Book Title
Book Editors
IOS Press, 2006

A Reactive Approach for Object Finding in Real World Environments [1]

Abdelbaki BOUGUERRA
Mobile Robotics Lab. University of Örebro ; Sweden
Email: abdelbaki.bouguerra@tech.oru.se

Abstract. In this paper we propose an approach to handle requests of finding objects in real world environments by mobile robots. The proposed approach checks candidate objects based on the likelihood they constitute an answer to the requests in a reactive way. As a result, run-time perceived objects are handled "on the fly" without extra cost. We present the theoretical concepts of the proposed approach, and describe the experiments we run to validate it.

Keywords. Active Search, Mobile Robots, Description Matching

1. Introduction

In [1], AI conditional planning was used to recover from failures resulting from ambiguities in anchoring an object with a specific description. Basically, this kind of failures occur when the robot's sensors detect several objects that fit the description of the object to anchor (please refer to [2] for details about the anchoring problem). In this paper we present a robust approach that can be used by mobile robots to find objects that fit a specific description by actively checking the features of candidate objects. As a result, the proposed approach can be used to recover from anchoring failures involving ambiguities.

Mobile robots face many challenges while acting in real world environments. Mainly because of the high dynamics of the environment, and lack of complete information there. We opted for a reactive approach to deal with the problem of missing information and used likelihood of objects fitting the desired description to select which candidate object to check first. This results in a robust handling of newly discovered objects at run-time without extra overhead, as opposed to techniques using planning [1]. However, being greedy in selecting a candidate, our approach does not guarantee an optimal strategy of actively observing the candidate objects.

We believe that the proposed approach performs better than using a planning technique for two reasons. First, a planner has to have access to a model of the world to be able to generate (optimal) plans that specify how to check the properties of candidate objects. Moreover, when observation actions have several outcomes, planning becomes quickly intractable for problems involving a high number of candidate objects. Second, because real environments are dynamic, plans become quickly out of date which dictates

[1] This work has been supported by the Swedish KK foundation.

replanning. As replanning produces a new plan, the same problem might occur again implying (re)replanning. In the worst case, the robot will spend most of its time replanning in response to perceiving new objects.

This paper is organized as follows: the next section reviews some of the literature related to our research work. In section 3 we specify how queries about objects are handled and how candidate objects are generated. Then, we present the algorithms used to actively infer which candidate object is an answer to the query. Before concluding we describe the experiments that we run to validate the proposed approach.

2. Related Work

The work presented here constitutes an alternative to the work reported in [1,3], where AI conditional planning was used to disambiguate objects, to recover from anchoring failures, under restricting assumptions: mainly ignoring the fact that new candidate objects might be perceived at run-time. The proposed approach bears some resemblance to active sensing strategies [4,5], where the aim is to collect information to reduce the uncertainty about the state of the mobile robot, such as self-localization [6], by selecting the action that minimizes the entropy of the next state. In our approach, the focus is on finding objects that fit a high-level description. There are also active approaches that try to solve low-level issues such as scene analysis and object recognition using single sensing modalities [7]. In comparison, different sensing modalities can be used together to infer the values of the properties of candidate objects within the proposed approach.

Partially Observable Markov Decision Processes (POMDPs) [8] are a powerful tool for acting in uncertain environments, but they are not able to scale up to solve problems involving a high number of objects. Moreover, POMDPs need a model of the environment to generate optimal policies. As for AI planning, handling newly discovered objects at run-time would involve recomputing the policy from scratch.

The natural language system Ludwig [9] is designed to answer natural language queries about objects of a scene (vision-based). Its main focus is the semantics of visual entities contained in the scene. Our focus is on acting to acquire more information rather than scene analysis. In fact the approach we propose can sit on top of a system such as Ludwig and use its services to compute values of object properties.

3. Handling Queries

We are interested in scenarios where a mobile robot, acting in an unstructured environment, gets requests (*queries*) to look for an object that fits a specific description, either for the purpose of anchoring or to satisfy users requests about objects of interest. The description specifies the properties that an object must satisfy to be an an answer to the query. A query is specified using the following syntax:

$$\texttt{find(object(}?x\texttt{)):(and } \langle f_1(?x) = v_1 \rangle, \langle f_2(?x) = v_2 \rangle, ..., \langle f_n(?x) = v_n \rangle \texttt{)}$$

Each $\langle f_i(?x) = v_i \rangle$ specifies a feature and its value, when the value v_i is omitted, it is assumed to be `True`. For instance, if we want to find a cup that contains tea, we write:

```
find(object(?x)):(and <shape(?x) = cup>,<contains(?x) = tea>).
```

Features can be either complex or simple. Simple features are properties that can be checked in the real world using observation actions. An example of a simple feature is the color of an object. Complex features, on the other hand, are combinations of other features (that can be simple or complex). Checking that a cup contains tea involves checking tow simple features (1)- the cup is not empty, and (2)- the content is tea. Thus the feature `contains` is a complex one.

In this work, complex features can be represented either as a conjunction or a disjunction of other features. A conjunctive complex feature holds if all its sub-features hold, whereas a disjunctive complex feature holds if at least one of its sub-features holds.

3.1. Candidate Objects

To generate the set of candidate objects that would constitute an answer to the query, a monitoring process uses the percepts detected by the onboard sensors to compute a representation of the outside world that comprises all detected objects and their observed properties. The features specified in the query are then matched against the properties of each detected object, resulting in three possible outcomes:

- The object does not match the description. This occurs when a feature of the checked object has a different value than the one specified in the query.
- The object is a complete match of the desired object. This happens when all the features of the query are satisfied by the properties of the checked object.
- The object is a partial match i.e. there is at least a feature whose value can not be asserted from the available properties of the checked object. An example of a partially-matching object for the previous query is a cup whose content has not been determined yet.

If an object is a complete match, it is returned as an answer to the query. In case no complete match was found, partially matching objects are considered as candidates, and therefore, their undecided features are checked.

4. Active Query Handling

In this section we detail how the mobile robot actively checks whether a candidate object is an answer to the query through checking its undecided features. The mobile robot has a repertoire of observation action templates that it can use to check simple features of candidate objects. An observation-action template specifies a sequence of robot actions to execute as a prerequisite to infer the value of the associated feature. Determining the value of complex features is done through determining the value of their sub-features.

Example 1 The following template specifies the robot actions to be executed in order to determine the value of the simple feature `smell-of(?obj)`:

`smell-of(?obj : object)`{ robot-actions: `(approach(?obj);smell(?obj))` }

4.1. Feature Checking

The algorithm "Check-Feature" in figure 1 implements the process responsible of actively checking features in the real world. Checking a simple feature f_i amounts to instantiating the associated observation action template and launching its execution by the robot (step 3) (notice that this blocks the current process "Check-Feature"). Upon finishing the execution of the observation action, the process is awaken so it computes the new state of the world (step 4) used to evaluate the truth value of the feature (step 5). The result of evaluation can be one of three values $\{0, 1, u\}$. Value u indicates that the truth value of the feature is not known (u for undecided). This can arise when the observation action has failed to observe the object (due to occlusion, for instance). Truth value 1 (resp. 0) indicates that the feature holds (resp. does not hold) in the world state.

If the feature f_i is complex, the process selects one of its best sub-features f_{ij} (step 8) (according to a selection criterion) for further checking (step 9). The evaluation result of the sub-feature f_{ij} is then used to evaluate the truth value of the parent feature f_i according to table1.

Process Active-Inference($candidates$)
1. **if** $candidates = \{\}$ return *failure* **endif**
2. $c \leftarrow$ best-candidate($candidates$)
3. **if** $undec(c) = \{\}$ return c **endif**
4. $f \leftarrow$ select-best-feature($undec(c)$)
5. $result \leftarrow$ Check-Feature(f_i)
6. **if** $result = 1$
7. $\quad sat(c) \leftarrow sat(c) \cup \{f_i\}$
8. $\quad undec(c) \leftarrow undec(c) - \{f_i\}$
9. **endif**
10. **if** $result = 0$
11. $\quad candidates \leftarrow candidates - \{c\}$
12. **endif**
13. $candidates \leftarrow candidates \cup newcand$
14. Active-Inference($candidates$)

END

Process Check-Feature(f_i)
1. **if** simple(f_i)
2. $action \leftarrow$ instantiate-action(f_i)
3. Execute($action$)
4. $ws \leftarrow$ new-world-state()
5. $result \leftarrow$ evaluate(f_i,ws)
6. return $result$
7. **else**
8. $f_{ij} \leftarrow$ best-sub-feature(f_i)
9. $result \leftarrow$ Check-Feature(f_{ij})
10. $truth \leftarrow$ evaluate-truth($f_i, result$)
11. return $truth$
12. **endif**

END

Figure 1. Processes of active checking of candidate objects features

4.2. Active Inference

The aim of active inference is to try to find the object that fully matches the query, and eliminate all those candidate objects that turn out not to fit the description of the desired object at run-time. The process of active searching for the fully-matching candidate is implemented by the algorithm "Active-Inference" shown in figure 1. Each candidate object c is associated with two sets $undec(c)$ and $sat(c)$, where $undec(c)$ contains the undecided features of object c, and $sat(c)$ stores the features of the query that where satisfied by the properties of c.

The process starts by checking that there is at least one candidate object, otherwise it returns *failure* to report that he query has no answer. This kind of failure triggers the

Table 1. Truth-value evaluation for complex feature f_i

case	type(f_i)	truth(f_{ij})	truth(fi)
1	OR	0	$\begin{cases} 0 : \forall k \neq j\text{: truth}(f_{ik}) = 0 \\ u : \exists k \neq j\text{: truth}(f_{ik}) = u \end{cases}$
2		1	1
3		u	u
4	AND	0	0
5		1	$\begin{cases} 1 : \forall k \neq j\text{: truth}(f_{ik}) = 1 \\ u : \exists k \neq j\text{: truth}(f_{ik}) = u \end{cases}$
6		u	u

generation of an exploration task to look for candidate objects in a non-visited map sector (room, corridor,..).

When the set of candidate objects is not empty, the process selects one candidate object that is judged as the best candidate which is returned as the answer to the query in case it is a complete match. This is equivalent to checking that the set of undecided features of the candidate is empty (step 3). Should the candidate be a partial match i.e. its $undec$ set is not empty, the process selects the best feature out of those undecided ones for further checking by the process "Check-feature".

As outlined in the previous sub-section, the result of checking the best feature can be one of three values $\{0, 1, u\}$. Value 1 means that the feature holds in the real world, thus the process removes the feature from $undec(c)$ (step 8) and adds it to $sat(c)$ (step 7). If on the other hand the feature does not hold in the real world state i.e. the result is 0, the process considers object c as a mismatch and therefore drops it from the set of candidate objects (step 11). When the feature checking process returns u (meaning that the truth value of the feature is unknown in the real world state), the process need not do anything, since the feature should be checked further at a later stage.

Finally, the process updates the set of candidate objects by adding any new candidate objects observed while checking the feature, and recursively performs the same operations on the updated set of candidates (step 14). Note that computing the set of new candidates is the task of the monitoring process as discussed earlier for the initial set of candidate objects.

4.3. Selection Criteria

To select the best candidate object as well as the best feature and sub-feature to check, one can use different criteria such as the euclidean distance between the current location of the robot and the location where the robot should move to observe a feature. However using distance as a measure of selection ignores how likely an object is the best match. Thus, we opted for using the probability of a candidate object matching the description of the query as a selection measure. The object with the highest probability is considered to be the best candidate. Please notice that one can also use a utility-based criterion to select candidate objects where the expected cost of checking an object is to be combined with its expected utility. Because of space limitations we only describe the probability criterion.

To simplify the computation of the probability $P(c)$ of a candidate object c, fitting the description of the query, we make the assumption that all complex features are independent.

$$P(c) = \prod_{f_i \in undec(c)} P(f_i) \tag{1}$$

Simple features probability distributions are given by the user. Computing the probability of a complex feature f_i, on the other hand, depends on its type (conjunctive or disjunctive) as well as the probabilities of all its n sub-features f_{ik} $_{(k=1,\ldots,n)}$ i.e.

$$P(f_i) = \begin{cases} \prod_{k=1,\ldots,n} (P(f_{ik})) & \text{if } f_i \text{ is conjunctive} \\ 1 - \prod_{k=1,\ldots,n} (1 - P(f_{ik})) & \text{if } f_i \text{ is disjunctive} \end{cases} \tag{2}$$

Equation (2) is used to determine the best feature and sub-feature to actively check in the real world.

Example 2 Figure 2 shows a scenario where the robot tries to find a marked object (query: `find(object(?x)):(<marked(?x)>)`). The feature `marked` is a disjunctive complex feature whose sub-features are instances of the simple feature `observed-mark(?x,?loc)` which is true if a mark is observed on object `?x` from location `?loc`. Say that the robot has instantiated feature `marked(?x)` for objects `box-1` and `gb-1` as follows:
`marked(box-1)=(or` $\langle f_1 \rangle, \langle f_2 \rangle, \langle f_3 \rangle, \langle f_4 \rangle$`)`, where f_i = `observed-mark(box-1,`l_i`)`
`marked(gb-1)=(or` $\langle f_5 \rangle, \langle f_6 \rangle$`)`, where f_i = `observed-mark(gb-1,`l_i`)`

Assuming that $P(f_i = T) = P(f_i = F) = \frac{1}{2}$ for all simple features f_i, then using Eqs. (1) and (2) we have:
$P($`box-1`$) = 1 - (1 - P(f_1))(1 - P(f_2))(1 - P(f_3))(1 - P(f_4)) = \frac{15}{16}$
$P($`gb-1`$) = 1 - (1 - P(f_5))(1 - P(f_6)) = \frac{3}{4}$

This shows that object `box-1` is more likely to be marked then object `gb-1`. Consequently object `box-1` is the best candidate object.

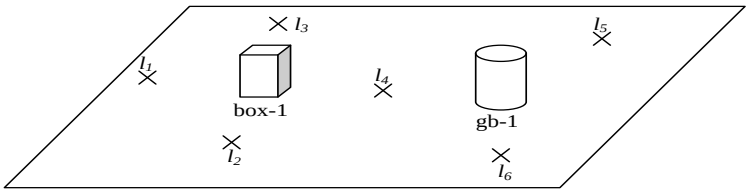

Figure 2. A scenario where the robot is looking for a marked object

5. Experiments

The active inference approach described in this paper has been implemented onboard a Magellan Pro mobile robot running the ThinkingCap behavior-based control architecture

[10]. Two sensing modalities were used, a pan-tilt CCD camera, and a a commercial electronic nose (Cyranose Sciences Inc. 2000). We validated the proposed approach by running several experiments for two scenarios.

5.1. Scenarios

The first series of experiments we run, involved only the vision system. The robot was asked to find a marked object in a room that contains several scattered objects with different shapes and colors. Figure 3 shows an experiment where the robot is asked to find a marked gas-bottle. At the beginning, only one gas bottle is perceived (gb-1), the robot determines three observation locations to check whether gb-1 is marked. As the robot moves to the first observation location, it perceives the other two gas-bottles (gb-2 and gb-3) with no mark on them. When at the new observation location, the robot checks gb-1 and perceives no mark on it. The robot, then, updates the set of candidate objects, to add gb-2 and gb-3, and computes their probabilities. At this stage, the probability of gb-1 has dropped and the two new objects have equal probability. Object gb-2 is randomly selected as the best candidate to be checked for the mark from a new location. The same process continues until the robot detects that gb-3 is an answer(fifth observation). The chart graph of figure 3 shows the evolution of the probabilities of the three objects, where the X-axis represents the observation action number. The experiment was repeated several times with different random configurations and occlusions of candidate objects. We used up to 5 candidate objects, and fixed the number of observation locations for each new candidate at 3 that are determined at run-time around the object. I ist worth noting, that the success of the approach ranged from 60% to 90%, and all failures of identifying the correct object where due to the vision system. This happened, mostly, because the vision system did not detect a mark on an object that has a mark, or the vision system failed to detect a candidate object.

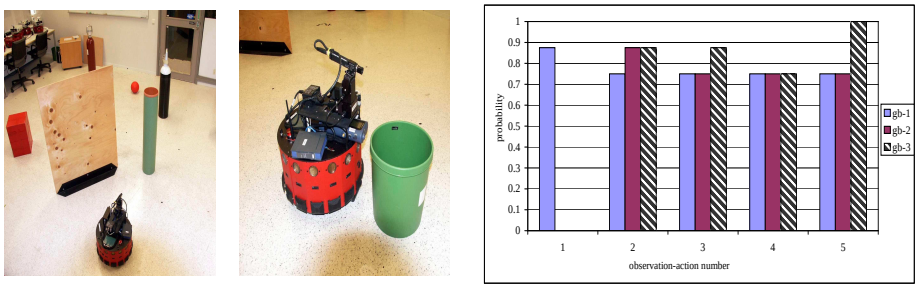

Figure 3. Experiments. (left) looking for a marked gas-bottle, (middle) smelling a garbage can, (right) probability of each of the three gas-bottles.

The second series of experiments involved using both modalities of sensing: vision and olfaction. The experiments consisted of finding an object that has the shape of a garbage can or a cup and that has the smell of ethanol (see Figure 3 (middle)). All candidate objects where first acquired using vision, and then the electronic nose was used to smell their contents until finding the one that contained ethanol. It is worth noting that we used different configurations of the layout of the objects and make some of them not visible from the first time to test how run-time perceived objects are considered. We also considered the case where the probability distribution of the simple feature of smell

is conditioned on the shape of objects (it is more probable that cups smell alcohol than garbage cans do). As a result, candidate objects that have the shape of a cup where always selected first for smelling. As in the first series of experiments, most failures where caused by the vision system not detecting candidate objects.

6. Conclusion

We have presented in this paper a reactive approach to search for objects that fit a specific description. Being reactive the approach presents the advantage of handling runtime perceived objects without any extra overhead. In comparison, a solution based on planning would have to replan to take into consideration the newly discovered candidate objects. However our approach does not guarantee to find the right object in an optimal way because of its greediness. Moreover we do not handle relations between objects as it is done in [1].

The aim of this work was to provide an alternative solution to recover from failures resulting from ambiguities in anchoring objects. We think that this approach can be used for other purposes where looking for objects is the task to be achieved by the mobile robot. Regarding future work, we will try to relax the independence assumption of features and use Bayesian networks to represent complex features. We also intend to make a comparison between the performances of using different selection criteria.

References

[1] M. Broxvall, S. Coradeschi, L. Karlsson, and A. Saffiotti. Recovery planning for ambiguous cases in perceptual anchoring. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence*, pages 1254–1260, 2005.

[2] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

[3] M. Broxvall, L. Karlsson, and A. Saffiotti. Steps toward detecting and recovering from perceptual failures. In *Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems*, pages 793–800, 2004.

[4] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[5] J. Porta, B. Terwijn, and B. Kr. Efficient entropy-based action selection for appearance- based robot localization. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 2842– 2847, 2003.

[6] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization by entropy minimization. In *Proc. of the 2nd Euromicro Workshop on Advanced Mobile Robots*, pages 155–162, 1997.

[7] T. Arbel and P. Ferrie F. On the sequential accumulation of evidence. *Int. Journal of Computer Vision*, 43(3):205–230, 2001.

[8] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[9] I. Horswill. Visual architecture and cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):277–292, 1997.

[10] A. Saffiotti, K. Konolige, and E. H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2):481–526, 1995.