The Benefits of Explicit Ontological Knowledge-Bases for Robotic Systems

Zeyn Saigol[1], Minlue Wang[2], Bram Ridder[3], and David M Lane[1]

[1] Ocean Systems Lab, Heriot-Watt University, Edinburgh, UK
[2] Intelligent Robotics Lab, University of Birmingham, UK
[3] Department of Informatics, King's College London, London, UK

Abstract With the increasing abilities of robots comes a corresponding increase in the complexity of creating the software enabling these abilities. We present a case study of a sophisticated robotic system which uses an ontology as the central data store for all information processing. We show how this central, structured and easily human-understandable knowledge-base makes for a system that is easier to develop, understand, and modify.

1 Introduction

Robotic systems are becoming more capable, but this makes the software engineering involved harder. Systems must rely on multiple components each with a different function, often written by different people with different technical specialities, and frequently wrapping "third-party" modules developed completely independently. This is a problem that can be tackled from many different directions: clear architecture and documentation (for both the system being developed, and any third-party libraries it relies on); good communication and a clear reporting structure within the development team; following best-practice for software development; and defining clear interfaces between the different components of the system. We address this last problem. According to Brooks [2]

> The most pernicious and subtle bugs are system bugs arising from mismatched assumptions made by the authors of various components.

which illustrates the importance of the issue.

Our solution is to use a central ontological knowledge-base, shared between all components in the system. A novelty of our approach is that we attempt to store all the high-level knowledge of the robot in the knowledge-base, including observed objects, the robot's plans, and execution data such as inspection waypoints.

We consider the scenario of a mobile robot trying to find an item of interest in an unmapped area, which has general application to domains such as searching for survivors following a disaster, finding suspected bombs in sensitive locations, or performing autonomous science surveys in the oceans or on extraterrestrial planets. The complete robotic system is presented along with experimental results in [12]; the aim of this paper is to explore in more detail how the components interact using the knowledge-base, and to highlight how this has benefited the development process.

2 Ontologies Background

We consider an ontological knowledge-base to be (i) a definition of the classes of things that may exist, properties they may have and potential relationships between them, and (ii) a store of instances of these classes and relations. While our system uses the W3C standard format for ontologies, OWL [9], we expect the benefits described in this paper would accrue to users of other systems for storing structured, semantically-tagged knowledge.

Ontologies are strongly associated with the semantic web [1], as they provide an ideal common language for the exchange of data between disparate web-enabled systems. However, they were used before that by the AI community to solve exactly the kind of issues outlined in the introduction (see for example [3]). The advantages of ontologies include the re-use of domain engineering outputs, readily available editing, consistency checking and reasoning tools, and an easily human-interpretable format for storing and amalgamating knowledge.

Recently they have gained in popularity in robotics, as shown by the IEEE working group developing a common ontology for robotics and autonomous systems [10,11]. Ontologies have been used to represent domain and common-sense knowledge for household robots, in the KnowRob [13] and ORO [5] systems. They have been used as a data store for underwater robots [7,8], been integrated with probabilistic reasoning systems [4], and as a bridge between perceptual data and semantic concepts for service robots [6]. We have adopted KnowRob for our ontology implementation.

3 System and Ontology

Our test setup is an indoor robot exploring an unknown area, but with knowledge of all of the object types it may encounter. We make the simplifying assumption that all faces of all objects are rectangular and of a single colour, but note that this framework extends to any object class which can be decomposed into recognisable sub-components. Our perception system can detect the colour, size and pose of rectangular surfaces, but observing one face of an object may not uniquely determine the object's class. Further observations will restrict the space of possible classes, and the set of possible worlds the robot might be in is maintained in the knowledge-base. Based on these possible worlds, the system creates a plan to explore the area so as to find a target object (of a known class) as quickly as possible.

The system is described and evaluated in [**?**], but here we give an overview of its main software components:

knowledge-base the central ontological store containing the robot's knowledge of the world.

perception extracts rectangular faces from RGB-D point clouds.

scene-generator builds up a set of scenes, each containing a concrete instantiation of shapes which is consistent with the faces observed so far.

waypoint-generator creates view-poses to observe locations likely to contain the target, using the scenes data in the knowledge-base.

contingent-planner uses the scenes representation and waypoints to define a plan to efficiently find a target, taking into account candidate objects that may restrict the robots path, and the different possible outcome from observation actions.

observation-action-planner accounts for noisy sensor observations by monitoring, at execution time, the improvement in value of the future plan if additional observation actions are performed.

scenes-to-belief-state converts the scenes into a probabilistic map of target location, for use by the observation-action-planner.

executor executes a mission by invoking other components in turn, generating waypoints, a contingent plan, executing the plan, and then repeating these steps until the target is found.

goal-evaluator determines if the knowledge-base has converged on a single, fixed location for the target.

Other functionality (including moving the robot and building an obstacle-avoidance map using SLAM) is handled by standard ROS components.

The ontology schema backing the system is shown in Figure 1, illustrating that we have used base classes from KnowRob where possible. The environment-model parts of the ontology are shown on the left of Figure 1, and include object templates (`Shape`) and observed objects (`ObjectInScene`). Multiple candidate `Scene`s are created by applying collision constraints with other objects. Note that we do not simply attach the pose corresponding to an `ObservedFace` to a `Shape`, as there may be several instances of a `Shape`, both within and across `Scene`s.

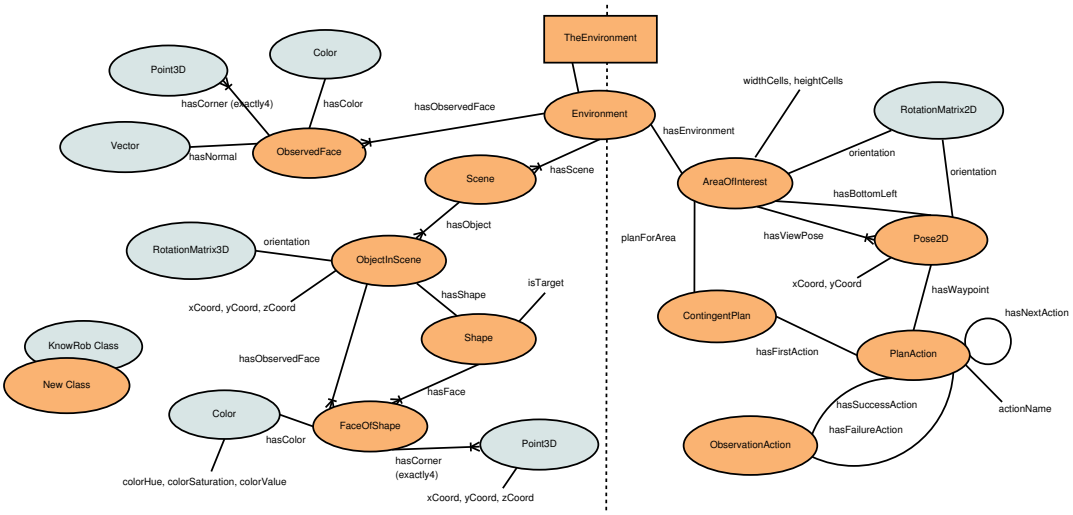


Figure 1: Ontology layout, showing environment-modelling classes to the left of the dashed line, and mission-related classes to the right.

A significant aspect of our ontology is that it also stores mission-related information, as shown on the right of Figure 1. The `AreaOfInterest` defines our search area, `Pose2D` instances are waypoints, and the branching `ContingentPlan` is stored in the ontology. This allows, for example, the waypoint-generator to store waypoints in the ontology, the contingent-planner to access them and store its output plan into the ontology, and finally the execution system to read and execute this plan.

The core knowledge-base architecture is shown in Figure 2, and consists of a custom C++ node which starts an embedded Prolog engine to run KnowRob. Some manipulation of scenes is a good fit for logical operations and runs in Prolog, but other components access the knowledge-base via a C++ interface. This interface uses SWI-Prolog's foreign language interface for fast access to the ontology (and is a more performant replacement for KnowRob's `json_prolog` module), and has a ROS service wrapper. A stub class calls either the ROS services or the native interface directly, depending on where the code is running, meaning performance-critical components can be moved inside the knowledge-base node by simply editing launch files.

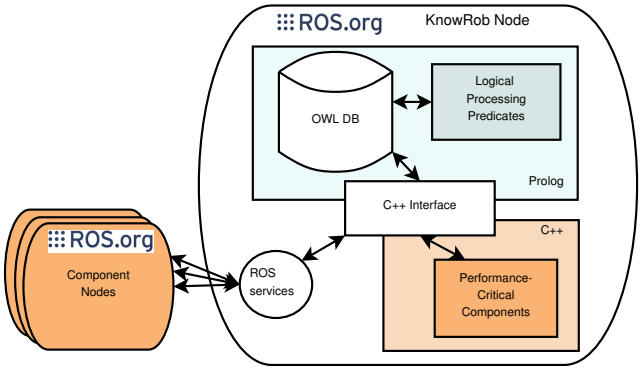
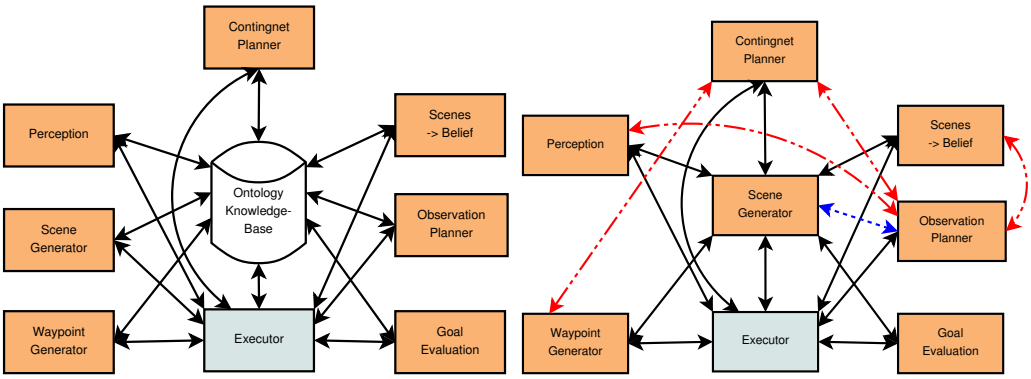


Figure 2: System architecture.

4 Evaluation

The evaluation of architectural aspects of robotic systems is extremely difficult. We cannot demonstrate that using an ontology allows our system to do things that would otherwise be impossible, as we do not claim this to be true. Instead we hope to convince the reader that our system is more robust, simpler to build, and easier to modify and maintain due to the use of a central ontological knowledge-base.

Figure 3(a) shows the inter-component dependencies of our system, and Figure 3(b) the expected dependencies a system without a central data store would have. For the non-ontology system we have assumed both the scene-generator and the contingent-planner will have their own persistent storage, whereas the waypoint-generator will pass data directly to the contingent-planner, which seems a reasonable design choice.



(a) With ontology (b) Without ontology

Figure 3: Inter-component dependencies for the complete system. (a) the system as it is implemented, (b) a possible dependency set resulting from removing the ontology. The dash-dotted red arrows indicate new dependencies, and the dotted blue arrow indicates a dependency that is no longer needed.

It turns out that for our implementation there is only a small reduction in the total number of inter-component connections compared to the non-ontology version (partly as there is one component fewer, namely the knowledge-base itself). However, the *complexity* of the interactions is much higher; instead of two fixed interactions (with the knowledge-base and the executor), components will have a varying number of interactions with unpredictable other components. Further, adding a new component may require changes to several other components, depending on which parts of the robot's knowledge the new component needs to make use of.

We present two examples of this: first, when the observation-action-planner was integrated, we only had to interface it to the knowledge-base, despite the fact it relies on knowing the plan produced by the contingent-planner, and the probability distribution over target locations produced by the scenes-to-belief-state which itself relies on the scenes representation. The second example is that we realised we needed to record the execution time of each processing component for results in [**?**]. These could only be produced by the components themselves, and were needed by the executor – but instead of needing new interfaces between the executor and each component, we could simply store the execution times in the knowledge-base.

Finally we note that having a central knowledge-base enables the executor to call each component directly. The non-ontology version in Figure 3(b) has a less clear design in two ways: it stores persistent knowledge in two different components (the scene-generator and the contingent-planner) instead of in just one place, and it passes knowledge between components*, cutting the executor out of loop*, when the waypoint-generator hands its waypoints on to the contingent-planner.

Whilst these dependencies and examples are specific to our system, we expect that all complex robotic systems with multiple modules would have similarly beneficial experiences if they were to use a centralised ontological knowledge-base.

5 Conclusions

We have outlined a robotic exploration system, and demonstrated how a centralised, semantic knowledge-base helps reduce the complexity of the interactions between components. With our ontological knowledge-base, a new developer does not need to guess at the interactions of a component: they are all invoked by the executor, perform some processing, and store any results into the knowledge-base.

Future work will include co-operation with a second robot, and we expect that having a well-defined ontology which can be easily shared (using ROS) with this second robot will make our task much easier.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
2. Brooks, F.P.: The Mythical Man-month: Essays on Software Engineering. Addison Wesley (1995)
3. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human Computer Studies 43(5-6), 907–907 (1995)
4. Hanheide, M., Gretton, C., Dearden, R., Hawes, N., Wyatt, J., Pronobis, A., Aydemir, A., Gbelbecker, M., Zender, H.: Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. In: IJCAI-11 (2011)
5. Lemaignan, S., Ros, R., Mosenlechner, L., Alami, R., Beetz, M.: Oro, a knowledge management platform for cognitive architectures in robotics. In: IROS-10 (2010)
6. Lim, G.H., Suh, I.H., Suh, H.: Ontology-based unified robot knowledge for service robots in indoor environments. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 41(3), 492–509 (May 2011)
7. Maurelli, F., Saigol, Z., Lane, D.M.: Cognitive knowledge representation under uncertainty for autonomous underwater vehicles. In: ICRA-14 Workshop on Persistent Autonomy for Underwater Robotics (2014)
8. Miguelanez, E., Patron, P., Brown, K.E., Petillot, Y.R., Lane, D.M.: Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles. IEEE Transactions on Knowledge and Data Engineering 23(5) (2011)
9. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M.: OWL 2 web ontology language: Structural specification and functional syntax. Tech. rep., W3C (2009)
10. Paull, L., Severac, G., Raffo, G., Angel, J., Boley, H., Durst, P., Gray, W., Habib, M., Nguyen, B., Ragavan, S., Sajad Saeedi, G., Sanz, R., Seto, M., Stefanovski, A., Trentini, M., Li, H.: Towards an ontology for autonomous robots. In: IROS-12 (2012)
11. Prestes, E., Carbonera, J.L., Fiorini, S.R., Jorge, V.A.M., Abel, M., Madhavan, R., Locoro, A., Goncalves, P., Barreto, M.E., Habib, M., Chibani, A., Gerard, S., Amirat, Y., Schlenoff, C.: Towards a core ontology for robotics and automation. Robotics and Autonomous Systems 61(11), 1193–1204 (2013)
12. Saigol, Z., Ridder, B., Wang, M., Dearden, R., Fox, M., Hawes, N., Lane, D.M., Long, D.: Efficient search for known objects in unknown environments using autonomous indoor robots. In: IROS-15 Workshop on Task Planning for Intelligent Robots in Service and Manufacturing (2015)
13. Tenorth, M., Beetz, M.: Knowrob: A knowledge processing infrastructure for cognition-enabled robots. International Journal of Robotics Research 32(5) (2013)