Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems.
Amsterdam, The Netherlands, March 2004 (to appear).

Steps Toward Detecting and Recovering from Perceptual Failures

Mathias Broxvall, Lars Karlsson, Alessandro Saffiotti
Center for Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University
SE-701 82 Örebro, Sweden
http://www.aass.oru.se

Abstract. An important requirement for autonomous systems is the ability to detect and recover from exceptional situations such as failures in observations. In this paper we investigate how traditional AI planning techniques can be used to reason about observations and to recover from these situations. In this first step we concentrate on failures in perceptual anchoring. We illustrate our approach by showing experiments run on a mobile robot equipped with a color camera.

1 Introduction

One of the main requirements for an autonomous system is the ability to cope with exceptional situations in the execution of its assigned tasks. One approach to face this requirement is to endow the system with the ability to use knowledge-based techniques to reason about the flow of execution, detect anomalies, and automatically generate a contingency plan for recovery. Most existing systems (e.g., [8, 3, 11, 16, 17]) tend to focus on the *external* state of the world, comparing the observed state with the expected one. Discrepancies can originate in the failure of actions performed by the robot as well as in exogenous events. There is however another common cause of problems in the execution of robot plans, which is more related to the *internal* state of the robot: failures in perception, including the inability to acquire the perceptual data needed to perform the desired actions and the incorrectness of these data. The ability of the robot to detect perceptual failures and to recover from them is pivotal to its providing autonomous and robust operation.

Several works in the field have addressed the problem of planning for perceptual actions. Perception planning has been studied as a means for gathering better visual information [14, 1], for achieving safer landmark-based navigation [15, 9], and for performing tasks that involve sensing actions [10, 13]. None of these works, however, deal with the problem of failures in the perceptual actions and of the automatic recovery from these failures.

In this paper, we propose to use AI planning techniques to automatically generate a plan to recover from failures in the perceptual processes. We focus here on one specific type of perceptual process: *perceptual anchoring*. Perceptual anchoring is the process of creating and maintaining the right correspondence between the symbols used by a symbolic planner to denote objects in the world and the perceptual data in the sensori-motoric system that refer to the same objects. This process may fail if the uncertainty in the sensor data becomes too high to allow the robot to unambiguously associate these data to a given symbol. In this paper, we show how we can model the anchoring process in a planning formalism, and use a planner to generate a sequence of actions to recover from a failure due the accumulation of uncertainty. In this context, the main contributions of this paper are:

1. We isolate one of the causes of perceptual failure in robot plan execution; and
2. We give an example of how the robot can recover autonomously from these failures using knowledge-based planning.

In the next section, we give a brief reminder of perceptual anchoring. In section 3 we discuss some ways in which anchoring could fail and some possible recovery actions. In section 4 we show how the anchoring process can be modeled in a planner and a recovery plan generated automatically. Finally, we present an actual experiment run on a mobile robot. This experiment is admittedly simple. The main purpose of this paper is to open a new research direction in the use of knowledge-based planning in robotics, showing how it can be used to deal with autonomous recovery from errors. Our first results indicate that this direction is promising, but more work will be needed to further explore this possibility.

2 Perceptual Anchoring

Autonomous systems embedded in the physical world typically incorporate two different types of processes: high-level cognitive processes, that perform abstract reasoning and generate plans for actions; and sensory-motoric processes, that observe the physical world and act on it. These processes have different ways to refer to physical objects in the environment. Cognitive processes typically (although not necessarily) use symbols to denote objects, like "cup-22". Sensory-motoric processes typically operate from sensor data that originate from observing these objects, like a region in a segmented image. If the overall system has to successfully perform its tasks, it needs to make sure that these processes "talk about" the same physical objects. This has been defined as the *anchoring problem* [5], illustrated in Fig. 1:

> Anchoring is the process of creating and maintaining the correspondence between symbols and sensor data that refer to the same physical objects.

In our work, we use the computational framework for anchoring defined in [4]. In that framework, the symbol-data correspondence for a specific object is represented by a data structure called an **anchor**. An anchor includes pointers to the symbol and sensor data being connected, together with a set of properties useful to re-identify the object, e.g., its color and position. These properties can also be used as input to the control routines.

Consider for concreteness a mobile robot equipped with a vision system and with a symbolic planner. Suppose that the planner has generated the action 'GoNear(cup-22)', where the symbol 'cup-22' denotes an object described in the planner as 'a green cup on the table'. The 'GoNear' operator is implemented by a sensori-motor loop that controls the robot using the position parameters extracted from a region in the camera image. In order to execute the 'GoNear(cup-22)' action, the robot must make sure that the image region used in the control loop is exactly the one generated by observing the object that the planner calls 'cup-22'. Thus, the robot uses a functionality called **Find** to link the symbol 'cup-22' to a region in the image that matches the description 'a green cup on the table'. The output of Find is an anchor

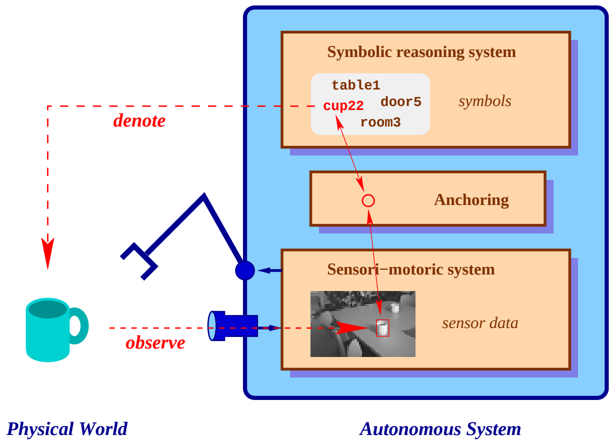

Figure 1: The anchoring problem.

that contains, among other properties, the (x, y) position of the cup, which is used by the 'GoNear' routine. While the robot is moving, a functionality called **Track** is used to update this position using new perceptual data. Should the cup go temporarily out of view, e.g., because it is occluded by another object, the **Reacquire** functionality would be called to update the anchor as soon as the cup is in view again. More details about perceptual anchoring and its use to perform actions in a robot can be found in [4, 6, 5].

3 Detecting Potential Failures in Anchoring

Perceptual anchoring can be affected by many sources of uncertainty, ranging from the intrinsic vagueness of symbols like "green" to the errors and imprecisions in the observed properties of objects. This uncertainty can lead to different types of failures in the anchoring process, such as failing to find/reacquire an object or reacquiring the wrong object. Our approach to dealing with these problems is to explicitly model the sources of uncertainty in the anchoring process and to use AI techniques to reason about them. This will increase our chances to detect potential failures and to recover from them.

One of these sources of uncertainty stems from predicting the properties of an object when it is re-observed after some time. For mobile robots, this uncertainty is mainly due to the accumulation of odometric errors as well as to errors in the estimate of the state of a moving object. Both factors may induce uncertainty in the expected position of the object relative to the robot. An effect of this uncertainty is that similar objects at nearby locations become more difficult to discriminate after a long motion, which can lead to errors in reacquiring an object.

Another source of uncertainty is due to varying observation conditions. For instance, the estimated color, size and position of an object in an image may change when observing the same object from different viewpoints or under different lighting conditions. In the scenarios we have run, the apparent changes of positions of objects caused by miscalculating the distance to those objects from different observation points is the main source of errors. We will therefore concentrate on this type of uncertainty and show how it can be modeled in order to detect and recover from situations where uncertainty creates ambiguity.

To illustrate ambiguities caused by both kinds of uncertainties, we will consider a scenario inspired by our on-going work on a fire-rescue robot, where we want a robot to perform some sensing actions on a gas bottle. A schematic view of the setup can be seen in Figure 2.

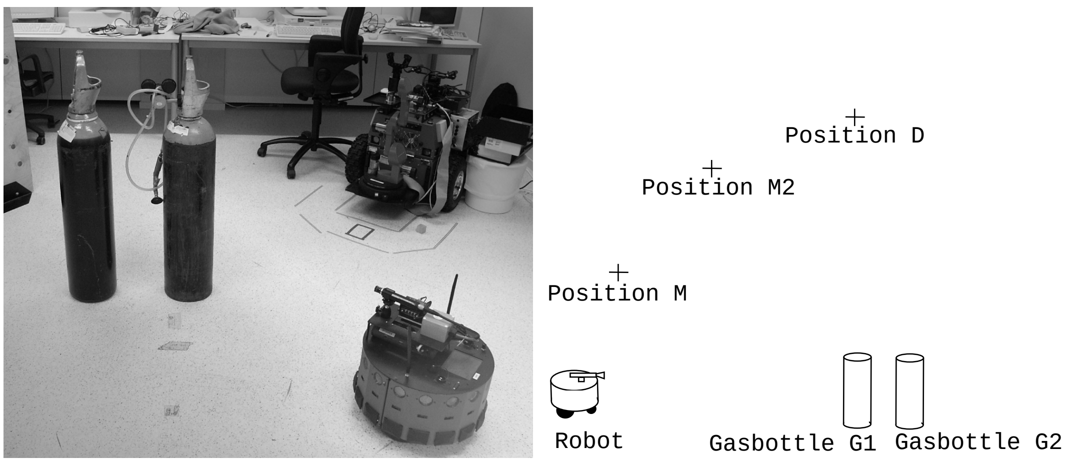

Figure 2: The initial setup of the test scenario

Initially the robot only receives percepts for one gas bottle approximately 2.5m in front of it, and anchors a newly created symbol 'gb-1' to these percepts. Since the robot starts at $(0, 0)$ it knows that 'gb-1' is approximately at coordinates $(2.5, 0.0)$. The robot then moves to a new observation point D and turns to face these coordinates. From here it receives percepts from two gas bottles. Since there are small errors in positioning (a few centimeters) and large errors in the relative positions of objects (a few decimeters) it receives percepts indicating gas bottles at $(2.3, 0.1)$ and $(2.7, 0.0)$. It has a choice of which of the two percepts to anchor to 'gb-1'. To simply pick the one closest to the expected position gives us no warning that there is a large possibility of reacquiring the wrong object and no means to recover.

To deal with this issue we use two functions $\mathrm{pos}(o \approx p)$ and $\mathrm{nec}(o \approx p)$ giving the degrees of *possibility* resp. *necessity* [7] for anchoring object o to percept p. The possibility function represents how well o matches p by a real number between 0 and 1, and can be defined in a domain specific way. For our scenario we have defined it as a fuzzy conjunction of functions comparing attributes (eg. position, color, shape etc.) of the object and percepts. By modeling the sources of error we can adjust the fuzziness of these functions to allow for larger deviations when the predicted position of the object has a high uncertainty.

The necessity function reflects to what extent a given percept is the only one matching an object. It is defined as $\mathrm{nec}(o \approx p) \doteq 1 - \mathrm{pos}(o \not\approx p)$, where $\mathrm{pos}(o \not\approx p)$ is computed as the maximum of the possibilities for anchoring o to the other percepts available, or to no percept at all. By computing the necessity of anchoring we can easily detect a number of situations in which there is a risk of failure and take appropriate actions. In the example above the necessities of matching the original object with any of the two gas bottles would be low even though the individual possibilities are high. This indicates a risk for failures.

For a human user the correct solution to deal with this anchoring failure is easy: by going back to a place close to the original position, where we had a good estimate of the position of the object, we can reacquire it again. Next, we would move toward the desired observation point in small steps and constantly reobserve the object. By incrementally observing the object as we are moving we get updated estimates of its position and can easily track it all the way to the destination point. In the next section we shall see how this solution can be generated automatically by modeling the uncertainties of the anchoring system in a system for plan generation.

4 Planning and Recovery

The symbolic level of the robotic system we use for planning and recovery is based on the planner PTLplan [12]. It is a progressive conditional planner that supports reasoning about incomplete information, sensing, and uncertainty both in probabilistic and possibilistic terms.

Being a symbolic planner, PTLplan uses a state based representation, where actions are defined in terms of what conditions should hold before and what will hold after the action. Typical examples of actions could be to pick up or put down an object, to move from one part of a room to another or to another room, and to check if a door is open or closed. A planning problem consists of a description of the initial situation, a set of actions, and a goal formula, and PTLplan finds plans by adding one action at a time, in different combinations, until a situation is reached where the goal is sufficiently likely to hold. By using heuristics expressed in logical formulas, the efficiency of the planner can be greatly enhanced.

In order to use PTLplan to solve problems related to anchoring and ambiguity, we need to provide it with the initial situation and the predicates needed to describe it, and specifications of the actions that the robot can execute. These need to model relevant aspects of the external world as well as of the anchoring and action execution systems. Once the model is in place, PTLplan can automatically generate plans that can then be executed by the robot. For the rest of this section, we focus on the predicates and actions, which include:

- Topological information about the world, giving places ($p_1, \ldots, p_n$) and connections.
- The robots position (ROBOT-AT=p_1) and actions to move between places (GOTO-POS(p_i)).
- Objects currently in the anchoring database (OBJECT(g_1),OBJECT(b_2),...) and their properties (GASBOTTLE(g_1),COLOR(g_1,RED),...).
- Sensing actions (OBSERVE(g_1)) and meta information relating to our ability to reacquire objects under different circumstances.

The first three parts above are quite standard in AI planning and we will rather focus on the fourth part. The purpose of this part is to represent in the planner the meta information of objects used in the anchoring process.

Two kinds of information are needed: from what position the object we are attempting to reacquire was last anchored, and given that, how likely are we to reacquire the object at the current position. The first kind is encoded by a function LAST-SEEN-FROM(o_i)=p_j mapping each object to the position it was last observed from. The second kind is encoded as a predicate CAN-REACQUIRE$(o, \text{pos}_i, \text{pos}_j)$ signifying that we estimate that o can be reacquired from pos_i if it was last seen from pos_j.

The CAN-REACQUIRE predicate depends on two factors: how different the viewpoints are at the two positions pos_i and pos_j, which will affect our position estimates for observed objects; and whether any similar objects are nearby. The combination of large position uncertainty and nearby similar objects will make anchoring difficult. Note that although we don't know in advance what percepts of similar objects we will actually receive when trying to reacquire o from position pos_im we can consider the objects that we know of beforehand and assume that for each such object o' visible from pos_i we will receive a percept matching o'. This gives us the following expected neccessity of reacquiring o:

$$\text{predict-reacquire}(o, \text{pos}_i, \text{pos}_j) = 1 - \max_{o' \neq o} \text{compare}(o, o', \text{pos}_i, \text{pos}_j)$$

where compare$(o, o', \text{pos}_i, \text{pos}_j)$ performs a fuzzy match on o and a percept p' actually computed from nearby object o' and returns a value in $[0, 1]$. The match function is selected in the context that we are currently at pos_i and that o was last observed at pos_j. If the final computed value (i.e. 1 minus the best matching) exceeds a threshold C for conflict given to the anchoring system then we add $\text{CAN-REACQUIRE}(o, \text{pos}_i, \text{pos}_j)$ to the initial situation given to the planner. This represents a high expected necessity degree nec$(o \approx p)$ (as described in section 3) when the anchoring system eventually will try to reacquire o. The complexity of computing one specific instance of CAN-REACQUIRE is linear on the number of objects.

The action $\text{OBSERVE}(o, \text{pos}_i, \text{pos}_j)$ is then defined to have the preconditions:

$$\text{ROBOT-AT} = \text{pos}_i \;\wedge\; \text{LAST-SEEN-FROM}(o) = \text{pos}_j \;\wedge\; \text{CAN-REACQUIRE}(o, \text{pos}_i, \text{pos}_j)$$

and the effect: $\text{LAST-SEEN-FROM}(o) = \text{pos}_i$. That is, in order to reacquire an object, it must be “reacquirable” from the current position pos_i given that it was last seen at pos_j. Thus, the planner will generate plans where the robot only tries to reacquire objects in situations where the conditions for reacquisition are predicted to be good. The result of reacquiring is that we register a new “last-seen-from” position for the object. Note that REACQUIRE can be made to have other kinds of effects as well, e.g. providing information about properties of o, but the above representation suffices for the purposes of this paper.

5 Integrating Planning, Sensing and Action

To test the concepts discussed in this paper we have built a system running on a Magellan Pro robot which integrates an anchoring system and the planner described above. This allows us to autonomously deliberate and act on objects in order to fulfill some goal. The system consists of the following parts:

- A controller allowing us to perform movements and use odometry for positioning.
- A simple vision system serving percepts generated from an onboard camera. The system runs onboard the robot and is segmenting camera images using standard algorithms, extracting information such as shapes and colors at approximately 1 frame/sec.
- An executor which translates high level plans into control commands.
- An anchoring module which continuously updates a database of real-world objects and their sensed attributes. This system also implements the ideas from section 3 and can interrupt currently executing actions/plans.
- The planner and the world model described in this section.

In our system the planner is used to generate a plan from the initial situation and the result is given to the execution and anchoring modules. As the plan is executed the anchoring module can discover potential failures by examining the computed necessity values and possibly interrupt the current plan. When this happens a new world model will be constructed from the database of currently observed objects and their properties. This model is then given to the planner which generates a new plan fulfilling the goal under these modified conditions.

Since the vision system used is quite simple there is a large degree of uncertainty in the percepts. An example of the output of the vision processing where we have two regions of pixels matching predefined shapes, can be found below.

```
 ((object (shape gasbottle)(color red)(pos 2.70 0.12)...)
  (object (shape box)(color yellow)(pos 3.70 -1.75)...))
```

In addition to properties such as shape and color which can be extracted from the region as such, the system also tries to estimate the would-be object's position and size based on its position in the camera image. Even though there is a large degree of error in the percepts the final system implementing the anchoring functionalities discussed in the previous section manages in most cases to maintain a correct model of the world and to operate on this model.

Next, we describe an experiment in which our system can detect and recover from a situation where incomplete initial knowledge leads to ambiguities. In this scenario we have a simplified world consisting only of the robot, predefined places and static objects. Furthermore, the planner is given only two actions GOTO-POS and OBSERVE.

We set up the scenario depicted in Figure 2 in which we have two identical gasbottles. Gasbottle G_1 which is initially seen by the robot and which we want to investigate from another angle (eg. in order to look for damages) and G_2 which is initially occluded by the first gasbottle. We gave the planner the goal LAST-SEEN-FROM(G_1)=D and the planner, after some exploration, returned the following plan:

$$\text{GOTO-POS}(D)\ ;\ \text{OBSERVE}(G_1, D, H)$$

This plan is expected to succeed since the lack of other objects means that the predicted neccessity of reacquiring G_1 is 1.0. However, when the robot had moved to position D the anchoring system got percepts from both gasbottles and since the distance from the last point where G_1 was observed was large (3m) the matching function allowed for large errors in the apparent position of percepts-objects. This means that both percept matched object G_1 with a high possibility and the necessity of reacquiring G_1 became too low (0.11). After noting the conflict the system triggered a replanning from the current situation containing two similar objects. The initial plan would not work since the estimated necessity of reacquiring G_1 from D, when it had last been observed from H is too low because of the presence of G_2. Thus the planner instead generated the somewhat longer plan:

$$\text{GOTO-POS}(M_1)\ ;\ \text{OBSERVE}(G_1, M_1, H)\ ;\ \text{GOTO-POS}(M_2)\ ;\ \text{OBSERVE}(G_1, M_2, M_1)\ ;\\ \text{GOTO-POS}(D)\ ;\ \text{OBSERVE}(G_1, D, M_2)$$

This plan uses a more cautious approach; first observing G_1 from M_1 means that a narrower match function for position will be used when later observing G_1 from M_2 and D which makes the observe actions more likely to succeed. Executing this plan succeded without errors and the neccessity of the final reacquire was 0.93.

6 Conclusions

Knowledge-based planners have been used to plan perceptual actions, and to recover from execution failures. The approach proposed in this paper puts these two ideas together. Although the proposed approach is very simple, we believe that it constitutes a first step toward the long term objective of using knowledge-based techniques to detect and recover from perceptual problems in order to increase the autonomy of a robotic system.

One way of proceeding would be to augment the world model with simple reasoning about occlusions which would provide the means to detect and recover efficiently from a larger range of situations. We would also like to consider more complex perceptual failures and use more sophisticated diagnosis and recovery techniques. Of further interest for this approach is also the work on SRIPPs by Beetz *et al.*[2] where the authors use high level planning to generate and reason about the image processing routines needed for some perceptual actions.

Acknowledgements

This work was funded by the Swedish Research Council under grants 621-2002-4571 and 621-2001-2433. We would like to thank Silvia Coradeschi for many helpful comments.

References

[1] C. Barrouil, C. Castel, P. Fabiani, R. Mampey, P. Secchi, and C. Tessier. Perception strategy for a surveillance system. In *Proc. of the 13th European Conference on Artificial Intelligence*, pages 627–631, 1998.

[2] M. Beetz, T. Arbuckle, A. B. Cremers, and M. Mann. Transparent, flexible, and resource-adaptive image processing for autonomous service robots. In *Proc. of the 13th European Conference on Artificial Intelligence*, pages 158–170, 1998.

[3] M. Beetz and D. McDermott. Expressing transformations of structured reactive plans. In *Proceedings of the European Conference on Planning*, pages 64–76, 1997.

[4] S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In *Proceedings of the 17th AAAI Conference*, pages 129–135, Menlo Park, CA, 2000.

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

[6] Silvia Coradeschi and Alessandro Saffiotti. Perceptual anchoring of symbols for action. In *Proc. of the 17th International Joint Conference on Artificial Intelligence*, pages 407 – 412, 2001.

[7] D. Dubois and H. Prade. *Possibility theory — an approach to computerized processing of uncertainty*. Plenum Press, 1988.

[8] R.E. Fikes, P. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[9] J. Gancet and S. Lacroix. PG2P: A perception-guided path planning approach for long range autonomous navigation in unkown natural environments. In *Proceedings of IROS*, Las Vegas, NV, 2003.

[10] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Planning with sensing for a mobile robot. In *Proc. of the 4th European Conference on Planning*, pages 158–170, 1997.

[11] K.Z. Haigh and M.M. Veloso. Interleaving planning and robot execution for asynchronous user requests. *Autonomous Robots*, 5(1):79–95, 1998.

[12] L. Karlsson. Conditional progressive planning under uncertainty. In *Proceedings of the 17th IJCAI Conference*, pages 431–438, 2001.

[13] L. Karlsson and T. Schiavinotto. Progressive planning for mobile robots: a progress report. In *Advances in Plan-Based Control of Robotic Agents*, pages 106–122. Springer, Berlin, DE, 2002.

[14] S. Kovacic, A. Leonardis, and F. Pernus. Planning sequences of views for 3-D object recognition and pose determination. *Pattern Recognition*, 31:1407–1417, 1998.

[15] A. Lazanas and J.C. Latombe. Motion planning with uncertainty: A landmark approach. *Artificial Intelligence*, 76(1-2):285–317, 1995.

[16] B. Pell, D.E. Bernard, S.A. Chien, E. Gat, N. Muscettola, P.P. Nayak, M.D. Wagner, and B.C. Williams. An autonomous spacecraft agent prototype. *Autonomous Robots*, 5(1):1–27, 1998.

[17] L. Seabra-Lopes. Failure recovery planning in assembly based on acquired experience: learning by analogy. In *Proceedings IEEE Intl. Symp. on Assembly and Task Planning*, Porto, PT, 1999.