Dynamically Extending Planning Models using an Ontology

Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, and **Bram Ridder**
King's College London
London WC2R 2LS
firstname.lastname@kcl.ac.uk

Valerio De Carolis, David Lane, and **Francesco Maurelli**
Herriot-Watt University
Edinburgh, Scotland, UK EH14 4AS
vd63@hw.ac.uk, *d.lane@hw.ac.uk*

Abstract

In this paper we couple a deterministic planner with an ontology, in order to adapt to new discoveries during plan execution and to reason about the affordances that are available to the planner as the set of known objects is updated. This allows us to extend the planning agent's functionality during execution. We use as an example planning for persistent autonomous behaviour in underwater vehicles. Planning in this scenario takes place in a symbolic model of the environment, simulating sequences of possible decisions. Ensuring that the simulation remains robust requires careful matching of the model to the real world, including dynamically updating the model from continuous sensing actions. We describe how our system constructs an initial state for planning, using the ontology; how the ontology is also used to determine the results of each action performed by the planner; and finally demonstrate the performance of the system in a simulation, in which two AUVs are required to cooperate in an unknown environment, demonstrating that with additional reasoning the planning system is able to make new efficient choices, taking advantage of the environment in new ways.

1 Introduction and Motivation

AI Planning [Ghallab *et al.*, 2004] supports a key requirement of intelligent robotics: the ability to perform strategic task-level planning, taking into account limited resources, time, environmental constraints and long-term goals. Planners rely on having access to a rich model of the environment, the objects within it, and the actions that can be performed on the different kinds of objects. A major challenge is that knowledge of the environment constantly changes during plan execution, so the set of objects that can be manipulated cannot be fixed in advance. Instead, the planner's model has to be adapted and updated as new discoveries are made. Acquiring new knowledge in an autonomous way, adapting behaviour accordingly, is a fundamental requirement of persistent autonomous behaviour. Path-planning [Lavalle, 2006] is also a fundamental underlying requirement, but we do not address this topic in this paper.

To enable the acquisition and interpretation of data, and inference over the resulting new knowledge, we provide an *ontology* as one of the components of an autonomous plan-based system. This paper addresses using an ontology in the dynamic construction of AI planning models, using underwater mission-planning for AUVs as an example. In the work presented here, the ontology has two roles: to recognise instances of known concepts from sensed data (eg: to be able to distinguish a valve from a weld), and to identify affordances with the newly recognised objects (eg: to recognise that, being a valve, the object can be grasped, turned, etc). In this way, interpretation of sensed data opens up new reasoning choices for the planner.

Our goal is to show that, equipped with task planning and an ontology, an autonomous system can perform long-term operations without human intervention, adapting to discoveries and increasing its functionality over time. Our approach is to plan operations over a horizon, and replan whenever the ontology updates the planning model. We make two main contributions:

1. The use of a temporal planner, rather than a reactive strategy, to control underwater missions, in order to anticipate and avoid problems rather than simply react to them when they cause actions to fail.
2. The use of an ontology to provide object classifications and affordances to the planner, to improve the planner's interaction with the world.

When planning in robotic domains the actions available to the planner correspond to interactions between the robot and recognised objects in the environment, an idea explored by Geib et al. [2006] as Object-Action complexes. If a plan-based intelligent robot is placed in an environment with a fixed model, the robot might not be able to exploit all of its capabilities as objects are not correctly recognised, or are of unknown types that nevertheless afford known interactions. Using ontological reasoning in conjunction with planning, we extend the capabilities of the planner to more closely match those of the robot in the environment.

We consider the problem of autonomous inspection and maintenance of a seabed oil installation. Regular inspection and maintenance of the facility is to be carried out using autonomous underwater vehicles (AUVs) over extended horizons, so the system must be able to deal with unexpected dis-

covery, and be able to model actions that can be performed in the environment in the symbolic language of the planner. The missions that must be undertaken have many temporal constraints and characteristics. For example, depending on how long it takes to achieve a planned task, the timeframe in which other tasks might be completed can be affected. Thus, depending on the importance of tasks, the planner might decide to make more time available for one task rather than another. In cooperative tasks, the planner has to time the behaviours of the cooperating robots so that they coincide at the right locations.

The combination of planning with other modules that model knowledge about the domain has been explored in other contexts. Several approaches to building semantic maps have been developed. Petrick et al. [2013; 2014] address the issue of joining continuous low-level sensor data with planning in a partially-known symbolic representation with sensing actions in contingent planning. This is similar to the problem that we propose to tackle with an ontology module. Where Petrick et al. cope with unknown knowledge at a Planning level, our focus is on using an ontology to reason about discovered objects, and by so doing extend the possible means of interacting with them. Tenorth et al. [2010] and Galindo et al. [2008] focus on combining semantic knowledge with spatial data to form a *semantic map* of the environment. Tenorth et al. in particular deal with attaching semantic information to a spatial map using an ontology in the ROS[1] framework for indoor household tasks. We define a similar, but general approach to linking an ontology with a planner in ROS.

We validate this approach in an under-water mission, which we describe in Section 2, along with discussion related work. In Section 3 we describe the details of our integration between planning and ontology. We describe the simulations with our system and then conclude in Sections 4 and 5.

2 Planning with Ontologies

Ontologies organise knowledge around concept hierarchies and the relationshipsand attributes between these concepts and instances of them. Considerable work has been carried out on developing representational languages based on description logics and inferences over the sentences they record [Gruber, 1993]. In our framework, an ontology is used as a way for the robot to organise the knowledge about the physical world in which the AUVs operate, not just as geometric concepts, but also as richer structures that offer access to affordances expressed as action templates available in a PDDL [Fox and Long, 2003] domain model (actions applicable to objects of the corresponding types).

In this work we use an ontology to detect new objects and reason about their identification and the relations between them. The planner adapts to unexpected features by revising its abstract, deterministic model of the world and replanning to take account of the new features, while relying on robust control and signal processing systems to handle inaccuracy and noise.

[1]Robot Operating System (ROS); http://www.ros.org; last accessed Apr. 2014

Combining task planning and control have been considered in much prior work. Recent works include: the constraint-based temporal planning system, EUROPA-2, in the T-REX framework [McGann *et al.*, 2008; Py *et al.*, 2010]; combining task and motion planning onboard the PR2 robot [Srivastava *et al.*, 2014]; using homotopy classes to guide path planning [Hernández *et al.*, 2011b; 2011a]; generating a coarse plan to initialise the inspection of an unknown hull [Englot and Hover, 2010]; using a plan-based policy to guide an AUV for autonomously tracking the boundary of the surface of a partially submerged harmful algal bloom [Fox *et al.*, 2012] and autonomous underwater maintanence, explored in the context of temporal planning [Cashmore *et al.*, 2013; 2014];.

2.1 Case Study

We consider a context in which two AUVs have to cooperate to accomplish tasks in a long term mission that requires sustained autonomy: the inspection and maintenance of a seabed facility. Some tasks cannot be achieved by a single AUV and coordination between two or more AUVs is required. Our problem scenarios in this paper are set in this context, focussing on a cluster of inspection tasks. Maintenance tasks take place in a dynamic environment; currents will move the robots, visibility might become obscured, and extraneous events, such as sea animals passing by, might interfere with the execution of an action. In order to find plans that are robust to the uncertainty inherent in the environment, we construct domain models that are conservative with respect to resource requirements (e.g. time and energy). The uncertainty is abstracted by embedding it in the resource estimates used in planning the actions. In this way, we can exploit powerful deterministic temporal planning methods, rather than probabilistic methods which, although they model and reason about uncertainty directly, are much less performant.

During the execution of the plan the ontology is continuously updated using parsed sensor data. As a result of some observations and updates to the model of the world, an executing plan can become invalid (some of the assumptions on which it rests can be violated). This situation is identified by the reasoning within the ontology, coupled with the content on the plan, and will trigger a replan. For example, consider the case where an AUV plans to perform an inspection of a structure, and then move on towards a valve panel. While performing the inspection, the sonar data continuously updates the map. Suppose, during execution of the inspection, the path planned between the structure and the valve panel is found to be blocked. This information is given immediately to the planning system, which will find a new path to the valve panel, and might decide to change the details of the structural inspection to better fit the new strategic plan. This is described in more detail in Section 3. It should be emphasised that the replanning carried out in this case is not path planning (although path planning is a part of the problem), but construction of a *task* plan, linking and coordinating actions between AUVs to achieve the goals of the original plan.

In this paper we focus on a mission involving two AUVs, which we call the *structure inspection task*. The objective is for the two AUVs is to inspect a structure. This requires

exploring a set of inspection points, by navigating close by and directing the sonar and imaging equipment towards them. However, if there are pillars in the structure, the AUVs are able to inspect each complete pillar as a single structure, by observing from a greater distance. This action explores a larger area than any single inspection point, but poses some challenges, as the pillars are deep down in the water, so that a light has to be shone upon the target for the observation to be of sufficient quality.

One solution would be to equip a single AUV with both a camera and a light, but in such a case the light would come from the same direction as the camera, and backscatter would compromise the image quality. Instead, one AUV should approach the pillar and shine the light from an oblique angle, while the other AUV approaches the target to make the observation. Note that it is not possible to sequentialise the behaviour of the two AUVs for the pillar inspection task. The concurrency of the behaviours is crucial to the success of the inspection. This task therefore requires temporal coordination of activities, and temporal reasoning is key for the effective planning of this mission. The combination of temporal coordination of two AUVs, the assignment of tasks between them and the timing and ordering of tasks is the role of a task plan and cannot be resolved by path planning, although path planning is necessary in determing which navigation tasks are achievable and by what route.

The AUVs are placed inside an environment with a number of pillars and other structures. Initially the AUVs have little knowledge of the environment. The AUVs and planning system are capable of performing an inspection in an unknown environment by observing dynamically generated inspection points and replanning as new observations invalidate the current plan.

3 Integration

The symbolic model of the world is provided by the ontology though a ROS interface that the planner uses to construct the initial state of the problem. The symbolic representation of the locations that the AUVs can visit, the objects, and the relationships between them are provided by the ontology. This process is described in four parts. First we describe our model for the structure inspection task, and how it is dynamically updated. We discuss how the model recorded in the ontology is delivered as a problem description for the planner. We then show some solutions to examples of these generated planning problems, and finally describe how these plans are executed.

3.1 Model

The objects collated in the ontology are either known a priori or are derived from analysing sensor data that is collected throughout the execution of a plan. The types of objects include 3d-points within the volume bounding the mission region, which has several subtypes (waypoints, inspection points and strategic waypoints), and structures, which includes the subtypes of pillars and one representing the image slice a sonar captures when intersecting a cylinder, which we call a *Circle* (though it might be elliptical, depending on angle, and will be partially occluded by the solid structure).

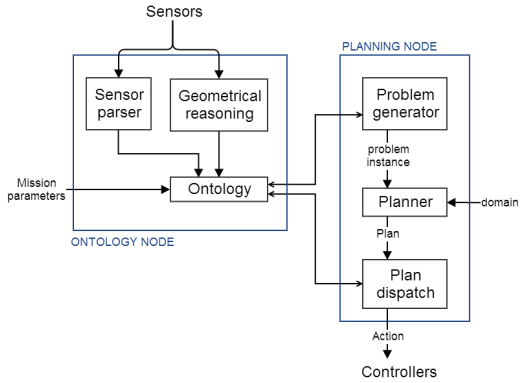

Figure 1: The architecture of integration between the planning system and ontology. The ontology provides the initial state of the planning problem. The ontology is also involved in the exectution of the plan, alerting the planning system when an important part of the environment changes, or is discovered to be different from what is expected.

Figure 1 displays the relationship between the planning system and the ontology.

The possible trajectories the AUVs can traverse are determined by a set of *waypoints*. These waypoints are created using a probabilistic road map (PRM) [Kavraki *et al.*, 1996; Lavalle, 2006], and stored in the ontology. We use an octomap [Wurm *et al.*, 2010] to check if a waypoint collides with an obstacle in the world. Similarly we use the octomap to determine whether the AUVs can traverse between two waypoints. The octomap is built from sensor data and continuously updated.

Inspection points are added to the ontology. These are areas that must be observed by the AUV, either areas of unexplored space, or the unseen sides of possible pillars.

The PRM is augmented with additional waypoints – called *strategic waypoints* – these waypoints are stored in the ontology to provide a denser collection of waypoints around points of interest (in our case the locations of possible pillars and unexplored space). For example, if the sonar picks up a signature that is roughly cylindrical, this is recognised as a new *Circle* and stored in the ontology. New inspection points are inferred as a consequence of the knowledge that such structures can be inspected on all sides. This has the effect of enabling and encouraging the planning system to plan to inspect the opposite side of this object to determine whether it is a *Pillar*.

This information can also be used to 'clean up' the octomap by removing noise. If an object is determined to be a pillar, its shape is known and errors from the sonar can be corrected.

3.2 Constructing the Planning Problem Instance

In our domain, the state of each AUV is partially described by its position, given by a waypoint. The AUV can perform six actions, namely *do_hover_fast*, *do_hover_controlled*, *correct_position*, *illuminate_pillar*, *observe_pillar*, and *observe_inspection_point*, as shown in Figure 2.

```
(:durative-action do_hover_fast
 :parameters (?v - vehicle ?from ?to - waypoint)
 :duration ( = ?duration (* (distance ?from ?to)
         (invtime ?v)))
 :condition (and (at start (at ?v ?from))
         (at start (connected ?from ?to)))
 :effect (and (at start (not (at ?v ?from)))
         (at end (near ?v ?to))))


(:durative-action illuminate_pillar
 :parameters (?v - vehicle ?wp - waypoint ?p - pillar)
 :duration ( >= ?duration 0)
 :condition (and (over all (at ?v ?wp))
        (at start (can_observe_pillar ?v ?wp ?p)))
 :effect (and
        (at start (pillar_illuminated ?p))
        (at start (not (can_observe_pillar ?v ?wp ?p)))
        (at end (not (pillar_illuminated ?p)))
        (at end (can_observe_pillar ?v ?wp ?p))
        (at end (near ?v ?wp))))

(:durative-action observe_pillar
 :parameters (?v - vehicle ?wp - waypoint ?p - pillar)
 :duration ( = ?duration 10)
 :condition (and
        (at start (at ?v ?wp))
        (at start (can_observe_pillar ?v ?wp ?p)))
        (over all (pillar_illuminated ?p))
 :effect (and
        (at start (not (can_observe_pillar ?v ?wp ?p)))
        (at end (observed_pillar ?p))
        (at start (not (at ?v ?wp)))
        (at end (near ?v ?wp))))
```

Figure 2: A fragment of the PDDL inspection-task domain.

The *do_hover_fast* action moves the AUV between two connected waypoints (which, by construction, are the end-nodes of a collision-free edge). Since the fast motion does not take into account final orientation, it only arrives `near` the desired pose. The position must then be corrected. The duration of the action depends on the distance between the two waypoints.

The observe actions allow the AUV to observe an inspection point or a pillar. The precondition requires the AUV to be at a waypoint from which the target inspection point is (partially) visible. Furthermore, the *observe_pillar* action requires the pillar to be illuminated over the whole duration of the observe action. The *illuminate_pillar* action, whose duration is decided by the planner, needs to be performed by a different AUV to meet this requirement.

The problem instance is described using a collection of *objects*, their initial states, and a goal. The objects correspond to object types known by the ontology, and the initial state of the problem instance is generated from the attributes of these objects, also stored in the ontology. This describes the current known state of the world. In the structure inspection task the goal is automatically generated from the initial state, given the current knowledge of the environment. This is done by adding the requirement that every inspection point and pillar has been fully observed. This data is accessed using a ROS interface. In a typical scenario, the goal is initially to observe a set of inspection points $p_1, \ldots, p_n$. When a pillar is discovered, the goal is dynamically updated, and some inspection points $p_j, \ldots, p_k$ are removed from the goal and replaced with the goal of observing the pillar (as it subsumes multiple

```
(define (problem inspection-task-p1)
 (:objects
       auv - vehicle
       wp1 wp2 wp3 ... - waypoint
       ip1 ip2 ip3 ... - inspectionpoint)
       p1 ...          - pillar)
 (:init
  (at auv wp1)
  (= (mission-time) 0)
  (= (observed ip1) 0)
  (connected wp1 wp2) (connected wp2 wp1)
  (= (distance wp1 wp2) 7.16958)
  (= (distance wp2 wp1) 7.16958)
  ...
  (cansee auv ip4 wp12)
  (= (obs ip4 wp12) 0.445331)
  ...
 )
 (:goal (and (>= (observed ip1) 1)
             (observed_pillar p1)
             ...
 ))
 (:metric minimize (total-time)))
```

Figure 3: A fragment of the PDDL inspection-task problem instance.

inspection points, $p_j, \ldots, p_k$, as determined by the appropriate geomtric reasoning in the ontology). Figure 3 shows a fragment of a problem instance.

3.3 Solving the Planning Problem

To solve the problem, we use the temporal planner POPF [Coles *et al.*, 2010]. As described earlier, the planner deals with coarse-grained events: in this case movement between waypoints and observation of inspection points. Example plans in PDDL representation are shown in Figures 4 and 5. In both plans, the AUVs are explicitly given concurrent activities and the planner minimises the duration of the plans. However, in the second case, the plan *requires* concurrency to allow the correct illumination of the pillar during the (long range) inspection task. Note that the illumination duration has been set by the planner to meet the demands of the inspection task.

3.4 Execution

The controllers are responsible for achieving the actions and providing feedback. There are two possible reasons for replanning:

1. *action failure*: an action execution reports failure, using the ROS action feedback, or times out; and
2. *change of environment*: the ontology notifies the planner of a change in the environment that invalidates the plan, or new information, such as new object instances, pertinent to mission goals.

A single plan governs both AUVs. We make the assumption that when replanning, the vehicles can coordinate and share information as required. In practice, this communication is difficult, and in future work we will consider how the plans execution and replanning requests can be coordinated between independent vehicles.

Once the planner has found a plan, the actions are converted into ROS messages and sent to the AUVs. This is done by tokenizing the plan (e.g. figures 4 or 5) and passing the

Without knowledge of Pillars		
Plan time	PDDL action	duration
0.000:	(correct_position auv0 wp_auv0)	[10.000]
0.000:	(correct_position auv1 wp_auv1)	[10.000]
10.001:	(do_hover_fast auv1 wp_auv1 s16)	[46.469]
10.001:	(do_hover_controlled auv0 wp_auv0 s0)	[14.274]
24.276:	(observe_inspection_point auv0 s0 i0)	[10.000]
34.277:	(correct_position auv0 s0)	[10.000]
44.278:	(do_hover_controlled auv0 s0 s1)	[16.971]
56.471:	(correct_position auv1 s16)	[10.000]
61.250:	(observe_inspection_point auv0 s1 i1)	[10.000]
66.472:	(observe_inspection_point auv1 s16 i16)	[10.000]
71.251:	(correct_position auv0 s1)	[10.000]
76.473:	(correct_position auv1 s16)	[10.000]
81.252:	(do_hover_controlled auv0 s1 s2)	[16.971]
86.474:	(do_hover_controlled auv1 s16 s20)	[15.000]
98.224:	(do_hover_fast auv0 s2 s10)	[66.000]
101.475:	(observe_inspection_point auv1 s20 i20)	[10.000]
111.476:	(correct_position auv1 s20)	[10.000]
121.477:	(do_hover_controlled auv1 s20 s17)	[22.649]
144.127:	(observe_inspection_point auv1 s17 i17)	[10.000]

Figure 4: A PDDL plan for an inspection task, found using POPF. Each action has an associated duration, and expected dispatch time, which may differ from the actual execution. Each waypoint and inspection point is associated with its coordinates, as stored in the ontology. In this plan no pillars have been recognised by the ontology.

With knowledge of Pillars		
Plan time	PDDL action	duration
0.000:	(correct_position auv0 wp_auv0)	[10.000]
0.000:	(correct_position auv1 wp_auv1)	[10.000]
10.001:	(do_hover_fast auv1 wp_auv1 s16)	[46.469]
10.001:	(do_hover_controlled auv0 wp_auv0 s0)	[14.274]
0.000:	(correct_position auv0 wp_auv0)	[10.000]
0.000:	(correct_position auv1 wp_auv1)	[10.000]
10.001:	(do_hover_controlled auv0 wp_auv0 s3)	[10.833]
10.001:	(do_hover_fast auv1 wp_auv1 s16)	[46.469]
20.835:	(do_hover_fast auv0 s3 s20)	[84.546]
56.471:	(correct_position auv1 s16)	[10.000]
75.383:	(illuminate_pillar auv1 s16 pillar2)	[50.000]
105.382:	(correct_position auv0 s20)	[10.000]
115.383:	(observe_pillar auv0 s20 pillar2)	[10.000]

Figure 5: A PDDL plan for an inspection task, found using POPF. Each action has an associated duration, and expected dispatch time, which may differ from the actual execution. Each waypoint and inspection point is associated with its coordinates, as stored in the ontology. In this plan, knowledge of pillars exists in the intial state, allowing faster observation of the structure.

actions to the AUV controllers. The actions are dispatched to the two AUVs *concurrently* as scheduled by the plan.

The AUV controllers provide feedback to the executor. If the action is successful, then at the scheduled time, the next action can be dispatched to that controller. If the action is failed, then replanning is triggered. If an action is taking too long to complete, the action is cancelled by the executor and replanning is triggered.

During the execution of an action the executor may cancel the action if the plan is invalidated or the current action

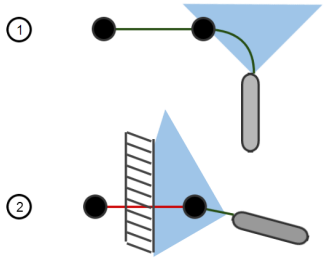

Figure 6: The plan is executed with sonar continuously updating the environment. The plan is invalidated in 2 and replanning is triggered.

is no longer desirable. Note that, in our framework, *replanning* is based on *reformulating* the inspection task as a new planning problem. This re-modelling is performed dynamically, as new information becomes available – the PRM is continuously updated according to the new information about the environment, and the ontology with detected objects and structures to inspect. This reformulation allows the AUVs to adapt to new discoveries during execution.

For example, by collating information continuously it is possible that the obstacle is detected long before the action is to be dispatched. The system can alter the plan before the action is dispatched, thereby avoiding dead-ends and inefficiencies. For example, consider the following simple scenario in figure 6: the AUV is moving between two waypoints, and the sonar detects a wall. The wall obstructs the planned hover action, but will not interfere with the current action. Clearly, replanning should take place to avoid dispatching the doomed action. In this case the obstructed connection is removed from the ontology, the planning system is notified of the change, and finds a new route before dispatching the action.

4 Experimentation

Our objective is to show that the planner and ontology can interact to support the execution of a complex mission. We have designed experiments that demonstrate the following features:

1. Controlled failure of an executing plan in the event of the discovery of new information
2. The discovery of new object instances and their affordances, and updating of the planning problem
3. Replanning of a cooperative mission involving coordinated activity of the two AUVs to achieve mission goals

We show that by continually augmenting the knowledge available, the ontology gives the planner access to previously unknown parts and affordances of the environment leading to plans that are shorter and more efficient than those that can be found without the ontology. We demonstrate the role of planning by considering a mission with interesting temporal structure. The two AUVs must act concurrently in order to inspect pillars. A reactive strategy, in which one or more AUVs patrol the site inspecting pillars as they are encountered, would not be able to arrange coordination of AUV activities and would

# of pillars	IPs only	IPs and Pillars
1	208.909	117.631
1	216.309	135.814
2	365.618	263.028
2	351.014	350.307
3	758.375	383.531
3	781.005	324.062
Avg time (s)	446.872	262.395
Avg # actions	58	23

Table 1: Plan quality for the structure inspection task. "IPs only" refers to the task undertaken without knowledge of pillars. "IPs and Pillars" shows the results when taking knowledge of pillars into account.

result in inefficient inspections. We illustrate this by comparing the behaviour of the AUVs under coordination of the planner with and without the support of the ontology.

The structure inspection missions are carried out in simulation, using a system that emulates an underwater environment and interfaces with ROS. The entire control system has been used to plan and execute missions using physical vehicles, but we have not had the opportunity to test physical missions with two AUVs together. In the test missions, the AUVs are sent to inspect different structures with various numbers of pillars. Initial inspection points are placed on the surface of the structure. Our hypothesis is that using the knowledge from the ontology will allow us to generate plans that have a shorter duration, and fewer actions, because the recognition of a specific instance of an object type gives the planner access to the best affordances to enable efficient interaction with the object. The simulation was run multiple times, first without taking into account knowledge of pillars from the ontology. In this case, all the inspection points had to be observed. Then, the ontological data was taken into account, and pillars could be inspected with the pillar-specific observation action, inspecting multiple inspection points at once.

The times taken to execute the missions are reported. The planning time on each planning cycle is limited to 10 seconds. Figure 7 shows the simulation during runtime. Using the knowledge provided by the ontology about Pillars greatly reduced the time taken to complete the mission (Table 1). Extending the functionality of the planner with new knowledge about the environment can be expected to increase the quality of the plans based on the improvement of affordances.

5 Conclusion

In this paper we describe the linkage of a planning system to an ontology within an execution framework, allowing the planner to exploit features and affordances of elements of the environment as they are identified and inferred by the ontology. As the world state is continually modified using processed sensor data to update the ontology, the executing plan is monitored for validity and replanning is invoked when it ceases to be valid. The result is a system robust to changes in the environment. We tested our system in simulation, showing that using the ontology to associate objects with affordances can result in plans with a shorter duration and fewer actions.

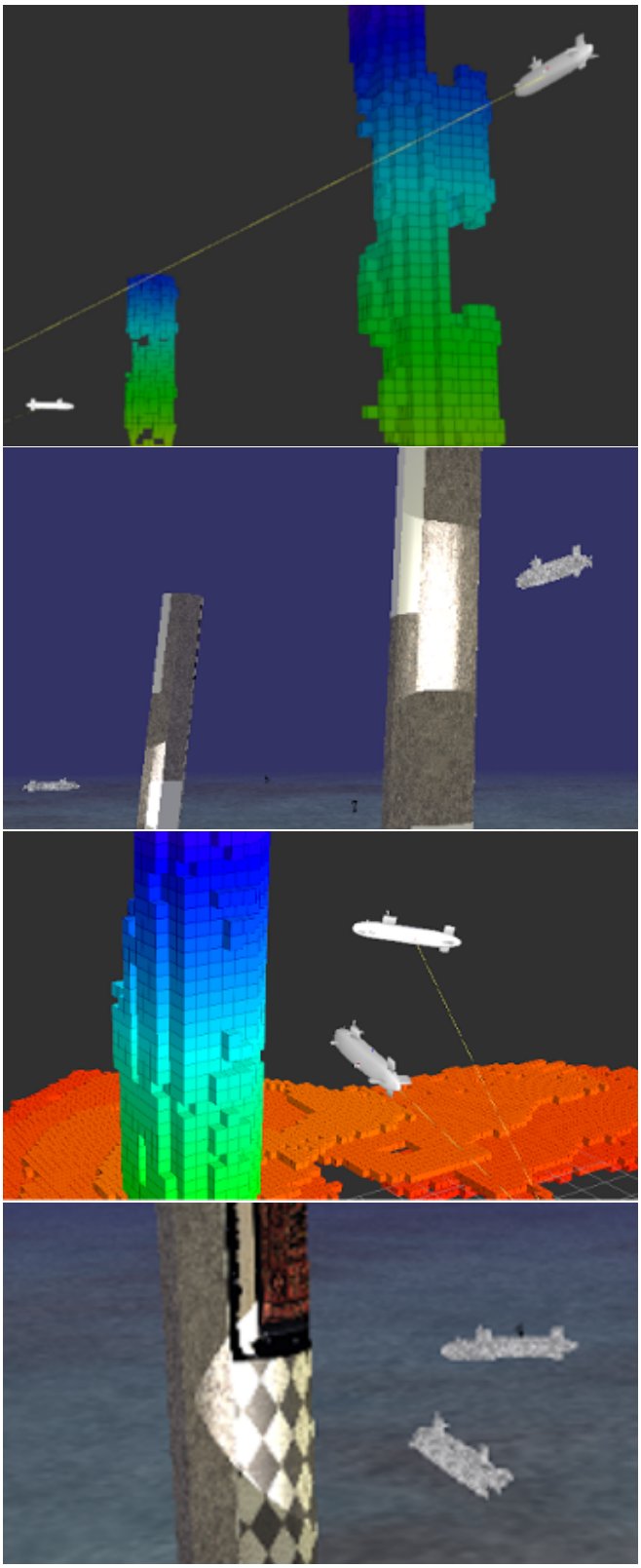

Figure 7: Images of the simulation environment, the 3D model of the AUV structure and sea-bed; and the rviz scene, the environment as detected by the AUV.

References

[Cashmore *et al.*, 2013] Michael Cashmore, Maria Fox, Tom Larkworthy, Derek Long, and Daniele Magazzeni. Planning inspection tasks for auvs. In *Proc. of the MTS/IEEE Oceans 2013 Conference, San Diego (OCEANS'13)*, 2013.

[Cashmore *et al.*, 2014] Michael Cashmore, Maria Fox, Tom Larkworthy, Derek Long, and Daniele Magazzeni. Auv mission control via temporal planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA'14)*, 2014.

[Coles *et al.*, 2010] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *Proc. of the 20rd Int. Conf. on Automated Planning and Scheduling (ICAPS'10)*, pages 42–49, 2010.

[Englot and Hover, 2010] B. Englot and F. Hover. Inspection planning for sensor coverage of 3D marine structures. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Res. (JAIR)*, 20:61–124, 2003.

[Fox *et al.*, 2012] Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based policy-learning for autonomous feature tracking. In *Proc. of the 22nd Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2012.

[Galindo *et al.*, 2008] Cipriano Galindo, Juan-Antonio Fernandez-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[Geib *et al.*, 2006] Christopher Geib, Kira Mourão, Ron Petrick, Nico Pugeault, Mark Steedman, Norbert Krueger, and Florentin Wörgötter. Object action complexes as an interface for planning and robot control. In *Proc. of the Humanoids-06 Workshop: Towards Cognitive Humanoid Robots*, 2006.

[Ghallab *et al.*, 2004] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[Gruber, 1993] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(1):199–220, 1993.

[Hernández *et al.*, 2011a] Emili Hernández, Marc Carreras, Javier Antich, Pere Ridao, and Alberto Ortiz. A topologically guided path planner for an auv using homotopy classes. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, pages 2337–2343, 2011.

[Hernández *et al.*, 2011b] Emili Hernández, Marc Carreras, and Pere Ridao. A Path Planning Algorithm for an AUV Guided with Homotopy Classes. In *Proc. 21st Int. Conf. on Automated Planning and Scheduling (ICAPS'11)*, 2011.

[Kavraki *et al.*, 1996] L. E. Kavraki, J.-C. Latombe P. Svestka, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Transactions on Robotics and Automation*, page 566580, 1996.

[Lavalle, 2006] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[McGann *et al.*, 2008] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn, and Robert S. McEwen. A deliberative architecture for auv control. In *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, pages 1049–1054, 2008.

[Petrick and Foster, 2013] Ronald P. A. Petrick and Mary Ellen Foster. Planning for social interaction in a robot bartender domain. In *Proc. of the 23rd Int. Conf. on Automated Planning and Scheduling (ICAPS'13)*, pages 389–397, 2013.

[Petrick and Gaschler, 2014] Ronald P. A. Petrick and Andre Gaschler. Extending knowledge-level contingent planning for robot task planning. In *Proc. of the ICAPS 2014 Workshop on Planning and Robotics (PlanRob)*, pages 157–165, 2014.

[Py *et al.*, 2010] Frederic Py, Kanna Rajan, and Conor McGann. A systematic agent framework for situated autonomous systems. In *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 583–590, 2010.

[Srivastava *et al.*, 2014] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'14)*, 2014.

[Tenorth *et al.*, 2010] Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. KNOWROB-MAP - knowledge-linked semantic object maps. In *Humanoids*, pages 430–435, 2010.

[Wurm *et al.*, 2010] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.