A CONTROL ARCHITECTURE FOR MULTIPLE SUBMARINES IN COORDINATED SEARCH MISSIONS [1]

João Borges de Sousa [*] **Karl Henrik Johansson** [**]
Alberto Speranzon [**] **Jorge Silva** [***]

[*] *DEEC, Faculdade de Engenharia da Universidade do Porto*
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal
E-mail: jtasso@fe.up.pt
[**] *Dept. of Signals, Sensors & Systems*
Royal Institute of Technology, SE-100 44 Stockholm, Sweden
E-mail: {kallej,albspe}@s3.kth.se
[***] *Instituto Superior de Engenharia do Porto,*
R. Dr. António Bernardino Almeida 431, 4200 Porto, Portugal
E-mail: jes@isep.ipp.pt

Abstract: A control architecture for executing multi-vehicle search algorithms is presented. The proposed hierarchical structure consists of three control layers: maneuver controllers, vehicle supervisors and team controllers. The system model is described as a dynamic network of hybrid automata in the programming language Shift and allows reasoning about specification and dynamical properties in a formal setting. The particular search problem that is studied is that of finding the minimum of a scalar field using a team of autonomous submarines. As an illustration, a coordination scheme based on the Nelder-Mead simplex optimization algorithm is presented and illustrated through simulations.

Keywords: Hierarchical control, autonomous vehicles, hybrid systems, search methods.

1. INTRODUCTION

The problem of coordinating the operations of multiple autonomous underwater vehicles (AUV's) in the search for extremal points of oceanographic scalar fields is addressed in the paper. The coordination entails exchanging real-time information and commands among vehicles and controllers whose roles, relative positions, and dependencies change during operations. The class of search algorithms for multi-vehicle systems considered in this paper is characterized by: i) a set $I \subset \mathbb{R}^3$ of initial points; ii) a measurement function $m : \mathbb{R}^3 \to \mathbb{R}$ from locations to measurements of a given scalar field; iii) a sequence L of visited locations and measurements; iv) a way-point generation function $g : L \to \mathbb{R}^3$, which returns the set of points to visit next; and v) a termination criteria.

The approach depicted in this paper is to structure the system into a control hierarchy (Varaiya, 1997), which consists of maneuver controllers, vehicle supervisors, and team controllers. The maneuver controllers implement elemental feedback control maneuvers for the AUV's. Each AUV has attached a vehicle supervisor, which makes decisions on what maneuver to execute. The team controllers run the multi-vehicle coordination algorithm, but also handle structural adaptation and reconfiguration for the system of

[1] J. 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 and the RUNES Projects, the Swedish Strategic Research Foundation through an INGVAR grant and the Swedish Research Council.

AUV's. The control hierarchy is described as interacting hybrid automata using the Shift specification language (Deshpande *et al.*, 1997).

The paper is organized as follows. Section 2 introduces the problem formulation and the system specification. Section 3 formulates the input–output behavior of the components and how they interact as a dynamic network of hybrid automata. Section 4 describes the controllers and how they implement the search strategy. In section 5 some system properties are enunciated. Section 6 presents the implementation of an optimization-based multi-vehicle search strategy together with some simulation results. In section 7, conclusions are drawn.

2. PROBLEM

The multi-vehicle system Σ is a set of vehicles $V = \{v_1, \ldots, v_n\}$ together with their control structures. The system specification S for the class of search algorithms under consideration is given by the hybrid automaton depicted in figure 1.

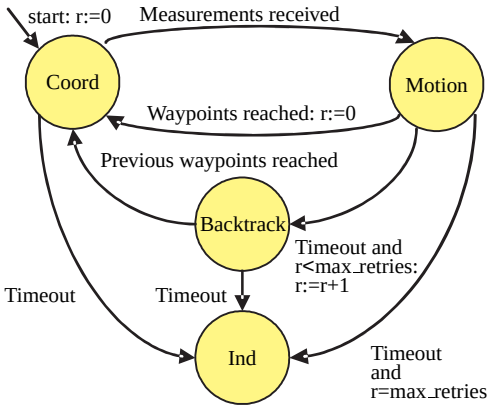


Fig. 1. System specification

The initial state is `coord`. In this state the vehicles in V exchange measurements to evaluate g and to determine waypoints. In the `motion` state the vehicles move to their designated waypoints. When they reach these points, a transition to the `coord` takes place and a new step begins. In the `motion` state it may happen that the transition to `coord` is not taken due to a communication time-out. In this case, a transition to `backtrack` is taken. In `backtrack` the vehicles move to their waypoints at the end of the previous `coord` and attempt to re-start the algorithm. If this is not possible, then a transition to `ind` is taken. At each step, the backtrack action may be attempted at most `max_retries` times. After that, a timeout in `motion` will trigger a transition to `ind`. In `ind`, each vehicle executes the search algorithm independently if coordination is no longer possible.

This paper addresses the following problem: given a multi-vehicle system Σ and a specification S prove that Σ implements the specification S.

3. COMPONENTS AND INTERACTIONS

3.1 Execution concepts

The concept of maneuver, a prototype of an action description for a single vehicle, is used as the atomic component of execution control. Thus each vehicle is abstracted as a provider of maneuvers, which allows for modular design and verification.

The design is structured in a three level control hierarchy. Proceeding bottom-up, there are the maneuver controllers (one per type of maneuver), the vehicle supervisors (one per AUV), and the team controllers (one per AUV). The next sections explain how this control hierarchy is implemented in Shift in the framework of dynamic networks of hybrid automata.

Shift users define types (classes) with continuous and discrete behavior. A simulation starts with an initial set of components that are instantiations of these types. The world-evolution is derived from the behavior of these components. The inputs and outputs of different components can be interconnected. Each discrete state has a set of differential equations and algebraic definitions (flow equations) that govern the continuous evolution of numeric variables. When transitions are taken, as determined by guards and/or by event synchronization, components can be created, interconnected, and destroyed.

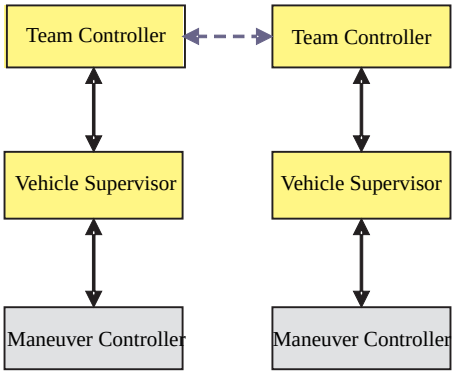


Fig. 2. Control hierarchies and links

3.2 AUV model

The Shift type definition for the AUV model is

```
type AUV  {
input      /* what we feed to it */
 array(number) tau;   // control settings

output     /* what we see on the outside */
 supervisor s;      // link to supervisor
 TeamController t;  // link to team controller
 number x,y,z;      // motion state
 array(number) eta; // body fixed velocities
 ...
flow
 x' = f1(eta,tau,x,y,z,phi,theta,psi);
 ...
}
```

The equations of motion follow the notation from the Society of Naval Architects and Marine Engineers (SNAME) (Lewis, 1989).

In Shift an instance of a type is called a component, and a component is created with a `create` statement. The first line of following example creates an instance of type AUV, with the output link variables `s` and `t` bound to components `sup1` and `tc1` respectively (link variables refer to other components), and assigns it to the link variable `v1`. The second line creates another instance `v2`. The third line creates a set V with `v1` and `v2`.

```
v1:= create(AUV, s:=sup1, t:=tc1);
v2:= create(AUV, s:=sup2,t:=tc2);
V := {v1, v2}; // set composed of v1 and v2
```

The output variable `s` of `v1` is accessed with the Shift construct `s(v1)`. Shift allows the user to assign components to link variables. This feature is used to create and maintain dynamic networks of hybrid automata. For example, the value of the link variable `t` of `v1` can be changed with the following construct `t(v1):= tc2`.

3.3 Controllers

3.3.1. Maneuver controller The object-oriented features of Shift are used to define a hierarchy of maneuver controllers. At its root there is the elemental maneuver type `MController`. The other maneuver controllers inherit from this one. The Shift data model is

```
type MController  {
 input
  number x,y,z;      // motion state
  mspec m;           // maneuver parameters
  ...

 output
  array(number) u;// control settings for actuators
  ...
}
```

The search algorithms are implemented with two maneuver types: goto(x, y, z, R, T) – reach the ball of radius R centered at (x, y, z) within time T; hold(D) – execute a holding pattern for time D.

The `goto` maneuver used in this control hierarchy is synthesized considering a differential games formulation from (Krasovskii and Subbotin, 1988) in order to ensure guaranteed performance.

3.3.2. Vehicle supervisor The Shift data model for the vehicle supervisor is

```
type supervisor  {
input      /* what is fed to it */
 TeamController tc;// link to team controller

state      /* what is internal */
 MController  mc; // link to maneuver controller
 mspec        mt;// current maneuver specification
 ...
discrete   /* discrete modes of behavior */
 Exec, Error, Idle; // 3 discrete states

transition
   Idle -> Exec {} ...
}
```

It interacts with `tc` through the exchange of the following input/output typed events:
`In_command(m)` – execute maneuver spec `m`.
`In_abort` – abort current maneuver.
`Out_donev` – completion of current maneuver.
`Out_errorv(ecode)` – error of type `ecode`.

The typed events exchanged with `mc` are:
`Out_exec(m)` – launch maneuver controller to execut maneuver specification `m`.
`In_donev` – maneuver reached completion.
`In_errorm(code)` – error of type `code`.

The transition system for this hybrid automaton is briefly described next. In the `Idle` state, the supervisor accepts a maneuver command, `In_command(m)`, from the team controller `tc`, and takes the transition to `Exec`. On this transition it creates a `MController` named `c` of type specified in m and sets the state variable `mc` to `c`. The transition from `Exec` to `Idle` is taken when an abort command is received from `tc` or when a `In_donev` event is received from `c`. On this transition the state variable `mc` is set to `nil`.

3.3.3. Team controller Each `AUV` component has a `TeamController`, which encodes the search algorithm for both independent and coordinated execution (respectively in state `Ind`) or in states `Coord, Motion, Backtrack` of the specification S. The Shift skeleton is

```
type TeamController {
 input
  set(AUV) V;            // AUVs in the team
  supervisor s;          // link to its supervisor

 state
  number step;           // last step of algorithm
  number x,y,z;          // (x, y, z) at last step
  number T1, T2, T3;     // coordination times
  number c;              // counts measurements
                         // received
  array(array(number)) L;// visited locations
  number t;              // timer

 output
  TeamController m;// link to master TeamController
  set(TeamController) tc;// links to TeamController
  symbol role;     // $master or $slave
  symbol nstate;   // name of discrete state
                   // $Init, $Error, $TMaster,
                   // $TSlave, $SingleN, $SingleI
  mspec ms;        // maneuver under execution
  set(array(number)) specs; // waypoints

 discrete    /* discrete modes of behavior */
  Init, Error, TMaster, TSlave, SingleN, SingleI;
  ...
}
```

There are six discrete states. `TMaster, TSlave` and `SingleN` concern the coordinated execution of the search algorithm. During coordinated execution one vehicle plays the role of master and the remaining vehicles play the role of slaves. In this implementation one `TeamController` is initialized in the master state and the others in the slave state, and these roles do not change. The construct `m:=self` is used in the initialization of the master `TeamController`. In

the TMaster state the TeamController receives measurements from all vehicles in V, calculates the next waypoints, and sends out the goto maneuver specifications to the vehicles in V through the link tc. In TSlave state it sends measurements to its master m and receives goto maneuver specifications from it. In SingleN it executes a default goto maneuver specification to go back to its previous waypoint and, upon its completion, it executes a hold maneuver. In SingleI it executes the algorithm by itself after all attempts to coordinate have failed.

TeamController interacts with tc through the exchange of the following input/output typed events:
Out_measurement(m) – measurement m.
Out_command(m, T1, T2, T3) – execute maneuver spec m with coordination times [T1, T2, T3]. The meaning of [T1, T2, T3] is explained in the next section.

4. EXECUTION CONTROL LOGIC

This section presents the TeamController control logic which implements a subset of the specification S, namely the one corresponding to coordinated execution. For the sake of clarity we have eliminated the transitions concerning fault-handling logic and considered that the initial allocations do not change over time. This leads to the partition of the transition system as two automata, one for the slave state and one for the master state. The automata for TMaster, TSlave are described below. In what follows, it is assumed that it is possible to keep the vehicles' clocks synchronized.

TMaster is shown in figure 3, where transitions are labelled with guards (boldface) and actions. When the system Σ enters a new step of the algorithm, the counter c is set to zero and the coordination times T1, T2, T3 set to define the time window for coordination. The master AUV is required to reach its waypoint during [T1, T2]. During [T2, T3] it receives measurements from the other vehicles and updates the counter c. When c=n it increments the step counter, calculates the new waypoints for all vehicles and coordination times for the next step T1, T2, T3, and commands all the vehicles to execute the corresponding goto maneuvers under the new coordination times. TSlave is shown in figure 4. It increments the step counter when it receives a goto command from the master m during [T2, T3]. It commands its supervisor to execute this maneuver and waits for its completion message from the supervisor. When it receives the completion message it commands the supervisor to execute a hold maneuver. Immediately after T2 it sends the measurement taken at the designated waypoint to the master and waits for the next goto command to arrive during [T2, T3]. The composition of these controllers results in an implementation which satisfies the specification. This is discussed next.

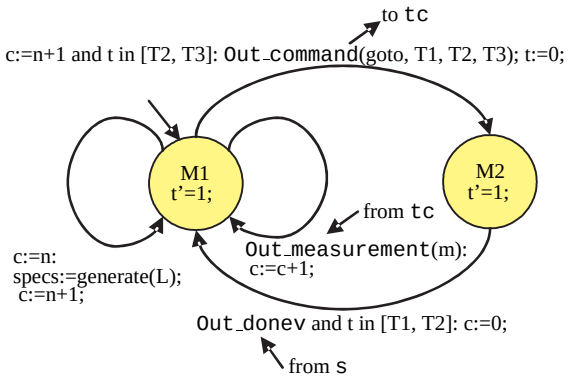


Fig. 3. Master mode operation

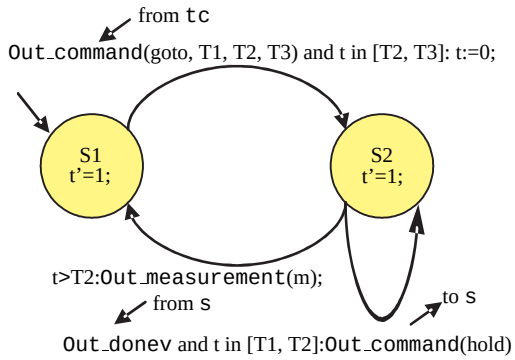


Fig. 4. Slave mode operation

Links among components of type TeamController change while the system implements the specification. The term configuration is used to denote a property satisfied by a set of interacting components. This provides for a compact notation to describe execution properties.

The system Σ evolves through four configurations to execute the specification S: ccoord; cmotion; cbacktrack; and cind.

In the ccoord configuration the system Σ satisfies the property:

$$\begin{aligned} \exists^1 \overline{v} \in V, \forall v \in V \backslash \overline{v} : m(tc(\overline{v})) = m(tc(v)) \quad (1) \\ \wedge\, m(tc(\overline{v})) = tc(\overline{v}) \\ \wedge\, tc(tc(\overline{v})) = \{tc(v_1), \ldots, tc(v_n)\} \\ \wedge\, step(tc(\overline{v})) = step(tc(v)) \\ \wedge\, nstate(tc(\overline{v})) = \$Tmaster \\ \wedge\, nstate(tc(v)) = \$TSlave \end{aligned}$$

This means that there is a master TeamController which resides in $\overline{v}$ (it is the master of itself). In this controller the value of nstate is $Tmaster; in the other controllers its value is $TSlave. The link variable m is set to the master. The master is linked to the other controllers, which are in the same step of the master: $step(tc(\overline{v})) = step(tc(v))$.

In the cmotion configuration some of the links from the ccoord configuration may have been removed as described next:

$$\exists^1 \overline{v} \in V, \forall v \in V \backslash \overline{v} : m(tc(\overline{v})) = m(tc(v)) \quad (2)$$
$$\wedge\, m(tc(\overline{v})) = tc(\overline{v})$$
$$\wedge\, step(tc(\overline{v})) = step(tc(v))$$
$$\wedge\, nstate(tc(\overline{v})) = \$Tmaster$$
$$\wedge\, nstate(tc(v)) = \$TSlave$$

In the `cbacktrack` configuration the system Σ satisfies the property:

$$\exists^1 \overline{v} \in V, \forall v \in V \backslash \overline{v} : m(tc(\overline{v})) = tc(\overline{v}) \quad (3)$$
$$\wedge\, step(tc(\overline{v})) = step(tc(v))$$
$$\wedge\, nstate(tc(\overline{v})) = \$SingleN$$
$$\wedge\, nstate(tc(v)) = \$SingleN$$

Configuration is a global concept. The controllers do not manipulate configurations directly. However, the controllers ensure that the system transitions between the `ccoord` and the `cmotion` configurations in the absence of faults.

5. SYSTEM PROPERTIES

This section discusses how the system Σ implements the specification S. For this to happen, Σ has to satisfy the following properties (space limitations preclude discussion of properties P2–P4).

P1 (Continuation): normal execution does not block, i.e., the target sets generated at each step are reachable and the vehicles are able to exchange coordination information at the end of the step to proceed to the next step. The target sets are specified in terms of way-points, radius, and time window.
P2 (Termination): execution terminates in a finite number of steps if the original search algorithm terminates in a finite number of steps.
P3 (Reconfiguration): normal execution continues if all vehicles in V are able to backtrack to the previous step and the system is able to resume execution.
P4 (Fault-handling): execution continues if there exists at least one vehicle in V after a failed attempt to reconfigure the system.

Consider the controlled motions of an AUV. The backward reach set $W[\gamma, t_\alpha, t_\beta, \mathcal{M}]$ at time $\gamma \leq t_\alpha$ is the set of points $X = (x, y, z)$ such that there exists a control $\tau(t)$ that drives the trajectory of the system $X[t] = X(t, \gamma, X)$ from state X to the target set $\mathcal{M}$ at some time $\theta \in [t_\alpha, t_\beta]$. Let $M_{i,j}$, $X_{i,j}$, $\gamma_{i,j}$, and $[T1_j, T2_j]$, denote respectively the target set, the initial position, the initial time, and the time window at step j for vehicle v_i. $M_{i,j}$ is a function of the output variable $specs$ generated by the master `TeamController`.

Theorem 1. Property P1 holds for a system implementation in which the following conditions are true:
i) $configuration(\gamma_{i,j}) =$ `ccoord` (the function $configuration(s)$ returns the configuration of the system at time s).
ii) $\forall i \in \{1, \ldots, n\} : X_{i,j} \in W[\gamma_{i,j}, T1_j, T2_j, M_{i,j}]$.
iii) configuration$(t) =$ `ccoord`, $t \in [T2, T_f]$ for some $T2 \leq T_f \leq T3$.
iv) the master-slave coordination does not block.

Condition i) means that the configuration of the system is such that communication was possible and that `TMaster` and `TSlave` are in the same `step`. Conditions ii) and iii) mean that the target sets are reachable and that the communication constraints are satisfied.

Consider the following assumptions on the way-point generation function g: (a) it generates reachable target sets; and (b) for all points in the target sets the communication constraints are valid.

Theorem 2. Conditions (i)-(iv) in theorem 1 are satisfied by the controllers described in section 4 and by the way-point generation function g.

Conditions i) and ii) result from the application of g. Condition iii) results the properties of the `goto` controller. Condition iv) follows from the structure of `TMaster` and `TSlave`.

The transitions in the specification automaton correspond to the transitions in `TMaster`. The last theorem states that the control hierarchy implements the specification.

6. SIMPLEX ALGORITHM IMPLEMENTATION

This section describes how the team controller can execute the Nelder-Mead simplex optimization algorithm, which is a direct search method used in many practical optimization problems. The method is suitable for coordinating a team of AUV's to localize a minimum of a scalar field in the plane. For particular fields, such as the quadratic field, it is possible to derive bounds on the distance to the minimizer when the simplex algorithm terminate (Silva *et al.*, 2004). The points generated by the simplex algorithm correspond to the target regions of the team controller. Following the description of the simplex implementation, numerical simulations illustrating the approach in a realistic setting are presented.

6.1 Simplex implementation

This section introduces the simplex optimization algorithm (Nelder and Mead, 1965). Consider a compact convex set $\Omega \subset \mathbb{R}^2$ containing the origin. Define a field through a scalar-valued measurement map $m : \Omega \rightarrow \mathbb{R}$ and a triangular grid $\mathcal{G} \in \Omega$ as depicted in Figure 5, with aperture $d > 0$. Introduce an arbitrary point $p_0 \in \Omega^\circ$ and a base of vectors given by b_1, b_2

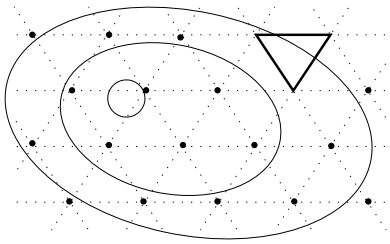


Fig. 5. A triangular grid with aperture d over a scalar field m depicted through its level curves. The solid line triangle illustrates the simplex location, which evolves on the grid.

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 given by

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

A simplex $z = (z_1, z_2, z_3) \in \mathcal{G}^3$ is defined by three neighboring vertices in $\mathcal{G}$. It is assumed, without loss of generality, that $V(z_3) \geq V(z_i)$, $i = 1, 2$. Given a simplex $z = (z_1, z_2, z_3)$ the next simplex, z', is generated from z by reflecting z_3 with respect to the other vertices, i.e., it is given by the mapping

$$z \mapsto z' = f(z) = (z_1, z_3, z_1 + z_2 - z_3). \quad (4)$$

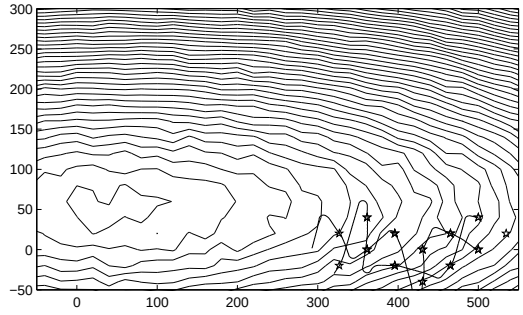
The map f defines the way-point generation function $g : L \to \mathbb{R}^3$ of the team controller, as described in the sequel. Consider a case with two AUV's: v_1 and v_2. (It is easy to incorporate more vehicles.) Suppose the team controller of v_1 will control both v_1 and v_2. According to the definition of `TeamController`, the following assignments are made:

```
role(tc1(v1)):=$master;
role(tc2(v2)):=$slave;
```

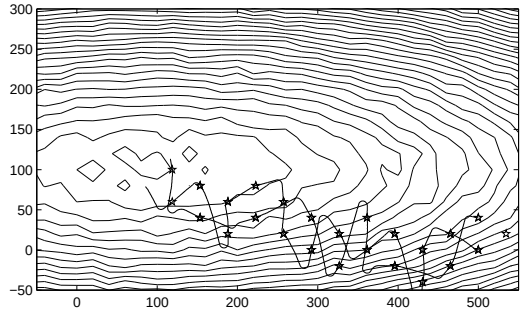
Note that `L(step)` denotes the visited location at the last step of the algorithm. If that location is denoted by $z = (z_1, z_2, z_3)$, as above, it simply follows that the next location set should be given by $z' = (z_1, z_3, z_1 + z_2 - z_3)$. This relation defines g.

6.2 Simulations

A simulation study was done to illustrate the behavior of the proposed hierarchical control structure. In particular, the simulation addresses the simplex search with two AUV's in a time-varying scalar field, which could represent salinity, temperature, etc. in a region of interest. Figure 6 shows four snapshots of the evolution of the AUV's. The field is quadratic with additive white noise and a constant drift. As illustrated in the figure, the vehicles are able to find the minimizer of the field.



(a) Situation after 12 time steps.



(b) Situation after 24 time steps.

Fig. 6. Simplex coordination algorithm executing a search in a noisy quadratic field with drift.

7. CONCLUSIONS

This paper shows how to map a conceptual control architecture onto a control design while preserving the architectural properties dictated by its specification. This is done for a multi-vehicle search problem. The architecture is modelled as a dynamic network of hybrid automata structured in a hierarchical fashion, specified using the Shift language. Some properties were inferred for the architecture and were posteriorly observed in the simulation.

REFERENCES

Deshpande, A., A. Gollu and L. Semenzato (1997). The shift programming language and run-time system for dynamic networks of hybrid automata. Technical Report UCB-ITS-PRR-97-7. California PATH.

Krasovskii, N.N. and A.I. Subbotin (1988). *Game-theoretical control problems*. Springer-Verlag.

Lewis, E., Ed.) (1989). *Principles of Naval Architecture*. Society of Naval Architects and Marine Engineers. 2nd revision.

Nelder, J.A. and R. Mead (1965). A simplex method for function minimization. *Computing Journal* **7**, 308–313.

Silva, Jorge, Alberto Speranzon, J. Borges de Sousa and Karl Henrik Johansson (2004). Hierarchical search strategy for a team of autonomous vehicles. In: *Proceedings of the 2004 IAV conference*. IFAC.

Varaiya, P. (1997). Towards a layered view of control. In: *IEEE Conference on Decision and Control*. Vol. 2. pp. 1187–1190.