Efficient Search for Known Objects in Unknown Environments Using Autonomous Indoor Robots

Zeyn Saigol[1], Bram Ridder[2], Minlue Wang[3], Richard Dearden[3], Maria Fox[2], Nick Hawes[3], David M Lane[1], and Derek Long[2]

***Abstract*— We consider the problem of a robotic system searching for a specific object in an unmapped environment. We operate in a human environment, using a wheeled robot equipped with vision and distance sensors. The vision system is linked to a knowledge base containing object templates constructed from primitive geometric components, supporting recognition of objects. Given that some visual inputs may match several different objects, we build a multiple-hypothesis representation of the environment. These multiple hypotheses are used by a task planner to generate paths that avoid obstacles and explore areas where the target object might be. Plans also exploit behaviour to disambiguate the possible worlds in order to reduce anticipated future search effort. Replanning takes place when actions fail or when it becomes clear that the current plan will not find the target. In this case, the most recent knowledge of the environment will be used to direct exploration to the most useful parts of the search area. Experimental evaluation in simulation confirms the validity of our methods, and shows decreased time to find the target compared with a reactive system which explores the area greedily based on estimates of the expected increase in area viewed.**

I. Introduction

Robotic systems searching for a target object face several challenges. In order to search efficiently (which we will take to mean minimizing the expected time to find the target), they must identify objects in their environment, both to navigate around them and in order to determine whether the target is found, and they must decide where to search and how best to navigate to those locations given the current uncertainty about the environment.

We consider the problem of searching for an object of a known class in an unmapped human environment. An example application is factory automation robots for operation in semi-structured environments, such as warehouses with corridors formed by moveable containers.

Our system identifies inspection points that it considers likely to provide good views of the area it is searching (and therefore increase the likelihood of finding the target) and plans how to visit these points. These plans are not path plans (although planning paths is necessary to identify possible routes), but task plans, where the challenge is to order inspections to minimize the expected time to discover the target, exploiting the dual role of the inspections in searching for the target and also exploring the environment.

The performance of our approach is enhanced by two specific contributions:

- Combined conceptual and geometric information is stored in a knowledge base, using an ontology, allowing representation of the possible worlds (the current belief state) in a structured way.
- A planning system that reasons over this belief state and plans to perform sensing actions to disambiguate between possible states consistent with this belief state, thus finding the target faster.

We compare the approach with one in which the robot follows a policy of greedily searching locations that appear to offer greatest chance of discovering the target. This approach does not consider the role of observations in uncovering environmental structure or the knowledge stored in the belief state.

II. System Overview

A. Problem Statement

Our goal is to find a target object within a pre-defined search area, which we term the *area of interest* (AoI). We assume there is exactly one target in the AoI which can be identified unambiguously provided the right observations are made. The robot can make as many observations as needed, but our objective is to minimize the time taken to find the target.

Our approach is targeted at searching in environments in which objects can be described as a composite of primitives [1]. We have prior knowledge of how these primitives combine to form objects in the environment and we assume that we can identify instances of these primitives via sensor data processing. In most human environments objects can be decomposed into components [2], for example chairs are composed of legs, a base and a back support. We are exploring an environment in which our component primitives are coloured rectangular object faces, and the objects they can form are 3D geometric shapes. The approach is applicable to more general settings, requiring extension of both the ontology of objects and the description of their formation from primitives, and of the image processing we use (which is not a focus of this work).

B. Architecture

Fig. 1 shows the workflow in our system. We briefly review the components supporting this workflow, including the

*The first three authors contributed equally to this work.

[1]Ocean Systems Lab, Heriot-Watt University, Edinburgh, UK `zs@zeynsaigol.com`

[2]Department of Informatics, King's College London, London, UK `bram.ridder@gmail.com`

[3]Intelligent Robotics Lab, University of Birmingham, UK `mxw765@cs.bham.ac.uk`

This work was supported by EPSRC.

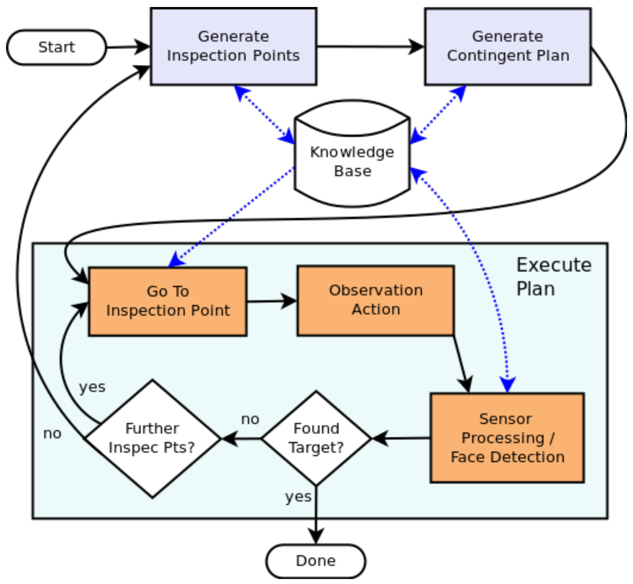


Fig. 1: Flowchart showing system operation, where the solid lines represent control flow, and dashed lines data flow.

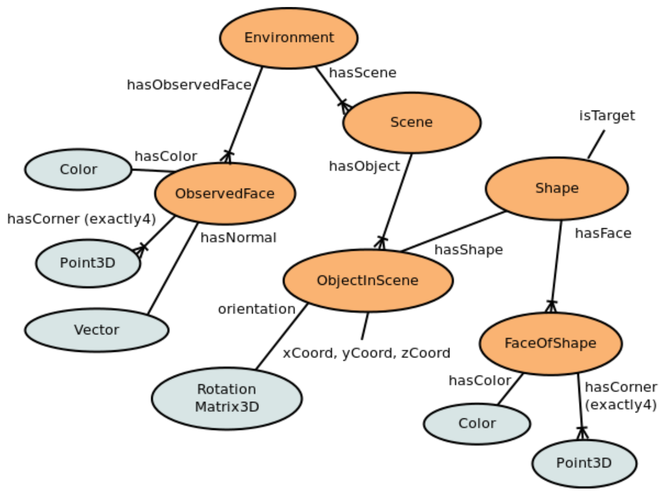


Fig. 2: Face processing sections of the ontology schema. The ontology provides a single, coherent store for key domain knowledge and execution data of the agent. Classes defined by KnowRob are shown in light-blue ovals.

knowledge base and its associated ontology, the generation of inspection points, the contingent planning and the execution of plans.

The system is implemented within ROS, using nodes to encapsulate the elements of this workflow. Note we are not addressing the SLAM problem or low-level path planning, so standard ROS components are used for these. This means the system maintains two maps, a semantic object-based map stored in the ontology, and the SLAM-generated occupancy grid used for low-level path planning and also when creating inspection points. The maps are generated from the same RGB-D data (from a simulated Microsoft Kinect), but the processing is completely independent.

1) Ontology: The domain and mission knowledge of the robot is stored in a knowledge base (KB), implemented as an ontology. Ontologies define the classes of thing that might exist and relations between them, and also store instances of these classes and their properties. Using an ontology has the advantage of supporting re-use of domain engineering outputs, readily available consistency checking and reasoning tools, and a human-interpretable format for storing and amalgamating knowledge. These advantages are increasingly being recognised by the robotics community [3]. We denote classes in our ontology with `FixedWidth` text. The KB stores the geometric shape of all object types we may encounter in the world, with each type or `Shape` being described by a set of rectangular faces, as shown in Fig. 2.

2) Sensing: The sensor processing performed by the system is the detection of object faces, which are then stored in the KB. Observed faces are converted into candidate concrete objects (with a 6D pose), and collision constraints are used to construct a set of candidate scenes, each one containing a set of possible concrete objects (see Section III). One of these candidate scenes will correspond to the true world.

We note the system does not know in advance how many objects are present in the AoI.

3) Inspection Point Generation: Following [4], we generate a set of object inspection points by first sampling a large number of waypoints uniformly over the AoI. These are then weighted according to the voxels that lie within a viewcone at each of these waypoints. Positive weights are assigned to the voxels that are occupied or unknown according to the occupancy grid map. Negative weights are assigned to the voxels that are within objects represented in the knowledge base, to prevent the robot from sensing already-found objects. The top N inspection points are then selected and stored as `Pose2D` instances in the KB. As the robot has no initial knowledge of the environment, the first set of inspection points is distributed randomly over the AoI.

In order to speed up the search, inspection point generation also takes ambiguity in the form of partially observed objects into account. For all the remaining unobserved faces of objects, we count the number of distractor objects[1] that do not have this face. The face with the maximum number is assumed to be the best face to disambiguate the target object. A large positive weight is then assigned to the voxels that are within this face.

While our inspection point generation is heuristic, using an information-gain approach is not straightforward, as areas may be completely observed according to the occupancy grid map but still contain objects that have not been identified in the KB scenes.

4) Contingent Planning: Section IV describes the contingent planning system, which uses the KB and a planner to construct a contingent plan to visit inspection points efficiently and perform observation actions. The plan checks whether there is a path that can visit all inspection points for every possible scene, and if not, additional observation

[1]A distractor object is an object that is not a target but shares some components of the target.

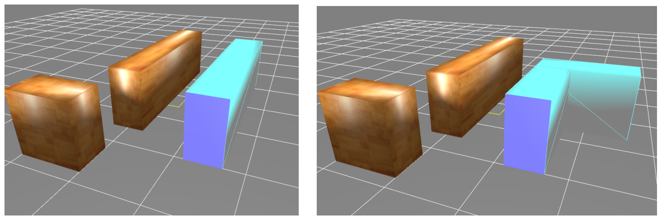


Fig. 3: Generation of two new `Scenes` from a single original `Scene` when an `ObservedFace` (shown in dark blue) matches two different `Shapes`.

actions are performed to reduce the size of the scenes. The plan is stored in the KB, enabling the other system components to easily access it.

5) Replanning: A replan consists of generating a new set of inspection points in the ontology, and then creating a new contingent plan based on these. At run time, we replan whenever it becomes clear that our current plan will not enable us to find the target. This happens if a navigation action fails or if all inspection points have been visited. An additional benefit of replanning is that we use the most recent set of scenes from the KB and the latest occupancy grid map when regenerating inspection points.

III. Scene Generation

The scene processing task is to create a representation that captures the belief state consistent with what has been observed of the objects in the environment. Our KB representation (illustrated in Fig. 2) comprises multiple candidate `Scenes`, each of which describes a possible state of the world. Observations create geometric constraints, and the set of `Scenes` is filtered to contain only those consistent with all observations so far. We represent the belief state so that all possible consistent states (`Scenes`) are included.

The inputs to scene processing are `ObservedFaces` extracted from RGB-D data. We follow the procedure of Holz et al. [5] to extract planes, making use of PCL [6], and then separate coplanar faces and fit them to rectangles.

The next step is to generate concrete `ObjectInScenes` from the observed faces. This starts with the matching of `ObservedFaces` to `Shape` faces, then the 6D pose of the resulting object is estimated using the optimal rigid-body transformation procedure [7]. Care is required to match the correct corners of the shape face and the observed face, taking into account which side of the face is visible.

The core algorithm for updating `Scenes` is shown in Algorithm 1. The key step is to generate the Cartesian product of existing scenes and new objects; Fig. 3 shows an example of one existing `Scene` becoming two `Scenes` when two `Shapes` match an `ObservedFace`. This process obviously raises a question about scaling and it remains a subject of future work to compress the belief state construction and representation by a lossy filtering of `Scenes`.

If a geometric collision is found between a new `ObjectInScene` and an existing one, an attempt is made to combine the two objects, by checking whether they have

Algorithm 1 Generation of `Scenes` following observation of a new `ObservedFace` F_o.

```
$E \leftarrow$ set of existing Scenes
$M \leftarrow$ set of ObjectInScene matches to $F_o$
$N \leftarrow \emptyset$ {set of new Scenes}
if $E = \emptyset$ then
    Add one Scene to $N$ for every member of $M$
else
    for all $S_E, o \in (E \times M)$ do
        if no collisions between $o$ and any $o_E \in M$ then
            Add $S_E \cup o$ to $N$
        else {collision between $o$ and $o_E$}
            if Shape($o$)=Shape($o_E$) pose($o$)$\approx$pose($o_E$) then
                Add $copy(S_E)$ to $N$
                if $F_o$ corresponds to different face than $F_{o_E}$ then
                    Add $F_o$ usedBy $o_E$
                end if
            else
                {no merge possible: discard candidate Scene}
            end if
        end if
    end for
    for all $S_E \in E$ do
        Remove $S_E$ from ontology
    end for
end if
```

the same `Shape` and approximately the same pose. If the objects cannot be merged, then two `Scenes` are created, one containing each object.

To help to avoid an explosion of scenes, we find the faces each `Scene` predicts the robot should be able to see, and delete the `Scene` if any of these faces are not actually observed. This is done using a graphics pipeline based on Bresenham's rasterisation algorithm and z-buffering.

The above functionality is developed with KnowRob [8], a state-of-the-art system targeted at mobile robots, and is mostly implemented in Prolog. The C++ libraries OpenCV and FCL (Flexible Collision Library) are used for two cases: for finding the singular value decomposition of a matrix (which is a step in the algorithm of [7]), and for calculating whether two objects collide. Rectangular faces from the knowledge base are converted to trimeshes for use by FCL. The C++ code accesses the ontology via SWI-Prolog's foreign language interface, meaning it runs in the same ROS node as KnowRob. This makes the system relatively efficient despite the use of diverse languages and components.

IV. Mission Planning

The task of the planning system is to create an efficient contingent plan that visits inspection points to perform sensing actions. The inspection points are provided as an input to the planning process; both inspection points and plan will be re-generated periodically during the mission, as described in Section II. The plan is contingent on the scenes that are generated by the ontology and the outcome of planned sensing actions that distinguish between alternatives.

A deterministic contingent problem is a tuple $P = \langle S, S_0, S_G, A, O, f, o \rangle$, where S is a finite set of states, $S_0 \subseteq S$ is the set of possible initial states, S_G is the set of goal states,

A is a set of actions with $A(s)$ denoting the actions in A that are applicable in the state s, and O is a set of observation tokens. An action a applicable in a state s changes the state to $s' = f(a,s)$ and results in the observation token $o(s',s) \in O$. Executions are sequences of action-observation pairs $a_0, o_0, a_1, o_1, \ldots$ and beliefs represent the sets of states that are possible. The initial belief state is $b_0 = S_0$ and, if action a is applied to belief b the resulting state is $b_a = \{s' \mid s' = f(a,s) \text{ and } s \in b\}$, while $b_a^o = \{s \mid s \in b_a \text{ and } o = o(s,a)\}$ is the belief after receiving observation token o.

An execution $\langle a_0, o_0, a_1, o_1, \ldots \rangle$ is possible in the model S if, starting from the initial belief b_0, each action a_i is applicable in the belief b_i resulting from the execution up to a_i, i.e. $a_i \in A(s)$ for all $s \in b_i$, and b_{i+1} is non-empty, where $b_{i+1} = b_a^o$ for $b = b_i$, $a = a_i$, and $o = o_i$.

The actions we model for our system are:

1) OBSERVE-FACE: perform a sensing action and branch the plan depending on whether the face is observed. One branch of the plan will respond to all the scenes that contain the face, the other branch to all the scenes that do not contain the face.
2) NAVIGATE: move the robot between waypoints. This action can only be performed if the edge between the current waypoint and the goal waypoint is traversable in *all* scenes on the current branch.
3) SENSE-VIEW-CONE: perform a sensing action from a currently occupied inspection point.

Whenever the *SENSE-VIEW-CONE* action is executed, the KB is updated as described in Section III. If all the scenes in the new belief state contain the target at the same location then we have found the target. The outcome of the *OBSERVE-FACE* action determines which branch is selected during execution.

We construct a contingent plan by compiling the contingent planning problem into a classical planning problem, using an approach due to Bonet and Geffner [9]. The compiled problem is solved using FF [10], since it supports the range of PDDL features that are required in the compiled domain model. This approach scales to larger instances than other planners we tested. However, an artifact of the compilation is to occasionally undermine the identification of helpful actions in the planner search guidance.

Whenever a new plan is requested, the planning system retrieves the belief state from the knowledge base and the set of inspection points that must be observed. It then generates a set of waypoints such that for every possible scene there is a path to visit all the inspection points. Finally, the connectivity between the waypoints is determined as well as which faces are visible from each waypoint for each scene.

For example, Fig. 4a depicts two possible scenes, one with a red wall on the left side and the other with a green wall on the right side. If we assume that *one* of these scenes must be the actual situation, then the yellow connection between *WP2* and *WP5* only exists if the former scene is the reality, while the purple connection between *WP4* and *WP5* only exists if the latter scene is reality. A solution for reaching *WP5* from the initial state of the robot (*WP0*) is depicted in Fig. 4b. Which branch is taken depends on the outcome of the sensing action. If the green wall is not detected we infer that the scene with the red wall must be the reality (reducing the belief state accordingly) and take the *FALSE* branch which navigates the robot to *WP5* via *WP2*. If the green wall is observed we infer that the red wall does not exist and navigate back and reach *WP5* via *WP4*.

This example illustrates the opportunity afforded by contingent planning: the planner can choose actions to efficiently respond to ambiguities in the belief state, since the planner can consider the contingent effects of actions and the consequences they will have in different outcomes within the belief state. This allows the planner to build efficient plans that minimize useless navigation around the environment. The advantage is small in the example, but grows for more complex scenarios, as discussed in Section V.

V. Experimental Evaluation

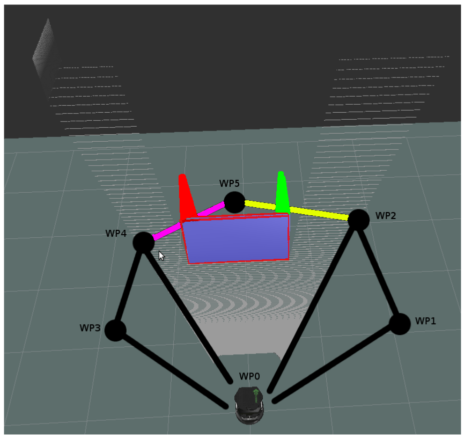
To evaluate our approach, we perform experiments in a simulated environment using the open source robot simulator Gazebo (Fig. 5). We use 8 test scenarios. In each one a number of objects including one target object are randomly positioned inside a bounded AoI ($9m \times 7m$).

We compare with a reactive execution system which attempts to explore the entire AoI greedily. The reactive execution uses the same face detector as our system to make observations, and a very similar inspection point generator. However inspection points do not take KB objects into account — no reasoning about structures is performed. The reactive system only uses the KB to determine whether the target is found. We note that our reactive comparison approach will outperform the common frontier-based exploration scheme (see for example [11]), as frequently the system will have explored all the space in the AoI without having identified all the objects.

With both systems, 5 inspection points were generated before a replan. For each mission, we limited the search time to 30 mins. If the target is found before time out, we record the number of replans, and the total time spent in execution, which includes all the compute time for planning and sensing. We ran 5 trials for each search (40 trials in total). Our scenarios are constructed as a sequence of scaled instances, with increasing numbers of objects, in order to test the scaling behaviour of our system. We hypothesise that the benefits of planning will become more significant in more complex environments, as the objects create both more sources of occlusion and also more difficulty in navigating around the environment. However, we also hypothesise that the time spent planning and reasoning will increase as the number of scenes increases.

Our system successfully finds the target in 30 of the 40 trials, taking an average of 496.7 seconds to find the target in successful trials, while the reactive system finds it in 22 trials (a subset of the 30 solved by the planning system) taking an average of 785.4 seconds.

Fig. 6 summarises the results of reactive execution and our system on the test scenarios. As can be seen in the bottom

(a) 

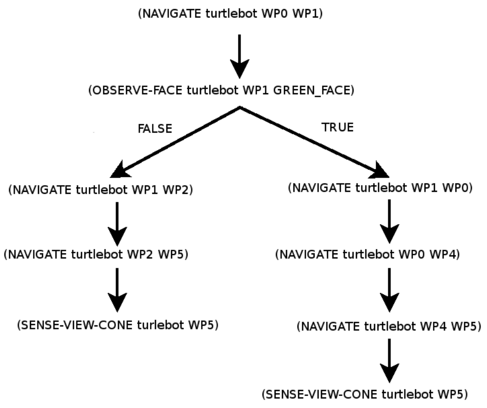
(b) 
(NAVIGATE turtlebot WP0 WP1)
(OBSERVE-FACE turtlebot WP1 GREEN_FACE)
FALSE
TRUE
(NAVIGATE turtlebot WP1 WP2)
(NAVIGATE turtlebot WP1 WP0)
(NAVIGATE turtlebot WP2 WP5)
(NAVIGATE turtlebot WP0 WP4)
(SENSE-VIEW-CONE turtlebot WP5)
(NAVIGATE turtlebot WP4 WP5)
(SENSE-VIEW-CONE turtlebot WP5)

Fig. 4: (a) Example of planning where the sensor processing has detected the blue face. (b) A plan branching on the outcome of the *OBSERVE-FACE* action.

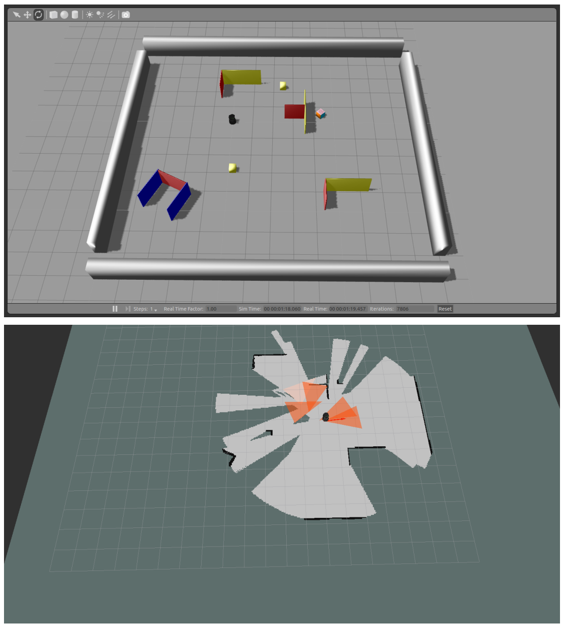


Fig. 5: The upper image shows a simulated environment, and the lower image shows an example of inspection points (red MarkerArray) generated for the robot to visit.

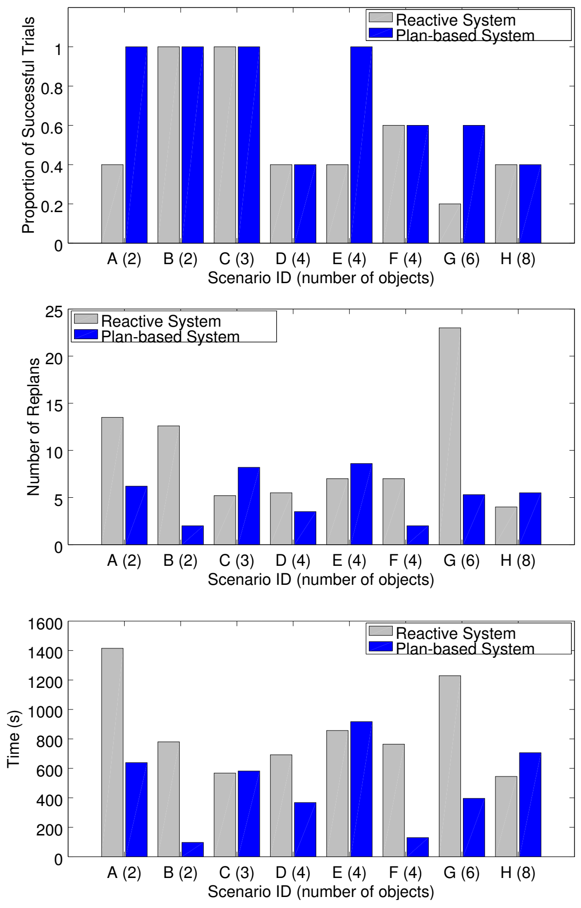


Fig. 6: Comparison between reactive and plan-based systems, with the sizes of the scenarios in terms of the number of objects they contain shown in brackets on the x-axes. For the lower two graphs, numbers are the mean of successful trials. The middle graph shows mean numbers of replans, where a replan corresponds to generating a new set of waypoints to inspect.

graph of Fig. 6, which shows average time taken to find the target, the plan-based approach finds the target faster than the reactive approach in most cases. It is important to note that the mean times are for *successful* trials — the numbers of successes in each scenario are shown in the top graph. The middle graph shows the mean numbers of replans. The fact that the plan-based approach solves some harder instances means that its mean performance is biased upwards by these instances compared with the reactive approach. Despite this, it is clear that the plan-based approach performs significantly better on these trials.

Both our system and reactive execution only find the target in 2 out of 5 trials for the world with 8 objects. The performance of the reactive system illustrates the complexity of this scenario; there are so many objects that observations made from random viewpoints are unlikely to have a non-occluded view of the target. The difficulty the plan-based system encounters is that the number of possible scenes is exponential in the number of faces for our deterministic scene generation approach. This 8-object scenario confirms our hypothesis that planning time scales up significantly as the environment becomes more complex: the mean total execution time is 706s, of which 50s is spent planning but 207s is spent on sensing and updating the scenes (with the remainder spent actually moving). This compares to an average total time of 472s for the 4-object scenarios, 23s of which is planning and 67s is perception and scene updating.

It is clear that filtering of scenes is essential to achieve a scalable approach that will handle more complex environments. A large portion of the time in the tests in the 8-object scenario is spent on scene generation for planning, rather than exploring. Although the sensing action is not the main contribution of this paper, we intend to investigate the possibility of applying pruning techniques to avoid the explosion of scenes in future.

VI. Related Work

Previous work on the problem of finding targets in unknown environments makes the optimistic assumption that unknown space is empty, and replans when sensor information contradicts this assumption [12]. Other work formulates the problem of navigating a partially-unknown environment using an A* algorithm that reasons over all the possible worlds and generates an occupancy map [13]. In this work we improve over the optimistic assumption and the A* solution by actively reasoning about all the possible worlds. The benefits of our approach are twofold: 1) the plans we create are valid for all known scenes and not for just a subset of them; 2) by exploiting knowledge of all possible worlds we can find efficient plans to explore the environment.

Previous research in scene analysis has focused on sensor processing and object identification (e.g. [14]), 3D reconstruction from 2D images (e.g. [15]), or creating CGI scenes (e.g. [16]). Our combination of the use of geometric constraints and maintaining multiple scene hypotheses is novel.

The problem of choosing a limited set of sensing locations has been addressed using submodularity [17], and using MDP methods to plan over uncertain pathways [18]. While these are relevant to our work, neither of them use a central conceptual knowledge base to enable cooperation between planning, sensing and execution monitoring systems.

A similar architecture to ours is proposed by Hanheide et al. [19], where a knowledge base is used, but not including both geometric and conceptual knowledge. Wong et al. [20] use a probabilistic model and known shapes to find occluded objects, in particular objects hidden within a container. Riazuelo et al. [21] link semantic concepts to a SLAM map in order to help locate an object.

VII. Conclusions

This paper describes a novel integration of knowledge handling and planning to direct a robot in a search for targets in unmapped environments. Our approach currently rests on several limiting assumptions, including the requirement that the environment be a bounded human structured space, that the number of such objects does not lead to an explosion in the size of our belief state and that our sensors are capable of recognising the primitives from which the objects are composed. In order to deal with ambiguity from sensor processing, our system uses a contingent planner to generate plans that can actively perform observation actions to disambiguate the candidates scenes in knowledge base. Combined with the replanning approach, our system has shown improvements compared to a greedy reactive execution in terms of success rate and execution time in finding the target.

References

[1] E. Krause, M. Zillich, T. Williams, and M. Scheutz, "Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues," in *AAAI-14*, 2014.

[2] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, 1987.

[3] L. Paull, G. Severac, G. Raffo, J. Angel, H. Boley, P. Durst, W. Gray, M. Habib, B. Nguyen, S. Ragavan, G. Sajad Saeedi, R. Sanz, M. Seto, A. Stefanovski, M. Trentini, and H. Li, "Towards an ontology for autonomous robots," in *IROS-12*, 2012.

[4] L. Kunze, K. K. Doreswamy, and N. Hawes, "Using qualitative spatial relations for indirect object search," in *ICRA-14*, 2014.

[5] D. Holz, S. Holzer, R. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *RoboCup 2011: Robot Soccer World Cup XV*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7416, pp. 306–317.

[6] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *ICRA-11*, 2011.

[7] K. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *Pattern Anal. Mach. Intell.*, 1987.

[8] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, 2013.

[9] B. Bonet and H. Geffner, "Flexible and scalable partially observable planning with linear translations," *AAAI-14*, 2014.

[10] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[12] A. T. d. A. Rui Araujo, "Exploration-based path-learning by a mobile robot on an unknown world," *Lecture Notes in Control and Information Sciences*, vol. 232, pp. 645–658, 2008.

[13] W. Hao and S. J. Julier, "Path planning in partially known environments," in *Proceedings of the IASTED International Conference*, 2011.

[14] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Anal. Mach. Intell.*, 2002.

[15] J. McGlone and J. A. Shufelt, "Projective and object space geometry for monocular building extraction," in *CVPR*, 1994.

[16] K. Xu, J. Stewart, and E. Fiume, "Constraint-based automatic placement for scene composition," in *Graphics Interface*, 2002.

[17] A. Krause and C. Guestrin, "Near-optimal observation selection using submodular functions," in *AAAI-07*, 2007.

[18] M. Kneebone and R. Dearden, "Navigation planning in probabilistic roadmaps with uncertainty," in *ICAPS-09*, 2009.

[19] M. Hanheide, C. Gretton, R. Dearden, N. Hawes, J. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, and H. Zender, "Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour," in *IJCAI-11*, 2011.

[20] L. Wong, L. Kaelbling, and T. Lozano-Perez, "Manipulation-based active search for occluded objects," in *ICRA-13*, 2013.

[21] L. Riazuelo, M. Tenorth, D. Di Marco, M. Salas, D. Galvez-Lopez, L. Mosenlechner, L. Kunze, M. Beetz, J. Tardos, L. Montano, and J. Martinez Montiel, "RoboEarth semantic mapping: A cloud enabled knowledge-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, 2015.