



spondence between this framework and the implementation of the anchoring functionalities in two systems: an autonomous helicopter for traffic surveillance, and a mobile robot performing navigation tasks.

## Anchoring in the literature

Although anchoring as defined in this paper has not been the subject of previous rigorous investigation, issues related to anchoring have been discussed in the fields of autonomous robotics, machine vision, linguistics, and philosophy.

The autonomous robotics literature contains a few examples in which the need and the role of anchoring, under different names, has been explicitly identified, e.g., (Hexmoor, Lammens, & Shapiro 1993), (Hutber *et al.* 1994), (Konolige *et al.* 1997), (Schlegel *et al.* 1999). Jung and Zelinsky (2000) use a similar concept to achieve grounded communication between robots. None of these works, however, pursue a systematic study of the anchoring problem. Bajcsy and Košecká (1994) offer a general discussion of the links between symbols and signals in mobile robotic systems. Yet, they do not deal with the problem of how to create and maintain these links, which is the main issue in anchoring.

The machine vision community has done much work on the problems of object recognition and tracking. While anchoring relies on these underlying perceptual abilities, it is mainly concerned with the integration of these with a symbol system. Some work in vision has explicitly considered the integration with symbols. Satoh *et al.* (1997) present a system which associates faces and names in news videos looking at co-occurrences between the speech and the video streams. Horswill’s Ludwig system (Horswill 1997) answers natural language queries by associating linguistic symbols to markers and to marker operations in the image space. Interestingly, Ludwig may refer to physical objects using indexical terms, like “the block on the red block” (Agre & Chapman 1987). Markers are also used by Wasson *et al.* (2000) to provide a robot with a perceptual memory similar to the LPS of (Konolige *et al.* 1997). Markers are similar to the “anchors” that we introduce in this work. However, all the works above describe specific implementations and do not attempt a study of the general anchoring concept.

The problem of connecting linguistic descriptions of objects to their physical referents has been largely studied in the philosophical and linguistic tradition, e.g., (Frege 1892), (Russell 1905). In fact, we have borrowed the term *anchor* from situation semantics (Barwise & Perry 1983), where this term denotes an assignment of variables to individuals, relations, and locations. These traditions provide a rich source of inspiration for the conceptualization of the anchoring problem, but they typically disregard the formal and computational aspects necessary to turn these ideas into techniques.

The literature on the philosophical foundations of AI presents a wide debate on a problem which is related

to anchoring: the *symbol grounding problem*, first stated by Harnard (1990). Symbol grounding is the problem of how to give an interpretation to a formal symbol system that is based on something that, contrary to classical formal semantics, is not just another symbol system. Anchoring is an important, concrete aspect of symbol grounding: connecting symbolic representations of specific objects to the perceptual image of these objects.

## A computational theory of anchoring

The goal of this section is to make the pre-theoretic notion of anchoring given in the Introduction a precise one. We proceed as follows. We consider an agent that includes a symbol system and a perceptual system. Moreover, we assume the existence of a correspondence  $g$  between predicates in the former and *properties* measured by the latter. We do not make any assumption about the origin of  $g$ : for instance,  $g$  can be hand-coded by the designer of the system, or it can be learnt by the system using neural networks. The task of anchoring is to use this  $g$  to create and maintain a correspondence between a symbol used in the symbol system to denote an *object* in the world, and the percepts generated in the perceptual system by the same object. An *anchor* is a reification of this correspondence. Since new percepts are generated continuously within the perceptual system, this correspondence is indexed by time.

### The underlying model

We now give formal definitions to the elements above. These will be illustrated with an example from one of our domains, in which the symbol ‘car-1’ has to be anchored to its perceptual counterpart in a camera image. We first introduce the elements that are time invariant.

- A *symbol system*  $\Sigma$  including: a set  $\mathcal{X} = \{x_1, x_2, \dots\}$  of individual symbols (variables and constants); a set  $\mathcal{P} = \{p_1, p_2, \dots\}$  of predicate symbols; and an inference mechanism whose details are not relevant here.
- A *perceptual system*  $\Xi$  including: a set  $\Pi = \{\pi_1, \pi_2, \dots\}$  of percepts; a set  $\Phi = \{\phi_1, \phi_2, \dots\}$  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  $\phi_i$  is a measurable property of percepts with values in the domain  $D(\phi_i)$ . We let  $D(\Phi) =_{def} \bigcup_{\phi \in \Phi} D(\phi)$ .
- A *predicate grounding relation*  $g \subseteq \mathcal{P} \times \Phi \times D(\Phi)$ , which embodies the correspondence between unary predicates and values of measurable attributes.

The set  $\Pi$  includes all the possible percepts, only some of which are realized at any given moment. The restriction to unary predicates is made here for simplicity and it will be relaxed in the future.

**Example** In our example, we have two individual symbols  $\mathcal{X} = \{car1, car2\}$ , and the predicate set is  $\mathcal{P} = \{car, small, big, red, blue\}$ . The percepts are the

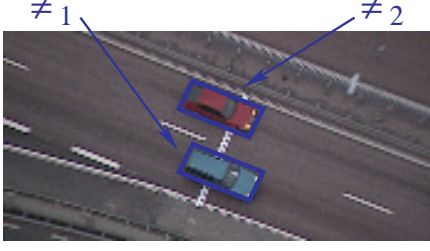


Figure 1: Two percepts  $\pi_1, \pi_2$  in a camera image.

two image regions labeled  $\pi_1$  and  $\pi_2$  in Fig. 1. The set of attributes is  $\Phi = \{type, color, shape\}$ . The domain of *type* is the set of all recognizable objects, e.g., *car*<sup>1</sup>; the domain of *color* is the set of triples of possible hue, saturation, and luminosity values; and the domain of *shape* is the set of pairs of possible length and width values.

The *predicate grounding relation*  $g$  can be seen as a table that encodes the attribute values compatible with a certain predicate. For instance, we have:

$$g(red, color, (h, s, l)) \text{ iff } h \in [-30, 30], s \in [0, 1], l \in [0, 1].$$

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  $\gamma : \Phi \rightarrow D(\Phi)$  is a partial function from attributes to attribute values. The set of attributes on which  $\gamma$  is defined is denoted by  $feat(\gamma)$ .

Intuitively, a symbolic description lists the predicates that are considered relevant to the perceptual identification of an object; and a perceptual signature gives the values of the measured attributes of a percept (and it is undefined for the unmeasured ones).

The  $g$  relation can be used to match a symbolic description  $\sigma$  and a perceptual signature  $\gamma$  as follows.

$$match(\sigma, \gamma) \Leftrightarrow \forall p \in \sigma. \exists \phi \in feat(\gamma). g(p, \phi, \gamma(\phi)) \quad (1)$$

Other definitions for *match* could be used, this would not change the rest of our model.

**Example** An example of a *symbolic description* is  $\sigma_1 = \{red, small\}$ . A *perceptual signature* could be  $\gamma_1 :$

$$\gamma_1 : color \mapsto (10, 1, 1) \text{ and } \gamma_1 : shape \mapsto (8, 4).$$

In this case, we have  $feat(\gamma_1) = \{color, shape\}$ . To see if  $\sigma_1$  and  $\gamma_1$  match, we must evaluate the following

$$match(\sigma_1, \gamma_1) \Leftrightarrow (g(red, color, (10, 1, 1)) \vee g(red, shape, (8, 4))) \wedge (g(small, color, (10, 1, 1)) \vee g(small, shape, (8, 4)))$$

The following is the time dependent part of the model.

- A *description state*  $\Delta_t : \mathcal{X} \rightarrow 2^{\mathcal{P}}$  associates each individual  $x \in \mathcal{X}$  with its symbolic description at time  $t$ .
- A *perceptual state*  $S_t : \Pi \rightarrow (\Phi \rightarrow D(\Phi))$  associates each percept  $\pi \in \Pi$  to its perceptual signature at time  $t$ . If  $\pi$  is not perceived at time  $t$ , then  $S_t(\pi)$  is undefined for every  $\phi \in \Phi$ . The set of percepts which are perceived at  $t$  is denoted by  $V_t$ .

$\Delta_t$  and  $S_t$  are generated by the symbol system and by the perceptual system, respectively, and describe the current properties of the objects of discourse. The decision about what predicates and what perceptual signatures should be present in  $\Delta_t$  and  $S_t$  is domain dependent. For the rest of this section, we assume that the above elements are all given and fixed.

We finally give our central definition.

**Definition 3** An anchor is any partial function  $\alpha$  from time to triples in  $\mathcal{X} \times \Pi \times (\Phi \rightarrow D(\Phi))$ .

An anchor is a unique internal representation of an object  $o$  in the environment. At every moment  $t$ ,  $\alpha(t)$  contains: a symbol meant to denote  $o$ ; a percept generated by observing  $o$ ; and a signature meant to provide the current (best) estimate of the values of the observable properties of  $o$ .

**Example** Suppose that at time  $t$  the *description state* is such that  $\Delta_t : car-1 \rightarrow \sigma_1$ , and the *perceptual state* is such that  $S_t : \pi_1 \rightarrow \gamma_1$ . An anchor connecting these elements would then be  $\alpha : t_1 \mapsto (car-1, \pi_1, \gamma_1)$ .

We denote the components of  $\alpha(t)$  by  $\alpha_t^{sym}, \alpha_t^{per}$ , and  $\alpha_t^{val}$ , respectively. If the object is not observed at time  $t$ , then  $\alpha_t^{per}$  is the ‘null’ percept  $\emptyset$  (by convention,  $\forall t, \emptyset \notin V_t$ ), while  $\alpha_t^{val}$  still contains the best available estimate.

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 (recall that  $V_t$  is the set of percepts which are perceived at  $t$ ).

**Definition 4** An anchor  $\alpha$  is grounded at time  $t$  iff both  $\alpha_t^{per} \in V_t$  and  $match(\Delta_t(\alpha_t^{sym}), S_t(\alpha_t^{per}))$ .

We informally say that an anchor  $\alpha$  is *referentially correct* if, whenever  $\alpha$  is grounded at  $t$ , then the physical object denoted by  $\alpha_t^{sym}$  is the same as the one that generates the perception  $\alpha_t^{per}$ .

**The anchoring problem**, then, is the problem to find referentially correct anchors.<sup>2</sup>

<sup>1</sup>The *type* attribute may be treated in a special way by the perceptual system, since it determines the structure of the percept. In our example, ‘car’ percepts are generated by the vision routines using a model of a car object.

<sup>2</sup>Another property that we may want an anchor  $\alpha$  to satisfy is that the estimated values in  $\alpha_t^{val}$  constitute a *good model* of the corresponding physical object. In this work, however, we concentrate on the reference problem.

## The functionalities of anchoring

In order to solve the anchoring problem for a symbol  $x$ , we need the ability to: (i) create a grounded anchor the first time that the object denoted by  $x$  is perceived; (ii) continuously update the anchor while observing the object; and (iii) update the anchor when we need to reacquire the object after some time that it has not been observed. The following functionalities realize these abilities. ( $t$  denotes the time at which the functionality is called.)

**Find** Take a symbol  $x$  and return a grounded anchor  $\alpha$  defined at  $t$ , and undefined elsewhere. In case of multiple matches, use a domain dependent selection function to choose one. This functionality is summarized by the following pseudo-code.

```

procedure Find ( $x$ )
   $\pi \leftarrow \text{Select}\{\pi' \in V_t \mid \text{match}(\Delta_t(x), S_t(\pi'))\}$ 
  if  $\pi = \emptyset$  then fail
  else  $\alpha(t) \leftarrow \langle x, \pi, S_t(\pi) \rangle$ 
  return  $\alpha$ 

```

Instead of using Select, Find may return one anchor for each matching percept and leave the selection problem to the symbolic system, as we will see in the UAV application.

**Reacquire** Take a symbol  $x$  with an anchor  $\alpha$  defined for  $t - k$  and extend  $\alpha$ 's definition to  $t$ . First predict a new signature  $\gamma$ ; then see if there is a new percept that is compatible with both the prediction and the symbolic description; if so, update  $\gamma$ . Prediction, verification of compatibility, and updating are domain dependent; verification should typically use *match*.

```

procedure Reacquire ( $x$ )
   $\alpha \leftarrow \text{anchor for } x$ 
   $\gamma \leftarrow \text{Predict}(\alpha_{t-k}^{\text{val}}, x, t)$ 
   $\pi \leftarrow \text{Select}\{\pi' \in V_t \mid \text{Verify}(S_t(\pi'), \Delta_t(x), \gamma)\}$ 
  if  $\pi \neq \emptyset$  then  $\gamma \leftarrow \text{Update}(\gamma, S_t(\pi), x)$ 
   $\alpha(t) \leftarrow \langle x, \pi, \gamma \rangle$ 
  return  $\alpha$ 

```

If Reacquire fails to find a matching percept, then  $\alpha(t)$  contains the predicted signature and the ‘null’ percept  $\emptyset$ . Note that in this case  $\alpha(t)$  is not grounded. The Reacquire procedure is used to find an object when there is a previous perceptual experience of it. The prediction function may be complex: it may use domain knowledge like information about occluding objects, and generate multiple hypotheses for the predicted properties. The verify function checks that the attribute values of a percept are compatible with both the predicted  $\gamma$  and the descriptor  $\Delta_t(x)$ . The compatibility criteria are domain dependent

A special case of Reacquire deserves special attention: when the object is kept under constant observation. In this case, Prediction can often be greatly simplified. We define a separate functionality for this case.

**Track** Take an anchor  $\alpha$  defined for  $t - 1$  and extend its definition to  $t$ .

```

procedure Track ( $\alpha$ )
   $x \leftarrow \alpha_{t-1}^{\text{sym}}$ 
   $\gamma \leftarrow \text{OneStepPredict}(\alpha_{t-1}^{\text{val}}, x)$ 
   $\pi \leftarrow \text{Select}\{\pi' \in V_t \mid \text{Verify}(S_t(\pi'), \Delta_t(x), \gamma)\}$ 
  if  $\pi \neq \emptyset$  then  $\gamma \leftarrow \text{Update}(\gamma, S_t(\pi), x)$ 
   $\alpha(t) \leftarrow \langle x, \pi, \gamma \rangle$ 
  return  $\alpha$ 

```

In our experience, these three functionalities, possibly combined together, have been sufficient to solve the anchoring problem in several domains. The next section will show two examples of their use.

## Anchoring in practice

The work in anchoring has originated in our concrete experience with implementations of systems integrating symbolic and perceptual knowledge. The nature of the anchoring problem, in fact, suggests that a general study of it must be solidly grounded in experiments performed on different systems. Here, we show experiments performed on two platforms: a wheeled mobile robot, and a unmanned airborne vehicle (UAV).

The two experimental platforms share the use of a layered architecture to integrate abstract reasoning with perceptual and control processes. However, these platforms differ significantly in terms of sensory-motoric capabilities and domains of application. The mobile robot uses sonars as its main sensor modality and moves in an office environment. The main aspect of anchoring here is the need to link the symbols used by a planner to denote static objects, like corridors and doors, to the sensor data coming from the sonars. The UAV has been developed in the WITAS project (Doherty 1999), and it currently operates in a simulated environment. It uses vision as its main sensor and performs traffic surveillance over urban and rural areas. The main aspect of anchoring here is the need to connect the symbols used by a planner and a plan executor to the sensor data about specific cars provided by the vision system.

## The robot navigation domain

Milou is a Nomad 200 robot equipped with an array of sonar sensors and controlled by an architecture similar to the one reported in (Saffiotti, Konolige, & Ruspini 1995), which includes a simple STRIPS-like planner. All the perceptual and prior information about the robot’s surroundings are maintained in a blackboard-like structure called “Local Perceptual Space” (LPS). Symbolic descriptions, percepts, and anchors are all Lisp structures stored in the LPS.

In terms of our framework, the *symbol system*  $\Sigma$  is given by the planner; individuals symbols ( $\mathcal{X}$ ) denote corridors and doors, and predicate symbols ( $\mathcal{P}$ ) refer to position and width. The *perceptual system*  $\Xi$  extracts linear contours (segments) from consistent sets of sonar measurements; percepts ( $\Pi$ ) include walls (individual

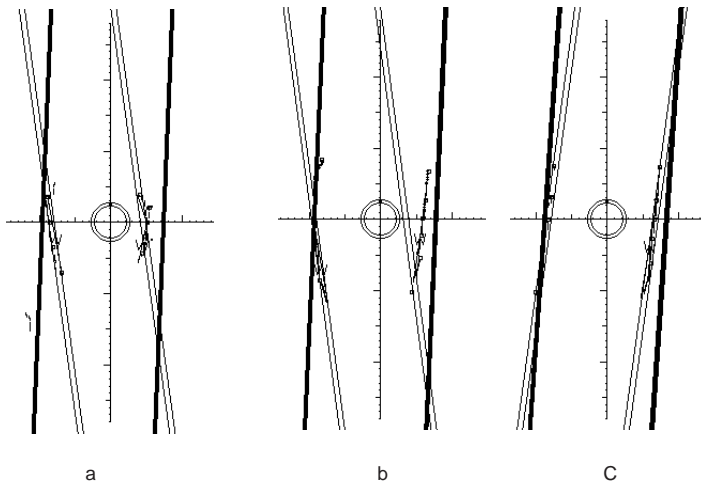


Figure 2: Anchoring a corridor. Thick line: description; ‘w’ segments: percepts; double line: anchor.

segments) and corridors (pairs of parallel segments); attributes ( $\Phi$ ) include position, orientation, length, and width. The *predicate grounding relation*  $g$  is hand-coded. Matching is done according to equation (1), with some extra provisions to take the measurement imprecision into consideration. Finally, an *anchor* is a structure containing pointers to the appropriate symbolic description, percept, and perceptual signature.

The planner puts the symbolic description of the objects to be used for a task into the LPS, based on map information. All the descriptors in the LPS are constantly anchored using **Find** (first time anchoring) and **Track** (afterwards). Both functionalities are implemented according to the general procedures above. In **Track**, prediction of relative position is based on the odometry of the robot, while updating is done by just copying the properties of the percept  $\pi$ . The Verify function is based on geometric distance. Select looks for the percept that best matches the predicted values  $\gamma$ ; if no good match is found, it looks for a percept that best matches the symbolic description.

Fig. 2 shows an example in which Milou recovers from an erroneous initial anchoring of a corridor. The robot is shown in the middle, facing upwards. At time  $t_0$  (a), the planner puts into the LPS the description of a corridor ‘corr-1’ that must be traversed (thick lines). **Find** matches this description with a corridor percept  $\pi_0$  generated by observing a pair of parallel short wall segments (marked by ‘w’), and creates an anchor  $\alpha$  s.t.  $\alpha(t_0) = (\text{corr-1}, \pi_0, \gamma_0)$ , where the value of the position attribute in  $\gamma_0$  is taken from  $\pi_0$ . This is shown by the double lines in the picture. Unfortunately,  $\pi_0$  was generated by spurious sonar readings produced by a peculiar configuration of obstacles. As Milou moves further along the corridor, the **Track** routine predicts the new relative position  $\gamma_1$ ; however, no percept matches this  $\gamma_1$ , and the  $\alpha^{\text{val}}$  is only updated by prediction:

$\alpha(t_1) = (\text{corr-1}, \emptyset, \gamma_1)$ . Note that  $\alpha$  is now ungrounded.

At time  $t_2$  (b), a new percept  $\pi_2$  is generated by the sonars’ observing the actual walls. This percept does not match the new prediction  $\gamma_2$ , but it closely matches the original description. Therefore,  $\pi_2$  is accepted by the Verify function and used to update the anchor (c):  $\alpha(t_2) = (\text{corr-1}, \pi_2, \gamma_2)$ , where the position of  $\gamma_2$  is taken from  $\pi_2$ . The anchor is now referentially correct, and subsequent percepts will easily be matched, thus keeping  $\alpha$  grounded.

## The aerial surveillance domain

The UAV system integrates a planner, a reactive plan executor, a vision system and a control system. Anchoring is done in a dedicated module called Scene Information Manager (SIM) (Coradeschi, Karlsson, & Nordberg 1999). The SIM is intermediate between the plan executor and the vision system and handles the anchoring of symbolic identifiers used in the plan executor to sensory data produced in the vision processing component.

In terms of our formal model, the *symbol system* consists of the planner and the plan executor, both coded in Lisp. Individuals denote cars, while predicates denote linguistic terms (e.g., ‘red’) and positions in a road network (e.g., ‘at-cross1’).

The *perceptual system* is a reconfigurable active vision system able to extract information about objects in aerial images. Percepts are sets of adjacent pixels (regions) each having a HSL (hue, saturation, luminance) value. Attributes of interest include position, length, width, and color (average HSL values) of a region.

The *predicate grounding relation*  $g$  is given as a hand-coded table that associates each predicate to a set of admissible values for the corresponding attribute. For instance, the predicate ‘red’ is associated with a set of admissible HSL values for the attribute ‘color.’<sup>3</sup>

*Symbolic descriptions* are tuples of predicates. The plan executor sends to the SIM the appropriate symbolic descriptions for the individual symbols that have to be anchored. For instance, if the executor is interested in finding Car1, a small red Mercedes at location cross-1, it sends to the SIM the list ‘(Car1 . (red small-Mercedes at-cross1))’. Note that while in the Milou testbed the anchoring module anchors all the objects in the LPS, the SIM only anchors the objects explicitly requested by the executor.

A *perceptual signature* is a list of attribute-value pairs. At each time step  $t$ , the vision system generates a perceptual state  $S_t$  based on information about the objects to look for. For instance, when asked to find all cars around location cross-1, the vision system (i) points the camera to cross-1, (ii) segments the image using a car model, and (iii) returns a set  $V_t$  of found regions, and a perceptual signature for each region.

<sup>3</sup>In the actual system, we use fuzzy sets to take into account vagueness of linguistic descriptions and uncertainty in vision data; we also use a fuzzy matching algorithm to compute a degree of matching (Coradeschi & Saffiotti 1999).

Matching of symbolic descriptions and perceptual signatures is done according to (a fuzzy version of) (1).

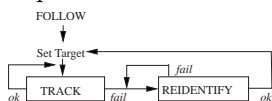
An *anchor* is a Lisp object that stores an individual symbol, the index of a region, and an association list recording the current estimates of the object’s properties (shape, color, etc.)

The functionalities provided by the SIM are *find*, *reidentify*, and *follow* a car, described below. All these functionalities configure the vision system according to the properties of the object of interest.

*FIND*( $x$ ) is implemented by instantiating the general **Find** procedure, with the provision that there is no ‘Select’, but all the found anchors are returned, sorted by their degree of matching.

*REIDENTIFY*( $x$ ) is implemented by instantiating the general **Reacquire** procedure, with the provision that ‘Predict’ generates multiple hypothetical positions according to the configuration of the road network, and considers the effect of known occluding objects (both derived from a GIS).

*FOLLOW*( $\alpha$ ) is implemented by combining the general **Track** and **Reacquire** procedures, which in turn are implemented as **TRACK** and **REIDENTIFY**, as follows.



In **TRACK**, the ‘OneStepPredict’ and ‘Update’ functions are implemented as a Kalman filter (KF), which resides in the vision system. Its initial parameters are set according to the properties in  $\alpha$ . The ‘Verify’ function checks the properties of the  $\gamma$  computed by the KF against a prediction of the possible properties based on domain knowledge. **REIDENTIFY** is repeatedly called when verification fails. It will search for a new percept for the anchor using more complex domain knowledge. When one is found, its properties are used to reset the parameters of the KF, and then continuous **TRACKING** is resumed.

Let us consider an example that illustrates the *FOLLOW* functionality. Two identical cars are present in the image, one traveling along a road which makes a bend under a bridge, and one which travels on the bridge — see Fig 3. The first car is being tracked by *FOLLOW*. The anchor has the form  $\alpha(t_0) = (\text{car-1}, \pi_0, \gamma_0)$  where the perceptual signature  $\gamma_0$  stores the attributes of  $\pi_0$ , in particular color, shape and position. At  $t_1$  the car disappears under the bridge and the second car is almost in the position in the image where the first car would have been, had it not been occluded. The perceptual signature is updated, in particular the expected position of the car is extrapolated from the previously stored position. The percept provided by the vision is the region containing the car over the bridge. The Verify function compares the attributes of this percept with the updated perceptual signature. Given the information about the road network, the percept is discarded. Notice that the KF in **TRACK**, left by its own, would

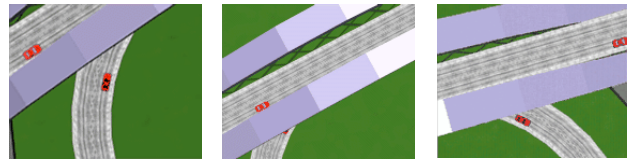


Figure 3: The followed car disappears under a bridge and a similar car appears at its place over the bridge.

start tracking the car on the bridge. **REIDENTIFY** is called to try to find the car again. The anchor is ungrounded until an appropriate percept is found while the perceptual signature continues to be updated at every time point also using the knowledge about the road network. The **REIDENTIFY** uses knowledge about objects that can possibly occlude part of the road to detect the presence of the bridge. The vision system is directed toward the first visible position after the bridge. When the car reappears from under the bridge, a percept is generated by the vision system that is compatible with the perceptual signature present in the anchor. The anchor is grounded and the perceptual signature is updated with the attributes of the newly found percept. Normal tracking is then resumed.

## Discussion

This paper makes two main contributions: (1) it defines the new concept of *anchoring* as a necessary component of any physically embedded symbolic system; and (2) it gives the basic ingredients needed to define the anchoring behavior of such a system, in terms of a formal model and of a set of functionalities. These functionalities rely on a number of building blocks, some of which, like the “match” and the “Predict” functions, may be complex. The study of these blocks is the subject of fields like pattern recognition, visual tracking, or estimation theory. Anchoring, by contrast, is concerned with the often underestimated problem of providing the semantic link between a perceptual system and a symbol system. It does so by combining these blocks into a general, coherent algorithmic structure.

In order to guarantee the generality of our theory, it is essential to test it on several applications. In this paper, we have done so in two substantially different ones. An interesting outcome has been the ability to deal with difficult tracking situations, like the bridge scenario, that require the integration of perceptual and symbolic knowledge. We believe that many of the systems discussed in the literature section could also be reformulated in our framework. We are currently working in applying our theory to other domains, including: the use of non-standard sensors like an artificial tongue and nose; the correspondence between a human provided map and a map built by a wheeled robot; and the RoboCup domain using the Sony legged robots (Saffiotti & LeBlanc 2000). In the first two cases the predicate grounding relation  $g$  is not hand-coded, but it is

automatically acquired by the robot.

Our formalization is still preliminary, since in order to make this first step we had to ignore a number of subtle issues that are hidden in the anchoring problem. First is the issue of uncertainty. Perceptual information is inherently affected by uncertainty due to noise in the sensors, to poor observation conditions, and to errors in the perceptual interpretation. Anchoring should take this uncertainty into account, and consider the quality of the matching, for instance to decide to get a better view of an object. Second, we should consider the possibility to anchor a symbol to multiple percepts: this would be necessary in order to fuse the information coming from different sensors that observe the same objects. Finally, we should distinguish between *definite* descriptions, like “the bottle of Barolo on the table”, *indefinite* descriptions, like “a bottle of Barolo”, and *indexical* descriptions, like “the bottle on my left”. These descriptions need different treatments. For instance, seeing two bottles of Barolo on the table is not a problem in the case of an indefinite description, but could be a problem in case of a definite one. Some of these issues have been investigated in our previous work, but a more formal treatment is needed.

We have not given a formal definition of the notion of an anchor being “referentially correct” in this first step. This is intentional. The theory presented here is internal to the agent, and can be computed by the agent. By contrast, referential correctness is an external notion, that should be studied by the designer once a formal model of the agent and one of the environment are available. Analyzing this notion is an important goal, that we leave as a future step.

**Acknowledgements** We are indebted to Dimiter Drankov, Ivan Kalaykov, and Lars Karlsson for fruitful discussions. This work has been partly funded by the Wallenberg Foundation, and partly by the Swedish KK foundation.

## References

- Agre, P., and Chapman, D. 1987. Pengi: an implementation of a theory of activity. In *AAAI-87*, 268–272.
- Bajcsy, R., and Košecák. 1994. The problem of signal and symbol integration: a study of cooperative mobile autonomous agent behaviors. In *Proceedings of KI-95*, LNCS, 49–64. Berlin, Germany: Springer.
- Barwise, J., and Perry, J. 1983. *Situations and Attitudes*. The MIT Press.
- Coradeschi, S., and Saffiotti, A. 1999. Anchoring symbols to vision data by fuzzy logic. In Hunter, A., and Parsons, S., eds., *Qualitative and Quantitative Approaches to Reasoning with Uncertainty*, LNAI. Berlin, Germany: Springer. 104–115.
- Coradeschi, S.; Karlsson, L.; and Nordberg, K. 1999. Integration of vision and decision-making in an autonomous airborne vehicle for traffic surveillance. In Christiansen, H. I., ed., *Computer Vision Systems*, 216–230. Berlin, Germany: Springer.
- Doherty, P. 1999. The witas integrated software system architecture. Linköping Electronic Articles in Computer and Information Science, Vol. 4 (1999): no. 17. <http://www.ep.liu.se/ea/cis/1999/017>.
- Frege, F. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik* 25–50.
- Harnard, S. 1990. The symbol grounding problem. *Physica D* 42:335–346.
- Hexmoor, H.; Lammens, J.; and Shapiro, S. C. 1993. Embodiment in GLAIR: A grounded layered architecture with integrated reasoning for autonomous agents. In *Proc. of the Florida AI Research Sympos.*, 325–329.
- Horswill, I. 1997. Visual architecture and cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2):277–292.
- Hutber, D.; Moisan, S.; Shekhar, C.; and Thonnat, M. 1994. Perception-interpretation interfacing for the Prolab2 road vehicle. In *7th Symp. on Transportation Sys.: theory and Application of Advanced Technology*.
- Jung, D., and Zelinsky, A. 2000. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots* 8(3). In press.
- Konolige, K.; Myers, K.; Ruspini, E.; and Saffiotti, A. 1997. The Saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence* 9(1):215–235.
- Russell, B. 1905. On denoting. *Mind* XIV:479–493.
- Saffiotti, A., and LeBlanc. 2000. Active perceptual anchoring of robot behavior in a dynamic environment. In *IEEE Int. Conf. on Robotics and Automation*.
- Saffiotti, A.; Konolige, K.; and Ruspini, E. H. 1995. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence* 76(1-2):481–526.
- Saffiotti, A. 1994. Pick-up what? In Bäckström, C., and Sandewall, E., eds., *Current trends in AI Planning*. Amsterdam, Netherlands: IOS Press. 266–277.
- Satoh, S.; Nakamura, Y.; and Kanade, T. 1997. Name-it: Naming and detecting faces in video by the integration of image and natural language processing. In *Proc. of IJCAI-97*, 1488–1493.
- Schlegel, C.; Illmann, J.; Jaberg, H.; Schuster, M.; and Wötzt, R. 1999. Integrating vision based behaviours with an autonomous robot. In Christiansen, H. I., ed., *Computer Vision Systems*, LNCS, 1–20. Springer.
- Wasson, G.; Kortenkamp, D.; and Huber, E. 2000. Integrating active perception with an autonomous robot architecture. *IEEE Trans. on Robotics and Automation*. To appear.