INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2005; **00**:1–6 *Prepared using* acsauth.cls *[Version: 2002/11/11 v1.00]*

A Verified Hierarchical Control Architecture for Coordinated Multi-Vehicle Operations[†]

João Borges de Sousa, Karl H. Johansson, Jorge Silva, Alberto Speranzon

DEEC, Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal. E-mail: jtasso@fe.up.pt
Electrical Engineering, Royal Institute of Technology, SE-100 44 Stockholm, Sweden. E-mail: {kallej,albspe}@ee.kth.se
Instituto Superior de Engenharia do Porto, R. Dr. António Bernardino Almeida 431, 4200 Porto, Portugal. E-mail: jes@isep.ipp.pt

SUMMARY

A layered control architecture for executing multi-vehicle team coordination algorithms is presented along with the specifications for team behavior. The control architecture consists of three layers: team control, vehicle supervision and maneuver control. It is shown that the controller implementation is consistent with the system specification on the desired team behavior. 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. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: Hierarchical control; Multi-robot systems; Autonomous underwater vehicle; Transition systems; Formal methods; Hybrid systems.

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.

1.1. Multi-vehicle operations

Today, there are automotive vehicles in various stages of automation ranging from automated highway systems [57, 24], 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

[†]J. Borges de Sousa and J. Silva are partially supported by Agência de Inovação through the PISCIS project. K. H. Johansson and A. Speranzon are partially supported by the European Commission through the RECSYS, HYCON and RUNES projects, by the Swedish Foundation for Strategic Research through an Individual Grant for the Advancement of Research Leaders and by the Swedish Research Council.

Received
Revised

games such as robotic soccer [58, 12] or the RobotFlag [13]. There are numerous applications for autonomous underwater vehicles, such as oceanographic surveys [51, 60, 29], operations in hazardous environments, inspection of underwater structures, mine search [23], and the Autonomous Ocean Sampling Network [10, 11], to name just a few. The Mobile Offshore Base illustrates the problem of coordinating the motions of sea-going vehicles [44, 14]. The application pull for the coordination and control of teams of unmanned air vehicles is driven mainly by military requirements [7]; some technologies have already been field tested [4, 50, 27] while others are being developed and tested in simulation [17]. A promising technological push comes from the inter-operation of multi-vehicle systems and sensor networks [9].

1.2. Approach and contributions

In this paper we present a control architecture for the implementation of a class of coordination strategies by a team of autonomous vehicles. This class is characterized by the alternation between two phases: a communication phase where the team exchanges messages to define 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 [48, 30]. 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.

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 [6] 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, 19] 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.

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 [39] and [52] 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 [55]. Dynamic programming techniques are used in [32] 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 [26, 8, 28], pursuit–evasion games [25, 59], multi-robot tracking problems [40] and multi-vehicle search missions [46, 15].

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 [21]. The fact of the matter is that the control of every large-scale system is organized in a distributed hierarchy [56]. 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 [56]: 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 [56]. 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 [45, 38], hybrid systems [53, 57, 54, 22], and linear temporal logic [18]. Our work is related to the layering concepts presented in [54]. The ideas used in execution control are inspired by [54, 53, 14, 16]. 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.

1.4. Outline

The paper is organized as follows. In Section 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 3 describes the hierarchical control structure in the framework of interacting hybrid automata. The main results are reported in Section 4 where properties of the hierarchical control structure are discussed and it is shown that such architecture implements the given system specification. In Section 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.

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 modelled 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.

2.1. Team coordination via waypoint generation

In this work 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 $e \in \Sigma$ is an event 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 $^+$ represents the update of the variable. We call $\phi(.)$ the team coordination strategy. The controller for each vehicle takes as inputs w_i^+ and t^+.

2.2. Vehicle model

Our approach encompasses general vehicle models, as we will infer from the developments in the following sections. However, and for the sake of our illustrative example with underwater vehicles, in the remainder 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

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix}, \tag{1}$$

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 modelled 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 [35, 41, 5, 36]. 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 Appendix we discuss the details of the approximation of an underwater vehicle as an unicycle model.

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.

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 [25, 59, 26, 8, 28]).

In this paper we model the system specification as a transition system.

Definition 2.1 (Transition system [43]) *A transition system T is a tuple*

$$T = (Q, \to, I, O, \mathrm{Init}, \mathrm{Final}),$$

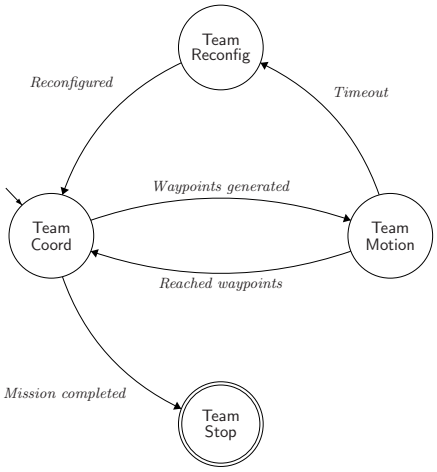


Figure 1. System specification for the team coordination.

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*
- $\mathrm{Init} \in Q$ *is the initial state*
- $\mathrm{Final} \in Q$ *is the final state*

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 $q \stackrel{i/o}{\rightarrow} q'$ instead of $(q, i, q', o) \in \rightarrow$. The graphical representation of T is a directed graph with vertices representing Q and arcs representing $\stackrel{i/o}{\rightarrow}$, an arc with empty origin representing Init and a vertex with an extra circle representing Final.

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 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

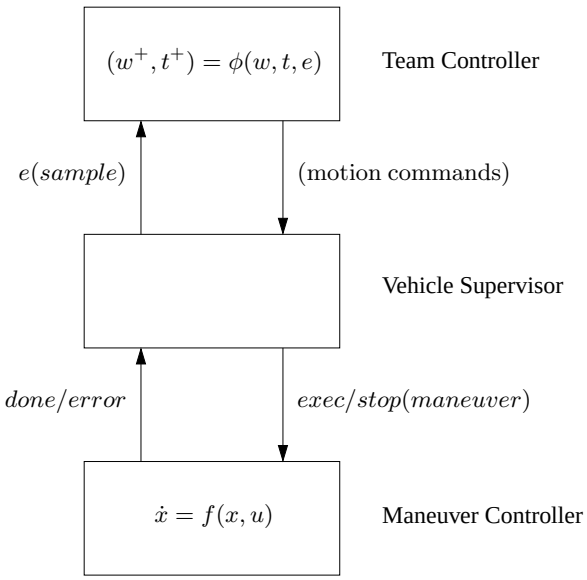


Figure 2. Hierarchical control structure for each vehicle.

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 4 we show that the design fulfills the specification.

3. HIERARCHICAL CONTROL STRUCTURE

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 2). This is an intuitive structure for program developers and system operators.

The team control architecture is depicted in Figure 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 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

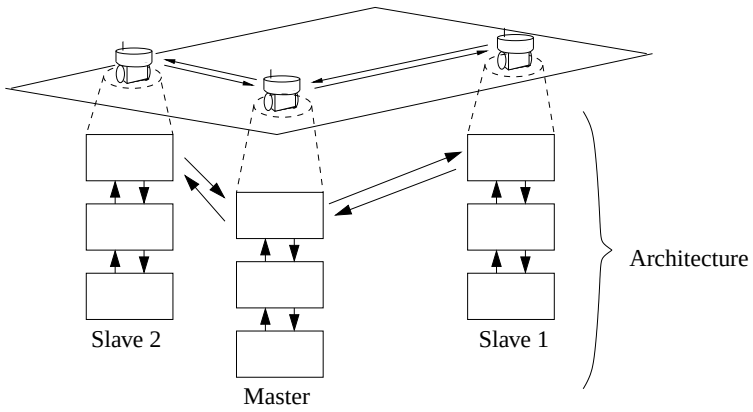


Figure 3. Control architecture for a multi-vehicle system with three vehicles resulting from the composition of the control structures for the three vehicles. Arrows between hierarchies represent communication links between vehicles. Arrows inside each hierarchical stack represent signals between different layers.

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 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 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.

3.2. Waypoint generation

As as discussed in Section 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.

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,

$$T_M = (Q_M, \rightarrow, I_M, O_M, \mathrm{Init}_M, \mathrm{Final}_M)$$

shown in detail in Figure 4, consists of the parallel composition of three transition systems. The main functionality is provided by the upper transition system of Figure 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 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 5. The team controllers of the $N-1$ slaves are identical and denoted

$$T_{S_1} = \cdots = T_{S_{N-1}} = (Q_S, \rightarrow, I_S, O_S, \mathrm{Init}_S, \mathrm{Final}_S).$$

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

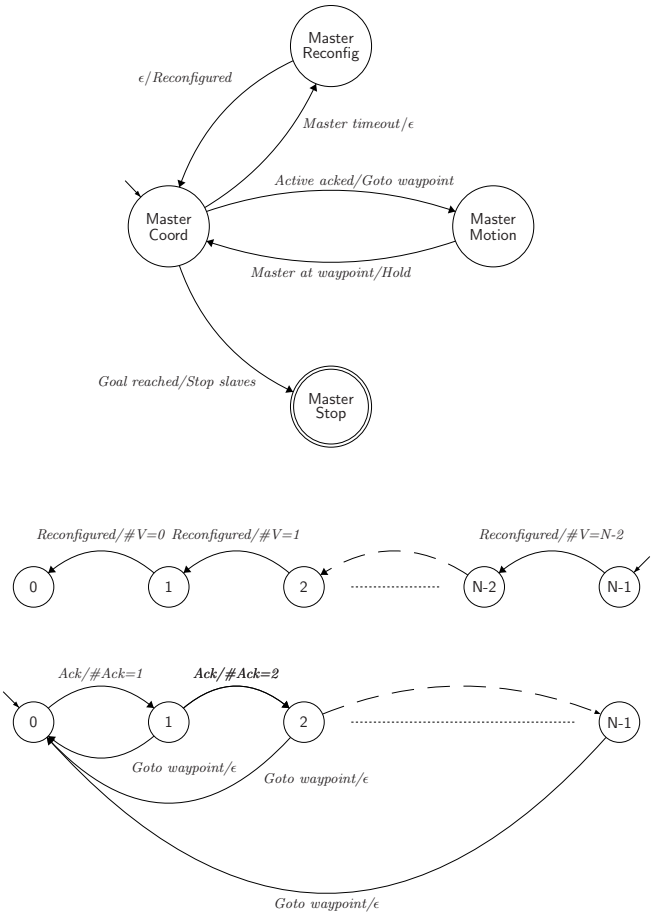


Figure 4. The master team controller is the parallel composition of three transition systems.

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 discussion of reconfiguration logic.

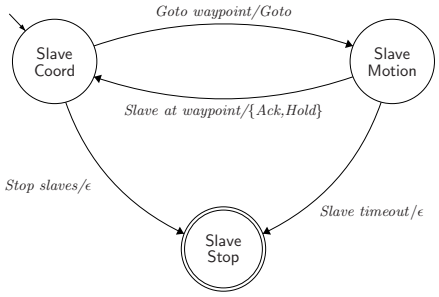


Figure 5. Slave team controller.

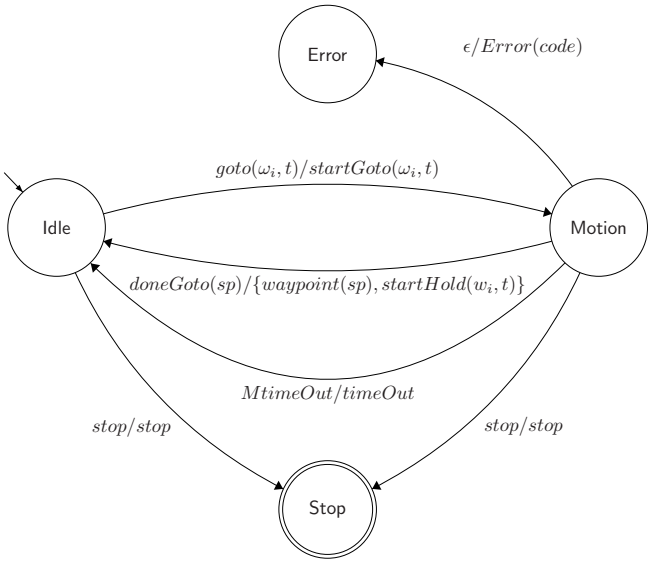


Figure 6. Vehicle supervisor.

3.4. Vehicle supervisor

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

$$T_V = (Q_V, \rightarrow, I_V, O_V, \mathrm{Init}_V, \mathrm{Final}_V)$$

is shown in Figure 6, where

- $Q_V = \{\mathsf{Idle}, \mathsf{Motion}, \mathsf{Error}, \mathsf{Stop}\}$
- $I_V = \{$*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*$\}$
- $\mathrm{Init}_V = \mathsf{Idle}$ and $\mathrm{Final}_V = \mathsf{Stop}$

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 sends the events *waypoint(sp)*, *error(code)* and *timeOut* to the team controller to indicate that the waypoint was reached, that an error of type *code* has occurred and that time out has occurred respectively. 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 modelled within the maneuver controllers.

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

$$T_C = (Q_C, \rightarrow, I_C, O_C, \mathrm{Init}_C, \mathrm{Final}_C)$$

where

- $Q_C = \{\mathsf{Init}, \mathsf{Motion}, \mathsf{Error}, \mathsf{Stop}\}$
- $I_C = \{$*start(.)*, *stop*$\}$
- $O_C = \{$*done(.)*, *error(code)*, *stop*, *timeOut*$\}$
- $\mathrm{Init}_C = \mathsf{Init}$ and $\mathrm{Final}_C = \mathsf{Stop}$

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.

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.

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 2. Since T is bisimilar to $T/\sim$ by construction, we conclude that the closed-loop system based on the composition of team controllers fulfills the specification.

Recall the definition of simulation and bisimulation for transition systems, e.g., [43].

Definition 4.1 (Simulation and bisimulation) *Given two transition systems*

$$T_1 = (Q_1, \rightarrow, I_1, O_1, \text{Init}_1, \text{Final}_1)$$

and

$$T_2 = (Q_2, \rightarrow, I_2, O_2, \text{Init}_2, \text{Final}_2),$$

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

$$T_M = (Q_M, \rightarrow, I_M, O_M, \text{Init}_M, \text{Final}_M)$$

with $N-1$ identical slave team controllers

$$T_{S_1} = \cdots = T_{S_{N-1}} = (Q_S, \rightarrow, I_S, O_S, \text{Init}_S, \text{Final}_S)$$

is illustrated in Figure 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 $T = (Q, \rightarrow, I, O, \text{Init}, \text{Final})$ 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 4), q_{S_i} is the state of slave i team controller (Figure 5), and k is the number of active slaves (middle transition system in Figure 4). (We disregard the lower transition system in Figure 4.)

We introduce the quotient transition system $T/\sim = (Q/\sim, \rightarrow, I, O, \text{Init}/\sim, \text{Final}/\sim)$ 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
- $q_M \rightarrow q_R$ provided that Master Motion $\rightarrow$ Master Reconfig, Slave Motion $\rightarrow$ Slave Coord and Slave Motion $\rightarrow$ Slave Stop
- $q_M \rightarrow q_C$ provided that Master Motion $\rightarrow$ Master Coord, Slave Motion $\rightarrow$ Slave Coord and Slave Motion $\rightarrow$ Slave Stop.

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

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

Lemma 4.2. *T and $T/\!\sim$ are bisimilar.*

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

Definition 4.3 (Isomorphic transition systems) *Two transition systems*

$$T_1 = (Q_1, \rightarrow, I_1, O_1, \mathrm{Init}_1, \mathrm{Final}_1)$$

and

$$T_2 = (Q_2, \rightarrow, I_2, O_2, \mathrm{Init}_2, \mathrm{Final}_2)$$

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

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

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

A suitable bijective map $h : Q/\!\sim \rightarrow Q_{\mathrm{Spec}}$ of Definition 4.3 is simply the relabelling:

- $h(Q_R) =$ Team Reconfig
- $h(Q_C) =$ Team Coord
- $h(Q_M) =$ Team Motion
- $h(Q_S) =$ 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 4.2) and thus also $T/\!\sim$ and T_{Spec} are bisimilar, we have the following main result.

Theorem 4.4. *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 fulfills the system specification.

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 4.5 (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:*

$$R[\tau, t_0, x_0] = \bigcup \{x[\tau] | u(s) \in U(s), s \in (t_0, \tau]\} \tag{2}$$

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

$$R[\tau, t_0, X_0] = \bigcup \{R[\tau, t_0, x_0] | x_0 \in X_0\} \tag{3}$$

Similarly, we can define reach sets for dynamic systems under disturbances and state constraints (see [32, 34, 33]). 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, $r_{\rm com}$, $v_{\rm com}$, $B_r(x)$, 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 4.7 (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*

$$\forall i, j, \|w_i - w_j\| \leq r_{\rm com} - 2m_d \tag{4}$$

$$t_3 - t_2 \geq \frac{2 \times r_{\rm com}}{v_{\rm com}}. \tag{5}$$

$$\exists t_m \in [t_c, t_1^+] : w_m^+ \in R[t_m, t_c, B_{\rm m_d}(w_m)]. \tag{6}$$

$$\forall i \in S\,, \exists t_i \in [t_1^+, t_2^+] : w_i^+ \in R[t_i, t_c, B_{\rm m_d}(w_i)]. \tag{7}$$

Condition (4) ensures that the waypoints satisfy the communication constraints (which must be valid for the next waypoints); condition (5) ensures that there is time for the communication round trip between each slave and the master; and conditions (6) and (7) ensure that the master and the slaves reach the new waypoints within the prescribed time intervals.

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 [32] 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 [39, 52].

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 [31] 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.

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 [49].

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.

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 usually applied in situations where computation capability is limited and gradient calculation is difficult, as happens in scalar fields corrupted by noise. We are interested in executing a search operation for finding the minimum of a

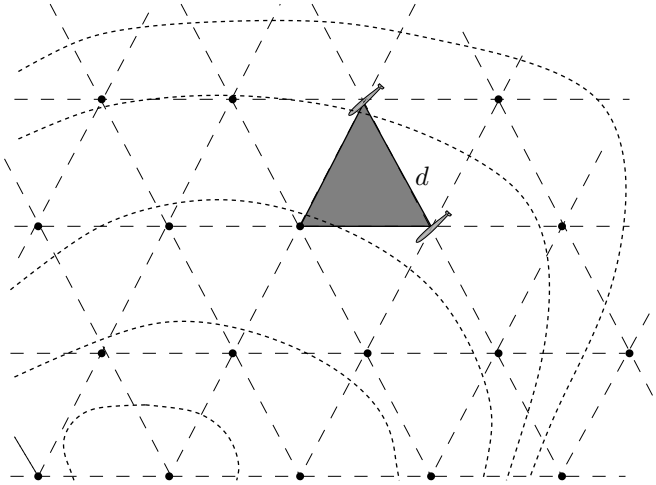


Figure 7. A triangular grid with aperture d over a scalar field depicted through its level curves (dark dashed lines). The shaded triangle illustrates the simplex location, which evolves on the grid.

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

Algorithm 1: Simplex algorithm.

planar field defined over a set $\Omega \subset \mathbb{R}^2$, see Figure 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 vertex. This procedure is described with more details below.

Consider a triangular grid $\mathcal{G} \subset \Omega$ with aperture d, as depicted in Figure 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

$$\mathcal{G} = \{p \in \Omega |\ p = p_0 + k b_1 + \ell b_2,\ k, \ell \in \mathbb{Z}\}\,.$$

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

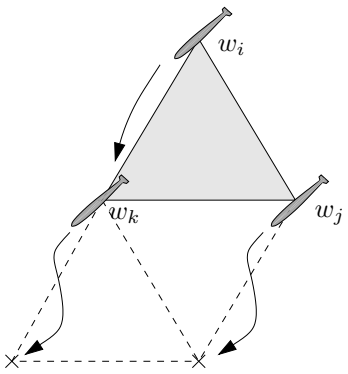


Figure 8. Assignment of the next waypoints for the three AUVs, by the master team controller, when $F(w_i) \geq F(w_j) \geq F(w_k)$.

next simplex is

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

The algorithm implementing the simplex is shown in 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 is that we are not guaranteed that when the algorithm stops we have reached a neighbor of the minimum. However the simplex can be used as a first strategy to get close to the minimum.

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 argument. 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.

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

$$\text{Team Coord} \xrightarrow{Active\ acked \big/ (w_1, w_2, w_3)^+} \text{Team Motion} \tag{8}$$

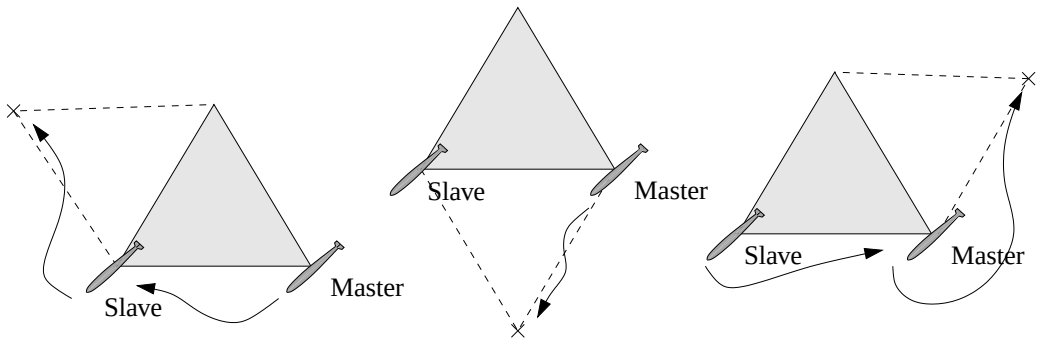


Figure 9. Assignment, by the master team controller, of the next waypoints when only one slave AUV is present.

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 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 9.

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 10. The continuous state space $X \subseteq \mathbb{R}^4$ since we have the state of the vehicle $(x, y, \psi)^T$ and the time t.

The controller 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 needs to steer the vehicle, tracking a trajectory of the type shown in Figure 11. Depending on the heading of the vehicle with respect to the final waypoint, the system will jump either to the state Turn CW or Turn CCW, turning clockwise or counter clockwise, respectively, with maximum angular velocity (see Figure 11). When the angle of the vehicle ψ is close to the angle ψ_{ref} the vehicle switches control jumping 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. If something goes wrong and the maneuver controller is not able to complete the $startGoto(w_i^+, \bar{t})$ command within time $\bar{t}$, then an error signal is communicated to the vehicle supervisor. In case of success a $doneGoto(sp)$ together with the coordinates of the reached point are signalled to the vehicle supervisor.

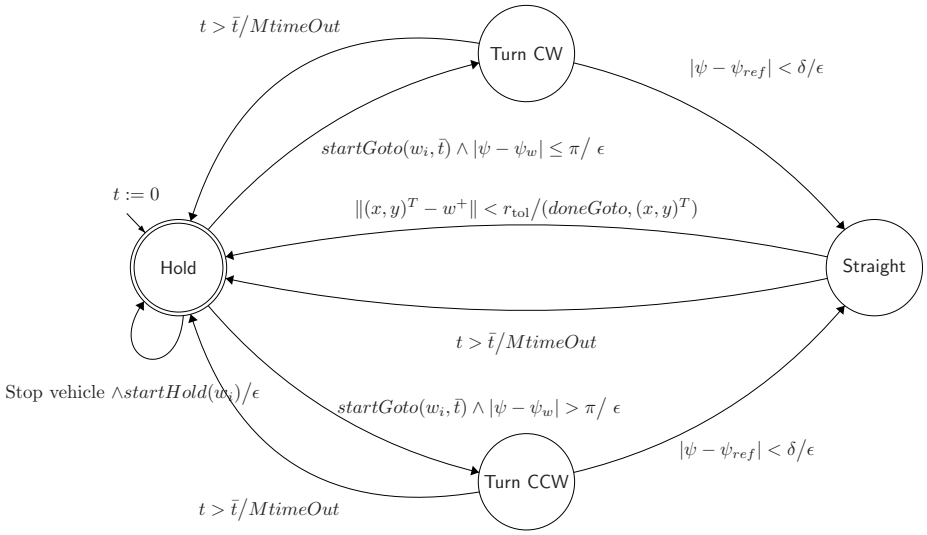


Figure 10. Hybrid automaton modelling the maneuver controller for the goto maneuver.

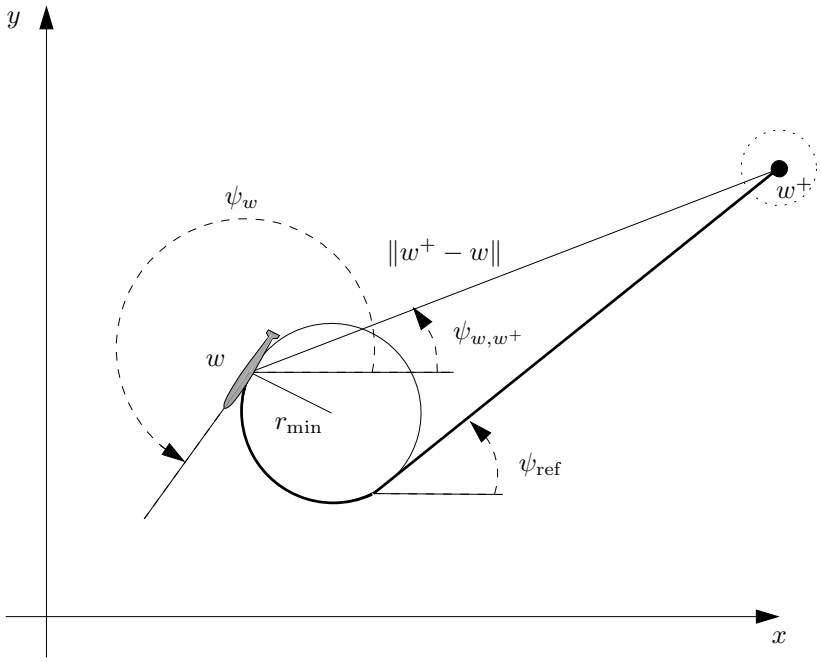
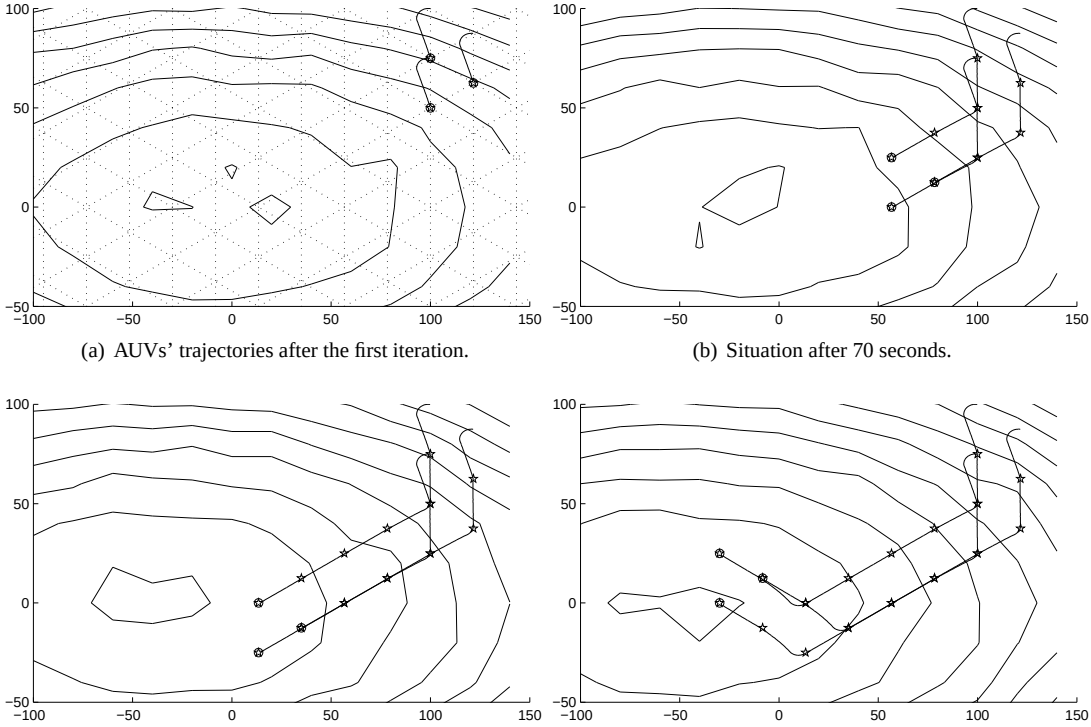


Figure 11. Example of a trajectory followed by a vehicle, for moving from w to w^+.

This is a very simple, though instructive, example of how to build a maneuver controller for this type of architecture. Complex control strategies, such as those discussed in [47], could be considered



(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 12. Simplex coordination algorithm executing a search in a noisy quadratic field with drift.

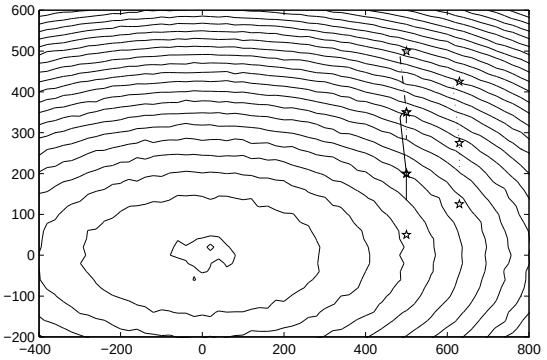
in this framework. In the case of disturbances acting on the vehicles, such as water streams, techniques as those in [2] could be used in order to counteract the actions of the disturbances.

5.4. Simulations results

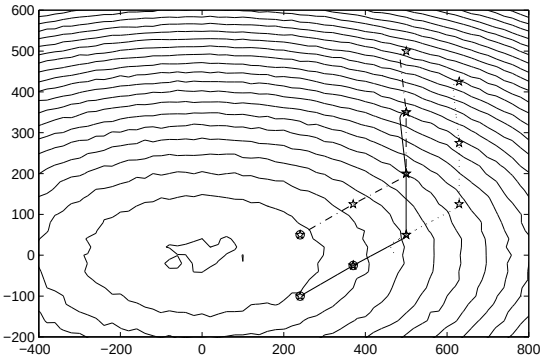
Computer simulations were performed to illustrate the behavior of the proposed hierarchical control structure applied to a team of AUVs. We considered the simplex based search with three AUVs in a time-varying planar scalar field (which could represent salinity, temperature, etc.).

Figure 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 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.

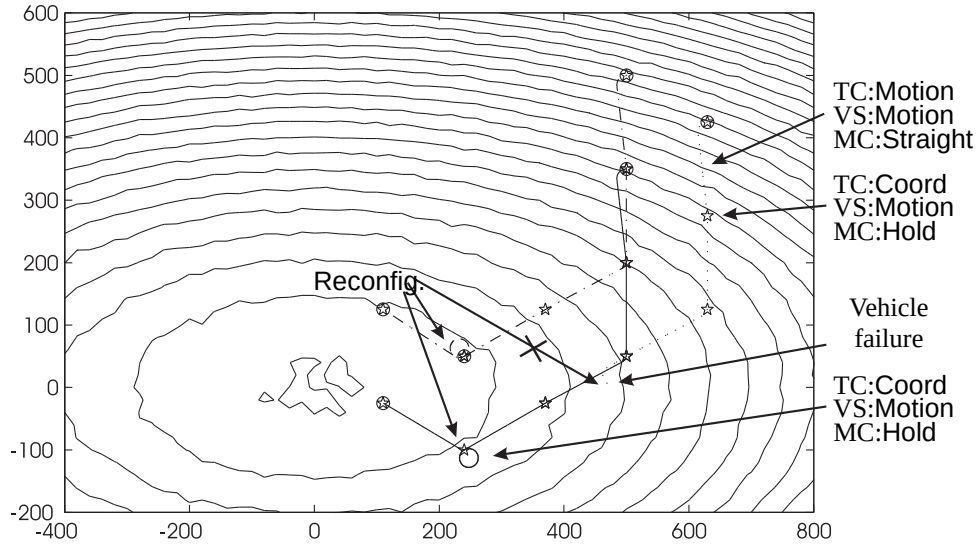
Figure 13 shows another scenario for the evolution of three AUVs towards the extremum of the scalar field. The initial simplex is $w = ((400, 300), (422, 312), (400, 275))$. The figure is labelled with the discrete states of the team controllers (TC), vehicle supervisors (VS) and maneuver controllers



(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 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.

(MC) for different phases of the operation. During the progression, one of the AUVs fails to reach its waypoint. In the figure, it corresponds to the AUV with the dotted trajectory. 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. 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, 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.

REFERENCES

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. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. D. E. Culler and H. Mulder. Smart sensors to network the world. *Scientific American*, 2004.
10. T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous ocean sampling networks. *Oceanography*, 6(3):86–94, 1993.
11. T. Curtin and J. G. Bellingham. Guest editorial: Autonomous ocean-sampling networks. *IEEE Journal of Oceanic Engineering*, 26(4):423, 2001.
12. R. D'Andrea. Robot soccer: a platform for systems engineering. *Computers in Education Journal*, 10(1):57–61, 2000.
13. R. D'Andrea and R. Murray. The roboflag competition. In *Proceedings of the American Controls Conference*, pages 650–655. IEEE, 2003.
14. 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.
15. 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.
16. 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.
17. 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.

18. 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.
19. 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.
20. T. I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley and Sons Ltd., 1994.
21. 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.
22. 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.
23. Gwyn Griffiths, editor. *Technology and applications of Autonomous Underwater Vehicles*. Ocean Science and Technology Volume 2. Taylor & Francis Group, 2003.
24. J. K. Hedrick, M. Tomizuka, and P. Varaiya. Control issues in automated highway systems. *IEEE Control Systems Magazine*, 14(6):21–32, 1994.
25. 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.
26. 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.
27. 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.
28. K. H. Johansson, A. Speranzon, and S. Zampieri. On quantization and communication topologies in multi-vehicle rendezvous. In *Proceedings of IFAC World Congress*, 2005.
29. 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.
30. 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.
31. N. N. Krasovskii and A. I. Subbotin. *Game-theoretical control problems*. Springer-Verlag, 1988.
32. A. B. Kurzhanskii and P. Varaiya. Dynamic optimization for reachability problems. *Journal of Optimization Theory & Applications*, 108(2):227–51, 2001.
33. A. B. Kurzhanskii and P. Varaiya. On reachability under uncertainty. *Siam Journal of Control and Optimization*, 41(1):181–216, 2002.
34. A. B. Kurzhanskii and P. Varaiya. Optimization methods for target problems of control. In *Proceedings of Mathematical Theory of Networks and Systems Conference*, 2002.
35. 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.
36. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. Also available at http://msl.cs.uiuc.edu/planning/.
37. E. Lewis, editor. *Principles of Naval Architecture*. Society of Naval Architects and Marine Engineers, 1989. 2nd revision.
38. 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.
39. 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.
40. 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.
41. G. Oriolo, A. Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: Design. *IEEE Transactions on Control Systems Technology*, 2002.
42. 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.
43. A. Puri and P. Varaiya. Decidable hybrid systems. *Computer and Mathematical Modeling*, 11(23):191–202, 1996.
44. 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.
45. G. N. Saridis and K. P. Valavanis. Analytical design of intelligent machines. *Automatic*, 24(2):123–33, 1988.
46. 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.
47. 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.
48. E. M. Sozer, M. Stojanovic, and J. G. Proakis. Underwater acoustic networks. *IEEE Journal of Oceanic Engineering*, 25(1):72–83, 2000.
49. W. Spendley, G.R. Hext, and F.R. Himsworth. Sequential applications of simplex designs in optimization and evolutionary operation. *Technometrics*, 4:441–461, 1962.

50. 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.
51. 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.
52. 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.
53. P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(3):195–207, February 1993.
54. P. Varaiya. Towards a layered view of control. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 1187–90. IEEE, 1997.
55. P. Varaiya. Reach set computation using optimal control. In *Proceedings of the KIT Workshop on Verification of Hybrid Systems*. Verimag, Grenoble, France, 1998.
56. P. Varaiya. A question about hierarchical systems. Personal communication, November 1999.
57. 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.
58. M. Veloso. Autonomous robot soccer teams. *The Bridge, National Academy of Engineering*, 33(1):8–12, 2003.
59. 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.
60. 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.

APPENDIX

Underwater Vehicle Model

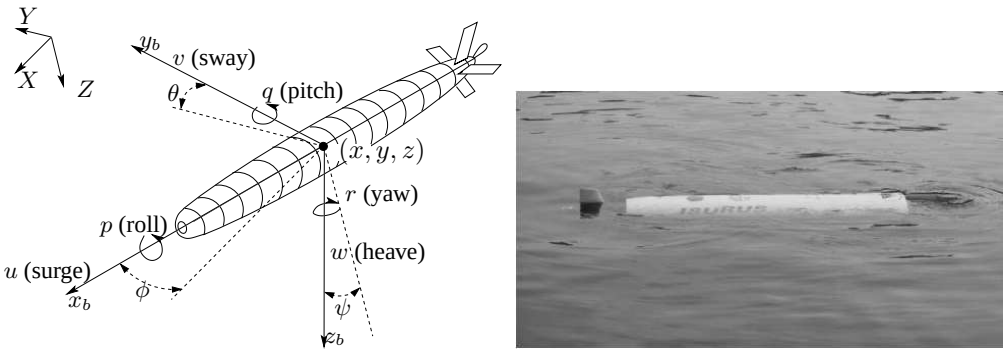
This section discusses the mapping of a nonlinear model of underwater vehicles to the kinematic model described on Section 2.2. Autonomous underwater vehicles (AUV's) are best described as nonlinear systems (see [20] for details). Two coordinate frames are considered: body-fixed and earth-fixed. In what follows, the notation from the Society of Naval Architects and Marine Engineers (SNAME) [37] is used. The motions in the body-fixed frame are described by 6 velocity components $\mathbf{v} = (v_1, v_2) = [u, v, w, p, q, r]$ respectively, surge, sway, heave, roll, pitch, and yaw, relative to a constant velocity coordinate frame moving with the ocean current. 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.

The velocities in both reference frames are related through the Euler angle transformation

$$\dot{\eta} = J(\eta_2)v \tag{9}$$

or

$$\begin{aligned}
\dot{x} &= u\cos\psi\cos\theta + v(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) + w(\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta) \\
\dot{y} &= u\sin\psi\cos\theta + v(\cos\psi\cos\phi + \sin\phi\sin\theta\sin\psi) + w(\sin\theta\sin\psi\cos\phi - \cos\psi\sin\phi) \\
\dot{z} &= -u\sin\theta + v\cos\theta\sin\psi + w\cos\theta\cos\phi \\
\dot{\theta} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\
\dot{\phi} &= q\cos\phi - r\sin\phi \\
\dot{\psi} &= q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta}\,, \qquad \theta \neq \pm 90^\circ
\end{aligned}$$



(a) (b)

Figure 14. Autonomous Underwater Vehicle.

In the body-fixed frame the nonlinear equations of motion are:

$$M\dot{\mathbf{v}} + C(\mathbf{v})\mathbf{v} + D(\mathbf{v})\mathbf{v} + g(\eta) = \tau \tag{10}$$

$$\dot{\eta} = J(\eta_2)v \tag{11}$$

M	Inertia and added mass matrix of the vehicle
$C(\mathbf{v})$	Coriolis and centripetal matrix
$D(\mathbf{v})$	Damping matrix
$g(\eta_2)$	Restoring forces and moments
τ	Body-fixed forces from the actuators

Figure (14(a)) depicts one of these vehicles. 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 some simplifications of the dynamic model. The body-fixed forces from the actuators τ depend 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 propeller 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 [42] and on our field experiments.

This work focuses on 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 small value (which in practice is not equal to zero due to the required pitch to compensate vehicle's buoyancy) and q converges to 0. The roll rate p converges to 0 due to the restoring moment of the vehicle and the roll angle ϕ converges to a value that depends on the propeller speed. In general, the pitch and roll angles can be made very small by physical configuration. Based on this assumptions and the physical shape of the vehicle, the approximated nonlinear model becomes

Table I. Steady state values of surge (u), sway (v) and yaw velocity (r) for different values of propeller actuation (% of maximum value)

Propeller actuation (%)	u (m/s)	v (m/s)	r (rad/s)
100	1.67	-0.16	0.45
75	1.25	-0.12	0.33
50	0.84	-0.08	0.22
25	0.42	-0.04	0.11

[20]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u\cos(\psi) - v\sin(\psi) \\ u\sin(\psi) + v\cos(\psi) \\ r \\ (m - X_{\dot{u}})^{-1}(X_{uu}u|u| + X_{vr}vr + X_{rr}r^2 + X_p(n)) \\ (m - Y_{\dot{v}})^{-1}(Y_{vv}v|v| + Y_{uv}uv + (Y_{ur} - m)ur + Y_{rr}r|r| + Y_{uudr}u|u|\delta_r) \\ (I_{zz} - N_{\dot{r}})^{-1}(N_{vv}v|v| + N_{uv}uv + N_{ur}ur + N_{rr}r|r| + N_{uudr}u|u|\delta_r) \end{bmatrix}. \quad (12)$$

For the purpose of motion planning, this model can be simplified. It can be seen, from physical experiments and simulations with the nonlinear model, that with constant actuation the steady state radius of curvature is constant and practically independent of the surge velocity. The curvature is determined by the angular position of the rudder fin (which is modelled 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 be as represented in Figure (15), i.e., after a very short transient period it converges to circular motion. Table I shows the steady state values of surge, sway and yaw velocity for different values of propeller actuation with maximum angular actuation. The results show, as expected, that the vehicle performs the circular motion pointing slightly inwards the circle, with an angle of $\arctan(v/u)$ in relation to the trajectory in the operation space (see the first two equations of system (12)). Notice that the value of this angle is very small for the considered vehicles (approximately 5 degrees). It can also be observed that the ratio v/u is approximately constant. By a simple trigonometric transformation the first three equations of system (12) become

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \sqrt{u^2 + v^2}\cos(\psi + \arctan(v/u)) \\ \sqrt{u^2 + v^2}\sin(\psi + \arctan(v/u)) \\ r \end{bmatrix} \quad (13)$$

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 (12), and taking in account the constant ration 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 basic motion of the vehicle will be made with relation to the moving column of water, as stated in the beginning of the section.

For these reasons, the kinematic model presented on Section 2.2 can be considered an acceptable approximation for trajectory planning. 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 the angular speed is allowed to vary instantaneously on the kinematic model while that is

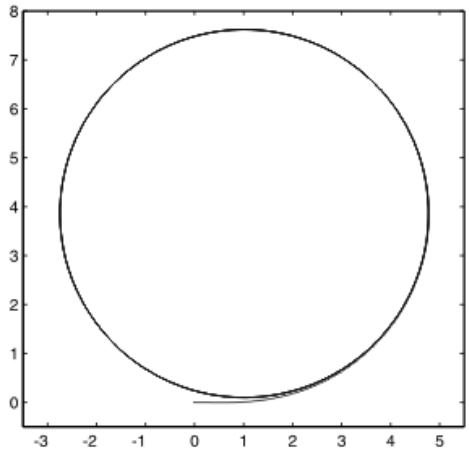


Figure 15. Trajectory of vehicle, on two dimensional operational space, starting from the origin, with null initial angular velocity and keeping constant angular actuation

not possible on the physical system (and neither on the nonlinear model). Therefore, unions between line segments and arcs of circle would not be perfectly tracked by a real vehicle. However, the main objective is to assure that the vehicles reach the destination at the desired time. If some slack is allowed when planning (for instance, considering $v'_{max} = v_{max} - \delta$), that can be achieved with a minimal deviation from the ideal trajectory.

In the paper v is used for the longitudinal velocity and ω for the angular velocity (assuming the planar motion, this replaces r in the SNAME notation).