Perceptual Anchoring of Symbols for Action

Silvia Coradeschi and **Alessandro Saffiotti**
Center for Applied Autonomous Sensor Systems
Örebro University, S-70182 Örebro, Sweden
http://www.aass.oru.se
silvia.coradeschi@aass.oru.se, alessandro.saffiotti@aass.oru.se

Abstract

Anchoring is the process of creating and maintaining the correspondence between symbols and percepts that refer to the same physical objects. Although this process must necessarily be present in any symbolic reasoning system embedded in a physical environment (e.g., an autonomous robot), the systematic study of anchoring as a clearly separated problem is just in its initial phase. In this paper we focus on the use of symbols in actions and plans and the consequences this has for anchoring. In particular we introduce action properties and partial matching of objects descriptions. We also consider the use of indefinite references in the context of action. The use of our formalism is exemplified in a mobile robotic domain.

1 Introduction

The focus of this paper is the connection between abstract- and physical-level representations of objects in artificial autonomous systems embedded in a physical environment. We call *anchoring* the process of creating, and maintaining in time, this connection.

Anchoring must necessarily occur in any physically embedded system that comprises a symbolic reasoning component. A typical example is the problem of connecting, inside an autonomous robot, the symbol used by a symbolic planner to refer to a physical object to the data in a perceptual system that pertains to the same object. This connection must be dynamic, since the same symbol must be connected to new percepts when the same object is re-acquired. For instance, a robot may be asked to identify and track a specific person in a crowd using visual data and given a linguistic description.

Anchoring is related to *symbol grounding*, defined as the problem of how to give an interpretation to a formal symbol system that is based on something that is not just another symbol system [Harnard, 1990]. Anchoring is an important special case of symbol grounding where the symbols denote individual physical objects.

The recognition of the anchoring problem as a problem *per se* is a recent phenomenon. Although all existing robotics systems that comprise a symbolic reasoning component implicitly incorporate a solution to the anchoring problem, this solution is typical hidden in the code, and it is developed on a system by system basis on a restricted domain. To the best of our knowledge, the first domain independent definition of the anchoring problem was given in [Saffiotti, 1994], while the first attempt at a computational theory of anchoring was reported in [Coradeschi and Saffiotti, 2000]. The goal of this theory was to specify the functionalities and the representations needed to perform anchoring in a general way, that is applicable to a large number of systems.

In this paper, we focus on one specific aspect of anchoring: the use of symbols to denote objects in actions and plans, and the anchoring of these symbols to objects in the world. Consider the action 'PickUp(A).' We are interested in the problem of how to anchor the symbol 'A' to the relevant physical object through perception. To do this, we start from the above theory of anchoring, and extend it in three ways. First, we make a distinction between the properties of 'A' which are needed to *identify* the physical object to be used for the action, and those which are needed to *perform* the action. Second, we introduce the concept of *partial matching*, where 'A' can be (tentatively) anchored to an object whenever a required perceptual property cannot be extracted from the sensor data. Third, we consider the use of *indefinite references* in the context of action. Definite references, like "*the* black suitcase,"are meant to refer to one specific object with given properties, while indefinite ones, like "*a* black suitcase" are meant to refer to an arbitrary object in a given class [Russell, 1905].

In order to clarify the use of anchoring, we show two experiments performed on a real robot. These examples illustrate how anchoring can be integrated in a robot architecture, and how its functionalities can be used to to connect the symbols used by a planner to the data acquired by a vision system. The examples also show the essential difference between two ways to treat actions that involve an indefinite reference. This reference can be resolved by the symbol system (planner), or by the anchoring process.

2 A Basic Model of Anchoring

We summarize here the basic elements of the computational theory of anchoring defined in [Coradeschi and Saffiotti, 2000]. The theory considers an agent that includes a symbol system and a perceptual system, and it focuses on the problem of creating and maintaining a correspondence between

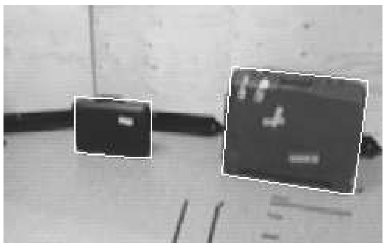

Figure 1: Two percepts extracted from a camera image.

symbols and percepts that refer to the same physical object. It consists of a static part and a dynamic part. The static part includes the following.

- A *symbol system* Σ including: a set $\mathcal{X} = \{x_1, x_2, \ldots\}$ of individual symbols (variables and constants); a set $\mathcal{P} = \{p_1, p_2, \ldots\}$ of predicate symbols; and an inference mechanism whose details are not relevant here.
- A *perceptual system* Ξ including: a set $\Pi = \{\pi_1, \pi_2, \ldots\}$ of percepts; a set $\Phi = \{\phi_1, \phi_2, \ldots\}$ of attributes; and perceptual routines whose details are not relevant here. A percept is a structured collection of measurements assumed to originate from the same physical object; an attribute ϕ_i is a measurable property of percepts, with values in the domain D_i. We let $D = \bigcup_i D_i$.
- A *predicate grounding relation* $g \subseteq \mathcal{P} \times \Phi \times D$, that embodies the correspondence between unary predicates and values of measurable attributes.

Example. Σ may be a planner that includes the individual symbol 'A' and the predicate symbols 'large' and 'small.' Ξ may be a vision system able to recognize suitcases: from the image shown in Fig. 1, Ξ may extract two percepts π_1 and π_2. Attributes computed by Ξ may include 'color' and 'width.' The predicate grounding relation g may include the triple $\langle \text{small}, \text{width}, 10 \rangle$: this says that the measure 10 for an object's observed width is consistent with the predication of its being small.[1]

The g relation concerns properties, but anchoring concerns objects. The following definitions allow us to characterize objects in terms of their (symbolic and perceptual) properties.

Definition 1 *A* symbolic description $\sigma \in 2^{\mathcal{P}}$ *is a set of unary predicates.*

Definition 2 *A* perceptual signature $\gamma : \Phi \to D$ *is a partial function from attributes to attribute values. The set of attributes on which γ is defined is denoted by feat(γ). $\Gamma = (\Phi \to D)$ is the set of all γ.*

Intuitively, a symbolic description lists the predicates that are considered relevant to the perceptual recognition of an object; and a perceptual signature gives the values of the measured attributes of a percept (and it is undefined for the remaining ones). The g relation can then be used to define a

[1]For the sake of simplicity we consider here a very simple g. The g relation can be quite complex in real domains.

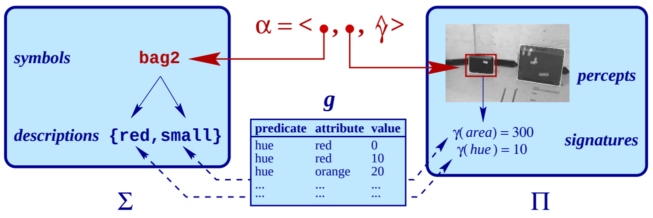

Figure 2: The elements of anchoring.

function *match*(σ, γ) that says whether or not the values in the perceptual signature γ are consistent with a given symbolic description σ.

The dynamic part of the model tells us, at any time t, what properties are associated to symbols in Σ, and what attribute values are associated to percepts in Ξ.

- A *description state* $DS_t : \mathcal{X} \to 2^{\mathcal{P}}$ that associates each individual x with its symbolic description at time t.
- A *perceptual state* $PS_t : \Pi \to \Gamma$ that associates each percept $\pi \in \Pi$ to its perceptual signature at time t. If π is not perceived at time t, then $PS_t(\pi)$ is everywhere undefined. The set of percepts which are perceived at t is denoted by V_t.

Example. Consider our previous example. At time t, the symbol system may associate property 'small' to symbol 'A' by having $\texttt{small} \in DS_t(A)$. The perceptual system may extract the width of the two percepts in the image, and associate them with the perceptual signatures $PS_t(\pi_1) = \gamma_1$ and $PS_t(\pi_2) = \gamma_2$ such that $\gamma_1(\text{width}) = 10$ and $\gamma_2(\text{width}) = 20$.

The role of anchoring is to establish a correspondence between a symbol x used in the symbol system to denote an *object* in the world, and a percept π generated in the perceptual system by the same object. This is done by comparing the symbolic description $DS_t(x)$ and the perceptual signature $PS_t(\pi)$ via the *match* function, hence via the g grounding relation. In the previous example, the g relation includes the triple $\langle \text{small}, \text{width}, 10 \rangle$, therefore the description of 'A' and the perceptual signature of π_1 can be matched, thus suggesting that the symbol 'A' might be anchored to the percept π_1.

The above correspondence is reified in an internal data structure α, called *anchor*. Since new percepts are generated continuously within the perceptual system, this correspondence is indexed by time.

Definition 3 *An* anchor α *is any partial function from time to triples in $\mathcal{X} \times \Pi \times \Gamma$.*

At every moment t, $\alpha(t)$ contains: a symbol, meant to denote an object inside Σ; a percept, generated inside Ξ by observing that object; and a signature, meant to provide the (best) estimate of the values of the observable properties of the object. We denote these components by $\alpha_t^{\text{sym}}, \alpha_t^{\text{per}}$, and α_t^{sig}, respectively. If the object is not observed at time t, then α_t^{per} is the 'null' percept $\perp$, and α_t^{sig} still contains the best available estimate. Intuitively, an anchor can be seen as an internal, sensori-motor level representation of a physical object — see Fig. 2.

In order for an anchor to satisfy its intended meaning, the symbol and the percept in it should refer to the same physical object. This requirement cannot be formally stated inside the system. What can be stated is the following.

Definition 4 *An anchor* α *is* grounded *at time t iff* $\alpha_t^{\mathrm{per}} \in V_t$.

We informally say that an anchor α is *referentially correct* if, whenever α is grounded at t, then the physical object denoted by α_t^{sym} is the same as the one that generates the perception α_t^{per}. The *anchoring problem*, then, is the problem to find referentially correct anchors.

3 Extending the Model

The above model provides the basic ingredients of a general theory of anchoring, with no assumption as to the task for which anchoring is performed. In this paper, however, we focus on the use of anchoring to connect symbols for actions to physical objects through perception. From this perspective, the anchoring process should make sure that the anchor: (i) represents a physical object which has the intended properties for the intended action, and (ii) contains information about the perceptual properties which are needed in order to perform the action. For instance, if the action is meant to pick up a green suitcase, then the anchor should represent an object which is a green suitcase, and its signature should include an estimate of the position and orientation of this suitcase.

3.1 Action properties

The *match* function makes sure that the anchor is adequate with respect to the properties stored in the description state, i.e., $\mathit{DS}_t(A)$. In order to make sure that the anchor's signature also contains the properties that are relevant for action, we extend the dynamic part of our model by including the following.

- An *action parameter state* $A_t : \mathcal{X} \to 2^{\mathcal{P}}$ that associates each individual x with the set of properties that need to be known in order to act on the object denoted by x.

In our example, $A_t(\mathrm{A})$ would specify the position and orientation of the suitcase. In the following we call *matching properties* the set of predicates in $\mathit{DS}_t(A)$ and *action properties* the set of predicates in $A_t(A)$

Note that the predicates in A_t are not used in the matching process: these predicates only indicate that the corresponding attributes must be included in the anchor's perceptual signature γ. However, these predicates can be used as *default assumptions* about the expected properties of the object in order to start action before the object is actually perceived. For instance, we may have an expectation about the position of a suitcase: this expectation can be included in the signature of the anchor in order to start approaching that position until the actual suitcase is perceived. These expectations can also be used to focus the perceptual system.

Having a property in the action state or in the description state may affect the meaning of an action. In our "PickUp(A)" example, if the position is not included in the description state, then 'A' will be anchored to any green suitcase, irrespective of its position. If the position of a given suitcase in included in the description state, then 'A' will only be anchored to the specific suitcase at that position. The first setup encodes the action "pick up *a* green suitcase," while the second one encodes the action "pick up *the* green suitcase at the given position" [Saffiotti, 1994]. Note that the object of the action is denoted by an indefinite reference in the first case, and by a definite one in the second case.

3.2 Partial Matching

Some actions may affect the perceptual information which is gathered by the agent: for instance, moving closer to an object may allow the perceptual system to observe more properties of the object. For example, suppose that we are interested in a suitcase with a white label on it: depending on the distance and angle of the suitcase, the label may not be visible.

Recognizing the fact that not all attributes of a percept may be extracted at all times brings about the need to redefine the meaning of the *match* function: $\mathit{match}(\sigma,\gamma)$ should check that the signature γ is consistent with the descriptor σ for those attributes which have actually been observed, but it should ignore the ones which have not been observed. The following is a possible way to define the *match* function.

$$\mathit{match}(\sigma,\gamma) = \begin{cases} \emptyset \text{ if } \exists p \in \sigma.\,(\mathit{obs}(p,\gamma) \wedge \neg \mathit{cons}(p,\gamma)) \\ \{p \in \sigma \mid \mathit{obs}(p,\gamma)\} \text{ otherwise} \end{cases}$$

where

$$\begin{aligned} \mathit{obs}(p,\gamma) &\Leftrightarrow \exists \phi \in \mathit{feat}(\gamma).\exists d \in D.g(p,\phi,d) \\ \mathit{cons}(p,\gamma) &\Leftrightarrow \exists \phi \in \mathit{feat}(\gamma).g(p,\phi,\gamma(\phi)) \end{aligned}$$

Intuitively, $\mathit{obs}(p,\gamma)$ says that an attribute related to p (via the g relation) has been observed in γ, and $\mathit{cons}(p,\gamma)$ says that the observations in γ are consistent with the p predicate according to g. $\mathit{match}(\sigma,\gamma)$ returns $\emptyset$ if there was a mismatch between some predicate in σ and the observed values; otherwise it returns the subset of the predicates in σ for which a (consistent) value has actually been observed.

It is useful to keep track of which of the predicates in the symbolic description for a symbol have actually been observed, and which ones have not. To do this, we extend the definition of an anchor to include a list of the observed predicates as follows.

Definition 3 (bis) *An* anchor *is any partial function from time to tuples in* $\mathcal{X} \times \Pi \times \Gamma \times 2^{\mathcal{P}}$.

We write α^{obs} to denote the fourth element of an anchor $\alpha(t)$.

4 The Functionalities of Anchoring

In order to turn the above model into a useful computational framework, we need to define which functionalities are needed in order to solve the anchoring problem for a given symbol x. In [Coradeschi and Saffiotti, 2000] three main functionalities have been identified: (i) to create a grounded anchor the first time that the object denoted by x is perceived; (ii) to update the anchor when we need to reacquire the object after some time that it has not been observed; and (iii) to continuously update the anchor while observing the object. Given the above extensions to the model, these functionalities can be defined as follows. (t denotes the time at which the functionality is called.)

Find Take a symbol x and return a grounded anchor defined at t, and undefined elsewhere. In case of multiple matching percepts, return one anchor for each of them. This is summarized by the following pseudo-code.

```
procedure Find $(x, t)$
  $\Pi \leftarrow \{\pi \in V_t \mid match(DS_t(x), PS_t(\pi)) \neq \emptyset\}$
  if $\Pi = \emptyset$
    then fail
    else for $\pi_i \in \{\pi_1, \ldots, \pi_n\} = \Pi$
           $\mu_i \leftarrow match(DS_t(x), PS_t(\pi_i))$
           $\gamma_i \leftarrow Attributes(A_t(x), PS_t(\pi_i), \mu_i)$
           $\alpha_i(t) \leftarrow \langle x, \pi_i, \gamma_i, \mu_i \rangle$
return $\{\alpha_1, \ldots, \alpha_n\}$
```

The *Attributes* function returns the part of the perceptual signature $PS_t(\pi_i)$ that only includes the "interesting" attributes, that is, those that correspond to either description properties or to action properties. (A specific implementation may also include attributes which are needed by the perceptual system to track the object, e.g., its position and velocity.)

Reacquire This function is used to find an object when there is a previous perceptual experience of it. Take an anchor α defined at time $t - k$ and extend α's definition to t. First, predict a new signature γ; then see if there is some new percept that is compatible with the prediction and the symbolic description; in case of multiple matching percepts, use a domain dependent selection function. If one percept is found, update γ. Prediction, verification of compatibility, and updating are domain dependent; verification should typically use *match*.

```
procedure Reacquire $(\alpha, t)$
  $x \leftarrow \alpha^{\text{sym}}_{t-k}$
  $\mu \leftarrow \emptyset$
  $\gamma \leftarrow \text{Predict}(\alpha^{\text{sig}}_{t-k}, x, t)$
  $\pi \leftarrow \text{Select}\{\pi' \in V_t \mid \text{Verify}(DS_t(x), PS_t(\pi'), \gamma) \neq \emptyset\}$
  if $\pi \neq \perp$ then $\mu \leftarrow Verify(DS_t(x), PS_t(\pi_i), \gamma)$
                    $\gamma \leftarrow \text{Update}(\gamma, PS_t(\pi), DS_t(x))$
  $\alpha(t) \leftarrow \langle x, \pi, \gamma, \mu \rangle$
return $\alpha$
```

If Reacquire fails to find a matching percept, then $\alpha(t)$ contains the predicted signature and the 'null' percept $\perp$. Note that in this case $\alpha(t)$ is not grounded.

Track Take an anchor α defined for $t - 1$ and extend its definition to t. It is used in the special case of reacquisition whenever the object is kept under constant observation. Prediction is in general much simpler than in the Reacquire case, and verification is only made with respect to the previously perceived attributes via a domain dependent function MatchSignature. This functionality could for instance be implemented with a Kalman filter.

```
procedure Track $(\alpha)$
  $x \leftarrow \alpha^{\text{sym}}_{t-1}$
  $\gamma \leftarrow \text{OneStepPredict}(\alpha^{\text{sig}}_{t-1}, x)$
  $\pi \leftarrow \text{Select}\{\pi' \in V_t \mid \text{MatchSignature}(\gamma, PS_t(\pi')) \neq \emptyset\}$
  if $\pi \neq \perp$ then $\gamma \leftarrow \text{Update}(\gamma, PS_t(\pi), x)$
  $\alpha(t) \leftarrow \langle x, \pi, \gamma, \alpha^{\text{obs}} \rangle$
return $\alpha$
```

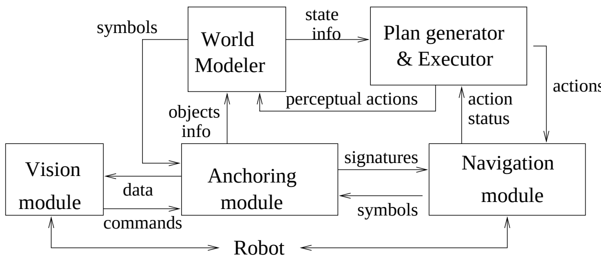

Figure 3: The robot architecture used in our examples.

Notice that the anchor(s) computed always include the best current estimate of the observable properties needed to perform the actions (γ), and an indication of which parts of the symbolic description have actually been observed (μ). The next section will show two examples of use of anchoring that further clarify the role of this information.

5 Examples

In this section we present two examples of anchoring with indefinite references implemented in a robotic system. The robot, a Nomad 200, uses sonars for navigation and vision data to identify objects in the environment. The two-layered decision making architecture of the robot is shown in Fig. 3.

The higher layer includes a plan generator (ETLplan), a plan executor, and a world modeler. ETLplan is a conditional planner capable of generating plans with perceptual actions and conditional branches [Karlsson, 2001]. The plan executor checks the conditions and invokes the actions in the plans generated by the planner. The world modeler maintains information about objects and places relevant for the task. It can obtain additional information about objects from the anchoring module.

The lower layer includes a navigation and a vision module. The navigation module is a simplified version of the fuzzy behavior-based controller defined in [Saffiotti *et al.*, 1995], which executes the actions sent by the plan executor. An example of an action is `(gonear A)`. The vision module contains vision routines for recognizing objects and for calculating properties such as color and size.

The anchoring module provides the connection between the symbols used by the planner and the world modeler, and the perceptual data provided by the vision module and used by the navigation module. It receives requests to create and update anchors and it provides the relevant information about the anchored symbols to the world modeler. It is in this module that the g function is encoded. In addition, the navigation module uses the symbols that constitute the arguments to its actions when requesting information from the anchoring module about the corresponding objects. For instance while executing the action `(gonear A)`, it regularly requests the position of `A`. Finally, the anchoring module controls the vision processing, activating routines for recognizing objects and providing parameters (e.g., expected position) to focus perceptual attention.

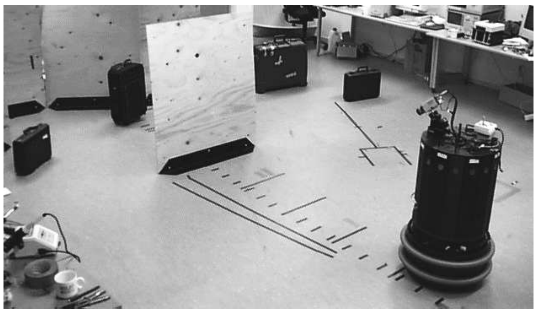

Figure 4: The initial setup in the first example.

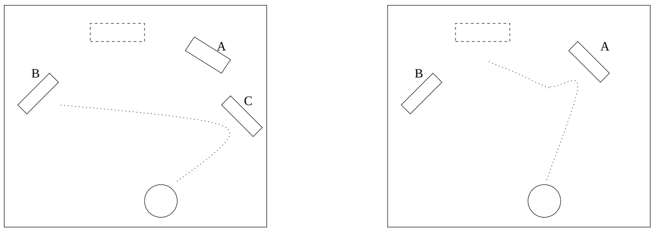

Figure 5: An illustration of the path of the robot in the first (left) and second (right) example.

5.1 Anchoring with partial matching

This example has the aim to show how the anchoring module works in practice and in particular the use of partial matching. It also serves to illustrate one way to handle indefinite references. The robot has the task to "find a black suitcase with a white mark and to go near it". The initial scenario is shown in Fig. 4. The largest suitcase is green, while the other three suitcases are black. The world model contains information about three objects: the two small black suitcases, identified in the world model by the symbols C (the one on the right) and B (the one on the left), and the green suitcase identified by the symbol A.

The goal given to the plan generator has the form `(exists (?x) (and (suitcase ?x) (black ?x) (white-mark ?x) (near ?x)))`, that is, the goal is to be near to a black suitcase with a white mark. The world modeler contains the information that both B and C are black suitcases, but does not have information about the mark. Therefore, the plan generator creates a plan that consists of first going near suitcase C and looking for the mark. If the mark is found, the execution stops with success. Otherwise, the robot goes near to suitcase B to look for the mark. Note that the original indefinite reference ("a black suitcase with a white mark") has now been turned onto two alternative definite references B and C. The actual plan is as follows:

```
((gonear C) (observe C)
 (if ((white-mark C . true)) (:success))
 (if ((white-mark C . false))
     ((gonear B) (observe B)
      (if ((white-mark B . true)) (:success))
      (if ((white-mark B . false)) (:fail)))))
```

Fig. 5 (left) schematically indicates the path followed by the robot during plan execution. The robot goes first to suitcase C, checks the mark (`(observe C)`), does not find it, and goes to suitcase B where it successfully identifies a mark.

The navigation module, while executing `(gonear C)`, is regularly requesting the anchoring module to keep C anchored. The latter uses the track functionality to keep track of C while the robot is moving.

The matching properties provided to the anchoring module are the properties attached to the symbol `C`, such as color, position, and shape, and the fact that the suitcase should have a white mark. The vision routines, due to the distance between the camera and the suitcase, cannot discriminate if there is a mark on suitcase C. A naive matching function might reject this suitcase, given that one property is not satisfied. Partial matching allows us to consider the case that the property is actually not perceivable, and the suitcase is still anchored.

A final observation is that while the robot is moving towards suitcase B, a third black suitcase, which was previously occluded, is also perceived. However, this suitcase is ignored because it does not match the perceptual signature (incl. position) of suitcase B.

5.2 Anchoring with indefinite references

This example shows the handling of indefinite references at the level of the anchoring module, as opposed to at the higher symbolic level in the previous example.

The task is to "go near a green suitcase and then go near a black suitcase". The world model contains initially two suitcases, one green denoted by the symbol A and one black denoted by B. In this case however the plan is to go near a generic object that has the properties of being a green suitcase and then to go near an object that has the properties of being a black suitcase, `((gonear x) (gonear y))`.[2] The difference is that the symbols used for action now do not denote fixed specific objects, but any object in the class of objects satisfying the given matching properties. In case of x, these are black and suitcase. The positions of the two known objects are used as action properties, that is, they are used as an initial value for the `gonear` action. However they are not considered to be matching properties.

The path of the robot is illustrated in Fig. 5 (right). The robot first moves successfully near the green suitcase. However, when it turns towards the black suitcase, this suitcase has been removed. The robot starts moving toward the recorded position of the black suitcase, as its position was given as action property. While the robot moves, a previously occluded black suitcase is perceived: the dotted suitcase in the figure. This suitcase matches the required properties required as it is black. It is therefore anchored and the robot goes near it. The fact that the position and the perceptual signature of this second black suitcase is different from those of the first one does not constitute a problem; the anchoring module just matches the percepts with the properties present in the description state, even if it uses the expected position of the first black suitcase to direct the perception and to provide information to the navigation module.

[2]This plan, and the corresponding symbolic descriptors for `x` and `y`, have been created by hand, because existing planning systems only consider specific, previously known objects.

6 Discussion

The autonomous robotics and machine vision literature contains a few examples in which the need and the role of anchoring, under different names, has been explicitly identified, e.g., [Hexmoor *et al.*, 1993], [Saffiotti *et al.*, 1995], [Jung and Zelinsky, 2000], [Chella *et al.*, 1998], [Bajcsy and Košecká, 1994], [Satoh *et al.*, 1997], [Horswill, 1997], [Wasson *et al.*, 1999]. However, all the works above describe specific implementations and do not attempt a study of the general concept of anchoring.

To our knowledge, [Coradeschi and Saffiotti, 2000] was the first attempt to state the general anchoring problem in formal terms. The main technical contribution of this paper is the extension of the above framework with the introduction of two elements needed to deal with the anchoring of symbols in actions and plans. First we have introduced action properties, that is properties that are used in the execution of an action, but are not relevant for finding the correct object. Second, we have introduced partial matching to be able to distinguish between properties that are not satisfied and properties that are not presently available for perception, that is one cannot presently determine whether they are satisfied or not. As we consider anchoring from an action perspective we can deal with the latter case actively, for instance by observing the object from a better position. Anchoring in the presence of partial matching is actually tentative: if a property previously not observed is observed at a later point and it is discovered not to match the description, the anchor can be removed.

The problem of connecting linguistic descriptions of objects to their physical referents has been largely studied in the philosophical and linguistic tradition. These traditions provide a rich source of inspiration for the conceptualization of the anchoring problem. For instance [Russell, 1905] made a distinction between definite and indefinite references. To this respect our examples show two different ways to resolve an indefinite reference in the context of an action. In the first example, the planner resolves the indefinite reference (*a* black suitcase with a mark) by instantiating a variable to known matching objects in the world model, and then it uses definite references (B and C) in its actions. Note that in order to handle the appearance of a new suitcase, it would be necessary to re-plan. In the second example, the plan contains indefinite references, which are sent as such to the anchoring module. Instantiation is delegated to the anchoring module, which is able to shift "on the fly" to the new suitcase. Note that resolution of ambiguities might be more difficult in this case, without the support of the high level reasoning.

The first approach seems to be suitable for relatively static domains where there is the time to re-plan, while the second approach seems to be more appropriate in highly dynamic domains. In fact, we have used a similar approach in the Sony AIBO robots in the RoboCup competition [Saffiotti and LeBlanc, 2000]. This domain is highly dynamic, but there are few ambiguities with respect of anchoring of objects.

Acknowledgements This work was funded by the Swedish KK Foundation. We thank Lars Karlsson and Zbigniew Wasik for providing substantial help in running the robot experiments.

References

[Bajcsy and Košecká, 1994] R. Bajcsy and J. Košecká. The problem of signal and symbol integration: a study of cooperative mobile autonomous agent behaviors. In *Proceedings of KI-95*, LNCS, pages 49–64, Berlin, Germany, 1994. Springer.

[Chella *et al.*, 1998] A. Chella, M. Frixione, and S. Gaglio. An architecture for autonomous agents exploiting conceptual representations. *Robotics and Autonomous Systems*, 25:231–240, 1998.

[Coradeschi and Saffiotti, 2000] S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 129–135, Austin, 2000.

[Harnard, 1990] S. Harnard. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[Hexmoor *et al.*, 1993] H. Hexmoor, J. Lammens, and S. C. Shapiro. Embodiment in GLAIR: A grounded layered architecture with integrated reasoning for autonomous agents. In *Proc. of the Florida AI Research Sympos.*, pages 325–329, 1993.

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

[Jung and Zelinsky, 2000] D. Jung and A. Zelinsky. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots journal*, 8(3), 2000.

[Karlsson, 2001] Lars Karlsson. Conditional progressive planning: a preliminary report. In B. Mayoh, J. Perram, and H. Lund, editors, *Proceedings of the Scandinavian Conference on Artificial Intelligence 2001*, 2001.

[Russell, 1905] B. Russell. On denoting. *Mind*, XIV:479–493, 1905.

[Saffiotti and LeBlanc, 2000] A. Saffiotti and K. LeBlanc. Active perceptual anchoring of robot behavior in a dynamic environment. In *IEEE Int. Conf. on Robotics and Automation*, pages 3796–3802, 2000.

[Saffiotti *et al.*, 1995] 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.

[Saffiotti, 1994] A. Saffiotti. Pick-up what? In C. Bäckström and E. Sandewall, editors, *Current trends in AI Planning*, pages 266–277. IOS Press, Amsterdam, NL, 1994.

[Satoh *et al.*, 1997] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in video by the integration of image and natural language processing. In *Proc. of IJCAI-97*, pages 1488–1493, 1997.

[Wasson *et al.*, 1999] G. Wasson, D. Kortenkamp, and E. Huber. Integrating active perception with an autonomous robot architecture. *Robotics and Autonomous Systems*, 26:175–186, 1999.