A Planning Approach to Active Visual Search in Large Environments

Moritz Göbelbecker
Albert-Ludwigs-Universität Freiburg, Germany

Alper Aydemir and **Andrzej Pronobis**
and **Kristoffer Sjöö** and **Patric Jensfelt**
Centre for Autonomous Systems
Royal Institute of Technology, Stockholm, Sweden

Abstract

In this paper we present a principled planner based approach to the active visual object search problem in unknown environments. We make use of a hierarchical planner that combines the strength of decision theory and heuristics. Furthermore, our object search approach leverages on the conceptual spatial knowledge in the form of object co-occurrences and semantic place categorisation. A hierarchical model for representing object locations is presented with which the planner is able to perform indirect search. Finally we present real world experiments to show the feasibility of the approach.

1 Introduction

When operating in partially unknown environments, key tasks for service robots, such as fetch-and-carry, require a robot to successfully find objects. It is evident that such a system cannot rely on the assumption that all object relevant to the current task are already present in its sensory range. It has to actively change its sensor parameters to bring the target object in its field of view. We call this problem *active visual search* (AVS).

Although researchers began working on the problem of visually finding a relatively small sized object in a large environment as early as 1976 at SRI (Garvey 1976), the issue is often overlooked in the field. A common stated reason for this is that the underlying problems such as reliable object recognition and mapping are posing hard enough challenges. However as the field furthers in its aim to build robots acting in realistic environments, this assumption needs to be relaxed.

In this work we consider the case where the environment may be completely unknown to the robot and has to be discovered first. However, the robot is given probabilistic default knowledge about the relation between objects and the occurrences of objects in difference room category following Aydemir et al. (2011). Decision making is done by a domain independent planner, making it possible to integrate search with other tasks.

Acknowledging the need to limit the search space and integrate various cues to guide the search, (Garvey 1976) proposed *indirect search*. Indirect search as a search strategy is a simple and powerful idea: it is to find another object first and then use it to facilitate finding the target object, e.g. finding a table first while looking for a phone. Tsotsos (Tsotsos 1992) approached the problem by analysing the complexity of the AVS problem and showed that it is NP-hard. Therefore we must adhere to a heuristics based solution.

There are several recent robotic systems that include high-level planning and execution monitoring subsystems. Kraft et al. (2008) and Talamadupula et al. (2010) both use deterministic planning models, the latter being able to reason about open worlds. The work by Hanheide et al. (2011) can exploit probabilistic background knowledge, but is limited environments whose structure is fully known.

Contributions

The contributions of this work are four fold. First we provide the domain adaptation of a hierarchical planner to address the AVS problem. Second we show how to combine semantic cues to guide the object search process in a more complex and larger environment then found in previous work. Third, we start with an unknown map of the environment and provide an exploration strategy which takes into account the object search task. Four, we present real world experiments searching for multiple objects in a large office environment, and show how the planner adapts the search behaviour depending of the current conditions.

Outline

The outline of this paper is as follows. First we present how the AVS problem can be formulated in a principled way using a planning approach (Section 2). Section 3 provides the motivation for and structure of various aspects of our spatial representation. Finally we showcase the feasibility of our approach in real world experiments (Section 4).

2 Planning

For a problem like AVS which entails probabilistic action outcomes and world state, the robot needs to employ a planner to generate flexible and intelligent search behaviour that trade off exploitation versus exploration. In order to guarantee optimality a POMDP planner can be used, i.e. a decision theoretic planner that can accurately trade different costs against each other and generate the optimal policy. However, this is only tractable for a complex problem like AVS when applied to very small environments. Another type of planner are the classical AI planners which requires perfect

knowledge about the environment and are thus generally not suitable for problems involving exploration.

A variation of the classical planners are the so called continual planners that interleave planning and plan monitoring in order to deal with uncertain or dynamic environments (Brenner and Nebel 2009). The basic idea behind the approach is to create an plan that *might* reach the goal and to start executing that plan. A monitoring component keeps track of the execution outcome and notifies the planner in the event of the current plan becoming invalid (either because the preconditions of an action are no longer satisfied or the plan does not reach the goal anymore). In this case, a new plan is created with the updated current state as the initial state and execution starts again. This will continue until either the monitoring component detects that the goal has been reached or no plan can be found anymore.

In this paper we will make use of a so called switching planner (Göbelbecker, Gretton, and Dearden 2011). It combines two different domain independent planners for different parts of the task: A *classical continual planner* to decide the overall strategy of the search (for which objects to search in which location) and a *decision theoretic planner* to schedule the low level observation actions using a probabilistic sensing model. Both planners use the same planning model and are tightly integrated.

We first give a brief description of the switching planner, focusing on the use of the planner and changes we made to the original implementation. We will also present the domain modeling for the planner, and give further details on various aspects of knowledge that the planner makes use of.

Switching Planner

Continual Planner (CP) The continual planner is based on an extended SAS^+formalism (Bäckström and Nebel 1995). Because our knowledge of the world and the effects of our actions are uncertain we associate a *success probability* $\mathrm{p}(a)$ with every action a. In contrast to more expressive models like MDPs or even POMDPs, actions do not have multiple possible outcomes, they just can succeed with probability $\mathrm{p}(a)$ or fail with probability of $1 - \mathrm{p}(a)$.

The goal of the planner is then to find a plan π that reaches the goal with a low cost. In classical planning the cost function is usually either the number of actions in a plan or the sum of all action's costs. In some cases one could use use a *self-loop determinisation* (Keyder and Geffner 2008) of the problem and compile probabilities into (expected) action costs. As we will use the probabilistic actions to model starting state distributions, this is however not a suitable approach for our problem.

We therefore chose a function that resembles the expected reward adjusted to our restricted planning model. With $\mathrm{p}(\pi) = \prod_{a\in\pi} \mathrm{p}(a)$ as the plans total success probability and $\mathrm{c}(\pi) = \sum_{a\in\pi} \mathrm{c}(a)$ as the total costs, we get for the optimal plan π^*:

$$\pi^* = \operatorname*{argmin}_{\pi} \mathrm{c}(\pi) + R(1 - \mathrm{p}(\pi))$$

This corresponds to maximising the expected reward if we assume that for each action a the robot will incur a reward of $-\mathrm{c}(a)$, transition to a sink state with probability $\mathrm{p}(a) - 1$ and is given the reward R on reaching the goal.

In order to find good plans with respect to this criterion, we modified the classical planner Fast Downward (Helmert 2006) to support the simple probabilities used in our model. For each planning state s we explicitly store its costs $\mathrm{c}(s)$ and probability $\mathrm{p}(s)$ and define the state's g-value as $g(s) = \mathrm{c}(s)+R(1-\mathrm{p}(s))$. Application of an action a transitions to a state s' with $c(s') = c(s) + \mathrm{c}(a)$ and $\mathrm{p}(s') = \mathrm{p}(s)\,\mathrm{p}(a)$. We then use weighted A^*-search with a h^{max}heuristic (Bonet and Geffner 2001) which was similarly modified to take the probabilities into account.

Assumptions: The defining feature of an exploration problem is that the world's state is uncertain. As we cannot model probabilistic planning states in our model, we use *assumptive actions* to model the probabilistic dependencies between facts, that allow the planner to construct parts of the initial state on the fly, and which allows us to map the spatial concepts to the planning problem in an easy way.

Definition 1 *Given a conditional probability $P(X = x|Y)$ with $Y = Y_0 = y_0 \wedge \ldots \wedge Y_n = y_n$, the assumptive action on X $\mathcal{A}^{X=x|Y}$ is defined as follows:*

$$\begin{aligned}\mathrm{pre}(\mathcal{A}^{X=x|Y}) &= \{Y_0 = y_y, \ldots, Y_n = y_n,\ \mathrm{def}(X) = \bot\}\\ \mathrm{eff}(\mathcal{A}^{X=x|Y}) &= \{X = x,\ \mathrm{def}(X) = \top\}\\ \mathrm{p}(\mathcal{A}^{X=x|Y}) &= P(X = x|Y)\end{aligned}$$

Using assumptions, the structure of the initial state can be modelled in PDDL. For example, given the default probability of an object existing in a room of category C, $P(ObjectAt\mathcal{L}^C)$ we can create the following operator:

```
(:action object-in-room
   :parameters (?cl - class ?r - room
                ?c - category)
   :probability (P(ObjectAtL^?c))
   :precondition (= (category ?r) ?c)
   :effect (obj-exists ?cl in ?r))
```

Decision Theoretic (DT) Planner When the continual planner reaches a sensing action (e.g. *search location1 for a object2*), we create a POMDP that only contains the parts of the state that are relevant for that subproblem with. This planner can only use MOVE and PROCESSVIEWCONE actions explained below. The DT planner operates in a closed-loop manner, sending actions to be executed and receiving observations from the system. Once the DT planner either confirms or rejects a hypothesis, it returns control back to the continual planner, which treats the outcome of the DT session like the outcome of any other action.

Domain Modeling

We need to discretize the involved spaces (object location, spatial model and actions) to make a planner approach applicable to the AVS problem.

Representing space For the purposes of obstacle avoidance, navigation and sensing action calculation, the search space Ψ is represented as a 3D metric map. Ψ is discretized

into i volumetric cells so that $\Psi = c_0 ... c_i$. Each cell represents the occupancy with the attributes OCCUPIED, FREE or UNKNOWN as well as the probability of target object's centre being in that cell.

However, further abstraction is needed to achieve reliable and fast plan calculation as the number of cells can be high. For this purpose we employ a topological representation of Ψ called *place map*, see Fig 1(a). In the place map, the world is represented by a finite number of basic spatial entities called *places* created at equal intervals as the robot moves. Places are connected using paths which are discovered by traversing the space between places. Together, places and paths represent the topology of the environment. This abstraction is also useful for a planner since metric space would result in a largely intractable planning state space.

The places in the place map are further segmented into rooms. In the case of indoor environments, rooms are usually separated by doors or other narrow openings. Thus, we propose to use a door detector and perform reasoning about the segmentation of space into rooms based on the doorway hypotheses. We use a template-based door detection algorithm which matches a door template to each acquired laser scan. This creates door hypotheses which are further verified by the robot passing through a narrow opening.

In addition, unexplored space is represented in the place map using hypothetical places called *placeholders* defined in the boundary between free and unknown space in the metric map.

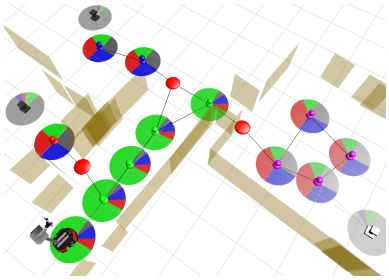
We represent object locations not in metric coordinates but in relation to other known objects or rooms to achieve further abstraction. The search space is considered to be divided into *locations* $\mathcal{L}$. A location is either a *room* $\mathcal{R}$ or a *related space*. Related spaces are regions connected with a *landmark object* o, either *in* or *on* the landmark (see (Aydemir et al. 2011) for more details). The related space "in" o is termed $\mathcal{I}_o$ and the space "on" o $\mathcal{O}_o$.

In order to reason about locations that are not yet known, we allow to refer to locations based on their *category* (for rooms) or *class* (for spaces related to objects). We refer to those locations as $\mathcal{L}^C$, with C specifying the category or class.

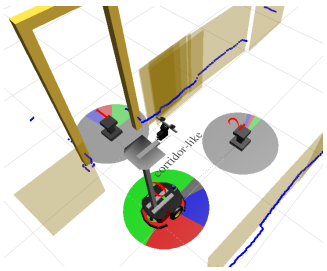
Modeling actions The planner has access to three physical actions: MOVE can be used to move to a place or placeholder, CREATEVIEWCONES creates sensing actions for an *object label* in *relation* to a specified *location*, PROCESSVIEWCONE executes a sensing action. Finally, the virtual SEARCHFOROBJECT action triggers the decision theoretic planner.

Virtual objects There are two aspects of exploration in the planning task: we are searching for an (at that moment) unknown object, which may include the search for support objects as an intermediate step. But the planner may also need to consider the utility of exploring its environment in order to find new rooms in which finding the goal object is more likely.

Because the planners we use employ the closed world assumption, adding new objects as part of the plan is impossible. We therefore add a set of *virtual objects* to the planning problem that can be instantiated by the planner as required by the plan. This approach will fail for plans that require finding more objects than pre-allocated, but this is not a problem in practice. The monitoring component tries to match new (real) objects to virtual objects that occur in the plan. This allows us to deliver the correct observations to the DT planner and avoid unnecessary replanning.



(a)



(b)

Figure 1: (a) A place map with several places and 3 detected doors shown as red. (b) Shows two placeholders with different probabilities for turning into new rooms: one of them is behind a door hypothesis therefore having a higher probability of leading into a new room. Colours on circular discs indicates the probability of room categories as in a pie chart: i.e. the bigger the colour is the higher the probability. Here green is *corridor*, red is *kitchen* and blue is *office*.

Probabilistic spatial knowledge The planner makes use of the following probabilistic spatial knowledge in order to generate sensible plans:

- $P_{category}(room_i)$ defines the distribution over room categories that the robot has a model for, for a given room.
- $P_{category}(placeholder_i)$ represents the probability distribution of a placeholder turning into a new room of a certain category upon exploration.
- $P(ObjectAt\mathcal{L})$ gives the probability of an object o being at location $\mathcal{L}$.
- $P(ObjectAt\mathcal{L}^C)$ gives the probability of an object o being at an abstract location $\mathcal{L}^C$. Those values represent *default knowledge* and are not updated during the task.

3 Spatial Representation

Conceptual Map All higher level inference is performed in the so called conceptual map which is represented by a graphical model. It integrates the conceptual knowledge

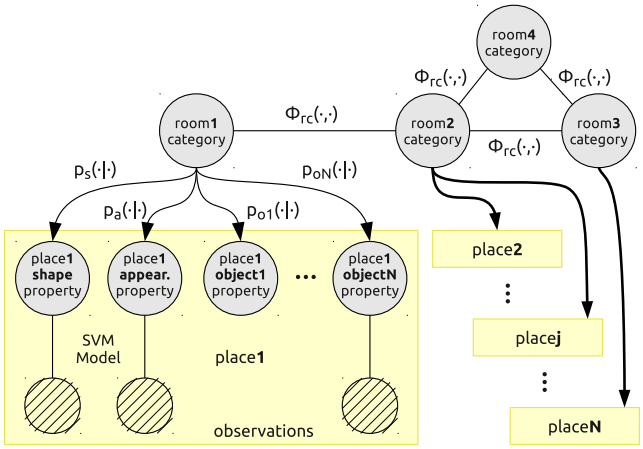


Figure 2: Schematic image of chain graph

(food items are typically found in kitchens) with instance knowledge (the rice package is in room4). We model this in a *chain graph* (Lauritzen and Richardson 2002), whose structure is adapted online according to the state of underlying topological map. Chain graphs provide a natural generalisation of directed (Bayesian Networks) and undirected (Markov Random Fields) graphical models, allowing us to model both "directed" causal as well as "undirected" symmetric or associative relations.

The structure of the chain graph model is presented in Fig. 2. Each discrete place is represented by a set of random variables connected to variables representing semantic category of a room. Moreover, the room category variables are connected by undirected links to one another according to the topology of the environment. The potential functions $\phi_{rc}(\cdot,\cdot)$ represent the type knowledge about the connectivity of rooms of certain semantic categories.

To compute $P_{category}(room_i)$ each place is described by a set of properties such as size, shape and appearance of space. These are are based on sensory information as proposed in (Pronobis et al. 2010). We extend this work by also including presence of a certain number of instances of objects as observed from each place as a property (due to space -limitations we refer to (Pronobis and Jensfelt 2011) for more details). This way object presence or absence in a room also affects affects room category. The property variables can be connected to observations of features extracted directly from the sensory input. Finally, the functions $p_s(\cdot|\cdot)$, $p_a(\cdot|\cdot)$, $p_{o_i}(\cdot|\cdot)$ utilise the common sense knowledge about object, spatial property and room category co-occurrence to allow for reasoning about other properties and room categories.

For planning, the chain graph is the sole source of belief-state information. In the chain graph, belief updates are event-driven. For example, if an appearance property, or object detection, alters the probability of a relation, inference proceeds to propagate the consequences throughout the graph. In our work, the underlying inference is approximate, and uses the fast Loopy Belief Propagation (Mooij 2010) procedure.

Object existence probabilities

To compute the $P(ObjectAt\mathcal{L})$ value used in active visual search in this paper, objects are considered to be occurring:

1. independently in different locations $\mathcal{L}$
2. independently of other objects in the same location
3. as Poisson processes over cells $c_0 ... c_i$ per location $\mathcal{L}$

In other words, each location has the possibility of containing, independently of all other locations, a number n_c of objects of a class c with probability

$$P(n_c = k) = \frac{\lambda^{\mathrm{k}}_{\mathcal{L},c} e^{-\lambda_{\mathcal{L},c}}}{\mathrm{k!}} \quad (1)$$

where $\lambda_{\mathcal{L},c}$ is the expected number of objects of class c in the location $\mathcal{L}$. The probability of *at least one* object in a location is

$$P(n_c > 0) = 1 - P(n_c = 0) = 1 - e^{-\lambda_{i,c}} \quad (2)$$

Because of the independence assumptions, the λ values for a location and all its subordinate locations can simply be added together to obtain the distribution of the number of objects of that class occurring in that whole hierarchy.

Exploration In addition to making inferences about explored space, the conceptual map can provide predictions about unexplored space. To this end, we extend the graph by including the existence of placeholders. For each placeholder a set of probabilities is generated that the placeholder will lead to a room of a certain category.

This process is repeated for each placeholder and consists of three steps. In the first step, a set of hypotheses about the structure of the unexplored space is generated. In case of our implementation, we evaluate 6 hypotheses: (1) placeholder does not lead to new places, (2) placeholder leads to new places which do not lead to a new room, (3) placeholder leads to places that lead to a single new room (4) placeholder leads to places that lead a room which is further connected to another room, (5) placeholder leads to a single new room directly, and (6) placeholder leads to a new room directly which leads to another room. In the second step, the hypothesised rooms are added to the chain graph just like regular rooms and inference about their categories is performed. Then, the probability of any of the hypothesised rooms being of a certain category is obtained. Finally, this probability is multiplied by the likelihood of occurrence of each of the hypothesised worlds estimated based on the amount of open space behind the placeholder and the proximity of gateways. A simple example is shown in Fig. 1(b)

4 Experiments

Experiments were carried out on a Pioneer III wheeled robot, equipped with a Hokuyo URG laser scanner, and a camera mounted at 1.4 m above the floor. Experiments took place in 12x8 m environment with 3 different rooms, *kitchen*, *office1*, *office2* connected by a corridor. The robot had models of all objects it searches for before each search run. 3 different objects (*cerealbox*, *stapler* and *whiteboardmarkers*) were used during experiments. The BLORT framework was used to detect objects (Mörwald et al. 2010).

To highlight the flexibility of the planning framework evaluated the system with 6 different starting positions and tasked with finding different objects in an unknown environment. Each sub-figure in Fig. 3 shows the trajectory of the robot. The colour coded trajectory indicates the room category as perceived by the robot: red is kitchen, green is corridor and blue is office. The two green arrows denote the current position and the start position of the robot.

In the following we give a brief explanation for what happened in the different runs.

- Fig. 3(a) Starts: *corridor*, Target: *cerealbox* in *kitchen*
  The robot starts by exploring the *corridor*. The robot finds a doorway on its left and the placeholder behind it has a higher probability of yielding into a kitchen and the robot enters *office1*. As the robot acquires new observations the CP's kitchen assumption is violated. The robot returns to exploring the corridor until it finds the kitchen door. Here the CP's assumptions are validated and the robot searches this room. The DT planner plans a strategy of first finding a table and then the target object on it. After finding a table, the robot generates view cones for the $\mathcal{O}_{table,cornflakes}$ location. The cerealbox object is found.
- Fig. 3(b) Starts: *office2*, Target: *cerealbox* in *kitchen*
  Unsatisfied with the current room's category, the CP commits to the assumption that exploring placeholders in the corridor will result in a room with category kitchen. The rest proceeds as in Fig. 3(a).
- Fig. 3(c) Starts: *corridor* Target: *cerealbox* in *kitchen*
  The robot explores until it finds *office2*. Upon entry the robot categorises *office2* as kitchen but after further exploration, *office2* is categorised correctly. The robot switches back to exploration and since the kitchen door is closed, it passes kitchen and finds *office1*. Not satisfied with *office1*, the robot gives up since all possible plans success probability are smaller than a given threshold value.
- Fig. 3(d) Starts: *office1* Target:*stapler* in *office2*
  After failing to find the object in *office1* the robot notices the open door, but finding that it is kitchen-like decides not to search the kitchen room. This time the *stapler* object is found in *office2*
- Fig. 3(e) Starts: *kitchen* Target: *cerealbox* in *kitchen*
  As before it tries locating a table, but in this case all table objects have been eliminated beforehand; failing to detect a table the robot switches to looking for a counter. Finding no counter either, it finally goes out in the corridor to look for another kitchen and upon failing that, gives up.
- Fig. 3(f) Starts: *corridor* Target: *whiteboardmarker* in *office1*. The robot is started in the corridor and driven to the kitchen by a joystick; thus in this case the environment is largely explored already when the planner is activated and asked to find a *whiteboardmarker* object. The part of the corridor leading to *office2* has been blocked. The robot immediately finds its way to *office1* and launches a search which results in a successful detection of the target object.

In the following, we describe the planning decisions in more detail for a run similar to the one described in Fig. 3(a), with the main difference being that the cereals could not be found in the end due to a false negative detection.

The first plan, with the robot starting out in the middle of the corridor, looks as follows:

ASSUME-LEADS-TO-ROOM place1 kitchen
ASSUME-OBJ-EXISTS table IN new-room1 kitchen
ASSUME-OBJ-EXISTS cerealbox ON new-object1 table kitchen
MOVE place1
CREATEVIEWCONES table IN new-room1
SEARCHFOROBJECT table IN new-room1 new-object1
CREATEVIEWCONES cerealbox ON new-object1
SEARCHFOROBJECT cerealbox ON new-object1 new-object2
REPORTPOSITION new-object2

Here we see several virtual objects being introduced: The first action assumes that *place1* leads to a new room *new-room1* with category kitchen. The next two assumptions hypothesise that a table exists in the room and that cornflakes exist on that table. The rest of the plan is rather straightforward: create view cones and search for the table, then create view cones and search for the cereal box.

After following the corridor, the robot does find the office, and returns to the corridor to explore into the other direction. It finally finds a room with a high likelihood of being a kitchen.

ASSUME-CATEGORY room3 kitchen
ASSUME-OBJ-EXISTS table IN room3 kitchen
ASSUME-OBJ-EXISTS cerealbox ON new-object1 table kitchen
MOVE place18
MOVE place16
CREATEVIEWCONES table IN room3
SEARCHFOROBJECT table IN room3 new-object1
CREATEVIEWCONES cerealbox ON new-object1
SEARCHFOROBJECT cerealbox ON new-object1 new-object2

The new plan looks similar to the first one, except that we do not assume the existence of a new room but the category of an existing one. Also, the robot cannot start creating view cones immediately because a precondition of the CREATEVIEWCONES action is that the room must be fully explored, which involves visiting all placeholders in the room.

After view cones are created, the decision theoretic planner is invoked. We used a relatively simple sensing model, with a false negative probability of 0.2 and a false positive probability of 0.05 – these are educated guesses, though. The DT planner starts moving around and processing view cones until it eventually detects a table and returns to the continual planner. At this point the probability of the room being a kitchen is so high, that it considered to be certain by the system. With lots of the initial uncertainty removed, the final plan is straightforward:

ASSUME-OBJ-EXISTS cerealbox ON object1 table kitchen
CREATEVIEWCONES cerealbox ON object1
SEARCHFOROBJECT cerealbox ON object1 new-object2
REPORTPOSITION new-object2

During the run, the continual planner created 14 plans in total, taking 0.2 – 0.5 seconds per plan on average. The DT planner was called twice, and took about 0.5 – 2 seconds per

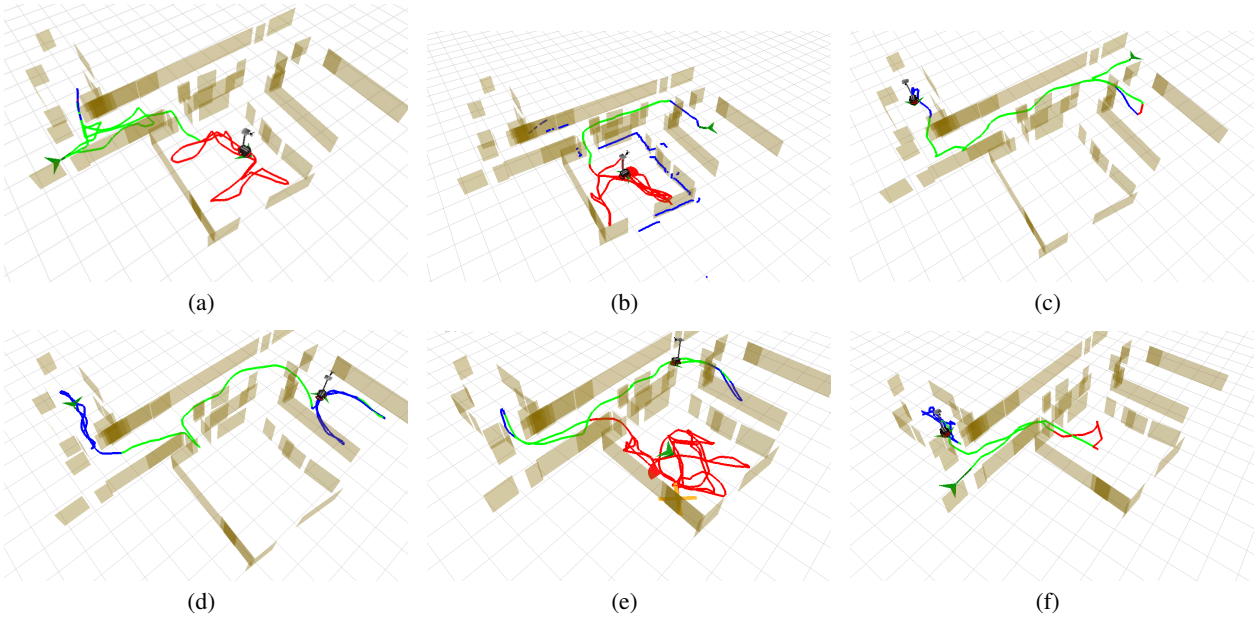


Figure 3: Trajectories taken by the robot in multiple experiments

action it executed.

5 Conclusions and Future Work

In this paper we have presented a planning approach to the active object search. We made use of a switching planner, combing a classical continual planner with a decision theoretic planner. We provide a model for the planning domain appropriate for the planner and show by experimental results that the system is able to search for objects in a real world office environment making use of both low level sensor percepts and high level conceptual and semantic information.

Our main goal for future work is to extend the planning and monitoring system to handle open world situations in a more principled way. Moreover, investigating how probabilities can be incorporated into other classical heuristics may be of interest.

References

Aydemir, A.; Sjöö, K.; Folkesson, J.; and Jensfelt, P. 2011. Search in the real world: Active visual object search based on spatial relations. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Comp. Intell.* 11(4):625–655.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5 – 33.

Brenner, M., and Nebel, B. 2009. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems* 19(3):297–331.

Garvey, T. D. 1976. Perceptual strategies for purposive vision. Technical Report 117, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.

Göbelbecker, M.; Gretton, C.; and Dearden, R. 2011. A switching planner for combined task and observation planning. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI-11)*.

Hanheide, M.; Gretton, C.; Dearden, R. W.; Hawes, N. A.; Wyatt, J. L.; Pronobis, A.; Aydemir, A.; Göbelbecker, M.; and Zender, H. 2011. Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In *IJCAI*.

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

Keyder, E., and Geffner, H. 2008. The hmdpp planner for planning with probabilities. In *Sixth International Planning Competition at ICAPS'08*.

Kraft, D.; Başeski, E.; Popović, M.; Batog, A. M.; Kjær-Nielsen, A.; Krüger, N.; Petrick, R.; Geib, C.; Pugeault, N.; Steedman, M.; Asfour, T.; Dillmann, R.; Kalkan, S.; Wörgötter, F.; Hommel, B.; Detry, R.; and Piater, J. 2008. Exploration and planning in a three-level cognitive architecture. In *CogSys*.

Lauritzen, S. L., and Richardson, T. S. 2002. Chain graph models and their causal interpretations. *J. Roy. Statistical Society, Series B* 64(3):321–348.

Mooij, J. M. 2010. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *J. Mach. Learn. Res.* 11:2169–2173.

Mörwald, T.; Prankl, J.; Richtsfeld, A.; Zillich, M.; and Vincze, M. 2010. BLORT - The blocks world robotic vision toolbox. In *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation at ICRA 2010*.

Pronobis, A., and Jensfelt, P. 2011. Hierarchical multi-modal place categorization. In *submitted to ECMR'11*.

Pronobis, A.; Mozos, O. M.; Caputo, B.; and Jensfelt, P. 2010. Multi-modal semantic place classification. *The International Journal of Robotics Research (IJRR), Special Issue on Robotic Vision* 29(2-3):298–320.

Talamadupula, K.; Benton, J.; Kambhampati, S.; Schermerhorn, P.; and Scheutz, M. 2010. Planning for human-robot teaming in open worlds. *ACM Trans. Intell. Syst. Technol.* 1:14:1–14:24.

Tsotsos, J. K. 1992. On the relative complexity of active vs. passive visual search. *International Journal of Computer Vision* 7(2):127–141.