Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 18 (2005) 513–531
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

A class of intelligent agents for coordinated control of outdoor terrain mapping UGVs

K. Fregene[a,*], D. Kennedy[b], R. Madhavan[c], L.E. Parker[d], D. Wang[e]

[a]*Honeywell Laboratories, 3660 Technology Drive, Minneapolis, MN 55418, USA*
[b]*Department of Electrical and Computer Engineering, Ryerson University, Toronto, Ontario, Canada M5B 2K3*
[c]*Intelligent Systems Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-8230, USA*
[d]*Department of Computer Science, University of Tennessee, Knoxville, TN 37996-3450, USA*
[e]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*

Received 21 February 2003; received in revised form 3 February 2004; accepted 18 December 2004
Available online 2 March 2005

Abstract

This article develops a systems- and control-oriented intelligent agent framework called the hybrid intelligent control agent (HICA) and describes its composition into multiagent systems. It is essentially developed around a hybrid control system core so that knowledge-based planning and coordination can be integrated with verified hybrid control primitives to achieve the coordinated control of multiple multi-mode dynamical systems. The scheme is applied to the control of a team of unmanned ground vehicles (UGVs) engaged in an outdoor terrain mapping task. Results are demonstrated experimentally.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Autonomous agents; Terrain mapping; Hybrid systems; Intelligent control

1. Introduction

One challenge that faces the intelligent systems and control communities is how to respond to the increasing demand for more intelligent higher autonomy systems-of-systems. These range from combat applications which require teams of coordinated autonomous unmanned air and ground vehicles (UAVs and UGVs) to complex interconnected electric power systems made up of several interacting entities. For the autonomous vehicles scenario, each team member (e.g., small unmanned rotorcrafts and certain ducted fan vehicles) may exhibit both continuous-valued and discrete-event dynamic behaviour (i.e., a hybrid system). The challenge is to synthesize low-level control schemes for the various modes of the vehicles which interact seamlessly with higher-level logic mechanism used to coordinate and/or supervise such a vehicle team. Owing to their inherent complexity, problems like these have been decomposed and modelled as systems of multiple interacting *intelligent agents* (Wooldridge, 1999; Jennings et al., 1998; Weiss, 1999; Georgeff and Rao, 1998).

For the purposes of this article, we define an agent as a system/process which is capable of sensing, computing and acting within an environment it occupies so that its actions are carried out in a flexible autonomous manner to optimize the attainment of its objectives and to affect its surroundings in a desired manner. Three broad classes of agents are identifiable (more specific classifications may be found in Russell and Norvig (1995)). In a *reactive* agent, perception is tightly coupled to action without the use of any symbolic or abstract internal model of the agent, the environment or their time

*Corresponding author. Tel.: +1 612 951 7521; fax: +1 612 951 7438.
E-mail addresses: kocfrege@ieee.org (K. Fregene), dkennedy@ee.ryerson.ca (D. Kennedy), raj.madhavan@ieee.org (R. Madhavan), parker@cs.utk.edu (L.E. Parker), dwang@kingcong.uwaterloo.ca (D. Wang).

0952-1976/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.12.007

histories. A *deliberative* agent system reasons and acts based on an internal model of both itself and the environment it inhabits. *Hybrid* agent systems combine aspects of both reactive and deliberative agency. The HICA belongs to this last category.

The field of multi-agent systems (MAS) is a vibrant part of distributed artificial intelligence with the broad aim of laying down the principles for the construction of complex systems involving multiple agents and providing tools/mechanisms for coordination of independent agent behaviours. A MAS environment is intended to interconnect separately developed agents so that the ensemble performs better than any one of its members. They also make it possible for the problem of constraint satisfaction to be sub-contracted to different problem-solving agents each with its own capabilities and goals. This MAS paradigm provides structures for building systems with high autonomy and for specifying interaction and coordination rules among agents. The core objective is the solution of problems too large for just one centralized agent to deal with, for instance control and/or resource allocation problems in distributed systems like air traffic control (Georgeff and Rao, 1998), telecommunication networks (Hayzelden and Bigham, 1998), industrial process control (Velasco et al., 1996) and electric power distribution (Wittig, 1992). Specific instances of successful practical application of MAS in these and other areas are summarized in the excellent work by Parunak (1999). The concept of agents situated in an environment is adapted from that work and shown in Fig. 1.

While the subfield of multiagent systems has a great deal to offer in terms of encoding flexible intelligent behaviour and coordination, its solutions are not easily represented or analyzed mathematically. By the same token, while there are many elegant mathematical results from decentralized control theory (Šiljak, 1996), systems based on these tend to be brittle and have primitive coordination abilities. Additionally, the class of systems to which much of the theory applies is still restricted.

For systems and control applications, it is important that there be transparent means of embedding control laws in an agent and for analyzing the resulting system behaviour. Since autonomy is central to the notion of agents, it is equally important to develop schemes which allow flexible coordination among agents. Accordingly, the key premise of this work is that future high autonomy systems-of-systems in which each system exhibits hybrid behaviour require *both provably stable control schemes (obtainable by applying control theoretic tools) and intelligent supervision and coordination abilities (developed by using tools from the subdiscipline of multiagent systems)*. We submit that such an agent framework that lends itself to mathematical representation/analysis requires a modification to the basic agent paradigm to make its dynamics accessible. In the modified agent framework, a hybrid control system is embedded in the core of the agent. Essentially, the abstract 'agent state' in Fig. 1 is replaced by a hybrid automaton and an associated controller. This makes it possible to transparently represent systems with multiple modes and to design suitable control laws that are valid for these modes. It also provides a more intuitive scheme to design for achieving specific control objectives for systems with multiple modes without sacrificing the intelligence and coordination inherent in the 'agent process' of Fig. 1. Additionally, mathematical tools from the theory of discrete event systems, hybrid systems and automata theory become applicable for the description of systems based on such enhanced agents.

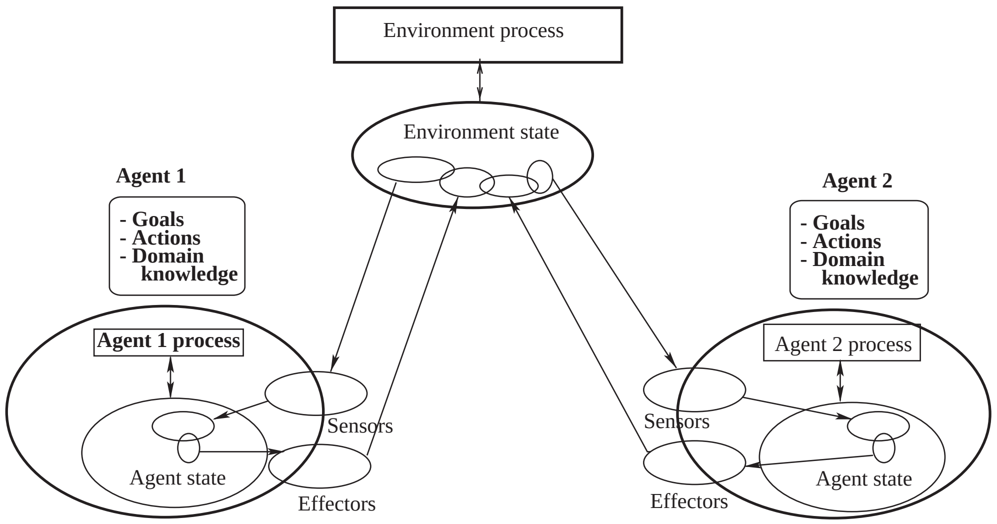

Fig. 1. Agents in an environment.

The hybrid intelligent control agent (HICA) represents our effort to enhance the intelligent agent paradigm in such a systems- and control-oriented manner. HICA is an intelligent agent wrapped around a hybrid control system core. It is intended to provide requisite infrastructure for the modelling and coordinated control of multiple multi-mode dynamical systems (see Fregene et al. (2003b) for details of an application to UAV/UGV pursuit-evasion).

In outdoor terrain mapping using a team of UGVs, the objective is to control this system of multiple agents while it traverses the terrain and develops maps that contain information about the location of objects/features in the environment and what the elevation gradient (or variation) is across the area. This type of information is typically used in path planning. In order to do this, each member the team of UGVs is modelled as a HICA with its own embedded hybrid control system, planning and coordination logic. In this way, all the functionalities required to accomplish the mapping task are embedded within the agent. The terrain mapping algorithm is based on merging multiple local maps obtained by each member of the vehicle team during specific motion segments into a globally consistent metric map.

The rest of this article is organized as follows: Section 2 contains a review of the literature in the areas that pertain to the subject matter of this article. Section 3 describes the HICA agent framework and its composition into specific classes of multiagent systems. The UGV team is described and modelled as HICA agents in Section 4. The actual multivehicle terrain mapping scheme is discussed in Section 5 followed by the synthesis of the coordinated control logic for mapping in Section 6. The various control modes are associated with specific mapping subtasks in Section 7. Experimental results and concluding remarks are provided in Sections 8 and 9, respectively.

2. Related work vs present work

In the area of agent-based control systems, Stothert and McLeod (1997) present distributed intelligent control application to a deep-shaft mine winder using fuzzy logic. Although the formulation is amenable to limited analysis, it is not well suited for problems involving multiple agents acting in realtime on uncertain information. Velasco et al. (1996) present a scheme in which the agents encapsulate different distributed hardware and software entities like neural/fuzzy controllers, data acquisition systems, etc. defined in an agent description language. However, the agent depends strongly on having accurate models of both itself and the environment; realtime issues are not convincingly addressed and it is not a formal repeatable synthesis scheme for controller design in diverse applications. Voos (2000) has recently presented some perspectives on agent-based intelligent control and a preliminary design which was applied to a wastewater plant. Apart from not being formally specified, the approach therein hardly exploits reactivity to the right extent so that realtime constraints are not satisfied. Perhaps one of the most comprehensive work on multiagent-based control systems is due to van Breemen (2001); van Breemen and de Vries (2001). In this work, a scheme for decomposing complex control problems so that multiple agent based controllers can be systematically applied to them is reported. Simulation and experimental results are presented to illustrate its applications to real-life problems. Control is typically of the unimodal PID/fuzzy variety. It is not apparent how processes with multiple modes (hybrid systems) would be accommodated in this scheme. Moreover, systematic analysis was not done to, for instance, guarantee properties of the resulting system such as stability.

In the area of the control of multiple agents with a hybrid systems flavour, Tomlin et al. (1998) present a hierarchical architecture and algorithms for conflict resolution and aerodynamic envelope protection in air traffic control systems using nonlinear hybrid automata and game theory. A hierarchical scheme for the control of large-scale systems with possibly semi-autonomous agents is presented in Lygeros et al. (1998) in which 'a priori' verified hybrid controllers are introduced to minimize the need for automated system verification. Along the same lines, a hierarchical architecture for multiagent multi-modal systems is proposed in Koo (2001). Bi-simulation and formal approaches to hybrid system design are applied in evaluating the proposed architecture. Furthermore, ongoing research efforts in the area of multiagent hybrid control in large-scale systems using methods of artificial intelligence are reported on the web (2002). But the work appears to be just beginning since, to the best of the authors' knowledge, results have yet to be published in the public domain. Nerode and Kohn (1993) proposed a multiagent hybrid control architecture as a software system for realtime implementation of distributed controllers. The architecture uses principles of declarative control, concurrent programming and dynamical hybrid systems so that each agent's control action is realized by solving a convex optimization problem.

In the arena of mapping using cooperating autonomous vehicles, Burgard et al. (2000) detail an explicit coordination mechanism that assigns appropriate target points to indoor robots in a team such that they effectively explore different regions using an occupancy grid map that is built based on the data sensed by individual robots. Relative position between robots is assumed to be known. A dead-reckoning-free robust positioning system for global mapping using multiple mobile robots is described by Dudek et al. (1993). Therein, robots define a local coordinate system without reference to environmental features. Although sensing errors remain localized to the individual robot, the resulting global map describes the neighbor relations between the robots in the form of a graph. Rekleitis et al. (1998) also report a graph-based exploration strategy for map construction in which two robots act in concert to reduce odometry errors. Each robot is equipped with a *robot tracker* sensor which tracks a geometric target installed on the other robot visually. The limitation of the approaches in Dudek et al. (1993) and Rekleitis et al. (1998) lies in the fact that the robots cannot be distributed over the operating environment as they are required to remain close to each other to maintain visibility. Cooperative localization and occupancy-grid mapping of two homogeneous indoor robots each equipped with a stereo-vision head is described by Jennings et al. (1999) and Little et al. (1998). Although grid-based maps have the advantage of allowing explicit modeling of free-space and ease of fusing data from different sensors Grabowski et al. (2000), they are often impractical for large unstructured environments due to the fixed grid-size and the accompanying computational burden. Other efforts in this area include the work of Howard and Kitchen (1999) and references therein.

In the present work, no arbitrary hierarchical decompositions are assumed or imposed. Rather, the modelling flexibility of hybrid systems and the facilities for coordination in intelligent multiagent systems are combined to develop a hybrid intelligent control agent in which the hybrid control and intelligent agency are parallel co-operating subsystems. Such an agent can be used as the basis for higher autonomy control and coordination of multiple dynamical systems since its development is properly cognisant of each system's natural dynamics.

Apart from concentrating primarily on indoor environments, many of the mapping schemes in the literature teleoperate the robots used for mapping. Even in cases where they operate autonomously, all vehicles are controlled from one central location. In addition to the robustness issues inherent in such centralized control schemes, systematic coordinated control synthesis is not done. In this work, vehicle control for terrain traversal is systematically developed as a set of hybrid control primitives sequenced by local deliberative planning/coordination on each vehicle. For mapping, a coordinate frame is centered at the location of a Differential Global Positioning System (DGPS) base station in the environment of interest. This ensures that the problem of merging locally generated maps is simplified. Furthermore, the scheme we present has no restriction on how many vehicles can move at any instant. Moreover, the sensors required are not unduly sophisticated. Mapping is decentralized and takes place in a realistic outdoor environment.

3. The HICA agent system

The HICA is an intelligent agent which has a hybrid control system at its structural core. It may also be properly conceived of as a hybrid control system which has an intelligent agent "wrapped around" it. Its internal structure is depicted in Fig. 2. Conceptually, it consists of local deliberative *planning/coordination* and reactive *hybrid control* sub-systems. The hybrid control system is itself made from a set of verified control primitives represented by $\{f_c, \eta_c, h_c\}$ in Fig. 2 and the controlled process or the plant represented by $\{f_p, \eta_p, h_p\}$ which are appropriately mapped to other agent variables.

The controlled process is, in this case, wholistic. That is, HICA models the dynamics of the plant *as a whole*. Therefore, full-authority control via the sequencing of verified hybrid controllers suitable for each discrete mode of the system is achievable. This framework is different from other agent-based control done by decomposition (e.g., van Breemen, 2001) in which different components of the system are represented by an agent and the overall system control problem becomes an appropriate interaction among all of these agents. For instance, consider the guidance, control and navigation of a helicopter-based UAV. The HICA approach models all the dynamics and flight modes of the UAV in one agent's `controlled process` rather than represent components like servos, blade rotor dynamics, actuators, etc. by several different agents which interact. The main advantage is that, in a system involving, for instance, *teams* of such UAVs, the number of agents that would have to be developed, accounted for and for which coordination protocols need to be developed is significantly reduced. The result is a simpler multiagent system abstraction.

The intelligent agent "wrapped around" the hybrid control system consists of local deliberative planning/coordination as well as interfaces to the environment. We say that this agent is "hybrid" partly because of the type of control system it encapsulates and partly owing to the fact that it combines deliberative planning/coordination with reactivity of the hybrid control system. It applies deliberative logic to its inputs and develops a control policy which the hybrid control system executes. An output from the agent is used to facilitate coordination with other agents.

The *hybrid state* of the HICA is given by the pair (Q, X) where Q is the set of discrete states and X is the set of continuous states. We shall henceforth refer to the discrete state simply as the *mode* of the agent. The other variables associated with Fig. 2 are described next.

- Y is internal to the agent and represents the output from the controlled process.
- Σ is a discrete control output from the hybrid control subsystem. It also (partially) determines *which* mode the agent transits to.

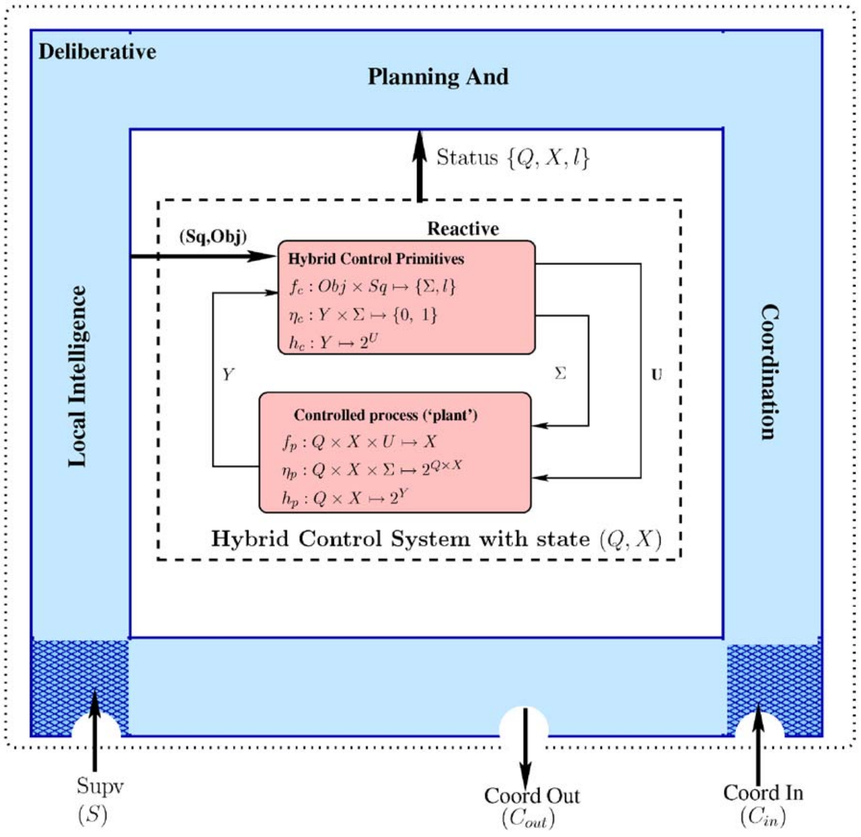
Deliberative
Planning And
Local Intelligence
Coordination
Status $\{Q, X, l\}$
Reactive
(Sq,Obj)
Hybrid Control Primitives
$f_c : Obj \times Sq \mapsto \{\Sigma, l\}$
$\eta_c : Y \times \Sigma \mapsto \{0, 1\}$
$h_c : Y \mapsto 2^U$
Y
Σ
U
Controlled process ('plant')
$f_p : Q \times X \times U \mapsto X$
$\eta_p : Q \times X \times \Sigma \mapsto 2^{Q \times X}$
$h_p : Q \times X \mapsto 2^Y$
Hybrid Control System with state (Q, X)
Supv (S)
Coord Out (C_{out})
Coord In (C_{in})

Fig. 2. Internal structure of an HICA agent.

- U is the continuous control applied to the plant. It has the most effect on the continuous state evolution of the agent within each mode.
- $C = [C_{\text{in}} C_{\text{out}}]^{\text{T}}$ is a set of *coordination factors* input to (or output from) the agent. It basically encapsulates the interaction rules for different HICA agents. So, when it is necessary for agents to influence one another in a cooperative multiagent system or for conflict resolution, these coordination factors serve as external inputs to the agent to achieve the objective. The local planner in each agent contains logic to translate these factors into actual action taken by the agent—e.g., in the form of a mode transition.
- S represents *supervisory commands*, usually from a higher level supervisory agent regarding strategic actions and overall mission objectives. Again, the local deliberative intelligence contains logic to translate supervisory commands to actual agent actions.
- Sq is the sequence of control primitives to achieve the objective. It is written as

  $$Sq = (Sq[l]), \ l = 1 \cdots n < \infty,$$

  where l is the *sequence index*. It provides a measure of how far along the agent has gone in completing the current mode sequences. Given that n is the number of modes in the sequence, the current sequence then has the form

  $$(Sq[1], Sq[2], \ldots, Sq[n]).$$

  It is generated from the local planning sub-system using embedded logic, current objectives, prior sequences, external inputs and feedback from the controlled process.
- Obj is the short-term objectives to achieve the overall goal of the agent. It can take the form of the desired mode of the controlled process or reference signals to track, etc. There are specific mode sequences associated with each short term objective.
- The `Status` represents a direct feedback of hybrid state (Q, X) and sequence index l to the local planning sub-system. This takes the form of a set $\{Q, X, l\}$ as shown in Fig. 2.

A conceptual view of the parallel cooperating subsystems inside the HICA is shown in Fig. 3.

The hybrid control system also contains `fail-safe` primitives which are primarily concerned with the safety of the agent in the event that sequences and short-term objectives are not provided to it in a timely manner. So, the general operation is to execute whatever sequences

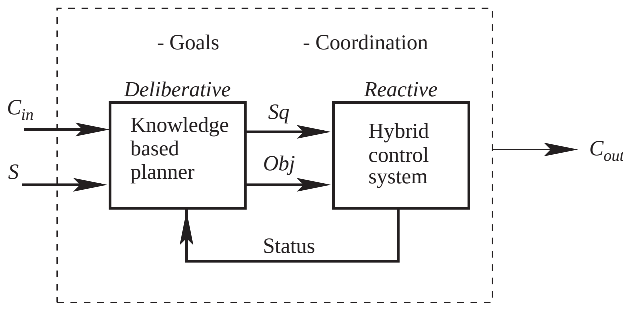

Fig. 3. Inside HICA—parallel cooperating subsystems.

arrive to meet objectives. If none is available, then $Sq =$ `fail-safe` and $Obj =$ `safety`.

3.1. Planning and coordination in HICA

The problem of planning and coordination in HICA is that of finding a sequence of modes which is guaranteed to achieve the (short-term) objectives of each agent while it coordinates its activities with other agents to attain team goals. The conflict implicit in this requirement is addressed by using a partial knowledge-based planning (PKBP) and coordination scheme (see Fregene, 2002 for more details). Essentially, the agent does not wait until a complete sequence is generated before these sequences are executed. That is, the PKBP planner does not necessarily generate a complete plan to the objective in one pass. A partial plan may be executed to meet real-time constraints or when portions of the generated mode sequences violate a priori determined safe transitions. It generates the short-term objective and mode sequences to accomplish this objective using the processes PLAN-OBJ and PLAN-SQ as illustrated in the pseudo-code of Algorithm 1.

The planner starts by generating an *Obj* from the goal. Then it checks the knowledge base (KB) for modes and/or sequences which support the current *Obj*. If all the required information is there already, the *Sq* and *Obj* are output to the hybrid control system which executes them. If portions of the required information are not in the KB, the planner next checks prior sequences and finally generates a random sequence by progressively expanding the nodes in the automaton which represents deliberative planning. If a particular transition violates safe transitions as previously defined, the *Sq* generated up to that point is executed and a replanning process begins. The last step is to update the KB when sequences not already there are shown to accomplish a particular *Obj*. The sequence index *l* is used in all cases to keep track of plan execution.

Because there is a verified hybrid control primitive associated with each mode, the job of the local planner does not include dynamically synthesizing the controller. It is rather that of selecting from available control primitives and sequencing them indirectly by providing desired mode sequences.

Although the emphasis is on making each agent as fully autonomous as possible, it is often necessary for several agents to coordinate their activities. Coordination factors are codified at design time so that each agent is able to translate them into actual conflict scenarios that may be happening. Some such inputs lead to replanning while others (e.g., those involving safety) fall within the purview of the fail-safe control primitives in the hybrid controller. For these, appropriate primitives are invoked in a specific manner. Like the PKBP scheme, coordination in HICA is also a knowledge-based affair. The knowledge-base (KB-COORD) contains appropriately codified coordination factors and their interpretations. It also contains the codes for different coordination outputs. These are closely linked to the agent goal since coordination inputs from other agents may require a change in plan. A pseudo-code for the coordination process is shown in Algorithm 2.

In the literature, the planning/coordination action heretofore described is variously represented using tools like predicate logic, propositional logic, temporal logic, finite automata etc. (Passino and Antsaklis, 1989). We adopt the finite automata representation because it provides a more intuitive way to express interfaces with hybrid control laws. The action of the planner is given by four major logical entities (B, w, ψ, p).

Definition 1 (*HICA Planner*). The HICA planner is the quadruple

$$\mathscr{H}p = (B, w, \psi, p)$$

Algorithm 1 HICA-PKBP planner

```
repeat
  PLAN-OBJ()
  PLAN-SQ()
until goal is achieved
PLAN-OBJ($C_{in}$, $S$, status)
uses: Goal, KB, $l$
Generate $Obj$ from goal
Output $Obj$ to PLAN-SQ()
if $Obj$ is attained then
  Generate a new $Obj$ from goal
  Output new $Obj$ to PLAN-SQ()
else
  Output current $Obj$ to PLAN-SQ()
end if
PLAN-SQ($Obj$, status, $Sq$)
uses: Safe transitions, KB, Prior sequences, $l$
Query KB for useful sequence
if Complete $Sq$ exists then
  Output $Sq$ and $Obj$
else if Partial $Sq$ exists then
  Output partial $Sq$ and $Obj$
  Generate remaining $Sq$ to attain $Obj$
```

```
else
    Output $Obj$
end if
Generate $Sq$
if $Sq$ violates safe transitions then
    Output $Sq$ up till offending mode
    Output $Obj$
    Re-plan using randomly generated $Sq$
end if
Update KB with successful new $Sq$
```

where

- B is a map which translate inputs, goal and agent current status to short term objectives.
- w is the external input-to-hybrid state logic function.
- ψ is the hybrid state-to-output logic function.
- p is a map which represents the underlying transition logic used in the local planner to generate feasible mode sequences. It is formed from verified safe transitions, objectives and prior sequences.

Algorithm 2 HICA-COORD (C_{in}, Status, S) coordination scheme

```
uses: KB-COORD, $l$
if Incoming $C_{in}$ or $S$ then
    Query KB-COORD for corresponding action
end if
Update goal to reflect action
output goal
if Status dictates coordination with other agent then
    Query KB-COORD for appropriate $C_{out}$ code
    Output $C_{out}$
end if
```

This representation of planning allows us to formalize the notion of a *canonical* agent.

3.2. Canonical HICA

Without any loss of generality, a canonical representation subsumes many of the internal variables shown in Fig. 2 and provides a more intuitive model for analysis. It is depicted in Fig. 4.

Definition 2 (*Canonical HICA*). A canonical HICA agent is an *input-state state-output* generalization of the HICA agent. It is given by the 9-tuple

$$\mathscr{H} = (Q, X, f, W, w, B, p, \Psi, \psi),$$

where

- (Q, X) is the hybrid state while f corresponds to f_p in Fig. 2.
- $W \in C \times S$ is the set of inputs to the agent and is a finite set. For discrete modes $i, j \in Q$ there exists $W_{ij} \subseteq W$ such that if the system was in mode i and it receives an input which lies in W_{ij}, it switches to mode j. Define $W_r^+ = \bigcup_i W_{ir}$ as the *arrival input set* for mode r—i.e., the set of inputs which constrain the agent to switch *to* mode r. Also define $W_r^- = \bigcup_i W_{ri}$ as the *departure set*—i.e., the set of inputs which constrain the agent to switch *from* mode r. Then $\forall i, j, k \in Q$, *W_{ki} and W_{kj} are disjoint*. This is also true for W_r^- and W_r^+.
- The maps (B, w, ψ, p), which have been described in Definition 1, are written in compact form in (1). Note that w and ψ respectively correspond to Inlogic and Outlogic in Fig. 4.
  Apart from mode changes in response to external events, transitions of the agent from mode i to j also happen when the system satisfies some transition conditions associated with the domain ($W \times$ Status) of the map B. This is different from the transition sets for W since the Status features in the domain of the map B.
  Denote the domain of B by Υ, that is
  $$\Upsilon := W \times \mathtt{Status}.$$
  Then the transition sets are written as Υ_{ij} to indicate an i to j mode change. The *arrival* and *departure* sets, Υ^+ and Υ^- respectively, can be similarly defined.
- Ψ is the output set.

Using the various pieces of the canonical agent, the action of local planning as in Definition 1 is then given by

$$\text{Local planning} \begin{cases} B : W \times \mathtt{Status} \mapsto Obj, \\ p : Sq \times \Upsilon \mapsto Sq, \\ w : (Q, X) \times W \mapsto (Q, X), \\ \psi : (Q, X) \mapsto \Psi. \end{cases} \tag{1}$$

The canonical HICA is partitioned according to the subsystems in Fig. 2 as follows:

Local planning	:	$B, p, \psi, w,$
Hybrid control system	:	$(Q, X), f,$
External	:	$W,$
Common	:	$\Psi.$

The map B generates the set of short-term objectives while p generates the sequences to achieve the objective. Additionally, the mapping from input to state is provided by w while ψ provides logic which maps from state to the output set, Ψ. In reality Ψ is used together with status to determine what an appropriate coordination output C_{out} should be.

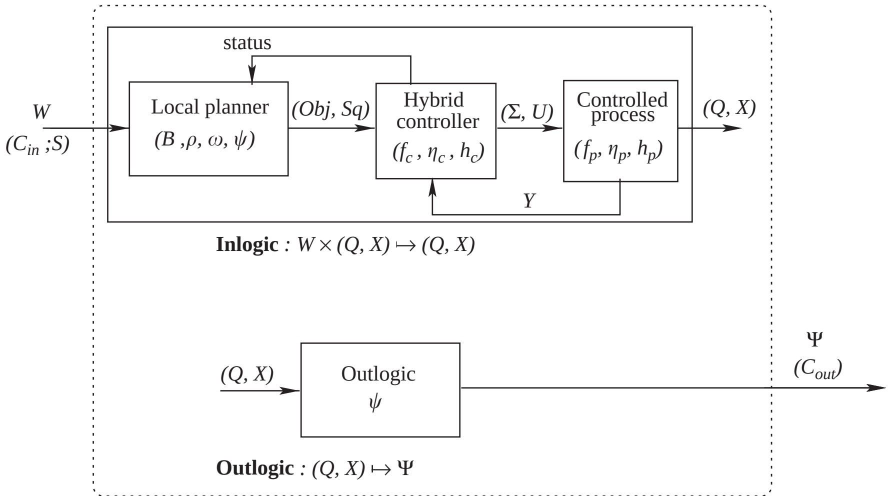

Fig. 4. Canonical agent logic.

3.3. Composing HICA into multiagent systems

We present two ways to organize the HICA into multiagent teams. In a *network of agents*, the output of each agent (Ψ) is available on a network as (a potential) input (W) to all the other agents apart from itself. This is depicted in Fig. 5 and is described by a special kind of composition operation with several desirable mathematical properties. In particular, it is shown in Fregene (2002) that a network of canonical HICA is itself a canonical HICA.

In a supervisory multiagent system, global coordination is enhanced by the introduction of a supervisor as depicted in Fig. 6. A supervisory *team* therefore consists of a supervisory agent connected to a network of agents. Accordingly, we shall refer to the supervisory agent as the team supervisory agent (TSA). It is depicted outside the dotted outline of Fig. 6. It receives as input the current output of the network of agents it supervises and supervisory coordination input from other TSAs and outputs appropriate supervisory events to the network of agents and coordination outputs to other TSAs. In Fig. 6, $C_{\{\cdot\}}$ denote the supervisory-level coordination factors while β_s denotes the supervisory logic. The frequency with which the TSA outputs supervisory events to the agent network is denoted by ϕ_s, where $0<\phi_s\leqslant 1$. On the whole, coordination takes place at three levels:

(i) *Topmost level*: At this level, coordination takes place between *teams* using the ($C_{\mathrm{TSA}_{\mathrm{out}}}$ $C_{\mathrm{TSA}_{\mathrm{in}}}$) coordination factors.

(ii) *Intermediate level*: Here, the coordination is between a supervisor and the network of agents it supervises. This is done using the supervisory input S.

(iii) *Lowest level*: The coordination here happens between and among individual agents in a team using C_{in} and C_{out}.

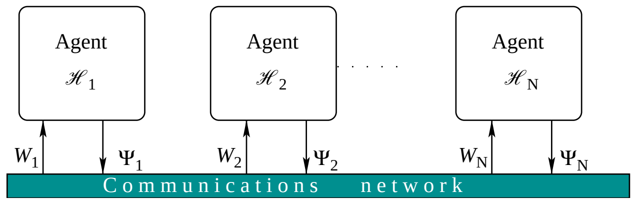

Fig. 5. A network of HICA agents.

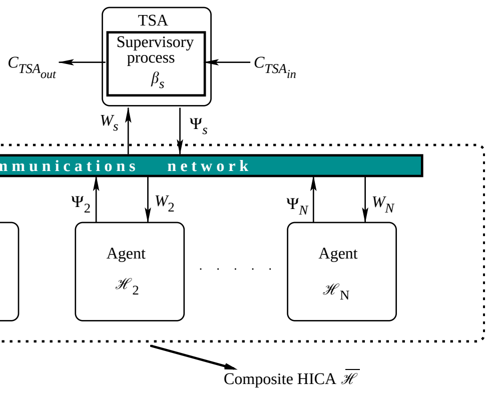

Fig. 6. A supervisory agent team.

Additional details of the TSA's action and some useful mathematical properties which enable team action to be characterized by a class of interconnected asynchronous discrete event systems can be found in Fregene et al. (2003a). The stability of multiagent systems based on the HICA has been analysed in such a discrete event framework.

4. HICA modelling of the vehicle team

In this section, we develop the representation of each UGV as a HICA and the multi-vehicle system as a supervisory HICA system.

4.1. UGV platform, experimental setup and sensor calibration

The UGV platform is an All TerRain Vehicle (ATRV) Mini wheeled mobile robot with 4-wheel differential-drive skid-steering. The experimental setup consists of two of these vehicles connected by a wireless mini-local area network (LAN) set-up directly outdoors to an operator console unit (OCU). The OCU is a rugged notebook computer equipped with a Breeze-COM access point and antennas. Each vehicle (which is essentially a node on the wireless network) has a BreezeNET station adapter and an antenna. Also available are local area differential GPS (LADGPS), a software environment for the vehicles and codes developed in-house under Linux to read and log the data for the sensors on each vehicle. The LADGPS is formed by the base station/antenna hardware connected to the OCU and remote stations/antennas directly mounted on each vehicle. Each vehicle's station receives differential corrections from the base station (sub-meter LADGPS accuracy was observed when 11 GPS satellites were acquired). A distributed Common Object Request Broker Architecture (CORBA)-based interface on the vehicles ensures that passing coordination and supervisory signals around the vehicle network happens in a transparent decentralized manner.

The sensor suite for localization and mapping (shown in Fig. 7) is comprised of encoders that measure the wheel speeds and heading, DGPS, magnetic compass, a

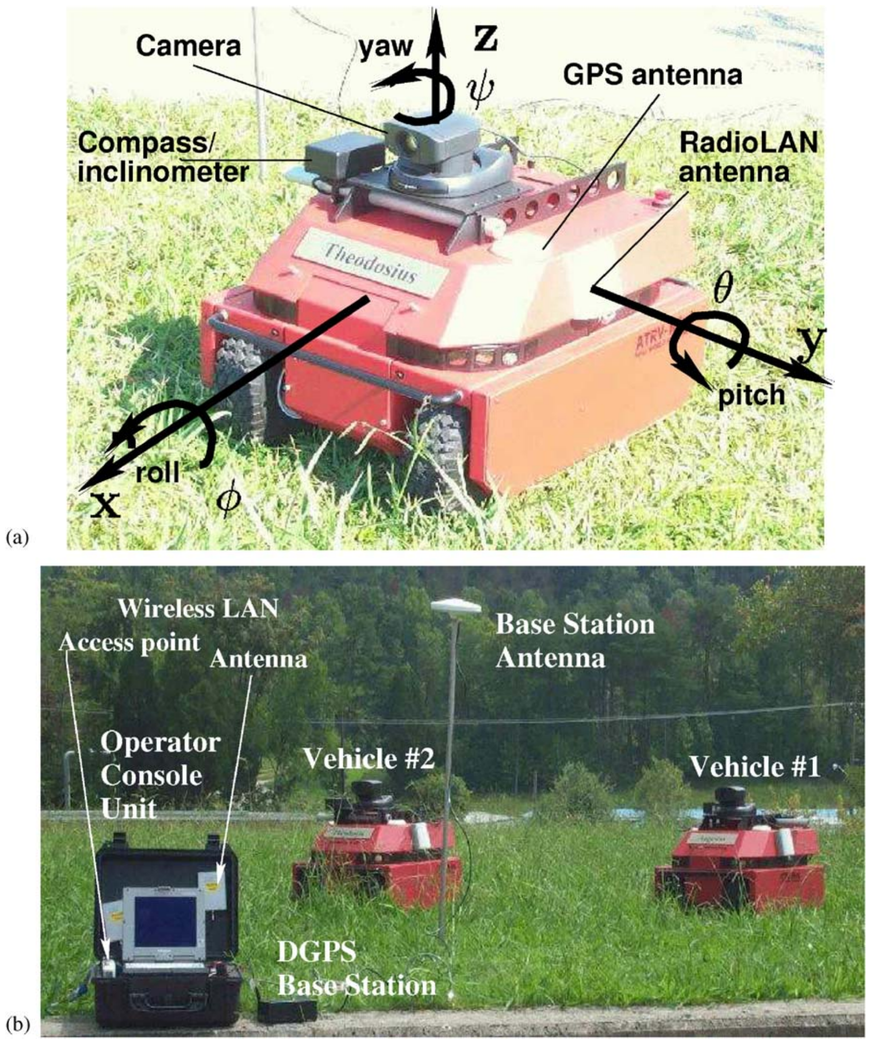

Fig. 7. The ATRV-Mini sensor suite and experimental setup. The sensor suite shown in (a) consists of encoders, DGPS, a compass and a PTZ camera (Also see Table 1). The experimental setup depicted in (b) consists of an operator console unit, a DGPS base station and a base station antenna. See text for further details.

Table 1
Sensor suite description

Sensor	Description	Freq. (Hz)
`Odometry`	Provides wheel speed and rate of change of heading (relative)	≈ 50
`DGPS`	Provides (x, y) position of vehicles (absolute)	≈ 1
`Vision`	Provides images using a pan-tilt-zoom (PTZ) capable camera (absolute)	≈ 10
`Compass`	Provides heading with respect to true north (absolute)	≈ 1
`Inclinometers`	Provides pitch and roll angles to true north (absolute)	≈ 1

pan-tilt-zoom (PTZ) capable camera for visual perception and inclinometers. The PTZ camera has a focal length in the range 5.4–64.8 mm (i.e., a variable zoom system). Its pixel resolution is 120×160 with a 37° field of view (FOV). The magnetic compass provides an external means of measuring the yaw angle. It is calibrated and corrected for magnetic variation in the Northern hemisphere. Inclinometers provide measurements of the body pitch and roll angles, respectively. Table 1 summarizes the sensor suite and its characteristics.

Since the experiments are carried out in an outdoor environment with the vehicles executing general motion (translation and rotation on all axes), sensor calibration is important to ensure accuracy of readings. For the encoder readings, external sensors (DGPS and magnetic compass) are used to obtain calibration factors corresponding to the various axes. The correction factor for magnetic compass is obtained by looking up geodesic charts to determine the angle of magnetic variation corresponding to the longitude/latitude of the experiment's location. During outdoor navigation, it is possible for the DGPS antenna to lose line of sight with the orbiting satellites (e.g., by going under a tree). To account for this, calibration trials were performed in which the error in the DGPS positions reported were obtained as a function of the number of satellites acquired (as an alternative, the so-called *dilution of precision* measure associated with the GPS can be used for the same purpose (see Grewal et al., 2001, pp. 18–24)). Because encoder readings are subject to drift during general motion, an extended kalman filter (EKF) based decentralized localization scheme described in Madhavan et al. (2002) serves to correct these errors in an incremental manner.

4.2. Modelling the agent

We now show how the mapping agent of Fig. 7a is modelled. As far as the HICA formulation goes, only the components that make up the hybrid control system are presented in this section. We shall save a description of the planning/coordination logic for Section 6.

Let x,y,α denote the x-coordinate, y-coordinate and heading of the vehicle's centre of gravity, respectively. Also let V and ω denote the corresponding translational and angular velocities. The coordinate frame is shown in Fig. 8. The equations of motion are

$$\left.\begin{aligned}\dot{x} &= V\cos\alpha\\ \dot{y} &= V\sin\alpha\\ \dot{\alpha} &= \omega\end{aligned}\right\}\textit{vehicle kinematics}$$

$$\left.\begin{aligned}m\dot{V} &= F\\ J\dot{\omega} &= \tau\end{aligned}\right\}\textit{force/torque dynamics}$$

in which m is the mass of the vehicle and (F, τ) is the force–torque pair applied at the vehicle's centre of mass. The force is in the same direction as V while the torque is positive in the counter-clockwise direction in Fig. 8.

In a general system, the control inputs would be (F, τ), but for autonomous vehicles, it is assumed that we have direct access to V and ω. In the experimental setup, the translational velocity and heading rate are specified directly on the vehicle's navigation software. Therefore, they will be used as our control signal and the force/torque dynamics given above may be ignored. This means that only vehicle kinematics will be used in the formulation that follows.

(A) *Nominal process model.* In continuous time, the nominal process model is given by

$$\begin{aligned}\dot{x}(t) &= V(t)\cos\alpha(t),\\ \dot{y}(t) &= V(t)\sin\alpha(t),\\ \dot{\alpha}(t) &= \omega(t),\end{aligned}$$

which has the general form

$$\dot{x} = f(X, U), \tag{1}$$

where X are the states (x, y, α) and U are the control inputs (V, ω). A discretized form of system (1) is more suitable for experimental implementation. Therefore the nominal discrete process model at time-instant kT, with $x[k]$ representing $x(kT)$

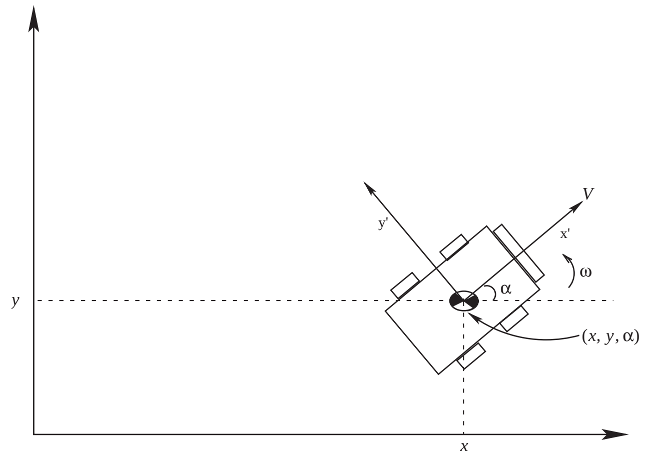

Fig. 8. Vehicle reference frame.

given by

$$\begin{pmatrix} x[k] \\ y[k] \\ \alpha[k] \end{pmatrix} = \begin{pmatrix} x[k-1] \\ y[k-1] \\ \alpha[k-1] \end{pmatrix} + \Delta T \begin{pmatrix} V[k]\cos\alpha[k-1] \\ V[k]\sin\alpha[k-1] \\ \omega[k] \end{pmatrix} \quad (2)$$

suffices for distributed localization of the vehicles (see Madhavan et al., 2002 for more details). In (2), ΔT is the sampling interval between states at discrete-time instants, $[k-1]$ and $[k]$. Observe that the entire RHS of (2) is a discretized form of the localized vector field $f(\cdot,\cdot)$ in (1) which is defined on each $[(k-1)T \;\; kT]$. It describes the state evolution of the agent. Denote this discretized field for mode q by f_q. The hybrid control primitives therefore modify f_q as the agent transitions from mode to mode.

(B) *Agent modes and state*. The state space of the agent is made of three states $[x[k] \; y[k] \; \alpha[k]]$ and five discrete modes $\mathbf{Q} = [q_1, q_2, \ldots q_5]$ which are fully described in Section 6.2.

(c) *Hybrid control primitives*. The control applied to the vehicle at any instant k within any mode q is given by

$\mathbf{u}_q[k] = [V[k], \omega[k]]'$.

These control primitives as well as local sequencing logic (discussed in Section 6) are used to implement the various motion segments of the mapping algorithm.

4.3. The multivehicle system—a supervisory HICA

The system shown in Fig. 7b consists of a network of mapping agents (the two UGVs) whose actions are moderated by the OCU. Therefore, the multivehicle terrain mapping problem is one that is readily represented by a supervisory HICA system. The TSA in this case is a software agent resident on the OCU. Detailed mathematical description of this multivehicle system in the HICA framework is developed in Fregene (2002).

5. Multivehicle terrain mapping

The UGVs accomplish terrain mapping by traversing an outdoor terrain systematically while combining vision-based range information of environmental features with an elevation profile across the terrain. This happens in four stages. Firstly, an incremental dense depth-from-camera-motion algorithm is used to compute ranges to various features in the environment. The relative pose and associated range covariances are used to specify regions with objects of interest. Then, an elevation gradient is determined by fusing DGPS altitude information with vertical displacements obtained from the inclinometer pitch angles as the vehicle moves. The range and elevation information are then registered with their associated covariances. Finally, the map is updated to incorporate the registered values at their proper coordinates. Covariances associated with these measurements provide a confidence measure of range determination. When more than one vehicles explore the same area, this confidence determines whether or not the map gets updated.

The overall schematic diagram of the algorithm is shown in Fig. 9.

Apart from the map update step, this algorithm runs in a decentralized manner on each vehicle. Therefore, the control routine runs in parallel with the mapping algorithm; the control algorithm is responsible for

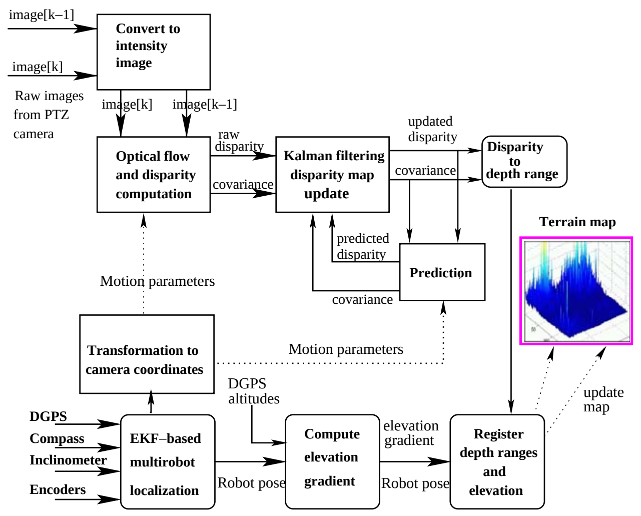

Fig. 9. The overall terrain mapping scheme.

navigating the vehicle while the mapping algorithm creates the map *during* these motion segments. The OCU contains both the global (updated) terrain map and a software agent that outputs supervisory factors to the vehicle network.

6. Coordinated control for terrain traversal

This section describes the control and coordination logic embedded in the vehicle team (represented as a supervisory HICA) for terrain traversal during mapping. The terrain traversal scheme is first described followed by details of the hybrid control primitives for each of the modes and the knowledge-based planning/coordination scheme adopted.

6.1. Terrain traversal procedure

Each member of the vehicle team traverses the environment systematically using the logic depicted in Fig. 10.

In the `scan` mode, the vehicle turns in place while scanning the environment noting objects of interest, their relative orientation and a confidence measure of which objects are more clearly observed than others. The most visible object is selected (`LockOn`) and its range is estimated (`AcquireImage`). The vehicle then moves towards this object while computing changes in the elevation across the terrain (`ElevGradient`). The vehicle turns away from the object (`ObjectAvoid`) when it gets too close but notes the object's location in the global frame so that it can be indicated on the global terrain map. A different object is selected and the procedure is repeated. The vehicle also backs away from regions where it may tilt over (`Boundary`). `Boundary` and `ObjectAvoid` are fail-safe modes to enhance the vehicle's survivability.

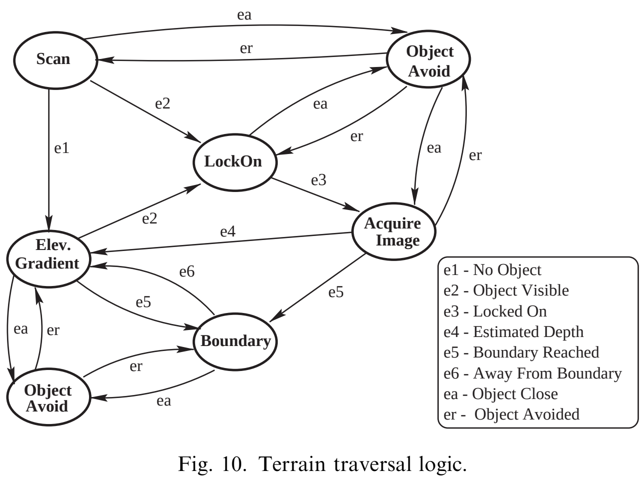

Fig. 10. Terrain traversal logic.

6.2. The control modes

Because the dynamics of the vehicle are not unduly complicated and the control objectives involve quite basic navigation, the individual control primitives for various stages of map-making are very simple. What our formulation provides is a way to embed all of these in an agent such that a suitable sequencing logic is generated to accomplish the objectives in the presence of coordination and supervisory constraints. In the discussions that follow, $K_{\{\cdot\}}$ is an appropriately selected gain. The modes are identified as follows:

(i) *Scan* (q_1). In this mode, the vehicle scans the environment noting objects of interest, their relative orientation and a confidence measure of which features are more clearly observed. The control and dynamics here are given by

$$\mathbf{u}_1[k] = \begin{bmatrix} 0 \\ K_1 \end{bmatrix},$$
$$f_1 = \begin{pmatrix} x[k-1] \\ y[k-1] \\ \alpha[k-1] + K_1\Delta T \end{pmatrix}.$$

The control $\mathbf{u}_1[k]$ ensures that there is no forward motion while the vehicle turns in place at a constant angular rate K_1. The system transitions to the `ElevGradient` or `LockOn` mode depending on whether or not objects are observed in the environment with sufficient confidence.

(ii) *LockOn* (q_2). The system transitions to this mode once objects have been sensed with sufficient confidence level. The controller essentially turns the vehicle until it is pointing directly at the observed object. It tracks a reference heading α_0 corresponding to the object with highest location confidence. The relevant equations in this case are:

$$\mathbf{u}_2 = \begin{bmatrix} 0 \\ K_2(\alpha_0 - \alpha[k-1]) \end{bmatrix},$$
$$f_2 = \begin{pmatrix} x[k-1] \\ y[k-1] \\ \alpha[k] + K_2(\alpha_0 - \alpha[k])\Delta T \end{pmatrix}.$$

Once the vehicle is locked on to the object, a dense depth from camera motion algorithm can then be run. To do this, the system transitions to the `AcquireImage` mode.

(iii) *AcquireImage* (q_3). For the depth estimation algorithm to be accurate, the inter-frame displacement of images acquired by the camera has to be very small. The control objective here is simply to have the vehicle move slowly toward the object so that the algorithm can run. The dynamics are:

$$\mathbf{u}_3 = \begin{bmatrix} K_3 \\ K_{33} \end{bmatrix},$$

$$f_3 = \begin{pmatrix} x[k-1] + K_3 \cos \alpha[k-1]\Delta T \\ y[k-1] + K_3 \sin \alpha[k-1]\Delta T \\ \alpha[k-1] + K_{33}\Delta T \end{pmatrix}$$

where K_3, K_{33} are small translational and angular velocities, respectively. Then the system transitions to either the `ElevGradient` mode or the `Boundary` fail-safe condition depending on whether the boundary of the area to be mapped has been reached.

(iv) *ElevGradient* (q_4). This obtains an elevation gradient of the area between the vehicle start point and where the depth estimation algorithm indicates that an object of interest exists. Let estimated location of object be (x_0, y_0) and the vehicle coordinates at the instant k be $(x[k], y[k])$. Also define the running Euclidean distance between the object and the vehicle as: $\rho[k] = [(x[k]-x_0)^2 + (y[k]-y_0)^2]^{1/2}$ from which the dynamic equations are

$$\mathbf{u}_4 = \begin{bmatrix} K_4\rho[k] \\ K_{44} \end{bmatrix},$$

$$f_4 = \begin{pmatrix} x[k-1] + K_4\rho[k] \cos \alpha[k-1]\Delta T \\ y[k-1] + K_4\rho[k] \sin \alpha[k-1]\Delta T \\ \alpha[k-1] + K_{44}\Delta T \end{pmatrix}.$$

Transitions occur from q_4 depending on whether the `Boundary` is reached or objects/other vehicles need to be avoided.

(v) *Avoid* (q_5). The same control scheme is active when an object or another vehicle has to be avoided. A repulsive vector field $f_{\mathrm{OA}}(\rho)$ (which depends inversely on the square of the distance between the vehicle and the item to be avoided) is generated and the direction of the object is estimated as $\tilde{\alpha}$. The variable $\alpha[k]$ still remains the heading of the vehicle. The dynamic equations in this case are

$$\mathbf{u}_5 = \begin{bmatrix} 0 \\ f_{\mathrm{OA}}(\rho)(\pi + \tilde{\alpha} - \alpha[k-1]) \end{bmatrix},$$

$$f_5 = \begin{pmatrix} x[k-1] \\ y[k-1] \\ \alpha[k-1] + f_{\mathrm{OA}}(\rho)(\pi + \tilde{\alpha} - \alpha[k-1])\Delta T \end{pmatrix}.$$

The system transitions back to the mode from which q_5 was activated once obstacle avoidance is complete.

(vi) *Boundary*. Although this has controls of the same form as the other modes, it is really a fail-safe mechanism which essentially turns the vehicle away from a priori specified boundaries of the region to be mapped. To be consistent, we shall treat this as a sixth mode. The vehicle heading changes by π from the heading α_b at which the boundary was reached. The controls result in system equations given by

$$\mathbf{u}_6 = \begin{bmatrix} 0 \\ K_6(\pi - \alpha_b) \end{bmatrix},$$

$$f_6 = \begin{pmatrix} x[k-1] \\ y[k-1] \\ \alpha[k-1] + K_6(\pi - \alpha_b)\Delta T \end{pmatrix}.$$

Transition typically takes place from here back to q_4.

6.3. Knowledge-based planning and coordination

The control laws are encoded in the local reactive control portion of the HICA agent for the vehicle while the sequencing logic is shown in the automaton of Fig. 10. The descriptions for the switching sets $e_1 \cdots e_6$ and e_a, e_r are indicated in the legend at the bottom (right) of Fig. 10. The local `HICA-PKBP()` determines when these conditions become true from sensor values, the `status` feedback from the controller and prior control sequences. Descriptions of the discrete modes and a list of events are provided in Tables 2 and 3, respectively. The † in Table 3 represents an event corresponding to the avoid mode. This is the default event assumed during uncertain or delayed receipt of coordination/supervisory inputs.

External coordinating inputs to the agent arrive from other agents during potential conflicts. It will be observed later that supervisory events regarding the boundary of the region to be mapped, whether or not particular areas have been explored and when to suspend the task are also possible. The coordination factors are codified as shown in Table 4. More of these factors are filled in as they are codified.

The overall design process is bottom-up: the control primitives are first synthesized, then provably stable switching paths are obtained. The local planning module is then developed to follow these paths. Using this procedure, it is no longer necessary to carry out automated verification of the hybrid system since the logic effectively constrains allowed mode changes to keep the system away from unstable/unsafe mode transitions.

7. Control modes for mapping tasks

The control problem of developing a scheme by which the vehicle team autonomously traverses the terrain in

Table 2
Description of the discrete modes

Q	Label	Description
q_1	*Initialize*	Vehicle turns around in place scanning the environment for objects of interest recording position and orientation (pose) and confidence
q_2	*LockOn*	Vehicle turns to track the pose of the inferred object with highest confidence
q_3	*AcquireImage*	Vehicle advances slowly to the object while depth estimation algorithm runs
q_4	*ElevGradient*	Once depth is estimated, vehicle increases speed and moves toward sensed object while elevation gradient is computed
q_5	*Avoid*	Vehicle turns to avoid obstacles in the environment, other vehicles or the object whose depth range has been estimated
q_6	*Boundary*	Vehicle turns back into the environment once boundary has been reached

Table 3
A list of discrete events

Event	Interpretation
e_1	No object is clearly identifiable within the field of view of the vehicle
e_2	An object has been identified with sufficiently high confidence and its location has been noted
e_3	Vehicle is now locked on to the object location
e_4	Depth estimation algorithm completed and inferred depth is registered with sufficiently high confidence
e_5	An a priori specified boundary of the region to be mapped has been reached
e_6	Vehicle's heading is now away from the boundary
†$\mathbf{e_a}$	**Object or other vehicle close—collision possible**
$\mathbf{e_r}$	Potential collision avoided

† are *avoid* mode events.

Table 4
Codifying the coordination and supervisory inputs for HICA-COORD(C_{in},Status,S)

C_{in}	S	Interpretation	Action
0	0	No conflict	None
1	0	Breached safety radius	Avoid
0	1	Previously mapped	Return to mode q_1
0	2	Boundary reached	Turn around
0	3	Terrain completely mapped	Return to origin
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

order to accomplish the mapping task is solved by associating specific control laws from Section 6 with each of the mapping processes of Section 5.

7.1. Range determination

The range of environmental features are determined from the optical flow between frames during small (known) camera motion. Assume a right handed coordinate system for the camera as shown in Fig. 11. Two simple homogeneous transformations are required to go from the vehicle coordinate frame to the camera coordinates. This transformation from **c**amera to **r**obot can be represented as

$$H_c^r : [X_c \ \ Y_c \ \ Z_c \ \ 1]^{\mathrm{T}} \longrightarrow [X \ \ Y \ \ Z \ \ 1]^{\mathrm{T}},$$

where

$$H_c^r = \begin{bmatrix} R_{z,\frac{\pi}{2}} R_{x,\frac{\pi}{2}} & \mathbf{0} \\ \mathbf{0} & r \end{bmatrix}$$

and r is the position vector of the camera center relative to the vehicle center of gravity while R_x, R_z are rotation matrices about the indicated axes.

In the camera frame, each point in a scene has an associated position vector $P = (X_c, Y_c, Z_c)^{\mathrm{T}}$. This is projected onto $p = (x, y)^{\mathrm{T}}$ in the image plane using the

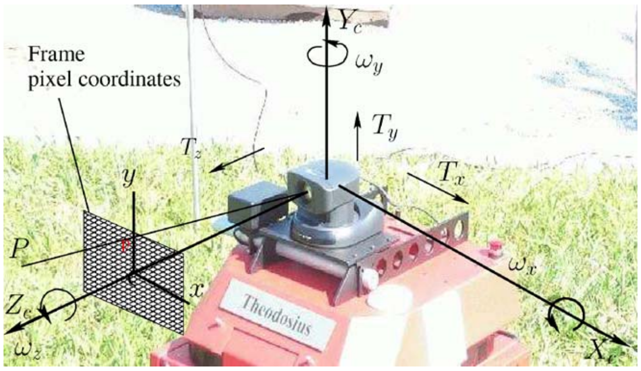

Fig. 11. Camera model and coordinates.

perspective projection

$$x = \frac{fX_c}{Z_c}, \quad y = \frac{fY_c}{Z_c},$$

where f is the focal length of the projection and the point is located at a *depth* Z_c from the camera center. The image coordinate system is assumed centered at a center of projection corresponding to the frame center. On this basis, a simple camera model (Sobel, 1974) is used to obtain the transformation from (x, y) in the image plane to actual pixel row and column. Define $T = (T_x, T_y, T_z)^T$ and $\Omega = (\omega_x, \omega_y, \omega_z)^T$ as the translational and rotational velocities of the point due to camera motion. Then the image velocity $V(x, y)$ is given by

$$V(x, y) = d(x, y)\mathbf{F}(x, y)T + \mathbf{G}(x, y)\Omega, \tag{3}$$

where $d(x, y) = \frac{1}{Z_c}$ is the inverse depth (or *disparity*) and

$$\mathbf{F}(x, y) = \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix},$$

$$\mathbf{G}(x, y) = \begin{bmatrix} \dfrac{xy}{f} & -\dfrac{f + x^2}{f} & y \\ \dfrac{f + y^2}{f} & -\dfrac{xy}{f} & -x \end{bmatrix}.$$

The motion parameters T and Ω in Eq. (3) can be either estimated from the EKF-based localization algorithm or numerically computed from sensor data and post-processed. Accordingly, the only unknowns are the inverse depth $d(x, y)$ and the image velocity $V(x, y)$. For small motions between frames, $V(x, y)$ corresponds to the *optical flow* (Matthies et al., 1989). Thus, computing the optical flow for each pixel in the image frame provides a way to estimate dense depth from camera motion and thus the range of environmental features.

The optical flow between two successive frames (and associated variances) are obtained by a sum of squared difference (SSD) correlation-based algorithm (Fregene et al., 2002). This algorithm runs until the entire frame has been processed. The variance associated with the optical flow is determined by fitting the flow corresponding to the smallest SSD and its two nearest neighbors to a parabola. Raw disparity is computed from Eq. (3) and fused with predicted disparity using a Kalman Filter (see Fregene et al., 2002; Matthies et al., 1989 for a detailed discussion). The next frame is then processed in the same way and so on until the variance indicates that the range has been estimated with sufficiently high accuracy.

The prevailing mode during this stage is `Acquire-Image` q_3.

7.2. Obtaining an elevation gradient

Initially, a coordinate frame is fixed local to the area of interest (but global from the viewpoint of the members of the vehicle team). Pose measurements are therefore referenced to the same coordinate system. The 3D pose of the vehicle at the start location is noted, then the vision-based range determination algorithm is run. The vehicle advances to a location near the most visible object. GPS readings are used to determine vertical displacement of the vehicle between these two locations. This is fused with another vertical displacement measure based on the inclinometer pitch angle θ as follows.

Let L be the offset between the vehicle's center of gravity and the camera's optical center, *NOS* the number of satellites acquired, Z_g the altitude reading from the DGPS and Z_{g0} the DGPS-measured altitude at the base station location. Then, the elevation gradient ΔZ is obtained as

$$\Delta Z = \tfrac{1}{2}[\underbrace{f_\theta L \tan\theta}_{\text{inclinometer}} + \underbrace{f_{\text{gps}}(Z_g - Z_{g0})}_{\text{DGPS}}], \tag{4}$$

where

$$f_{\text{gps}} = \begin{cases} 0.1 & \text{NOS} \leqslant 4, \\ \frac{\text{NOS}}{10} & 4 < \text{NOS} \leqslant 10, \\ 0.9 & \text{NOS} > 10, \end{cases}$$

$$f_\theta = \begin{cases} 0.9 & \text{NOS} \leqslant 4, \\ \left(1 - \frac{\text{NOS}}{10}\right) & 4 < \text{NOS} \leqslant 10, \\ 0.1 & \text{NOS} > 10. \end{cases}$$

We assume at the start that $Z_g = Z_{g0}$ and $\theta \approx 0$ so that the nominal starting value for ΔZ is zero. That is, the terrain is assumed flat until vehicles actually traverse specific portions and update the elevation information. In this formulation, the number of satellites acquired by the GPS is used as a measure of the confidence associated with the vertical displacements provided by the GPS. The vertical displacements are monitored between two positions, say P_1 and P_2 as in Fig. 12 to give an elevation profile for that motion segment corresponding to the depth map obtained. These range and elevation information are registered to a local map

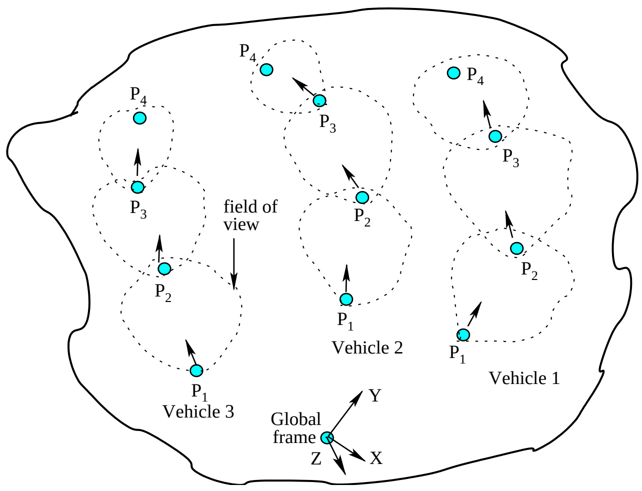

Fig. 12. Obtaining an elevation gradient.

which essentially encapsulates the terrain information within the vehicle's FOV between the two locations. At P_2, the vehicle turns away from objects, their locations are marked, then new range information is obtained. The vehicle advances to a point P_3, then creates a local map connecting P_2 and P_3 containing range and elevation information. This process continues with each vehicle updating the global map as information significantly different from what is already registered on that map becomes available. The map making is incremental in the sense that the terrain is effectively partitioned into segments corresponding to the location of objects of interest.

The prevailing mode here is `ElevGradient` q_4.

7.3. Elevation-range registration

The terrain is represented by a terrain matrix $\mathscr{T}$ whose rows/columns indices (i,j) correspond to X, Y coordinates (in the global frame) and the associated values at these indices contain the elevation gradient at that point. Then the terrain matrix elements are given by

$$\mathscr{T}_{ij} = \begin{cases} \Delta Z|_{XY}, & \text{explored region}, \\ \mathbf{0}, & \text{unexplored region}, \end{cases} \tag{5}$$

where the notation $\Delta Z|_{XY}$ means the elevation gradient for the corresponding X, Y coordinates. Range data from multiple viewpoints correspond to different row/column indices on $\mathscr{T}$. Accordingly, registration simply consists of matching the position coordinates (X, Y) within the immediate FOV of the vehicle with vertical displacements and updating the matrix $\mathscr{T}$ according to Eq. (5). This registration is done after each vehicle traverses the region between two points, (say, P_1 and P_2 in Fig. 12). When necessary, gaps are filled in by cubic interpolation so that when the vehicle arrives at P_2, a 3D map of a subset of the region it just departed is created. It is straightforward to mark object locations by assigning matrix entries with much larger magnitudes than surrounding locations. Bearing information is calculated from the camera's angle of view and the number of pixels per frame.

This process is made simpler because (a) there is a global coordinate frame in which all measurements are expressed; (b) all the information to be fused are available in metric (rather than topological) form; and (c) the manner in which exploration is carried out implicitly takes depth information into account.

Any mode may be active here.

8. Experimental results

Our experiments show results obtained using two vehicles (`Augustus` and `Theodosius`) in an outdoor environment. They move slowly around the environment such that the average inter-frame displacement is no more than 2 cm.

The scenery within the vehicle `Augustus`' FOV during this particular motion segment is shown in Fig. 13a. The prominent object in this scenery is a tree behind which is a building. The filtered depth map recovered for every pixel is shown in Fig. 13b. In this case, the lighter the area, the closer it is to the vehicle. Thus, using the mapping scheme described in Fig. 12, the next motion segment for this vehicle starts on the other side of the tree that is located about 10 m from the vehicle. Its orientation relative to the vehicle is obtained from the lateral pixel displacement relative to the center of the image frame. It should be noted that the sky line and the borders of the building behind the tree are too far away for their depth estimates to be reliable. Covariance plots which indicate how reliable the inferred depth of each pixel is can be found in Fregene et al. (2002). They are not included here due to space constraints. Fig. 13c shows the scenery within the FOV of `Theodosius`. The prominent features here are several culvert covers with a tree branch just overhanging them. The recovered depth map is depicted in Fig. 13d.

The paths which correspond to the motion segment at this instant are shown in Fig. 14. The solid lines correspond to the path of the vehicle `Theodosius` while the dash-dot lines show the path of the vehicle `Augustus` during this segment. The fact that the path data is not smooth is because the GPS position update used in localization is much slower than position encoders internal to the vehicles (see the third column of Table 1 for relevant frequency values). The diamonds in Fig. 14 represent the location of the objects in the terrain.

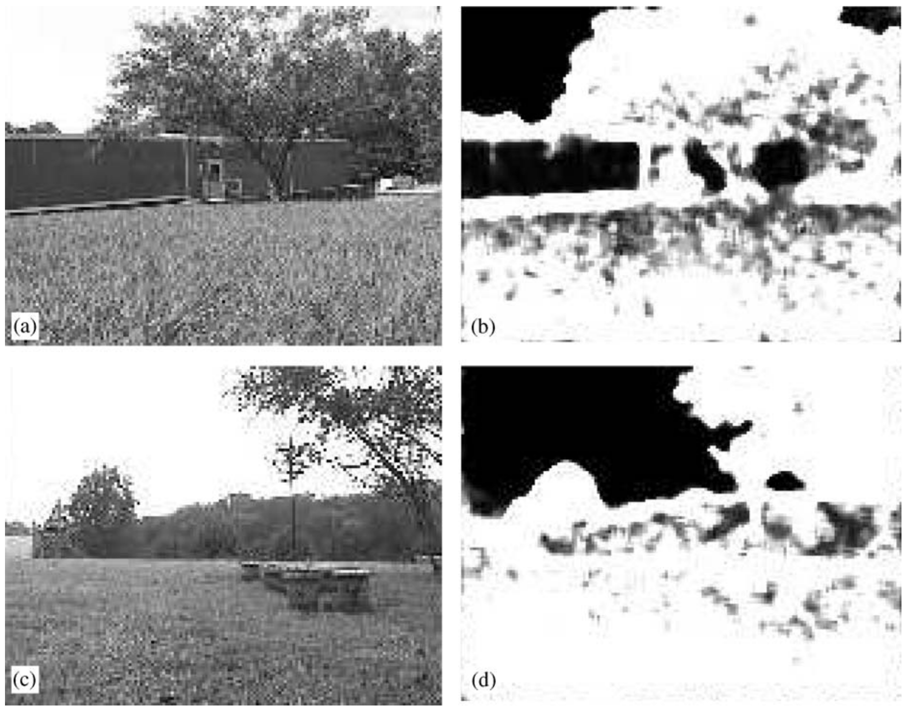

Fig. 13. Experimental results for `Augustus` and `Theodosius`: (a) Augustus: outdoor scene, (b) Augustus: depth map, (c) Theodosius: outdoor scene, (d) Theodosius: depth map.

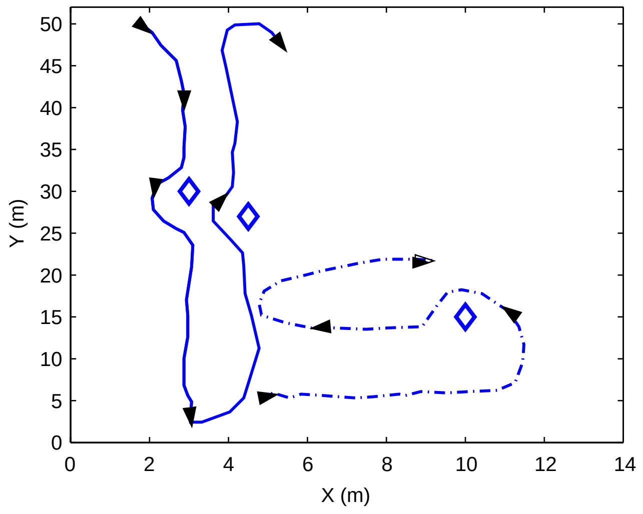

Fig. 14. Vehicles' path during a particular mapping segment.

Both the elevation profile for the motion segments and the feature locations are mapped as shown in the partially updated terrain map (Fig. 15). It shows the elevation profile across the area traversed by each vehicle (in the locally fixed coordinate frame centered at the LADGPS base station location) and prominent features within the vehicle's FOV during the motion segment are marked on the map. This terrain map is essentially a visualization of the terrain matrix $\mathcal{T}$. Areas with prominent features have $\mathcal{T}_{ij}$ entries several orders of magnitude higher than the neighboring displacement entries which serves to flag them as areas to be avoided. Portions of the terrain still unexplored by the vehicles contain no elevation information at the instant shown. Once markers are placed at the approximate locations of the environmental features, exploration then continues on the other side of these markers (this is illustrated in the area explored by the vehicle `Theodosius` in Fig. 15). A map of the entire terrain under consideration is not shown since it would essentially provide similar information (i.e., X, Y co-ordinates with associated elevation gradient) to the results shown for the explored area.

Although the scenery within both vehicles' FOV as shown in Fig. 13a and c appear to be relatively flat (due to the tall grass growing on the terrain), this is not really the case. The terrain is actually fairly undulatory. To show this, the scale of the vertical displacement measure (ΔZ) is slightly exaggerated in Fig. 15.

9. Concluding remarks

A systems- and control-oriented agent framework has been described which wraps an intelligent agent around a hybrid control system core. The framework is amenable to mathematical description and analysis. It can also be used as the basis for developing coordination and control mechanisms for high autonomy systems-of-systems. An example application to outdoor terrain

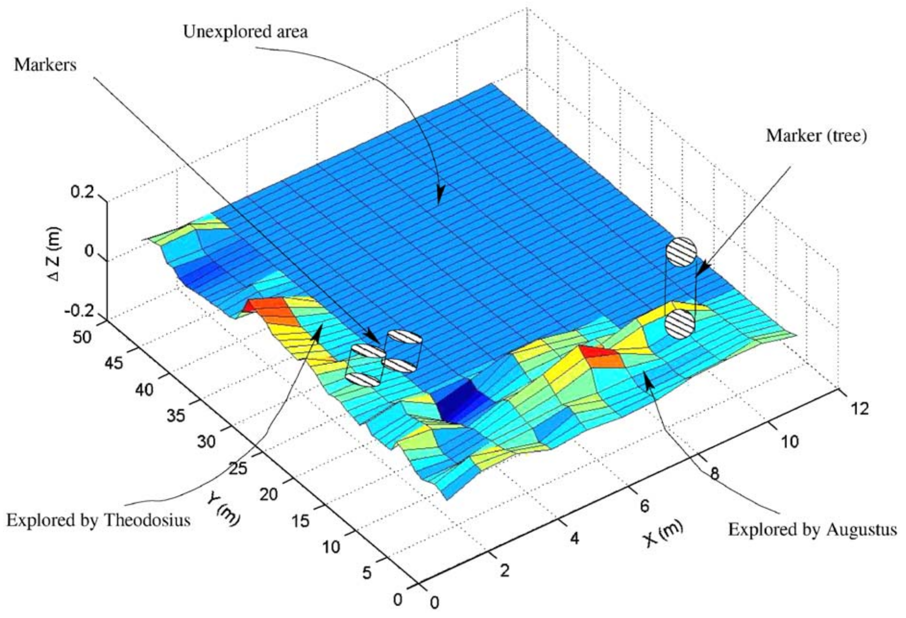

Fig. 15. Partially updated terrain map.

mapping using autonomous vehicles has also been described. In the example, coordinated control logic for autonomous navigation during mapping was synthesized by modelling each vehicle as an HICA agent and the vehicle team as a supervisory multiagent system. Experimental results have also been presented to illustrate the application of this scheme.

There are several ways to extend this work—both on the controller synthesis and the mapping fronts. For the controller, further theoretical studies relating to multiagent stability in the presence of significant communication delays would be particularly helpful. Additionally, the coordination scheme presented presupposes that the designer has significant prior knowledge about the system's expected behaviour. One way to extend this work would be to develop a system in which control and coordination are dynamically synthesized on-line or learned. The map update process described still needs to be improved. Off-line map update, as used in the experiments, would typically suffice for path-planning in static or slowly-changing environments. Schemes to update the maps online in dynamic environments would be a significant extension to this work. Ultimately, we would like to combine the terrain mapping scheme presented here with the multivehicle localization algorithms in Madhavan et al. (2002) so that robust cooperative localization and mapping (in an agent-based framework) becomes a reality.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada via research grants to the second author and by the Engineering Research Program of the Office of Basic Energy Sciences, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battele, LLC. while the first author was visiting Oak Ridge National Laboratory under the HERE@ORNL programme.

References

Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S., 2000. Collaborative multi-robot exploration. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, pp. 476–481.

Dudek, G., Jenkin, M., Milios, E., Wilkes, D., 1993. Robust positioning with a multi-agent robotic system. In: Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Dynamically Interacting Robots, pp. 118–123.

Fregene, K., 2002. Distributed intelligent control of hybrid multiagent systems. Ph.D. Thesis, University of Waterloo, Ontario, Canada.

Fregene, K., Madhavan, R., Parker, L., 2002. Incremental multiagent robotic mapping of outdoor terrains. Proceedings of the IEEE International Conference on Robotics and Automation, 1339–1346.

Fregene, K., Kennedy, D., Wang, D., 2003a. A study of supervisory constraints in a class of coordinated multiagent systems. In: American Control Conference, pp. 1044–1049.

Fregene, K., Kennedy, D., Wang, D., 2003b. Multivehicle pursuit-evasion: an agent-based framework. In: IEEE International Conference on Robotics and Automation, pp. 2707–2713.

Georgeff, M., Rao, A., 1998. Rational software agents from theory to practice. In: Jennings, N., Wooldridge, M. (Eds.), Agent Technology. Springer, Berlin, pp. 139–159.

Grabowski, R., Navarro-Serment, L., Paredis, C., Khosla, P., 2000. Heterogeneous teams of modular robots for mapping and exploration. Autonomous Robots 8 (3), 271–298.

Grewal, M., Weill, L., Andrews, A., 2001. Global Positioning Systems, Inertial Navigation and Integration. Wiley, New York.

Hayzelden, A.L.G., Bigham, J., 1998. Heterogenous multi-agent architecture for ATM virtual path network resource configuration. In: Albayrak, S., Garijo, F. (Eds.), IATA'98. Lecture notes on Artificial Intelligence, vol. 1437. Springer, Berlin, pp. 45–59.

Howard, A., Kitchen, L., 1999. Cooperative localisation and mapping: preliminary report. Tech. Rep. TR1999/24, Department of Computer Science & Software Engineering, The University of Melbourne.

Jennings, N.R., Sycara, K., Wooldridge, M., 1998. A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1, 275–306.

Jennings, C., Murray, D., Little, J., 1999. Cooperative robot localization with vision-based mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 2659–2965.

Koo, T., 2001. Hierarchical system architecture for multi-agent multi-modal systems. In: IEEE Conference Decision and Control, pp. 1509–1514.

Little, J., Jennings, C., Murray, D., 1998. Vision-based mapping with cooperative robots. In: Proceedings of SPIE—Sensor Fusion and Decentralized Control in Robotic Systems, vol. 3523, pp. 2–12.

Lygeros, J., Godbole, D.N., Sastry, S., 1998. Verified hybrid controllers for automated vehicles. IEEE Transactions on Automatic Control AC 43 (4), 522–539.

Madhavan, R., Fregene, K., Parker, L., 2002. Distributed heterogeneous outdoor multi-robot localization. In: IEEE International Conference on Robotics and Automation, pp. 374–381.

Matthies, L., Kanade, T., Szelinski, R., 1989. Kalman Filter-based algorithms for estimating depth from image sequences. International Journal of Computer Vision 3, 209–236.

Nerode, A., Kohn, W., 1993. Multiple agent hybrid control architecture. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (Eds.), Hybrid Systems. Lecture Notes in Computer Science, vol. 736. Springer, New York, pp. 297–316.

Parunak, H., 1999. Industrial and practical applications of distributed artificial intelligence. In: Weiss, G. (Ed.), Multiagent Systems. MIT Press, Cambridge, MA, pp. 377–457.

Passino, K., Antsaklis, P., 1989. A system and theoretic perspective on artificial intelligence planning systems. Applied Artificial Intelligence 3, 1–32.

Rekleitis, I., Dudek, G., Milios, E., 1998. On multi-agent exploration. In: Proceedings of Vision Interface, pp. 455–461.

Russell, S.J., Norvig, P., 1995. Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ.

Šiljak, D.D., 1996. Decentralized control and computations: status and prospects. Annual Review in Control 20, 131–141.

Sobel, I., 1974. On calibrating computer-controlled cameras for perceiving 3-D scenes. Artificial Intelligence 5, 185–198.

Stothert, A., McLeod, I.M., 1997. Distributed intelligent control system for a continuous state plant. IEEE Transactions on Systems, Man and Cybernetics 27 (3), pp. 395–401.

Tomlin, C., Pappas, G., Sastry, S., 1998. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. IEEE Transactions on Automatic Control AC 43 (4), 509–521.

van Breemen, A., 2001. Agent-Based Multi-Controller Systems. Twente University Press.

van Breemen, A., de Vries, T., 2001. Design and implementation of a room thermostat using an agent-based approach. Control Engineering Practice 9 (3), 233–248.

Velasco, J., Gonzalez, J., Magdalena, L., Iglesias, C.A., 1996. Multiagent-based control systems: a hybrid approach to distributed process control. Control Engineering Practice 4, 839–846.

Voos, H., 2000. Intelligent agents for supervision and control: a perspective. In: IEEE International Symposium on Intelligent Control, pp. 339–344.

web, 2002. Multiagent hybrid control in large-scale systems using methods of artificial intelligence. The WWW, http://www.tuke.sk/kkui/ar99/ar9919.htm, current September 26th, 2002.

Weiss, G. (Ed.), 1999. MultiAgent Systems—A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, MA.

Wittig, T., 1992. ARCHON: An Architecture for Multi-Agent Systems. Ellis Norwood, London.

Wooldridge, M., 1999. Intelligent agents. In: Weiss, G. (Ed.), Multi Agents Systems. MIT Press, Cambridge, MA, pp. 27–77.