Have another look
On Failures and Recovery Planning in Perceptual Anchoring

Mathias Broxvall and **Silvia Coradeschi** and **Lars Karlsson** and **Alessandro Saf£otti**
Applied Autonomous Sensor Systems, Örebro University, Sweden
{**mathias.broxvall,silvia.coradeschi,lars.karlsson,alessandro.saf£otti**}**@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 demonstrate how techniques for planning with sensing under uncertainty can play a major role in solving the problem of recovering from such situations. In this £rst step we concentrate on failures in perceptual anchoring, that is how to connect a symbol representing an object to the percepts of that object. We provide a classi£cation of failures and present planning-based methods for recovering from them. We illustrate our approach by showing tests run on a mobile robot equipped with a color camera.

1 Introduction

There is an increasing demand for intelligent robots capable of robust operation in unconstrained environments. One of the great challenges for these robots is the need to cope autonomously with exceptional situations that arise during the execution of the assigned tasks. Explicit coding of all the possible exceptions is clearly unfeasible for environments and tasks of a realistic complexity. A more effective approch is to endow the system with the ability to use knowledge-based techniques to reason about the state of the execution, detect anomalies, and automatically generate a contingency plan.

Most existing systems that take this approach (e.g., [8, 2, 11, 17, 19]) focus on the *external* state of the world, looking for discrepancies between the observed state and 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. The ability of the robot to detect perceptual failures and to recover from them is pivotal to its providing autonomous and robust operation. In this context, Murphy and Hershberger [16] have suggested a two-step approach: a generate and test strategy for classifying sensor failures, and a recovery strategy where the failing sensory process is replaced with an equivalent process.

Several works in the £eld 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], for performing tasks that involve sensing actions [10, 13], and for generating image processing routines [3]. None of these works, however, deal with the problem of failures in the perceptual actions and of the automatic recovery from these failures.

We propose to use AI planning techniques to automatically generate a plan to recover from failures in the perceptual processes. We focus on one speci£c type of perceptual process: *perceptual anchoring*. Perceptual anchoring is the process of creating and maintaining the right correspondence between the symbols used by the planner to denote objects in the world and the perceptual data in the sensorimotoric system that refer to the same objects. In a previous paper [4] a simple case of anchoring failure due to the accumulation of uncertainty has been investigated. In this paper, we extend that investigation and analyze all the different cases of ambiguity that can make the anchoring process fail. For each case, we show how that situation can be automatically detected and isolated, and how we can use a planner to generate a sequence of actions to recover from the failure when possible.

In the next section, we give a brief reminder of perceptual anchoring. In section 3 we classify the different ways in which anchoring can fail, and explain how they can be detected. In section 4 we show how the failure situation can be modeled in a planner and a recovery plan generated automatically for those cases that can be £xed. Finally, we demonstrate our technique by presenting a series of experiments run on a mobile robot equipped with a color camera.

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 ‘b1’. 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 de£ned as the *anchoring problem* [7], 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.

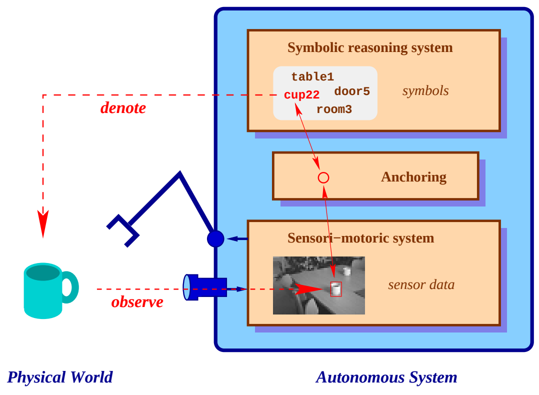

Figure 1. The perceptual anchoring problem.

In our work, we use the computational framework for anchoring de£ned in [5]. In that framework, the symbol-data correspondence for a speci£c 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(b1)’, where the symbol ‘b1’ denotes an object described in the planner as ’a tall green gas bottle’.[1] 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(b1)’ action, the robot must make sure that the region used in the control loop is exactly the one generated by observing the object that the planner calls ‘b1’. Thus, the robot uses a functionality called **Find** to link the symbol ‘b1’ to a region in the image that matches the description ’a tall green gas bottle’. The output of Find is an anchor that contains, among other properties, the (x, y) position of the gas bottle, 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 gas bottle 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 gas bottle is in view again. More details about perceptual anchoring can be found in [5, 6, 7].

A central ingredient in all the anchoring functionalities is the *matching* between the symbolic description given by the planner and the observed properties of the percepts generated by the sensor system. Matching is needed to decide which percepts to use to create or update the anchor for a given symbol. Matching between a symbolic description and a percept can be *partial* or *complete* [6]. It is complete if all the observed properties in the percept match the description and vice-versa. It is partial if all the observed properties in the percept match the description, but there is some property in the description that have not been observed (or not reliably).[2] For example, consider the description “a gas bottle with a yellow mark”. A gas bottle in an image where no mark is visible provides a partial match, since the mark might not be visible from the current viewpoint.

[1] Throughout this paper, we use examples inspired by an ongoing rescue project where the robot is supposed to £nd overheated and dangerous gas bottles in a burnt building.

[2] Note that matching does not have to be a binary concept, but can be considered in degrees. In such cases, we use a threshold to determine what is partial and what is complete; the threshold can be raised if there are several matching candidates.

3 Anchoring with ambiguities

An important challenge in the anchoring process is how to treat ambiguous cases, that is cases for which it is not clear to what perceptual data the symbol should be associated. The £rst step to treat ambiguities is to detect that an ambiguity is actually present. The second step is then to try to resolve the ambiguity if possible or otherwise admit failure. In this section we concentrate on the detection of ambiguity, while in the next section we present how some classes of ambiguous situations can be recovered from using planning techniques.

To better characterize the ambiguous cases we need £rst to clarify the distinction between de£nite and inde£nite descriptions for an object. A description is *de£nite* when it denotes an unique object, for instance ”the cup in my of£ce”, supposing that I have just one cup in my of£ce. Linguistically one use in this case the article ”the”. An *inde£nite* description requires that the object corresponds to the description, but not that it is unique, for instance ”a red cup”. De£nite descriptions are especially challenging when an object is conceptually unique, but its perceptual properties do not characterize it unequivocally, for instance ”the cup that I have seen before”. This is a common event in the Reacquire functionality when more than one object matches the description of a previously seen object (in Reacquire descriptions are always de£nite). An example of this situation is shown later in this paper.

An important case in anchoring is when we have multiple candidate percepts matching the description and by consentrating on this we can indentify a number of different failures. The important variables when detecting and identifying an ambiguity in anchoring are the number of candidate percepts that match a description completely and partially, and whether the description involved is de£nite or inde£nite. In the following, we give a classi£cation of these ambiguities. We also describe what the Find and Reacquire functionalities return in each case, and what could constitute a recovery from the situation.

Case 1: no candidates. In this case no object matching the description is found. Therefore, both Find and Reacquire return a failure message. If the object might be somewhere else we generate a search-plan. If one has exhausted the search possibilities, there is a failure.

Case 2: one or more partially matching candidates. A partial matching indicates that one has inadequate information about some relevant properties of the perceived object(s). When this happens, both functionalities create temporary anchors for each of the candidates and return these anchors to be used by the recovery planner. Sometimes, one might still be able to determine whether this is the requested object - one might e.g. have prior information that there are no other similar objects around — and in those occasions, the case turns into one of a complete matching (see below). However, in most situations one will need to try to acquire more information about the object(s) in question, in order to get a complete match. Therefore, one needs to generate a recovery plan. The anchors created by the functionalities let the planner access to information about these objects while the recovery plan is constructed and executed. If the situation is successfully disambiguated, the planner informs the anchoring module which of the candidate perceived objects should be used for anchoring.

Case 3: a single completely matching candidate. This is the ideal case: one just picks that candidate. Both functionalities create an anchor and return it to the planner.

Case 4: one completely matching candidate, and some partially matching ones. The inde£nite case is simple: one can just pick the completely matching candidate. For the de£nite case, that is also an option. However, if one is cautious and wants to ascertain that there is not an ambiguity hidden here, one might want to acquire more information to be able to rule out the candidates with incomplete matches. In our current implementation in the inde£nite case the Find functionality creates an anchor with the completely matching candidate and returns it to the planner. In the de£nite case both functionalities create anchors both for the complete matching candidate and the incomplete matching ones. They then return the anchors while making a distinction between the completely and partially matching ones.

Case 5: multiple completely matching candidates. Again, the inde£nite case is simple: just pick one of the candidates. The Find functionality creates an anchor with the completely matching candidate and returns it to the planner. In the de£nite case, however, this constitutes a serious problem: as the matchings are complete, the situation cannot be resolved by getting more perceptual information. Instead, the description has to be considered insuf£cient, and needs to be made more precise. (how to do that is not adressed in this paper).

Finally, we should point to some particular dif£cult situations: when the description is not only insuf£cient but also wrong; when important characteristics of the object have changed in a way we cannot predict (e.g. the shape has been deformed); and when our percepts are not just uncertain but misleading (e.g. a re¤ection is taken to be a color mark). In such cases, we might get mismatches that should have been matches, and vice versa, which in turn leads to an erroneous estimate of the situation and possibly also a misclassi£cation of what case we have.

4 Recovery planning for anchoring

In order to recover from cases 1, 2 and (optionally) 4 above, we encode the situations as planning problems for a conditional possibilistic/probabilistic planner called PTLplan [12]. The other cases either do not need to be solved (case 3) or cannot be solved (case 5).

PTLplan searches in a space of epistemic states, or e-states for short, where an e-state represents the agent's incomplete and uncertain knowledge about the world at some point in time. An e-state can be considered to represent a set of hypotheses about the actual state of the world, for instance that a certain gas bottle has a mark on it or has not a mark on it. The planner can reason about perceptive actions, such as looking at an object, and these actions have the effect that the agent makes observations that may help it to distinguish between the different hypotheses. Each different observation will result in a separate new and typically smaller e-state, and in each such e-state the agent will know more than before. For instance, looking at a gas bottle may result in two observations leading to two possible e-states: one where the agent knows there is a mark, and one where it knows there isn't a mark on that side.

A recovery situation in anchoring typically occurs when the robot is executing some higher-level plan and encounters one of the ambiguous but recoverable cases above. Such a situation is handled in £ve steps:

1. The problematic situation is detected and classi£ed as above, and the top-level plan is halted.
2. The planner automatically formulates an initial situation by considering the properties of the requested object and of the perceived objects, and generating different hypotheses for which of the objects corresponds to the requested object. It also formulates a goal that the requested object should be identi£ed if present.
3. The planner searches for a plan taking as parameters the e-state and the goal.
4. The plan is executed, and either the requested object is found and identi£ed and can be anchored, or it is established that it cannot be identi£ed.
5. If recovery was successful, the top-level plan is resumed.

The domain description used for anchoring recovery planning typically is not the same as is used for top-level plans (although in our case the planner is the same). Typically, the actions involved would be restricted to certain perceptual actions, and the description of the locality may be more detailed to facilitate search.

4.1 Formulating the initial situations and goals

In **case 1**, where no candidate for the requested object (say b1) has been found, a search needs to be performed. Therefore, the initial situation consists of a number of hypotheses of where the object can be found, including the hypothesis that it is nowhere around. For instance, if there are four places in the room of interest, and we have already searched at one of them, the hypotheses might be that b1 will be visible from one of the remaining places, or from none (f below). Note that the term following the "=" is the value of the property to the left of the "=", and the numbers are degrees of possibility associated with each hypothesis:

1.0	(visible-from b1 = r1_2)
1.0	(visible-from b1 = r1_3)
1.0	(visible-from b1 = r1_4)
0.5	(visible-from b1 = f)

To the above is added information about the topology of the room that is to be searched, and the description of the object to be anchored, e.g. (shape b1 = gasbottle). The goal is formulated as (exists (?x) (nec (visible-from b1 = ?x))), which means that the agent has determined from what place the object is visible.

In **case 2**, where there are one or more partially matching perceived objects, the agent needs to £gure out which of them actually matches the requested object b1. Thus, the hypotheses consists of the different ways b1 can be anchored, based on the known properties of b1 and the perceived properties of the perceived objects. Based on the descriptions d for the requested object and d_i for each perceived object po$_i$, two extra descriptions are formulated for every d_i: £rst, a description d_i^+ which completely matches d; and second, a non-matching description d_i^- which contains the different ways in which at least one incompletely speci£ed property in d_i may not match with d. For instance, if $d =$ (mark b1 = t) and $d_1 =$ (mark po1 = t f) (i.e. either true or false), then $d_i^+ =$ (mark po1 = t) and $d_i^- =$ (mark po1 = f). Each hypothesis then consists of the conjunction of one d_i^+ for one of the po$_i$ and d_j^- for all remaining $j \neq i$. To each hypothesis is also added the statement (anchor b1 = po$_i$) denoting that b1 should be anchored to the object anchored by po$_i$. There is also one hypothesis that no object matches: d_j^- for all j, and (anchor b1 = f). Finally, if the planner wishes to take a cautious approach and ascertain that no more than one object is matching, it might also add a number of hypotheses consisting of d_i^+, d_j^+ for two of the po$_i$, po$_j$ and d_k for all remaining $k \neq i, j$, and (anchor b1 = f).

For instance, if b1 is known to be a green gas bottle with a mark on it — (mark b1 = t) — and we perceive two green gas bottles po1 and po2 but are not able to see any marks on them from the current perspective, the (incautious) hypotheses might be:

1.0	(mark po1 = t), (mark po2 = f), (anchor b1 = po1)
1.0	(mark po1 = f), (mark po2 = t), (anchor b1 = po2)
0.5	(mark po1 = f), (mark po2 = f), (anchor b1 = f)

In addition, each of the two hypotheses can be subdivided further into three different hypotheses regarding from where the mark can be detected: (mark-visible-from po1 = r1_1) and so on.

The goal is achieved once a speci£c action (anchor b1 *x*) has been performed. This action has as a precondition that *x* is the only remaining anchor for b1: (nec (anchor b1 = *x*)). Thus, all other candidate anchors have to be eliminated before anchor is applied.

Case 4 is quite similar to case 2 above, but consists of one hypothesis where the completely matching percept is chosen for anchoring, and a number of hypotheses where there are other objects matching too.

4.2 Generating the recovery plan

After the initial situation and the goal have been established, plan generation starts, using the appropriate domain description. The following action, for instance, is for looking for marks (and other visual characteristics) on objects such as gas bottles.

```
(ptl-action
:name (look-at ?y)
:precond ( ((?p) (robot-at = ?p)) ((?y) (perceived-object ?y)) )
:results (cond
              ((and (mark ?y = t) (mark-visible-from ?y = ?p))
               (obs (mark! ?y = t)))
              ((not (and (mark ?y = t)
                         (mark-visible-from ?y = ?p)))
               (obs (mark! ?y = f))))
:execute  ((aiming-at me ?y)
           (anchor-£nd ?y :when (aiming-at me ?y))))
```

In short, the precond part states that the action requires a perceived object ?y and a current position ?p. The result part states that if ?y has a mark, and if the robot looks at ?y from the ?p from which the mark is visible, then the robot will observe the mark (and thus know that there is a mark), and otherwise it will not observe any mark. The obs form is the way to encode that the agent makes a speci£c observation.

The plans generated by PTLplan are conditional: after each action with observation effects (and with more than one alternative outcome), the plan branches. The plan below is generated for looking for marks on a single perceived object from three different positions, starting from a fourth position. Note how a conditional branching follows after each application of look-at: the £rst clause "(mark! po-4 = t/f)" of each branch is the observation one should have made in order to enter that branch, and the subsequent clauses are actions. The action (anchor b1 *x*) at the end of each branch represents the decision to anchor b1 to some speci£c perceived object (or to no object at all, if *x* = f).

```
((move r1_2) (look-at po-4)
 (cond
   ((mark! po-4 = f) (move r1_3) (look-at po-4)
    (cond
      ((mark! po-4 = f) (move r1_4) (look-at po-4)
       (cond
         ((mark! po-4 = t) (anchor b1 po-4) :success)
         ((mark! po-4 = f) (anchor b1 f) :success)))
      ((mark! po-4 = t) (anchor b1 po-4) :success))
   ((mark! po-4 = t) (anchor b1 po-4) :success)))
```

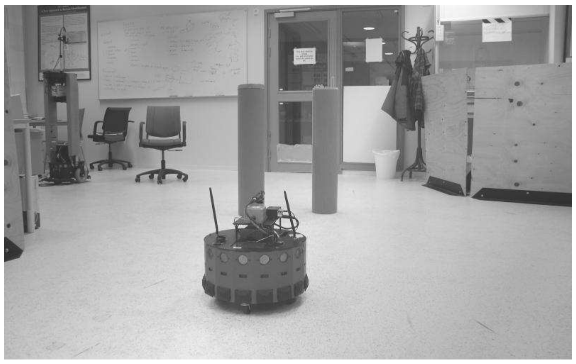

Figure 2. Our robot investigating two bottles

We omit the details of how the plan is generated here, as our approach is not dependent on the particular planning algorithm. Actually, another planner with corresponding expressive power could have been used instead.

4.3 Plan execution

The anchoring plan is then executed: the actions such as (look-at po-4) are translated into executable perceptive and movement tasks (see £eld :execute in the de£nition of look-at above). The anchor action has a special role: it causes the symbol of the requested object to be anchored to a speci£c perceived object. The robot can then continue performing the task in its top-level plan that was interrupted.

5 Tests on a robot

To be able to test the methods described above we have implemented and integrated them with a fuzzy behavior based system, the Thinking Cap [18], used for controlling a mobile robot. We have used this system to run a number of scenarios yielding different kinds of ambiguities. We give here a brief description of the system, the scenarios and the resulting executions.

The platform we have used is a Magellan Pro Research Robot. equipped with standard sonars, bumpers and IR sensors. In addition to the standard setup we have connected a camera and use a simple image recognition system to detect and extract information about objects matching a number of prede£ned patterns.

Apart from the anchoring, plan execution and planning modules described in the previous sections the complete system also consists of a number of other parts which allows the robot to navigate indoor environments safely and perceive the surroundings. Perception is accomplished by continuously receiving percepts from the vision system, associating them with earlier percepts and storing them for later use by the anchoring system.

In these test the robot operates in a room containing one or more gas bottles (Figure 2). These bottles can be of various colors and can optionally have a mark on some side. Typical tasks we have given the robot is to look for gas bottles matching a speci£c description, approaching them, moving around in or exiting the room and re-identifying previously found gas bottles. The actions available to the robot were to look for a speci£c object at a speci£c place, to look at a previously seen object, to move to different positions or near to an object, to select a speci£c object for anchoring, and to perform self-localization by moving to a £xed position.

Scenario 1: No ambiguity. The £rst and simplest scenario we have run is when we placed a green gas bottle in the room clearly visible from the robot's location and gave the planner the task to look for and approach b1 with the symbolic description ((color green) (shape gasbottle)). Initially, the plan executor called the Find functionality. Since there was only one completely matching percept (**case 3**) the system anchored b1 to this percept and continued with the plan. The position property of the b1 anchor was used to approach the gas bottle and £nish the original task.

Scenario 2: Searching the room. For the next tests we look at **case 1**, where we have no matching candidates to a Find. We set this up by partially obstructing the gas bottle so that it could be seen only from certain positions in the room. Next, we started the robot at a position where the gas bottle was not visible and gave it again the task to look for and approach b1 The £rst call to the Find functionality failed. This triggered the planner to generate a recovery plan from a description of the current world state, using the information that there should somewhere be a gas bottle. The result was a conditional plan that would navigate to different parts of the room, looking for the gas bottle, and announcing success when it was found. After this, the original task of approaching b1 could continue.

Scenario 3: Partially matching objects. In this scenario we choose to look at **case 2** were the system perceives one or more objects only partially matching the description. We did this by using a red gas bottle with a mark on it which was not visible from the initial position. We then asked the system to look for b1 matching ((color red) (mark t) (shape gasbottle)) and the Find functionality was called.

At this point in time the system perceived a red gas bottle but could not determine whether it was marked on some side. Thus we had only one partially matching candidate. The system now created a temporary anchor Anchor-1 for this object and the planner generated a recovery plan using the knowledge that Anchor-1 might be the same as b1 and then should have a mark visible from some side. The planner produced a conditional plan which would navigate through the room and observe Anchor-1 to see if a mark was visible from the different viewpoints and to halt when the mark was found. The robot navigated through the room, found the mark and concluded that the observed gas bottle was the right one.

We also successfully ran the same scenario with more advanced setups where we either had no mark on the gas bottle, or where we had two gas bottles of which only one was marked.

Scenario 4: Planning to reacquire. In order to test a reacquire ambiguity we had to setup a scenario where the position of an object could not be used to uniquely identify a previously acquired object. To do this we started with two gas bottles in front of the robot, one of the gas bottles had a mark on the side facing the robot. Next, we asked the robot to look for b1 with the inde£nite description ((marked yes) (shape gasbottle)); to exit to a corridor in the opposite side of the room; and £nally to again enter the room and reacquire b1.

In the initial Find we got one partial and one completely matching candidate (**case 4**) and the marked gas bottle was anchored to b1. After this the robot navigated to the opposite side of the room; entered the corridor and went back into the room again. The accumulated uncertainty in the robot's self localization was now so large that when the robot was doing the £nal reacquire, it failed to determine which percept corresponded to b1. Since the mark on the initially anchored gas bottle could not be seen from this position we had an ambiguity due to multiple partial matchings (**case 2**). Thus the planner was triggered to resolve the ambiguity and it generated a plan to investigate both gas bottles to see which one was marked. The result was that the robot reacquired the right gas bottle.

We also tested alternative versions of this setup where instead of failing due to bad self localization we either moved the gas bottles or introduced a new gas bottle before acquiring them again. Moving without observing, or introducing new bottles always gave ambiguities. Due to the implicit tracking done by the vision system, moving them while observed gave only ambiguities if the gasbottles overlapped from the cameras viewpoint during movement. In either case these version gave the same kind of ambiguities and was also solved correctly by observing the gas bottles from different positions until the mark was found.

Scenario 5: Planning for relocalization. Since our implemented system mainly uses odometry for localization the degree of uncertainties in the position of objects increases monotonically with movement, unless the objects are observed. This means that even though we have acquired an object and have a position property we may get only a partial matching during a later reacquire on the same object.

To see that this case could be handled, we setup a scenario with two identical gasbottles where we let the robot acquire one of them as b1. Next, we moved the robot (out of the room and back) and asked it to go near b1. Because of odometry errors the position property of b1 could not be used to acquire the right gasbottle and instead we got an ambiguity (**case 2**). The solution generated by the planner was to use a self localization action to remove the odometry error and acqurie the right gas bottle.

6 Conclusions

There are two main contributions in this paper. Firstly, we have highlighted the usefulness of knowledge-based planning in robotics in the context of autonomous recovery from perceptual errors. Our results indicate that this direction is very promising for what concerns recovery from anchoring failures, in particular as the complexity and variety of the problems involved motivates the use of on-line planning as opposed to a hard-coded approach.

Secondly, we have presented a classi£cation of different cases that can be the outcome when an embedded agent such as a robot is attempting to anchor symbols to percepts. We have also shown how to use planning techniques to automatically recover from some of these cases, and we have demonstrated our approach on a mobile robot confronted with a number of failure situations.

7 Acknowledgements

This work has been supported by The Swedish Research Council (Vetenskapsrådet) and by the Swedish KK foundation.

REFERENCES

[1] C. Barrouil, C. Castel, P. Fabiani, R. Mampey, P. Secchi, and C. Tessier. Perception strategy for a surveillance system. In *Proc. of ECAI*, pages 627–631, 1998.

[2] M. Beetz and D. McDermott. Expressing transformations of structured reactive plans. In *Proc. of the European Conf. on Planning,*, pages 64–76. Springer, 1997.

[3] Michael Beetz, Tom Arbuckle, Armin B. Cremers, and Markus Mann. Transparent, ¤exible, and resource-adaptive image processing for autonomous service robots. In *Proc. of the 13th European Conference on Arti£cial Intelligence*, pages 158–170. John Wiley and Sons, 1998.

[4] M. Broxvall, L. Karlsson, and A. Saf£otti. Steps toward detecting and recovering from perceptual failures. In *Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems (IAS)*, Amsterdam, NL, 2004.

[5] S. Coradeschi and A. Saf£otti. Anchoring symbols to sensor data: preliminary report. In *Proc. of the 17th AAAI Conf.*, pages 129–135, Menlo Park, CA, 2000. AAAI Press.

[6] S. Coradeschi and A. Saf£otti. Perceptual anchoring of symbols for action. In *Proc. of the 17th IJCAI Conf.*, pages 407–412.

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

[8] R.E. Fikes, P. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Arti£cial 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 *Proc. of IROS*, Las Vegas, NV, 2003. To appear.

[10] G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Planning with sensing for a mobile robot. In *Proc. of the 4th European Conf. on Planning*, pages 158–170. Springer, 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 *Proc. of the 17th IJCAI Conf.*, pages 431–438. AAAI Press, 2001.

[13] L. Karlsson and T. Schiavinotto. Progressive planning for mobile robots: a progress report. In M. Beetz, J. Hertzberg, M. Ghallab, and M. Pollack, editors, *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. *Arti£cial Intelligence*, 76(1-2):285–317, 1995.

[16] Robin R. Murphy and David Hershberger. Classifying and recovering from sensing failures in autonomous mobile robots. In *Proc. AAAI-96*, pages 922–929, 1996.

[17] 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.

[18] A. Saf£otti, K. Konolige, and E.H. Ruspini. A multivalued-logic approach to integrating planning and control. *Arti£cial Intelligence*, 76(1–2):481–526, 1995.

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