ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2006, Vol. 45, No. 5, pp. 824–830. © Pleiades Publishing, Inc., 2006.

CONTROL SYSTEMS FOR MOVING OBJECTS

A Set-Valued Framework for Coordinated Motion Control of Networked Vehicles[1]

J. B. Sousa and F. L. Pereira

Institute for Systems and Robotics-Porto, Porto, Portugal

Received April 1, 2004; in final form, April 21, 2005

Abstract—A problem of coordinated control for two networked vehicles is presented, formulated, and solved to illustrate a set-valued framework for the coordinated control of networked vehicles. First, an informal user specification for the coordinated operation of two vehicles is introduced. Second, a formal representation of the objects and relations of interest is presented, and the user requirements are mapped onto a formal specification. Third, the specification is transformed onto a control problem formulation. Finally, the problem is solved using techniques from dynamic optimization and generalizations of the problem are discussed.

DOI: 10.1134/S1064230706050133

INTRODUCTION

Technological developments and application requirements are attracting the attention of researchers in control engineering, computer science, and communications to the problem of coordinated control of networked vehicles and systems. From control engineering, this problem motivated research on the stability of interactions, modes of interaction, etc. From computer science, the topics of interest are formal specification, verification, mobility, and reasoning. From communications, the topics of interest are ad-hoc networking, route finding, etc.

The interdisciplinary nature of coordination and control for networked vehicle systems requires a new description language. In fact, control engineers have developed a collection of idioms, patterns, and styles of organization that serves as a shared, semantically rich, vocabulary among them. However, this shared vocabulary is still deeply rooted in the underlying mathematical framework—differential equations and dynamic optimization—and lacks some semantically rich concepts, such as those arising in distributed computing, or in computer science. The cause may be that experience and functionality in computing are acquired at a rate unmatched by the rate of evolution of concepts in control systems. For example, Systems Engineering manuals define two types of requirements: user requirements and technical requirements. User requirements express the user expectations about the system behavior. Technical requirements describe the user requirements in the technical language describing the operation of the system, i.e., in terms of components, sub-components, functionality, performance, etc. The problem in networked vehicle systems is that this technical description language is not available today. This makes the requirements-design–implementation cycle particularly difficult for these systems.

The paper illustrates, with a simple problem of coordinated control for two vehicles, a framework for the representation, formal specification, and control synthesis for networked vehicle systems [1, 2]. The framework relies on techniques from set theory and dynamic optimization.

First, we represent all of the entities, their dynamic behavior, and the relations among themselves. Relations of interest for us are (1) services and their composition; (2) services and their physical implementation; (3) services and their order relations; (4) entities and modes of coordination; (5) entities and properties of their composition; (6) services and service providers; (7) entities and their control structure; (8) control structures and services. Some of the relations are static; others concern the dynamic behavior of vehicles under coordination constraints that change with time. In order to represent the last ones, we use a set-valued description of the dynamic behavior of vehicles and teams. We use reach sets to describe the evolution of a dynamic system, invariant sets to describe the locations where the permanence of an entity within a certain set is ensured, and solvability sets to describe the locations from which a system can evolve to reach a given set. Second, we specify operations on these entities and express the specification in a formal language. The key observation is that we can represent the entities, their relations, and their operations in the language of sets. This way we are able to represent the world of systems of networked entities with simple concepts from set theory. This is why we will be able to formally relate design and specification. Fourth, we define a planning procedure that results in a data structure defining all of

[1] The text was submitted by the authors in English.

the controller specifications that precede controller design, and where all logical relations are already satisfied. Fifth, we use techniques from dynamic optimization to synthesize controllers that implement the plan, or that prove that the plan is not feasible. The consistency of the whole process results from considering a uniform representation formalism.

The paper is organized as follows. In Section 2, we informally introduce the requirements for the coordinated operation of two networked vehicles arising in applications involving air, ground, and underwater vehicles. In Section 3, we present a formal representation of this problem domain and transform the informal requirements onto a formal specification. In Section 4, we formulate the control problem, and, in Section 5, we propose a solution methodology. In Section 6, we discuss an example. In Section 7, we draw some brief conclusions, discuss coordination and control problems in this framework, and relate them to recent developments in the control of distributed systems, communication, and computation.

1. PROBLEM DESCRIPTION

We have used examples of the coordinated operation of autonomous vehicles for underwater—autonomous underwater vehicles (AUV)—and air—unmanned air vehicles (UAV)—applications as the motivation for our example. In the following section, we will see that the technical requirements for both application domains are essentially the same in spite of different user specifications. This is the kind of abstraction we are looking for.

AUVs are an enabling technology for oceanographic field studies [3, 4]. Although the technology is still maturing, we can anticipate interesting challenges to the networked operation of multiple AUVs in oceanographic field studies. These challenges come from the nature of these field studies, and from the major limitations of AUV technology. Oceanographic field studies are evolving to the construction of detailed models. These, in turn, have to be validated with sophisticated sampling strategies that provide the required spatial and temporal resolution. The major limitations of AUV technology are space, power, and communications, to name just a few. For example, underwater communications are based on acoustic technology that is severely constrained in terms of range and rate.

UAVs are in high demand for military, scientific, and civilian applications [5]. Military operations present the most challenging scenarios [6]. For military operations, UAVs are tasked to "dirty," "dull," and "dangerous" operations. "Dirty" refers to reconnoitering areas that may be contaminated, "dull" applies to surveillance or sentry duty, while "dangerous" is related to obvious threats, such as those posed by the suppression of enemy air defenses (SEAD). In reconnaissance missions, we have teams implementing search-based algorithms, with the integration of data from different sensors mounted on different vehicles. The motivation for our example comes from reconnaissance missions. Consider a ground vehicle traveling a known road to chart the terrain around it, and an UAV providing air surveillance, i.e., informing the ground vehicle of the presence of hostile assets. This is basically the setting of our example.

Consider the following informal user specification. We keep the description at a high level of abstraction to show that it may describe problems arising in applications for sea, land, and air vehicles.

Consider two vehicles, A and B, that coordinate their motions to execute a "mapping" task. The "mapping" task consists of having vehicle A following a prescribed path in the geographic (x, y) plane and taking measurements along that path without colliding with obstacles. There are no constraints on the z geographic coordinate except for those arising from unknown obstacles. Vehicle A has a mapping sensor and does not have any sensor for obstacle avoidance. Vehicle B surveys the area in front of vehicle B to identify the presence of potential obstacles. B is faster than vehicle A, and communicates the presence of obstacles to A. To do this, B carries an obstacle detection sensor.

The problem is to coordinate the motions of the two vehicles so that, under mild assumptions on the topography of the world, the vehicles are able to execute the mapping task successfully; i.e., vehicle A does not collide with an obstacle before reaching its destination.

Hereafter, and unless stated otherwise, we refer to this abstract user specification as "our example."

2. FORMAL REPRESENTATION AND SPECIFICATION

The world basically consists of **regions, physical objects, teams** of physical objects, and **networks** of teams. While some objects have a physical existence, others are brought into existence as software agents.

Regions are subsets $\mathcal{R}^n$. Physical objects are **vehicles** and **devices**. Each physical object is located within at least one region.

A vehicle/device has **attributes** (e.g., range), it is capable of delivering **atomic services** (e.g., sensing), of executing **tasks** (e.g., fly a certain path), and of performing **actions** (e.g., launch a missile). A vehicle is controlled to move, and to deliver atomic services while moving. Physical objects have the potential to establish interactions among themselves. This is done with **atomic links**. An atomic link is a relation on the positions, motions, and atomic services provided by two physical objects. An **atomic configuration** is a list of atomic links connecting a group of vehicles.

We use physical objects as the building blocks of **teams** of physical entities and of networks of teams. Teams and networks of teams are brought into existence to deliver complex services, and to perform tasks that cannot be delivered by a single physical object.

A **complex service** is a service that results from the composition of atomic and complex services and that cannot be delivered by a single physical object; i.e., it has to be delivered by the coordinated activities of a team of vehicles. In order to do this, the vehicles in the team have to be in a particular atomic configuration. In practice, complex services emerge from **modes of cooperation** among multiple entities, some of which are physical objects and others of which are software agents.

A **team** is a set of vehicles that is able to perform **team missions** A team mission consists of team tasks and of task switching logic (also called a team **play**). A team task consists of the delivery of task services and team motions. A task service consists of complex services and modes for the delivery of these services.

A **plan** is a data structure consisting of team tasks, controller specifications for each task, ordering constraints, variable binding constraints, and causal links. The plan is refined into team tasks where the issues of team composition and tasking, resource allocation, path planning, etc., are addressed.

We use the user requirements from the previous section to illustrate the main objects and relations among those objects in the formal representation of this problem domain.

In our example the set of `Vehicles` is

`Vehicles` = $\mathcal{V} = \{A, B, C, D\}$.

There are two types of vehicles `Mapper` and `Scout`:

`Type (A)` = Mapper, `Type (B)` = Scout, `Type (C)` = Scout, `Type (D)` = Mapper.

The function `ProvideAtomicService` returns the list of atomic services provided by each vehicle type. The function `AttributeAtomicService` returns the list of attributes of an atomic service and the function `ValueAttribute`(a, c) returns the value of attribute a of the type c atomic service.

`ProvideAtomicService (Mapper)` = {*Coms, MapSensor, Motion*},

`ProvideAtomicService (Scout)` = {*Coms, Obstacle Detections Sensor, Motion*},

`AttributeAtomicService (Coms = Range,`

`ValueAttribute(Range, Coms)` = R_{Coms}.

(Range, Coms) = R_{Coms}.

The equations of motion for all vehicles are given by

$$\dot{x}_v(t) = f_v(t, x_v(t), u_v(t)),\ u_v(t) \in \mathcal{U}_v,\ v \in \mathcal{V}.$$

The function `Position`(t, v) returns the geographic position (x, y, z) of vehicle v at time t:

$$\texttt{Position}(t, v) = \Pi(x_v(t)),$$

where Π gives the projection of the state of vehicle v onto the geographical position of the vehicle.

Atomic services are the building blocks of complex services. This is because some of the atomic services have the potential to establish interactions among the respective service providers. We call the atoms of these interactions **atomic links**: an atomic link is a relation on the relative motions, positions, and atomic services provided by two different service providers. The predicate *AtomicLink* (t, v_1, v_2) represents the fact that vehicles v_1 and v_2 are linked with a link of type t. The type defines the **role**—the atomic services and the list of commands accepted and issued—of each of the participants in the link and the **glue**—the way the two participants interact. The glue is a relation on the relative positions and motions of both service providers and on the commands they exchange. The glue is determined from the attributes of the corresponding atomic services.[2]

We represent the fact that any two vehicles in Vehicles are able to communicate under some well-defined conditions with the atomic link of type *Coms*:

$$AtomicLinks(Coms,\ v_1, v_2)$$
$$\Leftrightarrow \begin{cases} Coms \in ProvideAtomicService(v_1) \wedge \\ Coms \in ProvideAtomicService(v_2) \wedge \\ \overline{\phi_{Coms}} Position(t, v_1), Position(t, v_2) \le 1, \end{cases}$$

where $\overline{\phi_{Coms}}(a, b) : \Re^3 \times \Re^3 \longrightarrow \Re$, such that

$$\overline{\phi_{Coms}}(a, b) = \frac{d^2(a, b)}{R^2_{Coms}}, \text{ being } d(a, b) = \|a - b\|_2.$$

It is convenient to express the function $\overline{\phi_{Coms}}$ and terms of the full state of both vehicles v_1 and v_2

$$\phi_{Coms}(x_{v_1}, x_{v_2}) = \overline{\phi_{Coms}}(\Pi(x_{v_1}), (x_{v_2})).$$

We model the interactions described in the previous section between two generic `Mapper` and `Scout` vehicles, v_1, v_2 respectively, as the `ScoutedMapping` complex service. This service is informally described as follows. A vehicle of type `Scout`, v_2, evolves in a vicinity P(*Position* (t, v_1)) of the current geographic position of v_1 and informs v_1 of the existence of obstacles so that v_1 can perform obstacle avoidance successfully. We assume that P[3] is given as a set-valued map from the current geographic position of v_1 to a subset of $\Re^3$.

$$P(a) : a \in \Re^3 \longrightarrow P(a) \subset \Re^3.$$

[2] Note that the glue may involve more than one atomic service. This is the case when commands are exchanged.

[3] The derivation of the expression for P is based on the dynamics of vehicle v_1 and also on assumptions on the world.

We represent this type of interactions between two vehicles with the atomic link of type *Inside*.

$$AtomicLink(Inside,\ v_1, v_2) \Leftrightarrow \begin{cases} Type(v_1) = Mapper \wedge \\ Type(v_2) = Scout \wedge \\ \overline{\phi_{Inside}}(Position(t, v_1), Position(t, v_2)) \le 1, \end{cases}$$

where $\overline{\phi_{Inside}}: \Re^3 \times \Re^3 \longrightarrow \Re$ such that

$$\overline{\phi_{Inside}}(a,b) = d_c^2(b, P(a)) + 1, d_c(b,P) = \min_{s \in P} d(s,b).$$

As before, we define $\phi_{Inside}(x_{v_1}, x_{v_2})$ as follows:

$$\phi_{Inside}(x_{v_1}, x_{v_2}) = \overline{\phi_{Inside}}(\Pi(x_{v_1}), \Pi(x_{v_2})).$$

The implementation of the service `ScoutedMapping` also requires both vehicles to communicate. This means that they have to satisfy a configuration, i.e., a list of atomic links. We use an **atomic configuration style** as a compact representation of a set of atomic configurations sharing a common property. We represent the configuration style t with a predicate *ConfigurationStyle* (t, c), where c is a team of vehicles.

$$ConfigurationStyle(ScoutMapper, V) \rightleftharpoons \exists X, Y \in \nu : \begin{cases} Type(X) = Mapper \wedge \\ Type(Y) = Scout \wedge \\ AtomicLink(Coms, X, Y) \wedge \\ AtomicLink(Inside, X, Y). \end{cases}$$

We use the following predicates and functions to represent a type s complex service.

RequiredVehicleType(s) returns a list with the types of vehicles required to implement the service.

RequiredVehicles(s, c, $\mathcal{V}$) returns all the subsets of $\mathcal{V}$ capable of delivering the complex service of type s with the value of attributes specified in c.

RequiredConfigurationStyle(c, a) returns the configuration style that each set of vehicles in *RequiredVehicles*(c, a) must satisfy delivery of the service c with the value attributes as specified a.

We represent the `ScoutedMapping` complex service as follows.

`RequiredVehicleType(ScoutedMapping) = {Scout, Mapper},`

`RequiredVehicles (ScoutedMapping, nil, Vehicles) =` $\{\{A, B\}, \{A, C\}, \{D, B\}, \{D, C\}\}$,

`RequiredConfigurationStyle (ScoutedMapping, Vehicles) = ScoutMapper.`

Single vehicles and teams of vehicles execute tasks. A task has a type and is executed by a team of vehicles.

In our example, we are interested in the `Mapping` task. The task is defined as follows.

`Task(Mapping,` $\{X, Y\}$, `ScoutedMapping` (X, Y), `Path(`x_0, x_f, p, X`),` ϕ_0 `(Position(`t_0, X`), Position(`t_0, `Y)))`

where `Mapping` is the type of the task, $\{X, Y\}$ is the team of vehicles executing the task while delivering the service `ScoutedMapping`, $p = \{(x, y) \in \Re^2 : (x, y) = p(t), t \in [t_0, t_f]\}$, and `Path(`$x_0$, x_f, p, X`)` and ϕ_0`(Position(`t_0, X`), Position(`t_0, Y`))` are defined as follows (X, Y are vehicle variables)

$\forall t \in [t_0, t_F] : \overline{\phi_{path}}$`(`$t$, `Position(`$t, X$`),` $p(t)) \le 1$,

where

$\overline{\phi_{path}}(t, a, b) : \Re \times \Re^3 \times \Re^3 \longrightarrow \Re$ s.t.,

$\overline{\phi_{path}}(t, a, b) = d^2(a, b) - \delta + 1$,

ϕ_0`(Position(`t_0, X`), Position(`t_0, Y`))` ≤ 1,

where δ is the path-tracking tolerance and the last equation defines the set of initial positions for vehicles X and Y, $\phi_0(a, b) = d^2(a, b) + 1$. We define ϕ_{path} in the manner described above:

$$\phi_{path}(t, x_{v_i}, b) = \overline{\phi_{path}}(t, \Pi x_{v_i}, b).$$

The plan specification is a data structure consisting of tasks and a partial order on these tasks. The user requirements for our example are mapped onto the following plan specification (which consists of the mapping task):

`Plan = {Task(Mapping,` $\{X, Y\}$, `ScoutedMapping(`X, Y`), Path((0, 10), (100, 10),` p, X`),` ϕ_0`(Position(`t_0, X`), Position(`t_0, Y`)))}.`

We need to transform this plan specification onto an implementable plan; i.e., we need a planner. In this case, the role of the planner consists of selecting two vehicles from `Vehicles` such that the initial conditions in the `Mapping` task are satisfied at time t_0. This may require the introduction of additional tasks. In this paper, we are not concerned with planning procedures and we assume that the planner produced the following plan.

`Plan = {Task(Mapping,` $\{A, B\}$, `ScoutedMapping(`A, B`), Path((0, 10),(100, 10),` p, A`)),` ϕ_0`(Position(`t_0, A`), Position(`t_0, B`))}.`

At this point, the plan consists of control specifications from which we derive a feasible structure of con-

trollers in case it exists. Otherwise the plan is not implementable with the available assets.

Before we proceed to discuss the control formulation, some remarks are in order. The previous specification does not completely specify the role of vehicle B; i.e., we have a partial plan specification. This means that the plan may be implementable with different control structures. Although we haven't illustrated that feature, our approach also accommodates task performance specifications and design preferences. Plan parametrization allows one to adjust performance to the availability of assets to implement the plan.

3. FORMULATION

The control problem formulation arises naturally from the previous specification and is expressed as follows.

$$\forall t \in [0, 1]: \begin{cases} \phi_{path}(t, x_A(t), p(t)) \le 1 \wedge \\ \phi_{Inside}(x_A(t), x_B(t)) \le 1 \wedge \\ \phi_{Coms}(x_A(t), x_B(t)) \le 1 \wedge \\ \phi_0(x_A(t_0), x_B(t_0)) \le 1. \end{cases} \quad (3.1)$$

We obtain this formulation from the instantiated plan, where the variables X and Y are bound to vehicles A and B. In practice, we extracted the state-constraints from the configuration style and from the path specification.

Since the plan specification does not fully specify the behavior for vehicle B, we seek the least restrictive controller for this vehicle.

Informally, the least restrictive controller is a set-valued map from the current state to subsets of the control space from which the actual control is selected. It is the largest of such set-valued maps.

Under some mild assumptions (e.g., the control range constraint set is convex and has a nonempty interior), the joint set inclusion constraints on the values of the control variables induces a natural partial order. The selection of a value to one of the control variables, say u_1, defines a subset of one dimension lower of the range control set, say Ω_1 on which the values of the remaining controls are. For values of u_1 in certain domains, relations of the following type may be established:

$$u_1^1 > u_1^2 \longrightarrow \Omega_1(u_1^1) \subset \Omega(u_1^2).$$

A controller is said to be least restrictive if it takes a feasible value that yields a constraint set for the remaining control components which is maximal. In what follows, we consider the following hypotheses:

H1. The set-valued map P is closed, convex, and bounded.

H2. The path p is continuous in t.

H3. $\phi_0(x, y)$ is continuous in both variables.

Lemma. Under hypotheses H1 and H2, the functions $\phi_{path}(t, x, y)$, $\phi_{Inside}(x, y)$, and $\phi_{Coms}(x, y)$ are continuous and convex in x and y.

Define $\phi(t, x, y)$, u, and $\mathcal{U}$ as follows:

$\phi(t, x, y) = \max\phi_{path}(t, x, p(t)), \phi_{Inside}(x, y), \phi_{Coms}(x, y)\}$,

$u = \{u_A, u_B\}$,

$\mathcal{U} = \mathcal{U}_A \times \mathcal{U}_B$,

$f(t, x_A, x_B, u) = col(f_A(t, x_A, u_A), f_B(t, x_B, u_B))$.

We use the approach from [7] to formulate this control problem as an invariance problem (see [8–11]). To do this, we introduce the following value function:

$$V(t, x_A, x_B) = \min_{u(\cdot)}\{\max\{\max_{\tau \in [t_0, t]} \phi(\tau, x_a[\tau]x_B[\tau]), \phi_0(x_A(t_0), x_B(t_0))\}, \; x_A[t] = x_A, x_B[t] = x_B\}, \quad (3.2)$$

where $u(\cdot)$ is a feasible control function ($u(\tau) \in \mathcal{U}$, $\tau \in [t_0, t]$).

Now consider the sublevel set of this value function given by the following equation:

$$R(t, x_A, x_B) = \{(x_A, x_B): V(t, x_A, x_B) \le 1\}. \quad (3.3)$$

This set gives, for time t, the set of all the locations for both vehicles that satisfy Eq. (3.1).

Note that the set-valued map P is such that it ensures that A is able to avoid any obstacle by controlling the z variable only.

4. SOLUTION

In general, the value function V can be calculated through the generalized Hamilton–Jacobi–Bellman (HJB) equation. We can only do this if the value function satisfies the principle of optimality.

Theorem 1. The value function V satisfies the principle of optimality.

Using the techniques from [7], we can derive the HJB equation for this problem. First we introduce some notation:

$\mathcal{H}(t, x, y, V, u) = V_t(t, x, y) + (V_x(t, x, y)f(t, x, y, u))$.

The HJB equation for this problem is given by

$$\begin{cases} V_t(t, x, y) + \max_{u \in U}(V_x(t, x, y), f(t, x, y, u)) = 0, \\ \text{when } V(t, x, y) \ne \phi(t, x, y) \\ \max_{u \in U}\{\min(\mathcal{H}(t, x, y, V, u), \mathcal{H}(t, x, y, \phi, u))\} = 0, \\ \text{when } V((t, x, y),) = \phi(t, x, y) \\ V(t_0, x, y) = \max(\phi(t_0, x, y), \phi_0(t_0, x, y)), \end{cases} \quad (4.1)$$

where V_t, V_x represent the corresponding subdifferentials. This results from the fact that the value function is generally nondifferentiable, and we have to use generalized notions of derivatives.

Since V is nondifferentiable, the usual notion of solution of a partial differential equation does not apply. We consider generalized "viscosity," or equivalent concepts, of solutions for this equation (see [12, 13]). The following theorem asserts that the value function (3.2) satisfies the conditions for the existence of "viscosity" solutions. The proof technique can be found in [12].

Theorem 2. The value function V is the unique viscosity solution of (4.1).

Given a solution V to the Hamilton–Jacobi–Bellman equation, we are able to find the invariant set R. Now we have all of the ingredients required to synthesize the controller for our problem (see [13]).

Define $U(t, x_A, x_B)$ as the set of control values where the maximum for the Eq. (4.1) is attained when x_A and x_B are the values of the state of vehicles A and B at time t.

In the interior of R, we can use any feasible control. On the boundary of R, the control selection is restricted to the set-valued map $U(t, x_A, x_B)$.

5. EXAMPLE

Let us provide a very simple instance of the general class of coordinated control problems described in the second section of this article.

Vehicle A, the Mapper, moves in a straight line on the horizontal plane, say along the x axis, at a constant speed $V_A = 1$ and a constant depth. Vehicle B, the Scout, moves also at a constant linear speed $V_B = \frac{\pi}{2}$ While the dynamics of the mapper are simply given by $\dot{x} = 1$ and $\dot{y} = 0$, the Scout is a carlike vehicle with dynamics given by

$$\begin{cases} \dot{x} = \frac{\pi}{2}\cos(\omega) \\ \dot{y} = \frac{\pi}{2}\sin(\omega) \\ \dot{\omega} = \omega, \end{cases}$$

where $\omega \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and the initial position and orientation of the Mapper and of the Scout are, respectively, given by (0, 0, 0) and by (–1, 0, 0).

The control problem consists in finding a motion control strategy for the Scout so that constraints (3.1) are satisfied which, in this particular instance, are given by

$$\varphi(t, (x, y)) = (x - t)^2 + y^2 \le 1.$$

Let us consider the associated value function as defined in (3.2) with $x_A = (x, y, \omega)$ and x_B as the given fixed Mapper trajectory $(t, 0)$, $\phi = \varphi$, and $\phi_0 = -\infty$.

It is straightforward to conclude that the function ω^* undefined by $\omega^*(t) = (-1)^i\frac{\pi}{2}$, $\forall t \in [2i - 1, 2i + 1]$, $i = 0,1,\ldots$ is a feasible control policy since the corresponding trajectory (x^*, y^*), given by $x^*(t) = 2i - \sin(\pi(i - \frac{1}{2}t)$ and $y^*(t) = -\cos\left(\frac{\pi}{2}t\right)$, satisfies $V(t, x^*(t), y^*(t)) = 1$. Furthermore, $V(t, x, y) \ge 1\ \forall (x, y) \in \Re^2$.

It is not difficult to see that a feedback form of the control trajectory obtained by the maximization in (4.1) is given by $w^*(t) = k(x(t), y(t))$ where $k(x, y) = (-1)^i\frac{\pi}{2}$ being $i \in \operatorname{argmin}\|(2i, 0) - (x, y)\|_2$.

CONCLUSIONS

In this paper, we outline an integrated specification, planning, and control synthesis framework for networked vehicles systems. One of the main challenges to Control Theory comes from the distributed nature of the problem. For example, information and commands are exchanged among multiple vehicles, and the roles, relative positions, and dependencies of those vehicles change during operations. This challenge entails a shift in the focus of control theory—from prescribing and commanding the behavior of isolated systems to prescribing and commanding the behavior of interacting systems [14].

We use a simple motion coordination problem involving two vehicles to illustrate the framework.

ACKNOWLEDGMENTS

The authors thank Professors Pravin Varaiya and Alexander Kurzhanski for stimulating discussions and valuable comments, insights and contributions. The authors also thank Dr. Raja Sengupta for the motivation for the UAV example.

The authors gratefully acknowledge a grant of INVOTAN and support from Fundaçâo da Ciência e Tecnologia research project CorDyAL and from Fundaçâo das Universidades Portuguesas research project COOP.

REFERENCES

1. J. B. Sousa and F. L. Pereira, "Specification and Design of Coordinated Motions for Autonomous Vehicles," in *Proceedings of IEEE 2002 Conference on Decision and Control, Las Vegas, Nevada, USA, 2002.*
2. J. B. Sousa, A. Matos, and F. L. Pereira, "Dynamic Optimization in the Coordination and Control of Autonomous Underwater Vehicles," in *Proceedings of IEEE 2002 Conference on Decision and Control, Las Vegas, Nevada, USA, 2002.*
3. T. Curtin, J. Bellingham, and D. Webb, et al., "Autonomous Ocean Sampling Networks," Oceanography **6** (1993).
4. J. B. Sousa and A. Gollu, "A Simulation Environment for the Coordinated Operation of Multiple Autonomous Underwater Vehicles," in *Proceedings of the 1997 Winter Simulation Conference, 1997.*

5. B. J. Sousa and R. Sengupta, "CDC Tutorial on Autonomous and Semi-Autonomous Networked Multi-Vehicle Systems," http://www.fe.up.pt/ Ists/cdctutorial, 2001.
6. D. Van Cleave, *Software-Enabled Control: Information Technology for Dynamical Systems*, Ed. by T. Samad and G. Balas (Wiley, 2002).
7. A. B. Kurzhanskii and P. Varaiya, "Optimization Methods for Target Problems of Control," in *Proceedings of Mathematical Theory of Networks and Systems Conference, 2002.*
8. A. B. Kurzhanskn, *Advances in Nonlinear Dynamics and Control: A Report from Russia* (Birkhauser, 1993).
9. A. B. Kurzhanskii, *Ellipsoidal Calculus for Estimation and Control* (Birkhauser, 1997).
10. J. P. Aubin and H. Frankowska, *Set-Valued Analysis* (Birkhauser, 1972).
11. J. P. Aubin, *Viability Theory* (Birkhauser, 1991).
12. L. C. Evans, *Partial Differential Equations, Graduate Studies in Mathematics* (American Mathematical Society, 1998).
13. N. N. Krasqvskii and A. I. Subbotin, *Game-Theoretical Control Problems* (Springer-Verlag, Berlin, 1998).
14. T. Simsek, J. B. Sousa J., and P. Varaiya, "Communication and Control in Hybrid Systems," in *Proceedings of the American Control Conference, Washington, USA, 2001.*