MODELING, CONTROL, AND OPTIMIZATION OF NETWORKED VEHICLE SYSTEMS

By
João Tasso de Figueiredo Borges de Sousa

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILISOPHY
AT
UNIVERSIDADE DO PORTO
PORTO, PORTUGAL
AUGUST 2014

UNIVERSIDADE DO PORTO
DEPARTMENT OF
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Modeling, Control, and Optimization of networked vehicle systems**" by **João Tasso de Figueiredo Borges de Sousa** in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: August 2014

Supervisor: ________________________________

Readers: ________________________________

____

UNIVERSIDADE DO PORTO

Date: **August 2014**

Author:	**João Tasso de Figueiredo Borges de Sousa**
Title:	**Modeling, Control, and Optimization of networked vehicle systems**
Department:	**Engenharia Electrotécnica e de Computadores**

Degree: **PhD** Convocation: **October** Year: **2014**

To Fátima and to my family.

Contents

List of Figures

List of Tables

Abstract

This thesis concerns the development of a model and of a control framework for networked vehicle systems composed of physical and computational entities with coupled dynamics. This is done in the framework of dynamic optimization and of models of systems with evolving structure.

The model encompasses physical and computational dynamics coupled through physical interactions and communications. The physical entities and link layers establish computational environments within which computational entities evolve and interact. These provide the physical layer which will embody complex organizations, such as the ones envisaged for future generations of networked vehicle systems.

A control and computational framework for organizations of networked vehicles systems is proposed and a systematic design methodology within which properties of the organization can be proved is introduced. The framework encompasses a control and computation architecture and a design methodology. The architecture derives from a few design principles and is implemented with the help of a few mechanisms. The design methodology introduces a compositional layered approach to control and computation that is uniform for all vehicles.

The compositional layered approach allows the use of assume guaranteed reasoning techniques to prove properties of the system. Lower level properties are studied in the framework of reach set computations. This allows to check the feasibility of planned motions and provides a framework to derive motion controllers with guaranteed results. Higher level properties are then proved under the assumptions of guaranteed low level behavior. This is done in the framework of simulation and bi-simulation relations relative to formal specifications.

The problem of optimizing the behavior of networked vehicle systems is addressed in the framework of dynamic optimization. This allows the consideration of optimization problems for systems with non-trivial state-space and control spaces. The results are specialized to the problem of iterated multi-vehicle rendezvous. This is because the structure of these problems allows a structured application of the principle of optimality that results in coordinated optimization problems formulated in lower-dimensional spaces. Structure induces the composition of value functions in these lower-dimensional spaces. However, the structure of some problems, which is derived from the associated cost functions, may preclude the direct application of the principle of optimality. This problem is also addressed in this work.

Acknowledgements

“No man is an island, entire of itself; every man is a piece of the continent, a part of the main.”

— John Donne, Meditation XVII, 1624.

Donne’s thoughts, stressing the importance of how man thrives in the company of other human beings, are particularly useful to understand how complex networks are shaping today’s interconnected world. This is especially true in academia.

The academic world is an evolving network of people, institutions, concepts, theories, methods, and techniques, to name just a few. Access to the observables of the network, namely published research, has been made easy by the Internet. But a publication is just the outcome of a process that typically depends on human interactions; research thrives, in part, due to human interaction and relationships. This is why geography and institutions are still a major factor in the excellence of research and in the drive for innovation.

This work culminates a process, a journey of network development. It is about concepts and theories for systems of coupled computational and physical entities. But is also about how these concepts and theories evolved over time in a network fashion. I had the good fortune of interacting with a number of very influential people in Europe and in the United States of America. These interactions shaped my research, and also my approach to life, in very fundamental ways.

First, I would like to express deep gratitude to Professor Fernando Pereira. He introduce me to the elegance and power of advanced control methods, and to the world of research without geographical boundaries. This was at a time when the sources of knowledge were still geographically and psychologically remote. He created numerous opportunities for me to grow as a person, and as a researcher, within a growing network of international collaborations. He is also a good friend. He is the reason why I am here. I also thank Professor Jorge Martins de Carvalho for establishing, in cooperation with Professor Lobo Pereira, a good set of initial conditions for controls research at Porto University.

I had the privilege of working with Professor Pravin Varaiya from the University of California at Berkeley. This was a life changing experience. I am forever grateful for his example, support, and patience. His quiet leadership and intelligence are a source of inspiration to me. His views on control, communication, and computation helped shaping my own views in some fundamental ways. His network of contacts help me to overcome geographical and institutional limitations. Berkeley is an amazing place and a hub in the academic world.

I am forever grateful to Professor Alexander Kurzhanskii from Moscow State University and from the University of California at Berkeley. Professor Kurzhanskii introduced me to the insightful and original control developments from the Ekaterinburg school. Through him I learned a lot about the fundamental work of his advisor Professor Krasovskii on differential games and control. His unified approach to control and optimization has shaped my research. In him I have a good friend.

I thank Professor Healey for his example, for his support, and for the elegance of his professional and personal advice. Another good friend.

I thank the current and former members of the Laboratório de Sistemas e Tecnologias Subaquáticas (LSTS). They had the motivation, the drive, and the persistence to help creating a world class laboratory. This has been an amazing journey. Today I am proud to say that the laboratory has unique tools and vehicles to demonstrate the concepts developed in this work. I thank Jorge Estrela for discussions and contributions addressing numerical methods for solving the Hamilton-Jacobi-Bellman-Isaacs partial differential equation and for the development of feedback controllers in this framework. I thank José Pinto and Paulo Dias for developing, with the help of many others, the *NEPTUS* command and control framework for multi-vehicle systems. I thank Ricardo Martins for his leading role in the development and implementation of the LSTS *DUNE-IMC-NEPTUS* software tool chain. Ricardo is an example of quiet excellence and loyalty. I thank Eduardo Marques for discussions on languages for implementing controllers for networked vehicle systems, and for brainstorming on computation and control. I thank José Pinto for discussions on models and software implementations for multi-vehicle systems, with special emphasis on the fundamentally new control approach enabled by concepts of code migration. Special thanks to Paulo Lopes for his loyal support. I am proud of what we all achieved together.

I thankfully acknowledge the support of Porto University. This was made possible through the friendly support of Professors José Carlos Marques dos Santos, Sebastião Feyo de Azevedo, José Silva Matos, António Torres Marques, Nunes Ferreira, Jorge Gonçalves, and Pedro Guedes de Oliveira.

Special thanks go to Richard Morgan for his friendship, for broadening my perspectives on the social sciences, for trying to understand my motivations, and for helping to make me a better person. André Puga has not been forgotten. Vivid recollections of his friendship will last forever.

Finally, I cannot find words to express my gratitude to Fátima, and to family, for putting up with my absences, even when I was in Porto. I am fortunate to work on what I like. But this is not a way a life, just part of it. I thank my family for making this possible.

Notation and definitions

Here, we adopt the notation from [39]

- X – real Hilbert space or Banach space with norm $\| \, . \, \|$.
- B – open ball of radius 1 centered at 0. $\overline{B}$ is its closure.
- $\langle \xi, x \rangle$ – inner product of x and ξ when X is a Hilbert space or the evaluation at $x \in X$ of the linear functional $\xi \in X^*$, the space of linear continuous functionals defined in X, when X is a Banach space.
- w^* – $weak^*$ topology on the space X^*
- B_* – open unit ball in X^*
- $x =$ w-lim$_{i \to \infty} x_i$ signifies that the sequence $\{x_i\}$ converges weakly to x.
- Consider a set S. Then:
  - int S – interior of S.
  - cl S – closure of S.
  - bdry S – boundary of S.
  - co S – convex hull.
  - $\overline{co}$ S – closed convex hull of S.
- $\mathcal{F}$(U), where $U \subset X$ is open, designates the class of all functions $f : X \to (-\infty, \infty]$ which are lower semicontinuous on U and such that dom $f \cap U \neq \emptyset$.
- The graph and the epigraph of a function f $\in \mathcal{F}$(X) are given respectively by:
  - $grf := \{(x, f(x)) : x \in \text{dom} f\}$

  - $epif := \{(x,r) \in \text{dom}f \times \Re : r \geq f(x)\}$

- The indicator function of a set S, denoted by $I_S(.)$ or $I(S,.)$ is the extended-valued function defined by:

$$I_S(x) = \begin{cases} 0 & : \quad \text{if x} \in S, \\ +\infty & : \quad \text{otherwise} \end{cases}$$

We need the following definitions.

Definition 0.0.1 (Effective domain of a function)**.** Unless otherwise stated, we will be considering extended real valued functions of the class $\mathcal{F}$. To single out those points at which f is not $+\infty$, the effective domain of f is defined as the set:

$$domf := \{x \in X : f(x) < +\infty\}$$

Definition 0.0.2 (Positive homogeneous function)**.** A function g is positive homogeneous if:

$$g(\lambda v) = \lambda g(v) \text{for} \lambda \geq 0$$

Definition 0.0.3 (Subadditive function)**.** A function g is subadditive if:

$$g(w+v) = g(v) + g(w)$$

Definition 0.0.4 (Upper semicontinuous multifunction)**.** Consider the multi-function $F : \Re^n \to \Re^n$. The multivalued function F is upper semicontinuous at x if:

$$\forall \epsilon > 0, \exists \delta > 0 : \|x - y\| < \delta \Rightarrow F(y) \subset F(x) + \epsilon B$$

Definition 0.0.5 (Lower semicontinuous multifunction)**.** Consider the multi-function $F : \Re^n \to \Re^n$. The multivalued function F is lower semicontinuous at x if:

$$\forall v \in F(x) \quad \text{and} \quad \epsilon > 0, \exists \delta > 0 : x' \in \text{dom F}, x' \in x + \epsilon B \Rightarrow v \in F(x') + \epsilon B$$

Definition 0.0.6 (Norm in X^*)**.** $\|\xi\|_*$ denotes the norm in X^*

$$\|\xi\|_* := sup\{\langle \xi, v \rangle : v \in X, \|v\| = 1\}$$

Definition 0.0.7 (Lipschitz condition of rank K)**.** A function $f \in \mathcal{F}$ is said to satisfy a Lipschitz condition of rank K on a given set S if it is finite on S and satisfies:

$$|f(x) - f(y)| \leq K\|x - y\|, \forall x, y \in S$$

Definition 0.0.8 (Locally Lipschitz)**.** F is said to be locally Lipschitz provided that every point x admits a neighborhood $U = U(x)$ and a positive constant $K = K(x)$ such that:

$$\forall x_1, x_2 \in U \Rightarrow F(x_2) \subseteq F(x_1) + K \parallel x_1 - x_2 \parallel \overline{B} \tag{0.0.1}$$

Then F is said of rank K on the set U.

List of symbols

ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
CMRE	Centre for Maritime Research and Experimentation
CTD	Conductivity, Temperature and Depth
DOD	Department of Defense
DTN	Delay-Tolerant Networking
EDA	European Defense Agency
EEZ	Exclusive Economic Zone
GPS	Global Positioning System
GSM	Global System for Mobile communications
IMU	Inertial Motion Unit
JAUS	Joint Architecture for Unmanned Systems
JCGUAV	NATO Joint Capability Group UAV
LAUV	Light Autonomous Underwater Vehicle from LSTS
LSTS	Laboratório de Sistemas e Tecnologias Subaquáticos (in Portuguese)
	Underwater Systems and Technologies Laboratory from Porto University
LBL	Long BaseLine
LCM	Life Cycle Management
MTOW	Maximum Take-Off Weight
NATO	North Atlantic Treaty Organization
NRP	Navio da República Portuguesa
OCS	Outer Continental Shelf
ROV	Remotely Operated Vehicle
RPA	Remotely-Piloted Aircraft
RPV	Remotely Piloted Vehicle
SATCOM	Satellite Communication
SEAD	Suppression of Enemy Air Defenses
SLAM	Simultaneous Localization and Mapping
STANAG	Standardization Agreement (North Atlantic Treaty Organization (NATO))
UAV	Unmanned Air Vehicle (obsolete term)
UAS	Unmanned Aircraft System
USA	United States of America
UUV	Unmanned Underwater Vehicle
UXS	Unmanned Air/Surface/Ground/Underwater Vehicle System

Table 1: List of symbols

Glossary

A glossary of relevant terms is presented next.

Airworthiness. The basic requirement for any aircraft system, manned or unmanned, to enter the National Air Space.

Automated vehicles. Vehicles in which there is no on-board deliberation – "Automatic means that a system will do exactly as programmed, it has no choice" (from [18]).

Autonomous vehicles. Vehicles in which deliberation takes place on-board without human intervention – "Autonomous means that a system has a choice to make free of outside influence"(from [18]).

Capability. The ability to achieve a desired effect under specified standards and conditions through combinations of ways and means to perform a set of tasks [49].

Life Cycle Management (LCM). A management process, applied throughout the life of a system, that bases all programmatic decisions on the anticipated mission-related and economic benefits derived over the life of the system [49].

Mission. The objective or task, together with the purpose, which clearly indicates the action to be taken [49].

Mixed initiative planning and execution control. Computational planning and execution control procedures allowing intervention by experienced human operators.

Process. The combination of people, equipment, materials, methods, and environment that produces output - a given product or service [49].

System. The organization of hardware, software, material, facilities, personnel, data, and services needed to perform a designated function with specified results, such as the gathering of specified data, its processing, and delivery to users [49].

Subsystem. A functional grouping of components that combine to perform a major function within an element such as electrical power, attitude control, and propulsion [49].

Chapter 1

Introduction

"It is in beauty that the scientist and the engineer differ: for the scientist it is nature's beauty, for the engineer it is the beauty of creation."

— Louis Brown, Technical and Military Imperatives: A Radar History of World War, Taylor and Francis, 1997.

1.1 Scope and goals

Networks of systems are pervasive, their behaviors can be highly complex, and network effects are now being perceived as something fundamental to understand what is going around us. This is the reason why networks are receiving a significant attention in physics, biology, climatology, and social , to name just a few. In spite of this diversity, some patters of network behavior seem to be pervasive across these domains, as we can infer from the literature in the field [5]. Graph theory and statistical analysis have been used heavily in the literature, and the focus has been more on descriptive methods.

Network behavior is also receiving the attention of the robotics and, consequently, in control, computation, and communications. Swarm robotics is concerned with the problem of developing group behaviors from simple behaviors; the tools and techniques resemble some of the ones from physics and biology. Soccer robotics, and other types of robotic games, address problems of team control and coordination that go beyond swarm behavior. This is done with distributed algorithms and coordination. Formation flying and control is also receiving significant attention from the controls community. The study of feedback laws and of their dependency on communication topologies and sensing capabilities is underway and uses techniques from graph theory and stability theory. These techniques are also being used to study formation control in nature. Problems of controlling systems over networks are also receiving significant attention from both the control and communication communities. Estimation and control, along with event-based controllers, have been successfully used to tackle these problems.

However, there are multiple aspects of network behavior in multi-vehicle systems which have not been addressed in the literature. These aspects arise from the nature of network vehicle systems: information and commands are exchanged among multiple vehicles, and the roles, relative positions, and dependencies of these vehicles and systems change during operations. In addition, these systems may exhibit properties that are a function of structure, where structure arises from interactions established over physical, sensing and communication links

These aspects of the behavior of network vehicle systems pose new challenges to computation and control: from prescribing and controlling the behavior of isolated systems to prescribing and controlling the behavior of interacting systems. The is was first noticed in computer science by Robin Milner [76, 74, 75].

This thesis is about the development of a model and of a control framework for networked vehicle systems composed of physical and computational entities with coupled dynamics. The goal is to develop a framework within which we can analyze and design these systems. The ultimate goal is the deployment of this control framework in a new generation of networked vehicle systems under development at Porto University for novel scientific and military applications.

1.2 Why is it interesting

Autonomous and/or operator assisted networked multi-vehicle systems are on the rise, especially in new applications, such as oceanographic or atmospheric studies, or in new developments for more traditional applications, such as automated automotive systems. However, and in spite of this trend, the theory of networked vehicle systems is lagging behind the applications.

The control and computer science communities are addressing this challenge in the context of distributed systems, and contribute complementary views and techniques. The inter-disciplinary nature of networked vehicle systems requires a new description language. This language is in the process of being developed. Over the years control engineers have developed a collection of idioms, patterns and styles of organization that serves as a shared, semantically rich, vocabulary among them. However, this shared vocabulary is still deeply rooted in the underlying mathematical framework – differential equations – and lacks some semantically rich concepts invoked by distributed computing. The cause may be that experience and functionality in computing are acquired at a rate unmatched by the rate of evolution of concepts in control systems. For example, it was only recently that the expressiveness of the language of differential equations and dynamic optimization was enlarged with concepts from mathematical logic, under the denomination of hybrid control (see for example [81], [68], [11], [37]).

Today, we are still far from being able to design and deploy networked vehicle systems in a systematic manner and within an appropriate scientific framework. To achieve this goal we need a clear understanding of the fundamental challenges, which have to be abstracted away from a wide range of applications. This requires: 1) some familiarity with both applications and concepts in related fields, namely control theory and computer science; and 2) a framework within which we can structure our ideas.

While practitioners are focussed on the engineering systems and on operations, theoreticians may lack the in-depth understanding of the problem domain required to identify the fundamental questions in the field. In addition, there is no readily available mathematical tool set to address this problem domain.

This thesis builds on experience in the design, construction, and deployment of networked vehicles. Applications are as diverse as automated highway systems [102, 26], Mobile Offshore Base [83, 44], mixed initiative control of automata teams [34], coordinated air and ocean vehicles for marine biology and oceanography, plume tracking, and security and defense [71]. Tools and technologies included hybrid systems, dynamic optimization, models of computation, non-linear control, computer-aided verification, and, more recently, deliberative planning. This experience was also journey in networked development, in which I have been fortunate to interact with experts in all of these fields, each one with his own views and perspectives, contributed to bridge this gap between application and conceptual development. In addition, it contributed to the development of the approach presented in this work.

1.3 Approach

This is an inherently inter-disciplinary work at the intersection of control and computation.

The modeling problem is address in the framework of computational and physical models. This is done in a uniform manner to preserve notions of state and control spaces, as well as of system dynamics, which are extended to accommodate the novel aspects of network behavior and coupled dynamics. Models expressing the benefits of composition and the emergence of capabilities arising from interactions are handled in the framework of a layered control and computation architecture.

The control and computation problem is addressed in the framework of a layered control and computation architecture, and associated design methodology. The layered architecture introduces specialized controllers and controller dependencies within which organizations of networked vehicles are formed and controlled. This with done with the support of a few mechanism of access control embedded in hierarchies of physical and communication channels. The controller structures are design for structure control, but also to support organizations of networked vehicles in delivering unprecedented behaviors.

The architectural design is targeted at the formal verification of properties. This will be done in the framework of assume guaranteed techniques, to abstract low level motion behaviors, and automata theory, to analyze bi-simulation relations between implementations and formal specifications.

The planning and execution control problems will be addressed in the framework of dynamic optimization. This is because dynamic optimization, in spite of the associated computational complexity, provides a uniform framework for handling problems of reachability, invariance, and optimization formulated in extended state and control spaces. The structure of a significant number of problems in networked vehicle systems can be formulated as iterated rendezvous problems. This is because of operations in communications-challenged environments. The application of the principle optimality to iterated rendezvous problems induces an organization of coupled problems in lower-dimensional spaces. This significantly reducing the computational complexity associated to the dynamic programming approach.

1.4 Developments at the Laboratório de Sistemas e Tecnologias Subaquáticas

1.4.1 Laboratory overview

The Laboratório de Sistemas e Tecnologias Subaquáticas (LSTS) at the Faculty of Engineering from Porto University has been designing, building and operating unmanned underwater, surface and air vehicle systems for innovative applications with strong societal impact since it was established in 1997. Currently the LSTS team has over 30 researchers, including faculty and students, with Electrical and Computer Engineering, Mechanical Engineering and Computer Science backgrounds.

In 2006 the LSTS received the national BES Innovation National Award for the design of the Light AUV (LAUV). This vehicle is now in its 5th generation. The LSTS delivered three LAUV units to the

Portuguese Navy and has transitioned the technology to the spin-off company OceanScan Marine Systems and Technology.

The LSTS fleet includes two remotely operated submarines (rated for 200m), two autonomous underwater vehicles (AUV) of the Isurus class, six AUVs of the Light Autonomous Underwater Vehicle (LAUV) class, one autonomous surface vehicle (Swordfish), and twelve autonomous air vehicles (wingspans ranging from 1.8m to 3.6m).

LSTS uses the IEEE P1220-standard [52] for the systems engineering process in the design, development, and operation of several generations of unmanned air, surface, and air vehicles for defence and civil applications. Vehicles in the LSTS fleet are engineered for networked operations and use modular hardware and software components to facilitate development, maintenance, and operations. The LSTS open source software tool chain for networked vehicle systems is now in use in the United States, Germany, Switzerland, Norway, France, England, Spain, and India. This tool chain allows the operators at the LSTS control stations to command and control all types of vehicles in a uniform manner, with special support for sense and avoid. The tool chain supports onboard autonomy through the integration of the deliberative on-board planning framework *TREX* (developed at the Monterey Bay Aquarium Research Institute).

LSTS has successfully fielded unmanned air, ground, surface and underwater vehicles in innovative operations in Europe and in the United States of America. These include some world firsts, such as the underwater rendezvous between the Aries and Isurus AUVs, respectively from the Naval Postgraduate School and Porto University, which took place in 2006 in Monterey, California, under a cooperation project between the two institutions.

LSTS has been involved in fostering and growing a world-wide research network in the area of networked vehicle systems with yearly conferences and workshops, and, more recently, with large scale exercises at sea. LSTS researchers are well connected (with frequent visitors, seminar exchanges, and tools exchanges) to research efforts as collaborators (on other projects and developments) at MBARI (USA), Naval Postgraduate School (USA), Naval Undersea Warfare Center (USA), University of Michigan (USA), US Coast Guard (US), Naval Undersea Research Center (NATO), National Oceanography Center (UK), University of Limerick (IRL), Norwegian University of Science and Technology (NOR), Royal Institute of Technology (SWE), University of Delft (NL), and Swiss Federal Institute of Technology Zurich (CH). In Portugal the LSTS has a strategic cooperation with the Portuguese Navy (LSTS provided technical support to their unmanned underwater vehicles program, delivered three units of a mine warfare/REA version of the Light Autonomous Underwater Vehicle, and is evolving these vehicles for advanced operations with manned vessels under projects funded by the Portuguese Ministry of Defense) and Air Force (jointly developing the unmanned air vehicles program with funding provided by the Portuguese Ministry of Defense), with the Portuguese Task Group for the Extension of the Continental Shelf (systems and technologies for underwater exploration and mapping and in operation of the deep sea ROV Luso), and with the Porto Harbor (systems and technologies for harbor operations). The LSTS has been organizing, in cooperation with the Portuguese Navy, the Rapid Environmental Picture (REP) annual exercise since 2010. In addition, researchers from LSTS also participated in experiments, organized and hosted by collaborators, and taking place in the Pacific and Atlantic oceans, as well as in the Mediterranean and Adriatic seas. In the next 5 years LSTS will host tests and demonstrations of projects funded by the EU, EDA and NATO, thus making the Porto a hub for international collaboration in this field.

1.4.2 Software tool chain

The LSTS control architecture has an off-board component and an on-board component. These are implemented with the help of the LSTS *NEPTUS-IMC-DUNE* software tool chain. This is a software framework for mixed-initiative control (humans in the planning and control loops) of unmanned ocean and air vehicles operating in communications challenged environments with support for Disruptive Tolerant Networking (DTN) protocols. These unique features of the tool chain build on experience with the coordinated operation of heterogeneous vehicles. *NEPTUS* is a distributed command, control, communications and intelligence framework for operations with networked vehicles, systems, and human operators. *NEPTUS* supports all the phases of a mission life cycle: world representation; planning; simulation; and, execution and post-mission analysis [78]. *IMC* is a communications protocol that defines a common control message set understood by all types of LSTS nodes (vehicles, consoles or sensors) in networked environments [69]. This provides for standard coupling of heterogeneous components in terms of data interchange. *DUNE* is the system for vehicle on-board software. It is used to write generic embedded software at the heart of the vehicle, e.g. code for control, navigation, or to access sensors and actuators. It provides an operating-system and architecture independent C++ programming environment for writing efficient real-time reactive tasks in modular fashion.

The tool chain has support for DTN protocols [72]. Currently LSTS researchers are working with the Monterey Bay Research Institute on the integration of the deliberative onboard planning system *TREX* [82] with *DUNE*.

1.4.3 Operational experience

LSTS extensive operational experience in large scale exercises has been providing invaluable lessons on the deployment of networked vehicle systems. A list of selected exercises follows:

REP14–Atlantic, fifth edition of the Rapid Picture (REP) Atlantic exercise. REP14-Atlantic was a joint exercise of the Portuguese Navy, the NATO Centre for Maritime Research and Experimentation (CMRE), and the University of Porto. REP14-Atlantic was a large experiment conducted in the Lisbon Naval Base and off the coasts of Sesimbra and Sines, in Portugal. It included several various Autonomous Surface Vessels (ASVs), Autonomous Underwater Vehicles (AUVs) and Unmanned Aerial Vehicles (UAVs) equipped with different sensors and acoustic payloads, which were deployed from Portuguese Navy ships NRP Pegaso, NRP Auriga, submarine NRP Arpão and NATO Research Vessel Alliance. REP14-Atlantic aimed to test networks of unmanned maritime vehicles in mine countermeasures, maritime security, environmental knowledge, search and rescue, and maritime law enforcement operational scenarios. The participants included the Monterey Bay Research Institute (MBARI-USA), the University of Rome (Italy), the Norwegian University of Science and Technology (Norway), the Royal Institute of Technology, the Naval Undersea Warfare Center, and the companies Evologics (Germany) and Oceanscan (Portugal). Through REP14-Atlantic the participants demonstrated collaborative research to increase interoperability, underwater communications and disruption/delay tolerant networking (DTN) capabilities, automation and cooperation of not only unmanned underwater, but also surface and aerial vehicles. In addition some of these vehicles had deliberative planning capabilities on-board for unprecedented levels of autonomy. Deliberative planning techniques were used to support coordinated planning and execution control of multiple vehicles.

CANON experiment, Pacific Ocean, organized by the Monterey Bay Aquarium Research Institute (MBARI), 2013. LSTS participated in this exercise with one LAUV and technology for coordination with the Dorado AUV, the TEX Wave Glider and ships from MBARI for oceanographic data collection (http://www.mbari.org/canon/).

Breaking the surface experiment, Adriatic Sea, 2013. Demonstration of Unmanned Air Vehicle systems for maritime applications (http://bts.fer.hr/).

Fourth edition of the Rapid Environmental Picture (REP13) exercise organized in cooperation with the Portuguese Navy, Portugal, 2013 ((http://rep13.lsts.pt/)). The exercise took place off the coast of Sesimbra, Portugal, but continued further south from the Portimão airfield. It involved participants from MBARI (USA), Evologics (DE), and the Norwegian University of Science and Technology (NOR). Several large and small propeller-driven ASVs, AUVs with different sensors and acoustic modems were deployed from Bacamarte, a ship from the Portuguese Navy. Several PITVANT UAS were used in these experiments, some being deployed and recovered from civilian airports under monitoring of the Portuguese Air Force, and others launched and recovered aboard Bacamarte. The exercise was targeted at applications in Mine Warfare, Harbour Protection, Expeditionary Hidrography, Search and Rescue, Maritime Law Enforcement, and Rapid Environmental Assessment.

Demonstration of networked air and ocean vehicles in the maritime incident response Cathach, Ireland, 2013. This was done in the context of the Interreg Netmar project (http://www.shannonresponse.com/).

Third edition of the Rapid Environmental Picture (REP12) exercise organized in cooperation with the Portuguese Navy, Portugal, 2012. The REP-12 exercise took place off the coast of Sesimbra, Portugal, but continued further north from the Santa Cruz airfield. It involved participants from MBARI (USA), Centre for Maritime Research and Experimentation (NATO), Evologics (DE), Technion (IL), Norwegian University of Science and Technology (NOR) and University of Rome (IT). Several large and small propeller-driven ASVs, AUVs with different sensors and acoustic modems, as well as the wave-propelled Wave-Glider ASV (from Liquid Robotics, Inc.) were deployed from Bacamarte, a ship from the Portuguese Navy. Several PITVANT UAS were used in these experiments, some being deployed and recovered from civilian airports under monitoring of the Portuguese Air Force, and others launched and recovered aboard Bacamarte.

Breaking the surface experiment, Adriatic Sea, 2012 (http://bts.fer.hr/). Archeology and sea-bottom mapping with the SEACON AUV, Murter, Croacia.

Acommsnet 2012 experiment organized by NURC-NATO, Mediterranean Sea, 2012. Disruptive Tolerant Networking (DTN) experiments with multiple Autonomous Underwater Vehicles, 2012.

C4C - Control for Coordination FP7 project final demonstration (Porto, Portugal), 2011. Demonstration of the coordinated operation of multiple Autonomous Underwater and Surface Vehicles for plume mapping and formation control over acoustic modems.

Deep Divex 2011 NATO exercise, Halifax, Canada, 2011. Invited participation with SEACON AUVs for mine sweeping in a harbor environment.

Mar Menor Robotics experiment, Mar Menor, Spain, 2011. Demonstration of salinity plume mapping capabilities with multiple Autonomous Underwater Vehicles.

Second edition of the Rapid Environmental Picture (REP11) exercise organized in cooperation with the Portuguese Navy, Portugal, 2011. The REP11 exercise took place off the coast of Sesimbra. The REP-11 exercise was focused on the demonstration of the SEACON AUV and of DTN, including the transfer of sonar files from a SEACON AUV to a small Unmanned Air System (UAS) deployed from a ship from the Portuguese Navy. The exercise also included deployments of the SEACON AUV from manned submarines from the Portuguese Navy.

First edition of the Rapid Environmental Picture (REP10) exercise organized in cooperation with the Portuguese Navy, Portugal, 2010. The participants in the REP-10 AUV experiment included, in addition to the Portuguese Navy and LSTS, the following: Naval Undersea Warfare Center (Newport, USA), SeeByte (Edinburgh, United Kingdom), OceanScan MST (Porto, Portugal), OceanServer Technology (Fall River, Massachusetts) and YSI (Yellow Springs, Ohio). The experiment was targeted at assessing the endurance and performance of the SEACON AUV (developed by LSTS), evaluating and testing the coordinated operation of multiple AUVS from the Portuguese Navy (Gavias from Teledyne Gavia), NUWC (Iver2 from OceanServer) and LSTS (SEACON), extending the communication range of autonomous vehicles with fixed and mobile gateways, validating remotely sensed data, and testing ship and shore launching and recovery of AUVs.

LUSO ROV deep sea operations, Atlantic Ocean, 2009-2010. This was done in cooperation with the Portuguese Task Group for the Extension of the Continental Shelf and targeted data and sample collection from the sea floor.

Acommsnet 2010 experiment organized by NURC-NATO, Mediterranean Sea, 2010. DTN experiments (2010).

Spain Minex 2010, international exercise organized by the Spanish Navy, Mediterranean Sea, 2010. Mine sweeping deployments with Gavia AUVs from the PO Navy.

Rendezvous between two AUVs, Aries from the Naval Postgraduate School and Isurus from LSTS, Monterey, Pacific Ocean, 2006. Underwater rendezvous between the two AUVs using acoustic communications.

1.5 Outline of the thesis

Chapter 2 discussion the computation and control challenges arising in the coordination of multi-vehicle systems. This is done in the framework of coupled physical and computational dynamics governed by the laws of physical and computation. Several examples illustrate this point, that seems to be missing in the literature. The state space and dynamics for these systems are extended to show how these challenges can be formulated as classical control problems of optimization, invariance, and attainability. Directions for future research are discussed with special emphasis on the aspects of coupled dynamics and dynamic structure. Applications to other fields, such as biology, are briefly discussed. [33] is an abridged version of this chapter.

Chapter 3 introduces background in dynamic optimization that will be required in subsequent chapters. First, a derivation of the Hamilton-Jacobi-Bellman equation for a simplified version of the minimum time optimal problem is introduced to provide the background against which more refined derivations are discussed. This is done with the introduction of a value function for this problem. The derivation is based on assumptions of smooth behavior. The problem is that the general minimum time optimal control problem exhibits non-smooth behavior. This is why the reminder of the chapter is discussed in the framework of non-smooth analysis techniques described in an appendix. Several concepts of solution of the Hamilton-Jacobi-Equation are also introduced and briefly compared. This mathematical tool set is not specific to time optimal control problems. It provides a unifying framework to handle generic optimal control problems. Moreover, invariance and attainability problems can also be handled in this framework. The framework revolves around the notion of the value function associated to each specific optimization problem. The value function is the solution, in some appropriate sense, of the Hamilton-Jacobi-Equation. This is why the notion of viscosity solution of the Hamilton-Jacobi equation is also discussed in this chapter. Finally, the problem of deriving feedback controllers from the value function is also discussed. This is not a trivial matter because optimal feedback controllers tend to be discontinuous, another manifestation of non-smooth behavior.

Chapter 4 presents a formulation and a solution approach to the problem of optimal coordination of unmanned air vehicles for the Suppression of Enemy Air Defenses (SEAD). The problem consists in designing the attack of the Blue force of unmanned air combat vehicles against Red's ground force of SAM sites and radars. The design is structured in a two-level hierarchy of planning and execution. The plan, based on prior information, determines which targets are to be attacked and groups them into sub-tasks; allocates a UAV team to each sub-task; and selects a risk-minimizing path for each team. The planning procedure uses dynamic optimization techniques in two ways. First, to find risk-minimizing paths. Second, for optimal target selection. This is because the risk function for the UAVs depends on the state of the ground force of SAM sites which, in turn, can be affected by the actions of the Blue force. The planning procedure produces constraints for the attack of the Blue force in the form of a partial order in which targets should be engaged. Execution is organized in a hierarchy of real-time controllers, which determine the actual optimal flight paths, weapons release, and space-time coordination of the actions of a UAV team. The hierarchy embodies a system with evolving structure because it consists of interacting controllers which may change during operations: UAVs are assigned to specialized controllers in the hierarchy; assignments are instantiated through command links to controllers; and, assignments change in a feedback manner for adaptation. A control implementation was specified in *SHIFT*, a programming language for dynamic networks of hybrid automata. [34] is an abridged version of this chapter.

Chapter 5 introduces new optimal path coordination coordination for multiple vehicles and discusses how to formulate and solve them in the framework of dynamic optimization. An optimal path coordination for a two-vehicle system is considered to illustrate the approach and to discuss the novelties. The basic formulation is inspired by the developments presented in Chapter 4. The novelty of these problems arises in several ways. The cost function and the dynamics include non-trivial dependencies, modeled through existential quantification over groups of vehicles – this leads to non-Lipschitz behavior and to non-standard optimal control problems. There are consumable resources, modeled with the help of integral constraints – the structure of the constraints suggested new strategies for optimal cooperation which outperform the results obtained with standard formulations with state-constraints. The formulation presented in this chapter uses the structure of the problem to decouple the overall optimization into simpler coupled problems in lower-dimensional spaces. The structure of the coupling is encoded in a hybrid automaton. The principle of optimality is applied to this structure. This leads to coupled dynamic programming problems, which are solved sequentially to respect the principle of optimality. This is done with the help of numerical methods for solving the corresponding Hamilton-Jacobi-Bellman equations. The solution is encoded as the composition of value functions in lower-dimensional spaces. [25] is an abridged version of this chapter.

Chapter 6 discusses how to design a verified multi-vehicle control architecture to satisfy a formal specification. The developments are discussed with the help of an example in which a set of vehicles implements an optimization algorithm to search for the minimum of a scalar field. The layered control architecture for executing multi-vehicle team coordination algorithms is presented along with the formal specifications for team behavior. The specification consists of a two state model. The multi-vehicle system alternates between these states termed respectively *Communication* and *Motion*. The vehicles exchange messages to select the next sampling points in the *Communication* state. The vehicles move to the next sampling points in the *Motion* state. The transition from *Motion* to *Communication* takes place when all the vehicles reach the designated waypoints within a designed time interval. The transition from *Coordination* to *Motion* takes place when the vehicles reach a consensus on the next sampling points. The control architecture has three layers: team control, vehicle supervision and maneuver control. The implementation is proved to satisfy the specification for the team behavior. This is done in the framework of automata theory. The implementation and the specification are proved to be bi-similar. The proof uses assume guaranteed reasoning techniques. The assumptions are that the generated waypoints are reachable within a given time interval and that communication among the vehicles is feasible when the waypoints are reached. Reach set computation techniques are used to assist the planning procedure to ensure that the two assumptions hold. Computer simulations with accurate models of autonomous underwater vehicles illustrate the overall approach in the coordinated search for the minimum of a scalar field. The coordinated search is based on the simplex optimization algorithm. [28] is an abridged version of this chapter.

Chapter 7 presents the conclusions. These are organized in terms of what has been accomplished and directions for future research. The discussion of what has been accomplished presents a unified view of the developments. The directions for future research are organized in several lines of work: modeling frameworks, dynamic optimization, control architectures, and software frameworks. Some of these developments are already underway at LSTS. The concluding remarks discuss how the generic framework developed in this work could potentially contribute to new insights in other fields such as biology, ecology, and social sciences.

Appendix A briefly describes the unmanned air and ocean going vehicle systems designed and built at the Laboratório de Sistemas e Tecnologias Subaquáticas from Porto University.

Appendix B presents background information on non-smooth analysis. The focus is on the nature and geometry of non-smooth behavior. Two frameworks for non-smooth analysis are discussed. The first one, proximal analysis, has an intuitive geometric interpretation. The second one, also has a geometric interpretation. The constructs from the two frameworks are also briefly discussed. The two frameworks provide the mathematical tool set required to handle non-smooth behavior arising in dynamic optimization problems discussed in this thesis.

Appendix C describes a generic 6 degrees of freedom model of an autonomous underwater vehicle and discusses simplifications of the model for a torpedo shaped vehicle. The model is further simplified for planar motions. The approximation of the vehicle model by a kinematic model of a unicycle is also discussed for this class of motions.

1.6 Contributions

The main contribution of this dissertation is the development of a model and of a control framework for networked vehicle systems composed of physical and computational entities with coupled dynamics.

The model encompasses physical and computational dynamics coupled through physical interactions and communications. The physical entities and link layers establish computational environments within which computational entities evolve and interact. These provide the physical layers which will embody complex organizations, such as the ones envisaged for future generations of networked vehicle systems.

A control and computational framework for organizations of networked vehicles systems is proposed and a systematic design methodology within which properties of the organization can be proved to satisfy formal requirements is introduced. The framework encompasses a control and computation architecture and a design methodology. The architecture derives from a few design principles and is implemented with the help of a few mechanisms. The design methodology introduces a compositional layered approach to control and computation that is uniform for all vehicles.

A few mechanisms enable the correct composition of physical and computational entities. Correct is defined in the sense of conforming to organizational principles. The mechanisms include access control to physical entities, access control to specialized controllers over physical communication channels, switching control dependencies between controllers, and addressing controllers in charge of organizations. These mechanisms are sufficient to enable a vehicle to become part of an organization, thus benefiting from properties of the organization, while changing the way it presents itself to the external entities, to conform with organizational rules. An organization has properties that are a function of structure, which is controlled to deliver these properties while it may be controlled to satisfy high level behavior specifications. Organization arises from structure, and structure has to conform to organization principles that establish roles for constituent elements, controller dependencies, and message protocols. In addition, organization has provisions to cope with operations in communications challenged environments. In these environments communications came and go. This has implications in control and computation. The system will alternate between states of communications and of silent motions. Communications allow the development of coordinated plans ensuring that the vehicles will communicate again after periods of silent motions. Motion controllers will have to ensure that the vehicles execute these plans to rendezvous for communications after periods of silent motions. This is what keeps the system alive. This is also an essential property of these systems.

The compositional layered approach allows the use of assume guaranteed reasoning techniques to prove properties of the system. Lower level properties are studied in the framework of reach set computations. This allows to check the feasibility of planned motions and provides a framework to derive motion controllers with guaranteed results. Higher level properties are then proved under the assumptions of guaranteed low level behavior. This is done in the framework of simulation and bi-simulation relations relative to formal specifications.

Dynamic optimization plays a crucial role in our developments. First, reachability techniques allow assumed guaranteed reasoning. Reachability techniques are used for verifying plans and to synthesize controllers with guaranteed results. Second, the true power of dynamic optimization comes into play in problems with iterated multi-vehicle rendezvous. This is because the structure of these problems allows a structured application of the principle of optimality that results in coordinated optimization problems formulated in lower-dimensional spaces. The structure induces the composition of value functions in these lower-dimensional spaces, thus avoiding the problem of working in the product of the space states for all vehicles. Caution is required here. The structure of the problem, which is derived from the associated cost functions, may preclude the application of the principle of optimality.

Finally, the dynamic optimization framework developed in this work allows the consideration of optimization problems for systems with non-trivial state-space and control spaces.

Chapter 2

Challenges in networked vehicle systems

The computation and control challenges arising in the coordination of multi-vehicle systems are discussed in the framework of (coupled) physical and computational dynamics. The challenges are formulated as classical control problems of optimization, invariance and attainability for systems governed by the laws of physics and computation. Directions for future research are discussed with special emphasis on the aspects of coupled dynamics and dynamic structure that seem to be missing in the literature.

2.1 Introduction

This chapter is about the computation and control challenges posed by systems exhibiting both (coupled) physical and computational dynamics and dynamic structure. Section 2.2 discusses the networked vehicle systems problem domain.

Section 2.3 presents an example to illustrate these two aspects of the behavior of networked vehicle systems. The example draws from experience in designing, building and deploying networked vehicle systems and was motivated by the developments from the *Control for Coordination* FP7 project. Section 2.4 discusses elements of a control model for these systems. First, we introduce a clear distinction between physical and computation entities and briefly describe the underlying state and control spaces. Second, we explain the couplings between the physical and computation dynamics and show how these affect the selection controls. Section 2.5 shows how behavior specifications can be phased in terms of concepts used in traditional control specifications when we consider the modeling concepts introduced previously. Finally, in section 2.6 we discuss the computation and control challenges in this modeling framework. Directions for a research agenda are discussed with special emphasis on the aspects of coupled dynamics and dynamic structure that seem to be missing in the literature.

2.2 Networked vehicle systems

The rich and exciting research over the past decade concerning the coordination of multiple vehicles has been focused on systems with fixed structure [46]. Structure is typically described in terms of geometric formations, and the properties of formation controllers are studied in the framework of stability and graph theories. Recent developments have also incorporated new results from the theories of network control systems. However, the scope of coordination is still limited to relative motions.

Motion coordination is just one aspect of multi-vehicle coordination. This becomes more evident in networked vehicle systems consisting of heterogeneous ground, air and ocean vehicles interacting over inter-operated, and possibly intermittent, communication networks [28, 71, 31]. For example, in networked vehicle systems, information and commands are exchanged among multiple vehicles, sensor nodes and operators, and the roles, relative positions, and dependencies of these vehicles and systems change during operations. Moreover, these systems may exhibit properties that are a function of structure, where structure arises from interactions established over physical, sensing and communication links. Links change over time; the same happens with interactions established over these links. These are *systems with dynamic structure*.

The control of systems with dynamic structure poses new challenges to control engineering and computer science. These challenges entail a shift in the focus of existing methodologies: from prescribing and commanding the behavior of isolated systems, or tightly coupled systems, to prescribing and commanding the behavior of dynamically interacting networked systems – this may be one of the reasons why we are still far from realizing the potential of these systems.

The fact is that, in spite of developments in the control of distributed systems [104], research in control engineering has not yet incorporated fundamental concepts such as link, interaction, and dynamic structure. In contrast, computer scientists were already making strides in this area in the early 90's, in part because of the pioneering work of Robin Milner. The following quote from Milner highlights these points [74]:

> Dynamic reconfiguration is a common feature of communicating systems. The notion of link, not as a fixed part of the system but as a datum that we can manipulate, is essential for understanding such systems. What is the mathematics of linkage? The theories of computation are evolving from notions like value, evaluation and function to those of link, interaction and process.

Milner's questions were partially addressed in the Pi-calculus [75] (a continuation of Milner's work on the process calculus CCS (Calculus of Communicating Systems) [73]), a calculus of communicating systems in which the component agents of a system may be arbitrarily linked and the communication over linked neighbors may carry information which changes that linkage.

Meanwhile, the advent of ubiquitous mobile computing introduced a new modeling challenge. While the Pi-calculus deals well with mobile connectivity, it does not handle mobile locality. Ubiquitous systems need both. This was the motivation behind the development of the theory of Bi-graphical Reactive Systems (BRS's) by Milner and co-workers [76]. The theory is based on a graphical model of mobile computation that emphasizes both locality and connectivity. The theory evolved from process calculi, especially the calculus of Mobile Ambients (invented by L Cardelli and A Gordon [13] deals with spatial reconfiguration) and the Pi-calculus. A bi-graph comprises a place graph, representing locations of computational nodes, and a link graph, representing interconnection of these nodes. Mobile connectivity and locality are expressed with BRS's by defining a set of reaction rules. A reaction rule is a pair of bi-graphs, *redex* and *reactum*, where the *redex* defines a pattern to be matched with a bi-graph modeling the current state of a system. A reaction is simply the substitution of a *redex* with a *reactum*. In this model, systems of autonomous agents interact and move among each other, or within each other.

A careful examination of these developments in computer science may prove invaluable to control engineering, especially in what concerns the coordination of networked vehicle systems. First, because they draw our attention to models of mobile connectivity and mobile locality, which are intrinsic to dynamic structure and coupled dynamics – this is the true essence of cyber-physical systems [4]. Second, because they do not seem to tackle the control of mobile connectivity and mobile locality, namely how to "guide" systems of autonomous agents to interact and move among each other, or within each other, according to some specification.

In the theory of BRS's the structure of locations of a system is modeled with a place graph, which is restricted to have a tree-structure. The assumption is that the topography of a system can be modeled as a set of domains or objects contained within each other. Connections between objects or domains are modeled by links. A place graph may fail to capture several types of geometric relations occurring in networked vehicle systems. First, locations may move, change geometry, and intersect. Second, locations may be permanently associated with mobile computational nodes (e.g., communications range of a physical device). The link graph may fail to capture the intrinsic hierarchical structure of links and interactions among networked vehicles. This is because communication links, which are location-dependent, enable interactions among computational nodes; the failure of a communication link may entail the failure of a complex structure of interactions (which are basically another type of links). The BRS's models of mobility allow the migration of computational nodes, a capability that may open a completely new research direction for control engineering, but fail to capture the fine-grained space-time dynamics of interacting vehicle systems. In addition the mobility of vehicles may entail the mobility of locations, but this relation does not hold for computational processes.

Control engineers have approached the design of complex systems in the framework of control architectures [97], in which a complex design problem is partitioned into a number of more manageable sub-problems. There are several partitioning techniques, being layering the most used one in real applications [99]. However, the language of control architectures, with the exception of developments in the framework of dynamic networks of hybrid automata[1], has been missing the semantically rich concepts evoked by mobile connectivity and locality. On the other hand, research on BRS's models is missing the principled design approaches associated with control architectures. The interesting question is then: *What is the architectural organization required to support mobile connectivity and locality in a networked vehicle system tasked to satisfy some high level control specification?*

The architectural organization will have to include mechanisms for context awareness (to adapt the behavior depending on the "context" at hand), for robustness (to sustain computational interactions in the presence of failures of communication links), for estimation of external behavior (to estimate the evolution of components out of communications range), for state and data propagation (to ensure delivery of data and state updates in the presence of intermittent communications), and for setting up controller structures (to evolve the architecture).

[1]Informally, dynamic networks of hybrid automata [36] allow for interacting automata to create and destroy links among themselves, and for the creation and destruction of automata. Formally, for each hybrid automaton, there are two types of interactions (mediated by means of communications): 1) the differential inclusions, guards, jump and reset relations are also functions of variables from other automata, and, 2) exchange of events among automata. At the level of software implementation, the mechanisms by which software modules interact are called models of computation. The choice of the model of computation (or mix of models) is quite application dependent [65]. This is particularly difficult for dynamic networks of hybrid automata.

2.3 Example

One example will help to understand the challenges posed by the organization of future generations of networked vehicle systems – the field is still in its infancy and reliability is still the main concern for the current generations.

Consider the problem of managing a team of autonomous vehicles operating 24/7 in a remote region. The operation consists of monitoring a geographically distributed phenomena (e.g., the levels of radiation at sea). The vehicles operate from a base, which is used for refueling and mission planning. There are no direct communication links between the base and the remote region. Vehicles are used as data mules to transport data between the base and the remote region. Short range communications are used for team coordination in the remote region. Team coordination is done by a team controller, a computational entity which runs on a designated vehicle, the team leader. The team controller migrates to a new vehicle when the vehicle where it resides returns to the base for refueling – this is an instance of the coupling between physical and computational dynamics. There is another controller at the base to control the overall operation. Data arriving from the remote region is used to update estimates of the status of the remote operations. Based on these updates, the base controller may generate a new controller for the remote team leader. The new controller is sent to the team leader by vehicles departing to the region.

2.4 Models

This is an example of coordination problems for systems consisting of entities that evolve, interact and communicate in a common environment that can be modified through the actions of these entities. There are two types of entities in these systems: physical and computational entities. The former are governed by the laws of physics, the later by the laws of computation. Physical entities may interact among themselves and with the environment and can be “composed” to form other physical entities. Computational entities interact through communications. Physical entities may affect computational entities through sensing links. Examples of physical entities include vehicles, sensors, communication devices, computers, and human operators. Physical entities have attributes, which may change with time, e.g., consumable resources, and lodge computational entities. Computational entities may create other computational entities, and may be deleted as well. Some computational entities may be the capability to migrate between physical entities over communication channels. Communications may be geographically constrained. Physical entities can be used to transport computational entities and information across regions where communications are not available. This is also used for maintaining knowledge representation and consistency across the system. The "composition" of computational entities is either local, with respect to the one physical entity where they reside, or distributed, over communicating physical entities.

In what follows we consider systems of the form *System* = (*PhysicalEntities*, *Environment*, *ComputationalEntities*, *SystemConstraints*).

PhysicalEntity = (*StaticAttributes*, *Dynamics*, *Outputs*, *Constraints*). *StaticAttributes* is the vector of the time-invariant attributes such as type, computational and communication capabilities; *Dynamics* are the continuous and discrete dynamics which may affect, and be affected, by the environment (this may lead to non-intended consequences or side-effects through causal pathways); *Outputs* is the vector of outputs; and *Constraints* represent the state and control constraints. The discrete dynamics has set-valued state variables to model (dynamic) physical and computational interactions with other entities.

Environment models the environment where the elements of *PhysicalEntities* evolve. It has a controlled component, to model environmental aspects that depend on the actions of physical entities (e.g., electromagnetic radiation generated by a set of radars), and an uncontrolled component, to model the aspects of the environment which do not depend on these actions (e.g., terrain and wind fields).

ComputationalEntity is a generic term for software components that encode controllers and other computations. There are two types of computational entities: atomic and composed. An atomic computational entity resides on a physical entity; a composed entity may be distributed over a network in strict accordance to composition rules to ensure that these are well formed. Composition is dynamic in that it can evolve over time, for example, in a dynamic communication network. Atomic computational entities may be allowed to migrate between physical entities over a communication channel. Computational entities can be created and deleted on the fly. Each *PhysicalEntity* is abstracted by one atomic computational entity to bridge the physical and computational worlds. Abstractions of physical entities are not allowed to migrate.

SystemConstraints model constraints in the complex state-space of *System*. In the previous example, the team controller runs on a team leader. The team leader exists only in the given region and it changes over time.

2.5 Specifications

The modeling concepts from the previous section allow the specification of behaviors for a networked vehicle system in terms of traditional specification patterns from control engineering. We discuss specifications for a few representative problems to deepen our understanding of the underlying computational and control challenges.

Invariance. The generic problem of invariance involves a pair (*System*, S), where S is a set in the state-space of *System*. In this problem, the state of the *System* is required remain in S if the initial state is in S. This generic formulation allows us to express constraints on the controlled component of the environment, as well as on physical and computational mobility, physical interactions, communication links, etc. For example, we may require a controller to "stay" in a given region independently of the physical entity where it resides; or we may want to have at least one vehicle in a given region.

Attainability. In the problem of attainability we require the state of the *System* to "attain" a set Γ within a given time interval τ. As with invariance, this specification allows us to consider complex physical and computational target sets.

Optimization. The specification of optimization problems involves departure and target sets, state constraints, information structures, control spaces, cost functions, and the "mood" of the problem (cooperative, adversarial, etc.).

As before, departure and target sets and state-constraints are defined in the complex state-space of the *System*. This enables us to encode non-standard specifications (e.g., permissions for the migration of computational entities or for network access).

Full state information may not be accessible in the *System*. Information structures, which concern who knows what and what is sent to whom, are affected by the mobility of physical entities in communications challenged environments. This leads to dynamic information structures, i.e., those depending on the state of the system.

Control spaces and control constraints can be very complex. Each physical entity may affect other physical entities and the environment. This may lead to some level of indirectness when it comes to finding optimal controls. For example, the cost function may depend on the environment, which may be affected by the motions of physical entity A which, in turn, may be disabled by the actions of physical entity B. We need to identify causal control pathways, which link actions to their effects, with causal constraints not only based on commitments in the past, but potentially in the future. The challenge is that causal control pathways may be dynamic since future commitments might change with time. In addition, the effects of control actions may be significantly delayed (e.g., dropping a bomb or migration of a computational entity). Finally, it is up to the designer to specify the control space for the controllers in a system (e.g., change control authority and add/remove state-constraints or permissions for establishing links of communication).

The global performance (or cost) of a set of interacting computational entities and supporting physical entities depends on the initial, terminal, integral and switching costs (incurred when switching between discrete controls). Each of these costs may have terms associated to physical and computational interactions (e.g., cost may depend on the structure), in addition to terms associated to physical and computational entities (e.g., the cost of computations). Cost functions may also depend on predicates on the state of the world (e.g., in military operations we may want to switch from minimum risk to optimal time formulations when the level of threat drops below some threshold), thus introducing non-Lipchitz dependencies. The performance evaluation of persistent 24/7 operations also presents new challenges to optimization. This is partially related to the fact that physical entities enter and leave the system.

A high level interpretation of the behaviors exhibited by the systems under consideration is in order. Generally speaking these systems evolve through phases. In each phase, a sub-set of the constituent vehicles may operate on their own, while the remaining vehicles may form clusters where several types of interactions may take place. Switching between consecutive phases is triggered by events such as the achievement of partial or global goals, failures, or environmental changes – vehicles can modify and sense the environment, which can be used for signaling. The switching logic triggers the formation of new clusters, the generation of the corresponding goals, and distributed goal allocation. The new goals should enable communications at the end of the phase, so that the process can start again – this is what keeps the system alive. Basically the system alternates between the computation of goals and the control of itself to reach these goals.

An abstract control interpretation of these behaviors is as follows. In each phase there is a set of concurrent, and possibly coupled, invariance, attainability and optimal controllers; phase switching entails changing controllers and associated interactions. The hypothesis is that control-inspired specifications suffice to specify the behaviors for a large class of systems, if not for all networked vehicle systems.

2.6 Control and computation

The problem of designing controllers for physical entities, either operating in isolation or in a system with fixed structure, is generally well understood. This is not the case with a networked vehicle system, where loosely coupled physical and computational entities interact in communications challenged environments.

Given a generic specification for the behavior of a system, the design problem consists of deriving a structure of computational entities which, when "composed" with the system, will satisfy the specification in some sense to be defined.

This design problem presents new challenges to computation and control: 1) these systems have complex state and control spaces and coupled physical and computational dynamics; 2) physical and computational dynamics may depend both on physical and computational interactions through complex pathways of causality; 3) physical and computational interactions are dynamic, and have constraints on location and linking; 4) networks of physical and computational entities have properties which depend on the structure of these networks; 5) physical entities may enter and leave the system, while computational entities may be created/destroyed on the fly; 6) physical and computational entities are distributed over the underlying physical and computational spaces; 7) physical entities may have limited autonomy, thus requiring periodic refueling; 8) control actions available to computational entities may include the generation of new controllers (this requires controllers to know how to generate other controllers); and, 9) state may not be directly accessible by all computational entities. These challenges are not unique to networked vehicle systems. This discussion may lead to new insights in other fields, such as biology or ecology. Moreover, comparative studies may lead to new insights for architectural design in networked vehicle systems and, why not, to new ways of co-designing computational and physical components, vehicles included.

Chapter 3

Dynamic optimization background

Key concepts and results in dynamic optimization are introduced as background for developments in control and optimization of networked vehicle systems.

3.1 A simple derivation of the Hamilton-Jacobi-Bellman equation

We start with a simplified version of the time optimal problem to reach a target set S for the purpose of illustrating the main ideas behind the derivation of the Hamilton-Jacobi-Bellman equation for this problem. For a thorough treatment of this problem see [6], pag. 239.

3.1.1 The problem

Consider the following model of a system whose state x evolves in $\mathbb{R}^n$:

$$\dot{x}(t) = f(x, u), u \in U \subset \mathbb{R}^p \tag{3.1.1}$$

where f satisfies the conditions for existence and uniqueness of the ordinary differential equation and u is our control.

Consider the equivalent representation of the same system:

$$\dot{x}(t) \in F(x) \subset \mathbb{R}^n \tag{3.1.2}$$

where $F(x) := \{s : s = f(x, u), u \in U\}$.

Remark 3.1.1 (Local controllability). The condition $0 \in Int(F)$ is necessary for local controllability.

Assumption 3.1.1. *In what follows we assume that the system (3.1.1) is locally controllable.*

Consider a bounded and closed set S with non-empty interior.

Let t_f denote the first time when the trajectory of the system hits the target set S.

$$t_f = \inf\{t : x(t) \in S\} \tag{3.1.3}$$

Consider the following problem.

Problem 3.1.2. *Let $x(0) = x_0$. Find:*

$$\inf_{u(.)} t_f \tag{3.1.4}$$

where $u(.) : \mathbb{R} \to \mathbb{R}^p$ is an admissible control function.

Under the stated assumptions for system and for the target set the infimum is attained at a time $T \in \mathbb{R}$.

Introduce the value function $T : \mathbb{R}^n \to \mathbb{R}$ as

$$T(x) = \inf_{u(.)} t_f \tag{3.1.5}$$

3.1.2 Principle of optimality

Take a trajectory departing from $x(0) = x_0$. Consider a pair (x^*, t) on this trajectory. The principle of optimality for this problem can be expressed as follows:

$$T(x_0) \leq t + T(x^*) \tag{3.1.6}$$

Equality holds for optimal trajectories. The interpretation is quite simple. If a point is on the optimal trajectory, it is optimal to stay on the optimal trajectory.

3.1.3 Hamilton Jacobi Bellman equation

Assumption 3.1.3. *The value function T is differentiable.*

The Hamilton-Jacobi-Bellman equation for this problem can be interpreted as an infinitesimal version of the principle of optimality. For this purpose divide both terms of equation (3.1.6) by t and take limits when $t \to 0$. Keep in mind that we are taking the total derivative of T with respect to t. This means that first we take the derivative with respect to x and multiply it by the derivative of x with respect to time.

$$\inf_{u \in U} -\nabla T(x_0) \cdot f(x, u) = 1 \tag{3.1.7}$$

This is a partial differential equation. The boundary condition is $T(x) = 0, x \in S$.

3.2 The Hamilton-Jacobi Equation and Viscosity Solutions

This section introduces a more thorough discussion of the Hamilton-Jacobi-Bellman equation and presents concepts of solution of this equation. The section follows closely the book [39] and the article [15]. The developments are done in the framework of non-smooth analysis (background is provided in an Appendix).

3.2.1 The optimal control problem

Consider the following optimal control problem OCP:

$$\text{Minimize} \quad l(x(T)) \quad \text{subject to}$$

$$\dot{x}(t) \in F(x(t)), x(0) = x_0 \tag{3.2.1}$$

Where:

- $T > 0, x_0 \in \mathbb{R}^n$
- F satisfies the Standing Hypotheses
- The function $l : \mathbb{R}^n \to \mathbb{R}$ is continuous.

Assumption 3.2.1. *Throughout this section we will assume that F is locally Lipschitz and autonomous.*

Definition 3.2.1 (Value Function V)**.** Consider the problem 3.2.1. The corresponding Value function V is defined as:

$$\forall \tau \leq T, \forall \alpha \in \mathbb{R}^n, V(\tau,\alpha) := inf\{l(x(T)) : \quad \text{x is a trajectory of F on } [\tau, T] \text{ with } \quad x(\tau) = \alpha\} \tag{3.2.2}$$

Proposition 3.2.2. *Consider the hypotheses of problem 3.2.1. Then:*

1. *The infimum defining the Value function is attained.*
2. *Since F is locally Lipschitz then V is continuous on $(-\infty, T] \times \mathbb{R}^n$ and locally Lipschitz if l is locally Lipschitz.*

Remark 3.2.1. This proposition asserts the existence of solution for problem 3.2.1

Remark 3.2.2. The results of this chapter are extended for the case $l \in \mathcal{F}(\mathbb{R}^n)$ in [15]. This extension allows the implicit incorporation of explicit end-point constraints of the form $x(T) \in D$ since, in this case, $l(x(T)) = \infty$ when x(T) fails to lie in D.

3.2.2 Verification functions

An extension of an idea in the calculus of variations, introduced by Legendre, leads to sufficient conditions for optimality in control problems.

Problem 3.2.3. *Given a feasible arc $\overline{x}$, how can we confirm that $\overline{x}$ is a solution for the problem 3.2.1?*

Definition 3.2.2 (Verification function φ)**.** Produce a C^1 function φ such that:

$$\varphi_t(t,x) + \langle \varphi_x(t,x), v\rangle \geq 0, \forall (t,x,v) \tag{3.2.3}$$

$$\varphi(T,.) = l(.) \tag{3.2.4}$$

$$\varphi(0,x_0) = l(\overline{x}(T)) \tag{3.2.5}$$

To prove that the existence of φ verifies that $\overline{x}$ is optimal consider another feasible arc x. Then, a.e. on [0,T]:

$$\frac{d}{dt}\varphi(t,x(t)) = \varphi_t(t,x(t)) + \langle \varphi_x(t,x(t)), \dot{x}(t)\rangle \geq 0 \tag{3.2.6}$$

Integrating on [0,T] yields:

$$\varphi(T,x(T)) = l(x(T)) \geq \varphi(0,x_0) = l(\overline{x}(T)) \tag{3.2.7}$$

It also follows that:

$$\varphi(0,x_0) = V(0,x_0) \tag{3.2.8}$$

Definition 3.2.3 (Extended hamiltonian)**.** $h_e : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$:

$$h_e : (x,\theta,\xi) \mapsto \theta + h(x,\xi) \tag{3.2.9}$$

Remark 3.2.3. The Hamilton-Jacobi inequality 3.2.3 was used to deduce that the map $t \to \varphi(t,x(t))$ is increasing whenever x is a trajectory. In other words, we want to find φ such that (φ, F) is strongly increasing. It is simple to extend the results concerning monotonicity when φ has t-dependence. Consider t as an extra variable subject to $\dot{t} = 1$ and define $F_e(x,t) = F(x) \times \{1\}$. Then (φ, F) is strongly increasing iff $h_e(x, \partial_P\varphi(t,x)) \geq 0, \forall (t,x)$. This proves the following result.

Proposition 3.2.4. *Let $\overline{x}$ be feasible for problem 3.2.1, and suppose there exists a continuous $\varphi(t,x)$ on $[0,T] \times \mathbb{R}^n$ satisfying:*

$$h_e(x, \partial_P\varphi(t,x)) \geq 0, \forall (t,x) \in (0,T) \times \mathbb{R}^n \tag{3.2.10}$$

$$\varphi(T,.) = l(.) \tag{3.2.11}$$

$$\varphi(0,x_0) = l(\overline{x}(T)) \tag{3.2.12}$$

Then $\overline{x}$ solves problem 3.2.1 and $\varphi(0,x_0) = V(0,x_0)$

The obvious question is:

Problem 3.2.5. *Does a verification function φ exists when $\overline{x}$ is optimal?*

In order to answer this question we consider the invariant embedding of the problem 3.2.1 in a family of problems $P(\tau,\alpha)$ parametrized by the initial data $(\tau,\alpha) \in [0,T] \times \mathbb{R}^n$.

Remark 3.2.4. φ is increasing along all trajectories of F, including those starting at (τ, α). Consider the optimal trajectory x^* for $P(\tau, \alpha)$. Hence we have:

$$V(\tau, \alpha) = l(x^*(T)) = \varphi(T, x^*(T)) \geq \varphi(\tau, \alpha)$$

This proves that:

$$V(\tau, \alpha) \geq \varphi(\tau, \alpha), \forall (\tau, \alpha) \in [0, T) \times \mathbb{R}^n \tag{3.2.13}$$

This fact raises the question: is V a verification function? The following proposition answers this question affirmatively.

Proposition 3.2.6. *A feasible arc $\overline{x}$ is optimal iff there exists a continuous verification function for $\overline{x}$. The Value Function V is one such verification function.*

Proof. Here we sketch part of the proof.

- V satisfies 3.2.10 since V(t, x(t)) is always increasing when x is a trajectory. (Principle of optimality).
- V is continuous from proposition 3.2.2.
- V satisfies equations 3.2.11, 3.2.12 by definition.
- V satisfies equation 3.2.13 if x is optimal.

□

Remark 3.2.5. The continuity hypothesis on l is too restrictive since it rules out endpoint constraints. The following theorem, addresses this question, by producing a continuous verification function that is not (it cannot be) the Value Function V when the endpoint constraints are active.

Theorem 3.2.7. *When problem 3.2.1 is normal (see [15]), a feasible arc $\overline{x}$ is optimal iff there exists a Lipschitz continuous verification function φ for $\overline{x}$.*

3.2.3 The proximal Hamilton-Jacobi equation

The following theorem shows that the Value Function V is the unique continuous solution of a suitable generalization of the Hamilton-Jacobi equation whose general form is:

$$\varphi_t + H(t, \varphi_x) = 0$$

with boundary condition:

$$\varphi(T, .) = l(.)$$

Theorem 3.2.8. *There is an unique continuous function $\varphi : (-\infty, T] \times \mathbb{R}^n \to \mathbb{R}$, called a proximal solution of the Hamilton-Jacobi equation, satisfying:*

$$h_e(x, \partial_P \varphi(t, x)) = 0, \forall (t, x) \in (-\infty, T) \times \mathbb{R}^n. \tag{3.2.14}$$

$$l(x) = \varphi(T, x), \forall x \in \mathbb{R}^n \tag{3.2.15}$$

This function is the Value Function V.

Proof. There remains to prove that:

$$h_e(x, \partial_P \varphi(t,x)) \leq 0, \forall (t,x) \in (-\infty, T) \tag{3.2.16}$$

and the uniqueness statement.

Whenever $V(\tau, \alpha)$ is finite, there is an optimal arc $\overline{x}$ for the problem $P(\tau, \alpha)$ and, along this arc, V is constant. Then, the system (V,F) is weakly decreasing relative to $t \in (-\infty, T)$ and this implies 3.2.16.

To prove uniqueness consider another function φ satisfying 3.2.14.

We will prove first that $(\varphi \geq V)$. Consider any point $(\tau, \alpha) : \tau < T$. Then (φ, τ) is weakly decreasing relative to $t < T$. This implies the existence of a trajectory

$$x([0,T]), x(\tau) = \alpha : \varphi(t, x(t)) \leq \varphi(\tau, \alpha), \forall t \in [\tau, T)$$

As $t \uparrow T$ we derive $(l(x(T)) = \varphi(T, x(T)) \leq \varphi(\tau, \alpha)) \Rightarrow V(\tau, \alpha) \leq \varphi(\tau, \alpha)$

Now we will show that $V \geq \varphi$. Consider any point $(\tau, \alpha) : \tau < T$. Then, there exists an optimal trajectory $\overline{x}$ for $P(\tau, \alpha)$. Because (φ, F) is strongly increasing we have:

$$\varphi(T, \overline{x}(T)) \geq \varphi(\tau, \alpha)$$

But $\varphi(T, \overline{x}(T)) = l(\overline{x}(T)) = V(\tau, \alpha)$. □

Corollary 3.2.9. *Let $\varphi : (-\infty, T] \times \mathbb{R}^n \to \mathbb{R}$ be continuous and satisfy:*

1. $h_e(x, \partial_P \varphi(t,x)) \leq 0, \forall (t,x) \in (-\infty, T) \times \mathbb{R}^n$.
2. $l(x) \leq \varphi(T, x), \forall x \in \mathbb{R}^n$

Then $\varphi \geq V$

Corollary 3.2.10. *Let $\varphi : (-\infty, T] \times \mathbb{R}^n \to \mathbb{R}$ be continuous and satisfy:*

1. $h_e(x, \partial_P \varphi(t,x)) \geq 0, \forall (t,x) \in (-\infty, T) \times \mathbb{R}^n$
2. $l(x) \geq \varphi(T, x), \forall x \in \mathbb{R}^n$

Then $\varphi \leq V$

Remark 3.2.6. The corollary 3.2.9 is valid without the Lipschitz hypothesis on F, in contrast with the other one.

3.2.4 Minimax solutions

It is possible to express the extended Hamilton-Jacobi equation in terms of other constructs of nonsmooth analysis, for example via subderivatives. The next concept of solution was called minimax solution by Subbotin.

Proposition 3.2.11. *V is the unique continuous function $\varphi : (-\infty, T] \times \mathbb{R}^n \to \mathbb{R}$ satisfying:*

1. $\inf_{v \in F(x)} D_\varphi((t,x);(1,v)) \leq 0, \forall (t,x) \in (-\infty, T] \times \mathbb{R}^n$
2. $\sup_{v \in F(x)} D_\varphi((t,x);(-1,-v)) \leq 0, \forall (t,x) \in (-\infty, T] \times \mathbb{R}^n$
3. $\varphi(T, .) = l(.)$

Proof. It suffices to prove:

- Condition 1) of corollary 3.2.9 is equivalent to condition 1) of the proposition.
- Condition 1) of corollary 3.2.10 is equivalent to condition 2) of the proposition.

This can be concluded from the properties of the subderivative, namely lower semicontinuity in v. □

3.2.5 Viscosity solutions

The value function is also the unique viscosity solution of the Hamilton-Jacobi boundary value problem. This solution concept, developed by Crandall and Lions is bilateral like minimax solutions, and additionally it uses both subdifferentials and supperdifferentials.

Remark 3.2.7. Given the uniqueness of the solution, all solution concepts coincide in the present setting.

Proposition 3.2.12. *V is the unique continuous function* $\varphi : (-\infty, T] \times \mathbb{R}^n \to \mathbb{R}$ *satisfying:*

1. $h_e(x, \partial_D \varphi(t,x)) \le 0, \forall (t,x) \in (-\infty, T) \times \mathbb{R}^n$
2. $h_e(x, \partial^D \varphi(t,x)) \ge 0, \forall (t,x) \in (-\infty, T) \times \mathbb{R}^n$
3. $\varphi(T, .) = l(.)$

3.2.6 Feedback synthesis from semi-solutions

Definition 3.2.4 (Semi-solution)**.** The Hamilton-Jacobi inequality:

$$h_e(x, \partial_P \varphi(t,x)) \le 0, \forall (t,x) \in (-\infty, T) \times \mathbb{R}^n$$

with the boundary condition

$$l(x) \le \varphi(T,x), \forall x \in \mathbb{R}^n$$

defines a semisolution.

Remark 3.2.8. These conditions imply $V \le \varphi$. Thus, for each (τ, α) there is a trajectory $\overline{x}, \overline{x}(\tau) = \alpha$ such that $l(\overline{x}(T)) \le \varphi(\tau, \alpha)$.

Problem 3.2.13. *How to build a semi-solution?*

The classical approach

Assume that φ is smooth. Then,

1. For each (t,x) select a point $\overline{v}(t,x) \in F(x)$ such that the minimum defining $h_e(x, \nabla\varphi(t,x))$ is atained, i.e.:

$$\varphi_t(t,x) + \langle \varphi_x(t,x), \overline{v} \rangle = h_e(x, \nabla\varphi(t,x))$$

2. Then define the trajectory $\overline{x}$ as:

$$\dot{\overline{x}}(t) = \overline{v}(t, \overline{x}(t)), \overline{x}(\tau) = \alpha$$

3. If all these steps are feasible then

$$l(\overline{x}(T)) \leq \varphi(\tau, \alpha)$$

The proof of the last result is simple:

$$\begin{aligned} l(\overline{x}(T)) - \varphi(\tau, \alpha) &\leq \varphi(T, \overline{x}(T)) - \varphi(\tau, \alpha) \\ &= \int_0^T \frac{d}{dt}\varphi(t, \overline{x}(t))dt \\ &= \int_0^T \left\{\varphi_t(t, \overline{x}) + \langle \varphi_x(t, \overline{x}), \dot{\overline{x}} \rangle\right\} dt \\ &= \int_0^T h_e(\overline{x}(t), \nabla\varphi(t, \overline{x}(t)))dt \end{aligned}$$

Remark 3.2.9. The difficulties of this 'dynamic programming approach' are:

- Smoothness of φ
- Regularity of $\overline{v}$
- Existence of $\overline{x}$

Proximal aiming allows to overcome these difficulties for merely lower semicontinuous functions. The system monotonicity is proved using proximal methods instead of the integration step.

Theorem 3.2.14. *Let F be locally Lipschitz and let $\varphi \in \mathcal{F}((-\infty, T) \times \mathbb{R}^n)$ satisfy:*

$$h_e(x, \partial_P\varphi(t, x)) \leq 0, \forall (t, x) \in (-\infty, T) \times \mathbb{R}^n$$

and

$$l(x) \leq \lim_{t' \uparrow T, x' \to x} \varphi(t', x'), \forall x \in \mathbb{R}^n$$

Then for given $\tau, \alpha \in (-\infty, T) \times \mathbb{R}^n$, there exists a feedback selection $\overline{v}$ of F with the property that every Euler solution $\overline{x}$ of the initial value problem:

$$\dot{x}(t) = \overline{v}(t, x(t)), x(\tau) = \alpha$$

satisfies $l(\overline{x}(T)) \leq \varphi(\tau, \alpha)$

Chapter 4

Optimal UAV coordination in SEAD missions

We design the attack of the Blue force of unmanned air combat vehicles (UAV), against Red's ground force of SAM sites and radars. The design is structured in a two-level hierarchy of planning and execution. The plan, based on prior information, determines which targets are to be attacked and groups them into sub-tasks; allocates a UAV team to each sub-task; and selects a risk-minimizing path for each team. Execution is organized in a hierarchy of real-time controllers, which determine the actual flight path, weapons release, and space-time coordination of the actions of a UAV team. Plan development uses an algorithm that determines the sequence in which targets are to be attacked. Execution design is specified in *SHIFT*, a programming language for dynamic networks of hybrid automata.

4.1 Problem

The *Blue force* consists of UAVs with different capabilities (sensors, weapons). There is a *Red force* of ground SAMs and radars. We design Blue's attack against Red. The Red force threatens Blue's attacking UAVs. Blue's task is to destroy *primary* Red targets. To reduce risk Blue may destroy additional Red targets defending the primary targets. The technology and use of UAVs are discussed in [17].

Two difficulties must be faced. First, the design space of Blue's attack is large and heterogeneous: selecting targets and the order in which they are attacked; equipping and assigning UAV teams to these targets; coordinating the attack so that the order of attack is maintained; determining nominal flight paths; and then designing real-time feedback strategies for UAV flight and weapons use. We address this difficulty by structuring the design in two layers: an off-line plan, and an online execution control.

The planning procedure first invokes an algorithm that selects the targets and prescribes the order in which they must be attacked, keeping risk below a given threshold. As a side effect, the algorithm selects risk-minimizing nominal flight paths. The procedure next groups these targets into sub-tasks, assigns a UAV team to each, and specifies spatial and temporal coordination points so that the target attack order is maintained.

The execution control is decomposed into a hierarchy of task, sub-task, and sub-team controllers, vehicle supervisors, and elemental maneuver feedback controllers, which determine the actual flight path and weapons release. These controllers are described as interacting hybrid automata using *SHIFT*, a programming language for dynamic networks of hybrid automata. The ideas used in execution control are inspired by [99, 98, 27, 32].

The second difficulty originates in the requirement that the planning procedure and execution control must allow intervention by experienced human operators. In part this is because essential experience and military insight of these operators cannot be reflected in mathematical models, so the operators must approve or modify the plan and the execution. Also, it is impossible to design (say) vehicle and team controllers that can respond satisfactorily to every possible contingency. In unforeseen situations, these controllers ask the human operators for direction. Space limitations preclude discussion of how the plan and execution designs accommodate human intervention.

This chapter is an abridged version of [105]. Section 4.2 presents the planning procedure. Section 4.3 briefly describes the *SHIFT* language. Section 4.4 presents the distributed control architecture of UAV teams, and the controllers themselves. Section 4.5 collects some final remarks.

4.2 Planning procedure

The planning procedure organizes Blue's attack into a *plan*, comprising a set of *sub-tasks*, each of which is a list of targets to be destroyed. The plan must include the *primary* targets. We formally describe the plan 'design space', and performance measures to compare plans.

4.2.1 Threat

We need some definitions. *Target* is a generic term for Red force entities of different types such as SAM launchers and radars. There is a finite set of *types*, called *TargetTypes*. A target is characterized by its type and its (two-dimensional) location (x, y). So, a Red force with N targets is thus described by a set of the form

$$\begin{aligned} \textit{Targets} = \{\textit{target}_1 = (\textit{type}_1, (x_1, y_1)), \cdots, \\ \textit{target}_N = (\textit{type}_N, (x_N, y_N))\}. \end{aligned} \tag{4.2.1}$$

Prior knowledge of *Targets* is given by the initial distribution (at time $t = 0$), $P_{\textit{threat}}(0)$. This probability distribution is updated during execution when Blue makes observations.

We restrict $P_{\textit{threat}}(0)$ to a special form: The Red force is distributed over areas $A_1, \cdots, A_k$. In area A_j there are N_{tj} targets of type $t \in \textit{TargetTypes}$ whose locations are independently and uniformly distributed. The random number of targets N_{tj} are all independent with distribution $P_{tj}(N)$. This yields the form

$$\begin{aligned} P_{\textit{threat}}(0)(\textit{Targets}) = \\ \prod_t \prod_{j=1}^{k} \prod_{i=1}^{N_{tj}} p_{tj}(\textit{type} = t, (x_i, y_i)) P_{tj}(N_{tj}), \end{aligned} \tag{4.2.2}$$

in which t ranges over *TargetTypes*, and

$$p_{tj}(\textit{type} = t, (x_i, y_i)) = \begin{cases} |A_j|^{-1}, & (x_i, y_i) \in A_j \\ 0, & \text{otherwise} \end{cases} . \tag{4.2.3}$$

As the distribution (4.2.2)-(4.2.3) is specified by the list of areas $A = \{A_1, \cdots, A_k\}$ and the random vector $N = \{N_{tj}; t \in \textit{TargetTypes}, 1 \leq j \leq k\}$, we may denote it by $P_{A,N}$.

A target poses a *threat* over a circular area, centered at the target and with radius that depends on its type (not all targets present a threat). A UAV in this threat region will be destroyed with a certain probability that depends on the target type, the attitude and type of the UAV (which determines its radar signature), and the amount of time that the UAV is in the region. We assume that a numerical value can be assigned to the threat posed by a target at any location. We define the instantaneous threat function at any point (x, y) as

$$\begin{aligned} r(x, y; P_{A,N}) = \textstyle\sum_{j=1}^{k} \sum_t \sum_{N_{tj}=0}^{\infty} \sum_{n=1}^{N_{tj}} \\ \left[\int_{A_j} f_t(|(x, y) - (x_n, y_n)|) |A_j|^{-1} dx_n dy_n\right] P(N_{tj}). \end{aligned} \tag{4.2.4}$$

$f_t(d)$ is the instantaneous threat posed by a target of type t to a UAV at a distance d from the target. f_t may be of the form of an indicator function of a circle centered at the target location. So the integral in (4.2.4) is the expected value of this instantaneous threat posed by a target of type t located at a random point (x_n, y_n) that is uniformly distributed over area A_j. The sum over n is the threat posed by N_{tj} such targets. The sum over N_{tj} accounts for the random distribution of N_{tj}. The sum over t accounts for different types of targets. Finally the sum over j accounts for all the areas. The argument $P_{A,N}$ in r emphasizes the role of the threat. For the sake of clarity we have not incorporated the dependence on the UAV type in expression (4.2.4).

4.2.2 Minimum risk paths

We are given a set O of possible UAV origins, perhaps the UAV base locations, from which a UAV may be dispatched. The risk faced by a UAV flying at speed $v \geq v_{min} > 0$ along a path γ from $\gamma(0) = o \in O$ to a destination $\gamma(\tau) = d$, facing threat $P_{A,N}$ is defined as

$$\rho(\gamma; P_{A,N}) = \int_{\sigma=0}^{\tau} r(\gamma(\sigma); P_{A,N}) \frac{d\sigma}{v}, \tag{4.2.5}$$

in which r is given by (4.2.4). Hence the value function for threat $P_{A,N}$, with $\gamma(\tau) = (\bar{x}, \bar{y})$, is

$$V((\bar{x}, \bar{y}); P_{A,N}) = \min_{\gamma} \rho(\gamma; P_{A,N}). \tag{4.2.6}$$

The value function satisfies the *eikonal* equation

$$|\nabla V(x, y)| = c(x, y), c(x, y) > 0 \tag{4.2.7}$$

with boundary condition

$$V(x, y) = 0, \quad (x, y) \in O. \tag{4.2.8}$$

In (4.2.7),

$$|\nabla V(x, y)| = [(dV/dx)^2 + (dV/dy)^2]^{1/2},$$
$$c(x, y) = \frac{r(\gamma(\sigma); P_{A,N})}{v}.$$

The optimal paths γ^* follow $-\nabla V$,

$$\frac{d\gamma^*}{d\sigma}(\sigma) = -\nabla V(\gamma(\sigma)) \tag{4.2.9}$$

The 'fast marching' algorithm [88] efficiently solves the eikonal equation.

A UAV dispatched to attack a target will fly over a path γ, during which it will incur a certain risk, $\rho(\gamma)$, which can be translated into the probability that the UAV will survive the whole flight path

$$p(\gamma, P_{A,N}) = e^{-k\rho((\gamma; P_{A,N}))} \tag{4.2.10}$$

where $k > 0$ is a scaling factor.

We assume that the planner selects (or is given) a maximum risk threshold ρ_{max} and the probability $p_r(i)$ for removing each primary target i.

4.2.3 Plans

A *feasible plan* is a 4-tuple *plan* $= (\textit{TargetList}, \textit{TaskList}, \textit{PathList}, \succ)$, in which

(1) *TargetList* $= \{\textit{target}_1, \cdots, \textit{target}_n\}$ is a set of targets, and *PathList* $= \{\gamma_1, \cdots, \gamma_n\}$ are the paths;

(2) *Task* is a partition of *TargetList* into sub-tasks, and an assignment of teams of UAVs to sub-tasks, and

(3) $\succ$ is a precedence relation or partial order on *TargetList*, so that for all $i = 1, \cdots, n$

$$\rho(\gamma_i) \leq \rho_{max}. \tag{4.2.11}$$

The risk $\rho(\gamma_i)$ is calculated under the assumption that all targets $\textit{target}_j \succ \textit{target}_i$ have been destroyed.

The risk R associated with a plan is the maximum risk incurred along any path,

$$R(\textit{plan}) = \max_i \rho(\gamma_i). \tag{4.2.12}$$

The *plan design space* is the space of all feasible plans. An optimal plan has minimum risk,

$$R(\textit{plan}^*) = \min\{R(\textit{plan}) \mid \textit{plan} \in \textit{plan design space}\}.$$

The planning procedure consists of three procedures:

1. Target selection and path planning. This procedure is assisted by a software tool called Interactive Task Planner (ITP). For the deterministic planning problem (the locations of the targets are known) the ITP-planner interactions are as follows (see pages 34–37 of [105] for details). In *Step 1*, the planner selects a subset of the targets as *primary*, which he wants to attack. The ITP then calculates and displays additional, *potential* targets. These ‘protect’ the primary targets, i.e. one or more primary targets are included in the threat range of the potential targets. The planner next chooses a *maximum* risk level. In *Step 2*, the ITP determines the subset of potential and primary targets that can be attacked along paths starting at the Blue base, whose risk is less than the chosen maximum risk level. This subset of targets is called *Wave 1*. The ITP also determines the nominal risk-minimizing paths from the Blue base to each of Wave 1 targets. They are 'nominal' because the actual paths taken by the UAV are somewhat different and are determined by the 'task execution' module. We continue in this way. In *Step k*, the ITP determines the subset *Wave k-1* of additional targets that can be attacked along paths starting at either the Blue base or one of the *Wave k-2* target locations, whose risk is less than the maximum. The ITP determines and displays the nominal risk-minimizing paths for *Wave k-1* targets. The ITP graphically displays 'minimum risk' contours of locations that can be reached for each level of total risk.

Stopping condition The process stops at the smallest k for which one of two conditions holds:

$$\mathit{Wave}_{k-1} \neq \emptyset \quad \wedge \quad \mathit{Wave}_k = \emptyset \tag{4.2.13}$$

$$\mathit{PrimaryTargets} \quad \subset \quad \cup_{i=1}^{k} \mathit{Wave}_i. \tag{4.2.14}$$

The result is:

- A set of targets *TargetList* and a partial order $\succ$ (describing the waves), and the minimum risk for each target, based on the assumption that targets in Wave $i+1$ are attacked after those in Wave i have been destroyed;
- A set of paths *PathList* of nominal risk-minimizing paths for each target in Wave $i+1$, starting at the Blue base or at any target locations in Wave i;
- A 'sensitivity' matrix that gives the reduction in risk for each target in Wave $i+1$ due to the elimination of each target in Wave i.

Theorem 4.2.1. *If condition (4.2.13) holds, there is no plan that can destroy all primary targets with risk at most ρ_{max}. If condition (4.2.14) holds, the plan with*

$$\begin{aligned} \mathit{TargetList} &= \cup_{i=1}^{k} \mathit{Wave}_i, && (4.2.15)\\ \mathit{PathList} &= \cup_{i=1}^{k} \mathit{PathList}_i, && (4.2.16)\\ \succ &:= \mathit{Wave}_1 \succ \cdots \succ \mathit{Wave}_k, && (4.2.17) \end{aligned}$$

destroys all primary targets with risk at most ρ_{max}. Moreover, this plan is optimal if $R(plan)$ is the smallest risk for which (4.2.14) holds.

A *Task* is a pair $Task = \{(SubtaskList, \succ, (TeamList, assign, schedule)\}$ in which:

(1) $SubtaskList = \{subtask_1, \ldots, subtask_n\}$ is a set of sub-tasks, each of which is an array of legs,

$subtask_i = [leg_{i,1}, \dots, leg_{i,in}]$. Each leg consists of a single target and a 'nominal' path to the target. There is a partial order $\succ$ on the legs composing a task. The legs composing a sub-task satisfy a total order.

(2) $TeamList = \{team_1, \dots, team_n\}$, $\{assign : TeamList \to SubtaskList\}$ and $\{schedule : TeamList \to DeadlineList\}$ are respectively assignments of teams to sub-tasks, and of teams to deadlines.

2. Task specification. In this procedure the planner generates the *Task* structure with the help of the ITP. First, he uses the 'sensitivity' matrix and his knowledge of the battlefield to remove and/or adds targets to *TargetList* and to update (*PathList*, $\succ$) accordingly (this is the case when he adds targets of opportunity). This can be done automatically or visually with the help of the ITP. Observe that the targets and paths form a 'directed tree'. The idea is to construct sub-tasks that form chains and to generate $SubtaskList$. The tree structure results from the waves which present successive targets to a UAV flying to a primary target. This is why these waves are eliminated in sequence to minimize the risk accrued by an UAV flying over them. This opens safe corridors to the primary targets while minimizing the risk accrued by a team of UAVs attacking the primary targets.

Consider, as an example, the task represented in Figure 4.1, with 3 primary targets (P_1, P_2 and P_3) and 5 secondary targets ($S_1, \dots, S_5$). It consists of 3 sub-tasks to be executed concurrently. There are precedence relations on the order of execution of the legs composing the task. For example leg_0 of *sub-task*$_2$ precedes leg_4 of *sub-task*$_1$.

3. Team composition and tasking. This procedure consists in determining the assignments $(assign, schedule)$ (see pages 38–46 of [105] for details). The inputs are $p_r(i)$, the probability for removing each primary target, the set of available UAVs and their weapons. First, the minimum number of UAVs min_i^{sb} required to execute each sub-task and leading to the removal of the corresponding primary targets are calculated as follows: 1) the number of UAVs and weapons required to remove the final target in the sequence is calculated; 2) the calculation proceeds backwards in the sub-task sequence to determine min_i^{sb} at the beginning of the sequence. These calculations are based on the probabilities of surviving the paths and on the effectiveness of the weaponry. Finally, we determine the assignments $(assign, schedule)$ by solving a linear programming problem typical of resource allocation and scheduling formulations.

Each *plan* prescribes a set of constraints for execution. However, it does not prescribe a sequence of events and/or actions which would lead to poor execution performance in a non-deterministic world. This is why we need feedback strategies which, based on the available information and the constraints from the plan,

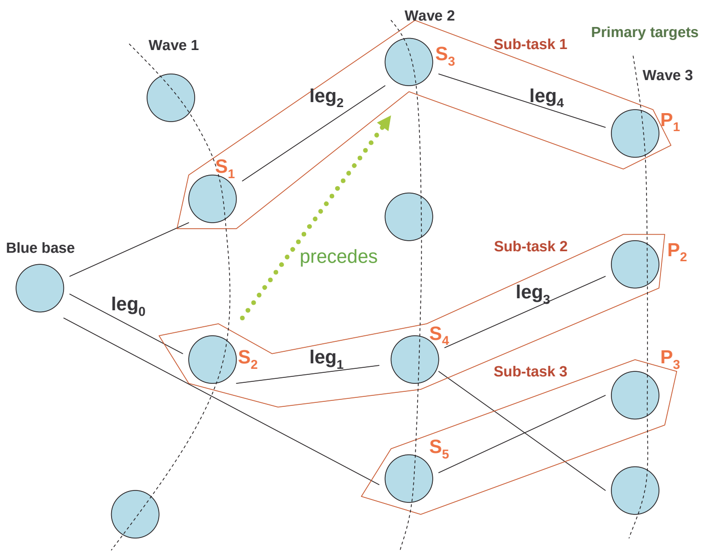


Figure 4.1: Task specification.

command UAVs to execute actions conforming to these constraints and leading to plan completion, while seeking to maximize some performance criteria.

4.3 An aside on *SHIFT*

SHIFT is a language for describing networks of hybrid automata. A *SHIFT* program begins with a definition of types (classes) with continuous and discrete behavior. A type includes a data model and a behavior specification. The data model consists of numerical variables, link variables, a set of discrete states, and a set of event labels. The variables are grouped into input, state, and output variables. The inputs and outputs of different components can be interconnected.

```
type Vehicle  {
  input      (what we feed to it)
  output     (what we see on the outside)
  state      (what is internal)
  discrete   (discrete modes of behavior)
  export     (event labels seen from the outside)
  flow       (continuous evolution)
  transition (discrete evolution)
  setup      (actions executed at create time)
}
```

The behavior is determined as follows. Each discrete state has a set of differential equations and algebraic definitions (flow equations) that govern the continuous evolution of numeric variables. The differential equations may involve outputs of other components accessible through link variables. Thus the set of components evolves as a hybrid automaton. System evolution alternates between the continuous mode, during which the evolution is governed by the flow equations, and the discrete mode, when simulation time is stopped and all possible (discrete) transitions are taken, as determined by guards and/or by event synchronization among components. During a discrete step components can be created, interconnected, and destroyed. *ShHIFT* allows dynamically reconfigurable input/output connections and synchronous composition. The first order predicate constructs of *SHIFT* (e.g. existential and universal quantification) provide compact representations of dynamic synchronous composition.

A simulation starts with an initial set of components that are instantiations of the defined types. Instances of components have unique names. The time-evolution of the set of components is derived from the behavior of these components.

4.4 Execution control framework

In this section we present the execution control framework and illustrate it with an attack task when the locations of the targets are known and the weapons' effectiveness is 1. The control framework is designed for more complex tasks (see pages 48–71 of [105] for details).

4.4.1 Execution concepts

The concepts for execution control build on experience in the modular design of distributed control hierarchies described in [99, 98, 27, 32]. We use the concept of maneuver – a prototype of an action/motion description for a single vehicle – as the atomic component of all execution concepts. Thus we abstract each UAV as a provider of maneuvers. A simple protocol governs the interactions between the UAV and an external controller: the external controller sends a maneuver (configuration) command to the UAV; the UAV either accepts the command and executes the maneuver (changes its configuration), or it does not accept the command and sends an error message to the controller; the UAV sends a 'done' message or an error message to the controller depending on whether the maneuver terminates successfully or fails. The UAV control is decomposed into a 2-level hierarchy of UAV supervisor and elemental maneuver feedback controllers (for details on UAV control see section V of [35]).

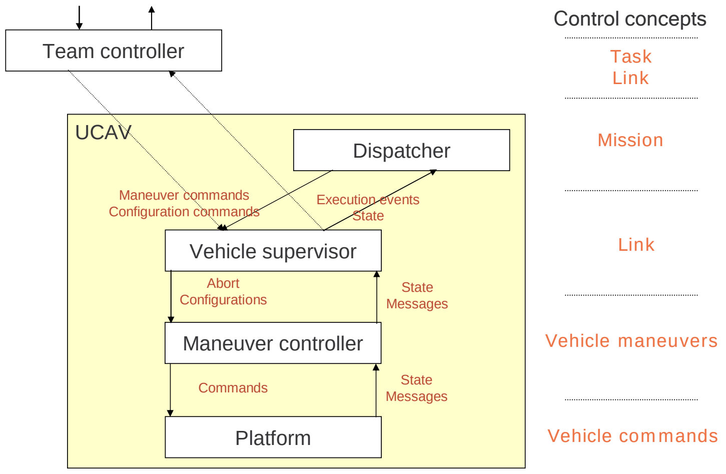


Figure 4.2: UAV control architecture.

The maneuvers required to execute the attack task are: *follow_path* – path following maneuver; *attack_target* – attack maneuver that maximizes the probability of destruction of the target while minimizing the accumulated risk; and *hold* – a holding pattern. The attack maneuver encodes the attack logic, specifically the control of weapons and risk reduction devices, such as jammers, and path optimization and execution procedures (this is done in a feedback manner). Other tasks may require other maneuvers such as: *bda* – execute battle damage assessment in a given region; *map* – map a given region; *search_destroy* – search for targets and attack them; and *evade* – evade some threat.

Next we summarize the execution strategy for the attack task. Each sub-task is executed by a team of UAVs. Each team in turn is organized as two sub-teams: *attacker* and *reserve*. Initially, the *attacker* sub-team is empty; the *reserve* sub-team starts with the team of vehicles allocated to the sub-task. Execution starts with the first leg of the *sub-task*. The *reserve* vehicles execute a *follow_path* maneuver to the farthest safe point in this leg. When this point is reached one of the *reserve* UAVs is transferred to the role of *attacker* and the two sub-teams start two concurrent threads of execution until the *sub-task* terminates successfully or fails. The *attacker* UAV leads the execution: it executes the *sub-task specification* until successful termination, or until it is destroyed or runs out of bombs – in both cases the UAV is removed from the sub-team *attacker* and another UAV from *reserve* is tasked to replace it. The *reserve* sub-team follows the *attacker*: it moves to the farthest safe point of the *sub-task* legs terminated so far by the *attacker*; when this point is reached they execute a *hold* maneuver until this point is moved further as a result of the actions of the *attacker*.

This execution strategy has two important properties. First, only one UAV at a time is exposed to risk when attacking a target. This minimizes the total risk incurred by a team. Second, it builds reserve teams whose UAVs can be reassigned for other purposes if, depending on execution performance, the number of UAVs min_i^{sb} required to execute each sub-task i proves excessive.

4.4.2 Organization and control structures

The execution control framework accommodates different types of tasks of varying complexity. Given a task specification it automatically creates the initial structure of controllers, including the links or communication channels connecting them, and the information structures for task execution. It adapts the initial structure to changes in the world and to the execution requirements. It does this by creating and/or removing links and controllers while preserving structural and task invariants. Links play an important role in this framework. We use the *SHIFT* link concept. In *SHIFT* a *Link* variable is basically a pointer to another component. Links are unidirectional: if A communicates to B it has a link to B; B is required to have a link to A to communicate with it.

The components, organization, and evolution of the execution control framework are briefly described next.

Controllers. There is a task controller for each *Task*, one sub-task controller for each sub-task, one sub-team controller for each sub-team, and one controller for each UAV (described before).

Specifications. Controllers execute specifications. Specifications are separated from the control code for re-use and modularity. There is one specification type per type of controller.

Localization. The UAV controllers are non-mobile and reside onboard the UAV. The other controllers are mobile, i.e., they have a link to a physical location and we can change this location. The location of a mobile controller is part of its state.

Control structure. It is a ‘tree’-like graph of controllers: the nodes are the controllers and the edges are the links connecting them (see [35] for details). There are four layers in this ‘tree’, one layer for each type of controller – task, sub-task, sub-team and UAV respectively. The root node is the task controller. It is linked to the sub-task controllers. Each sub-task controller is linked to the *attacker* and *reserve* sub-team controllers.

Each sub-team controller is linked to the UAV controllers in the sub-team.

Creation and initialization. The initial control structure is built in successive steps, one per layer, starting at its root. Except for the UAV controllers, which come with the UAVs allocated to the task, all the other controllers are created in this process. Upon its creation, each controller is initialized and creates its dependants (with the exception of the sub-team controllers) and links to them. The process starts with the creation of the task controller. The task specification encodes the information required to create and maintain this control structure.

Adaptation. The 'tree' structure and the 4 layers are an invariant of the execution control framework. However, this structure may evolve with time. First, we can change the location of a mobile controller. This is done by changing the link to its location. Second, we can 're-create' mobile controllers to 'regenerate' the control structure. This may be the case when the physical location of a controller is destroyed; we create a new controller and re-establish its state and links. Third, we can add and/or delete mobile controllers upon initiation/completion of their specifications. This is the case when the *attacker* sub-team succeeds in removing the last target in the corresponding sub-task; this event signals the completion of the sub-task; and the sub-task and the sub-team controllers remove themselves from the control structure. Fourth, we can change the control dependencies for each UAV controller. To do this we 'move' the link to UAV controller from one sub-team controller to another sub-team controller and/or from one sub-task controller to another sub-task controller. We also use this last feature to 'reinforce' a team with additional UAVs and to transfer UAVs among *reserve* sub-teams.

Patterns of coordination. The task, sub-task and sub-team controllers follow the same patterns of coordination. This is because the control structure is organized as a tree. Each controller has in its state a few coordination variables and links to each of its dependants (controllers). The coordination variables describe the state of its dependants and the state of execution of the controller specification. The controller receives state updates from each dependant, updates the coordination variables, and commands its dependants accordingly. This allows for distributed decision making. The task controller coordinates leg dependencies and task failures. The sub-task controller coordinates the *reserve* and *attacker* sub-teams. The sub-team controllers coordinate the maneuvers of each UAV in the sub-team.

This coordination structure facilitates the addition of controller dependants at each layer. This is done by extending the coordination variables and the control logic, and by adding links to the new dependants.

This allows for more complex patterns of interaction and control. For example, the attack task can be easily extended to accommodate the coordination with Battle Damage Assessment (BDA) sub-teams. We do this with a few changes to the sub-task controller: extend the coordination variables to include the state of the BDA sub-team and the state of each target requiring BDA; extend the control logic; and create a BDA sub-team controller.

Space limitations preclude discussion of how this organization and control structures can be fully utilized in a complex military scenario.

In the reminder of this section we explain how this organization and control structure is implemented in *SHIFT* in the framework of dynamic networks of hybrid automata. For details see pages 48–71 of [105].

4.4.3 *SHIFT* implementation

Task. The *SHIFT* component *Task* encodes the *Task* data model (cf. section 4.2.3).

Task controller. The left box in figure 4.3 depicts the *SHIFT* data model for the *task_controller* component. The coordination variables are updated by the sub-task controllers *st*. The *fail* flag is set to *true* when one of the sub-tasks fails – in this implementation the task fails if one of the sub-tasks fails. The set *legs_done* is updated upon successful completion of a leg in a sub-task.

```
type task_controller
{
  input
    task t;

  state
    set(leg) task_legs_done:={};
    number fail;

  output
    set(sub_task_controller) st;
  ….
}
```

```
type sub_task_controller
{
  input
    subtask p;
    task_controller task_c;

  output
    set(ucav) reserve;
    number accept;
    set(leg) executed:={};

  state
    set(ucav) attackers;
    set(ucav) reserve_hold;
    set(vehicle_supervisor) vsa
    set(vehicle_supervisor) vsr;
    leg current_a_leg;
    leg current_r_leg;
    symbol attack_stage;
    symbol reserve_stage;
    symbol dependency_stage;
    set(leg) preceeding_legs:={};
  ...
}
```

Figure 4.3: Left is a skeleton of a task controller; right is a sub-task controller.

Sub-task controller. The right box of figure 4.3 depicts an excerpt of the *SHIFT* skeleton of the *sub-task controller* component. It has three discrete states: *execution, initialize, error*. The transition structure (not depicted in this figure) consists mostly of self-loops in the *execution* state, where normal execution takes place. The initial state is *initialize*. There is a transition from *initialize* to *execution* which is taken immediately after the creation of this controller. There are two actions on this transition: the *reserve* sub-team is initialized with the set of UAVs allocated to execute the sub-task specification *p*, and *attacker* is initialized with the empty set.

The coordination variables *attack_stage* and *reserve_stage*, which are updated by the sub-team controllers *rc* and *ac*, describe the execution state of the *current_a_leg* and *current_r_leg* legs for the *attacker* and *reserve* sub-teams respectively (cf. in sub-section 4.4.1 that both sub-teams execute the same sequence of legs with different execution policies). The states of the *attacker* sub-team are: *attack* (attack segment of a leg); *path* (safe path segment); and *hold* (holding pattern at the end of the leg). The states of the *reserve* sub-team are: *hold_end* (holding pattern at the end of the leg), *hold_path* (holding pattern at the end of the safe path), *path* (executing the safe path of the leg), and *path_attack* (executing the attack path of the leg).

The self-loops in the *execution* state model the coordination of the *attacker* and *reserve* sub-teams. Consider the following sequence of events to describe coordination in this controller. 1) the *attacker* UAV is destroyed; 2) the *sub-task controller* removes the corresponding supervisor from *vsa*, removes the UAV from *attacker*, and updates *current_a_leg* to the value of *current_r_leg*; 3) if *reserve* is empty the sub-task fails; if not the *sub-task controller* transfer one UAV from *reserve* to *attack*; the corresponding supervisor is also transferred from *vsr* to *vsa* – we use predicates on the state of the UAVs in each sub-team to access their vehicle supervisors. The *SHIFT* code for the actions described in 1) and 2) is

```
execution -> execution{vsa:destroyed(one:p)}
 do
 /* u(p) is the UAV whose supervisor is p */
 {
  vsa:= vsa - {p};                 // remove p from vsa
  attacker:= attacker-{u(p)};     // remove u(p) from attacker
  current_a_leg:=current_r_leg;   // reset current_a_leg
  attack_stage:= $path;           // reset attack_stage
  blue:=blue-{u(p)};              // remove UAV from Blue
 }
```

The transition from *execution* to itself takes place when there is one vehicle supervisor *p* in *vsa* signaling that the corresponding UAV was destroyed. The corresponding variables are updated in the *do* statement.

The *SHIFT* code for the actions described in 3) is

```
execution -> execution {} when (exists x in reserve: x = minel y in reserve:(x(p(x))*x(p(x))
             + y(p(x))*y(p(x)) + z(p(x))*z(p(x))) - d)
 do
   /* vs(x) is the supervisor of UAV x */
   {
    reserve:= reserve - {x}; // remove x from reserve
    vsa:= vsa + {vs(x)};      // add vs(x) to vsa
    vsr:= vsr - {vs(x)};      // remove vs(x) from vsr
    attacker:= attacker+{x}; // add x to attacker
   }
```

This transition takes place when there is at least one UAV in *reserve*. In this case *x*, the UAV that is closest to the target destination, is transferred to *attacker* and the corresponding variables updated in the *do* statement.

Attacker and reserve controllers. These controllers follow the patterns of organization described above to implement the control strategies described in sub-section 4.4.1. They have links to their *sub-task controller* and to the vehicle supervisors *vsa* and *vsr* for the *attacker* and *reserve* sub-teams respectively.

The following excerpt of *SHIFT* code models the case when the *task* fails and all the supervisors *vsr* are commanded to abort the current maneuver under execution.

```
 execution -> execution{vsr:abort(all)} when (fail(task_c)=true)
```

4.4.4 Simulation example

Figure 4.4 presents a *SHIFT* specification for the attack task sketched in Figure 4.5.

Figure 4.5 depicts a scenario from the Boeing Open Experimental Platform. The threats are located at the center of the red circles which indicate their range at a given altitude. The names of the targets are displayed in small caps. The attack task is composed of two sub-tasks *subtask1* and *subtask2*. *subtask1* is composed of legs 1-5 and *subtask2* is composed of legs 6-8. Each leg consists of a target and a path, which may be empty. There are 4 UAVs of type *small_combo* which are allocated to 2 teams, one per sub-task. The *SHIFT* code illustrates how to create a complete simulation. This is done in several steps (transitions of the hybrid automaton *task_simulation*) to respect the creation dependencies. The *task* specification *tarefa1* is created before the *task_controller ctarefa1*. The *task_controller* creates the control structure (cf. above) and the simulation starts. *subtask2* is executed by one UAV while the other one stays in *reserve*. *subtask2* is executed

```
type task_simulation
{
  output
    ucav u1, u2, u3, u4;
          leg leg1, leg2, leg3, leg4, leg5, leg6, leg7, leg8;
          subtask subtask1, subtask2;
          task_controller ctarefa1;
          task tarefa1;
          set(ucav) team1:={}, team2:={};

  state
    number t; // time

  flow default {t' =1;};

  discrete
    i0, i1, i2, i3, i4, normal;

  transition
    i0 -> i1 {} do                                                    // create ucavs with a control structure
          {
                    u1 := create(ucav, p:= small_combo_1);
                    u2 := create(ucav, p:= small_combo_2);
                    u3 := create(ucav, p:= small_combo_3);
                    u4 := create(ucav, p:= small_combo_4);
          },
    i1 -> i2 {} do                                                    // create all legs
         {
            leg1:= create(leg, path:= [ [[93517.725, 111320.00],
                                         [150000.00, 158235.2],
                                         [151000.00, 158000.00]],
                                         [[93517.725, 111320.00],
                                         [150000.00, 158235.2],
                                         [151000.00, 158000.00]]],
                                         vehicles:=[u1,u2], p:=medium_sam12);
            leg2:= create(leg, p_attack:= [], p:=long_sam5_trk);
            leg3:= create(leg, p_attack:= [ [230679.38, 135172.16],
                                           [ 231679.38, 135172.16 ]], p:=medium_sam13);
            leg4:= create(leg, p_attack:= [ [246924.42 , 134016.28],
                                           [ 246924.42, 151410.03]], p:=medium_sam15);
            leg5:= create(leg, p_attack:= [], p:=long_sam6_trk);
            leg6:= create(leg, path:= [ [[93517.725, 111320.00],
                                        [108000.00 , 240000.00]],
                                        [[93517.725, 111320.00],
                                        [108000.00 , 240000.00]]],
                                        vehicles:=[u3,u4], p:=long_sam8_trk);
            leg7:= create(leg, p_attack:= [], p:=long_sam7_trk);
            leg8:= create(leg, p_attack:= [], p:= medium_sam14);
            team1:={u1, u2};                                          // create teams to execute subtasks
            team2:={u3, u4};
         },
    i2 -> i3 {} when (t>1) do                                         // creates subtasks and leg dependencies
         {
           subtask1:= create(subtask, p:=[leg1, leg2, leg3, leg4, leg5], team:= team1);
           requires(leg3):={leg6};
           subtask2:= create(subtask, p:=[leg6, leg7, leg8], team:= team2);
         },
    i3 -> i4 {} do {tarefa1:= create(task, s:=[subtask1,subtask2]);},          // creates task
    i4 -> normal {} do {ctarefa1:= create(task_controller, t:=tarefa1);};      // creates task controller
}
```

Figure 4.4: *Shift* specification.

by the both UAVs since the first one runs out of bombs. When the task terminates with success the control structure removes itself from the simulation, and the UAVs are free to engage in other actions.

4.5 Conclusions

We presented a design for task planning and execution of UAV teams. The ‘design space’ is large and heterogeneous. We structure the space by first decomposing it into off-line planning and online execution control. A plan determines a sequence of sub-tasks, with precedence constraints, assigns a UAV team to each sub-task, and produces a nominal risk-minimizing path for each team. Execution control is organized as a

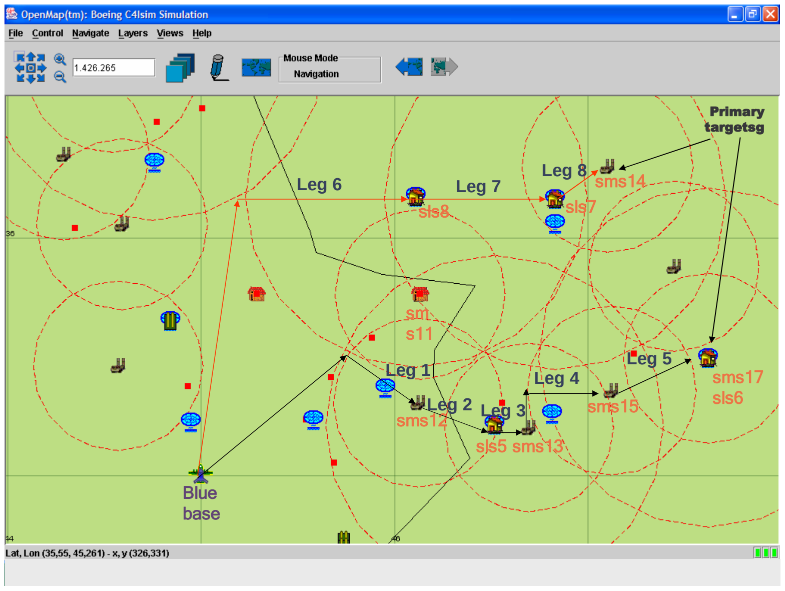


Figure 4.5: Attack task and scenario.

three level hierarchy of task controller, UAV supervisor, and maneuver controllers.

The controller presented in this chapter is an advance over current practice, in that it specifies in a hybrid automaton, both logical structure and the selection of continuous control variables. The specification takes advantage of the object-oriented nature of *SHIFT* and the abstract constructs that it provides. In particular: 1) specification of controllers is separated from their instantiation; 2) controllers are hierarchically organized; 3) structure of multi-vehicle task controllers is independent of the number of vehicles (because of the set construct in *SHIFT*); 4) user intervention is explicitly made available at all levels of the hierarchy in terms of a well-defined interface of command and response messages; and 5) controllers can be extended through specialization (because *Shift* allows inheritance).

Chapter 5

Optimal path coordination for a two-vehicle system

An optimal path coordination problem for a two-vehicle system is formulated in the framework of hybrid systems and solved using dynamic programming techniques. The problem consists of finding the optimal path for the first vehicle given that the path cost has a discontinuous dependence on the distance to the second vehicle. The second vehicle is fuel constrained and has to perform a closed path starting at its initial position.

5.1 Introduction

Problems of collaborative multi-vehicle control are posing new challenges to control. In some problems, cooperation concerns distributing similar vehicles over an area to optimize the coverage rate for surveillance missions. In other problems, heterogeneous vehicles with complementary capabilities can be used more advantageously when other forms of cooperation take place. One such example arises when planing operations of unmanned air vehicles (UAV) in hostile air spaces. The probability of survival of an UAV is directly proportional to the value of the path integral taken with respect to some risk function [34]; the level of risk is significantly reduced when the UAV flies under the protection of an UAV carrying a jamming device. This is an example of a collaborative control problem where vehicles interact to improve individual or group performance.

The interesting questions are:

1. How is optimal vehicle control related to optimal group control?

2. What is the value of cooperation?

These questions are better understood in the framework of dynamic programming (DP) [9]. DP approaches the problem of optimizing the behavior of a dynamic system with respect to some cost function by introducing a value function which gives, at each point of the state space, the optimal cost to go for the system. When the optimization problem is properly formulated (see [58] for details), the value function satisfies an equation which is derived from the Principle of Optimality which basically states that in an optimal sequence of decisions or choices, each subsequence must also be optimal.

Here I discuss research on DP for collaborative control problems with the help of a simple two-vehicle optimal path coordination control problem (see [42] for related work on DP for collaborative control). This problem is representative of more general optimal coordination problems.

Vehicle v_1 has to find the optimal trajectory from some initial location α to some destination γ. The instantaneous path cost for v_1 is reduced by a fixed amount l when the position of this vehicle "coincides" with the position of another vehicle, v_2; this means that the path cost for v_1 is a discontinuous function of the relative positions of the two vehicles. v_2 has a limited amount of fuel; it departs from $\beta \neq \alpha$ and is required to return to β before it runs out of fuel. The vehicles are allowed to met once and move together up to the point where v_2 has enough fuel to return to β.

The collaborative control problem for v_1 and v_2 is formulated as an optimal control problem for a hybrid automaton with three discrete states (the hybrid automaton models the combinatorial aspects of the problem) and find the structure of the solution using DP techniques. In this formulation, the state of the two-vehicle system has two components: a memoryless component, given by the continuous state, and a component with memory, given by the discrete state which describes the history of motions up to the current discrete state. This is because the system has to "remember" if the vehicles met at a given point, to prevent them from meeting again (as required). The jump sets are given by the set reachable by v_2 for a round trip from β (see [60] for details on dynamic optimization techniques for reachability analysis).

Surprisingly the problem is not solved in the product space of the state spaces for the the two vehicles. The problem is solved through the coordination of optimization problems for lower dimensional state spaces. This property is generalized to multi-vehicle rendezvous problems.

The motivation for this formulation comes, in part, from two problems of motion coordination discussed in [87] to illustrate the use Ordered Upwind Methods for solving optimal hybrid control problems. The first problem consists of finding an optimal trajectory on a surface, given that there are discrete transitions between a finite number of points on the continuous state-space. This problem can be interpreted as one of motion coordination between a person and a bus running between two or more bus stops: in some cases it may be better to take the bus. The directed discrete links change only the position in the continuous state space, but not the underlying dynamics. The problem is solved with the help of one value function defined on the continuous state-space. The second problem consists of finding an optimal trajectory for a person walking on a varied landscape and carrying a pair of inline roller skates. The person has the option to switch between walking and skating by paying a time penalty. This is modeled with two discrete states and two copies of the continuous-time state-space. The problem is solved with the help of a value function defined on the hybrid state-space.

The chapter is organized as follows. Section 5.2 provides background on dynamic optimization for hybrid systems. Section 5.3 presents the formulation of the path coordination problem in the framework of hybrid systems. Section 5.4 discusses the use of DP techniques to characterize the solution to the problem. Section 5.5 discusses the optimal strategies. An example is presented in in section 5.6 and the conclusions are drawn in section 5.7.

5.2 Background

The literature on DP for optimal hybrid control problems is briefly reviewed here.

A full-fledged hybrid system model, which subsumed previous models, was introduced by M. Branicky in [11]. The model includes autonomous and controlled jump sets and destination sets. Controlled jump sets model "lazy" transition systems in the sense that the controller can decide to jump or not to jump in these sets – this is the "lazy" transition semantics in the terminology of computer science. The transition maps associated to each jump may introduce discontinuities in state and time. The dimension of the continuous-time state space is allowed to change with the discrete state. Branicky introduces an optimal control problem over an infinite horizon with three terms discounted over time: running cost, transition cost and impulse cost. The transition maps and the cost functions are assumed to be bounded, uniformly continuous, and the vector fields associated to each discrete state are assumed to be bounded and uniformly Lipschitz in the state. The distances between autonomous and controlled jump sets (and also between autonomous jump and destination sets) are assumed to be strictly positive to prevent the occurrence of multiple transitions in zero time. The flow lines are assumed to be transversal to the boundaries of the autonomous and controlled jump sets, and the vector field is not allowed to vanish in these boundaries. This is required to prove continuity from the right of the value function for the optimal control problem. The consideration of DP techniques leads to a system of Quasi Variational Inequalities (QVI). No further analysis is carried out concerning the solution of the QVI. In [38], the value function is proved to be the "viscosity" solution to this system of QVI. The transversality assumptions lead to two modeling difficulties: 1) the state of the system is supposed to "freeze" during the time jump; however this is not possible at the boundary of the autonomous and controlled jump sets; and 2) when the state enters a controlled jump set it can only leave the set through a discrete transition, which was supposed to be optional (cf. [109]).

A set of QVI conditions similar to those presented in [11] is presented in [10]. The viscosity solution to the Hamilton-Jacobi-Bellman (HJB) is discussed. This is because under their assumptions the value function is continuous. The problem is that the value function for general hybrid control problems may be discontinuous (this is mainly due to the forced jumps, controlled jumps and discontinuous jump relations). This problem is studied in [109]. In this case, the value function is not continuous and the solution of the QVI is interpreted in the discontinuous viscosity setting.

A simplified version of the hybrid system model introduced by Branicky is presented in [89]. The keys simplification are: 1) the state is kept continuous at switching times; and 2) the dimension of the continuous-time state space is kept constant. There is a discrete transition map which defines, at each discrete state, the discrete states that can be reached in one discrete transition. The assumptions also include transversality conditions as in [11]. The author introduces a class of optimal control problems with terminal and running cost functions that depend on the discrete state; there are no switching costs. A set of necessary conditions in the form of a hybrid maximum principle are introduced. The corresponding value function is shown to be bounded and continuous. A HJB equation is derived with the help of the principle of optimality. The minimization in the HJB is taken over the continuous-time control settings and the discrete states. This is because the switching costs are zero. The HJB equation is used to establish a verification theorem for optimal control candidates, but there is no discussion on viscosity solutions. The discrete transition map is not taken into consideration as a constraint in the HJB minimization. This can only happen if all discrete states can be reached in a finite number of transitions. This important consideration is not stated as an assumption and it is not discussed in the paper.

5.3 Problem formulation

5.3.1 The system

Consider planar motion models (evolving in $\mathbb{R}^2$) for two vehicles $v_i, i = 1, 2$

$$\dot{x_i}(t) = f_i(x_i, u_i), u_i \in U_i, t \geq 0$$

$$x_1(0) = \alpha, x_2(0) = \beta$$

where u_i are the controls and U_i are closed sets.

Consider v_1. The cost of a path joining α and γ is

$$J_1(u_1(.), \gamma) = \int_0^{t_f} l(x_1, x_2) \cdot k_1(x_1, u_1) ds \tag{5.3.1}$$

where $k_1(., .) \geq 0$, $l : \mathbb{R}^2 \times \mathbb{R}^2 \to [0, 1]$ is a piecewise constant function ($l = c, 0 < c < 1$ if $x_1 = x_2$ and $l = 1$ otherwise) and t_f is the first time when $x_1(t_f) = \gamma$ under the control function $u_1(.)$. The function l models the fact that the path cost for v_1 is reduced when the positions of v_1 and v_2 coincide.

v_2 is fuel constrained. The model of fuel consumption is captured by an additional state variable $c_2 \in \mathbb{R}$ (indicating the amount of fuel in the fuel tank)

$$\dot{c_2}(t) = g_2(x_2, u_2) = \begin{cases} w_2(x_2, u_2) & \text{if } c_2 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$c_2(0) = \theta$$

where $w_2(., .) \leq 0$.

We introduce a second cost function J_2 to model the fuel remaining in v_2 when it reaches x at time t under the control $u_2(.)$

$$J_2(u_2(.), x) = c_2(t) \tag{5.3.2}$$

The standing assumptions are:

A1) $f_i, w_2 : \mathbb{R}^2 \times U_i \to \mathbb{R}^2$ are uniformly Lipschitz in x and uniformly continuous in the control variable. This condition ensures existence and uniqueness of solutions for the differential equations.

A2) There exist $K_1 < \infty$ and $1 \leq \varsigma_1 < \infty$ such that $\|l(x_1, x_2) \cdot k_1(x_1, u_1)\| \leq K_1(1 + \|(x_1, x_2)\|)^{\varsigma_1}$ for $(x_1, x_2) \in \mathbb{R}^2 \times \mathbb{R}^2, u_1 \in U_1$.

A3) There exist $K_2 < \infty$ and $1 \leq \varsigma_2 < \infty$ such that $\|g_2(x_2, u_2)\| \leq K_2(1 + \|x\|)^{\varsigma_2}$ for $x \in \mathbb{R}^2, u_2 \in U_2$.

This assumption and the previous are related to the existence of solution to the problem.

A4) $0 \in$ int $f_i(x_i, U_i)$. This means that each vehicle is locally controllable.

A5) $f_1(x, U_1) \subseteq f_1(x, U_2)$. This means that v_2 is capable of replicating the motions of v_1.

5.3.2 The case for coordination

The optimal path planning problem for v_1 when operating in isolation is ($l = 1$) is

Problem 5.3.1. *[Uncoordinated] Find*

$$\inf_{u_1(.)} J_1(u_1(.), \gamma) \tag{5.3.3}$$

The path planning problem becomes more interesting when the two vehicles are allowed to coordinate their motions. Two additional operational constraints are considered:

1. if v_2 leaves β, then it must return to β;

2. the vehicles are allowed to meet only once and then move together up to the point where v_2 returns to β (this precludes behaviors where the vehicles move together and separate repeatedly).

In what follows, and to simplify the analysis of the problem, assumption A6 is considered. This assumption means that the problem is symmetric in the terminology of [6].

A6) The fuel optimal paths for v_2 are also fuel optimal for the path traveled in the opposite direction.

Let R denote the set of point reachable by v_2 for a round trip from β under fuel budget θ. This is the set of points where the two vehicles can meet at one point. A characterization of R is in order. For this purpose consider the value function for the problem of minimizing the fuel consumption for vehicle v_2

$$V_2(x) = \max_{u_2(.)} J_2(u_2(.), x)$$

$$V_2(\beta) = \theta$$

Proposition 5.3.2. *Under the standing assumptions the value function V_2 is continuous in x.*

The proof is standard and we omit it.

Proposition 5.3.3. *R is a closed set given by*

$$R = \{x : V_2(x) \leq \frac{\theta}{2}\} \tag{5.3.4}$$

Proof. The expression for R follows from the consideration of Assumption A6. The fact that R is closed follows from the continuity of V_2. □

It may be worthwhile for v_1 to deviate from the optimal path for Problem 5.3.1 to join v_2 at a point in R, before reaching γ. The following example illustrates this point.

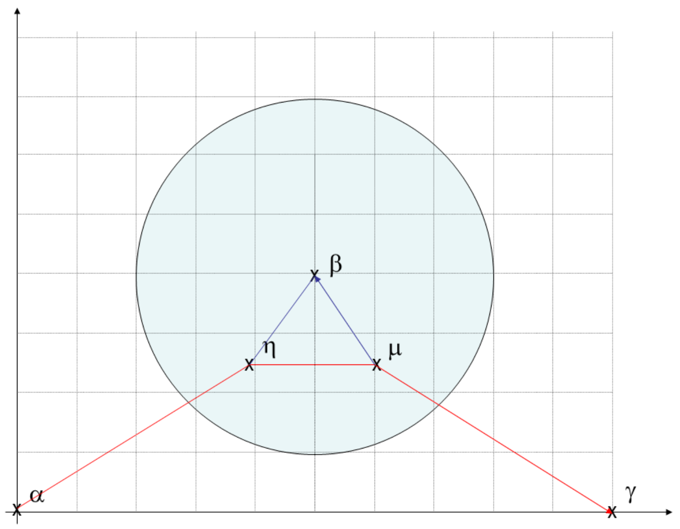


Figure 5.1: Example of coordinated paths.

Example 5.3.4. *Consider Figure 5.1. Let:*
$\dot{x}_i(t) \in B_0, i = 1, 2$ (B_0 is the closed unit ball in $\mathbb{R}^2$).
$\alpha = (0, 0), \beta = (50, 40), \gamma = (100, 0)$.
$\eta = (39.2000, 24.1254), \mu = (60.7999, 24.1254)$.
$c_2(0) = \theta = 12$.
$k_1(x_1, u_1) = 1, -w_2(x_2, u_2) = 0.2$, $l(x, x) = 0.1$.
R is the circle of radius 30 with center β (the optimal fuel cost of the round trip from β to the boundary of the circle is $60 \times 0.2 = 12 = \theta$). This is because this system satisfies the assumption A6: 1) the cost function does not depend on the direction of motion; and 2) the system dynamics are reversible. Observe that this is the set of points where the two vehicles can start to move together.

The fuel optimal paths for v_2 are straight lines. The same happens with the optimal paths for v_1 (for fixed values of l). This is because the we have simple dynamics and piecewise constant cost functions. The straight line joining α and γ is the optimal path for Problem 5.3.1; the optimal cost is 100. The cost of the path $(\alpha, \eta, \mu, \gamma)$, where v_1 deviates from the original optimal path to benefit from a cost reduction in the segment (η, μ), is 94.2182. v_2 complies with the constraints by taking a loop (triangle) from β, with fuel cost 12.0000 (within the fuel budget).

The structure of the solution is discussed next. Consider, for the sake of our discussion, that the optimal coordinated path for v_1 is $(\alpha, \eta, \mu, \gamma)$. Then the two path segments (α, η) and (μ, γ) are optimal with respect to the uncoordinated cost function. Otherwise we could pick other paths to connect these points with a lower cost. This is impossible since the path $(\alpha, \eta, \mu, \gamma)$ is optimal under our assumption. This means that up to the point η, the path optimization for v_1 is independent of what v_2 does. The same happens with v_2 for the path segments (β, η) and (μ, β). On the other hand, when the two vehicles meet at point η, the path optimization for both vehicles is no longer decoupled. Here, we need a third state variable to describe the evolution of the system. This is because the motions of the vehicles coincide, and because we need to keep track of the fuel consumption for v_2. This means that, from the perspective of v_1, all that really matters in what concerns v_2 is: 1) the point where the meeting takes place; and 2) the amount of the fuel remaining in the fuel tank of v_2. We observe that the amount of the fuel in v_2 at the meeting point should be optimal (otherwise this vehicle spent more fuel than what was needed to reach that point).

5.3.3 Hybrid model

The formulation of the coordinated optimal path planning problem for vehicle v_1 requires the consideration of a state variable that keeps track of what each vehicle does. We do this with a 3-state hybrid automaton. The hybrid state space is $S = \bigcup_{v \in \{a,b,c\}} (S_v \times v)$. v_1 evolves in $S_a = \mathbb{R}^2$ after departing from α. The positions of the two vehicles coincide in the discrete state b. We need an additional variable to keep track of the fuel consumption for v_2; this is why $S_b = \mathbb{R}^2 \times \mathbb{R}_0^+$. v_1 moves in $S_c = \mathbb{R}^2$ after taking the transition from discrete state b to discrete state c (after leaving v_2).

There is a controlled vector field f_v associated to each discrete state, where $f_a = f_c = f_1$ and $f_b = \{f_1, g_2\}$. The control constraints are $U_a = U_1, U_b = U_1 \times U_2$ and $U_c = U_1$. In the terminology of [11], associated to each discrete state v there are autonomous jump sets $A_{v,v'}$, controlled jump sets $C_{v,v'}$ and jump destination sets $D_{v,v'}$. The trajectory of the system jumps from S_v to $S_{v'}$ upon hitting the autonomous jump set $A_{v,v'}$; it may or may not leave S_v upon hitting the controlled jump set $C_{v,v'}$ and it can leave S_v at any point in $C_{v,v'}$; the destination of a jump is $D_{v,v'}$.

In what follows, x^i represents the $i - th$ component of x.

The autonomous and controlled jump sets for the system are respectively $A = \bigcup_{v,v'} A_{v,v'}$ and $C =$

$\bigcup_{v,v'} C_{v,v'}$. The jump set is $J = A \bigcup C$. These are given by

$$\begin{aligned} C_{a,b} &= R \\ A_{b,c} &= \{(x^1, x^2, x^3) : x^3 = V_2(x^1, x^2)\} \\ D_{a,b} &= \{(x^1, x^2, x^3) : x^3 \geq V_2(x^1, x^2)\} \\ D_{b,c} &= S_c \end{aligned}$$

with R given by equation 5.3.4. The transition maps are

$$\begin{aligned} G_{a,b} &: C_{a,b} \to D_{a,b}, G_{a,b}(x) = (x, \theta - V_2(x)) \\ G_{b,c} &: A_{b,c} \to D_{b,c}, G_{b,c}(x) = (x^1, x^2) \end{aligned}$$

The interpretation is as follows (see Figure 5.2). v_1 starts moving in S_a; if $x_1(.)$ enters $C_{a,b}$ then it may continue in S_a, or take a controlled jump to S_b. In the case of a controlled jump, the transition map $G_{a,b}$ maps the current state of v_1 to a state extended to include the optimal amount of fuel remaining in v_2 at the same location after departing from β with an initial amount of fuel θ. In S_b, the positions of the two vehicles coincide; there is an autonomous jump from S_b to S_c when the trajectory of the system hits $A_{b,c}$. This means that v_2 had to leave, since there was just enough fuel to go back to β. The jump relation consists of eliminating the third component of the state. The transition maps imply that v_2 uses fuel optimal strategies to travel to the meeting point and to reach β after leaving v_1. One could ask why is it necessary to include the discrete state c in the model (instead of having the autonomous jump from discrete state b to discrete state a). An autonomous transition from b to a could lead to trajectories in the controlled jump set $C_{a,b} = R \subset S_a$. But this jump can only be taken once. We need to keep track of the jump. We do this with the discrete state c.

With this transition system we model the fact that the state of the system has two components: a memoryless component given by the continuous state and a component with memory given by the discrete state which describes the trajectory of the system up to the current location. For example, if the discrete state is b this means that the two vehicles met at a given point and are moving together; if the discrete state is c this means that the vehicles moved together until the point where vehicle v_2 had to go back to β in an optimal fashion. This is why we need three discrete states.

In what follows we adopt the notation from [109]. Time is measured continuously with a real variable t in $[0, +\infty)$ and the state variable is (x, v). Trajectories are piecewise continuous in x and are normalized

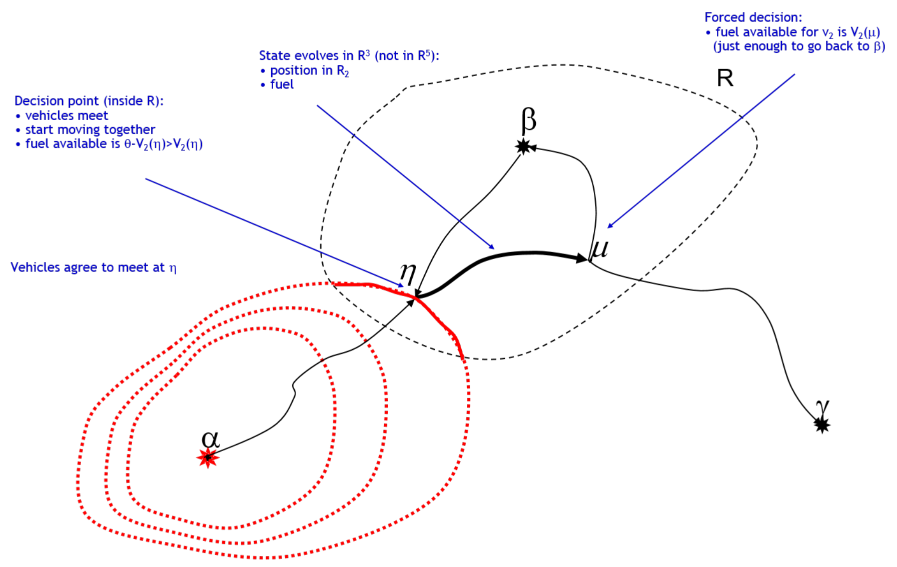

Forced decision:
• fuel available for v_2 is $V_2(\mu)$ (just enough to go back to β)
State evolves in R^3 (not in R^5):
• position in R_2
• fuel
R
Decision point (inside R):
• vehicles meet
• start moving together
• fuel available is θ-$V_2(\eta)$>$V_2(\eta)$
β
μ
Vehicles agree to meet at η
η
γ
α

Figure 5.2: Structure of the solution.

to be right-continuous. The hybrid control input is $I = (\{t_0, u_{v(0)}(.)\}\{t_i, u_{v(i)}\}_1^N), N \in \{0, 1, 2\}$, where $t_i \leq t_{i+1}(t_0 = 0)$ gives the sequence of times selected to switch the discrete dynamics. The activation of hybrid control input can only take place in the set C, or in the boundary of the set A. This spatial dependence translates to time dependence as follows.

Given (x, v) and $u(.)$, define the hitting times of A and J as

$$T^A(x, v, u(.)) = \inf\{t \geq 0 : (x(t), v) \in A\}$$
$$T^J(x, v, u(.)) = \inf\{t \geq 0 : (x(t), v) \in J\}$$

where $x(.)$ is the trajectory departing from (x, v) under the control function $u(.)$.

Definition 5.3.1. Given a hybrid state (x, v) a hybrid control I is called an admissible control with respect to (x, v) if:

- $0 = t_0, t_i \leq t_{i+1}$
- $T^J(x(t_i^+), v, u(.)) \leq t_{i+1} - t_i \leq T^A((t_i^+), v, u(.))$

This means that between discrete jumps the trajectory may evolve in J. Jumps may take place in C and must take place in ∂A (the boundary of A).

In our model $D_{a,b} \cap A_{b,c} \neq \emptyset$ and $D_{a,b}$ is not a closed set. This makes it possible for an instantaneous jump from discrete state a to c to occur: first as a controlled jump from a to b at the points in ∂R, and then as an autonomous jump to c. This problem can be solved by changing these sets to impose a strictly positive distance between them.

Let $I(x, v)$ denote an admissible control with respect to (x, v) and $\Lambda(x, v)$ denote the set of all admissible controls.

Proposition 5.3.5. *Given an initial hybrid state (x, v) the hybrid system possesses a unique hybrid execution.*

Proof. The proof follows standard arguments from [89]. □

5.3.4 Optimal control

Now consider the running cost maps $k_v : S_v \times U_v \to \Re^+$:

$$
\begin{aligned}
&k_a(x, u) = k_1(x, u) \\
&k_b(x, u) = \sigma l(x, x) k_1((x^1, x^2), u_1) - (1 - \sigma) g_2((x^1, x^2), u_2) \\
&k_c(x, u) = k_1(x, u)
\end{aligned}
$$

where $\sigma \in [0, 1]$. An explanation for the definition of k_b (and σ) is in order. The positions of the two vehicles coincide in the discrete state b. However, the minimization of the path cost for v_1 may not be compatible with the minimization of the fuel consumption for v_2. The problem is that v_2 is fuel constrained. The longer the fuel lasts, the longer v_1 benefits from the path coordination. We model this trade-off with $k_b(x, u)$ which is a convex combination of the two other cost functions.

Consider the coordinated path optimization problem for v_1. The cost of a path joining (α, a) and (γ, v) is

$$
\begin{gathered}
\tilde{J}_1((I(\alpha, a), (\gamma, v), \sigma) = \\
\sum_{i=0}^{N} \int_{t_i}^{t_{i+1}} k_{v(i)}(x(s), u_{v(i)}(s)) ds
\end{gathered} \tag{5.3.5}
$$

where $N \leq 2, t_{N+1} = t_f$ and $x(t_f) = \gamma$.

We introduce the explicit dependence on σ to remind us that the optimal solution depends on this parameter.

Problem 5.3.6. *[Coordinated] Find*

$$
\inf_{I(\alpha,a) \in \Lambda(\alpha,a)} \tilde{J}_1(I(\alpha, a), (\gamma, v), \sigma) \tag{5.3.6}
$$

Let T denote the set of points reachable by v_2 in S_b under the fuel constraint θ for a round-trip from β. T is the set of all $(x^1, x^2, x^3) \in S_b$ such that the first two components (x^1, x^2) are in R and the last component (x^3) satisfies the fuel constraint:

$$\begin{aligned} T = \{x \in S_b : (x^1, x^2) \in R \wedge (x^3 \geq V_2(x^1, x^2)) \wedge \\ ((\theta - V_2(x^1, x^2)) \geq x^3)\} \end{aligned}$$

Remark 5.3.1. $M = \{S_b \backslash T, b\}$ is not reachable in S.

5.4 Dynamic programming

In the spirit of DP we embed Problem 5.3.6 in a family of optimization problems where the final position varies. Introduce the value function

$$\begin{aligned} V(x, v, \sigma) &= \inf_{I(\alpha,a) \in \Lambda(\alpha,a)} \tilde{J}_1(I(\alpha, a), (x, v), \sigma) \\ V(\alpha, a, \sigma) &= 0 \end{aligned}$$

where $\forall x \in (S_b \backslash T) : V(x, b, \sigma) = +\infty$.

The fact that not all points in S_b are reachable under the constraints imposed on v_2 leads to this extended-valued value function.

In what follows we drop the explicit dependence of V on σ to simplify the notation.

The following theorem, presented without proof, states two important properties of the value function.

Theorem 5.4.1. *The value function $V(x, v)$ is bounded and continuous in $S \backslash M$.*

The following theorems can be proved with the help of the results from [109].

Theorem 5.4.2. *The value function $V(x, v)$ satisfies the principle of optimality for every $v \in \{a, b, c\}$.*

Theorem 5.4.3. *The value function $V(x, v)$ is the viscosity solution of the HJB equation.*

$$\begin{aligned} V_t(x, v) + \inf_{u \in U} [V_x(x, v) \cdot f_v(x, u) - k_v(x, u)] = 0 \\ V(\alpha, a) = 0 \end{aligned}$$

5.5 Optimal strategies

The optimal strategy for v_1 is derived from the value function $V(x, v)$. This requires some additional computations.

The position of v_1 is given by the continuous state of the hybrid automaton in the discrete states a and c, and by the first two components of the continuous state in the discrete state b; the third component, x^3, is the fuel remaining in v_2. However, the value function V in b depends not only on the position of v_1 (x^1, x^2), but also on the fuel remaining in v_2 (x^3). An additional minimization over x^3 is required. This is done next with the help of a new function, $\tilde{V} : \mathbb{R}^2 \to \mathbb{R}$.

$$\begin{aligned}
\tilde{V}(x,a) &= V(x,a) \\
\tilde{V}(x,b) &= \min_{x^3 \in [V_2(x), \theta - V_2(x)]} V((x,x^3),b) \\
\tilde{V}(x,c) &= V(x,c)
\end{aligned}$$

$\tilde{V}(x,a)$ is also the optimal value function for Problem 5.3.1.

Keep in mind that the discrete state keeps the history of the system. So v_1 can reach a same position in the three discrete states. To find the optimal path cost at $x \in \mathbb{R}^2$ we need to drop the dependence of $\tilde{V}$ on the discrete state with another minimization. This is done with the the help of a new function, $\overline{V}(x) : \mathbb{R}^2 \to \mathbb{R}$.

$$\overline{V}(x) = \min_{v \in \{a,b,c\}} \tilde{V}(x,v) \tag{5.5.1}$$

The optimal discrete state at x is given by

$$v^* = argmin_{v \in \{a,b,c\}} \tilde{V}(x,v) \tag{5.5.2}$$

Observe that v^* is not necessarily a singleton. We summarize these observations in the theorem.

Theorem 5.5.1. *$\overline{V}(\gamma)$ is the optimal value for solving Problem 5.3.6. If $v^* = a$ then path coordination is not optimal.*

The optimal control is given by u^* as follows

$$\begin{aligned}
u^* = argmin_{u \in U} \qquad & V_t(x,v) + \\
& [V_x(x,v) \cdot f_v(x,u) - k_v(x,u)]
\end{aligned} \tag{5.5.3}$$

Both the dynamics and the cost function do not depend directly on time. This simplifies the coordination of the optimal paths for the case when path coordination is the optimal solution: the vehicles are required to meet at the point where the two paths intersect for the first time.

We now study the conditions under which the solutions to Problems 5.3.1 and 5.3.6 differ. These are aimed at simplifying the process of finding numerical solutions to the coordinated problem.

Proposition 5.5.2. *Let $\Upsilon = V(\gamma, a)$ and $Q = \{x \in S_a : V(x, a) \leq \Upsilon\}$. If $Q \cap R = \emptyset$, then the solutions of Problems 5.3.1 and 5.3.6 coincide.*

*P*roof. The condition $Q \cap R = \emptyset$ means that γ can be reached with cost budget less than the one required to reach the set R, where coordination is possible. $\square$

Proposition 5.5.3. *The optimal cost for Problem 5.3.6 is l times the optimal cost for Problem 5.3.1 when there exists a trajectory $x_2(.)$ leaving β passing through α and γ and returning to β such that: 1) $x_2(.)$ satisfies the fuel constraint θ; and 2) the segment of $x_2(.)$ joining α and γ coincides with the optimal path for Problem 5.3.1.*

Proof. Consider first that v_2 is not fuel constrained. Then, the trajectories of v_2 can be made to coincide with the trajectories of v_1 along the path for v_1. This means that: 1) there exists a path as the one in the statement of the proposition; and 2) that v_1 benefits from a constant cost reduction along its path. Now consider the case when v_2 is fuel constrained. If there is a path satisfying the conditions of the proposition, the optimal cost for v_1 cannot be further reduced from the optimal level obtained without fuel constraints.$\square$

5.6 Numerical example

Consider Example 5.3.4 again. The computation of the value function becomes easier because of the simplicity of the considered cost function (piecewise constant over the state and input spaces, and time-invariant) and system dynamics. A numerical algorithm was especially tailored to take in account those specific assumptions. The value function is computed over a equally spaced grid.

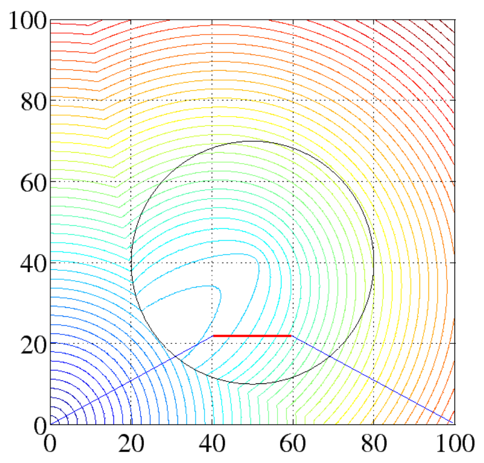
The computation of the value function is done in three stages. In the first stage, the system is in the discrete state a. Since the running cost, $k_1(x, u) = 1$, is independent of the input and vehicle's position, the optimal trajectory from the initial position to any position x is a straight line, traveled at unit speed (the maximum speed). It is trivial to note that $V(x, a) = \|x\|_2$. Therefore the exact value of $V(x, a)$ is known at the grid points.

In the second stage, the algorithm considers only the points in T (i.e., the set of all points that can be reached by v_2 while allowing the return to its initial position β within fuel budget). This is the closure of the circle shown in Fig. 5.1 (see also Eq. 5.3.4). Remember that those whose position laying outside R will have an infinite cost, due to the fuel constraint of v_2. For each point in T, the algorithm computes the cost to every other point (in $\mathbb{R}^3$) that can be reached respecting the fuel constraint, and updates $V(x, b)$ accordingly. The computation of $\tilde{V}(x, b)$ is straightforward.

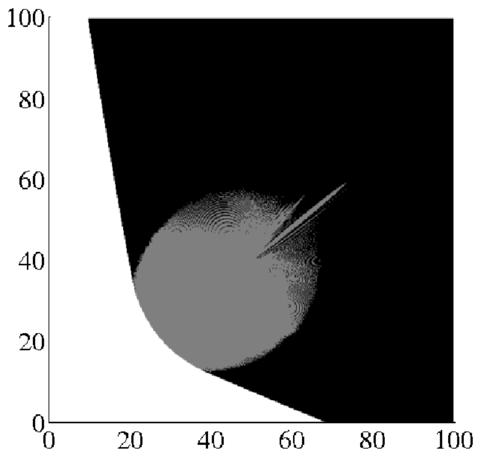
Another version of this algorithm, which computes directly $\tilde{V}(x, b)$ performing all computations on a two dimensional grid (therefore demanding smaller computation time), was also implemented. This version considers only the points in R: for each point in R, it computes the cost to every other point that can be reached respecting the fuel constraint, and updates $\tilde{V}(x, b)$ accordingly. However, this version does not allow the determination of the optimal trajectory using Eq. 5.5.3.

Finally, in the third stage the algorithm starts from the positions where $\tilde{V}(x, b)$ is finite, computed on the previous stage, and propagates the value function. In this final stage, $\overline{V}$ is computed as defined in Eq. 5.5.1.

The level sets of $\overline{V}$ are plotted on Fig. 5.3(a) along with the optimal trajectory from $\alpha = (0, 0)$ to $\gamma = (100, 0)$. The circle of radius 30 centered at $(50, 40)$ delimits R. Fig. 5.3(b) identifies two distinct regions of the x-y plane: in white, the final destinations for which the optimal strategy is the uncoordinated motion (no collaborative operation of v_1 with v_2); in black, the final destinations for which v_1 will benefit from coordinated motion with v_2, i.e., the set of points x such that $V(x, b) < V(x, a)$ and $V(x, c) < V(x, a)$.



(a) Level sets of $\overline{V}$ and optimal trajectory of v_1 for Example 5.3.4.



(b) Destinations in the black region benefit from coordinated motion.

Figure 5.3: Optimal strategies.

5.7 Conclusions

We have formulated and solved a path coordination problem to illustrate the use of DP techniques in collaborative control problems. The problem consists of minimizing the path cost for v_1 when this cost is a discontinuous function of the relative positions of the two vehicles and v_2 is required to return to its starting point. The problem is formulated as an optimal hybrid control problem. The state has a memoryless component and a component with memory. The autonomous and controlled jump sets are both given by the set reachable by v_2 when departing from β under the given fuel constraints. The optimal strategies for both vehicles are derived from value function, which depends on the location of v_1 and on the discrete state. The optimal path cost for v_1 at a given location is given by two sequential minimizations of the value function for the optimal hybrid control problem. Transitions in the hybrid automaton take place when collaboration is the optimal solution. The transition to the second state is taken by v_1 under the assumption that it meets v_2 and that v_2 followed a fuel-optimal path. This is a non-standard hybrid control problem: the jump sets are given by reach sets; and the value function for the coordinated problem assumes compatible optimal behavior by v_2 (this is given by a different value function for v_2).

Chapter 6

Verified control architecture

A layered control architecture for executing multi-vehicle team coordination algorithms is presented along with the formal specifications for team behavior. The control architecture has three layers: team control, vehicle supervision and maneuver control. The implementation is proved to satisfy the specification for the team behavior. This is done in the framework of automata theory. The implementation and specification are bi-similar. Computer simulations with accurate models of autonomous underwater vehicles illustrate the overall approach in the coordinated search for the minimum of a scalar field. The coordinated search is based on the simplex optimization algorithm.

6.1 Introduction

The last decade has witnessed unprecedented interactions between technological developments in computing, communications and control which have led to the design and implementation of robotic systems consisting of networked vehicles and sensors. These developments enable researchers and engineers not only to design new robotic systems but also to develop visions for systems that could have not been imagined before.

6.1.1 Multi-vehicle operations

Today, there are automotive vehicles in various stages of automation ranging from automated highway systems [103, 50], to coordinated adaptive cruise control systems [1], to "platooning" of passenger and military vehicles. Other examples for ground vehicles include border patrol, search and rescue, and games such as robotic soccer [106, 23] or the RobotFlag [24]. There are numerous applications for autonomous underwater vehicles, such as oceanographic surveys [95, 108, 56], operations in hazardous environments, inspection of underwater structures, mine search [48], and the Autonomous Ocean Sampling Network [21, 22], to name just a few. The Mobile Offshore Base illustrates the problem of coordinating the motions of sea-going vehicles [83, 27]. The application pull for the coordination and control of teams of unmanned air vehicles is driven mainly by military requirements [17]; some technologies have already been field tested [7, 94, 54] while others are being developed and tested in simulation [34]. A promising technological push comes from the inter-operation of multi-vehicle systems and sensor networks [20].

6.1.2 Approach and contributions

In this chapter we present a control architecture for the implementation of coordination strategies by a team of autonomous vehicles. These strategies are characterized by the alternation between two phases: a communication phase where the team exchanges messages to assign waypoints for each vehicle; and a motion phase where the vehicles move to the designated waypoints, where a new communication phase will take place. The strategy specification is encoded as an automaton.

Several difficulties must be faced in developing a control architecture for the implementation of this class of coordination strategies. We illustrate these difficulties and discuss our contributions in the context of the coordinated search for the minimum of a scalar field by a team of autonomous underwater vehicles with limited communication capabilities. The coordination strategy is inspired by a class of optimization algorithms with phased operations: each phase starts with the selection of points to sample and terminates when these points are sampled.

First, there are severe limitations on communications. For example, autonomous underwater vehicles use acoustic communications which pose significant restrictions on range and bandwidth [92, 57]. This precludes the use of communications for low-level feedback control. We address this difficulty by restricting communications to the exchange of a few coordination messages.

The second difficulty is in that the design space of the team search is large and heterogeneous. The design involves generating sampling points and arrival times to ensure communications at the end of each phase; assigning vehicles to the sampling points; and designing real-time feedback strategies for each vehicle. We address this difficulty by structuring the design into two pieces: generation of sampling points and execution control. We present conditions for the generation of sampling points and arrival times with the required properties; this is done in the setting of dynamic optimization and reach set computations. We introduce a layered design for the execution control. This is done in the framework of hybrid automata: there is a team controller, a vehicle supervisor and several maneuver controllers per vehicle. The coordination strategy is implemented through the interactions of the team controllers during the coordination phase. In this phase, one team controller, the master controller, receives the samples sent by the other team controllers, calculates the sampling points and arrival times for the next motion phase and sends them to the other team controllers. The motion phase is executed independently by each vehicle.

The third difficulty originates in the requirement that the execution control must indeed implement the search strategy. We addressed this difficulty by layering the execution control and designing each layer to ensure that their controllers produce guaranteed results under the assumption that the controllers at the adjacent layers also produce guaranteed results. This is done in a modular fashion. The vehicle supervisor and the maneuver controllers guarantee that each sampling point is visited within a given tolerance of the arrival time. Under these assumptions the composition of the team controllers is shown to implement the specification. This is done using automata-based techniques.

Our contributions concern the design of a modular architecture and the proof that the modules and the interactions within the architecture implement a given specification. This is done in the framework of automata-theoretic techniques and reach set analysis.

Summarizing, our design touches upon several related problems: finding the minimizer of a scalar field through the coordinated motions of multiple vehicles; guaranteed maneuver design; waypoint based coordination schemes, and control architectures. Next, we briefly compare our approach to related work on these problems.

6.1.3 Related work

The problem of finding the minimum of a scalar field with the coordinated motions of autonomous vehicles with sampling capabilities has received large attention in the last decade. A significant body of this work concerns the adaptation of optimization algorithms to single- or multi-vehicle search strategies. Search strategies for single vehicle operations inspired by different optimization algorithms are reported in [12] along with illustrative examples. Pure gradient-based methods for scenarios where a vehicle platoon searches the minimum of general convex and smooth scalar fields are presented in [3]. Lyapunov-based arguments are used in [3, 41] for the gradient descent of a scalar field. These approaches result in feedback control laws that require closing the control loop around communicated measurements. We take the view of considering limited and sporadic communications, which preclude the use of these techniques. This view is exercised in a distributed minimum search application. A team of vehicles is tasked to implement an optimization algorithm for this purpose. We formalize a complete control design under communication constraints which is implementable in a distributed fashion with guaranteed properties. We do this in the framework of dynamic networks of hybrid automata.

The problem of guaranteed maneuver design with logic switching is a difficult one, and has received significant attention from researchers in hybrid systems. Techniques from optimal control and game theory are used in [68] and [96] to design controllers for safety specifications in hybrid systems. Their methodology consists of three phases. First, they translate safety specifications into restrictions on the set of reachable sets. Second, they formulate a differential game and derive Hamilton-Jacobi-Bellman equations whose solutions describe the boundaries of reachable sets. Third, they synthesize the hybrid controller from these equations. The controller assumes the form of a feedback control law for the continuous and discrete variables, which guarantees that the hybrid system remains in the safe subset of the reachable set. This formulation is strongly related to the problem of reach set computation. Several techniques for reachability analysis of dynamic systems have been proposed. An approach for reach set computation for linear systems based on the Pontryagin maximum principle of optimal control theory and the separation property is presented in [100]. Dynamic programming techniques are used in [60] to describe reach sets and related problems of forward and backward reachability; extensions to the problem of reach set computation under adversarial behavior are also accommodated in this setting. These problems are formulated as optimization problems that are solved through the Hamilton-Jacobi-Bellman equations. The reach sets are the level sets of the value function solutions to these equations.

Quite a number of motion coordination problems proposed in the literature are captured by event-based way-point generation algorithms. They include consensus problems [53, 19, 55], pursuit–evasion games [51, 107], multi-robot tracking problems [70] and multi-vehicle search missions [90, 29, 30].

A vast majority of multi-vehicle systems are organized into hierarchical control architectures. For a comprehensive review of the issues concerning coordination and control of multiple vehicles consult [45]. The fact of the matter is that the control of every large-scale system is organized in a distributed hierarchy [101]. This way, a complex design problem is partitioned into a number of more manageable sub-problems that are addressed in separate layers. The problem is that different layers may be described within different theories making it difficult, if not impossible, to do a formal analysis of the control architecture. This is problem of one-world semantics [101]: properties of high level abstractions are translated into properties of lower level behaviors. However, hierarchical controllers are not designed that way. Typically, the design of a large system is broken into controllers. The design of each controller is evaluated in a mathematical world in which alternate controller designs can be compared. The mathematical world for one controller makes implicit assumptions about the behavior of lower-layer controllers. This is multi-world semantics [101]. We take this approach in our design.

There is a substantial body of work on the formalization of control architectures. Examples include the use of Petri nets and stochastic hybrid automata [86, 67], hybrid systems [98, 103, 99, 47], and linear temporal logic [40]. Our work is related to the layering concepts presented in [99]. The ideas used in execution control are inspired by [99, 98, 27, 32]. Here we formalize the components and interactions and introduce a layered analysis framework where we use automata theoretic concepts and dynamic optimization techniques in our proof techniques.

The ideas used in this chapter are inspired by [90, 30], where search missions for underwater autonomous vehicles (AUVs) have been studied. The algorithm for the generation of sampling points is executed at the end of each search phase. The algorithm generates both the sampling points and the time intervals for the corresponding arrival times. In this design, the algorithm runs on a AUV designated as the master. At the end of the search phase, the AUVs in the team send the observations at the sampling points to the master; the master runs the algorithm upon the reception of the last observation and sends the assigned sampling points to the AUVs in the team. Two communication exchanges that take place during each phase. For these communications to happen the sampling points and the corresponding time intervals must satisfy two properties: (1) the sampling points are reachable within these time intervals; and (2) the relative distances among the AUVs during these time intervals satisfy the communication constraints. The execution control is decomposed into a hierarchy of team controllers, vehicle supervisors, and elemental maneuver feedback controllers. The controllers are described as interacting distributed hybrid automata [104]. Interactions occurs through the exchange of messages among the controllers.

6.1.4 Outline

The chapter is organized as follows. In Section 6.2 we introduce the problem formulation. In particular we highlight the constraints and assumptions under which the control architecture is developed. Moreover we define the system specification, namely a mathematical description of the overall system behavior, which is used in the verification of the architecture. Section 6.3 describes the hierarchical control structure in the framework of interacting hybrid automata. The main results are reported in Section 6.4 where properties of the hierarchical control structure are discussed and it is shown that such architecture implements the given system specification. In Section 6.5 we present simulation results to illustrate the implementation of our design in a team search mission for a team of underwater vehicles. Finally, the conclusions and future developments are discussed in Section 6.6.

6.2 Problem formulation

Let us consider a set $V = \{v_1, v_2, \ldots, v_N\}$ of $N \geq 1$ vehicles. Each vehicle v_i is modeled as a nonlinear control system

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t)),$$

where $x_i(t) \in \mathcal{X} \subset \mathbb{R}^n$ is the state of the vehicle, $u_i(t) \in \mathcal{U} \subset \mathbb{R}^m$ the control, and $f_i : \mathcal{X} \times \mathcal{U} \to T\mathcal{X}$ the vector field.

6.2.1 Team coordination via waypoint generation

We assume that the team is coordinated by an event-based controller that generates *waypoints*, namely a point $w = (w_1, \ldots, w_N) \in \mathcal{W} \subseteq \mathcal{X}^N$. The team coordination is defined by the following update map

$$(w^+, t^+) = \phi(w, t, e),$$

where $e = (e_1, \ldots, e_N) \in \Sigma^N$ is a vector of events, each of which is defined on an event alphabet Σ, $t = \{t_1, t_2, t_3\} \in \mathcal{T} \subset \mathbb{R}^3$ is a set of coordination times which are defined in the following section, and $^+$ is used to distinguish the most recent value of a variable. We call $\phi(.)$ the team coordination strategy. The controller for each vehicle takes as inputs w_i^+ and t^+.

6.2.2 Vehicle model

Our approach encompasses general vehicle models, as we will infer from the developments in the following sections. However, in the reminder of the paper we consider unicycle vehicle models. This is because many vehicles used in robotics can be precisely or approximately described by a unicycle model together with extra kinematic constraints. We then have that each vehicle is described by the following differential equations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix}, \tag{6.2.1}$$

where v is the linear forward velocity, ψ is the orientation of the vehicle and ω is the angular velocity.

The synchro drive vehicle can be precisely described by the previous kinematic model. In this type of vehicle, indeed, the linear and angular velocities can be controlled independently and are the same for all wheels. Differential drive vehicles, where the locomotion system is comprised by two parallel driving wheels that can be controlled independently, are described by a unicycle model if we impose that $v = (v_1 + v_2)/2$ and $\omega = (v_1 - v_2)/\ell$, where v_1 and v_2 are the right and left wheel speeds and ℓ is the distance between the driving. Notice the kinematic constraint between angular and linear speed. Tricycle and car-like vehicles where only the front wheel (or wheels) is (are) actuated, can be modeled by the previous kinematic model. In this case if α is the angle of the turning wheel with respect to the heading of the vehicle, then $v = v_s \cos\alpha$ and $\omega = v_s/d \sin\alpha$ where v_s is the linear velocity of the steering wheel and d is the distance between passive axle and the steering wheel [63, 77, 8, 64]. Also underwater vehicles (and similarly aerial vehicles) that move on a plane can be very well approximate with the unicycle model. For this type of vehicles the extra kinematic constraints impose that $v_{\min} > 0$, that is the vehicle requires a minimum velocity ("stall" velocity) to maintain controllability, and the angular velocity depends on the linear velocity $\omega = c\,v$ where c is a constant related to the maximum curvature of the trajectory that the vehicle can follow.

In the following we will also consider the case of external slowly-varying disturbances acting on the vehicles. This is the case of water streams for underwater vehicles. We then have the following modified dynamic equations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix} + v_d \begin{bmatrix} \cos\psi_d \\ \sin\psi_d \\ 0 \end{bmatrix}.$$

where v_d and ψ_d is the velocity and the direction, respectively, of the disturbance acting on the vehicle.

6.2.3 System specification

We introduce a formal specification to prescribe the behavior for the multi-vehicle system. This includes a model of the interactions between communication and control. Models of communication constraints, including the ordering of messages, are not considered in some control designs for multi-vehicle systems proposed in literature (see for example [51, 107, 53, 19, 55]).

We model the specifications for the system in the framework of a transition system.

Definition 6.2.1 (Transition system [80])**.** A transition system T is a tuple

$$T = (Q, \rightarrow, I, O, \mathrm{I}nit, \mathrm{F}inal),$$

where

- Q is the set of states
- I and O is the set of inputs and outputs, respectively
- $\rightarrow \subset Q \times I \times Q \times O$ is the transition relation
- $\mathrm{I}nit \in Q$ is the initial state
- $\mathrm{F}inal \in Q$ is the final state

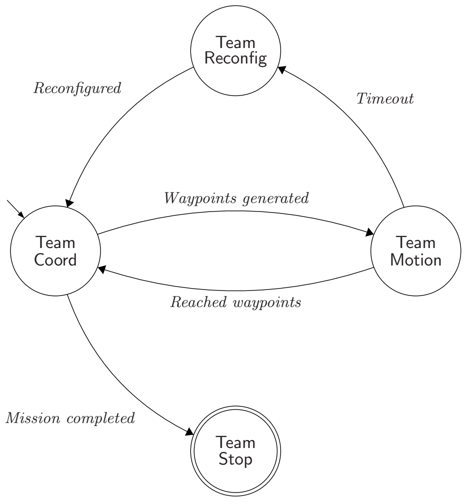


Figure 6.1: System specifications for team coordination.

The interpretation is that an input $i \in I$ cause the system to move from one state $q \in Q$ to another state $q' \in Q$ producing the output $o \in O$. It is convenient to write $q \overset{i/o}{\rightarrow} q'$ instead of $(q, i, q', o) \in \rightarrow$. The graphical representation of T is a directed graph with vertices representing Q and arcs representing $\overset{i/o}{\rightarrow}$, an arc with empty origin representing $\mathrm{I}nit$ and a vertex with an extra circle representing $\mathrm{F}inal$.

The system specification for a coordinated search mission is given by the transition system

$$T_{\text{Spec}} = (Q_{\text{Spec}}, \rightarrow, I_{\text{Spec}}, \emptyset, \text{Team Coord}, \text{Team Stop})$$

shown in Figure 6.1. It has four discrete states: Team Coord, Team Reconfig, Team Motion, Team Stop. In a nominal mission the system alternates between two states, Team Coord and Team Motion, until the mission is completed when a termination condition is satisfied. Note that this system specification is fairly general, and captures a wide class of multi-vehicle control problems.

The system starts in the Team Coord state. A transition to Team Stop takes place if the termination condition is true. Otherwise, in Team Coord the vehicles exchange their positions and sampled data prior to the generation of the new waypoints w^+ and coordination times t^+. The transition to Team Motion takes place upon the reception of w_i^+. While in Team Motion, each vehicle is controlled to the designated waypoint within a given coordination time interval. The transition to Team Coord takes place when all the vehicles reach their designated waypoints. If one vehicle is not able to reach its waypoint within a given coordination time interval a timeout event is generated and the transition to Team Reconfig is taken. In Team Reconfig the team executes a reconfiguration operation, which involves a re-allocation of roles. After reconfiguration, the system goes to Team Coord, where nominal execution is resumed for the currently active vehicles. The transition to Team Stop takes place when the mission is completed.

In the next section we present our design for the hierarchical control architecture and in Section 6.4 we show that the design satisfies the specification.

6.3 Hierarchical control structure

6.3.1 Organization and concepts of operation

The vehicles in V have the same control structure. Our design for the vehicle control structure is organized into two pieces: generation of sampling points and execution control. The execution control, in turn, is structured into three layers: team control, vehicle supervision and maneuver control (see Figure 6.2). This is an intuitive structure for program developers and system operators.

The team control architecture is depicted in Figure 6.3 as the composition of the control structures for each vehicle, where one of the vehicles is configured as the *master* and the others as *slaves*. The composition of the team controllers encodes the team control logic. The composition of the vehicle supervisor and of the maneuver controllers encodes the motion control logic for each vehicle. The concepts of operation behind the team control architecture are described now.

We assign roles to vehicles in the team control architecture. This amounts to configuring their control

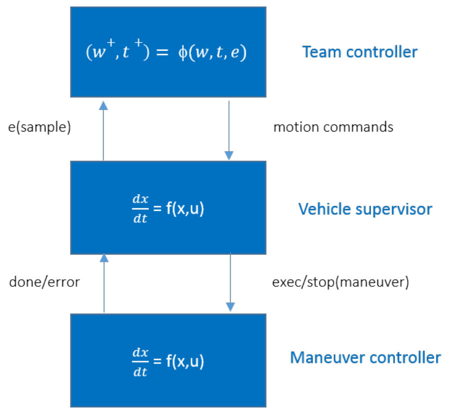


Figure 6.2: Hierarchical control structure for each vehicle.

structures differently. The configuration is done at the team controller layer: one team controller is configured as the *master* and the others as *slaves*. Communication exchanges in the team are restricted to interactions between the team controllers. These take place during the coordination phase. The pattern of interactions is as follows: the *master* team controller executes the procedure for the generation of sampling points and communicates the sampling points together with the coordination times to the other team controllers; upon arrival at the designated sampling point each team controller sends the a message with the sample to the *master*; the process starts again when the *master* receives the samples from the other team controllers and the termination condition is not true. In this design there is no need the vehicles to communicate during the time elapsed between the reception of the next sampling point and the arrival at the sampling point.

From the motion control point of view, each vehicle is abstracted as a provider of prototypical maneuvers: different maneuvers may be required for different missions; and the same motions may be accomplished by different maneuvers. There is one maneuver controller for each type of maneuver.

Consider figure 6.2 for a description of the motion control logic for each vehicle. The vehicle supervisor mediates the interactions between the team controller and the maneuver controllers. This is done for the purpose of modularity; there is a library of maneuvers and of maneuver controllers; and the addition and deletion of maneuvers to the library does not require changes to the team controller and to the vehicle supervisor. The vehicle supervisor accepts maneuver commands (or commands to abort the current maneuver) from the team controller and passes the maneuver parameters to the corresponding maneuver controller for execution, and signals back to the team controller the completion or failure of the maneuver. The maneuver controller takes as input a maneuver specification, sends low-level control commands to the actuators in continuous time, and

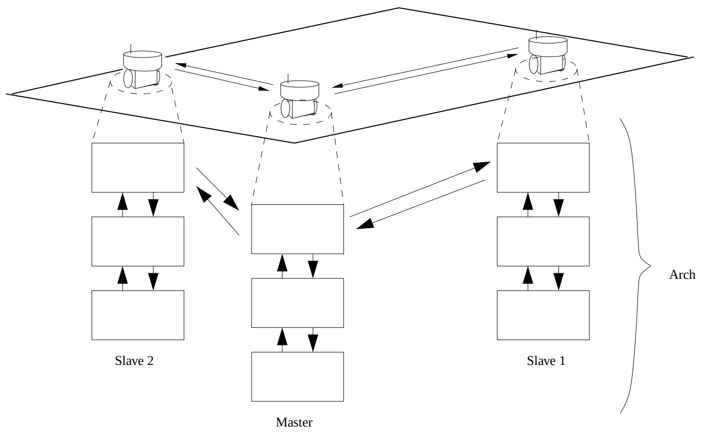


Figure 6.3: Control architecture for a three vehicle system. The architecture is obtained from the composition of the control structures for the three vehicles. Arrows between hierarchies represent communication links between vehicles. Arrows inside each hierarchical stack represent signals between different layers.

signals back to the vehicle supervisor the success or failure of the maneuver.

As we go down in the hierarchy there are certain aspects of the design that become more dependent on the dynamics of the vehicles. Thus, in order to explain how we design maneuvers we consider a specific coordination mission, namely the search for the minimum of a scalar field by a team of underwater vehicles. In our design this mission uses two types of maneuvers: goto waypoint and hold. The first maneuver drives the vehicle from its current position to the a given waypoint w_i within a given coordination time interval t. The second maneuver keeps the vehicle within a neighborhood of a given waypoint.

6.3.2 Waypoint generation

As as discussed in Section 6.2 the way-point generation procedure $\phi(.)$ produces the set of sampling points w (or way-points in a more general context) and the set of coordination times $t = \{t_1, t_2, t_3\}$. The coordination times are defined as follows:

(i) the *master* vehicle is required to arrive at its designated waypoint before t_1 and to stay within a given range of the waypoint until the end of the communication phase.

(ii) each *slave* vehicle is required to arrive at its designated waypoint (where it sends the sample to the *master*) in the time interval $[t_1, t_2]$ and to stay within a given range of the waypoint until the end of the communication phase.

(iii) the communication phase is required to terminate before t_3; each vehicle receives the next waypoint from the *master* during the time interval (t_2, t_3).

This is done to ensure that the vehicles are able to communicate among them during the communication phase, even in the presence of disturbances.

6.3.3 Team controller

We model each team controller as a transition system. Since the team controller can be in either master or slave mode, we have two team controller transition systems. They are described below. The master team controller,

$$T_M = (Q_M, \rightarrow, I_M, O_M, \mathrm{I}nit_M, \mathrm{F}inal_M)$$

shown in detail in Figure 6.4, consists of the parallel composition of three transition systems. The main functionality is provided by the upper transition system of Figure 6.4, which has four states Master Coord, Master Reconfig, Master Motion, Master Stop. The other two transition systems are counters: one stores the number of active slaves and the other keeps track of the number of received acknowledgments from the slaves during the coordination phase. The acknowledgment sent by each slave vehicle when it reaches the designated sampling point also encodes the corresponding sample.

In the state Master Coord, the master waits for the "Acks" (and samples) transmitted by the slaves. The transition to the state to Master Motion is taken when the *ack* counter reaches the number of active slaves and the termination condition is not true; on this transition the master computes the new sets of waypoints and coordination times, sends them to the slaves and resets the *ack* counter. The transition from Master Coord to Master Stop is taken if the termination condition is true when the *ack* counter reaches the number of slave vehicles. The transition from Master Coord to Master Reconfig takes place if a *Master timeout* is triggered before the *ack* counter reaches the number of slave vehicles. This happens if some of the slaves do not reach their assigned waypoints within the prescribed time frame. A team reconfiguration takes place in Master Reconfig and the number of active slaves is then updated through a state transition in the active slaves counter given by the middle transition system in Figure 6.4.

The transition from Master Motion to Master Coord is taken when the master reaches its waypoint. On this transition it commands its vehicle supervisor to execute a hold maneuver.

The slave team controller transition system is shown in Figure 6.5. The team controllers of the $N - 1$

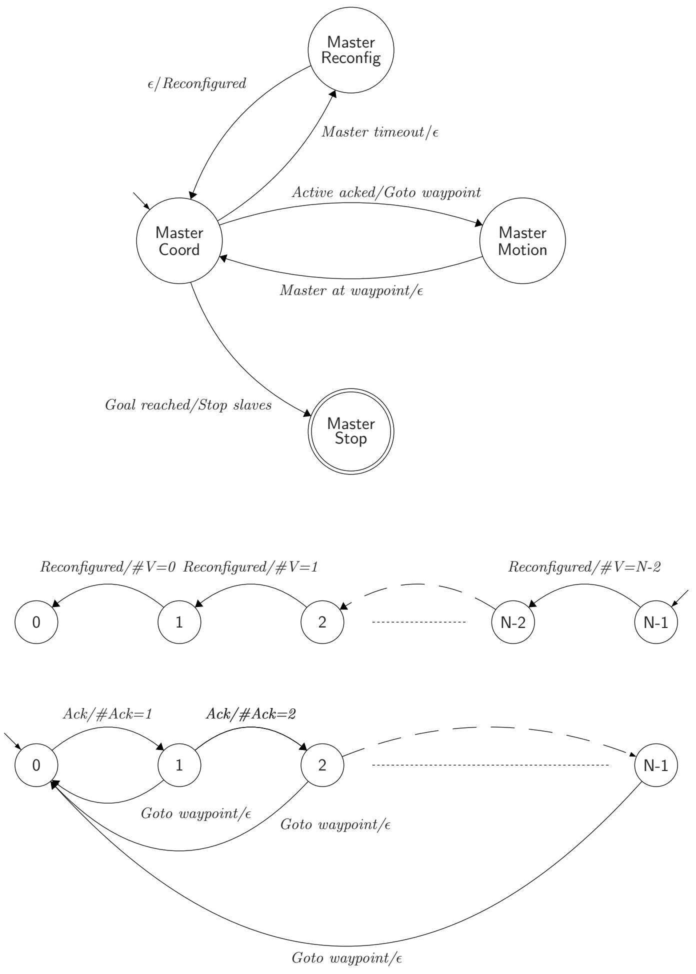


Figure 6.4: The master team controller is the parallel composition of three transition systems.

slaves are identical and denoted

$$T_{S_1} = \cdots = T_{S_{N-1}} = (Q_S, \rightarrow, I_S, O_S, \mathrm{I}nit_S, \mathrm{F}inal_S).$$

The states are Slave Coord, Slave Motion, Slave Stop. The initial state is Slave Coord, where the slave team controller is waiting for the next waypoint from the master team controller. The transition to Slave Motion is taken when the waypoint is received. On this transition it commands its vehicle supervisor to execute the goto waypoint maneuver. In Slave Motion the vehicle moves to the designated waypoint. The transition to Slave Coord is taken if the vehicle reaches the waypoint before the slave timeout expires; otherwise the slave team controller goes to Slave Stop. On the transition from Slave Motion to Slave Coord an ack is sent to the master team controller (together with the corresponding sample) and the vehicle supervisor is commanded to execute a hold maneuver. The slave team controller may also go to Slave Stop from Slave Coord. This transition typically takes place when the master has decided that the goal is reached and therefore forces all slaves to stop. Space limitations preclude a detailed description of the reconfiguration logic.

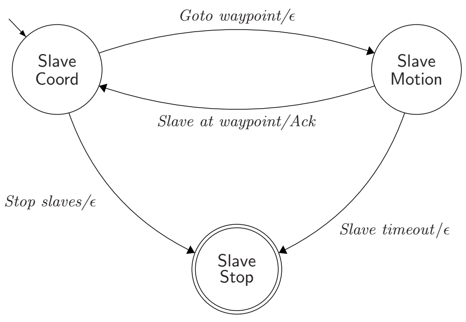


Figure 6.5: Slave team controller.

6.3.4 Vehicle supervisor

The vehicle supervisor interfaces the team controller with the maneuver controllers. The vehicle supervisor

$$T_V = (Q_V, \rightarrow, I_V, O_V, \mathrm{I}nit_V, \mathrm{F}inal_V)$$

is shown in Figure 6.6, where

- $Q_V = \{\text{Idle}, \text{Motion}, \text{Error}, \text{Stop}\}$
- $I_V = \{$*goto(w_i,t)*, *hold(w_i,t)*, *doneGoto(sp)*, *MtimeOut*, *stop*, *error(.)*$\}$
- $O_V = \{$*waypoint(sp)*, *startGoto(w_i,t)*, *startHold(w_i,t)*, *error(code)*, *stop*, *timeOut*$\}$
- $\mathrm{I}nit_V = \text{Idle}$ and $\mathrm{F}inal_V = \text{Stop}$

The input and output events model interactions with the team controller and with the maneuver controller: the supervisor receives the events *goto(.)* and *stop* from the team controller to execute a goto maneuver (with the specified parameters) and to stop the current maneuver respectively; it receives the events *doneGoto(.)*, *error(.)* and *MTtimeOut* from the current maneuver controller to indicate the termination of the current maneuver, the occurrence of an error, or the occurrence of a time out respectively; it sends the events *startGoto(.)*, *startHold(.)* and *stop* to start executing a goto or a hold maneuver and to stop the current maneuver; and it

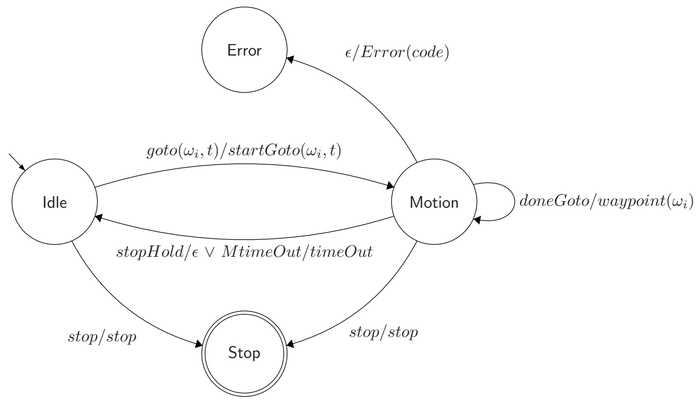

Error
$\epsilon/Error(code)$
$goto(\omega_i,t)/startGoto(\omega_i,t)$
Idle
Motion
$doneGoto/waypoint(\omega_i)$
$stopHold/\epsilon \vee MtimeOut/timeOut$
$stop/stop$
$stop/stop$
Stop

Figure 6.6: Vehicle supervisor.

sends the events *waypoint(sp)*, *error(code)* and *timeOut* to the team controller to indicate respectively that the waypoint was reached, that an error of type *code* has occurred and that time out has occurred. In the absence of errors, execution alternates between the states Idle and Motion.

Note that there are no clocks in the vehicle supervisor. The reasons for this are that: (i) both the supervisor and the maneuver controllers reside on the same vehicle and we can therefore assume reliable communications between them; and (ii) maneuver timeouts are modeled within the maneuver controllers.

6.3.5 Maneuver controller

The aspects of maneuver design are quite dependent on the dynamics of each vehicle. However, and for the purpose of modularity, maneuver controllers have to conform to a standard interface for the interactions with the vehicle supervisor. We describe this interface now.

The structure of each maneuver controller is as follows

$$T_C = (Q_C, \rightarrow, I_C, O_C, \mathrm{I}nit_C, \mathrm{F}inal_C)$$

where

- $Q_C = \{$Init, Motion, Error, Stop$\}$
- $I_C = \{$*start(.)*, *stop*$\}$
- $O_C = \{$*done(.)*, *error(code)*, *stop*, *timeOut*$\}$
- $\mathrm{I}nit_C =$ Init and $\mathrm{F}inal_C =$ Stop

In the motion state there is a low-level control law which generates references to actuators in continuous time. In practice, there may exist states other than Motion to encode the maneuver control logic.

6.4 System properties

In this section we show how the team control architecture implements the specification. This is done in a modular fashion. First, we show that the high level team coordination implemented through the composition of the master and slave team controllers is consistent with the specification under the assumptions that: 1) the generation of waypoints and coordination times produces points reachable both in space and time; and 2) the online execution control ensures that these points are indeed reached. Second, we state a set of conditions which ensures that the waypoint generation procedure produces waypoints and coordination times that are reachable both in time and space. Third, we discuss how the online execution control ensures that the waypoints are indeed reached under the assumption that the maneuver controllers produce guaranteed results. Fourth, we discuss the design of maneuver controllers which produce guaranteed results.

This modularity decouples efficiently the behavior of the team from that of the underlying coordination algorithm.

6.4.1 Team coordination

In this section we define a quotient transition system $T/\sim$ for the system T derived from the composition of the master and slave team controllers. We show that $T/\sim$ is isomorphic to the team coordination specification T_{Spec} in Section 6.2. Since T is bisimilar to $T/\sim$ by construction, we conclude that the closed-loop system based on the composition of team controllers satisfies the specification.

Recall the definition of simulation and bisimulation for transition systems (from [80]).

Definition 6.4.1 (Simulation and bisimulation)**.** Given two transition systems

$$T_1 = (Q_1, \rightarrow, I_1, O_1, \mathrm{I}nit_1, \mathrm{F}inal_1)$$

and

$$T_2 = (Q_2, \rightarrow, I_2, O_2, \mathrm{I}nit_2, \mathrm{F}inal_2),$$

we say that T_2 simulates T_1 with relation $R \subset Q_1 \times Q_2$ if $(x, y) \in R$ and $x \rightarrow x'$ implies that there exists $y' \in Q_2$ such that $y \rightarrow y'$ and $(x', y') \in R$. If T_1 simulates T_2 and T_2 simulates T_1, we say that T_1 and T_2 are bisimilar.

The composition of the master team controller

$$T_M = (Q_M, \rightarrow, I_M, O_M, \mathrm{I}nit_M, \mathrm{F}inal_M)$$

with $N-1$ identical slave team controllers

$$T_{S_1} = \cdots = T_{S_{N-1}} = (Q_S, \rightarrow, I_S, O_S, \mathrm{I}nit_S, \mathrm{F}inal_S)$$

is illustrated in Figure 6.3. Recall that to simplify notation we do not distinguish the transition relations, but the interpretation in each case should be clear from the context. The overall transition system $T = (Q, \rightarrow, I, O, \mathrm{I}nit, \mathrm{F}inal)$ is given by the parallel composition

$$T = T_M \| T_{S_1} \| \ldots \| T_{S_{N-1}}.$$

The state of T is denoted

$$q = (q_M, q_{S_1}, \ldots, q_{S_{N-1}}, k) \in Q = Q_M \times Q_S^{N-1} \times \{0, \ldots, N-1\},$$

where q_M is the state of the main part of the master team controller (upper transition system in Figure 6.4), q_{S_i} is the state of slave i team controller (Figure 6.5), and k is the number of active slaves (middle transition system in Figure 6.4). (We disregard the lower transition system in Figure 6.4.)

We introduce the quotient transition system $T/\sim = (Q/\sim, \rightarrow, I, O, \mathrm{I}nit/\sim, \mathrm{F}inal/\sim)$ with equivalence relation $\sim \subset Q \times Q$, which partitions the state space of T into four equivalence classes $Q_R, Q_C, Q_M, Q_S \subset Q$ (the indices indicate “Reconfiguration”, “Coordination”, “Motion” and “Stop” to highlight the idea behind the partition). The equivalence classes are defined as follows:

$$\begin{aligned}
Q_R &= \big\{q = (\textsf{Master Reconfig}, q_1, \ldots, q_{N-1}, \cdot) \in Q : \; q_i \in \{\textsf{Slave Coord}, \textsf{Slave Stop}\}\big\} \\
Q_C &= \big\{q = (\textsf{Master Coord}, q_1, \ldots, q_{N-1}, \cdot) \in Q : \; q_i \in \{\textsf{Slave Coord}, \textsf{Slave Stop}\}\big\} \\
Q_M &= \big\{q = (\textsf{Master Motion}, \cdot, \ldots, \cdot) \in Q\big\} \\
Q_S &= \big\{q = (\textsf{Master Stop}, \textsf{Slave Stop}, \ldots, \textsf{Slave Stop}, \cdot) \in Q\big\}.
\end{aligned}$$

Consider four elements $q_R \in Q_R$, $q_C \in Q_C$, $q_M \in Q_M$ and $q_S \in Q_S$. The transition relation for $T/\sim$ is then defined as follows:

- $q_R \rightarrow q_C$ provided that Master Reconfig $\rightarrow$ Master Coord and Slave Coord $\rightarrow$ Slave Stop
- $q_C \rightarrow q_M$ provided that Master Coord $\rightarrow$ Master Motion and Slave Coord $\rightarrow$ Slave Motion
- $q_C \rightarrow q_S$ provided that Master Coord $\rightarrow$ Master Stop and Slave Coord $\rightarrow$ Slave Stop

- $q_M \to q_R$ provided that Master Motion $\to$ Master Reconfig, Slave Motion $\to$ Slave Coord and Slave Motion $\to$ Slave Stop
- $q_M \to q_C$ provided that Master Motion $\to$ Master Coord, Slave Motion $\to$ Slave Coord and Slave Motion $\to$ Slave Stop.

The inputs I, outputs O, initial states $\mathrm{I}nit/\sim$ and final states $\mathrm{F}inal/\sim$ of $T/\sim$ are easily derived from T.

The following result follows from construction with R being the equivalence relation defined previously.

Lemma 6.4.1. *T and $T/\sim$ are bisimilar.*

We next show that $T/\sim$ and $T_{\mathrm{S}pec}$ are isomorphic. We recall the following definition.

Definition 6.4.2 (Isomorphic transition systems)**.** Two transition systems

$$T_1 = (Q_1, \to, I_1, O_1, \mathrm{I}nit_1, \mathrm{F}inal_1)$$

and

$$T_2 = (Q_2, \to, I_2, O_2, \mathrm{I}nit_2, \mathrm{F}inal_2)$$

are isomorphic if there is a bijection $h : Q_1 \to Q_2$ such that for all $x, y \in Q_1$ it holds that $x \to y$ if and only if $h(x) \to h(y)$.

In order to relate $T/\sim$ and $T_{\mathrm{S}pec}$, we need to identify the inputs of $T/\sim$ with the inputs of $T_{\mathrm{S}pec}$. It can easily be done by relating each transition of $T/\sim$ with a transition of $T_{\mathrm{S}pec}$:

- $q_R \to q_C$ corresponds to Team Reconfig $\to$ Team Coord
- $q_C \to q_M$ corresponds to Team Coord $\to$ Team Motion
- $q_C \to q_S$ corresponds to Team Coord $\to$ Team Stop
- $q_M \to q_R$ corresponds to Team Motion $\to$ Team Reconfig
- $q_M \to q_C$ corresponds to Team Motion $\to$ Team Coord.

A suitable bijective map $h : Q/\sim \to Q_{\mathrm{S}pec}$ of Definition 6.4.2 is simply the relabelling:

- $h(Q_R) =$ Team Reconfig
- $h(Q_C) =$ Team Coord
- $h(Q_M) =$ Team Motion

- $h(Q_S) = \textsf{Team Stop}$.

It then follows that $T/\sim$ and T_{Spec} are isomorphic. Two transition systems that are isomorphic are obviously also bisimilar. Since T and $T/\sim$ are bisimilar (Lemma 6.4.1) and thus also $T/\sim$ and T_{Spec} are bisimilar, we have the following main result.

Theorem 6.4.2. *T and T_{Spec} are bisimilar.*

The transition systems T and T_{Spec} are hence equivalent in the sense of a bisimulation relation. The implementation of the interconnected team controllers will thus satisfies the system specification.

6.4.2 Waypoint generation and online execution control

We have proved that the composition of the team controllers implements the specification under the assumption that the waypoint generation procedure and the online execution control satisfy a set of properties. We derive these properties in the framework of dynamic optimization.

The dynamic behavior of each vehicle is characterized by the set of reachable states. Recall some definitions of reach sets.

Definition 6.4.3 (Reach set starting at a given point)**.** Consider a trajectory $x(.)$ of a control system $\dot{x} = f(x, u), u(t) \in U(t)$ departing from $\{x_0, t_0\}$. The reach set $R[\tau, t_0, x_0]$ of the system at time τ, starting at position and time (x_0, t_0) is given by:

$$R[\tau, t_0, x_0] = \bigcup \{x[\tau] | u(s) \in U(s), s \in (t_0, \tau]\} \tag{6.4.1}$$

where $x[\tau]$ is the state of the system at time τ when driven by some measurable control $u(.)$ from (x_0, t_0).

Definition 6.4.4 (Reach set starting at a given set)**.** The reach set at time $\tau > t_0$ starting from set X_0 is :

$$R[\tau, t_0, X_0] = \bigcup \{R[\tau, t_0, x_0] | x_0 \in X_0\} \tag{6.4.2}$$

Similarly, we can define reach sets for dynamic systems under disturbances and state constraints (see [60, 62, 61]). The definition of reach set under uncertainty is quite useful to model the behavior of underwater vehicles under bounded disturbances, such as currents. In what follows we use the definition of reach set given above. However, nothing prevents us from using the other definitions in our approach.

We need some definitions. Let m_d, r_{com}, v_{com}, $B_r(x)$, t_c, S and m denote respectively the maximum distance from w_i during the hold maneuver, the maximum communication range, the velocity of propagation for communications, the closed ball of radius r centered at x, the time when the vehicles in V start a new motion phase, the set of indices for the slave vehicles and the index for the master vehicle.

Recall that each vehicle enters a hold maneuver after reaching its designated waypoint w_i.

Definition 6.4.5 (Admissible generation of waypoints and coordination times)**.** The generation of waypoints w_i and coordination times t_1, t_2 and t_3 is admissible if the following conditions hold

$$\forall i, j, \|w_i - w_j\| \leq r_{com} - 2m_d \tag{6.4.3}$$

$$t_3 - t_2 \geq \frac{2 \times r_{com}}{v_{com}}. \tag{6.4.4}$$

$$\exists t_m \in [t_c, t_1^+] : w_m^+ \in R[t_m, t_c, B_{\mathrm{m}_d}(w_m)]. \tag{6.4.5}$$

$$\forall i \in \mathbf{S}\ , \exists t_i \in [t_1^+, t_2^+] : w_i^+ \in R[t_i, t_c, B_{\mathrm{m}_d}(w_i)]. \tag{6.4.6}$$

Condition (6.4.3) ensures that the waypoints satisfy the communication constraints (which must be valid for the next waypoints); conditions (6.4.5) and (6.4.6) ensure that the master and the slaves reach the new waypoints within the prescribed time intervals; and condition (6.4.4) ensures that there is time for the communication round trip between each slave and the master.

A verified waypoint generation procedure is one which is admissible. The first two conditions do not rely on the dynamic properties of the vehicles. The last two conditions, however, require the calculation of the reach sets for each vehicle. This is a non-trivial task. Dynamic optimization techniques are used in [60] for this purpose. The observation is that the reach set is the sub-zero level set of a certain value function. The value function is obtained from the solution of a Hamilton-Jacobi equation. For linear systems with ellipsoidal constraints duality techniques are used to construct this solution.

The advantage of using value functions for reach set computations is that this approach also enables us to derive controllers which guarantee that the waypoints are reached. This is in line with the approach proposed in [68, 96].

The reach set formulation enables us to derive maneuver controllers for the hold and goto maneuvers which ensure guaranteed results. In these maneuvers, we are basically concerned with controlling the distance function from the current position of the vehicle to a given waypoint. In this case, we can use the construction proposed in [59] to calculate the safe set for a one-dimensional pursuit–evasion differential game which is easily extended to higher dimensional systems. This construction involves the integration of an ordinary differential equation, which describes how the distance evolves with time, and does not require the integration of a Hamilton-Jacobi equation.

6.5 Autonomous underwater vehicles in search mission

In this section we show how to implement a search strategy for a team of autonomous underwater vehicles (AUV) with our control architecture; this basically involves specializing the waypoint generation procedure and the maneuver design for this search strategy. We also illustrate these developments with simulation results.

The problem consists of finding the minimum of a temperature field with a search strategy based on a fixed-size version of the simplex optimization algorithm introduced in [93].

The underwater operations pose one additional challenge to the general search problem for a team of vehicles. The challenge comes from the nature of underwater communications. Typically autonomous underwater vehicles use acoustic communications which are quite constrained in range and in bandwidth. This is basically due to the problems associated with the propagation of sound underwater.

In what follows we consider a team of autonomous underwater vehicles equipped with acoustic modems for communication and some sensing device to measure some scalar variable, for example temperature.

The simplex algorithm is particularly suited for this challenging application. It is quite simple, robust, and very effective in finding the extremum of a scalar field from few samples. This leads to feasible requirements for underwater communications.

What also makes this method appealing is the fact that it allows reasoning about vehicle motion in discrete terms: indeed the simplex algorithm imposes a discretization of the configuration space which facilitates the implementation of the proposed hierarchical structure. For example, the conditions for the generation of admissible waypoints are trivially satisfied with an appropriate choice of the grid size.

For the purpose of clarity we also restrict our search to motions in the horizontal plane.

6.5.1 Simplex algorithm

The simplex optimization algorithm is a direct search method which behaves much like a gradient descent method but with no explicit gradient calculation. It is typically applied in situations where the gradient calculations are quite difficult and the computation power is quite limited. This happens, for example, with scalar fields corrupted by noise. We are interested in executing a search operation for finding the minimum of a planar field defined over a convex set $\Omega \subset \mathbb{R}^2$ (see Figure 6.7).

The simplex optimization method starts by evaluating the scalar field at the vertices of a three-sided simplex, placed at an initial guess position. It then proceeds by creating a new simplex, obtained by reflecting the vertex associated to the sample with higher field value. The reflection is with respect to the line passing through the two remaining vertices. The algorithm stops when the newly generated simplex coincides with the simplex generated two iterations before, namely after two reflections step we need to reflect the starting

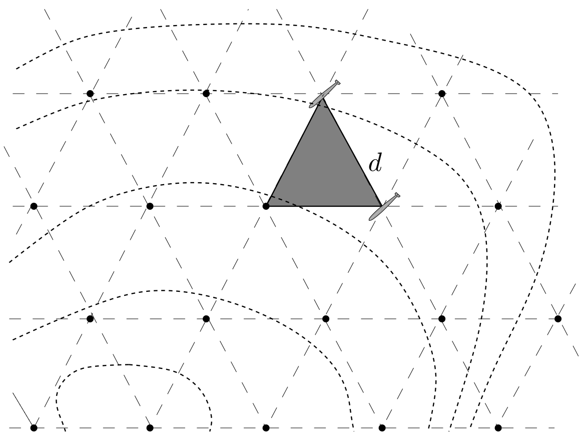


Figure 6.7: A triangular grid with aperture d over a scalar field depicted through its level curves (dark dashed lines). The shaded triangle illustrates the simplex location, which evolves on the grid.

vertex. This procedure is formally described below.

```
1: $z(0) := (z_1(0), z_2(0), z_3(0))$
2: $k := 0$
3: **while** $k < 2 \vee z(k) \neq (k-2)$ **do**
4:     $i := \arg\max_i F(z_i(k))$
5:     $z_i' := z_j + z_h - z_i$ with $j, h \in \{1,2,3\}$ and $j \neq h, j \neq i, h \neq i$
6:     $z_j' := z_j$
7:     $z_h' := z_h$
8:     $z(k+1) := (z_1', z_2', z_3')$
9:     $k := k+1$
10: **end while**
```

Algorithm 1: Simplex algorithm.

Consider a triangular grid $\mathcal{G} \subset \Omega$ with aperture d, as depicted in Figure 6.7. Introduce an arbitrary point $p_0 \in \Omega$ and a base of vectors given by b_1, b_2 such that $b_1^T b_1 = b_2^T b_2 = d^2$ and $b_1^T b_2 = d^2 \cos \pi/3$. The grid is then the set of points

$$\mathcal{G} = \{p \in \Omega |\, p = p_0 + kb_1 + \ell b_2,\ k, \ell \in \mathbb{Z}\}\,.$$

A simplex $z = (z_1, z_2, z_3) \in \mathcal{G}^3$ is defined by three neighboring vertices of $\mathcal{G}$, which belong to a triangle. Let $F : \Omega \to \mathbb{R}$ the scalar field. The reflection rule updates the simplex in the following way. Suppose, without loss of generality, that $F(z_3) \geq F(z_i)$, $i = 1, 2$. Given a simplex $z = (z_1, z_2, z_3)$ the next simplex is

$$z^+ = (z_1, z_2, z_3)^+ = (z_1, z_2, z_1 + z_2 - z_3)\,.$$

The simplex algorithm is summarized as Algorithm 1.

We see from the condition on line 3 that the algorithm stops at iteration $\bar{k}$ when $z(\bar{k}) = z(\bar{k} - 2)$. Since the algorithm is deterministic, it follows that a continuation after step $\bar{k}$ would lead to an oscillation between the two discrete states $z(\bar{k})$ and $z(\bar{k} - 1)$.

The main limitation of the simplex algorithm comes from the fact that there no guarantees that a vicinity of the minimum has been reached when the algorithm stops. However, it can be used as a first strategy to get close to the minimum.

6.5.2 Waypoint generation

The waypoint generation procedure is based on a modified version of the simplex algorithm. It runs on the master vehicle and it is invoked to generate the new waypoints after the reception of the measurements from all the vehicles in the team.

Let assume $N = 3$. Let us denote with (w_1, w_2, w_3) the current simplex and with $(w_1, w_2, w_3)^+$ the next simplex. For simplicity of notation we define the reflecting operator

$$\xi : \mathcal{G}^3 \to \mathcal{G}^3 : (w_1, w_2, w_3) \mapsto \gamma(w_1, w_2, w_3) = w_3 + w_2 - w_1 \,,$$

that is $\gamma(w_1, w_2, w_3)$ takes the first argument and computes its reflection with respect to the second and third arguments. Thus the simplex algorithm can be then described by the map $(w^+, t^+) = \phi_{\text{simplex}}(w, t, e)$ where $w \in \mathcal{G}^3$ is a simplex, w^+ is computed through the reflecting operator and an event e is related to the fact a vehicle arrived in a neighborhood of the waypoint.

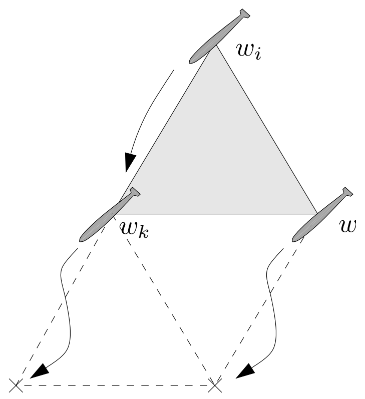


Figure 6.8: Assignment of the next waypoints for the three AUVs, by the master team controller, when $F(w_i) \geq F(w_j) \geq F(w_k)$.

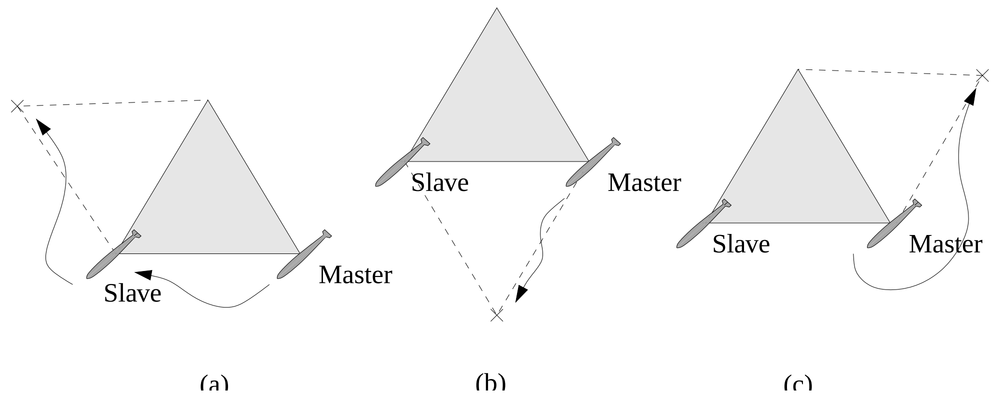


Figure 6.9: Assignment, by the master team controller, of the next waypoints when only one slave AUV is present.

We observe that the master can compute two steps of the simplex algorithm without knowing the new samples. Let us assume, without loss of generality that we start with the simplex (w_1, w_2, w_3) such that $F(w_1) \geq F(w_2) \geq F(w_3)$. Applying the simplex algorithm we have $(w_1, w_2, w_3)^+ = (\gamma(w_1, w_2, w_3), w_2, w_3)$. However in this situation the master can already compute the next simplex. Indeed two situations could occur. The case $F(\gamma(w_1, w_2, w_3)) \geq \max(F(w_2), F(w_3))$ implies that $(\gamma(w_1, w_2, w_3), w_2, w_3) = (w_1, w_2, w_3)$, and thus the algorithm stops. Otherwise we compute the reflected waypoint of w_2 with respect to $\gamma(w_1, w_2, w_3)$ and w_3. We have that the transition

$$\textsf{Team Coord} \xrightarrow{Active\ acked \big/ (w_1, w_2, w_3)^+} \textsf{Team Motion} \tag{6.5.1}$$

is such that

$$(w_i, w_j, w_k)^+ = (w_k, \gamma(w_i, w_j, w_k), \gamma(w_j, w_k, \gamma(w_i, w_j, w_k)))$$

with $F(w_i) \geq F(w_j) \geq F(w_k)$. The situation is represented in Figure 6.8.

The algorithm can be easily modified to incorporate the reconfiguration logic discussed in the previous section. This happens when one the slave vehicles is not able to reach its designated waypoint. Notice that the master keeps track of the field values for the previous simplex. This is enough to compute the next simplex. The waypoint assignment for two vehicles is as shown in Figure 6.9.

6.5.3 Maneuver controller design

We design the maneuver controllers in the framework of hybrid automata. We present the maneuver controller for the goto maneuver. Due to space limitations we embed a simplified design of the hold maneuver controller as a state of this controller, in order to fully illustrate the control logic. The hybrid automaton model of the goto maneuver controllers is depicted in Figure 6.10. The continuous state space is $X \subseteq \mathbb{R}^4$ since we have the state of the vehicle $(x, y, \psi)^T$ and the time t.

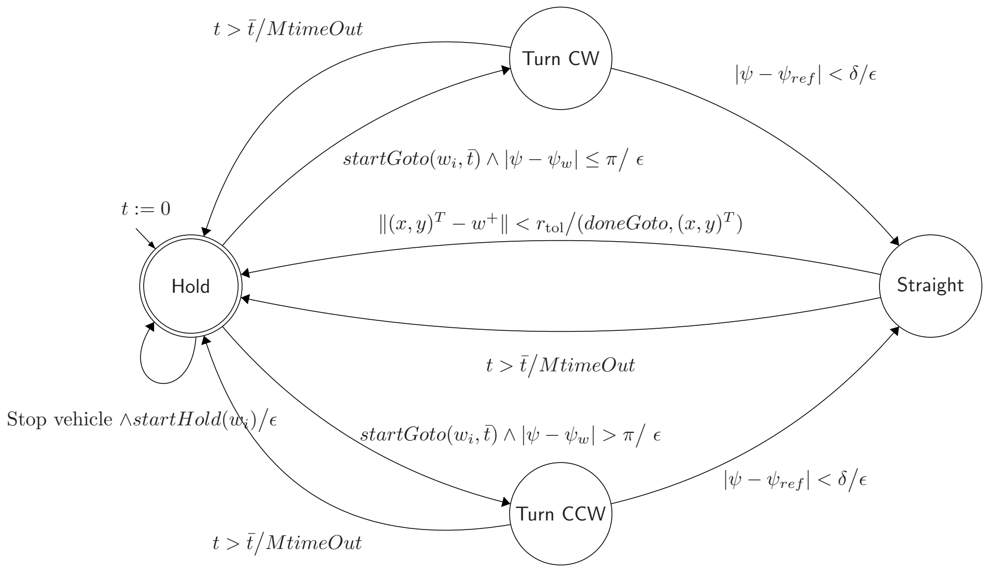

Turn CW
Hold
Straight
Turn CCW
$t > \bar{t}/MtimeOut$
$|\psi - \psi_{ref}| < \delta/\epsilon$
$startGoto(w_i, \bar{t}) \wedge |\psi - \psi_w| \leq \pi/\ \epsilon$
$t := 0$
$\|(x, y)^T - w^+\| < r_{\mathrm{tol}}/(doneGoto, (x, y)^T)$
$t > \bar{t}/MtimeOut$
Stop vehicle $\wedge startHold(w_i)/\epsilon$
$startGoto(w_i, \bar{t}) \wedge |\psi - \psi_w| > \pi/\ \epsilon$
$|\psi - \psi_{ref}| < \delta/\epsilon$
$t > \bar{t}/MtimeOut$

Figure 6.10: Hybrid automaton model of the maneuver controller.

The system starts in the Hold state. In this state the controller maintains a constant velocity with a fixed turn rate so that the vehicle follows a circular trajectory; this is because the vehicle is not capable of hovering in place. If the vehicle supervisor sends a $startGoto(w_i^+, \bar{t})$ command, then the maneuver controller of vehicle i needs to steer the vehicle tracking a trajectory of the type shown in Figure 6.11. Depending on the heading of the vehicle with respect to the final waypoint, the system will transition to state Turn CW or to state Turn CCW, turning clockwise or counter clockwise, respectively, with maximum angular velocity Figure 6.11. When the angle of the vehicle ψ is close to the angle ψ_{ref} the vehicle switches to the Straight state. The value of ψ_{ref} is chosen such that in state Straight the controller will make the vehicle follow a straight line passing through the next waypoint. When the distance between the vehicle and the final waypoint w_i^+ is less than a given threshold, r_{tol}, the maneuver controller returns to the Hold state maintaining the vehicle close to this waypoint. If the vehicle is not able to finish the $startGoto(.)$ maneuver by time $\bar{t}$ an error signal is communicated to the vehicle supervisor. In case of success a $doneGoto$ together with the sample taken at the waypoint are sent to the vehicle supervisor.

This is a very simple, though instructive, example of how to build a maneuver control for this type of

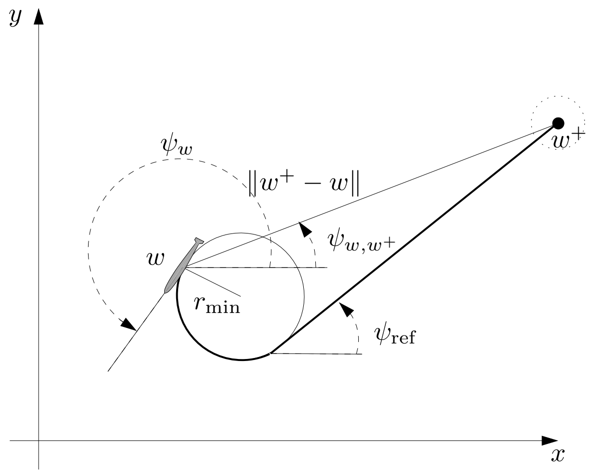


Figure 6.11: Example of a vehicle trajectory linking w to w^+.

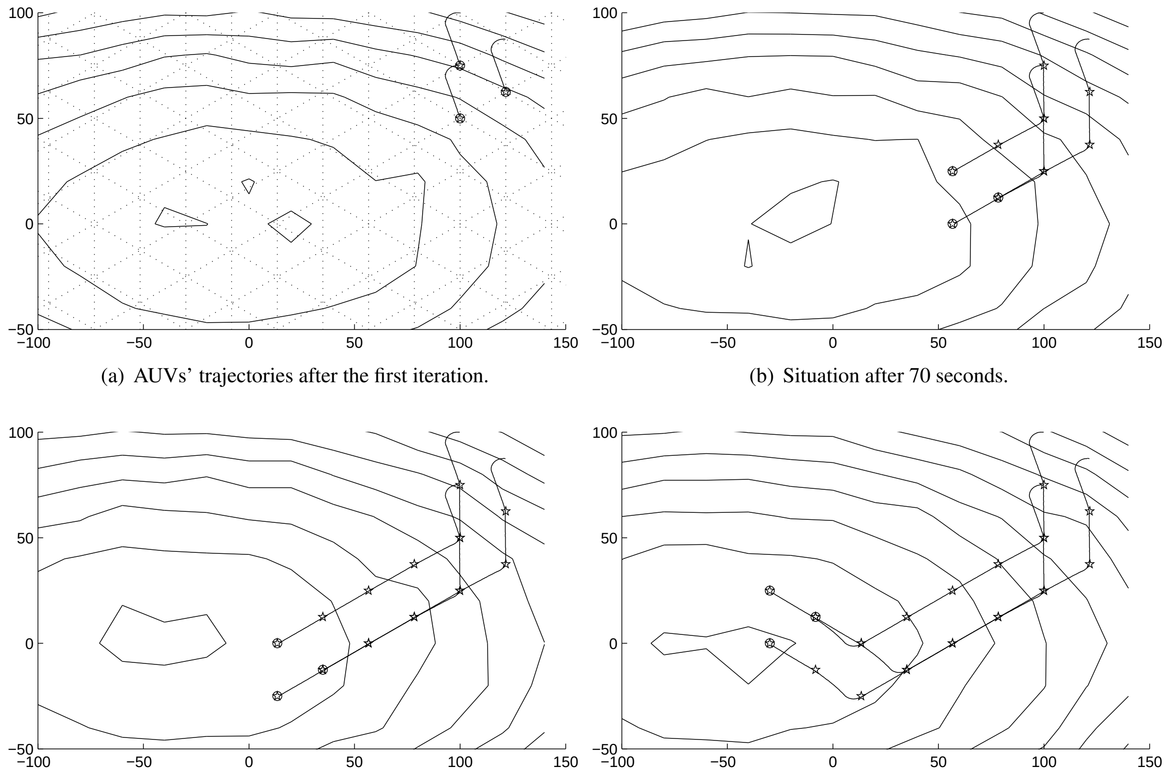
architecture. Complex control strategies, such as those discussed in [91], could be easily considered in this framework. The same applies to the techniques proposed in [2] to counteract the action of disturbances, such as currents.

6.5.4 Simulations results

Computer simulations were performed to illustrate the behavior a team of AUVs operating under the proposed hierarchical control structure. We considered the simplex based search with three AUVs in a time-varying planar scalar field (which could represent salinity, temperature, etc.).

Figure 6.12 shows four snapshots of the evolution of the AUVs' positions in a scalar field. The field is quadratic with additive white noise and a constant drift of $(-0.4, 0)$ m/s. The approximately ellipsoidal lines are the level curves of the scalar field. Notice that we have added noise to the measurements, which is the reason why the level curves are not smooth. The simulation starts with the AUVs at the desired depth and at the vertices of a predefined initial simplex $w = ((100, 50), (122, 62), (100, 75))$. Figure 6.12(a) shows the initial trajectory of the AUVs. The grid implicitly imposed by the simplex algorithm is illustrated in this plot. The multi-vehicle system completes the search procedure after 135 s.

Figure 6.13 shows another scenario for the evolution of three AUVs towards the extremum of the scalar field. The initial simplex is $w = ((400, 300), (422, 312), (400, 275))$. The figure is labeled with the discrete states of the team controllers (TC), vehicle supervisors (VS) and maneuver controllers (MC) for different phases of the operation. During the progression, one of the AUVs fails to reach its waypoint (this is the AUV



(a) AUVs' trajectories after the first iteration.

(b) Situation after 70 seconds.

(c) Situation after 100 seconds.

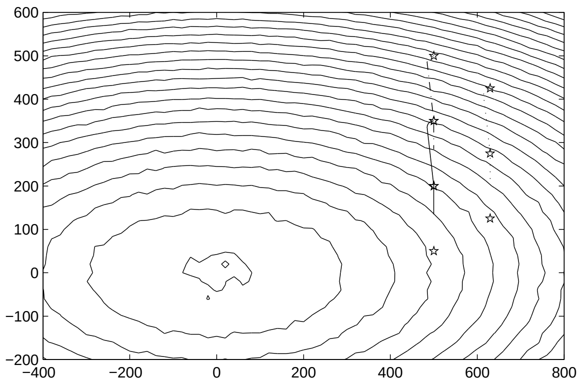
(d) Search mission completed after 135 seconds.

Figure 6.12: Simplex coordination algorithm executing a search in a noisy quadratic field with drift.

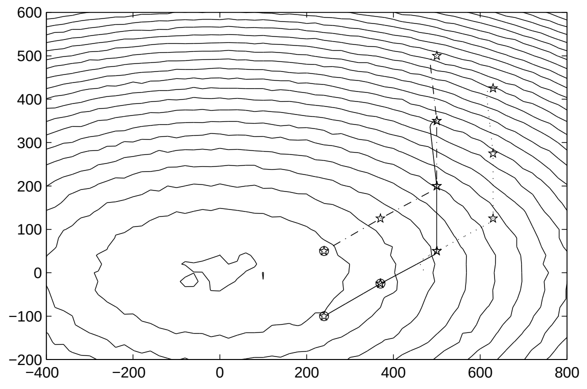
performing the dotted trajectory in this figure). The other two AUVs reach their corresponding waypoints and wait there until the timeout occurs. Note the circular trajectories of these two AUVs while waiting. At timeout, the system is reconfigured and the team, now composed of two vehicles, proceeds with the execution of the search. The team is able to progress towards the extremum of the field, despite the failure of one of the vehicles.

6.6 Conclusions

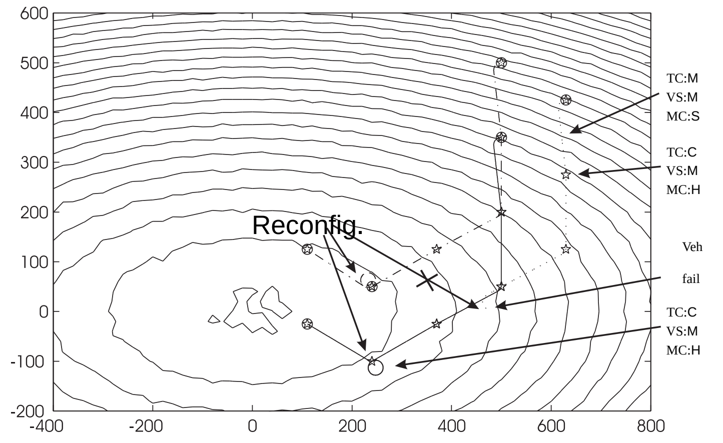
We presented a design of a hierarchical control architecture for coordinated multi-vehicle operations. The design space is large and heterogeneous. We structure the space by first decomposing it into waypoint generation and online execution control. The waypoint generation procedure generates the waypoints for the team to search for the minimum of a scalar field under dynamic and communication constraints and in accordance to a given optimization algorithm. Execution control is organized as a three level hierarchy of team controller,



(a) One step of the search algorithm



(b) The vehicle with the dotted trajectory does not reach the assigned waypoint



(c) Reconfiguration and continuation of the search algorithm

Figure 6.13: Trajectories of three AUVs (solid, dotted, dash-dot) moving towards the minimizer of a scalar field. The stars correspond to the generated waypoints. Note the reconfiguration after a vehicle failure. The convention for describing the state of the controllers is as: $M-Motion; C-Coord; S-Straight; H-Hold.$

supervisor, and maneuver controller.

It is shown that the controller implementation is consistent with the system specification on the desired team behavior. This is done in a modular fashion by layering the execution control and designing each layer to ensure that the controllers produce guaranteed results under the assumption that the controllers at the adjacent layers also produce guaranteed results.

Computer simulations illustrate the overall system performance for a multi-vehicle search mission which is motivated by the classical simplex optimization algorithm. This example illustrates the specialization of the design to a specific application. Basically this involves specializing the waypoint generation procedure according to the coordination strategy and the maneuver controllers according to the specific dynamics of each vehicle.

Chapter 7

Conclusions

In this work we introduced a modeling framework and a control framework for systems with coupled computational and physical dynamics. These developments are targeted at the deployment of the next generation of networked vehicle systems in remote and communications-challenged environments. The focus will be on systems endowed with organizational properties that transcend the capabilities of each constituent subsystem. In this chapter we first summarize what has been accomplished and discuss future research.

7.1 What has been accomplished

The main contribution of this dissertation is the development of a model and of a control framework for networked vehicle systems composed of physical and computational entities with coupled dynamics. This model introduces a new perspective on the operation of networked vehicle systems and presents a methodology to implement complex organizations to control these systems. The model highlights the coupling between computational and physical entities and provides a uniform analysis and design framework. The model is not specific to networked vehicle systems and may help to shed a new perspective on cyber-physical systems [4].

The developments in this thesis were motivated by real applications and are targeted at a new generation of networked vehicle systems. A description of what has been accomplished would not be complete without a reference to the developments and field experiments of the Laboratório de Sistemas e Tecnologias Subaquáticas of Porto University. Real vehicles and operational deployments provide the ultimate test of our developments. This is already happening, as described in subsection 1.4.3, and future deployments are already in preparation. In a recent experiment, which took place during the REP14–Atlantic exercise, an UAV was able to control, in a feedback manner, a submerged AUV. This was done with the help of an

ASV, a Wave Glider, carrying a Manta gateway to bridge acoustic and wireless communications. The three unmanned vehicles formed an organization with unprecedented capabilities.

The modeling framework highlights the coupling of computational and physical dynamics and the new types of control actions, such as the migration of controllers. This is described in extended state-spaces and control spaces allowing us to express formal specifications in the languages of attainability, invariance, and optimization.

The control and computational framework for organizations of networked vehicles systems allows a systematic design methodology within which properties of the organization can be proved to satisfy formal requirements. The framework encompasses a control and computation architecture and a design methodology. The architecture derives from a few design principles and is implemented with the help of a few mechanisms. The design methodology introduces a compositional layered approach to control and computation that is uniform for all vehicles.

The optimization of the behavior of networked vehicle systems is addressed in the framework of iterated multi-vehicle rendezvous problems with coordination constraints. This is because of intermittent communications. The structure of multi-vehicle rendezvous problems allows a structured application of the principle of optimality which results in coordinated optimization problems formulated in lower-dimensional spaces. Structure induces the composition of value functions in these lower-dimensional spaces, thus avoiding the problem of working in the product of the space states for all vehicles. The same structure of problem, namely that of the associated cost functions, may preclude the application of the principle of optimality. This is also studied in this work.

7.2 Directions for future research

This work opens several directions for future research. Some of them address modeling frameworks, others concern fundamental developments in control, optimization, and computation, others concern operational deployments and associated software frameworks, and others relate to applications in other fields. Some of these developments are already underway at the LSTS.

7.2.1 Modeling framework

There are several directions for future research on modeling frameworks for networked vehicle systems. First, the interplay between physical and logical communication channels, and the consideration of channel capacity. Second, the migration mechanism for computational entities. Third, the composition of organizations of networked vehicle systems. Fourth, mechanisms for self-aware and self-guided migration of computational entities.

7.2.2 Dynamic optimization

There are several directions for future research in dynamic optimization. First, the investigation of other collaborative control problems in the Dynamic Programming framework. Second, the derivation of conditions under which the Principle of Optimality holds for cooperative control problems with intermittent communications. Third, the generalization of the results leading to coordination and control formulations in lower dimensional spaces; this will be done in the framework of structured principles of optimality and decomposability. Fourth, a comprehensive dynamic optimization formulation to encompass the extended control and state spaces discussed in this work. Fifth, the development of approximate numerical methods for solving the Hamilton-Jacobi-Bellman equation. Sixth, the derivation of conditions for the convergence of the controls derived from the Hamilton-Jacobi-Equation. Seventh, we the investigation of turnpike theory in connection to problem of the controlled modification of cost functions. Finally, we will also investigate how to remove the more restrictive assumptions used in our developments.

7.2.3 Control architectures

The techniques used to develop a verified control architecture by design will be extended to accommodate more complex requirements for a networked vehicle systems. These include persistent operations and control of organizations.

7.2.4 Software frameworks

The LSTS software tool chain, briefly described in subsection 1.4.2, is a work in progress targeted at the deployment of our computation and control frameworks. Work is already underway to deploy mobile controllers in our vehicle networks. A prototype of a programming language for distributed control of multiple autonomous vehicles, called the Networked Vehicles Language (NVL), has already been deployed. A NVL program runs on a dynamic network environment, specifying on-the-fly selection of vehicles and their engagement to multi-vehicle controllers. The language defines primitives for vehicle selection, concurrent controller execution, timed synchronization, and general program control flow. The language assumes the two phase approach for coordination introduced in Chapter 6.

7.2.5 Other fields

A generic description of the controlled networked vehicle systems under discussion is required. These are systems where computational and physical entities form complex organizations with the help of a few fundamental mechanisms. Physical entities lodge computational entities and have access to physical communications that depend on relative distances to other physical entities. Physical communications allow logical communication channels. Computational entities are created and destroyed on the fly and are allowed to migrate, as messages, between physical entities. Computational and physical entities form organizations. Organizations can be addressed and controlled as a whole. Organizations exist because of communications, access control, and specialized controllers. Organizations embody functions required to move, to communicate, to coordinate, and to respond to external interactions. This generic description is not specific to networked vehicle systems. These mechanisms occur in natural systems and in man-made systems/organizations. Of special interest are cyber-physical systems [4].

This work may contribute to new insights in other fields such as biology, ecology, and social sciences. We need to select a few case studies from these fields for analysis. This requires a highly inter-disciplinary approach. Some of the underlying assumptions and modeling simplifications may have to be challenged. In the process computation, control, and communication mechanisms of value to networked vehicle systems may be identified.

Appendices

Appendix A

LSTS vehicle systems

A brief description of the vehicle systems designed and developed by LSTS follows.

Light Autonomous Underwater Vehicle (LAUV). This is a torpedo shaped vehicle with one propeller and 4 control fins which is available in several configurations. The maximum operating depth is 100m. The maximum speed is 2 m/s and the maximum distance that can be traveled on a battery charge is over 50km. The LAUV class vehicles are equipped with GPS/WiFi/GMS/Iridium communications, several types of side-scan sonars (Yellowfin from Imagenex, Marine Sonics and 2205 from Edgetech), multi-beam sonars (DeltaT from Imagenex), acoustic modems (Evologics and Micro-modem from Woods Hole), environmental sensors (CTD, chlorophyll, velocity of sound, backscatter), video cameras, and imaging sonars (Blueview P900) and profiling sonars (Imagenex) for obstacle avoidance. Two navigation suites are available for the LAUV class vehicles: 1) Long baseline navigation, which relies on pre-positioned external beacons; and 2) Inertial navigation, which relies on tactical grade Inertial Measurement Units (Honeywell 1700) and Doppler Velocity Log (Linkquest), thus making the vehicle independent of external navigation aids. External beacons and/or acoustic modems (Evologics and Micro-modem) are available for AUV navigation.

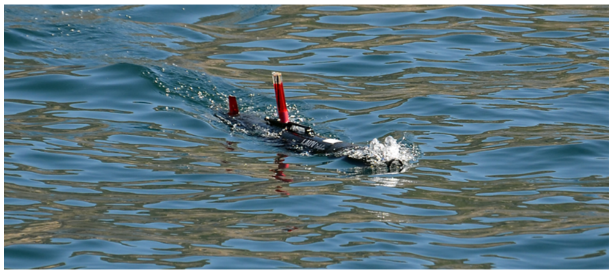


Figure A.1: Light Autonomous Underwater Vehicle.

Swordfish autonomous surface vehicle. This is a 4.5m long catamaran based platform equipped with computers, electric motors and sensor systems mounted on the twin hulls for autonomous operation. The maximum speed is 4m/s and the endurance is in the order of 6 hours. Swordfish is equipped with GPS/WiFi/GMS/Iridium communications, AIS, several types of side-scan sonars (Yellowfin from Imagenex, Marine Sonics and 2205 from Edgetech), environmental sensors (CTD, chlorophyll, velocity of sound, backscatter), video cameras, imaging sonars (Blueview P900), and radar for obstacle avoidance. Navigation is based on a GPS compass and on an Inertial Measurement Unit (Honeywell 1700).

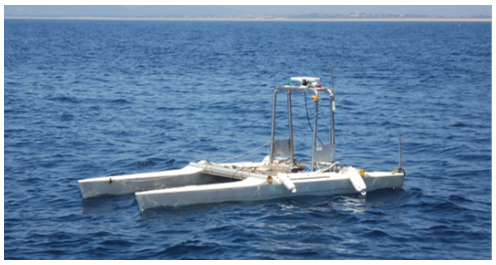


Figure A.2: Swordfish Autonomous Surface Vehicle.

Remotely operated vehicle Adamastor. This is a modular ROV for underwater inspection and intervention. It has advanced thrust and power control for operations at sea. Dimensions: 120 x 70 x 90 cm; weight: 90 kg; 5 Seaeye SI-MCT01 Thrusters; max operating depth: 200m; Power: 3Kw. It has a video camera and a 2-degree of freedom robotic arm for interventions. A ROV from Deep Ocean Engineering is also available.

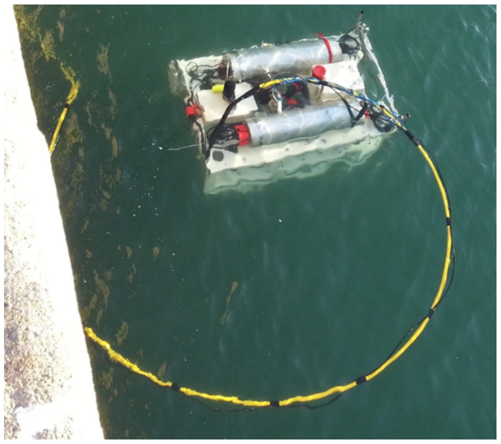


Figure A.3: Adamastor Remotely Operated Vehicle.

Manta gateways The Manta Gateway is a portable centralized communication hub supporting several types of wireless and acoustic networks. The system is capable of transparently route data between heterogeneous network links, balancing bandwidth and range. Additionally the device is capable of providing information about the localization of underwater vehicles and narrow band acoustic transponders. The gateway can be mounted on the buoys. It supports up to three 12V power over Ethernet radios connected at the same time; provides 10 hours of autonomy (with two radios); and runs open-source GLUED Linux distribution and SDK.

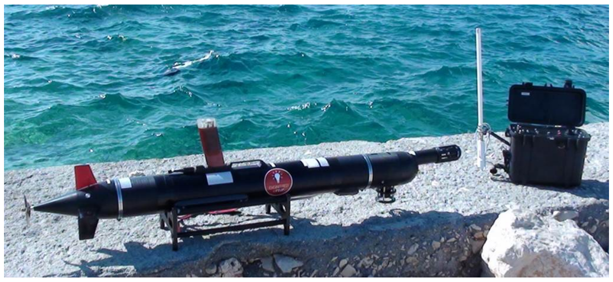


Figure A.4: Light Autonomous Underwater Vehicle and Manta communications gateway.

Shore side control station. The shore side control stations are based on networks computers with additional support for communications (WiFi, GSM, acoustic, satellite).

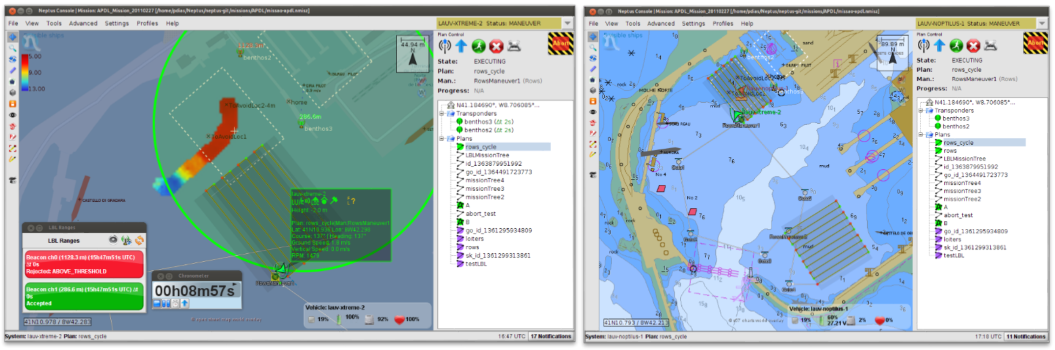


Figure A.5: Command and control interfaces.

X8 mini-UAS. The X8 is fixed wing, battery powered, hand-launched, with a wingspan of 1.8m, maximum takeoff weight (MTOW) of 3Kg, and 50 minutes endurance.

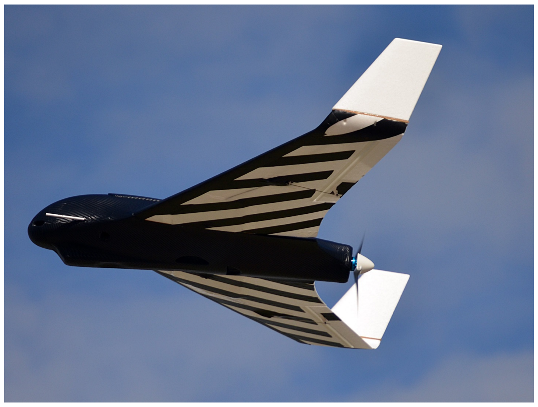


Figure A.6: X8 Unmanned Air Vehicle.

Antex -UAS. The Antex is a family of fixed wing, combustion engine UAS with wingspans ranging between 2.4m and 6m, maximum takeoff weight (MTOW) between 12kg and 150kg, and up to 12hours endurance. These vehicles are under development in the framework of the PITVANT project, a 7 year collaborative UAS development program undertaken by the Portuguese Air Force Academy and Porto University with funding from the Portuguese Ministry of Defense.

Appendix B

Non-smooth analysis background

B.1 The first constructs

B.1.1 Dini derivatives

The increasing sophistication of control and optimization methods has been a strong motivation for the development of nonsmooth calculus and the latest developments (see for example [84], [16]) in this field exhibit a great potential for control theory and applications.

Nonsmooth calculus has been in the mind of mathematicians since the last century. The first constructs of nonsmooth analysis were introduced by Dini in the 19th century. One of this constructs is the lower right derivative.

Definition B.1.1 (Lower right derivative)**.** Let f be a continuous function $f : \Re \to \mathbb{R}$. The lower right derivative at a point x, denoted Df(x), is defined as:

$$Df(x) := \lim_{t \downarrow 0} inf \frac{f(x+t) - f(x)}{t}$$

The extension of this concept to a larger class of functions, briefly sketched in the following subsection, illustrates one route to nonsmooth calculus.

B.1.2 Subderivatives and D-subgradients

Definition B.1.2 (Subderivative)**.** Given a function $f \in \mathcal{F}(\mathbb{R}^n) : \mathbb{R}^n \to \mathbb{R}$ and a vector $v \in \mathbb{R}$, the subderivative of f at x in the direction of v, denoted Df(x;v) is defined as:

$$Df(x;v) := \lim_{\substack{t \downarrow 0 \\ v' \to v}} \quad \inf \quad \frac{f(x+tv') - f(x)}{t} \tag{B.1.1}$$

Proposition B.1.1 (Properties of the subderivative)**.** *:*

1. The function $v \mapsto Df(x;v)$ is lower semicontinuous.

2. *If f is Lipschitz of rank K near x, then the function $v \mapsto Df(x;v)$ is Lipschitz of rank K on $\mathbb{R}^n$ and:*

$$Df(x;v) := \lim_{t \downarrow 0} \quad \mathit{inf} \quad \frac{f(x+tv)-f(x)}{t}$$

Definition B.1.3 (D-subgradient)**.** Let $f \in \mathcal{F}(\mathbb{R}^n)$. Then ξ is the directional subgradient or D-subgradient of f at x provided that $x \in$ dom f and:

$$Df(x;v) \geq \langle \xi, v \rangle, \forall v \in \mathbb{R}^n$$

Definition B.1.4 (D-subdifferential)**.** Let $f \in \mathcal{F}(\mathbb{R}^n)$. The set of all D-subgradients ξ is the D-subdifferential, denoted by $\partial_D f(x)$

Proposition B.1.2 (Properties of the D-subdifferential)**.** *:*

- $\partial_D f(x)$ *is closed and convex and reduces to* $\{f'(x)\}$ *if f is Frechet differentiable at x.*
- $\partial_D f(x)$ *is bounded if f is Lipschitz near x.*
- $\partial_D f_1(x) + \partial_D f_2(x) \subseteq \partial_D(f_1 + f_2)(x)$

These constructs incorporate some of the main ingredients for developing a nonsmooth calculus:

- Extended notions of limits in the definition of derivatives and defining different types of calculus.
- Set-valued gradients and, as a consequence, a set-valued calculus.
- Geometric interpretation where one can expect some forms of duality.

B.2 Proximal calculus

B.2.1 Introduction

Proximal normals are direction vectors pointing outward from a set, generated by projecting a point to the set.

Proximal subgradients have a certain local support property to the epigraph of a function. Namely, the proximal gradient of a lower semicontinuous function f at $x \in$ dom f is, within an appropriate scaling factor, a component of the proximal normal to the epigraph of the function at the point $(x, f(x))$.

The geometric interpretation of these concepts is intimately related to the properties of:

- The projection of a point on a set.
- The distance function.

Here, we present the proximal calculus in the Hilbert space setting. First, the main definitions (proximal normals and proximal subgradients) and some of the proximal calculus rules are introduced. Next, we discuss an important fact: the proximal subdifferential can be empty at some points. In fact, at each point, the existence of the proximal subdifferential can be interpreted in terms of a quadratic function that approximates the original function from below. However, a profound result, the density theorem, asserts the existence of proximal subdifferentials on a dense set. This result is then used to deduce two minimization principles asserting that, by adding a certain small perturbation to a bounded from below lower semicontinuous function defined on a compact set a minimum is always attained. Obviously, this is not always the case for infinite dimensional systems. The existence of this minimum is interpreted by the inclusion of the zero vector in the subdifferential of the perturbed function at the minimum. These minimization principles are then contrasted with the celebrated Ekeland's theorem, a result of the same nature whose proof relies on an interesting notion of partial order. The existence of proximal subdifferentials is a recurrent operational difficulty that can also be addressed in the context of limiting processes, discussed in the last subsection of this section. Devices obtained through limiting processes solve the existence problem and, moreover, contribute to the enlargement of the proximal subdifferential in some cases. To conclude, the nonexistence of proximal subdifferential at some points does not prevent the application of proximal analysis.

B.2.2 Proximal normals

Let X be a real Hilbert space, and let S be a non-empty closed subset of X. In this section the focus is on closed sets.

Definition B.2.1 (Projection of a point on a set)**.** Consider a point $x \notin S$. Suppose that there exists a point s in S whose distance to x is minimal. Then s is the *projection of x onto S*. See figure B.2.3 for the geometric interpretation. The set of all those points is denoted by $proj_S(x)$:

$$s \in proj_S(x) \iff \big(\{s\} \subset S \cap \overline{B}(x; \| x - s \|)\big) \wedge (S \cap B(x; \| x - s \|) = \emptyset) \tag{B.2.1}$$

Definition B.2.2 (Proximal normal)**.** A vector $\xi \in X$ is a proximal normal to S at s provided there exists $t > 0$ so that $d_S(s + t\xi) = t \| \xi \|$. The set of all proximal normal vectors to S at s is denoted by $N_S^P(s)$. Moreover:

If $s \notin S, N_S^P(s)$ is undefined.

If $s \in S : \forall x \in X \backslash S, s \notin proj_S(x)$, then $N_S^P(s) = \{0\}$

Remark B.2.1. In finite dimensions the existence of closest point is ensured since S is closed. In infinite dimensions existence is more subtle.

Definition B.2.3 (Distance function)**.**

$$d_S : X \to \Re : d_S(x) := inf\{\| x - s \| : s \in S\}.$$

B.2.3 Properties of the proximal normals

Proposition B.2.1. *Let S be a non-empty subset of X, and let $x \in X, s \in S$. The following are equivalent:*

1. $s \in proj_S(x)$;
2. $s \in proj_S(s + t(x - s)) \forall t \in [0, 1]$;
3. $d_S(s + t(x - s)) = t \parallel x - s \parallel, \forall t \in [0, 1]$;
4. $\langle x - s, s' - s\rangle \leq \frac{1}{2} \parallel s' - s \parallel^2 \forall s' \in S$.

Proposition B.2.2. *Let S be a closed subset of* $\mathbb{R}^n$*. Then,* $proj_S(x) \neq \emptyset$ *for all x and the set* $\{s \in proj_S(x) : x \in \mathbb{R}^n \backslash S\}$ *is dense in bdry S.*

Proposition B.2.3 (Proximal normal inequality)**.** *Consider:*

1. *A vector* $\xi \in N_S^P(s) \iff \exists \sigma = \sigma(\xi, s) \geq 0$ *st:*
$$\langle \xi, s' - s\rangle \leq \sigma \parallel s' - s \parallel^2 \forall s' \in S. \tag{B.2.2}$$

2. $\forall \delta > 0 : \xi \in N_S^P(s) \iff \exists \sigma = \sigma(\xi, s) \geq 0 :$
$$\langle \xi, s' - s\rangle \leq \sigma \parallel s' - s \parallel^2 \forall s' \in S \cap B(s; \delta) \tag{B.2.3}$$

Remark B.2.2. From the previous proposition it is easy to conclude that $N_S^P(s)$ is convex. However, it may be neither open nor closed.

Remark B.2.3. The properties of proximal normals are illustrated in figure B.2.3.

- $s_2, s_3, s_4 \in proj_S(x_2)$
- The existence of an infinite number of balls tangential to s_4 with centers lying in the line $x_2 - s_4$ ensures the existence of the proximal normal at point s_4.
- The same reason explains the inexistence of a proximal normal at point s_1.
- Proximal normals at point s5 define a multi-directional cone.

B.2.4 Constrained optimization

The concept of proximal normal generalizes two classical definitions. Consider a closed subset $S \subset \mathbb{R}^n$ that admits a representation of the form:

$$S = \{x \in \mathbb{R}^n : h_i(x) = 0, i = 1, 2, ..., k\}, \quad \text{where} \quad h_i : \mathbb{R}^n \to \mathbb{R} \quad \text{is} \quad C^1 \tag{B.2.4}$$

Proposition B.2.4. *Let* $s \in S$*, where S is given by the previous expression, and assume that the set of vectors* $\{\nabla h_i(s)\}(i = 1, 2, ...)$ *is linearly independent. Then:*

- $N_S^P(s) \subseteq span\{\nabla h_i(s)\}(i = 1, 2, ..., k)$.

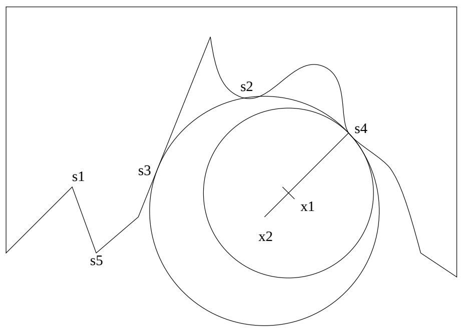


Figure B.1: Proximal normals and projections

- *If, in addition, each h_i is C^2, then equality holds in the previous expression.*

Proposition B.2.5. *Let S be closed and convex. Then:*

- $\xi \in N_S^P$ *iff* $\langle \xi, s' - s \rangle \leq 0, \forall s' \in S$
- *If X is finite dimensional and $s \in bdry(S)$, then $N_S^P(s) \neq \{0\}$.*

B.2.5 Proximal sub-gradients

Definition B.2.4 (Proximal subgradient)**.** Consider $\theta : \mathbb{R}^n \to (-\infty, \infty] \in \mathcal{F}$. A vector $\xi \in \mathbb{R}^n$ is a proximal subgradient of θ at x provided $(\xi, -1) \in N_{epi\theta}^P(x, \theta(x))$. Note that epi θ is a closed subset of $\mathbb{R}^{n+1}$.

Definition B.2.5 (Proximal subdifferential)**.** The set (that can be empty) of all proximal subgradients of $\theta(.)$ at x is denoted by $\partial_P \theta(x)$. If $x \notin \text{dom}\theta$, then $\partial_P \theta(x) = \emptyset$ by definition.

Next we provide an operational definition of the proximal sub-gradient:

Theorem B.2.6. *[Proximal subgradient inequality] Let f* $\in \mathcal{F}$*. Then* $\xi \in \partial_P f(x) \iff$

$$\exists \sigma, \eta : f(y) \geq f(x) + \langle \xi, y - x \rangle - \sigma \parallel y - x \parallel^2, \forall y \in B(x; \eta) \tag{B.2.5}$$

Remark B.2.4 (Geometric interpretation). The proximal subgradient inequality asserts the existence of a parabola $p(y) = f(x) + \langle \xi, y - x \rangle - \sigma \parallel y - x \parallel^2, \forall y \in B(x; \eta)$ which “locally fits” under epi f at (x, f(x)).

Remark B.2.5 (Properties of proximal subdifferentials). These are illustrated in figure B.2.3:

- At point x_1 the proximal subdifferential is a single vector.
- At point x_2 the proximal subdifferential is a closed cone.
- At point x_3 the proximal subdifferential is an unbounded set. To see this, note that the subgradient can be depicted as the set resulting from the projection, into the x domain, of vectors lying in the normal cone and such that the z component is -1. Obviously, the projections of vectors approaching the dashed horizontal line tend to infinity.

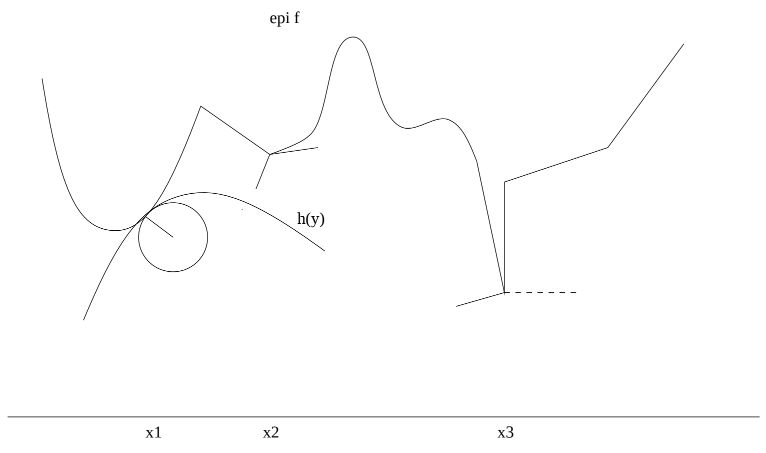


Figure B.2: Proximal sub-gradients

The following corollary establishes relations with Gateaux and Frechet differentials:

Corollary B.2.7. *Let $f \in\mathcal{F}$ and $U \in X$ be open.*

1. *If f is Gateaux differentiable at $x \in U$ then:*

$$\partial_P f(x) \subseteq \{f'_G(x)\}, \quad \textit{where} \quad f'_G(x) \textit{designates the Gateaux derivative} \tag{B.2.6}$$

2. *If $f \in C^2(U)$, then:*

$$\partial_P f(x) = \{f'(x)\}, \forall x \in U \tag{B.2.7}$$

3. *If f is convex, then:*

$$f(y) \geq f(x) + \langle \xi, y - x \rangle, \forall y \in X \tag{B.2.8}$$

Corollary B.2.8. *Suppose $f \in\mathcal{F}$.*

1. *If f has a local minimum at x, then $0 \in \partial_P f(x)$.*
2. *Conversely, if f is convex and $0 \in \partial_P f(x)$, then x is a global minimum of f.*

Remark B.2.6. The proximal subgradient provides a "one-sided" characterization of a lower semi-continuous function. The corresponding notion for upper semi-continuous functions is the proximal supperdifferential $\partial^P f(x)$

Definition B.2.6 (Superdifferential)**.** Let f be upper semicontinuous. Then $\partial^P(x)$ denotes the superdifferential of f at x:

$$\partial^P(x) := -\partial_P(-f)(x) \tag{B.2.9}$$

Remark B.2.7. It is legitimate to ask about the additional information provided by the supperdiferential. The following proposition answers part of the question.

Proposition B.2.9. *Let $U \subset X$ be open, $x \in X, f : U \to \mathbb{R}$ be continuous and both $\partial_P(f)(x)$ and $\partial^P(x)$ be nonempty. If f is Frechet differentiable at x, then $\partial_P(f)(x) = \partial^P(x) = \{f'(x)\}$*

B.2.6 The density theorem

This section is entirely dedicated to the density theorem, an important result stating that the set dom($\partial_P f$) of points in dom f at which at least one proximal subgradient exists is dense in dom f. This result has important operational applications.

Theorem B.2.10 (Density Theorem)**.** *Suppose $f \in\mathcal{F}$. Let $x_0 \in$ dom f, and let $\epsilon > 0$ be given. Then there is a point $y \in x_0 + \epsilon B$:*

1. $\partial_P f(y) \neq \emptyset$
2. $f(x_0) - \epsilon \leq f(y) \leq f(x_0)$.

In particular, dom($\partial_P f$) is dense in dom f.

Remark B.2.8. The proof for the finite dimensional X is quite simple. For the infinite dimensional the proof technique is based on an interesting inductive procedure that generates an infinite sequence of sets satisfying relative set inclusions. The completeness of the underlying space ensures the existence of the minimizer that is generated by the intersection of all sets in the sequence.

B.2.7 Minimization principles

For infinite dimensions a lower semicontinuous function defined on a closed bounded set (with respect to the strong topology) may not attain a minimum. In certain situations, an arbitrary small perturbation of the function will attain a minimum.

The Stegall and Borwein-Preiss minimization principles, presented next, can be derived as a consequence of the proximal analysis of inf-convolutions. Inf-convolutions are used to generate small perturbations to a function. The proof of both principles is a direct application of theorem B.2.11.

Definition B.2.7 (Inf-convolution)**.** The inf-convolution of a function two functions f,g is the function h defined as:

$$h(x) := \inf_{y\in X}[f(y) + g(x-y)] \tag{B.2.10}$$

Remark B.2.9. A special case of inf-convolutions is the function $f_\alpha := \inf_{y\in\mathbb{R}^n}[f(y) + \frac{1}{2\alpha^2} \parallel y - x \parallel^2]$ that is known, in classical convex analysis, as the 'Iosida-Moreau regularization' of the (convex) function f.

Definition B.2.8 (Minimizing sequence)**.** $\{x_i\}$ is a minimizing sequence for an infimum of the type $\inf_{x\in S} g(x)$ provided that all points $x_i \in S$ and satisfy $\lim_{i\to\infty} g(x_i) = \inf_{x\in S} g(x)$

Theorem B.2.11. *Suppose that f $\in\mathcal{F}$ is bounded below by some constant c, and f_α is the inf-convolution of the function f. Then f_α is bounded below by c, and is Lipschitz on each bounded subset of X (and in particular is finite valued). Furthermore, suppose $x \in X$ is such that $\partial_P f_\alpha(x) \neq \emptyset$. Then, there exists a point $\overline{y} \in X$ satisfying the following:*

1. *If $\{y_i\} \subset X$ is a minimizing sequence for the infimum in B.2.10, then $lim_{i\to\infty} y_i = \overline{y}$*
2. *The infimum in B.2.10 is attained uniquely at $\overline{y}$.*

3. *The Frechet derivative $f'_\alpha(x)$ exists and equals $2\alpha(x-\overline{y})$. The proximal subgradient $\partial_P f_{\alpha(x)}$ is the singleton $\{2\alpha(x-\overline{y})\}$.*

4. $2\alpha(x-\overline{y}) \in \partial_P f(\overline{y})$.

Theorem B.2.12 (Stegall's minimization principle)**.** *Let f $\in\mathcal{F}$ and suppose that f is bounded below on the bounded closed set $S \in X$, with $S \cap dom f \neq \emptyset$. Then, there exists a dense set of points x in X having the property that the function $y \mapsto f(y) - \langle x, y\rangle$ attains an unique minimum over S.*

Theorem B.2.13 (Borwein and Preiss minimization principle)**.** *Let f $\in\mathcal{F}$ be bounded below, and let $\epsilon > 0$. Suppose that x_0 is a point satisfying*

$$f(x_0) < \inf_{x\in X} f(x) + \epsilon$$

Then:

$$\forall\lambda, \exists y, z : \|z - x_0\| < \lambda, \|y - z\| < \lambda, f(y) \leq f(x_0)$$

and having the property that the function:

$$x \mapsto f(x) + \frac{\epsilon}{\lambda^2}\|x - z\|^2$$

has an unique minimum at x=y.

Remark B.2.10. The conclusion of this theorem is strongly dependent on the given point x_0.

Remark B.2.11. Another important result along these lines is Ekeland's theorem (see [14]).

Theorem B.2.14. *Let X be a complete metric space with associated metric Δ and let F $\in\mathcal{F}$ be bounded below. If u is a point in X satisfying:*

$$F(u) \leq \mathit{inf}\, F + \epsilon \qquad \text{(B.2.11)}$$

for some $\epsilon > 0$, then, for every $\lambda > 0$ there exists a point v in X such that:

1. $F(v) \leq F(u)$

2. $\Delta(u, v) \leq \lambda$

3. *For all $w \neq v$ in X, one has $F(w) + \frac{\epsilon}{\lambda}\Delta(w, v) > F(v)$.*

Remark B.2.12. The proof of this theorem is an interesting application of the following definition of partial order. To some extend, the technique resembles the one used to prove the density theorem B.2.10.

Definition B.2.9. For any $\alpha > 0$ define a partial ordering $\leq_\alpha$ on $X \times \mathbb{R}$ by (see figure B.2.7):

$$(v_1, r_1) \leq_\alpha (v_2, r_2) \iff r_2 - r_1 + \alpha\Delta(v_1, v_2) \leq 0 \qquad \text{(B.2.12)}$$

Proposition B.2.15 (Properties of the partial order)**.** *:*

- *This relation is reflexive, antisymmetric and transitive.*

- $\forall(v_1, r_1) \in X \times \mathbb{R}$*, the set :*

$$\{(v, r) : (v_1, r_1) \leq_\alpha (v, r)\}$$

is closed.

Lemma B.2.16. *Let S be a closed subset of $X \times \mathbb{R}$ such that, for some scalar m, every element (v,r) of S satisfies $r \geq m$. Then, for every (v_1, r_1) in S, there exists an element $(\overline{v}, \overline{r})$ satisfying $(v_1, r_1) \leq_\alpha (\overline{v}, \overline{r})$ which is maximal in S for the ordering $\leq_\alpha$.*

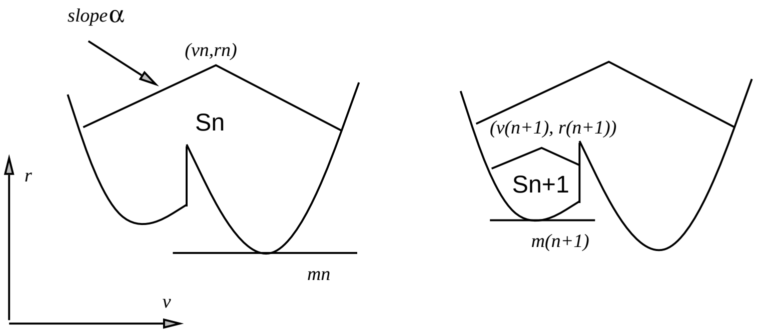


Figure B.3: Proof of Ekeland's theorem

Proof. The intuition for the proof is easily captured by picture B.2.7 . The idea is to construct a sequence $\{(v_n, r_n)\}$ in S by induction, starting with (v_1, r_1), where:

$$S_n := \{(v, r) \in S : (v_n, r_n) \leq_\alpha (v, r)\}$$

$$m_n := \inf\{r : (v, r) \in S_n \quad \text{for some v}\}$$

Clearly $m_n \geq m$. Define (v_{n+1}, r_{n+1}) to be a point of S_n such that:

$$r_n - r_{n+1} \geq \frac{1}{2}(r_n - m_n)$$

The sets S_n are closed and nested. A limiting argument yields the result. □

To prove Ekeland's theorem just take $S := epif$ and apply the lemma with $\alpha = \frac{\epsilon}{\lambda}$

B.2.8 The distance function

The properties of the distance function are now examined in the context of proximal analysis. For the corresponding analysis in terms of generalized gradients see [14]. The following theorems are the geometric analogues of the minimization principles discussed in the previous subsection.

Consider a nonempty closed set S.

Theorem B.2.17. *Suppose $x \notin S$ and $\xi \in \partial_P d_S(x)$. Then there exists a point $\overline{s} \in S$ so that the following holds:*

1. *Every minimizing sequence $\{s_i\} \subset S$ of $\inf_{s \in S} \|s - x\|$ converges to $\overline{s}$.*
2. *The set of closest points $proj_S(x)$ in S to x is the singleton $\{\overline{s}\}$.*
3. *The Frechet derivative $d'_S(x)$ exists, and $\{\xi\} = \partial_P d_S(x) = \left\{ \frac{x - \overline{s}}{\|x - \overline{s}\|} \right\}$*
4. *$\xi \in N_S^P(\overline{s})$*

Remark B.2.13. The proof of this theorem consists of an elegant application of inf-convolutions and theorem B.2.11. The result follows from the consideration of the definitions of:

- Proximal subgradient for the distance function d_S.
- The quadratic inf-convolution of the function

$$I_S(x) : d_S^2(y) = \inf_{z \in X} \{I_S(z) + \|z - y\|^2\}$$

Remark B.2.14. The uniqueness of the projection and of the subgradient has a very interesting geometric interpretation in terms of the properties of the multi-function $proj_S$.

Suppose a given point $x \notin S$ has two projections s_1, s_2 in S. Consider the epigraph of the distance function. Then, there are two directions of decrease for the distance function. The decrease along these lines is linear. These, can be interpreted as directions for the corresponding subgradients (if any). In the space where the epigraph lives this means that the point $(x, d_S(x))$ is the vertex of a polygon. This prevents the existence of quadratic approximations to the epigraph at this point.

Corollary B.2.18. *Suppose $S \subset X$ is closed.*

- *There is a dense set of points in $X \backslash S$ which admits unique closest points in S.*
- *The set of points $s \in bdryS$ for which $N_S^P(s) \neq \{0\}$ is dense in bdry S.*

Remark B.2.15. The next proposition has important applications in solving constrained optimization problems of the form:

$$\min_{s \in S} f(s) \tag{B.2.13}$$

where, $f \in \mathcal{F}$ and S is a closed set S.

The exact penalization technique, described next, transforms this problem into a unconstrained one.

Proposition B.2.19. *Suppose S is a closed subset of X and f is Lipschitz of rank K on an open set U that contains S. Assume that $s \in S$ solves B.2.13. Then the function $x \mapsto f(x) + Kd_S(x)$ attains its minimum over U at x=s. Conversely, if $K' > K$ and $x \mapsto f(x) + Kd_S(x)$ attains a minimum over U at x=s, then s belongs to S and solves B.2.13.*

Remark B.2.16. The last proposition provides a link between the geometric and the functional interpretations.

Proposition B.2.20. *Suppose S is closed and $s \in S$. Then*

$$N_S^P(s) = \{t\xi : t \geq 0, \xi \in \partial_P d_S(s)\} \tag{B.2.14}$$

B.2.9 Lipschitz functions

The Lipschitz property can be characterized in proximal terms.

Theorem B.2.21. *Let $U \subset X$ be open and convex, and let $f \in \mathcal{F}(U)$. Then, f is Lipschitz on U of rank $K \geq 0 \iff$*

$$\|\xi\| \leq K, \forall \xi \in \partial_P f(x), \forall x \in U \tag{B.2.15}$$

Corollary B.2.22. *Let $U \subset X$ be open and convex, and let $f \in \mathcal{F}(U)$. Then f is constant in U $\iff$*

$$\partial_P f(x) \subset \{0\}, \forall x \in U \tag{B.2.16}$$

B.2.10 Limiting calculus

The following concepts are used when the associated proximal normals or proximal subgradients are empty or some enlargement is required.

Definition B.2.10. The limiting subdifferential $\partial_L f(x)$ is defined as:

$$\partial_L f(x) := \{\text{w-lim}\xi_i : \xi_i \in \partial_P f(x_i), x_i \xrightarrow{f} x\} \tag{B.2.17}$$

where w-lim signifies the set of all vectors ξ that can be expressed as the weak limit of some sequence $\{\xi_i\}, \xi_i \in \partial_P f(x_i)$ and $x_i \to x, f(x_i) \to f(x)$

Definition B.2.11. The limiting normal cone to S at $x \in S$ is given by:

$$N_S^L(x) := \{\text{w-lim}\xi_i : \xi_i \in N_S^P(x_i), x_i \xrightarrow{S} x\} \tag{B.2.18}$$

where $x_i \xrightarrow{S} x$ signifies that $x_i \to x$ and $x_i \in S, \forall i$

B.2.11 Relations to subderivatives and Dsubgradients

It is worth investigating relations between subderivative and proximal calculus. These relations can be very useful since results obtained in one field can be used in the other. The main result is due to Subbotin.

Theorem B.2.23 (Subbotin)**.** *Let $f \in \mathcal{F}(\mathbb{R}^n)$, $x \in dom f$, and let E be a nonempty compact convex subset of $\mathbb{R}^n$. Suppose that for some scalar η we have:*

$$Df(x;e) > \eta, \forall e \in E$$

Then, for any $\epsilon > 0$, there exist $z \in x + \epsilon B$ and $\xi \in \partial_P f(z)$ such that:

$$|f(z) - f(x)| < \epsilon; \langle \xi, e\rangle > \eta, \forall e \in E$$

The following result implies that $\partial_D f(x)$ is nonempty on a dense set of dom f.

Proposition B.2.24. $\partial_P f(x) \subset \partial_D f(x)$.

One can now expect some approximation results.

Proposition B.2.25. *Let $\xi \in \partial_D f(x)$. Then for any $\epsilon > 0$ there exists a $z \in x + \epsilon B$ and $\eta \in \partial_P f(x)$ such that:*

$$|f(x) - f(z)| < \epsilon \quad \text{and} \quad \|\xi - \eta\| < \epsilon$$

Proposition B.2.26. *Let $f \in \mathcal{F}(\mathbb{R}^n)$ and $x \in$ dom f. Then*

$$\partial_L f(x) := \{\lim i \to \infty \xi_i : \xi_i \in \partial_D f(x_i), x_i \xrightarrow{f} x\}$$

B.3 Generalized calculus

B.3.1 Introduction

The generalized calculus (see for example [14]) extends classical results to a class of nonsmooth functions.

Here, it will be developed for an arbitrary real Banach space X. First we present the basic results for the class of locally Lipschitz functions. Then, the generalized gradient is characterized in terms of classical gradients when $X = \mathbb{R}^n$. The underlying geometric interpretation is further extended by exploring the elements of the theory of generalized calculus that induce a complete duality between tangency and normality, and functions and sets. The starting point for duality is the generalized gradient of the distance function that leads to one definition of tangent cone to a set, Clarke's tangent cone, and then, by polarity, to the definition of the normal cone. This definition of tangency is then contrasted to the one associated with the Boulingand's or contingent cone. The polar of the contingent cone can be empty. Clarke's tangent cone is a subset of the former and is always nonempty. The sets for which the two notions of tangency coincide are termed regular. We can either choose tangency or the generalized gradient of the distance function as the starting point for duality. This is due to the fact that, in general, it is not known how to make normality the starting point unless the Banach space has additional properties, such as the Hilbert space. Finally, the generalized gradient is related to the proximal analysis constructs when X is a Hilbert space.

B.3.2 Definition and properties

Here, we develop the generalized calculus for the family of locally Lipschitz functions.

Definition B.3.1 (Generalized Directional Derivative)**.** Let $f : X \rightarrow R$ be Lipschitz of rank K near $x \in X$. The generalized directional derivative of f in the direction v, denoted $f^0(x; v)$, is defined as:

$$f^o(x; v) := \lim_{y \to x, t \downarrow 0} sup \frac{f(y + tv) - f(y)}{t}$$

where t is a positive scalar.

Proposition B.3.1 (Properties of the generalized directional derivative)**.** *Let f be Lipschitz of rank K near x. Then:*

1. *The function $v \rightarrow f^o(x; v)$ is finite, positively homogeneous, and subadditive on X and satisfies:*

$$|f^o(x; v)| \leq K\|v\|$$

2. *$f^o(x; v)$ is upper semicontinuous as a function of (x,v) and, as a function of v alone, is Lipschitz of rank K on X.*

3. $f^o(x;-v) = (-f)^o(x;v)$

Remark B.3.1. The definition involves an upper limit only and does not presuppose the existence of any limit.

Remark B.3.2. This notion of derivative is more robust than the traditional one since the base point of the quotient is also involved in the limiting process.

Remark B.3.3. The Hahn-Banach theorem (see [85]), can be invoked to assert the existence of at least one linear functional $\xi : X \to \mathbb{R}$ that it is majorized by the generalized directional derivative (a positively homogeneous and subadditive functional on X), such that $\forall v \in X, f^o(x;v) \geq \xi(v)$. It follows that ξ is bounded and therefore belongs to the dual space X^* of continuous linear functionals on X.

Definition B.3.2 (Support function of a closed convex set)**.** Consider a nonempty closed subset $\Sigma \subset X^*$, where X^* is the dual space of continuous linear functionals on X. Its support function $H_\Sigma : X \to (-\infty, \infty]$ is defined as follows:

$$H_\Sigma(v) := sup\{\langle \xi, v\rangle : \xi \in \Sigma\},$$

Where we denote the value of the linear functional ξ at v by $\langle \xi, v\rangle$

Proposition B.3.2 (Properties of the support function)**.** *:*

1. *Let Σ be a nonempty subset of X^*. Then H_Σ is positively homogeneous, subadditive, and lower semicontinuous.*

2. *If Σ is convex and w^*- closed, then a point $\xi \in X^*$ belongs to Σ iff $H_\Sigma(v) \geq \langle \xi, v\rangle, \forall v \in X$.*

3. *More generally, if Σ and Λ are two non-empty, convex, and w^*- closed subsets of X^*, then $\Lambda \subset \Sigma$ iff $H_\Sigma \leq H_\Lambda, \forall v \in X$.*

4. *If $p : X \to \mathbb{R}$ is positively homogeneous, subadditive and bounded on the unit ball, then there is an uniquely defined nonempty, convex and w^*- compact subset $\Sigma \subset X^*$ such that $p = H_\Sigma$*

Definition B.3.3 (Generalized Gradient)**.** The generalized gradient of the function f at x, denoted $\partial f(x)$, is the nonempty w^*- compact subset of X^* whose support function is $f^o(x;.)$

Proposition B.3.3 (Properties of the generalized gradient)**.** *Let f be Lipschitz of rank K near x. Then:*

1. *$\partial f(x)$ is a nonempty, convex, $weak^* -$ compact subset of X^*, and $\|\xi\|_* \leq K, \forall \xi \in \partial f(x)$*

2. *$\forall v \in X : f^o(x;v) = max\{\langle \xi, v\rangle : \xi \in \partial f(x)\}$*

3. *$\xi \in \partial f(x) \iff f^o(x;v) \geq \langle \xi, v\rangle, \forall v \in X$*

4. *If $\{x_i\} and \{\xi\}$ are sequences in X and X^* such that $\xi_i \in \partial f(x_i)$ for each i, and if x_i converges to x and ξ is a $weak^*$ cluster point of the sequence $\{\xi_i\}$, then we have $\xi \in \partial f(x)$*

5. *If X is finite dimensional, then ∂f is upper semicontinuous at x.*

6. *$\partial f(x)$ is independent of the particular norm on X.*

Example B.3.4. *Let $f(x) = max\{0, x\}$. Then:*

- $f^o(0;v) = max\{0, v\}$.
- $\partial f(0) = [0, 1]$.

B.3.3 Generalized calculus

We will assume that all the given functions are Lipschitz near the point of interest.

Proposition B.3.5. *For any scalar λ, $\partial(\lambda f)(x) = \lambda \partial f(x)$*

Remark B.3.4. As one can infer from the properties of the support function of a convex set, proving an inclusion between convex sets is equivalent to proving an inequality between the corresponding support functions. Moreover, the support function of a sum of sets is the sum of the support functions. Then it is easy to prove the following result:

$$(f+g)^o(x;v) \leq f^o(x;v) + g^o(x;v) \iff \partial(f+g)(x) \subset \partial f(x) + \partial g(x)$$

The following proposition is an extension of this result.

Proposition B.3.6 (Sum Rule)**.** *Let $f_i(i = 1, 2, ..., n)$, be Lipschitz near x, and let $\lambda_i, (i = 1, ..., n)$ be scalars. Then, $f := \sum_{i=1}^{n} \lambda_i f_i$ is Lipschitz near x and we have:*

$$\partial(\sum_{i=1}^{n} \lambda_i f_i)(x) \subset \sum_{i=1}^{n} \lambda_i \partial f_i)(x)$$

Theorem B.3.7 (Lebourg's Mean Value Theorem)**.** *Let x and y belong to X, and suppose that f is Lipschitz on an open set containing the line segment [x,y]. Then, there exists a point u in (x,y) such that:*

$$f(y) - f(x) \in \langle \partial f(u), y - x \rangle$$

Theorem B.3.8 (Chain Rule)**.** *Let $F : X \to \mathbb{R}^n$ be Lipschitz near x, and let $g : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz near F(x). Then the function $f(x') := g(F(x'))$ is Lipschitz near x and we have:*

$$\partial f(x) \subset \overline{co}^* \{\partial \langle \gamma, F(.) \rangle (x) : \gamma \in \partial g(F(x))\},$$

where $\overline{co}^*$ signifies the $w^* - closed$ convex hull.

Proposition B.3.9 (Generalized derivatives and classical derivatives)**.** *Let f be Lipschitz near x. Then:*

1. *If f admits a Gateaux derivative $f'_G(x)$ at x, then $f'_G(x) \in \partial f(x)$.*
2. *If f is continuously differentiable at x, then $\partial f(x) = \{f'(x)\}$.*

For the class of smooth functions the above results assume the form of equalities. One may ask what is the class of nonsmooth functions, if any, that also gives rise to equalities, even when nonsingleton sets are involved. The following results address this question.

Proposition B.3.10 (Generalized derivatives of convex functions)**.** *Let f be convex on U and Lipschitz near $x \in U$. Then the directional derivatives $f'(x; v)$ exist, and we have $f'(x; v) = f^0(x; v)$. A vector ξ belongs to $\partial f(x)$ iff:*

$$f(y) - f(x) \geq \langle \xi, y - x \rangle, \forall y \in U$$

Definition B.3.4 (Regular function)**.** A function f is regular at x provided that f is Lipschitz near x and admits directional derivatives $f'(x; v)$ for all v, with

$$f'(x; v) = f^0(x; v)$$

Remark B.3.5. Regularity sharpens some of the previous calculus rules,as expected:

- Equality holds in the Sum Rule, proposition B.3.6.
- $f + g$ inherits regularity from f and g.
- Equality holds in the chain rule, proposition B.3.8, when g is regular and each $\gamma \in \partial g(F(x))$ has nonnegative components.

B.3.4 The gradient formula in finite dimensions

Theorem B.3.11 (Rademacher)**.** *If a function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz on an open set U, then it is differentiable almost everywhere on U.*

The following theorem asserts that, in $\mathbb{R}^n$, the generalized gradient $\partial f(x)$ of a function f, at a point x, can be generated by the values of $\nabla f(x')$ at nearby points x', at which $f'(x')$ exists. Furthermore, $\partial f(x)$ is blind to sets of measure zero, in the sense that in this process we can ignore sets of measure zero. As usual in Hilbert space we identify $\partial f(x)$ with a subset of $\mathbb{R}^n$.

Theorem B.3.12 (Generalized Gradient Formula)**.** *Let $x \in \mathbb{R}^n$, and let $f : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz near x. Let Ω be any subset of zero measure in $\mathbb{R}^n$, and let Ω_f be the set of points in $\mathbb{R}^n$ at which f fails to be differentiable. Then:*

$$\partial f(x) := co\{lim \nabla f(x_i) : x_i \to x, x_i, \notin \Omega, x_i \notin \Omega_f\}$$

The following figure provides the corresponding geometric interpretation.

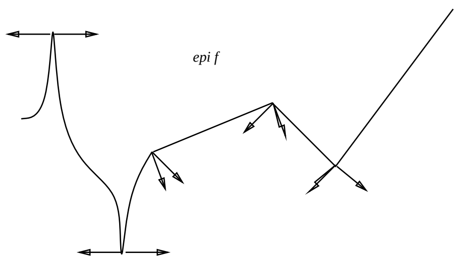


Figure B.4: Generalized gradients for a function f when $X = \mathbb{R}^n$

Proposition B.3.13.

$$f^o(x; v) := \lim_{y \to x} sup\{\nabla f(y).v : y \notin \Omega \cup \Omega_f\}$$

The expression for the derivative of the distance function $d_S(.)$ (definition B.2.3) to a closed set S in $\mathbb{R}^n$ is similar to the corresponding one for the proximal subgradient. This property will be used later.

Proposition B.3.14 (Derivative of the distance function)**.** *Let $d'_S(x)$ exist and be different from 0. Then $x \notin S$, $proj_S(x)$ is a singleton $\{s\}$, and*

$$\nabla d_S(x) = \frac{(x - s)}{\|x - s\|}$$

B.3.5 Clarke's tangent and normal cones

In this section generalized calculus is applied in order to define geometric constructs for a nonempty closed subset S of X. These are based on the properties of the distance function $d_S(x)$, a globally Lipschitz function that completely characterizes a closed set S. We use the generalized directional derivative of $d_S(x)$ to define the tangent cone $T_S^C(x)$ to a set S at a point x. The definition of normal cone follows from the consideration of polarity.

Definition B.3.5 (Tangent direction to a set S)**.** A direction v tangent to the set S at $x \in S$ is defined as:

$$d_S^o(x; v) \leq 0$$

Definition B.3.6 (Tangent cone to S at x)**.** The tangent cone to S at x, denoted $T_S^C(x)$ and also called Clarke's tangent cone, is the set:

$$T_S^C(x) := \{v \in X : d^o(x; v) \leq 0\}$$

Proposition B.3.15 (Properties of the tangent cone)**.** *:*

- $0 \in T_S^C(x)$.
- $T_S^C(x)$ *is a closed and convex cone.*

The following proposition shows that tangency does not depend on the choice of equivalent norms for X as does the distance function.

Proposition B.3.16 (Clarke's characterization of tangency)**.** *An element v of X is tangent to S at x iff, for every sequence x_i in S converging to x and sequence t_i in $(0, \infty)$ decreasing to 0, there exists a sequence v_i in X converging to v such that $x_i + t_i v_i \in S$ for all i.*

Proof. The proof of the first implication is constructive and provides an important geometrical insight.

Suppose that $v \in T_S(x)$ and the sequences $x_i \to x$ (with $x_i \in S$, $t_i \downarrow 0$ are given. Then, we need to produce a sequence $v_i \to v$. Since, by definition of v, $d_S^o(x; v) = 0$, we have:

$$\lim_{i\to\infty} \frac{d_S(x_i + t_i v) - d(x_i)}{t_i} = \lim_{i\to\infty} \frac{d_S(x_i + t_i v)}{t_i} = 0$$

Let $s_i \in S$ satisfy:

$$\|x_i + t_i v - s_i\| \leq d_S(x_i + t_i v) + \frac{t_i}{i}$$

Set $v_i = \frac{s_i - x_i}{t_i}$.
Then $\|v - v_i\| \to 0$ and $x_i + t_i v_i = s_i$ as required.

The other implication is easy to prove. □

Remark B.3.6. The following figure provides a graphical interpretation of this characterization of tangency, that may seem a bit surprising at first. Consider the closed set S, the point x at the boundary and the vector v. v does not qualify for tangent vector. To see this consider a sequence of points x_i in the boundary of S, as shown in the picture. Then, the construction used in the proof for generating a sequence of $v_i \to v$ provides a sequence of vectors that does not converge to v. In fact these vectors can be parallel to the boundary of S at each of the points in the sequence x_i.

The analysis of $d_S^o(x;v)$ provides another perspective of the same fact. Consider again the vector v and the points x_i. Then:

$$d_S^o(x;v) = \sup_{y\to x, t\downarrow 0} \frac{d_S(y+tv)-d(y)}{t} \geq \sup_{x_i\to x, t\downarrow 0} \frac{d_S(x_i+tv)-d(x_i)}{t} =$$

$$\sup_{x_i\to x, t\downarrow 0} \frac{d_S(x_i+tv)}{t}$$

But this last limit can be made constant, and strictly greater than zero, by an appropriate choice of x_i, t_i.

The fact that $d_S^o(x;v)$ is given by a *lim sup* is essential for the result.

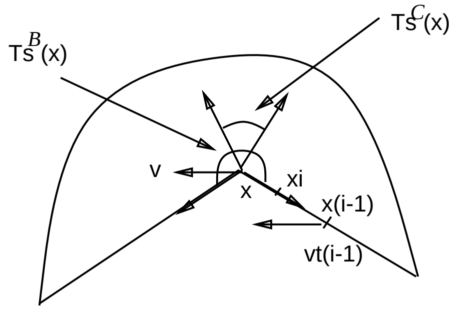


Figure B.5: Clarke and Boulingand tangent cones

One may be tempted to establish relations to the notions of tangent spaces and normal spaces for differentiable manifolds. The following definition provides the enabling mechanism.

Definition B.3.7 (Polar cone)**.** Consider a cone T with apex at the origin. The polar of T denoted T^P, is defined as follows:

$$T^P := \{\xi \in X^* : \langle \xi, v\rangle \leq 0, \forall v \in T\}$$

Definition B.3.8 (Normal cone)**.** The normal cone, also called Clarke's normal cone, to a set S at x, denoted $N_S(x)$, is the following set:

$$N_S(x) := T_S^C(x)^P := \{\xi \in X^* : \langle \xi, v\rangle \leq 0, \forall v \in T_S^C(x)\}$$

Remark B.3.7. The convexity of the tangent cone ensures the nonemptiness of the normal cone.

Proposition B.3.17 (Properties of the normal cone)**.** *:*

- *$N_S(x)$ is a w^*- closed convex cone.*
- *$N_S(x) = cl^*\{\cup_{\lambda\geq 0}\lambda\partial d_S(x)\}$.*
- *$T_S(x)$ is the polar of $N_S(x)$.*

As one could expect, there is a special representation for the normal and tangent cones when the set S is convex.

Proposition B.3.18. *Let S be convex. Then:*

- $T_S(x) = cl\{\lambda(s-x) : \lambda \geq 0, s \in S\}$
- $N_S(x) = \{\xi \in X^* : \langle \xi, x' - x \rangle \leq 0, \forall x' \in S\}$.

The duality function-sets is established in the following theorem.

Theorem B.3.19 (Duality)**.** *Let f be Lipschitz near x. Then:*

- $T_{epif}(x, f(x)) = epif^0(x; .)$
- $\xi \in \partial f(x) \iff (\xi, -1) \in N_{epif}(x, f(x))$.

B.3.6 Boulingand tangent cone and regular sets

Definition B.3.9 (Boulingand or contingent cone to a closed set S)**.** It is denoted as $T_S^B(x)$, and it is defined as:

$$T_S^B(x) := \{\lim_{i\to\infty} \frac{x_i - x}{t_i} : x_i \overset{S}{\to} x, t_i \downarrow 0\}$$

where, $x_i \overset{S}{\to} x$ means that $x_i \in S, \forall i = 1, 2, ..$ and $\lim_{i\to\infty} x_i = x$.

An equivalent definition is:

$$T_S^B(x) := \{v : \forall \epsilon > 0, \exists t \in (0, \epsilon), w \in v + \epsilon B : x + tw \in S\}$$

Remark B.3.8. Some of the properties of the Boulingand's cone are exhibited in figure B.3.5:

1. The strict inclusion relation: $T_S^C(x) \subset T_S^B(x)$.
2. The polar cone of $T_S^B(x)$ is an empty set.

Hence, in this case, and contrary to Clarke's cone, the Boulingand's cone does not provide normal directions to the set S at point x.

Proposition B.3.20 (Properties of the Boulingand tangent cone)**.** *:*

1. $v \in T_S^B(x) \iff \lim_{t\downarrow 0} inf(\frac{d_S(x+tv)}{t}) = 0$
2. $T_S^C(x) \subset T_S^B(x)$
3. *If X is a Hilbert space then:* $T_S^B(x) \subset (N_S^C(x))^P$

Remark B.3.9 (Relations to the D-tangent cone $T_S^P(x)$). Note that:

$$\lim_{t\downarrow 0} inf(\frac{d_S(x+tv)}{t}) = Dd_S(x; v)$$

This is easy to conclude from the application of proposition B.1.1 to the Lipschitz function $d_S(x)$. In fact, if, in the definition of Clarke's cone we use the subderivative of the distance function we get the Boulingand's cone. Now, it becomes clear the role of $liminf$ and $limsup$ in the corresponding constructs.

Definition B.3.10 (Regular set)**.** A set S is regular provided that

$$\forall x \in S, T_S^B(x) = T_S^C(x)$$

Which sets are regular and what is the relation to regular functions?

Proposition B.3.21. *A convex set is regular at each of its points.*

Proposition B.3.22. *Let f be Lipschitz near x. Then f is regular at x iff epi f is regular at (x,f(x)).*

Set regularity provides more "exact" estimates in the previous formulas for tangent and normal cones.

Remark B.3.10. Clarke's tangent cone plays an important role in defining the class of normal directions to a closed set S. In this sense it is more useful than the Boulingand's tangent cone. This concept of normality has many relevant applications:

- Provides a test for determining if the flow of a vector field in the boundary of a closed set, is such that the boundary will be crossed.
- Can be used for defining extensions of the Hahn-Banach theorem for some classes of nonconvex sets.

B.3.7 Relationship to proximal analysis

Now suppose that X is a Hilbert space, with $\|x\| = \langle x, x, \rangle^{\frac{1}{2}}$ for an inner product, Then, $\partial f(x)$ and $N_S(x)$ are generated by the weak limits of their proximal counterparts and identified with subsets of X.

Theorem B.3.23. *Let X be a Hilbert space.*

- *If f is Lipschitz near x, then:*
$$\partial f(x) = \overline{co}\{w\text{-}lim_{i\to\infty}\xi_i : \xi_i \in \partial_P f(x_i), x_i \to x\}$$
- *If S is a closed subset of X containing x, then:*
$$N_S^C(x) = \overline{co}\{w\ lim_{i\to\infty}\xi_i : \xi_i \in N_P f(x_i), x_i \to x\}$$

Remark B.3.11. The theorem asserts that in a Hilbert space we have $N_S^C(x) = \overline{co}N_S^L(x)$ and when f is Lipschitz near x $\partial f(x) = \overline{co}\partial_L f(x)$.

Remark B.3.12. This theorem validates the use of some results from the previous chapter in the Hilbert space setting.

Proposition B.3.24. *Let S be a subset of $\mathbb{R}^n$ given as follows:*
$$S := \{x \in \mathbb{R}^n : f_j(x) = 0, j = 1, 2, ..., k\},$$
where each $f_j : \mathbb{R} \to \mathbb{R}$ is locally Lipschitz and admits one-sided directional derivatives $f_j'(x; v)$ for each v. Then:
$$T_S^B(x) \subset \{v \in \mathbb{R}^n : f_j'(x; v) = 0, j = 1, ..., k\}$$
If, in addition, each f_i is C^1 near x and the vectors $\{f_j'(x)_{j=1}^k\}$ are linearly independent, then:
$$\{v : \langle f_j'(x), v\rangle = 0, j = 1, ..., k\} \subset T_S^C(x)$$
In this case, equality holds in both estimates, and the set is regular at S.

B.3.8 Relationship to subderivatives

Proposition B.3.25. *Let $f \in \mathcal{F}(\mathbb{R}^n)$ and $x \in$ dom f. Then, if f is Lipschitz near x, then:*

$$\partial_D f(x) \subset \partial_L f(x) \subset \partial_C f(x)$$

with equality iff f is regular at x.

Remark B.3.13. From the above proposition it is easy to conclude that it is not true in general that $Df(x; v) = sup\{\langle \xi, v \rangle : \xi \in \partial_P f(x)\}$. This is due to the fact that $\partial_D f(x) \subset \partial_C f(x)$. The observation that Df(x;v) is not subadditive, thus preventing the application of the Hahn-Banach theorem, is another way of concluding this.

Proposition B.3.26. *Let S be a closed nonempty subset of $\mathbb{R}^n$. Then:*

$$N_S^D(x) \subset N_S^L(x) \subset N_S^C(x)$$

Equality holds if S is regular at x.

B.3.9 Tangents and interiors

Since Clarke's tangent cone is nonempty, the meaning of being minimal, in the sense of having an empty interior, at some point x leads to an interesting local characterization of sets. In fact, if $intT_S^C(x)$ is nonempty, then it is possible to locally fit a small cone inside the set.

Remark B.3.14 (Relations to local attainability of a set). This geometric property of a set will provide necessary conditions for local attainability of the set by some trajectories of a controlled differential inclusion. This intuition will be further developed by the following results.

Definition B.3.11 (Wedged set)**.** A closed nonempty set S of $\mathbb{R}^n$ is termed wedged at x if:

$$intT_S^C(x) \neq \emptyset$$

Theorem B.3.27. *A vector $v \in \mathbb{R}^n$ belongs to $intT_S^C(x)$ iff:*

$$\exists \epsilon > 0 : y \in x + \epsilon B, w \in v + \epsilon B, t \in [0, \epsilon) \Rightarrow d_S(y + tw) \leq d_S(y)$$

From the graphical interpretation of figure B.3.5 it is clear that the tangent cone will become smaller and smaller when the outer angle at x tends to 2π. The underlying geometric interpretation provides a local characterization of the set S at the point s. Next, equivalent characterizations are provided by using other geometric constructs.

Definition B.3.12 (Wedge)**.** A set $W(v; \epsilon) \subset \Re^n$ is a wedge of axis v and radius ϵ if:

$$W(v; \epsilon) := \{tw : t \in [0, \epsilon), w \in v + \epsilon B\}$$

Proposition B.3.28 (Properties of wedged sets)**.** *:*

1. *S is wedged at x iff there exists a wedge $W(v;\epsilon)$ such that:*

$$y + W(v;\epsilon) \subset S, \forall y \in S \cap B(x;\epsilon)$$

2. *If S is wedged at x, then $intS \neq \emptyset$ and $x \in cl(int(S))$. If S is wedged at each of its points, then $S = cl(intS)$.*
3. *If $v \in intT_S^C(x)$, then $v \in T_S^C(x')$ for all x' near x.*
4. *If $T_S^C(x) = \mathbb{R}^n$ then $x \in intS$.*

Proposition B.3.29. *If a set S is wedged at x, then $T_S^C(.)$ is lower semicontinuous at x.*

Proposition B.3.30. *$T_S^C(.)$ is lower semicontinuous at x iff $N_S^C(.)$ is graph-closed at x.*

Definition B.3.13 (Pointed cone)**.** A cone $K \in \mathbb{R}^n$ is called pointed if it contains no two nonzero elements whose sum is zero.

Proposition B.3.31. *A convex cone K in $\mathbb{R}^n$ has nonempty interior iff its polar K^o is pointed.*

The following corollary summarizes some of the previous results.

Corollary B.3.32. *If $N_S^C(x)$ is pointed, then $N_S^C(x)$ is graph-closed at x, $T_S^C(.)$ is lower semicontinuous at x, and S is wedged at x.*

B.3.10 The general relation between T_S^C and T_S^B

The following theorem asserts that a vector v lying in $T_S^C(x)$ also lies in $T_S^B(x')$ for x' near x.

Theorem B.3.33.

$$v \in T_S^C(x) \iff \lim_{x' \overset{S}{\to} x} \sup d(v, T_S^B(x')) = 0$$

This theorem provides a subtle link to regularity if we consider an alternative definition of lower semicontinuity.

Proposition B.3.34. *Consider a multifunction $F : X \to X$. Let $\Delta :=$ dom F. Then F is lower semicontinuous at $x \in \Delta$ iff:*

$$\forall v \in F(x), \lim_{x' \overset{\Delta}{\mapsto} x} \sup d(v, F(x)) = 0$$

Now the proof of the following corollary is trivial.

Corollary B.3.35. *S is regular at x iff $T_S^B(.)$ is lower semicontinuous at x.*

Appendix C

Underwater vehicle model

This section discusses how to approximate a nonlinear model of underwater vehicles by a kinematic model of a unicycle. This is done for a torpedo-shaped vehicle and under some simplifying assumptions.

Autonomous underwater vehicles (AUV's) are best described as nonlinear systems (see [43] for details). Two coordinate frames are considered: body-fixed and earth-fixed (see Figure below).

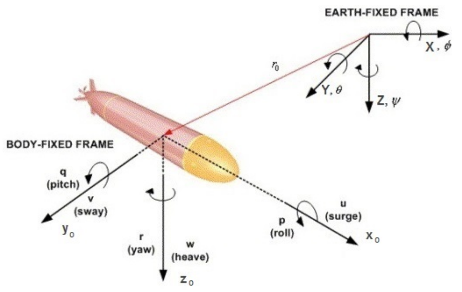


Figure C.1: Body-fixed and earth-fixed coordinate frames.

In what follows, the notation from the Society of Naval Architects and Marine Engineers (SNAME) [66] is used. The motions in the body-fixed frame are described by 6 velocity components $[u, v, w, p, q, r]$ respectively, surge, sway, heave, roll, pitch, and yaw, relative to a constant velocity coordinate frame moving with the ocean current. Define $\nu = [\nu_1^T, \nu_2^T]$ with $\nu_1 = [u, v, w]^T$ $\nu_2 = [p, q, r]^T$. The corresponding forces and moments in the body-fixed frame are $\tau = [\tau_1^T, \tau_2^T]$; $\tau_1 = [X, Y, Z]^T$ $\tau_2 = [k, M, N]^T$. The six components of position and attitude in the earth-fixed frame are $\eta = (\eta_1, \eta_2) = [x, y, z, \phi, \theta, \psi]$. The earth-fixed reference frame can be considered inertial for the AUV.

	Definition	*Force/Moment*	*linear/angular speed*
Motion			
x-direction	surge	X	u
y-direction	sway	Y	v
z-direction	heave	Z	w
Rotation about			
x-axis	roll	K	p
y-axis	pitch	M	q
z-axis	yaw	N	r

Table C.1: Forces, moments and velocities in body-fixed coordinates

Definition	*Position/Euler angle*
Motion	
x-direction	x
y-direction	y
z-direction	z
Rotation about	
x-axis	ϕ
y-axis	θ
z-axis	ψ

Table C.2: Forces and velocities in earth-fixed coordinates

The velocities in both reference frames are related through the Euler angles transformation

$$\dot{\eta} = J(\eta_2)\nu \tag{C.0.1}$$

In the body-fixed frame the nonlinear equations of motion are:

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{C.0.2}$$

$$\dot{\eta} = J(\eta_2)\nu \tag{C.0.3}$$

where M is the inertia and added mass matrix of the vehicle, $C(\nu)$ is the Coriolis and centripetal matrix, $D(\nu)$ is the damping matrix, $g(\eta_2)$ is the vector of the restoring forces and moments, and τ is the vector of the body-fixed forces from the actuators.

We consider the model of a torpedo-shaped vehicle of the LAUV class (see Figure A.1). This AUV is not fully actuated. There is a propeller for actuation in the longitudinal direction (surge, in the naval terminology) and fins for lateral and vertical actuation. The effect of the fins depends on the longitudinal velocity of the vehicle (for zero speed they do not provide actuation).

The mechanical configuration of the AUV leads to a simpler dynamic model. The body-fixed forces from the actuators τ depends only on 3 parameters: propeller velocity n ($0 < n \leq n_{max}$), horizontal fin inclination δ_s ($-\delta_{smax} \leq \delta_s \leq \delta_{smax}$) and vertical fin inclination δ_r ($-\delta_{rmax} \leq \delta_r \leq \delta_{rmax}$). The dynamics of the thruster motor and fin servos are generally much faster than the remaining dynamics therefore, for the purposes of this work, they can be excluded from the model.

System identification for autonomous underwater vehicles is quite difficult and expensive for two reasons: the large number of model parameters (matrix coefficients) and the complexity of the experimental setup for system's identification. In our developments we use a set of coefficients based on the results from [79] and on our field experiments.

We are concerned with operations on the horizontal plane. This restricts the motions of the AUV to planar motions at constant depth. We assume the existence of controllers that stabilize vehicle's depth and pitch, i.e., w converges to a number close to zero (which in practice is not equal to zero due to the required pitch to compensate for vehicle's buoyancy) and q converges to zero. The roll rate p converges to zero due to the restoring moment of the vehicle and the roll angle ϕ converges to a value which depends on the thruster speed. In general, the pitch and roll angles can be made very small by design of the physical configuration. Under these assumptions, and also because of the shape of the AUV, the approximated nonlinear model becomes [43]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u\cos(\psi) - v\sin(\psi) \\ u\sin(\psi) + v\cos(\psi) \\ r \\ f_u(\mathbf{v}, \tau) \\ f_v(\mathbf{v}, \tau) \\ f_r(\mathbf{v}, \tau) \end{bmatrix} \tag{C.0.4}$$

with

$$\begin{aligned}
f_u(\mathbf{v}, \tau) =&(m - X_{\dot{u}})^{-1}(X_{uu}u|u| + X_{vr}vr + \\
&X_{rr}r^2 + X_p(n)) \\
f_v(\mathbf{v}, \tau) =&(m - Y_{\dot{v}})^{-1}(Y_{vv}v|v| + Y_{uv}uv + \\
&(Y_{ur} - m)ur + Y_{rr}r|r| + Y_{uudr}u|u|\delta_r) \\
f_r(\mathbf{v}, \tau) =&(I_{zz} - N_{\dot{r}})^{-1}(N_{vv}v|v| + N_{uv}uv + \\
&N_{ur}ur + N_{rr}r|r| + N_{uudr}u|u|\delta_r)
\end{aligned}$$

For the purpose of motion planning, this model can be further simplified. Physical experiments and simulations with the non-linear model show that, with constant actuation, the steady state radius of curvature is constant and practically independent of the surge velocity. The curvature is mainly determined by the angular displacement of the rudder fin (which would be modeled by c in the kinematic model). In practice, if the vehicle sets constant angular actuation (e.g. a fixed angular position for the rudder of the AUV), the motion of the vehicle will after a very short transient period converge to circle. In practice, a constant angular actuation (e.g. a fixed angular position for the rudder of the AUV) will result in a trajectory that will converge, after a very short transient period, to a circle. Moreover, the ratio $\frac{v}{u}$ will be approximately constant. By a simple trigonometric transformation the first three equations of system C.0.4 become

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \sqrt{u^2+v^2}(\cos(\psi) + \arctan(\frac{v}{u})) \\ \sqrt{u^2+v^2}(\sin(\psi) + \arctan(\frac{v}{u})) \\ r \end{bmatrix} \tag{C.0.5}$$

which as v goes to zero, or with an adequate change of variables, become those of the unicycle model. From the last equation of system C.0.4, and taking in account the constant ratio between u and v it is possible to verify that, in steady state, r is directly proportional to u and directly related to δ_r.

A slow varying water current with velocity $v_d < v_{max}$ and direction θ_d can be considered as an additive disturbance on the vehicle velocity: the motion of the vehicle will be made with relation to the moving column of water, as stated previously.

These are the reasons why the kinematic model of an unicycle, presented in Section 6.2 of Chapter 6 can be considered an acceptable approximation for trajectory planning purposes. Marine and aerial vehicles do not posses the sideslip constraint, i.e., they move sideways (sway velocity on the AUV model). However, this motion is encompassed by the considered radius of curvature. If operation at constant speed is considered, the main difference is the fact that angular speed is allowed to vary instantaneously on the kinematic model while that is not possible on the physical system (and neither on the nonlinear model). Therefore unions between segments and arcs would not be perfectly tracked by a real vehicle. The modeling error can be minimized by considering a larger radius of curvature but the imperfection will still be noticeable for small angular displacements. Moreover, the main objective is that the vehicles reach the destination at the desired time. This goal can be achieved with minimal deviations from the ideal trajectory if some planning slack is allowed (e.g., considering $v'_{max} = v_{max} - \delta$).

In Chapter 6 we use v for the longitudinal velocity (replacing u in the SNAME notation) and ω for the angular velocity (assuming planar motion this replaces r in the SNAME notation).

Bibliography

[1] S. Spry A. Girard and J. K. Hedrick. Real-time embedded hybrid control software for intelligent cruise control applications. *IEEE Robotics and Automation Magazine – Special Issue on ITS*, 12(1):22–28, 2005.

[2] M. Aicardi, G. Casalino, G. Indiveri, A. Aguiar, P. Encarnação, and A. Pascoal. A planar path following controller for underactuated marine vehicles. In *Proc. 9th Mediterranean Conference on Control and Automation*, 2001.

[3] R. Bachmayer and N. E. Leonard. Vehicle networks for gradient descent in a sampled environment. In *Proceedings of IEEE Conference on Decision and Control*, pages 112–117, 2002.

[4] Radhakisan Baheti and Helen Gill. *The Impact of Control Technology*, chapter Cyber-Physical Systems, pages 161–166. IEEE Control Systems Society, 2011.

[5] Alberto-László Barabási. *LINKED: The New Science of Networks*. Plume, New York, 2003.

[6] Martino Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhauser, 1997.

[7] S. Bayraktar, G. Fainekos, and G. J. Pappas. Experimental cooperative control of unmanned aerial vehicles. In *Proceedings IEEE Conference Decision and Control*. IEEE Control Society, 2004.

[8] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

[9] R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[10] A. Bensoussan and J. L. Menaldi. Hybrid control and dynamic programming. *Dynamics of Continuous Discrete and Impulsive Systems*, 3(4):395–442, 1997.

[11] Michael Branicky. *Studies in Hybrid Systems: Modeling, Analysis and Control*. PhD thesis, MIT, 1995.

[12] E. Burian, D. Yoerger, A. Bradley, and H. Singh. Gradient search with autonomous underwater vehicles using scalar measurements. In *IEEE Symp. Autonomous Underwater Vehicle Technology*, pages 86–89, 1996.

[13] L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of the First international Conference on Foundations of Software Science and Computation Structure*, Lecture Notes in Computer Science: 1378, pages 140–155. Springer-Verlag, 1988.

[14] F. H. Clarke. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

[15] F. H. Clarke, Y.S. Ledyaev, R. J. Stern, and P.R. Wolenski. Qualitative properties of trajectories of control systems: A survey. *Journal of Dynamical Control*, (1):1–48, 1995.

[16] F. H. Clarke, Y.S. Ledyaev, and R.J. Stern. Asymptotic stability and smooth lyapounov functions. *submitted*.

[17] D. Van Cleave. Trends and technologies for uninhabited autonomous vehicles. In T. Samad and G. Balas, editors, *Software-Enabled Control: Information Technology for Dynamical Systems*. IEEE Press/John Wiley and Sons, 2002.

[18] B. T. Clough. Metrics, schmetrics! how the heck do you determine a uav's. In *Performance Metrics for Intelligent Systems (PerMIS) conference*, 2002.

[19] J. Cortés, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transaction on Automatic Control*, 2004. To appear.

[20] D. E. Culler and H. Mulder. Smart sensors to network the world. *Scientific American*, 2004.

[21] T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous ocean sampling networks. *Oceanography*, 6(3):86–94, 1993.

[22] T. Curtin and J. G. Bellingham. Guest editorial: Autonomous ocean-sampling networks. *IEEE Journal of Oceanic Engineering*, 26(4):423, 2001.

[23] R. D'Andrea. Robot soccer: a platform for systems engineering. *Computers in Education Journal*, 10(1):57–61, 2000.

[24] R. D'Andrea and R. Murray. The roboflag competition. In *Proceedings of the American Controls Conference*, pages 650–655. IEEE, 2003.

[25] J. Borges de Sousa. Optimal path coordination problems. In *Proceedings of the IEEE Conference on Decision and Control*. IEEE, 2008.

[26] J. Borges de Sousa and A. Deshpande. Real-time multi-agent coordination using diadem: applications to automobile and submarine control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1769–74. IEEE, 1997.

[27] J. Borges de Sousa, A. Girard, and K. Hedrick. Real-time hybrid control of mobile offshore base scaled models. In *Proceedings of the American Control Conference*, 2000.

[28] J. Borges de Sousa, K. H. Johansson, J. Estrela da Silva, and Alberto Speranzon. A verified hierarchical control architecture for coordinated multi-vehicle operations. *International Journal of Adaptive Control and Signal Processing - Special Issue: Autonomous and adaptive control of vehicles in formation*, 21(2-3):159 – 188, 2005.

[29] J. Borges de Sousa, K. H. Johansson, A. Speranzon, and J. Silva. A control architecture for multiple submarines in coordinated search missions. In *Proceedings of IFAC World Congress*, 2005.

[30] J. Borges de Sousa, K. H Johansson, A. Speranzon, and J. Silva. A control architecture for multiple submarines in coordinated search missions. In *Proceedings of the 2005 IFAC conference*. IFAC, 2005.

[31] J. Borges de Sousa, B. Maciel, and F. Lobo Pereira. Sensor systems on networked vehicles. *Networks and Heterogeneous Media – American Institute of Mathematical Sciences*, 4(2):223– 247, 2009.

[32] J. Borges de Sousa and F. Lobo Pereira. A generalized vehicle-based control architecture for multiple auvs. In *Proceedings of the OCEANS '95 MTS/IEEE*, pages 1643–50. IEEE, 1995.

[33] J. Borges de Sousa and F. Lobo Pereira. *Coordination Control of Distributed Systems*, chapter Coordination challenges in networked vehicle systems: are we missing something? Springer Verlag, 2014.

[34] J. Borges de Sousa, T. Simsek, and P. Varaiya. Task planning and execution for uav teams. In *Proceedings of the IEEE Conference on Decision and Control*. IEEE, 2004.

[35] J. Borges de Sousa, Pravin Varaiya, and Tunc Simsek. Distributed control of teams of unmanned air vehicles. In *Proceedings of Mathematical Theory of Networks and Systems Conference*, 2004.

[36] A. Deshpande, A. Gollu, and L. Semenzato. The shift programming language and run-time system for dynamic networks of hybrid automata. Technical Report UCB-ITS-PRR-97-7, California PATH, 1997.

[37] A. Deshpande and P. Varaiya. Viable control of hybrid systems. In *Hybrid Systems II*, pages 128–147. Springer, 1995.

[38] S. Dharmatti and M. Ramaswamy. Hybrid control systems and viscosity solutions. *SIAM Journal of Control and Optimization*, 44(4):1259–1288, 2005.

[39] F. H. Clarke et. al. *Nonsmooth Analysis and Control Theory*. Springer, 1998.

[40] G. Fainekos, H. Kress-Gazit, and G. Pappas. Hybrid controllers for path planning : a temporal logic approach. In *Proceedings IEEE Conference Decision and Control*. IEEE Control Society, 2005.

[41] E. Fiorelli, P. Bhatta, and N. E. Leonard. Adaptive sampling using feedback control of an autonomous underwater glider fleet. In *Proc. 13th Int. Symp. on Unmanned Untethered Submersible Technology (UUST)*. IEEE, 2003.

[42] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand. Cooperative control for multiple autonomous uavs searching for targets. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 2823–28. IEEE Control Society, 2002.

[43] T. I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley and Sons Ltd., 1994.

[44] A. Girard, K. Hedrick, and J. Borges de Sousa. A hierarchical control architecture for mobile offshore bases. In *Proceedings of Third International Workshop on Very Large Floating Structures*, pages 447–456, 1999.

[45] A. Girard, J. B. Sousa, and K. Hedrick. An overview of emerging results in networked multi-vehicle systems. In *Proceedings of the Decision and Control Conference*, Orlando, USA, 2001.

[46] A. R. Girard, J. Borges de Sousa, and J. K. Hedrick. A selection of recent advances in networked multi-vehicle systems. *Proceedings of the I MECH E Part I Journal of Systems & Control Engineering*, (I1):1–14, 2005.

[47] D. N. Godbole, J. Lygeros, and S. Sastry. Hierarchical hybrid control: An ivhs case study. In A. Nerode P. Antsaklis and S. Sastry, editors, *Hybrid Systems II*, LNCS, pages 166–90. Birkhauser, 1995.

[48] Gwyn Griffiths, editor. *Technology and applications of Autonomous Underwater Vehicles*. Ocean Science and Technology Volume 2. Taylor & Francis Group, 2003.

[49] G. Hagan. *Glossary Defense Acquisition Acronyms and Terms*. Defense Acquisition University Press, Fort Belvoir, Virginia 22060-5565, 2009.

[50] J. K. Hedrick, M. Tomizuka, and P. Varaiya. Control issues in automated highway systems. *IEEE Control Systems Magazine*, 14(6):21–32, 1994.

[51] J. P. Hespanha, H. J. Kim, and S. Sastry. Multiple-agent probabilistic pursuit–evasion games. In *IEEE Conference on Decision and Control*, volume 3, pages 2432–2437, 1999.

[52] IEEE, editor. *IEEE standard for application and management of the systems engineering process*. IEEE, 1999.

[53] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[54] J. Jang and C. Tomlin. Autopilot design for the stanford dragonfly uav: Validation through hardware-in-the-loop simulation. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. AAIAA, 2001.

[55] K. H. Johansson, A. Speranzon, and S. Zampieri. On quantization and communication topologies in multi-vehicle rendezvous. In *Proceedings of IFAC World Congress*, 2005.

[56] L. C. Kendall, D. K. Costello, H. Warrior, L. C. Langebrake, W. Hou, J. T. Patten, and E. Kaltenbacher. Ocean-science mission needs: Real-time auv data for command, control and model inputs. *IEEE Journal of Oceanic Engineering*, 26(4):742–751, 2001.

[57] D. B. Kilfoyle and A. B. Baggeroer. The state of the art in underwater acoustic telemetry. *IEEE Journal of Oceanic Engineering*, 25(1):4–27, 2000.

[58] A. N. Krasovskii. *Control under lack of information*. Birkhauser, 1995.

[59] N. N. Krasovskii and A. I. Subbotin. *Game-theoretical control problems*. Springer-Verlag, 1988.

[60] A. B. Kurzhanskii and P. Varaiya. Dynamic optimization for reachability problems. *Journal of Optimization Theory & Applications*, 108(2):227–51, 2001.

[61] A. B. Kurzhanskii and P. Varaiya. On reachability under uncertainty. *Siam Journal of Control and Optimization*, 41(1):181–216, 2002.

[62] A. B. Kurzhanskii and P. Varaiya. Optimization methods for target problems of control. In *Proceedings of Mathematical Theory of Networks and Systems Conference*, 2002.

[63] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. *Guidelines in Nonholonomic Motion Planning for Mobile Robots*, volume 299, chapter 1. Lectures Notes in Control and Information Sciences, 1998.

[64] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. Also available at http://msl.cs.uiuc.edu/planning/.

[65] E. Lee and A. Sangiovanni-Vincentelli. Comparing models of computation, http://ptolemy.eecs.berkeley.edu/ eal. 1996.

[66] E. Lewis, editor. *Principles of Naval Architecture*. Society of Naval Architects and Marine Engineers, Jersey City, New Jersey, 2nd revision edition, 1989.

[67] P. Lima and G. N. Saridis. *Design of Intelligent Control Systems Based on Hierarchical Stochastic Automata*. Intelligent Control and Intelligent Automation. World Scientific Publisher Co., 1996.

[68] J. Lygeros, Datta N. Godbole, and Shankar Sastry. A game theoretic approach to hybrid system design. Technical Report UCB/ERL M95/77, University of California, Berkeley. Electronics Research Laboratory, 1995.

[69] R. Martins, Paulo Sousa Dias, Eduardo R. B. Marques, José Pinto, J. Borges de Sousa, and F. Lobo Pereira. Imc: A communication protocol for networked vehicles and sensors. In *Proceedings of the IEEE OCEANS'2009*. IEEE, 2009.

[70] M. Mazo, A. Speranzon, K. H. Johansson, and X. Hu. Multi-robot tracking of a moving object using directional sensors. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2004.

[71] P. McGuillivary, J. Borges de Sousa, and Ricardo Martins. Connecting the dots. networking maritime fleets of autonomous systems for science and surveillance. *Marine Technology Reporter*, (October), 2012.

[72] D. Merani, A. Berni, J. Potter, and R. Martins. An underwater convergence layer for disruption tolerant networking. In *Internet Communications*. BCFIC, 2011.

[73] Robin Milner. *A calculus of communicating systems*. Springer Verlag, 1980.

[74] Robin Milner. Semantic ideas in computing. In Ian Wand and Robin Milner, editors, *Computing tomorrow : future research directions in computer science*, pages 246–283. Cambridge University Press, 1996.

[75] Robin Milner. *Communicating and mobile systems : the* Π*-calculus*. Cambridge University Press, 1999.

[76] Robin Milner. *The space and motion of communicating agents*. Cambridge University Press, 2009.

[77] G. Oriolo, A. Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: Design. *IEEE Transactions on Control Systems Technology*, 2002.

[78] José Pinto, Paulo Sousa Dias, Rui Gonçalves, Eduardo Marques, G. Gonçalves, J. Borges de Sousa, and F. Lobo Pereira. Neptus – a framework to support the mission life cycle. In IFAC, editor, *Proceedings of the 7th Conference on Manoeuvring and Control of Marine Craft (MCMC'2006)*, 2006.

[79] T. J. Prestero. Verification of a six-degree of freedom simulation model for the remus auv. Master's thesis, Massachusetts Institute of Technology / Woods Hole Oceanographic Institution, Departments of Ocean and Mechanical Engineering, 2001.

[80] A. Puri and P. Varaiya. Decidable hybrid systems. *Computer and Mathematical Modeling*, 11(23):191–202, 1996.

[81] Anuj Puri. *Theory of hybrid systems and discrete event systems*. PhD thesis, University of California at Berkeley, 1995.

[82] K. Rajan, F. Py, and J. Barreiro. *Marine Robot Autonomy*, chapter Towards Deliberative Control in Marine Robotics. Springer Verlag, 2012.

[83] G. Remmers, R. Taylor, P. Palo, and R. Brackett. Mobile offshore base: A seabasing option. In *Proceedings of Third International Workshop on Very Large Floating Structures*, pages 1–7, 1999.

[84] R. Rockafellar and R. J. B. Wets. *Variational analysis*. Springer, 1998.

[85] H. L. Royden. *Real Analysis*. Macmillan, 1988. 3rd ed.

[86] G. N. Saridis and K. P. Valavanis. Analytical design of intelligent machines. *Automatic*, 24(2):123–33, 1988.

[87] J. Sethian and A. Vladimirsky. Ordered upwind methods for hybrid control. In *Proceedings of the hybrid systems workshop*, pages 393–406. Springer-Verlag, 2002.

[88] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*. Cambridge Univ. Press, 1999.

[89] Mohammad Shahid Shaikh. *Optimal control of hybrid systems: theory and algorithms*. PhD thesis, Department of Electrical and Computer Engineering, McGill University, Montréal, 2004.

[90] J. Silva, A. Speranzon, J. B. de Sousa, and K. H. Johansson. Hierarchical search strategy for a team of autonomousvehicles. In *Proceedings of the 2004 IAV conference*. IFAC, 2004.

[91] P. Souères, A. Balluchi, and A. Bicchi. Optimal feedback control for line tracking with a bounded-curvature vehicle. *International Journal of Control*, 74(10):1009–1019, 2001.

[92] E. M. Sozer, M. Stojanovic, and J. G. Proakis. Underwater acoustic networks. *IEEE Journal of Oceanic Engineering*, 25(1):72–83, 2000.

[93] W. Spendley, G.R. Hext, and F.R. Himsworth. Sequential applications of simplex designs in optimization and evolutionary operation. *Technometrics*, 4:441–461, 1962.

[94] J. Sprinkle, J. Eklund, and S. Sastry. Deciding to land a uav safely in real time. In *Proceedings of American Control Conference*, pages 8–10. AAIAA, 2005.

[95] Roger Stokey, Ben Allen, Tom Austin, Rob Goldsborough, Ned Forrester, Mike Purcell, and Chris von Alt. Enabling technologies for remus docking: An integral component of an autonomous ocean-sampling network. *IEEE Journal of Oceanic Engineering*, 26(4):487–497, 2001.

[96] C. J. Tomlin, J. Lygeros, and S. Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–70, 2000.

[97] P. Varaiya. Theory of hierarchical, multilevel systems. *IEEE Transaction on Automatic Control*, 17(2):280–281, 1972.

[98] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(3):195–207, February 1993.

[99] P. Varaiya. Towards a layered view of control. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 1187–90. IEEE, 1997.

[100] P. Varaiya. Reach set computation using optimal control. In *Proceedings of the KIT Workshop on Verification of Hybrid Systems*. Verimag, Grenoble, France, 1998.

[101] P. Varaiya. A question about hierarchical systems. Personal communication, November 1999.

[102] P. Varaiya. Control design of an automated highway system. *Proceedings of the IEEE*, 88(7):913–25, July 2000.

[103] P. Varaiya and S. E. Shladover. Sketch of an ivhs systems architecture. In *Proceedings of the VNIS '91. Vehicle Navigation and Information Systems Conference*, pages 909–922. IEEE, 1991.

[104] P. Varaiya, T. Simsek, and J. Borges de Sousa. Communication and control of distributed hybrid systems - tutorial session. In *Proceedings of the 2001 American Control Conference*, pages 4968–83. IEEE, 2001.

[105] Pravin Varaiya. http://paleale.eecs.berkeley.edu/ varaiya/papers_ps.dir/ mica_final.pdf. 2004.

[106] M. Veloso. Autonomous robot soccer teams. *The Bridge, National Academy of Engineering*, 33(1):8–12, 2003.

[107] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 8(5):662–669, 2002.

[108] J. Scott Willcox, James G. Bellingham, Yanwu Zhang, and Arthur B. Baggeroer. Performance metrics for oceanographic surveys with autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 26(4):711–725, 2001.

[109] Huan Zhang and Matthew R. James. Optimal control of hybrid systems and a systems of quasi-variational inequalities. *SIAM Journal of Control and Optimization*, 48(2):722–761, 2006.