Auton Robot (2014) 37:47–69
DOI 10.1007/s10514-013-9375-7

Policy learning for autonomous feature tracking

Daniele Magazzeni · Frédéric Py · Maria Fox · Derek Long · Kanna Rajan

Received: 14 December 2012 / Accepted: 14 November 2013 / Published online: 1 December 2013

Abstract We consider the problem of tracing the structure of oceanological features using autonomous underwater vehicles (AUVs). Solving this problem requires the construction of a control strategy that will determine the actions for the AUV based on the current state, as measured by on-board sensors and the historic trajectory (including sensed data) of the AUV. We approach this task by applying plan-based policy-learning, in which a large set of sampled problems are solved using planning and then, from the resulting plans a decision-tree is learned, using an established machine-learning algorithm, which forms the resulting policy. We evaluate our approach in simulation and report on sea trials of a prototype of a learned policy. We indicate some of the lessons learned from this deployed system and further evaluate an extended policy in simulation.

Keywords Policy-based control · Planning-based policy-learning · Autonomous underwater vehicles

D. Magazzeni (✉) · M. Fox · D. Long
Department of Informatics, King's College London, London, UK
e-mail: daniele.magazzeni@kcl.ac.uk

M. Fox
e-mail: maria.fox@kcl.ac.uk

D. Long
e-mail: derek.long@kcl.ac.uk

F. Py · K. Rajan
Monterey Bay Aquarium Research Institute, Monterey, CA, USA
e-mail: fpy@mbari.org

K. Rajan
e-mail: kanna.rajan@mbari.org

1 Introduction

This paper describes the application of plan-based policy-learning to the problem of tracking the boundaries of oceanological features. Examples of features of interest include Harmful Algal Blooms (HABs), anoxic zones and fronts (Das et al. 2010). The toxic nature of HABs and their economic impacts to coastal populations are substantial (Hoagland and Scatasta 2006; Ryan et al. 2005). Understanding the ecological and spatio-temporal dynamics of such phenomenon is therefore an important scientific problem. In this work we have focussed on tracking HAB boundaries using a plan-based policy to cope with the dynamic nature of a bloom. Blooms and HABs by extension are patchy and dynamic; characterizing the boundary is a challenge especially in the context of observing a bloom patch over a period of days or weeks.

Autonomous control of underwater vehicles in challenging missions is an area of increasing interest, with increasingly sophisticated approaches being deployed (McGann et al. 2008a; Zhang et al. 2011). The range and duration of missions is increasing rapidly, from long range powered vehicles such as Autosub (Oceanography 2003) deployed on missions beneath the icecaps to long duration trans-Atlantic glider missions (Glenn et al. 2010). The challenge in these missions is to engage the autonomous underwater vehicles (AUVs) in purposeful activity that coherently contributes to scientific survey goals, avoids risking the vehicles and manages limited resources effectively. One approach to tackling this challenge is to equip AUVs with on-board decision-making capability at a level above the traditional control-level (which remains essential), to allow the vehicle to autonomously engage in a series of planned activities that contribute towards its mission-objectives. An example of an architecture and decision-making framework is

T-REX (McGann et al. 2008a). T-REX deploys actions implemented as *reactors*, which encapsulate basic competences of the AUV, and it organises these actions using a planner (Frank and Jónsson 2003). The planner orchestrates the basic actions and monitors their execution, reacting to the success or failure of actions by adjusting the mission plan appropriately. We describe T-REX in more detail in Sect. 2. Tracing of HAB structures can be seen as one of many mission tasks that might be undertaken by an AUV and, therefore, is one of the actions that can be deployed by the on-board mission planner in T-REX. In this paper we describe how the basic competence is acquired as an encapsulated policy and the way in which this action is then provided as a reactor within the T-REX architecture.

The approach we have taken to construction of the basic HAB-tracing competence is based on sampling a large number of simulated static surface bloom instances, which we call *patches*, solving each of these as a deterministic planning problem, and then learning a general purpose policy by classification methods. We make an important advance over the method presented in Fox et al. (2011), which is to learn improvements to the policy during a second training phase. In the interests of clarity, we emphasise that the planning activity performed on-board within T-REX is significantly different from the planning techniques we employ to solve our simulated patch-tracking instances off-board, during the policy learning process. In the former, the planner is required to interact with a dynamic execution environment, handling uncertainty and making decisions under both time and computation resource constraints, while in the latter the sampled problem instances do not contain uncertainty and the planner is under far less arduous resource constraints.

This paper significantly extends on the work reported in Fox et al. (2012a). In this paper we explain the approach in detail and present extended experimental results that demonstrate the effectiveness of the method. We also describe how the policy we learn is deployed within the T-REX framework and we demonstrate its operation in both sea-trials, on-board the Monterey Bay Aquarium Research Institute (MBARI) *Dorado* Autonomous Underwater Vehicle (AUV) in Monterey Bay, and in simulation. This paper describes the results of our analysis and the plans for future development of the approach and its role in autonomous robotics in the ocean sciences.

The paper is organised as follows. In Sect. 3 we describe our general approach for plan-based policy-learning. In Sect. 2 we describe the T-REX framework, which is used by MBARI in control of some of its autonomous oceanological and oceonographic science gathering missions. In Sect. 4 we show how the plan-based approach can be used to learn a policy for patch following. In Sect. 5 we describe the how the policy-learning algorithm has been integrated into T-REX and then, in Sect. 6, we report on two sea tests, presenting the results and indicating some of the lessons learned. In Sect. 7 we present simulation results based on 3-d blooms. In Sect. 8 related works are discussed and Sect. 9 concludes the paper outlining our plans for future work.

2 The T-REX framework

The role of AUVs in scientific surveys of our oceans has been steadily increasing over the past decade, as power management, instrument technology, control systems and robotics have advanced. The use of the systems has become increasingly ambitious and one of the visions of their future use is that they undertake extended unsupervised science missions, gathering data in a directed and purposeful way, engaging in coherent scientific surveys that are planned on-board, in response to observed oceanological phenomena and mission priorities set by scientists. Realising this vision requires technology capable of organising and planning AUV activity over extended timeframes and this remains a significant challenge to artificial intelligence, autonomous robotics and control.

In this section we briefly describe an open-source plan-execution framework called the Teleo-Reactive EXecutive (T-REX) flown routinely on a *Dorado* AUV shown in Fig. 1, at MBARI. The *Dorado* running T-REX is the only operational marine platform anywhere being used for routine scientific surveys with on-board plan synthesis.

T-REX is an on-board adaptive control system that integrates AI based planning and state estimation in a hybrid executive (McGann et al. 2008c,b; Py et al. 2010). State estimation allows the system to keep track of world evolution as time advances based on a formal model distributed through the architecture. Onboard planning and execution enables adaptation of navigation and instrument control based on estimated state. It further enables goal-directed commanding within the context of projected mission state and allows for replanning for off-nominal situations and opportunistic science events. The framework in addition to being used on

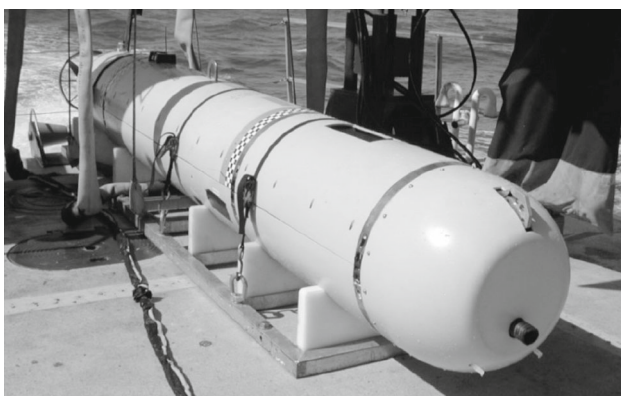


Fig. 1 MBARI's *Dorado* AUV on the R/V *Zephyr*

an AUV, is general enough to be used for controlling a personal robot (Meeussen et al. 2010; McGann et al. 2009) and is deployed on a European planetary rover testbed (Ceballos et al. 2011) and for experimental engineering missions on AUVs at the Univ. of Porto, Portugal (Pinto et al. 2012). Deliberation and reaction are integrated systematically over different temporal and functional scopes within a single agent and a single model that covers the needs of high-level mission management, low-level navigation, instrument control, and detection of unstructured and poorly understood phenomena.

T-REX distributes computation into smaller fragments in systematic functional components called *reactors*. The overall system is a composition of these reactors with each deliberating on a subset of the problem. Generative planning is *distributed* between these components which can interact through state updates and goals for future values of a state variable. Further, planning and execution within this framework is tightly coupled, with consequent broader semantics of execution to imply *dispatching* of goals from one reactor to another. The architecture manages information flow within the partitioned structure to ensure consistency so as to direct the flow of goals and observations in a timely manner. The resulting control structure improves scalability since many details of each controller can be encapsulated within a single control loop. Furthermore, partitioning increases robustness since controller failure can be localized to enable graceful system degradation making this an effective divide-and-conquer approach to the overall robot control problem. The role of the T-REX agent then, is to ensure that all the reactors will be able to interact *concurrently* so that they are informed of state evolution that may impact them, have a sufficient amount of time to synthesize plans and coordinate plan dispatch across reactor boundaries. Figure 2 shows an instantiation of a T-REX agent.

To represent the evolving state information coupled with the notion of time, T-REX uses state variables as *timelines*. This is a flexible representation to describe one sequence of states for a given state variable (Muscettola 1994; Jónsson et al. 2000; McGann et al. 2009). A timeline represents the temporal evolution of a single state variable allowing each reactor to deliberate about the future evolution of each of these states. At any given time each state variable can have only one state value which is described as a predicate. Thus each timeline consists of a sequence of predicates which encapsulate and describe state evolution; we call these instantiated predicates *tokens*. A token therefore describes a predicate, the state variables on which it can occur, the parameter values of the predicate, and the temporal values defining the *interval* during which this predicate holds true. Each reactor then operates within a systematic framework of concurrently evolving timelines updated by the T-REX framework to tracking world state evolution. Each T-REX agent is, in turn, a composition of reactors with specialized functional

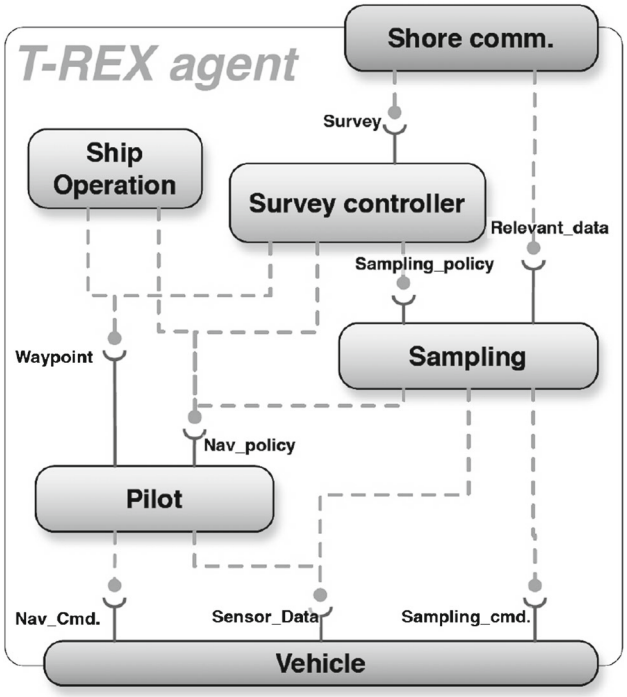


Fig. 2 A T-REX agent is composed of multiple reactors or control loops (*rounded boxes*) which are connected through state variables provided by one reactor (*solid lines* ending with a ⌣) with multiple possible clients (*dashed lines* ending with a •)

roles with timelines that track associated sub-systems appropriate for these roles. Further details of the T-REX can be found elsewhere (McGann et al. 2008c; Py et al. 2010).

3 The patch tracking problem in the ocean sciences

Harmful Algal Blooms develop in the coastal oceans due to oceanographic processes which cause microorganisms to come to the surface and bloom (Glibert et al. 2005). Some of these organisms release trace chemicals that are toxic to plant and marine life leading to large-scale marine mortalities as well as to shell-fish poisoning and events affecting human beings. Ecological understanding of such blooms is poor, with model skill related to their formation and collapse in their life-cycle poorer still. The scientific need to understand and characterize the environment of such blooms is therefore of substantial interest.

The bloom-tracking (or patch-tracking) problem is, for our purposes, the problem of following the 3-D edge-structure of a patch of algae with an Autonomous Underwater Vehicle (AUV) for a fixed amount of time. The objective is to help determine the shape and extent of the surface of the bloom. The AUV makes control decisions (what path to follow in order to remain close to the edge), based on taking regular readings with a chlorophyll sensor. The objective is

to follow a contour within the patch that represents a particular chlorophyll concentration, allowing the surface of the bloom to be tracked and mapped over a series of missions. A chlorophyll threshold defining the contour to be followed is used to determine whether the AUV is inside or outside the contour at each time that a navigation decision is made. To identify the approximate path of the contour requires that the AUV alternately read values above and below the threshold, crossing the contour left and right to achieve this. For reasons that we discuss below, the decision-making frequency is lower than the sampling frequency, so the AUV must make a decision about which way to turn, in order to follow the contour, based on the combination of the most recent measurement and the history of observations collected since the last decision. The path of the AUV should ideally zig–zag over the edge, turning into the bloom when a reading below the threshold is taken, and turning out of the bloom when the most recent reading is above the threshold.

At first sight this might seem similar to a line-following problem of the kind encountered in many mobile robotics applications (Joshi et al. 2009). However, there is a very important difference which makes our problem challenging. The AUV has constrained manoeuverability, so can only make navigation decisions at a coarse granularity relative to its chlorophyll sampling rate. For scientific reasons the AUV follows a yo-yo path in the water with a fixed diving angle and a minimum depth.[1] This means that, even though chlorophyll can be sampled at a high frequency, waypoints are encountered some metres apart so that the AUV might travel a considerable distance before it becomes apparent that it has lost the edge of the patch.

Unlike in robot line-following Martin (1996), where typical control strategies are constructed as hand-built policies, in the patch following problem there is no obvious solution strategy to hand-code as a policy. Line-following robots typically have both a high frequency sampling rate and high degree of manoeuverability compared with the curvature of the line they are following. This makes it possible to react to a sensor detecting the transition from one side of the edge to the other, in a distance that is negligible compared with either the curvature of the line or the ability of the robot to turn back towards the line. The line is usually well-defined and a single reading is typically sufficient for a line-following robot to determine whether it is on the line or not. In contrast, in the patch-following problem the contour is poorly defined, with potentially much greater curvature than the AUV can achieve in manoeuvres.

[1] The vehicle can only communicate and acquire GPS coordinates when on the surface, but the blooms are usually concentrated below the surface. Furthermore, changing bouyancy has low energy cost and allows the vehicle to achieve lateral velocity by gliding through the water as it rises and falls. Finally, a yo-yo pattern allows the vehicle to track the structure of the bloom in three dimensions.

When a series of successive chlorophyll readings remain above or below the threshold, indicating that the contour has been lost, the AUV has to take corrective action to get back to the edge. This is the crux of the problem that we address, since knowing in which direction to turn to relocate the contour is the part of the problem complex enough to benefit from planning. In contrast with line-following, which is purely reactive, planning helps to equip the AUV with a reasoned basis for its navigation choices when there is insufficient data for the system to make purely data-driven decisions.

Saigol et al. (2009) explore the problem of finding ocean-bed vents using AUVs, by learning a policy for plume tracking. Although the problem is rather different to ours, the application of learning to policy construction for control of AUVs shares some similarities.

3.1 Policy learning for patch tracking

Experience with tracking similar shaped patches in the past can assist an AUV in deciding how to respond in its current situation, including how best to react to losing the contour. The first step is to solve a large number of sampled patch-tracking problems as planning problems, setting as the goal that the AUV circumnavigate the patch as efficiently as possible. The next step is to learn a policy from the experience gained. By learning a policy we provide the AUV with advice about which way to turn based on prior experience of having traversed a large number of surface patches in the past. The AUV will turn left (or right) if it has normally turned left (or right) in similar situations in the trajectory around previously seen blooms. As our experiments show, this learned policy behaviour often guides the AUV successfully back onto the edge of a patch when it has temporarily lost track of the appropriate contour. It turns out that learning this response from previous experience on a large number of similar patches leads to good behaviour at run-time, even on patches that are different in shape to those seen in training. Our results show a high level of robustness to the shape and dimensions of surface patches.

The approach we use for constructing policies in this work is one that we have developed over several earlier works (Fox et al. 2011, 2012b,a), and is summarised in Fig. 3. In the following we describe the technique in detail.

3.2 Plan-based policy learning

Partially observable environments can be described through Partially Observable Markov Decision Processes (POMDPs), which generalise standard MDPs with the assumption that the states are not completely observable.

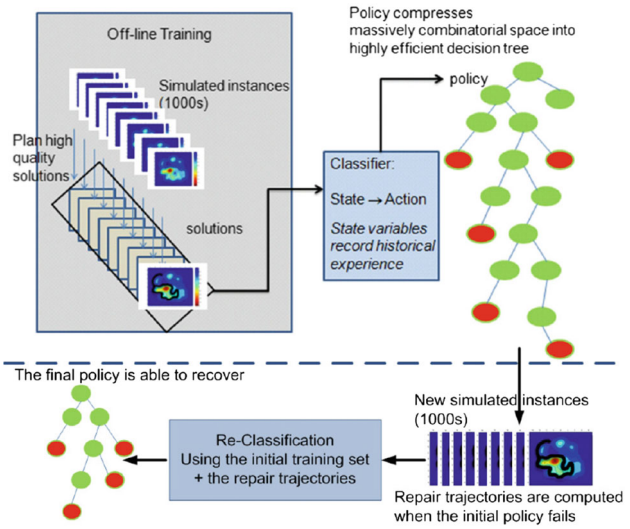


Fig. 3 The plan-based policy-learning method

Formally, a POMDP is a tuple (S, A, P, R, O, Q), where:

- S is a set of states.
- A is a set of actions.
- P is a probability function, where $P_a(s, s') = \Pr(s'|s, a)$ is the probability that action $a \in A$ will cause a transition from state $s \in S$ to $s' \in S$.
- $R : S \times A \to \mathbb{R}$ is a reward function, where $R(s, a)$ is the reward earned by executing action a in state s.
- O is a set of observations.
- Q is a probability function, where $Q_a(o|s) = \Pr(o|a, s')$ is the probability of observing o if action a is executed and the resulting state is s'.

In particular, we consider the subset of POMDPs in which there is a probability distribution over initial states (which, in turn, provides an implicit representation for functions P and Q), but the action transition and observation functions are both deterministic. This means that all of the uncertainty lies in the initial state and the executive can attempt to narrow the set of candidate initial states by executing actions and observing their effects. An important consequence of limiting attention to this subset of POMDPs is that, were it possible to observe the whole of the initial state, the problem would be entirely deterministic. A further constraint we place is that the reward function for the problem (R) yields (the same) reward at a set of *goal* states and no reward at any other states. We treat the goal states as terminal states. To complete the mapping between the formal definition of a POMDP and the patch tracking domain, we define a state $s \in S$ as the position and the status footnoteDetails on how the status of the AUV is described are provided in Sect. 4.2. of the AUV, an observation $o \in O$ as the position of the patch (i.e., whether the AUV is currently inside or outside the patch) and the set of actions A as the possible moves for the AUV (i.e., `left`, `forward-left`, `forward`, `forward-right`, `right`).

Given a POMDP of the sort we are considering, we perform the following sequence:

1. Use Monte Carlo sampling to construct a set of N initial states, $\mathcal{I}$, using the initial state distribution. We refer to these states as *samples*.
2. Construct a plan, π_I, for each sampled initial state, I. In the present work, plans are simply sequences of actions, although in other applications of the idea they can be more complex (Fox et al. 2011, 2012b). For the patch tracking domain considered in this work, plans are generated using the heuristic forward search planner LPRPG (Coles et al. 2013).[2]
3. For each plan, π_I, simulate execution of π_I starting in state I. This is straightforward, since the POMDP is deterministic once the initial state is specified. In particular, each step of the execution of π_I is defined by the observation of the current state ($\omega_I(i)$) and the plan action executed immediately following the observation. Therefore, each execution gives rise to a sequence of observation-actions pairs $\langle \omega_I(i), a_i \rangle$ where $\omega_I(i) \in O$ and $a_i \in A$.
4. Collect the set of all the observation–action pairs, $\mathcal{OP}$, constructed in the preceding step and apply a *feature generator*, f, to the observation in each pair, to obtain: $\{\langle f(\omega), a \rangle | \langle \omega, a \rangle \in \mathcal{OP}\}$.
5. Apply a decision tree classifier learning algorithm to this set, to learn the action decision for the corresponding observed features. In particular, each observation is mapped into one of the classes represented by the possible moves of the AUV. This decision tree classifier defines the policy, which is used in conjunction with the same feature generator function to construct the inputs to the classifier from observations made during execution. The classifier is the `J48` decision tree method provided by the WEKA toolkit (Hall et al. 2009).

In general, we assume that the states of our POMDP include sufficient history to allow observations to track some of the recent activity of the executive, together with the accumulated information about the world state. The role of the feature generator is to extract or compute appropriate features from the observations to support the classifier learning. For example, in the patch following problem we can record the series of recent manoeuvres of the AUV and compute a count of the number of each type executed since the start of the plan. Feature extraction is an important (and difficult) part of the task of harnessing good performance from machine learning

[2] More details on plan generation are provided in Sect. 4.2.

tools (Guyon and Elisseeff 2003). We discuss the selection of features in Sect. 4.3.1. A benefit of our approach is that the states that define the problem, and that are used by the solver in step 2, can be distinct from the observed features used in step 5. The classifier can find and exploit any correlations between the observed features of the states and the actions that are selected. This can be exploited effectively to allow relaxation of the constraint that the observation function must be deterministic: the solver we use in step 2 can ignore the observation process and simply operate directly on the fully observed states, using a deterministic action transition function. In the simulation of the execution of these plans in step 3 we can generate observations non-deterministically from the states visited in these execution traces and use those to construct the data set for classification in step 4. A useful consequence of this approach is that the distribution governing the association between states and observations can be left implicit in the simulation system rather than being constructed explicitly as part of a formal description of the POMDP.

The use of a planner in the second step, to solve the sampled problem instances, can require that the goals be constructed from the specific initial state. For example, in the patch-following problem the intention is to visit as much of the edge of the patch as possible. In practice, this goal is difficult to specify to planners, so the goal is, instead, specified as a destination for the planner to reach (on the edge of the patch) with constraints that prevent the planner from deviating far from the edge on its path. We discuss this in more detail in Sect. 4.

The most important limitation of the approach we have described is that the classifier can only be expected to contain sensible action responses to (observed) states that are very similar to states seen on the plans generated for the samples. Thus, if the initial state encountered during an actual execution of the policy lies significantly outside the range of previously observed states then the classifier will generate spurious responses (or no response, depending on whether the classifier is able to determine the range of its own application). We tackle this problem by adding a set of *default actions* to the policy. An initial test is performed on the observed features of the current state to determine whether it falls outside the deemed range of applicability of the learned policy. If so, then a default action is selected, based on the value of the observed features.

In general, constructing a basic policy for a domain is non-trivial, but it is important to realise that the basic policy need not necessarily provide a solution to the original problem; it need only guide the execution back into the envelope of application of the classifier. Identifying the range of application of a classifier depends on characterising the initial sample set in some way that can be quickly tested when confronted with a new observed feature set. In practice, defining such a test is domain dependent and we describe the test we use for patch-following in Sect. 4.

As a further stage in the construction of the policy we can perform an additional step. We generate a further set of samples, as in step 1, and apply the policy constructed in step 5 to the collection, using the simulator to construct the sequence of visited states. If the policy visits states that lie outside its applicability (as described above), we apply the corresponding default actions. This leads to a new set of pairs (constructed as in steps 3 and 4 above) and a new classifier can be learned from the combined data sets generated in this stage and in the first stage. This process enhances the robustness of the policy and extends its range of applicability. The process can be iterated, although we do not do so in the application discussed in this paper.

The approach most closely related to ours is *Hindsight Optimisation* Planning (Chang et al. 2000; Fern et al. 2006). HOP is based on sampling a large number of deterministic instances of an MDP with initial state $\mathcal{I}$, then solving these instances using a deterministic planner over a fixed horizon. Finally, the estimated value for the state s is computed as the average value obtained from the deterministic plans. A key difference between HOP and the plan-based approach (PBL) proposed in this paper is that PBL is based on determining state values and selecting actions based on these, while PBL uses classification over a set of examples to decide which action to use in a state. Furthermore, PBL only undertakes one phase of planning (or two phases if, as in this paper, a second stage is used to supplement the learning examples based on executions of the first learned policy). All the planning is done in one or two phases, while HOP performs planning at each reachable node. This gives PBL a potentially important computational advantage over HOP.

4 Applying plan-based policy learning to patch following

In this section we discuss the details of application of the approach described in the previous section to the patch following problem.[3]

4.1 The relevance of planning

The edge of a patch cannot be seen by the AUV—the AUV can only infer where the edge lies by seeing consecutive readings that lie on opposite sides of the patch edge. This means that there is inherent uncertainty in the problem of following the edge, which is why the problem can be usefully seen

[3] Several figures presented throughout the rest of the paper are presented in colour and are difficult to interpret in monochrome. The reader is recommended to view the figures using an appropriate medium.

as a POMDP. However, the state space of the POMDP is too large to allow it to be solved by direct means, even using modern factoring techniques and other speed-ups. Therefore, we approach the uncertainty using sampling and policy learning.

In a sampled instance of the patch problem the position and edge of the patch are precisely known, so it might appear that following it is best solved by simply constructing a path that zig–zags optimally across the patch edge. Although this would yield many examples of good actions to perform when the AUV had just crossed the patch edge, it would provide no examples of good actions when the AUV has not just crossed the edge. In practice, the AUV will find itself in the latter situation much more often than the former, precisely because of the uncertainty in the real problem.

To increase the coverage of the examples to include situations in which the AUV has left the patch edge, we must generate paths that do not simply zig–zag optimally across the edge. In addition, we want the paths to exploit state variables that can actually be measured during execution (i.e. do not rely on knowledge of unseen parts of the patch edge). We achieve this by introducing a variable we call the *confusion level*, which increases when the AUV moves without crossing the edge and is reset when it crosses the edge. The problem is then modified to require a path that is as short as possible, but which keeps the confusion level within a small bound.

The solutions to this modified problem create examples in which the AUV is at varying distances away from the patch edge (up to half the threshold bound value, since the AUV must return to the edge before its confusion level exceeds the bound). These examples provide a representative range of states from which to learn a policy, because they correspond to states the AUV is likely to visit in practice and they depend on the value of the confusion level whose value can be monitored by the AUV during execution.

Therefore, we must solve a problem that is not simple edge following, but that of finding a short path around the patch that keeps the confusion level within a given bound. This problem can be solved efficiently by a planner. The problem involves managing numbers (the confusion levels), so we chose to use LPRPG (Coles et al. 2013). LPRPG is one of the currently highest-performing planners capable of reasoning with numeric conditions and effects. However, any numeric planner could be substituted for LPRPG (for example, metricFF Hoffmann 2003, LPG-td Gerevini et al. 2004, Fast Downward Helmert 2006, to name some of the most easily available such planners).

4.2 Patch tracking as a planning problem

In this section we briefly describe how the deterministic version of this problem can be modelled as a planning problem in PDDL (McDermott et al. 1998), and how a planner can be used to solve it. Sections 4.2–4.4 reprise the descriptions given in Fox et al. (2012a) for ease of reference.

In order to define a domain for the patch tracking scenario, we model the bounding box enclosing the patch as a grid of $k \times k$ cells, where each cell is typed as inside or outside the patch. The state of the AUV is described through its position in the grid, its facing direction and its *confusion level* which measures how far the AUV has travelled since last crossing the edge. Namely, the confusion level C at time t is recursively defined as:

$$C(t) = \begin{cases} 0, & \text{if } O(t) \neq O(t-1) \\ C(t-1)+1, & \text{otherwise} \end{cases}$$

where $O(t)$ is the result of the chlorophyll observation at time t (ie whether the reading is above the threshold for the patch edge or not).

The action the AUV can perform (as shown in Fig. 4) is to move between two connected cells through the following directions, which reflect constraints on manoeuverability of the AUV: left, forward-left, forward, forward-right, right. The action model is interpreted as follows: the parameters are listed with a question mark denoting the variable tokens and their types following the hyphens, then the precondition is a conjunction of conditions that must be satisfied in the current state in order for the action to be applied, while the effect updates the state (the confusion level, the location and the facing of the vehicle). The predicate `colourOf` is used to indicate whether the particular cell is inside or outside the patch (we use black cells inside the patch and white outside). The action model shown here is used when the vehicle is navigating between two cells of equal colour (both inside or both outside the patch), with the consequence that the confusion level increases. The cells all begin unvisited and

```
(:action move
 :parameters
  (?v - vehicle ?from ?to - location
   ?dold ?dint ?dnew - direction)
 :precondition
  (and (< (confusion ?v) 4) (new ?to)
       (colourOf ?from ?c1) (colourOf ?to ?c2)
       (eq ?c1 ?c2) (at ?v ?from)
       (facing ?v ?dold)
       (linked ?from ?to ?dint)
       (validDirection ?dold ?dint ?dnew))

 :effect
  (and (increase (confusion ?v) 1)
       (at ?v ?to) (not (at ?v ?from))
       (facing ?v ?dnew)
       (not (facing ?v ?dold)) (not (new ?to)))
)
```

Fig. 4 PDDL action for patch tracking

visits are recorded (by deleting the `new` predicate for a visited cell): the plan is constrained to never revsit cells (because of the precondition requiring the target cell to be `new`).

The patch to be tracked is described as a PDDL problem by giving the set of all the connections between adjacent cells, together with the type of each cell.

Since the patch is known, the planner can plan to keep the confusion level below a given threshold by choosing to cross the edge as often as necessary while keeping the plan as short as possible. To solve the problems, we use the heuristic forward search planner LPRPG (Coles et al. 2008). The patches to be tracked cover an area of several square kilometers, represented by grids of 180×180 cells. Note that this results in a huge state space, whose cardinality is in the order of magnitude of 2^K, where K is the number of cells in the grid (in this case $K = 32400$).

To cope with this state explosion, we use a problem decomposition method, dividing the patch into nine squares, then generating partial solutions which are finally linked to each other to obtain the complete path. The choice of nine was determined empirically, by trying to identify the smallest subdivision that would yield practically solvable problem sizes for our planner. This decomposition into nine sub-grids is applied without consideration of the structure of the patch, taking into account only the fixed size of the bounding box. If the bounding box size varied, and patches contained concentrations of interesting points to be explored, distributed in some non-uniform way over the patch, then a variable discretisation might be worth considering. However, in our case, nine equal sub-divisions serves to address the state explosion. Our experiments show that our learning strategy is robust to variability in patch structure. Therefore, we have not further considered the variable discretisation approach.

4.3 Policy learning

Having shown how to generate plans for a known patch, we now focus on how to deal with unknown patches. In general, it is not possible to predict the exact shape of the surface patch of a bloom. However, remote sensing imagery might be available to be used to approximate its extent in order to define the bounding box enclosing the patch, or else a bounding box might be defined using approximate surveys with surface vessels. Moreover, we have a probabilistic model, implemented as a patch generator, which creates plausible chlorophyll distributions and one that can be used to generate a representative set of cases.[4] An equivalent reimplementation of the code is included in "Appendix 10". While the physical patches can have sections of the edge that are diffused, the sampled patches never do because the sensor will always determine a chlorophyll reading to be either above or below the threshold.

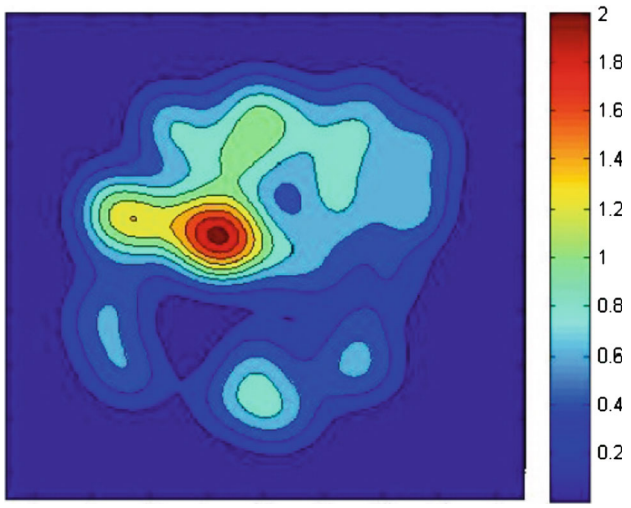


Fig. 5 Example of chlorophyll distribution in a patch—the contours show increasing chlorophyll intensity towards the centre of the patch, but no 'holes' where lower intensity chlorophyll lies inside the patch

An example of a patch is shown in Fig. 5, where different contours are highlighted according to chlorophyll thresholds. In general, HABs can take complex and distorted shapes, influenced by current, wind, nutrient concentrations and many other variables (many of which are unknown or unobservable). Thus, the patch generator cannot be seen as a definitive model of the distribution of initial states we confront. Since the policy is learned on a constrained set of patches, we are not licensed to infer that it can be used effectively for a wider range of patches. However, we can determine, empirically, that it does, in fact, perform robustly across a wide range of different patch structures (see Sect. 3).

In order to learn a policy, the patch generator is used to create a set of deterministic problems which can be solved using the approach described above. For our training set, we considered a chlorophyll level such that all of the patches generated were roughly square or circular, approximately filling the bounding box region and having no internal structure (similar to the one shown in Fig. 5 corresponding to a chlorophyll level of 0.2). We refer to this kind of patch as *simple*. The generated plans can then be aggregated through classification to obtain a policy.

4.3.1 Policy state

As explained in Sect. 3.2, the key to an efficient learning process using our approach lies in defining a suitable policy state. In fact, an informative set of state variables is essential to enable the classifier to structure the decision tree so that particular actions are preferred over others in particular states.

[4] We gratefully acknowledge Mike Godin for his work on this generator.

The states used as the basis for classification cannot be the same states as are used for planning, since only observable state values can be used to apply the policy in execution. Identifying appropriate state variables is the hard problem of feature selection encountered in all machine learning tasks (Liu and Yu 2005). In our problem, the immediately observable features are the position and heading of the AUV and the current chlorophyll level (which determines whether the AUV is inside or outside the patch). These features alone are insufficient for policy learning, because the grid size is so large that it would require a huge number of training examples to ensure that at least one example appeared in which the AUV was in each combination of position and orientation and chlorophyll reading.

Instead of using the exact position, we abstract the position by tracking how many moves of each type have been executed since the start (which offers the policy access to an abstract representation of how far around the patch the AUV has come) and the average bearing over the last ten moves, which gives an indication of the approximate alignment of the most recently seen part of the patch. The decision to use ten moves was reached after some experimentation, considering values between five and twenty. We have already identified confidence level as a state variable that can be tracked and which acts as a proxy for the growing uncertainty about the position of the edge as the AUV fails to cross it during successive actions. The orientation of the vehicle (abstracted to the principle compass directions) is an accessible state variable that we considered likely to be useful in deciding how to act. A feature we considered, but turned out to be unhelpful, was a record of the last n moves for some fixed n (we tried values between five and ten).

After experimentation, we therefore adopted the following state as the basis for classification:

$$s = (\theta, L_c, R_c, F_c, FL_c, FR_c, P, F, C)$$

where:

- $\theta \in [0, 360)$ is the average bearing over the last 10 moves;
- $L_c, R_c, F_c, FL_c, FR_c \in \mathbf{N}$ count the number of times each of the five actions (Left, Right, Forward, Forward-Left, Forward-Right, respectively), has been performed in the plan so far;
- $P \in \mathbf{B}$ is true if the last visited cell was within the patch, false otherwise;
- $F \in [N, S, E, W]$ denotes the current facing direction;
- $C \in \mathbf{N}$ is the confusion level.

The values of these state variables are constructed by tracing through execution of plans in building up the training data, associating action choices with corresponding state values. The policy maps values of these states to actions and, during execution, will give rise to a sequence of transitions of the following form (assuming, for example, that the action $a =$ *Go Right* is performed):

$$s(t) = (\theta, L_c, R_c, F_c, FL_c, FR_c, P, F, C) \xrightarrow{a}$$
$$s(t') = (\theta', L_c, R_c + 1, F_c, FL_c, FR_c, P', F', C')$$

where θ' and F' are the new average bearing and the new facing direction respectively, P' is true if the last visited cell is within the patch, otherwise is false, and $C' = C + 1$ if $P' = P$, else $C' = 0$.

4.3.2 Classification

Following Fox et al. (2011), classification is performed using WEKA (Hall et al. 2009). In particular, we used the `J48` classifier, which implements the machine learning algorithm C4.5 (Quinlan 1993). The output is a decision tree whose leaves represent, in this context, the next AUV action.

We found that the best results are obtained by classifying 2,000 plans, as further extending the training set does not make any significant improvement to the policy performance, but increases memory and time requirements. As each plan consists, on average, of 500 actions, the classification process involves considering about 10^7 values.

We converted the resulting decision tree into a C program, in the form of a set of nested if-then-else statements checking the values of the state variables and returning the corresponding action. The compiled policy is about 2 MB in size and takes negligible time to propose each action. A fragment of a learned decision tree is shown in Fig. 6, to indicate the form of the conditions that are learned.

4.3.3 Recovery action

It is not possible for the policy to cover the whole space of reachable states since it is too large. Therefore the policy must be supplemented with a sensible default action to be used when the AUV gets lost at execution time. This happens when the input state is too distant from any of the states

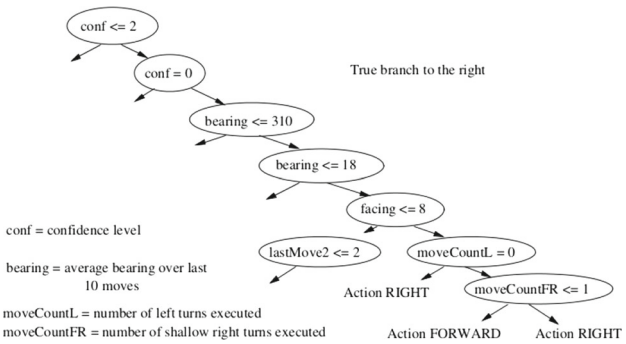


Fig. 6 Fragment of learned decision tree: the leaves of the tree (i.e. the labels) are the actions

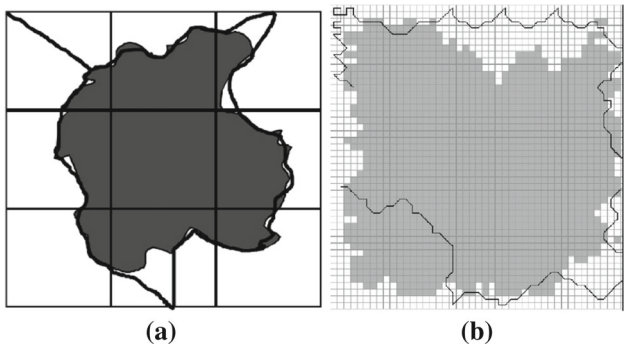


Fig. 7 Ideal (**a**) and actual (**b**) behaviour of the *bouncing box* repair action

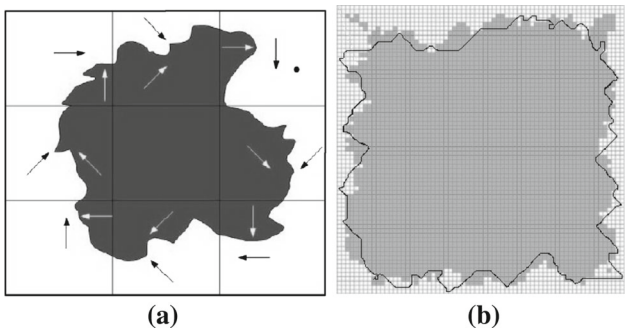


Fig. 8 The area-based default action (**a**) and its behaviour (**b**)

encountered during learning. In our setting, a default action is required when the AUV confusion level exceeds the critical level used in planning. Just as the selection of the variables for the policy state is key to performance of the policy, so the choice of the recovery action is key for determining the robustness of the policy.

Although a simple approach, in which the AUV is simply constrained to remain within the bounding box by 'bouncing' it off the edges, might seem a plausible option, it is a poor strategy (as can be seen in Fig. 7). When selecting a default action it is important to observe that in order to return to a state in which the policy can resume execution, it is necessary to ensure that the default action maintains a policy state that will be consistent with the observed states seen during learning. In particular, since our policy state records an abstracted history of the trajectory, the history of the trajectory when partly derived through the use of default actions must be similar to the trajectories generated under policy control. The bouncing behaviour fails to achieve this, since it leads to long detours towards the edge of the box that distort the numbers of actions executed along particular paths without reflecting an underlying patch structure.

We therefore designed a default action based on the area of the bounding box currently occupied by the AUV: we divided the bounding box into 9 areas and keyed a particular action to each area, according to whether the AUV is currently inside or outside the patch, designed to take the AUV towards the centre of the box when outside the patch and towards the edge of the box when inside the patch, while progressing around the area. The directions keyed to the areas are shown in Fig. 8 (left). The repair action moves the AUV in a straight path following the indicated bearing.

This solution performs well: an example of its behaviour is shown in Fig. 8 (right). In this test, repair was triggered when the confusion level increased beyond a threshold of 5. The area-based default action results in a state that is consistent with the policy, which is still able to correctly handle the AUV states obtained after default actions have been applied.

Finally, we improve the policy using a second classification where ⟨`state,default-action`⟩ pairs are added to the initial training set, as described in Fox et al. (2012a).

4.4 Evaluation

In order to evaluate our approach, we compared the plan-based policy with a static hand-coded policy, based on the strategy of simply using the default actions described in Sect. 4.3.3, and equipped with a loop-avoidance routine. The static policy performed very well on the patches used in our training set, thus it proved to be a good competitor.

The shape of the patches can vary significantly due to effects of ocean currents, wind and distribution of nutrients. Therefore, a key requirement for the policy is its robustness to these variations. To compare the robustness of the static and plan-based policy, together with the simple patches (the ones we used for the training set), we considered three other kinds of patch, described in Fox et al. (2012a) as *horizontal*, *thin* and *inner-structured* patches.

Horizontal patches are flatter than they are wide, representing just a segment of the kind of patch used in training. Thin patches are very flat and wide, representing thinner such segments. Inner-structured patches are more interesting: they are broken up into several loosely connected or disconnected regions, which are close enough together to be considered part of the same patch.

Distortion of the shapes of HABs can be a consequence of wind and current actions, which can elongate the shape of a patch and, over time, cause a break up of the surface structure.

We considered two measures in determining how successfully policies perform, letting $\mathcal{E}$ be the set of points along the edge of the patch and $\mathcal{T}$ be the points along the AUV trajectory:

Table 1 Policy performance and robustness evaluation

Bloom shape	Static policy				Plan-based policy			
	Ψ	Ω	Confusion	# Actions	Ψ	Ω	Confusion	# Actions
Simple	39.37	2.56	2.52	510.94	39.48	3.28	3.59	533.10
Horizontal	1,741.18	51.72	19.93	328.77	187.83	15.08	14.67	387.23
Thin	56.95	45.14	45.14	314.39	25.09	35.80	9.71	353.60
Inner	1,009.51	21.40	16.23	358.45	585.66	20.07	15.56	382.94

– *Minimal distance of the patch edge from the trajectory.*

$$\Psi = \frac{\sum_{\forall e \in \mathcal{E} \cdot t^* \in \mathcal{T} \cdot \nexists t' \in \mathcal{T} \cdot ||e-t'|| < ||e-t^*||} (||e - t^*||^2)}{|\mathcal{E}|}$$

Ψ measures how closely the AUV tracks the patch-edge. High Ψ value will arise when large parts of the contour are missed by the trajectory. In this case, all of the trajectory might follow the contour very closely, but cover only a small fraction of it.

– *Minimal distance of the trajectory from the patch.*

$$\Omega = \frac{\sum_{\forall t \in \mathcal{T} \cdot e^* \in \mathcal{E} \cdot \nexists e' \in \mathcal{E} \cdot ||t-e'|| < ||t-e^*||} (||t - e^*||^2)}{|\mathcal{T}|}$$

Ω measures how efficiently the AUV tracks the patch. High Ω values will arise when part of the AUV trajectory is distant from the patch edge. In this case, all of the patch edge might be visited, but there might additionally be unnecessary detours away from the patch edge.

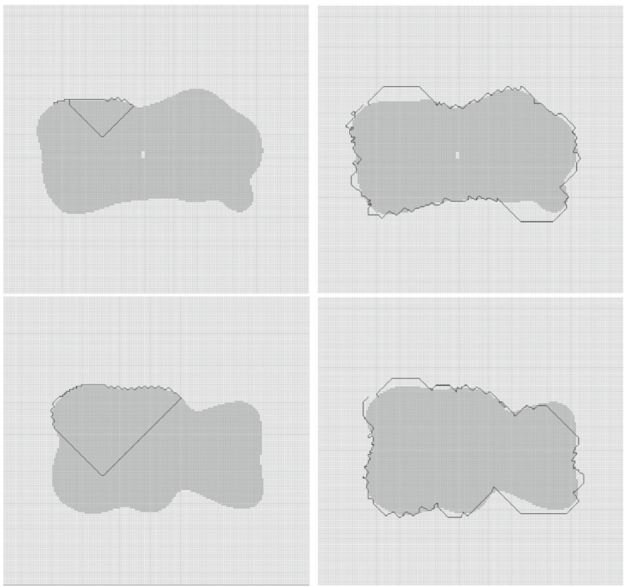


Fig. 9 *Horizontal* blooms tracked using static (*left*) and plan-based (*right*) policy

We also measured the average *confusion level* of the AUV, reflecting the global performance of the AUV in tracking the patch edge.

A detailed description of the results, together with examples of the different patches used in the evaluation, can be found in Fox et al. (2012a). A summary of the results is shown in Table 1, while Fig. 9 shows an example of the behaviour of the policies when dealing with *horizontal* patches. The static policy starts zig–zagging over the edge of the patch, but then gets lost inside it. In contrast, the plan-based policy successfully follows the edge of the patch. The fact that the static policy does not guide the AUV around the horizontal blooms, while the plan-based one does, is confirmed more generally by the big difference ($\simeq$90 %) in the Ψ values in Table 1.

Note that, for all the measures, the lower the value (which refers to the average value over 10,000 tests for each kind of patch), the better. As can be seen, the plan-based policy significantly outperforms the static policy, demonstrating a high degree of robustness. The two policies perform quite similarly when dealing with simple patches. This is not surprising, since the simple patches are convex shapes, which is straightforward for the hand-build policy to follow. The challenge in patch following is when the shape is not convex, since this leads to the need for the AUV to travel in a direction that is essentially opposite to the usual direction in that part of the bounding box.

In contrast to the behaviour on standard patches, horizontal, thin, and inner-structured blooms are far more challenging as their shapes are quite irregular, requiring unusual (or unusually sustained) direction of travel according to the part of the bounding box being traversed. In these cases the plan-based policy performs well, reducing Ψ up to one order of magnitude compared with the static policy. This robustness indicates that the policy is more sophisticated than the simple hand-built policy.

We also tested the policy on more distorted shapes, as shown in Fig. 10.

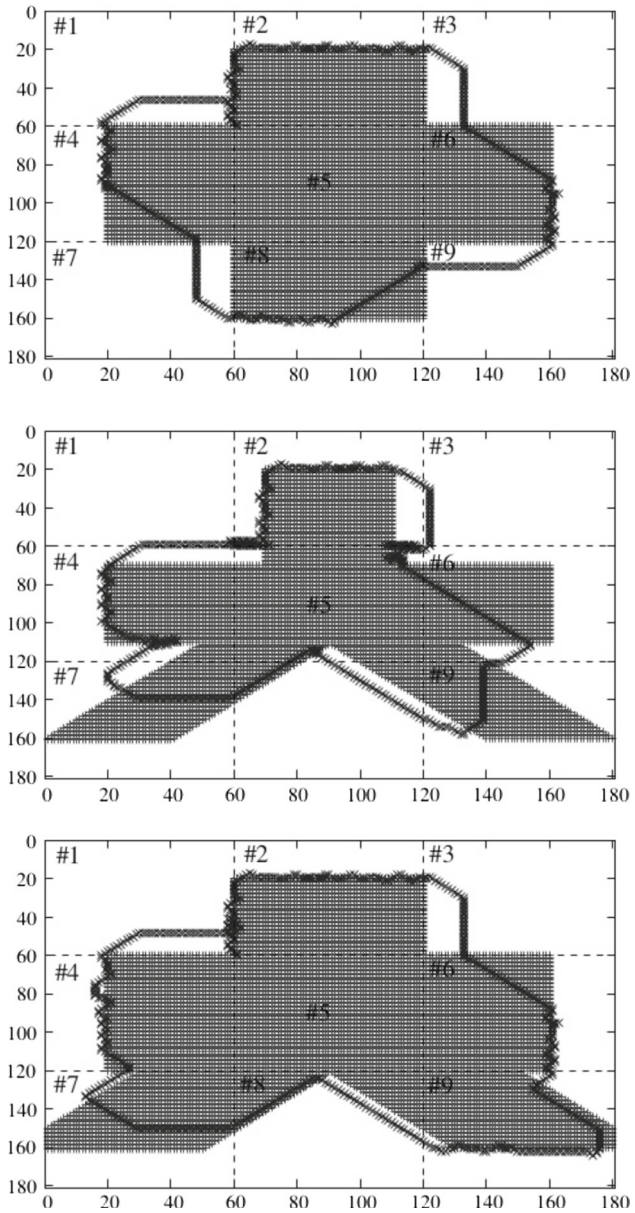


Fig. 10 Distorted shapes being tracked using the learned policy

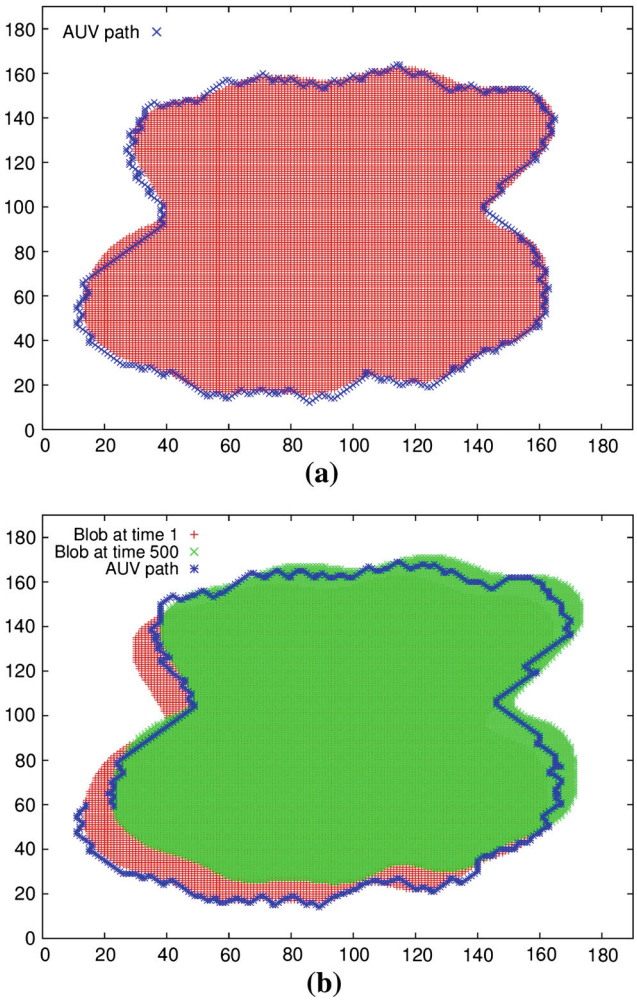


Fig. 11 Policy behaviour when dealing with a stationary (**a**) and moving (**b**) patch

4.4.1 Policy behaviour with moving patches

Real world patches can move over time, therefore we ran a set of experiments in order to test the policy behaviour when faced with such a situation. Namely, we generated a set of patches using the simulator introduced in Sect. 4.3. We then defined the position of each patch as a function of time and executed the policy against the *moving* patch. We observed, empirically, that the policy is able to react to the patch movements when it moves at reasonable velocity.

As an example, in Fig. 11 we compare the behaviour of the policy when faced with a patch fixed in its position (Fig. 11a), and when dealing with the same patch which is moving with a velocity of 50 m/h (Fig. 11b).

5 Policy integration with T-REX

We integrated our learned policy into T-REX by encapsulating it as a reactor. When deployed as part of a plan, the reactor interacts with the vehicle reactor to acquire position and chlorophyll data and generates navigation actions on the vehicle control timeline. The T-REX planner decides when the policy should be active and has freedom to stop and start it at will. When initiated, the policy is given the position of the bounding box for the patch tracking behaviour.

A fundamental part of achieving interaction between a symbolic action choice and the control system of the vehicle is the ability to translate both the low level data streams into symbolic state and the high level action choices into appropriate commands for the lower level. T-REX simpli-

```
1: PatchTracking(BoundingBox (UL,LR)) {
2:  link grid[0][0] to UL;
3:  link grid[SIZE-1][SIZE-1] to LR;
4:  repeat
5:   (x,y)=UTMtoGRID(N,E);
6:   S'=ApplyPolicy(S);
7:   (N,E)=GRIDtoUTM(x',y');
8:   //command the AUV to the new destination (N,E)
9:  until (mission-complete); }
```

Fig. 12 Patch tracking procedure

fies some of this task, so in order to deploy the learned policy it is only necessary to abstract position data (given as UTM[5] coordinates) into a grid coordinate position within the bounding box space and, conversely, action choices (forward, left-forward etc) need to be translated into waypoints for navigation actions. The use of the policy is shown in Fig. 12.

Embedding the policy in a real world environment requires an implementation of line 8 in Fig. 12, which is done via communication with the vehicle controller in T-REX.

A second important issue in adapting the simulation framework to the real environment is the reading of the chlorophyll level. In fact, in the 2D setting, we assumed to have a single chlorophyll level associated with each grid cell (which defines its black or white colour), so that the policy action only depends on that value (and the other state components). However, in a real setting, such an assumption is no longer valid, as multiple readings are taken within the real space corresponding to virtual cells. Thus, the set of all the new readings since the last decision point needs to be taken into account. To this end, we consider *voting* among all the new readings and we set the colour of the reached cell black or white, depending on whether the majority of the readings were greater or less than the threshold. We use this simple voting mechanism as our edge detection approach to provide a way of determining whether the chlorophyll is high enough for the location to be counted as being inside or outside the patch.

In the existing control system for the *Dorado* AUV, chlorophyll sensor data is available within the VehicleState timeline which updates inside a 1 Hz control loop, while navigation control is performed via the PositionTracker timeline which tracks vehicle waypoint execution. Both of these timelines were shared with the new PatchTracking reactor which, in turn, uses a timeline containing predicates Inactive or Active to decide when to activate the patch-tracking policy. Any higher level reactor could make use of this new timeline to activate the policy. In addition, the T-REX design allows an external user to send a new objective when the AUV is on the surface.

Beyond those steps most of the work of integration was to adapt the algorithm execution to the vehicle navigation constraints. For example, in order to consistently map the first 25 m of the water column the vehicle needs to go up and down this depth with its maximum pitch angle around 25°; this results in a minimum desired separation between waypoints of around 100 m.[6] This resulted in the need to adapt the interface between PatchTracker and its *External* timeline in order to modify the grid to be explored, so that each of its cells in turn is approximately this size. Further, the chlorophyll data sampled was integrated into a single value indicating whether the AUV crossed a chlorophyll patch during the last waypoint or not. This effort was internal to the reactor and resolved during the synchronization phase (Rajan and Py 2012) for this reactor. Overall, the net development effort to integrate the Patch Tracking algorithm into T-REX took at most 1 week's work and reflects robustness of its inherent design.

6 Sea tests

Our sea tests were conducted using the *Dorado* AUV in Monterey Bay in October 2011 with two missions on two consecutive days, both evaluating the success of the integration of the patch-tracking reactor with T-REX and the success of the patch-tracking policy. Both missions were of approximately 6 h duration and the AUV was under T-REX control for the entire duration of each mission while using the PatchTracking algorithm. The chlorophyll data was collected using a HydroScat-2^{TM} backscattering sensor from HOBI Labs which provides the chlorophyll fluorometry using its backscattering measures at 420 and 700 nm wavelengths.

6.1 First sea test

The first sea test comprised a mission to navigate from a starting position to the edge of a chlorophyll patch, and to track this edge while remaining inside a predefined 5 × 5km safety area. The AUV was dropped at the starting position and then monitored by the support vessel throughout the mission.

We hypothesised that the 3-d patch-tracking problem could be projected onto a 2-d surface without significant loss of information about the surface extent of the patch. On this basis, we built a grid at 25 meter granularity, so that the AUV would visit waypoints at 25 meter intervals. This constraint meant that the AUV would be deemed to have reached its target waypoints at underwater locations if it had traversed 25 horizontal meters but not reached the surface. Because of our hypothesis, this was not expected to be problematic.

[5] Universal Transverse Mercator coordinates.

[6] $2 * (25/\tan(25))$.

The yo-yo path of the AUV traverses a sinusoidal pattern from the surface to about 15 meters depth or more, and back.[7] Depending on the dive angle, this implies a minimal horizontal distance between waypoints visited on the surface. This behaviour is shown in Fig. 14. This figure shows a reconstruction of a slice of the water column based on the readings taken by the chlorophyll sensor during the first mission. The yo-yo of the AUV is superimposed.

It can be seen that the AUV followed a path attempting to track a frontier between low and medium to high chlorophyll levels. The high chlorophyll is at a depth of about 5 meters. Because the edge of the patch is not sharply defined, a decision about whether the AUV is in or out of the patch cannot be made based on instantaneous readings but has to be made over a period of reading generally high or generally low values. Therefore, readings were taken over segments of the path and majority voting was used to make the in/out decision. We used the entire dive depth for voting so that our edge detection mechanism was taking into account as much information as possible about the depth of the patch as well as its surface extent.

Figure 13 shows how the yo-yo path interacts with decision-making at 25 meter waypoints, leading to situations in which a succession of the decision points can be underwater. Because the patch has layers below the surface, this situation led to positive chlorophyll readings sometimes being taken underwater, which are then projected onto the surface patch.

The blank regions in the water column show that sometimes the AUV did not sample the full water column, but stayed below the chlorophyll layer for some period of time before resurfacing. If the AUV had been sensing below-threshold chlorophyll for some successive readings, its confusion level would rise and the policy would propose new waypoints to the side of the path in an effort to relocate the patch edge. If the maximum depth bound had not yet been reached, then the AUV control software might command the AUV to dive, in which case it would meet its new waypoint at an even deeper position. Figure 13 shows this scenario, and explains the shortened yo-yo peaks that can be seen in Fig. 14.

When decisions are made at deep waypoints, the upper part of the water column is not visited. A succession of deep waypoints might be visited before the depth of the yo-yo is sufficiently close to the maximal depth bound for the AUV to start to traverse the rising part of the yo-yo. This explains why the AUV often stays low down in the water for periods of time before resurfacing. There are at least 5 places in the overall mission where this occurs, and the behaviour is rational according to the policy which never realises that it is not traversing the surface edge.

There are two negative consequences of this behaviour: (i) if the vehicle stays low down for long periods then its localisation is compromised and its ability to visit specified waypoints is compromised as well; (ii) at low depths the AUV would tend to read low chlorophyll levels (most of the chlorophyll exists high up in the water column), causing it to infer that it is on the outside of the patch when it is actually underneath it.

This test demonstrated that the patch-tracking reactor was successfully integrated into `T-REX` and incorporated into its mission plan. The policy executed reliably and did not endanger the vehicle at any point during the mission. During the mission it appeared to successfully traverse a portion of a chlorophyll contour, defined by a specified chlorophyll threshold. However, our hypothesis was not fully supported because, at low levels, the AUV became lost and the projection of these sections of its path onto the surface cannot be interpreted as tracking a chlorophyll contour. Since the problem of measuring chlorophyll at depth was created by having too short a distance between waypoints, we decided to run a second experiment with waypoints separated by 100 meters instead of 25 meters. By setting the dive bound appropriately, a 100 meter surface distance could be achieved, but for engineering reasons a shorter surface distance could not be achieved reliably.

6.2 Second sea test

The first test revealed that the waypoints needed to be further apart to mitigate the problem of arriving at the waypoints at various depths. Our expectation was that 100 meter waypoints would reduce the occurrence of underwater readings, and focus the path of the AUV on the surface patch. We hypothesised that removal of the deep water readings would enable a clear surface contour to be reliably tracked. If this hypothesis were supported, it would give us a basis for tracking a 3-d structure using a layering approach, to be described in Sect. 7.

The second sea test took place the day after the first, so learning a new policy was not possible. We therefore adapted the operation of the existing policy in order to support 100 meter separation between waypoints. Our approach was to reinterpret the existing policy using a mechanism that rotates the virtual bounding box of the grid used by the policy, in order to align its top left corner with the top left corner of the actual bounding box (which is smaller than the grid).

During operation, as soon as the edge of the real bounding box is reached by the AUV, the virtual box is rotated to realign the edges with the real bounding box and the policy is reset to start. This means that, on every rotation, the policy starts

[7] Variability in the water column is along the vertical dimension. Since the *Dorado* platform can only move forward, the yo-yo pattern is the most efficient mechanism for the scientific study of water column properties.

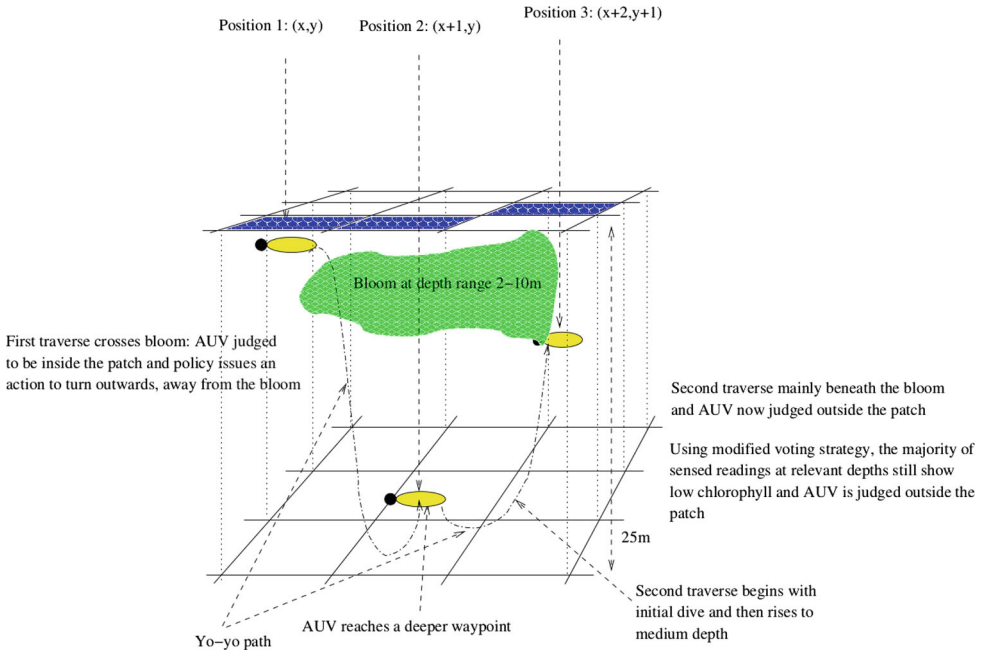


Fig. 13 Illustration of the problem of navigating the bloom in 3 dimensions. Decisions are made at positions 1, 2 and 3. The short traverses cause the AUV to visit successive deep waypoints, leading to a failure to observe the patch at shallower levels

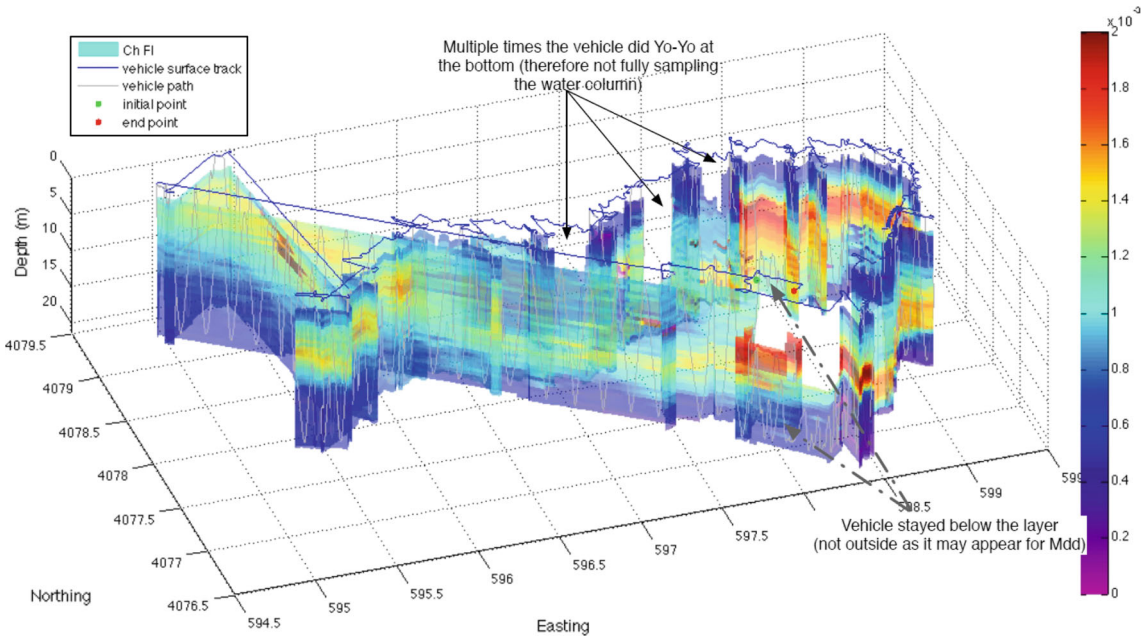


Fig. 14 Plot showing the path of the AUV during the first sea trial, with the measured (and interpolated) chlorophyll levels along the path. Note that the AUV trajectory is shown as a white line yo-yoing within the ribbon marking its progress through the ocean

again from its new top-left corner. This approach reduces the information that the policy can exploit, and might be expected to lead to a coarser reconstruction of the patch edge. However, the beneficial effect is that the AUV is commanded to reach 100 meter waypoints without changing the granularity of the policy grid and without requiring a larger bounding box. Figure 15 illustrates this process.

Figure 16 shows the path followed by the AUV under this modified policy. Looking at the West side, it can be seen that the AUV traversed about 200 m South of the box edge before apparently picking up the edge of the patch. It then followed this edge North. At the North West corner the bounding box was rotated. The AUV then followed the patch edge all the way across to the East side, following a similar contour to the one seen in Fig. 14 (but elongated, as described above). The rotating mechanism ensured that the AUV then successfully turned South when it reached the bounding box edge on the East side. The policy had some difficulty in the North

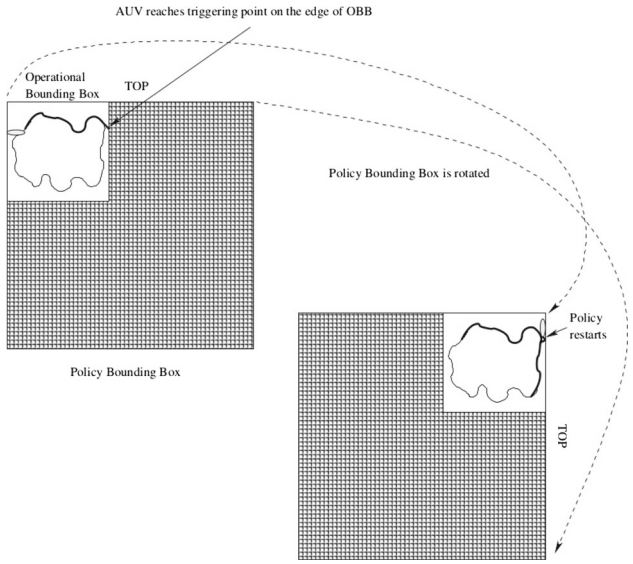


Fig. 15 Illustration of the rotating virtual bounding box positioned around the real bounding box during second sea trial

East corner, prior to traversing the East edge. It appears that the patch goes outside the box at this point and the policy was not designed to cope with this situation (it would require the AUV to cross the patch in order to pick up the East edge).

Figure 16 shows that the AUV appeared to pick up a strong chlorophyll signal and follow it around a 3-sided path. However, on inspection it can be seen that some decisions were made in deep water (indicated by the presence of attenuated yo-yos). This problem was caused by low-level control issues in the *Dorado* controller this time, and not by the selection of waypoints.

A difficulty presented in the second sea trial was that the localisation was poorer than in the first trial due to drift. This led to the AUV spending more time trying to reach waypoints, correcting its position locally until it was sufficiently close to a target to consider it achieved. This can be seen in Fig. 17, showing the estimated track followed by the AUV (in blue) based on DVL[8] and the corrected track (in red) following GPS fixes acquired when the AUV surfaces (at points marked). The corrected track shows that there was significant southerly drift over the course of the mission. The policy is affected by this because it makes it more difficult to correctly locate the chlorophyll readings within the cells being visited and to select target waypoints accordingly. The problem is potentially amplified by the greater separation of waypoints, although the more reliable surfacing gives access to more opportunities for GPS fixes than are possible when the AUV spends more time at depth.

The 100 meter separation did reduce the occurrence of underwater readings as expected, but the presence of marked drift, coupled with the control problems suffered by the *Dorado* on that day, made it difficult to determine whether a chlorophyll contour was successfully tracked over the duration of the mission. As a result, our second hypothesis was neither supported nor refuted by the test.

A significant difficulty with the interpretation of both tests lies in the absence of any ground truth that can be used to confirm whether the contour followed was meaningful. This is something that can only be done in detail in simulation. We therefore moved on to consider, in simulation using the `T-REX` simulator, better strategies for mapping the 3-d structure of blooms.

7 Simulation results on 3-D blooms

Our experiences in the sea trials led us to reconsider the tracking procedure. We recognised that the 3-d passage of the AUV through the 3-dimensional bloom requires a better treatment of the overall structure of the bloom itself. In particular, the tracking must focus on a specific depth-layer of the bloom, aiming to obtain the complete 3-d boundary through successive or cooperative missions at different depths.

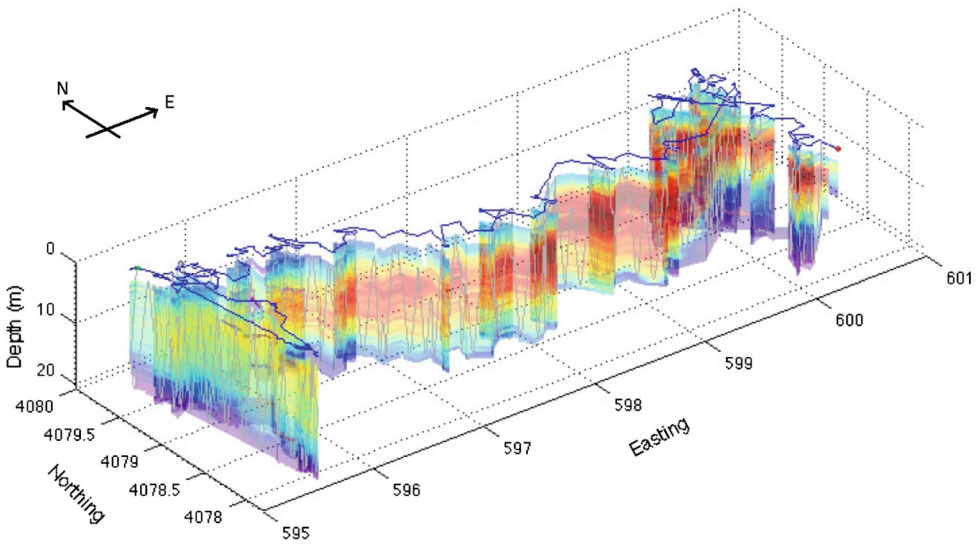
However, with the layer-based tracking a new problem arises, because multiple readings are taken within the same layer, and only the last one would not be representative. To this end, we decided to use a system of voting among all the readings within the same layer, to determine where the AUV is inside or outside the bloom.

We therefore extended our simulation testing to perform a set of 3-d simulations investigating the effectiveness of the layer-based tracking. We configured the `T-REX` framework to simulate the policy behaviour against a set of 3-d blooms. This uses the `T-REX` system in simulation mode, so that the AUV is simulated moving in three dimensions. The simulation does not consider environmental factors such as drift or currents, nor does it consider noise in the navigation sensing. Furthermore, because the patch is simulated by exact functions, the edge of the patch is well-defined and the simulated sensors reliably detect it. These are all limitations on the simulation compared with physical deployment. In the following we describe one of the simulation runs in more detail.

We considered the same bounding box used for the sea tests and we used the layer-based strategy running missions at different depths. In particular, we ran 10 missions, sampling the water in the depth layers $[i - 0.5 : i + 0.5]$, $\forall i \in [1 \ldots 10]$, and we used a chlorophyll threshold of 0.002 to determine whether the AUV was reading a significant chlorophyll level or not. Figure 18 shows a fragment of the `T-REX` log for the mission at 5 meter depth.

Lines 1–4 assert the current waypoint reached by the AUV. At this point, the voting of chlorophyll readings is computed (-1 in this case) and then the policy is used to determine the

[8] Doppler Velocity Log which estimates speed over ground.



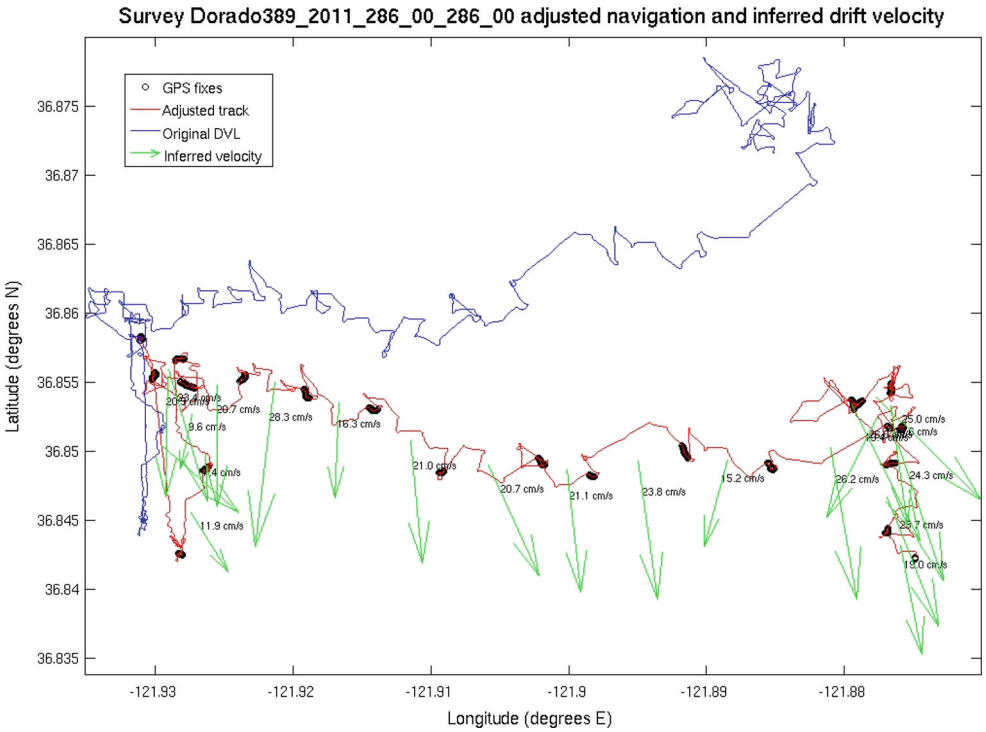


Fig. 16 Plot of the AUV path and interpolated chlorophyll levels during the second sea trial

Fig. 17 Drift rates and corresponding estimated corrected path of the AUV during the second sea trial

action to do next (line 5). Based on the suggested action, the resulting position in the virtual grid is computed, and then it is converted into the real point the AUV has to reach (line 6).

Lines 8–9 report the readings taken by the AUV when it was within the reference depth range and before reaching the next waypoint, which is asserted in lines 11–14. Hence, as before, the voting of chlorophyll readings is computed. Note that with new voting still negative (implying that the AUV is still outside the patch), the confusion level increases to 1 (line 15). The new destination point is then computed, according to the action suggested by the policy, which proves to be adequate, as the following readings are all significant (lines 18–20), meaning that the AUV is now back to the edge of the bloom (accordingly, the new confusion level is zero, as in line 26).

Two graphical representations of this simulation are shown in Figs. 19 and 20.

Fig. 18 Fragment of T-REX mission simulation log

```
1:  [exec][16976]  ON positionTracker ASSERT PositionTracker.Holds WITH {
2:      northing=4077304.7733 & easting=596466.0239 & heading=355.2507
3:      & minDepth=2.0000 & maxDepth=20.0000 & speed=1.5000
4:      & latitude=36.8367 & longitude=-121.9181 }
5:  [sim][16976] ChlVoting=-1. Policy(current_state,chl=0,confusion=0)
6:      ==> New destination: (104,29)=(4077374.5860,596530.7641)
7:
8:  [sim]x=4.07734e+06 y=596488. Chlorophyll level=0.00198796
9:  [sim]x=4.07736e+06 y=596511. Chlorophyll level=0.00189811
10:
11: [exec][17052]  ON positionTracker ASSERT PositionTracker.Holds WITH {
12:     northing=4077374.5860 & easting=596530.7641 & heading=46.3300
13:     & minDepth=2.0000 & maxDepth=20.0000 & speed=1.5000
14:     & latitude=36.8373 & longitude=-121.9174 }
15: [sim][17052] ChlVoting=-2. Policy(current_state,chl=0,confusion=1)
16:     ==> New destination: (105,30)=(4077298.3451,596600.9135)
17:
18: [sim]x=4.07737e+06 y=596551. Chlorophyll level=0.00209803
19: [sim]x=4.07735e+06 y=596568. Chlorophyll level=0.002027
20: [sim]x=4.07731e+06 y=596600. Chlorophyll level=0.00213546
21:
22: [exec][17143]  ON positionTracker ASSERT PositionTracker.Holds WITH {
23:     northing=4077300.0704 & easting=596608.8121 & heading=138.9608
24:     & minDepth=2.0000 & maxDepth=20.0000 & speed=1.5000
25:     & latitude=36.8367 & longitude=-121.9165 }
26: [sim][17143] ChlVoting=3. Policy(current_state,chl=1,confusion=0)
27:     ==> New destination: (106,30)=(4077297.7041,596610.6532)
```

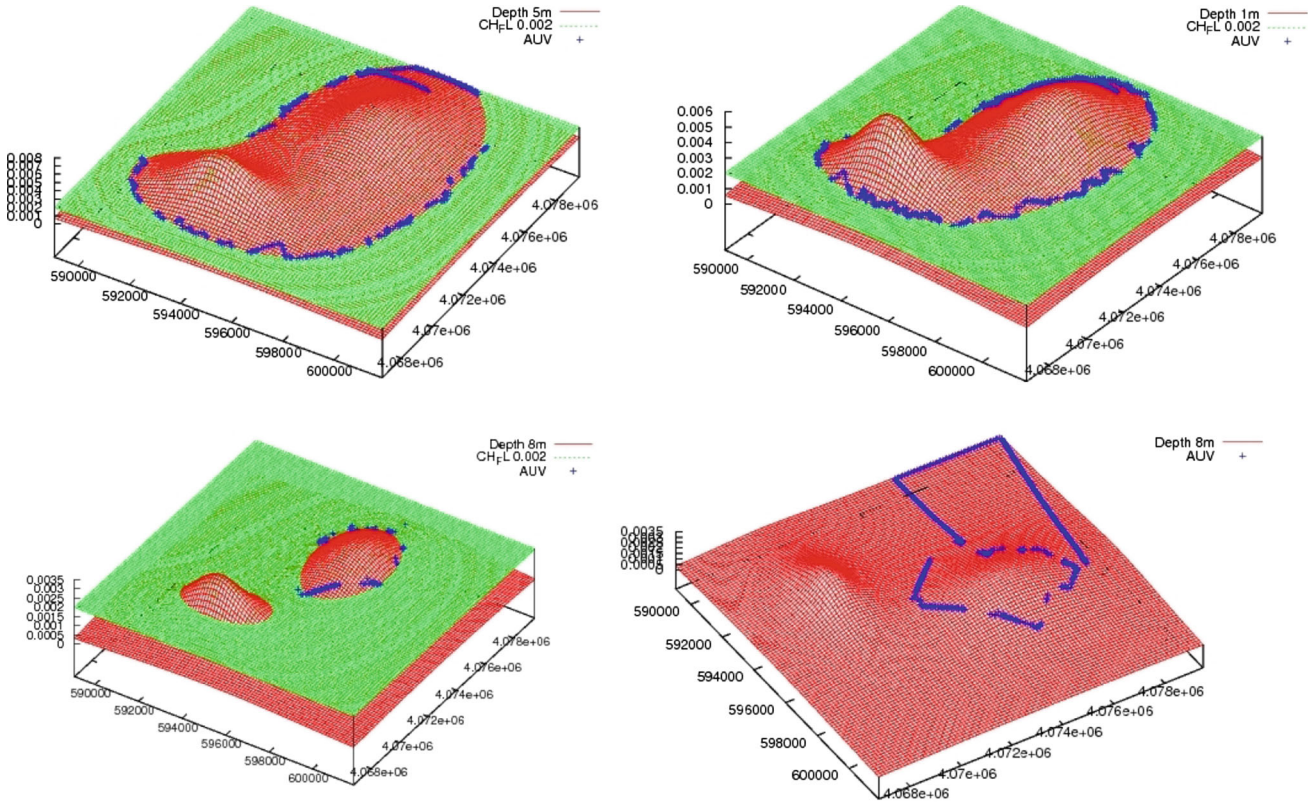


Fig. 19 AUV trajectories at different depths, showing simulated chlorophyll intensity at each depth

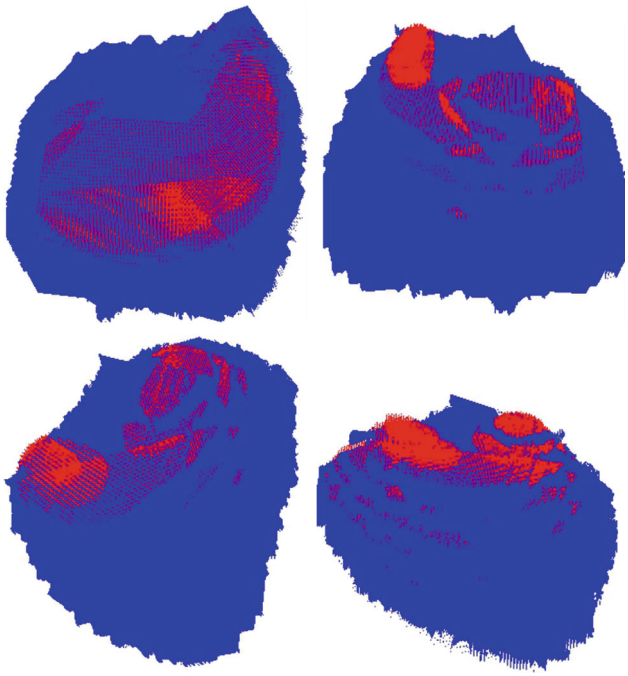


Fig. 20 Three dimensional view of a patch tracking mission, showing parts of the patch that are covered (*blue*), and not covered (*red*) by the AUV trajectories (Color figure online)

Figure 19 shows the behaviour of the AUV at different layers. In particular, the red surface represents the concentration of the chlorophyll in the patch at different depths (here we report only the missions at depths 1, 5 and 8 meters). The blue points represent the trajectory followed by the AUV, and the green layer is used to highlight the points where the chlorophyll level is higher than the reference threshold. As can be seen, at depths 1 meter and 5 meters the AUV performed very well, as it was able to follow the edge of the bloom perfectly. On the other hand, at depth 8 meters, the bloom was split in two small parts, and the AUV was able to track only one of them. The last figure shows the complete trajectory followed by the AUV at depth 8 m, and provides a graphical description of the initial behaviour of the AUV that, as described in the previous section, reaches the upper left point of the bounding box and then starts searching for the edge of the patch.

Figure 20 refers to the same test, but provides a set of 3-d views from different angles of the mission result (these figures are difficult to interpret without colour rendering). Each of the four 3-d views is produced by re-composing all the layers into a single 3-d image.

In each figure, the blue region represents the surface covered by the AUV, while the parts of the patch which have not been covered are in red. In order to make these images, all the data resulting from the 10 different mission simulations on that patch have been collected and analysed.

The second representation is consistent with the first as the flaw shown in Fig. 19 (trajectory at depth 8) can be seen in Fig. 20. However, looking at the overall behaviour of the AUV, only a very small portion of the bloom was not tracked by the AUV, at the deepest level of the patch where it was split in two small parts.

8 Related work

Fox, Long and Magazzeni previously demonstrated the successful use of the plan-based policy-learning approach in application to multiple battery load management, first in theory (Fox et al. 2011) and subsequently also in practice (Fox et al. 2012b). Ideas similar to plan-based policy-learning have been explored in a number of guises in the literature on Planning under Uncertainty and Machine Learning. Reinforcement Learning (Sutton and Barto 1998) has been well-explored and our approach can be seen as closely related to it. In Reinforcement Learning, the learning depends on having a reward function that can be used to evaluate actions and sequences of actions. Such a function could be constructed for our problem, based, for example, on the sum of the confusion levels over all visited states. The state space of the problem we consider is too large for direct solution using Reinforcement Learning methods (such as Q-learning Watkins 1989), so other techniques, such as function approximation, would be required.

In recent literature, sampling-based approaches such as the Policy Rollout and Improvement method of Fern et al. (2006), Yoon et al. (2002) or Teichteil-Königsbuch et al. (2010) have been contrasted with methods for planning under uncertainty (Sanner and Boutilier 2009). Hindsight Optimisation (HOP) (Chang et al. 2000; Fern et al. 2006) is a relatively well-researched technique for learning policies based on plans, as outlined in Sect. 3.2. An important distinction between HOP and our plan-based policy-learning method is that HOP is concerned with evaluating the costs of all of the states that might be visited during execution of the policy, while our approach is concerned with correlations between observable variables and actions selected in plans. In applications where a cost model is difficult to find, correlating actions to observables might be preferable.

Although our work is not directly concerned with the broader issue of control of AUVs through plan-based architectures, there is a strong history of work in plan-based control of robotic systems. Work in this area can be seen as providing context for the exploitation of our learned policy behaviours. In addition to the work of Rajan and his colleagues on `T-REX` , notable examples include Chien's EO-1 onboard planner (Tran et al. 2004) and AUV path-planning system (Thompson et al. 2010), Cogito (Beetz et al. 2012), Pearl (Pollack 2002) and Xavier (Simmons et al. 1997),

Amelia (Haigh and Veloso 1997) and Bullwinkle (Singh et al. 2000). In addition, Li and Williams (2008) and Patrón et al. (2008) have explored aspects of plan-based control of AUV missions in deployment.

Our work can be seen as closely related to Learning from Demonstration (LfD) (Argall et al. 2009). In LfD high quality example executions of a task are used as data in a learning process that constructs a control system capable of replicating the execution in similar situations to those encountered in the examples. LfD has been used to learn robot execution control systems (Peters and Schaal 2008), actions for planning (Mehta et al. 2011; Burbridge et al. 2012) and planner search guidance (Fern et al. 2004; Yoon et al. 2008). LfD relies on access to a large source of examples and the construction of a decision tree from these examples is a common technique for policy-construction from the dataset. To generate the basis for learning the policy, we use a planner to automatically plan the training examples. This is in contrast to the approach typically used in LfD, where the training examples consist of successful efforts at the task carried out by an expert.

Furthermore, in our approach, the training examples are not of the same structure as the problem to be solved by the learned policy. The training examples are planned paths around patches whose shapes were known to the planner in advance, while the policy must track the edge of an unknown patch by dynamically considering the values of observable variables. The planning problem has to be designed to allow the correspondence between state variables and observable variables to be learned. Thus, unlike LfD, the training examples do not have to resemble the actual mission situations.

9 Conclusions and future work

The work we report in this paper is a demonstration, in part in simulation, of the application of a plan-based policy-learning approach to the construction of controllers (at the policy level). Policies act as controllers at a suitable level of abstraction, selecting appropriate actions in response to sensed states. This level of control couples well with lower-level continuous reactive control systems, providing a mode-changing behaviour that can generate set points (such as navigation waypoint targets) for the execution-level control systems. The approach we demonstrate is an application of the earlier work on multiple battery management (Fox et al. 2011, 2012b), which was pursued both in simulation and, subsequently, in physical experiments. In a similar vein, the work we have presented here on the problem of tracking algal blooms is explored both in simulation and, to some extent, in physical deployment. The logistical constraints on deploying AUVs under autonomous control, purely for the evaluation of the control system itself, make it difficult to perform extensive testing (the AUVs and the associated ship support—both vessel and crew—are required for a demanding schedule of oceanographic study). Our sea trials offered important insights into the behaviour of our policies in practice, the limitations of the initial assumptions and the impact of navigational difficulties (both in control and in localisation).

We have shown that, inspired by the sea trials, a deployment of the policy to manage 3-dimensional tracking is possible and the approach gives us very positive results in simulation, offering a way to map the contours of a bloom at multiple depths. There remains much work to be done: we are still investigating the robustness and power of the policy-learning approach and of the policies it creates. In particular, we are anxious to better understand the ways in which the approach can be applied when uncertainty arises not only from the environment (which can be modelled through context-independent sampling of the possible environmental situations), but also from the actions of the executive itself (which cannot easily be managed by a priori sampling, since the samples depend on actions that are not yet selected). In terms of the continued evaluation of the specific application of the approach to the problem of patch-tracking, we are continuing to collaborate to find opportunities for future deployments and also exploring opportunities for deployment on alternative hardware systems.

Acknowledgments K.C.L. authors are partially funded by the EU FP7 Project 288273 PANDORA and the EPSRC Project "Automated Modelling and Reformulation in Planning" (EP/G0233650). MBARI authors are funded by a block grant from the David and Lucile Packard Foundation to MBARI and in part by NSF Grant No. 1124975 and NOAA Grant No. NA11NOS4780055.

10 Appendix A: Patch generation code

```
#include <cstdlib>
#include <iostream>
#include <cmath>
#include <ctime>

using namespace std;

double sqr(double s) {return s*s;};

int main(int argc,char * argv[])
{
 srand ((unsigned) time(0));

 double minx = -3;
 double maxx = -minx;
 double miny = minx;
 double maxy = -miny;
 double sigma = 0.5;
 double r0 = 1;

 int szx = (int) ((maxx-minx + 2*r0+2*sigma)/0.05)+1;
 int szy = (int) ((maxy-miny + 2*r0+2*sigma)/0.05)+1;

 double Z[szx][szy];

 for(int i = 0;i < szx;++i)
```

```
 {
   for(int j = 0;j < szy;++j)
   {
     Z[i][j] = 0;
   }
 }

 for(int i = 0;i < 50;++i)
 {
   double blobx = minx + rand()*(maxx-minx)/RAND_MAX;
   double bloby = miny + rand()*(maxy-miny)/RAND_MAX;

   for(int j = 0;j < szx;++j)
   {
     for(int k = 0;k < szy;++k)
     {
       Z[j][k] += 1.0/(2*M_PI*sigma) *
         exp(-((sqr(minx-r0-sigma+0.05*j-blobx)+
          sqr(miny-r0-sigma+0.05*k-bloby))/2/sqr(sigma)));
     }
   }
 }

 for(int i = 0;i < szx;++i)
 {
   for(int j = 0;j < szy;++j)
   {
     cout << Z[i][j] << `` '';
   }
   cout << ``\n'';
 }

 return 0;
};
```

References

Argall, B., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Beetz, M., Jain, D., Mösenlechner, L., Tenorth, M., Kunze, L., Blodow, N., et al. (2012). Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, *100*(8), 2454–2471.

Burbridge, C., Saigol, Z. A., Schmidt, F., Borst, C., & Dearden, R. (2012). Learning operators for manipulation planning. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 686–693), IROS.

Ceballos, A., Bensalem, S., Cesta, A., de Silva, L., Fratini, S., Ingrand, F., et al. (2011). A goal-oriented autonomous controller for space exploration. In *Proceedings of the 11th symposium on advanced space technologies in robotics and automation*, Noordwijk, the Netherlands.

Chang, H. S., Givan, R., & Chong, E. K. P. (2000). On-line scheduling via sampling. In *AIPS* (pp. 62–71).

Coles, A., Fox, M., Long, D., & Smith, A. (2008). A hybrid relaxed planning graphlp heuristic for numeric planning domains. In *Proceedings of international conference on automated planning and scheduling (ICAPS)* (pp. 52–59).

Coles, A. J., Coles, A., Fox, M., & Long, D. (2013). A hybrid lp-rpg heuristic for modelling numeric resource flows in planning. *Journal of Artificial Intelligence Research (JAIR)*, *46*, 343–412.

Das, J., Rajan, K., Frolov, S., Py, F., Ryan, J., Caron, D.A., et al. (2010). Towards marine bloom trajectory prediction for AUV mission planning. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 4784–4790).

Fern, A., Yoon, S. W., & Givan, R. (2004). Learning domain-specific control knowledge from random walks. In *Proceedings of the fourteenth international conference on automated planning and scheduling (ICAPS 2004)* (pp. 191–199).

Fern, A., Yoon, S. W., & Givan, R. (2006). Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. *Journal of Artificial Intelligence Research (JAIR)*, *25*, 75–118.

Fox, M., Long, D., & Magazzeni, D. (2011). Automatic construction of efficient multiple battery usage policies. In *Proceedings of the international conference on automated planning and scheduling (ICAPS)* (pp. 74–81).

Fox, M., Long, D., & Magazzeni, D. (2012a). Plan-based policy-learning for autonomous feature tracking. In *Proceedings of international conference on automated planning and scheduling (ICAPS)* (pp. 38–46).

Fox, M., Long, D., & Magazzeni, D. (2012b). Plan-based policies for efficient multiple battery load management. *Journal of AI Research (JAIR)*, *44*, 335–382.

Frank, J., & Jónsson, A. K. (2003). Constraint-based attribute and interval planning. *Constraints*, *8*(4), 339–364.

Gerevini, A., Saetti, A., & Serina, I. (2004). Planning with numerical expressions in LPG. In *Proceedings of the 16th Eureopean conference on artificial intelligence (ECAI)* (pp. 667–671).

Glenn, S., Kohut, J., McDonnell, J., Seidel, D., Aragon, D., Haskins, T., Handel, E., Haldeman, C., Heifetz, I., Kerfoot, J., Lemus, E., Lictenwalder, S., Ojanen, L., Roarty, H., Atlantic Crossing Students Jones, C., Webb, D., & Schofield, O. (2010). The trans-Atlantic Slocum glider expeditions: A catalyst for undergraduate participation in ocean science and technology. *Marine Technology Society*, *45*(1), 52–67.

Glibert, P. M., Anderson, D. M., Gentien, P., Granéli, E., & Sellner, K. G. (2005). The global complex phenomena of harmful algal blooms. *Oceanography*, *18*(2), 130–141.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Haigh, K. Z., & Veloso, M. M. (1997). High-level planning and low-level execution: Towards a complete robotic agent. In *Agents* (pp. 363–370).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, *26*, 191–246.

Hoagland, P., & Scatasta, S. (2006). The economic effects of harmful algal blooms. In E. Graneli & J. Turner (Eds.), *Ecology of harmful algae studies series, Chap. 30*. Berlin: Springer.

Hoffmann, J. (2003). The metric-FF planning system: Translating "Ignoring Delete Lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)*, *20*, 291–341.

Jónsson, A. K., Morris, P., Muscettola, N., Rajan, K., & Smith, B. (2000). Planning in interplanetary space: Theory and practice. In *Proceedings of the artificial intelligence planning and scheduling (AIPS)*.

Joshi, A., Ashley, T., Huang, Y. R., & Bertozzi, A. L. (2009). Experimental validation of cooperative environmental boundary tracking with on-board sensors. In *American control conference, 2009 (ACC'09)* (pp. 2630–2635), IEEE.

Li, H. X., & Williams, B. C. (2008). Generative planning for hybrid systems based on flow tubes. In *ICAPS* (pp. 206–213).

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 491–502.

Martin, F. (1996). *Kids learning engineering science using LEGO and the programmable brick*. In *Proceedings of the annual meeting of the American Educational Research Association*.

McDermott, D., et al. (1988). The PDPL planning domain definition language. In The AIPS 1998 *Planning Competition Commitee*. www.cs.yale.edu/homes/avm.

McGann, C., Berger, E., Boren, J., Chitta, S., Gerkey, B., Glaser, S., et al. (2009). Model-based, hierarchical control of a mobile manipulation platform. In *4th workshop on planning and plan execution for real world systems*, ICAPS.

McGann, C., Py, F., Rajan, K., & Olaya, A. (2009) Integrated planning and execution for robotic exploration. In *International workshop on hybrid control of autonomous systems (IJCAI'09)*, Pasadena, CA.

McGann, C., Py, F., Rajan, K., Ryan, J. P., & Henthorn, R. (2008). Adaptive control for autonomous underwater vehicles. In *Proceedings of the 23rd AAAI conference on artificial intelligence (AAAI)*, (pp. 1319–1324).

McGann, C., Py, F., Rajan, K., Ryan, J. P., & Henthorn, R. (2008b). Adaptive control for autonomous underwater vehicles. In *AAAI*, Chicago, IL.

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., & McEwen, R. (2008a). A deliberative architecture for AUV control. In *IEEE international conference on robotics and automation (ICRA)*, Pasadena.

Meeussen, W., Wise, M., Glaser, S., Chitta, S., McGann, C., Mihelich, P., et al. (2010). Autonomous door opening and plugging in with a personal robot. In *IEEE international conference on robotics and automation (ICRA)* (pp. 729–736). Anchorage, AK: IEEE.

Mehta, N., Tadepalli, P., & Fern, A. (2011). Autonomous learning of action models for planning. In *25th Annual conference on neural information processing systems (NIPS)* (pp. 2465–2473).

Muscettola, N. (1994). HSTS: Integrating planning and scheduling. In M. Fox & M. Zweben (Eds.), *Intelligent scheduling*. Los Altos: Morgan Kaufmann.

Oceanography, T. C. (2003). Robots in the deep. *Nature*, *421*(6922), 468–470.

Patrón, P., Miguelanez, E., Petillot, Y. R., & Lane, D. M. (2008). Fault tolerant adaptive mission planning with semantic knowledge representation for autonomous underwater vehicles.In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2593–2598).

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, *21*(4), 682–97.

Pinto, J., Sousa, J., Py, F., & Rajan, K. (2012). Experiments with deliberative planning on autonomous underwater vehicles. In *Workshop on robotics for environmental monitoring (IROS)*, Algarve, Portugal.

Pollack, M. E. (2002). Planning technology for intelligent cognitive orthotics. In *Proceedings of the 6th international conference on artificial intelligence planning systems (AIPS)* (pp. 322–332).

Py, F., Rajan, K., & McGann, C. (2010). A systematic agent framework for situated autonomous systems. In *International conference on autonomous agents and multiagent systems (AAMAS)*, Toronto, Canada.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Los Altos: Morgan Kaufmann.

Rajan, K., & Py, F. (2012). T-REX: Partitioned inference for AUV mission control. In G. N. Roberts, R. Sutton (Eds.), *Further advances in unmanned marine vehicles*, IEE (to be published).

Ryan, J. P., Dierssen, H. M., Kudela, R. M., Scholin, C. A., Johnson, K. S., Sullivan, J. M., et al. (2005). Coastal ocean physics and red tides: An example from Monterey Bay, California. *Oceanography*, *18*, 246–255.

Saigol, Z. A., Dearden, R., Wyatt, J. L., & Murton, B. J. (2009). Information-lookahead planning for auv mapping. In *Proceedings of the 21st international joint conference on artificial intelligence (IJCAI)* (pp. 1831–1836).

Sanner, S., & Boutilier, C. (2009). Practical solution techniques for first-order MDPs. *Artificial Intelligence*, *173*(5–6), 748–788.

Simmons, R. G., Goodwin, R., Haigh, K. Z., Koenig, S., O'Sullivan, J., & Veloso, M. M. (1997). Xavier: Experience with a layered robot architecture. *SIGART Bulletin*, *8*(1–4), 22–33.

Singh, S., Simmons, R. G., Smith, T., Stentz, A., Verma, V., Yahja, A., et al. (2000). Recent progress in local and global traversability for planetary rovers. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 1194–1200).

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction. Adaptive computation and machine learning*. MIT Press . http://books.google.co.uk/books?id=CAFR6IBF4xYC.

Teichteil-Königsbuch, F., Kuter, U., & Infantes, G. (2010). Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems (AAMAS)*.

Thompson, D. R., Chien, S. A., Chao, Y., Li, P., Cahill, B., Levin, J., et al. (2010). Spatiotemporal path planning in strong, dynamic, uncertain currents.In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 4778–4783).

Tran, D., Chien, S. A., Sherwood, R., Castaño, R., Cichy, B., Davies, A., et al. (2004). The autonomous sciencecraft experiment onboard the EO-1 spacecraft. In *Proceedings of the 19th national conference on artificial intelligence, 16th conference on innovative applications of artificial intelligence (AAAI/IAAI)* (pp. 1040–1041).

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. Thesis, Cambridge University, England.

Yoon, S. W., Fern, A., & Givan, R. (2002). Inductive policy selection for first-order MDPs. In *Proceedings of the conference on uncertainty in AI (UAI)* (pp. 568–576).

Yoon, S. W., Fern, A., & Givan, R. (2008). Learning control knowledge for forward search planning. *Journal of Machine Learning Research*, *9*, 683–718.

Zhang, Y., McEwen, R. S., Ryan, J. P., Bellingham, J. G., Thomas, H., Thompson, C. H., et al. (2011). A peak-capture algorithm used on an autonomous underwater vehicle in the 2010 Gulf of Mexico oil spill response scientific survey. *Journal of Field Robotics*, *28*(4), 484–496.

Daniele Magazzeni received a Ph.D. in Computer Science from University of L'Aquila, Italy, in 2009. He is a Lecturer in the Department of Informatics at King's College London. His research explores the links between automated planning, controller synthesis and model-checking verification, with a particular focus on temporal continuous planning, planning in mixed discrete-continuous domains, and hybrid systems control and verification. His main contributions on these fields are represented by a model-checking based planner for hybrid domains, and the new plan-based policy-learning technique to deal with planning under uncertainty.

Frédéric Py received a Ph.D. in robotic architectures from the Universiy Paul Sabatier at the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS) research institute, Toulouse, France, in 2005. He is a Software Engineer in the Autonomous Systems Group, Monterey Bay Aquarium Research Institute (MBARI), Moss Landing, CA. His work is focused on the design and implementation of autonomous underwater vehicles (AUVs) of the T-REX executive supporting constrained-based planning and plan execution with a focus on simultaneous deliberation and execution for autonomous systems.

Maria Fox is a Professor in AI Planning at King's College London. Her research is in temporal and numeric planning and planning in mixed discrete-continuous domains. Her recent work focusses on the relationship between task planning and control of robotic systems including underwater vehicles.

Derek Long is a Professor in AI Planning at King's College London. He has made many contributions in temporal and numeric planning and their application, in particular considering practical frameworks for plan repair and re-planning in the context of onboard planning for space robotics.

Kanna Rajan is the Principal Researcher for Autonomy at the Monterey Bay Aquarium Research Institute where he is engaged in autonomy for marine robotics. Prior to coming to MBARI in 2005 he spent a decade at the NASA Ames Research Center where he was on two NASA missions, the New Millennium Deep Space 1 Remote Agent Experiment in 1999 and the 2003 Mars Exploration Rovers mission. He was in the doctoral program at NYU/Courant Institute prior to which he was in the Knowledge Systems group at American Airlines.