an important mechanism for the debugging of the overall system by controlling if any task is not respecting its deadline. To do so, it simply checks if the relative synchronization semaphore is free or not: in case of normal functioning, the periodic task will be blocked on the semaphore waiting for the next step. Instead, if the task has not yet finished its single loop, the semaphore will be free.

When the Scheduler detects a missed deadline, it promptly signals it with a log message directed towards the Debug Console.

5.2.3 Debug Console

Debugging software is one of the longest parts of software development; when dealing with real time software, the standard tools for doing it cannot be used because they obviously break the real time requirements. With this in mind, a unified log message mechanism has been developed in order to properly debug the framework's libraries, while at the same time providing the user with useful information in order to let him know if anything is going the wrong way. Indeed, all the libraries send log messages to a predefined Mailbox, specifying: the name of the library, the level of the message (verbose, info, notice, warning, error), the message itself and the task inside which the library has generated the error. The Debug Console simply waits for new messages on this mailbox and outputs them to the screen each time it receives a new one.

5.2.4 Workframe Console

The Workframe Console is the user's way to give commands to the framework. They range from necessary commands to start and stop the functioning of the

system, to debugging ones that require specific libraries to print to screen some key tables.

5.2.5 Watchdog

The Watchdog task is in charge of periodically monitoring the state of all system and user tasks during the runtime. In case any task has stopped working, the Watchdog immediately reports the failure to the user with a message in the Debug Console and to the System Manager, which, depending on the actual policies, can decide to issue an automatic shutdown of the system in order to let the user investigate the problem and fix it.

5.3 Network Communication Library

The NET library contains all the classes and functions needed to perform remote communications. In particular, the well known Client/Server paradigm is implemented with dedicated classes, making connecting to a remote tasks extremely easy, as the underlying details of the communication channels are hidden. At the moment, Bluetooth and Ethernet are the physical channels supported. However, it is planned to add support for other channels like CAN bus and similar in the near future.

5.4 BlackBoard System

This part of the framework is in charge of a data sharing mechanism between tasks, both local to the real time system and remote ones. The main goal is providing a very clean and easy to use way for sharing data between tasks, taking care of mutually exclusive access to the information.

Three dedicated tasks are assigned to the fulfillment of those aims:

1. the BlackBoard Manager;
2. the Execution Manager;
3. the Connection Manager.

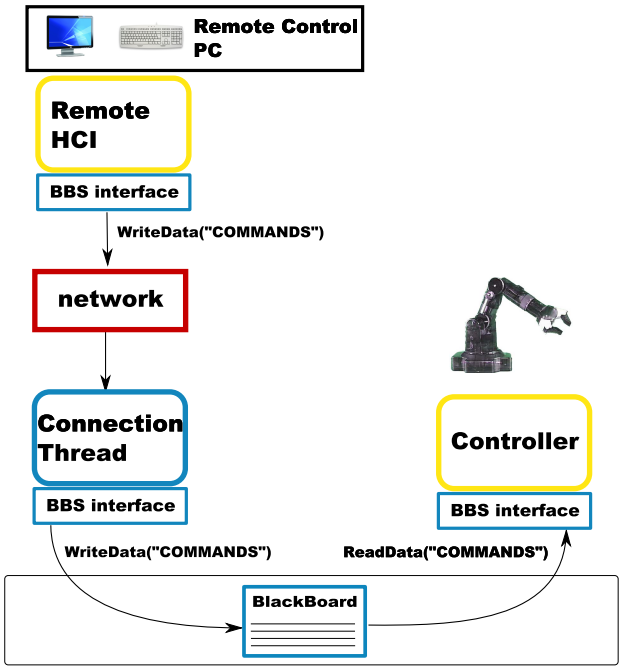


Figure 5.5: Exchange of data between a remote task and a local task.

The first one handles the declaration of all data and it is in charge of creating the resources to implement mutual access. Also, just before going in the *running* phase, the BlackBoard Manager creates the "BlackBoard", which is a shared memory segment containing both the data and an header part used internally by the classes of the BlackBoard System.

Moreover, using this library it is possible to write tasks that are normally blocked for a request of execution by another task, and upon receiving it, they execute once and eventually write their outputs to the blackboard. All the management of the request/execution/acknowledge is handled by the Execution Manager. An example is reported in Fig. 5.6, where the Path Planner task is normally blocked and executes only upon request, which happens when the remote Human Computer Interface (HCI) creates a new reference point for the vehicle to go to.

Finally, the Connection Manager is the task whose job is to interface with all the remote tasks. It exploits the services the NET library, and spawns a dedicated thread to handle each connection with a remote task, which in turn relays all the request from and to the remote system.

An important feature of this library is that it allows remote tasks running over non real time operating systems such as Linux or Windows to connect to a remote framework and easily exchange data. This possibility enables the user to

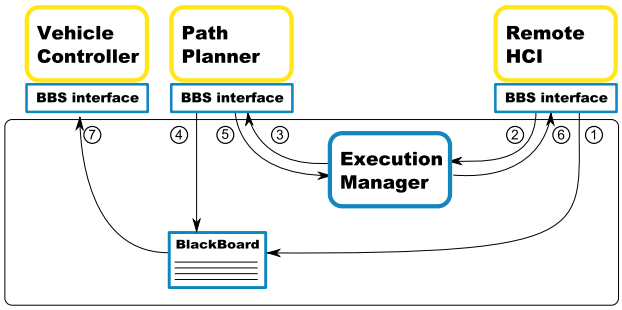


Figure 5.6: Example of use of the request/execution/acknowledge mechanism: 1) the user, through the remote HCI, requests the rover to go to a new position and writes it to the blackboard; 2) the remote HCI asks the framework to execute the Path Planner task; 3) the Execution Manager forwards the request to the Path Planner task. 4) the Path Planner executes and computes a new path, which is written to the blackboard; 5) the Path Planner ends its execution and informs so the Execution Manager; 6) the remote HCI is informed of the conclusion of the Path Planner execution 7) the vehicle controller continuously read from the blackboard and now finds the new path to follow.

easily develop HCI on desktop systems, for example to plot data from the robotic system. Moreover, since the remote interface and the local one are the same, some parts of the code can be debugged and then moved to the actual system afterwards. However, the remote part does not run in real time.

5.5 Mathematical Library

The main idea behind the mathematical library is to have a minimal set of classes to handle matrices and vectors, with particular methods, very often used in robotics, for operations like the Singular Value Decomposition, to transform Roll-Pitch-Yaw vectors to their correspondent Rotational matrices and vice versa, and so on. Exploiting the C++ overloading of operators, it was possible to obtain a very clean implementation of mathematical formulae. Indeed the following formula:

$$\dot{z} = (\Gamma K)^{\#} (\dot{w} - K\dot{x})$$

can be implemented as:

```
zdot = (Gamma * K).PseudoInv() * (wdot - K * xdot);
```

which makes the debugging of the code really easier than using a C library (where each operation would have required a separate function call).

5.6 Control Library

The control library aims to improve the reuse of code when developing the control algorithms. For example, controlling a robotic arm requires a lot of software which is not strictly related to the very simple kinematic inversion law. Such software must take care of:

- the internal state of the controller, as usually the controller are finite state machines, whose behaviour is strictly dependent on their current state;
- safety measures such as checking the joints' end of race;
- parsing the commands from the user;

and so on. These routines are generally independent of the specific robotic arm, or at most can be parametrized by few values. Thus, it makes no sense to re-implement such software each time a new arm must be controlled.

With this in mind, the control library comprises four main classes:

- Controller, which is the finite state machine that implements the management of the internal state of the controller;
- Model, which contains the model of the robot and is used by the Controller for computing its control law;
- ControllerInterface, which is the interface that the rest of the software can use in order to send commands to the controller itself;
- Driver, which is the class whose job is to interact with the underlying hardware, in order to send the control signals computed by the controller to the robot.

Such classes are at the top of their hierarchy and are thus extremely generic. Each of them is then derived for the control of specific robots, such as: arms, vehicles and grippers. Thus, as example, there are four derived classes named ArmController, ArmModel, ArmControllerInterface and ArmDriver. Each of these classes exploit all the code implemented in the parent classes and add a lot of features which are strictly connected with the control of robotic arms.

The specific controller is the implemented by deriving each of those four Arm classes (if necessary). Such hierarchy permits to reuse most of the code, leaving

only few robot-specific details to be implemented. As a matter of fact, this allowed to reduce the number of lines of code for a Unimate PUMA controller from about 1200 to just 300.

Chapter 6

Conclusions

Different algorithms and methodologies for the coordination and control of marine vehicles have been presented in this thesis.

First, the results concerning USVs have been shown in Chapter 2. The Chapter introduced the joint research project between DIST and Selex SI called Swarm Management Unit, a tool for the control and coordination of a team of USVs performing patrolling and surveillance activities within civilian harbors. The research has thus been focused on algorithms and methodologies for the operation of USVs inside manned areas. As a consequence, the first of the proposed techniques tackled the challenging problem of the obstacle avoidance, which is probably the main critical issue that USVs have to face when they are operating in manned areas. The proposed obstacle avoidance algorithm exploits the information of the current speed and heading of the vessels operating in the area, an information extracted from the radar system. The planner, given a specific goal position, finds the time optimal path for the vehicle, considering the vessels moving with their current speed and heading. To do so, the algorithm considers every vessels as rectangular obstacles and the USV as a pointwise robot, via the use of a bounding box large enough to guarantee that no collision can occur whenever the USV is outside it. The planner exploits the vertexes of the bounding box as the delimiting boundaries for the obstacles, and uses their initial position and kinematics to create via points in a three dimensional space (x, y, t). Then, each path between the current analyzed position and the new via point is checked for possible collision with other bounding boxes and eventually discarded. An A^* search is employed to select the next via point to explore, until the goal is reached. On the assumption that the USV moves faster than the obstacles, that within the

specific security scenario is quite reasonable, the search yields the optimal result.

The successive problem that has been dealt with is the interception of unknown vessels, to allow a quick classification. It has been shown that getting as close as possible to the unknown vessel with minimum time is just a special instance of the above obstacle avoidance problem, if the goal, instead of being stationary, is considered as moving with a certain speed and heading. The performance of the system are strictly depending on the positioning of the team of USVs whenever an unknown vessels is detected and needs to be classified. Thus the off-line problem of positioning the team in order to maximize the performances of the fleet of vessels has been tackled. A particular scenario has been considered, where specific assets inside the harbor must be protected and the performances must be tuned for them. Two criteria have been detected in order to drive the optimization process. The first criterion is to be able to guarantee a minimum interception distance d_{th} from the asset, given that the unknown vessels can at worst be detected at a certain distance r, with $r > d_{th}$. In the considered scenario this is not an unreasonable assumption, since the vessels must identify themselves when approaching a civilian harbour. The result of the optimization is both the number of vehicles that are necessary to achieve the desired performance and their positioning. The remaining USVs are instead exploited to minimize the interception time. Such criteria is important to spread the vehicles in all the considered area, and not just nearby the assets. Finally, the development of a prototype of USV and its control board has been presented, along with some field tests of the localization algorithms done with a RC car.

The following Chapter 3 has instead presented the work done for AUVs which is framed within the European FP7 project CO^{3}AUVs, presented in the introduction. The underwater world presents quite different challenges for the autonomous vehicles. Obstacle avoidance, while still being important, is certainly not the primary issue for the autonomy of vehicles. The localization of the vehicle is very difficult due to the fact that the GPS signal is not available underwater. Thus, the localization is usually based on estimating the range from buoys or transponders using acoustic waves and their time of flight. However, in the underwater case, the acoustic waves do not propagate in a rectilinear way due to the anisotropy of the channel, since the speed of sound depends on the salinity of the water and its temperature. All these considerations make the localization of an AUV a

challenging problem, and indeed it is still a very active topic of research. Due to its importance, two different algorithms have been introduced, differing from the assumptions and the required information that is needed to operate.

The first algorithm that has been presented is named RT^2 and requires the knowledge of the SoS profile, which is assumed depth-dependent only. On such basis, it was shown that the propagation of acoustic waves is periodic. This allowed to structure the algorithm in order to use two lookup tables containing the partial travel times and distance between different depths, calculated off-line. It has been shown that with simple operations, the range between the emitter and the receiver can be calculated, given that the depths of both are known. Simulative results have demonstrated that the algorithm yields the solution with a sub-millimeter precision whenever all the assumptions hold. Moreover, a sensitivity analysis has been included which shown that even in presence of uncertainties in the SoS profile the proposed algorithm still outperforms a straight ray approximation in all cases.

The second localization algorithm that has been proposed does not assume any a-priori knowledge of the SoS profile, as its availability requires a time consuming sampling of the water column, which cannot be always possible. Thus, the algorithm assumes a straight ray propagation and estimates the best value of the SoS, in order to minimize the least square error. This technique has already been implemented on board the eFOLAGA, the AUV that is extensively used by our research team.

Chapter 4 has presented the work that has been performed for the TRIDENT European project regarding the I-AUVs. In the project, currently on-going, our research unit has to tackle the challenging problem of the free floating manipulation. In particular, our objective is the coordinated control of the movement of the arm and the vehicle, in order to increase the autonomy and dexterity of the system compared to the previous sequential approaches. Thus, a priority based dynamic programming approach has been presented. All the goals of the system have been classified and implemented as either set-rate task, or precision task. The main difference between these two kind of tasks is that a set-rate task is accomplished once the monitored quantity is within certain bounds (inequality constraint), whereas the precision task is considered fulfilled only when the error has been zeroed (equality constraint). A priority between those task has

been established, where tasks such as keeping away from joint limits or keeping a good manipulability measure have an higher priority rather than the end effector position control. Simulative results have been presented and show that the overall structure accomplishes all the goals, even in presence of vehicle velocity disturbances.

The successive Chapter 5 has outlined the software framework that has been developed over the course of the last 4 years by our research team. The principal objective of the framework is to ease the programming effort required to implement the control algorithms of new robots, or to improve the existing ones. To this aim, the framework has been divided into different libraries and task with precise aims. The KAL library is dedicated to render all the software independent from the underlying operating system, making possible to use the framework with different RTOS. The WF library takes charge of the resources of the system and provides easy to use scheduling mechanisms for periodic tasks. The BBS library offers a shared memory based communication mechanism that greatly simplifies the exchange of data between control tasks, and even with remote interfaces. A mathematical library has also been developed, which permits to implement the control laws, such as the one developed for the TRIDENT project, with extreme ease. A network library has also been included inside the framework, to allow an easy communication with remote tasks. Finally, a control library has been devised to provide with the necessary code for handling finite state machines and basic loops for the control of vehicles and robotic arms.

6.1 On-going Activities @ GRAAL

The current works are focused on the implementation and field test of many of the algorithms presented in this work. As presented in Section 2.5, the prototype of USV is currently under construction. At this time, the control board of the USV is being finalized and realized and should be ready for mid-March. The assembled prototypes should be ready for use near April. The availability of 10 of these vehicles will allow to test the obstacle avoidance algorithms as developed in Section 2.2 and the interception algorithms as proposed in Section 2.3. A demo with Selex-SI is currently scheduled for September.

In parallel to the development of the USVs, the current works are also focused

on the implementation of the free floating manipulation control, as shown in Section 4.2. Indeed for May it is expected to have an integrated demo of the capabilities of the overall system in a simulated environment, to be then replicated as much as possible in a real scenario in October in Majorca.

6.2 Future Research Activities

For what concerns USVs, the future research activities will be aimed at extending the algorithms presented in this work. Possible extensions are the inclusion of a minimum value for the curvature range of the USVs, instead of assuming that the vehicle can instantly turn in any direction. A second extension is to implement in the obstacle avoidance algorithm the use of the COLREGS, especially needed when the vehicle needs to move without any emergency. Then, another important trend of activity will be aimed at the cooperative intercept of vehicles. In this work, it was assumed that one USV needs to reach the position of the unknown vehicle as fast as possible, to allow a fast classification. An extension of this would be to use multiple vehicles in a cooperative manner. This would allow to carry out more complex tasks, like for example surrounding the unknown vehicle, closing any possible escape routes, or at least any routes leading to sensible assets in the area.

The work on I-AUVs will be extended by considering more complex tasks to be performed, like the grasp of an object in a cluttered scenario, where the arm needs to avoid obstacles in order to reach the object itself. Moreover, the cooperation between two I-AUVs could also be a topic of future activities, notwithstanding the very limited communication bandwidth when acoustic communication is used. However, if some sort of cooperative manipulation is to be performed, there would be the possibility to use the still prototypical optical communication, since its very short range would be sufficient for the application.

References

[1] T. J. Crone and M. Tolstoy, "Magnitude of the 2010 Gulf of Mexico oil leak," *Science*, vol. 330, no. 6004, p. 634, October 2010.

[2] R. Camilli, C. M. Reddy, D. R. Yoerger, B. A. S. Van Mooy, M. V. Jakuba, J. C. Kinsey, C. P. McIntyre, S. P. Sylva, and J. V. Maloney, "Tracking hydrocarbon plume transport and biodegradation at Deepwater Horizon," *Science*, vol. 330, no. 330, pp. 201–204, October 2010.

[3] C. R. German, D. R. Yoerger, M. Jakuba, T. M. Shank, C. H. Langmuir, and K. Nakamura, "Hydrothermal exploration with the Autonomous Benthic Explorer," *Deep Sea Research Part I: Oceanographic Research Papers*, vol. 55, no. 2, pp. 203–219, 2008.

[4] D. L. Valentine, C. M. Reddy, C. Farwell, T. M. Hill, O. Pizarro, D. R. Yoerger, R. Camilli, R. K. Nelson, E. E. Peacock, S. C. Bagby, B. A. Clarke, C. N. Roman, and M. Soloway, "Asphalt volcanoes as a potential source of methane to late pleistocene coastal waters," *Nature Geoscience*, vol. 3, pp. 345–348, 2010.

[5] G. Griffiths, R. Davis, C. Erikson, P. Frye, T. Marchand, T. Dickey, and R. Weller, "Towards new platform technology for sustained observations," in *First International Conference on Ocean Observations for Climate*, C. Koblinsky and N. Smith, Eds. Melbourne, Australia: Bureau of Meteorology, 1999, pp. 324–337.

[6] M. Dunbabin, A. Grinham, , and J. Udy, "An autonomous surface vehicle for water quality monitoring," in *Australasian Conference on Robotics and Automation (ACRA)*, Sydney, Australia, December 2009.

[7] J. R. Higinbotham, J. R. Moisan, C. Schirtzinger, M. Linkswiler, J. Yungel, and P. Orton, "Update on the development and testing of a new long duration solar powered autonomous surface vehicle," in *OCEANS 2008*, Quebec City, Canada, 2008.

[8] N. A. Cruz and J. C. Alves, "Autonomous sailboats: an emerging technology for ocean sampling and surveillance," in *OCEANS 2008*, Quebec City, Canada, 2008.

[9] M. Caccia, R. Bono, G. Bruzzone, G. Bruzzone, E. Spirandelli, G. Veruggio, A. Stortini, and G. Capodaglio, "Sampling sea surface with SESAMO," *IEEE Robotics and Automation Magazine*, vol. 12, no. 3, pp. 95–105, 2005.

[10] N. Brown, "More than just a remote possibility: USVs enter the fray," *Jane's Navy International*, pp. 14–19, December 2003.

[11] T. Vaneck, J. Manley, C. Rodriguez, and M. Schmidt, "Automated bathymetry using an autonomous surface craft," *Navigation*, vol. 43, no. 4, pp. 407–418, 1996.

[12] J. Manley, "Development of the autonomous surface craft ACES," in *MTS/IEEE Oceans 97*, Halifax, Nova Scotia, Canada, October 1997.

[13] J. Manley and T. Vaneck, "High fidelity hydrographic surveys using an autonomous surface craft," in *MTS/IEEE Oceans 98*, Baltimore, MD, USA, September 1998.

[14] J. Curcio, J. Leonard, and A. Patrikalakis, "Scout - a low cost autonomous surface platform for research in cooperative autonomy," in *Proc. MTS/IEEE OCEANS*, 2005, pp. 725–729.

[15] J. Majohr and T. Buch, "Modelling, simulation and control of an autonomous surface marine vehicle for surveying applications measuring dolphin MESSIN," in *Advances in Unmanned Marine Vehicles*, ser. IEE Control Series, G. Roberts and R. Sutton, Eds. Institution of Engineering and Technology, March 2006, ch. 16, pp. 329–351.

[16] A. Pascoal, P. Oliveira, C. Silvestre, L. Sebastiao, M. Rufino, V. Barroso, J. Gomes, G. Ayela, P. Coince, M. Cardew, A. Ryan, H. Braithwaite, N. Cardew, J. Trepte, N. Seube, J. Champeau, P. Dhaussy, V. Sauce,

R. Moitie, R. Santos, F. Cardigos, M. Brussieux, and P. Dando, "Robotic ocean vehicles for marine science applications: the European ASIMOV project," in *OCEANS 2000*, 2000.

[17] A. Pascoal, C. Silvestre, and P. Oliveira, "Vehicle and mission control of single and multiple autonomous marine robots," in *Advances in Unmanned Marine Vehicles*, ser. IEE Control Series, G. Roberts and R. Sutton, Eds. Institution of Engineering and Technology, March 2006, ch. 17, pp. 353–386.

[18] J. Ebken, M. Bruch, and J. Lum, "Applying unmanned ground vehicle technologies to unmanned surface vehicles," in *SPIE Unmanned Ground Vehicle Technology VII*, Orlando, FL, USA, 2005, pp. 585–596.

[19] H. Ferreira, R. Martins, E. Marques, J. Pinto, A. Martins, J. Almeida, J. Sousa, and E. Silva, "SWORDFISH: an autonomous surface vehicle for network centric operations," in *OCEANS 2007*, Aberdeen, Scotland, 2007.

[20] J. Vaganay, J. J. Leonard, J. A. Curcio, and J. S. Willcox, "Experimental validation of the moving long base-line navigation concept," in *Autonomous Underwater Vehicles (AUV 2004)*, 2004.

[21] F. M. Fallon, G. Papadopoulos, J. J. Leonard, and N. M. Patrikalakis, "Cooperative auv navigation using a single maneuvering surface craft," *International Journal of Robotics Research*, vol. 29, no. 12, pp. 1461–1474, 2010.

[22] J. Wang, W. Gu, J. Zhu, and J. Zhang, "An Unmanned Surface Vehicle for multi-mission applications," in *Electronic Computer Technology*, 2009.

[23] R. Muller and C. Brook, "Vessel protection in expeditionary operations: at anchor and in foreign harbours," in *Waterside Security Conference (WSS 2010)*, Carrara, Italy, November 2010.

[24] M. Caccia, "Autonomous surface craft: prototypes and basic research issues," in *Proc. 14th Mediterranean Conference on Control and Automation MED '06*, June 2006, pp. 1–6.

[25] International Maritime Organization, "Colregs 1972," in *Convention on the International Regulations for Preventing Collisions at Sea*, 1972.

[26] M. R. Benjamin, J. A. Curcio, J. J. Leonard, and P. M. Newman, "Navigation of unmanned marine vehicles in accordance with the rules of the road," in *Proc. IEEE International Conference on Robotics and Automation ICRA 2006*, May 15–19, 2006, pp. 3581–3587.

[27] ——, "A method for protocol-based collision avoidance between autonomous marine surface craft," *Journal of Field Robotics*, vol. 23, no. 5, May 2006.

[28] J. Veers and V. Bertram, "Development of the USV multi-mission surface vehicle III," in *5th International Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT 2006)*, May 2006.

[29] J. Larson, M. Bruch, and J. Ebken, "Autonomous navigation and obstacle avoidance for unmanned surface vehicles," in *Proc. SPIE Unmanned Systems Technology VIII*, April 2006.

[30] A. J. Healey, D. P. Horner, S. P. Kragelund, B. Wring, and A. Monarrez, "Collaborative unmanned systems for maritime and port security," in *Proc. of the 7th IFAC Conference on Control Applications in Marine Systems*, Zagreb, Croatia, 2007.

[31] M. Caccia, M. Bibuli, R. Bono, and G. Bruzzone, "Basic navigation, guidance and control of an unmanned surface vehicle," *Autonomous Robots*, vol. 25, no. 4, pp. 349–365, November 2008.

[32] M. Bibuli, G. Bruzzone, M. Caccia, and L. Lapierre, "Path-following algorithms and experiments for an unmanned surface vehicle," *Journal of Field Robotics*, vol. 26, no. 8, pp. 669–688, June 2009.

[33] J. S. B. Mitchell, "A new algorithm for shortest paths among obstacles in the plane," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 83–105, March 1991.

[34] Y.-H. Liu and S. Arimoto, "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles," *Int. J. Rob. Res.*, vol. 11, no. 4, pp. 376–382, 1992.

[35] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[36] J. A. Storer and J. H. Reif, "Shortest paths in the plane with polygonal obstacles," *J. ACM*, vol. 41, no. 5, pp. 982–1012, 1994.

[37] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[38] J. S. B. Mitchell, "Geometric shortest paths and network optimization," in *Handbook of Computational Geometry.* Elsevier Science Publishers B.V. North-Holland, 1998, pp. 633–701.

[39] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Proc. 28th Annual Symposium on Foundations of Computer Science*, 12–14 Oct. 1987, pp. 49–60.

[40] K. Fujimura and H. Samet, "Planning a time-minimal motion amidst moving obstacles," *Algorithmica*, vol. 10, no. 1, pp. 41–63, 1993.

[41] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[42] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[43] C. F. Huang and Y. C. Tseng, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, vol. 10, pp. 519–528, 2005.

[44] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[45] D. Blidberg, "The development of autonomous underwater vehicles (AUV); a brief summary," in *IEEE ICRA 01*, 2001.

[46] A. MacNaughton, R. J. Swanson, and R. White, "A new navy deep-water survey vehicle: the naval oceanographic office's subsurface autonomous mapping system (SAMS)," in *Oceans 2003*, vol. 3, 2003, pp. 1703–1708.

[47] C. Stranne, R. A. Sohn, B. Liljebladh, and K.-I. Nakamura, "Analysis and modeling of hydrothermal plume data acquired from the 85 degrees E segment of the Gakkel Ridge," *Journal of Geophysical Research*, vol. 115, January 2010.

[48] A. E. Nevala, "A glide across the gulf stream," *Oceanus*, 2005.

[49] D. Bingham, T. Drake, A. Hill, and R. Lott, "The application of autonomous underwater vehicle (AUV) technology in the oil industry - vision and experiences," in *FIG XXII International Congress*, Washington D.C., April 2002.

[50] B. Morr, "All quiet on the auv front," *Underwater Magazine*, 2003.

[51] G. Trimble and E. Belcher, "Ship berthing and hull inspection using the CETUS II AUV and MIRIS high-resolution sonar," in *IEEE Oceans*, vol. 2, 2002, pp. 1172–1175.

[52] J. Vaganay, M. Elkins, D. Esposito, W. O'Halloran, F. Hover, and K. M., "Ship hull inspection with the HAUV: US Navy and NATO demonstration results," in *Oceans 2006*, 2006, pp. 1–6.

[53] P. Ryan, "Mine countermeasures a success," in *United States Naval Institute Proceedings*, vol. 129, no. 5, May 2003, p. 52.

[54] E. Bovio, D. Cecchi, and F. Baralli, "Autonomous underwater vehicles for scientific and naval operations," *Annual Reviews in Control*, vol. 30, no. 2, pp. 117–130, 2006.

[55] A. Alvarez, A. Caffaz, A. Caiti, G. Casalino, L. Gualdesi, A. Turetta, and R. Viviani, "Folaga: a very low cost autonomous underwater vehicle combining glider and AUV capabilities," *Ocean Engineering*, vol. 36, no. 1, pp. 24–38, 2009.

[56] D. Horner, A. Healey, and S. Kragelund, "AUV experiments in obstacle avoidance," in *Oceans 2005*, vol. 2, 2005, pp. 1464–1470.

[57] A. Healey, "Guidance laws, obstacle avoidance, artificial potential functions," in *Advances in Unmanned Marine Vehicles*, ser. IEE Control Series, G. Roberts and R. Sutton, Eds. Institution of Engineering and Technology, March 2006, ch. 3, pp. 43–65.

[58] A. Caiti, G. Casalino, E. Lorenzi, A. Turetta, and R. Viviani, "Distributed adaptive environmental sampling with auvs: Cooperation and team coordination through minimum-spanning-tree graph searching algorithms," in *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV 2008)*, April 2008.

[59] A. Munafo, E. Simetti, A. Turetta, A. Caiti, and G. Casalino, "Autonomous underwater vehicle teams for adaptive ocean sampling: a data-driven approach," *Journal of Ocean Dynamics*, July 2011.

[60] A. Caiti, A. Garulli, F. Livide, and D. Prattichizzo, "Localization of autonomous underwater vehicles by floating acoustic buoys: a set membership theoretic approach," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 1, January 2005.

[61] V. Chandrasekhar, W. K. Seah, Y. S. Choo, and H. V. Ee, "Localization in underwater sensor networks - survey and challenges," in *International Workshop on UnderWater Networks (WUWNet 06)*, Los Angeles, CA, USA, September 2006.

[62] S. Lanzisera, D. Lin T., and K. S. J. Pister, "RF time of flight ranging for wireless sensor network localization," in *International Workshop on Intelligent Solutions in Embedded Systems*, June 2006.

[63] D. E. Manolakis, "Efficient solution and performance analysis of 3D position estimation via trilateration," *IEEE Trans. On Aerospace and Electronic Systems*, vol. 32, no. 4, 1996.

[64] N.-H. Kussat, C. D. Chadwell, and R. Zimmerman, "Absolute positioning of an autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 1, pp. 153–164, January 2005.

[65] F. B. Jensen, W. A. Kuperman, M. Porter, and H. Schmidt, *Computational Ocean Acoustics*. New York: Springer-Verlag, 2000.

[66] D. R. C. Philip, "An evaluation of USBL and SBL acoustic systems and the optimization of methods of calibration," *The Hydrographical Journal*, no. 108, pp. 18–25, April 2003.

[67] X. Xiang, G. Xu, Q. Zang, Y. Guo, and X. Huang, "A novel acoustic navigation scheme for coordinated heterogeneous autonomous vehicles," in *IEEE International Conference on Mechatronics and Automation*, Harbin, China, August 2007.

[68] Y. Bar-Shalom, X. Rong-Li, and T. Kirubarajan, *Estimation with Application to Tracking and Navigation*. John Wiley, 2001.

[69] B. Ristic, S. Arulampalm, and N. Gordon, *Beyond the Kalman Filter*. Artech House, 2004.

[70] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[71] J. Jaffe and C. Schurgers, "Sensor networks of freely drifting autonomous underwater explorers," in *International Workshop on UnderWater Networks (WUWNet 06)*, Los Angeles, CA, USA, September 2006.

[72] D. Mirza and C. Schurgers, "Motion-aware self-localization for underwater networks," in *International Workshop on UnderWater Networks (WUWNet 08)*, San Francisco, CA, USA, September 2008.

[73] F. Maurelli, S. Krupinsky, Y. Petillot, and J. Salvi, "A particle filter approach for AUV localization," in *IEEE Oceans 2008*, Quebec City, Quebec, Canada, 2008.

[74] H. Wang, S. Rock, and M. Lees, "Experiments in automatic retrieval of underwater objects with an AUV." in *MTS/IEEE Oceans '95*, vol. 1, October 1995, pp. 366–373.

[75] G. Casalino, D. Angeletti, G. Cannata, and G. Marani, "The functional and algorithmic design of Amadeus multirobot workcell," in *Underwater Vehicle Technology*, S. K. Choi and J. Yuh, Eds., 2002, vol. 12.

[76] J. Evans, P. Redmond, C. Plakas, K. Hamilton, and D. Lane, "Autonomous docking for Intervention-AUVs using sonar and video-based real-time 3D pose estimation," in *Oceans 2003*, vol. 4, September 2003, pp. 2201–2210.

[77] G. Marani, S. K. Choi, and J. Yuh, "Underwater autonomous manipulation for intervention missions AUVs," *Ocean Engineering*, vol. 36, pp. 15–23, 2008.

[78] G. De Novi, C. Melchiorri, J. Garcia, P. Sanz, P. Ridao, and G. Oliver, "New approach for a reconfigurable autonomous underwater vehicle for intervention," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 11, pp. 32–36, 2010.

[79] G. Casalino and A. Turetta, "Coordination and control of multiarm nonholonomic mobile manipulators," in *Advances in Control of Articulated and Mobile Robots*, B. Siciliano, A. De Luca, C. Melchiorri, and G. Casalino, Eds. STAR 10, 2004, pp. 171–190.

[80] E. Simetti, A. Turetta, and G. Casalino, "Distributed control and coordination of cooperative mobile manipulator systems," in *Distributed Autonomous Robotics Systems (DARS 2008)*, Tsukuba, Japan, November 2008.

[81] G. Casalino, E. Simetti, A. Turetta, and E. Zereik, "Distributed self-organizing control and coordination of cooperative mobile manipulator systems," in *Control Themes in Hyperflexible Robotic Workcells*, F. Basile and P. Chiacchio, Eds., July 2010, pp. 87–100.

[82] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," in *Preprints of the the 8-th IFAC World Congress*, Kyoto, Japan, 1981.

[83] A. A. Macjeiesky and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *International Journal of Robotic Research*, vol. 4, no. 5, pp. 109–117, 1985.

[84] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based control of robotic manipulators," *International Journal of Robotic Research*, vol. 6, no. 1, pp. 3–15, 1987.

[85] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization.* Addison Wesley, 1991.

[86] B. Siciliano and J. J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Highly Redundant Robotic Systems, Proc. of ICAR'91*, vol. 2, 1991, pp. 1211–1215.

[87] O. Kanoun, F. Lamiraux, and P. B. Wieber, "Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[88] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," in *Proceedings of the 2005 IEEE International Conference on Robotics & Automation*, Barcelona, Spain, 2005.

[89] G. Casalino and A. Turetta, "Coordination and control of multiarm, nonholonomic mobile manipulators," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 3, 2003, pp. 2203–2210.

[90] E. Zereik, A. Sorbara, A. Merlo, E. Simetti, G. Casalino, and F. Didot, "Space robotics supporting exploration missions: Vision, force control and coordination strategy for crew assistants," *Intelligent Service Robotics*, January 2011.

[91] R. Skjetne and T. Fossen, "Nonlinear maneuvering and control of ships," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 3, 2001, pp. 1808–1815 vol.3.

[92] G. Casalino, "Trident overall system modelling, including all variables needed for reactive coordination," ISME, Tech. Rep., 2011.

[93] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1986.