Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem?

Andrea Bonarini, Matteo Matteucci, Marcello Restelli
Politecnico di Milano Artificial Intelligence and Robotics Project
Department of Electronics and Information – Politecnico di Milano, Milan, Italy
{bonarini,matteucc,restelli}@elet.polimi.it

Abstract

We propose a model to represent knowledge in an agent possibly participating to a multi–agent system, such that the anchoring problem can be faced by well known techniques. We discuss how this model could reliably support the instantiation of concepts by aggregating percepts affected by uncertainty, sensed by several sensors, and obtained from different agents.

Introduction

We present a model of knowledge needed for anchoring in embodied agents, and a procedure to carry on anchoring. A formal definition of anchoring and his solution has been presented in (Coradeschi & Saffiotti 2000) by Coradeschi and Saffiotti; we present an alternative approach to achieve flexibility and overcome some issues emerging from their approach. We consider this contribution as a further step towards a sound formalization of the anchoring problem, which puts in evidence how assessed AI techniques can support its solution.

The agents that we are considering operate in an environment containing other *physical objects*. These are perceived by *smart sensors* that translate some physical features acquired by *physical sensors*, into information (*features*) represented by an internal formalism compatible with the other modules of the agent. When this information is used to maintain a model of the environment, it is important that features referring to a physical object be collected around an internal representation of the same object, referred to as its *perceptual image*. The problem of relating a perceptual image with the corresponding physical object is part of the more general problem known as *Symbol Grounding Problem* (Harnad 1990). In particular, it is relevant to relate the perceptual image with a *concept*, by instantiating it, since this makes it possible to reason about the perceptual image with categories typical of the corresponding concept. An embodied agent *anchors* a physical object when it can instantiate a concept compatible with the perceived features, and maintain the relationship between such an instance and the physical object it refers to during its activities.

We decided to study the anchoring problem in order to design a model and realize a tool that could be used in a very wide range of robotics applications. These applications often require an anchoring system, that, in general, is hidden in the code and provides only an ad-hoc solution. We have tried to build a general, flexible and extensible anchoring system, taking in consideration some central issues in robotics such as: noisy measurements, uncertainty and partial information. We did not adopt the model proposed by Coradeschi and Saffiotti since it had some drawbacks that we have overcome by changing the approach to the problem:

- in their symbolic system there must be a symbol for each object that has to be anchored; this enumerating approach does not scale and it is not suited for applications with a large (eventually indefinite) number of objects
- since the symbols are not bound by any kind of relation it is not possible to use inferential mechanisms in the anchoring process; e.g. if I know that handles are placed on doors or windows, I will never anchor an object on the floor with the symbol handle
- a perceived object is anchored to a symbol only if it matches all the predicates in the symbolic description. In this way, if descriptions are too detailed it will hard, with poor sensors, to match them with perceived objects; on the other hand, general description may lead to anchor the same perceived object with different symbols
- it is not explained how their model could be used with information coming from different sensors
- it is not explained how they deal with uncertainty

Recently they had submitted a further work on this topic (Coradeschi & Saffiotti to appear) in which they extend their model in order to solve some of the problems above mentioned. Although some motivations do not yield anymore, we think that our model is more flexible and rigorous under a formal point of view.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

In the next section, we introduce a general model to represent concepts and instances, and a procedure to relate them to the corresponding physical objects through the respective perceptual image. The aim of this modeling proposal is to provide a basis to make the knowledge needed to the anchoring activity explicit, so that it will be possible to implement standard libraries to realize anchoring activities. As we will see, this is enabled by standard, largely available AI techniques. The availability of such libraries will make the development of autonomous agents much easier than now, by supporting a rapid diffusion of this technology in the society.

We also discuss the role of physical sensors in this process; in particular, the acquired information should be enough to assign features to concept instances with at least a certain degree of reliability. Some problems are related to this activity; among them, we mention here the *Perceptual Aliasing* problem (Whitehead & Ballard 1991), where sensor information is not enough to discriminate between two situations where the agent should take different actions. Poor sensors may not be able to provide the information needed to assign a percept to the proper perceptual image, thus affecting the anchoring performance. This problem is reduced by designing appropriate smart sensor systems, with the support of the proposed modeling approach. Object tracking, another component of anchoring affected by perceptual aliasing, can be supported by introducing some model to consider past features and infer from them the missing information.

The anchoring process is also important to provide the agent controller with reliable data: instead of considering directly the input data, the controller may be more robust by taking its input from a conceptual model, maintained by anchoring, which is more reliable than simple data (Bonarini, Matteucci, & Restelli Submitted).

The use of a symbolic model enables data fusion among different sensors and agents operating in related environments. Again, symbols are more reliable than numerical data to be shared, and make it possible the implementation of compact models and their effective fusion. The higher the abstraction level of the symbol, the higher its reliability and its expressive power. In the third section, we discuss how our model applies to multi-agent systems, by focusing on sharing knowledge, and on how agents can take into consideration the information coming from the others, possibly maintaining different models of the same physical objects, for different purposes.

Knowledge representation and anchoring

In this section, we first motivate anchoring, then we introduce a model of the knowledge needed by the agent to operate and the anchoring procedure, which maps the real world to an effective internal representation.

Motivations for symbolic models

In this presentation, our main motivation is to put in evidence the knowledge needed for the anchoring process, independently from the specific application, thus opening the way towards the definition of general libraries able to face the various aspects of the anchoring activity.

Some of the more important properties that can be obtained by basing anchoring on a knowledge model are summarized here below.

- *noise filtering*: using a conceptual model of the environment it is possible to filter out–layers and noisy data coming from sensors; this produces more reliable information for the other modules controlling the agent
- *sensorial coherence*: by fusing sensorial information referring to the same objects, but coming from different sources, it is possible to check the coherence of perception, thus enhancing fault tolerance and enabling on–line diagnosis
- *virtual sensing*: the model of the environment can be used to infer new features, not perceived by physical sensors, to be added to the local map by sensor fusion; these new features can be considered as perceived by virtual sensors, and are homogeneous to the physical ones
- *time consistency*: the instances in the conceptual model represent a state of the environment; it is possible to maintain and monitor its consistency in time; this activity can be used to learn and check models, i.e., detect whether the selected model is correct despite changes in a dynamical environment)
- *abstraction*: the use of conceptual instances (see next sections) instead of raw data, or features, in the behavior definition gives more abstraction in designing robot behaviors, and robustness to noisy data; it also facilitates design, since gives the designer the possibility to reason in symbolic terms. Moreover, the information exchange is effective when agents of a Multi-Agent System (MAS) share the same semantics for symbols.

Model structure

We structure the conceptual model as most of the classical AI hierarchical models, usually implemented by classes, or frames (Minsky 1975), or some logical formalism. The definition of concepts at different levels of abstraction is important to support the classification of percepts, and the instantiation of concepts. Properties related to concepts provide additional information once a perceptual image has been related to a concept. Concepts are organized in *ontologies*, which may be partially defined independently from the specific application, at least up to a certain abstraction level. For instance, it is possible to give general properties of *movable objects*, as a concept specializing the more general *objects*, and in turn specialized by *mobile robots* and *human beings*. Such general concepts may also participate in general inferential processes, which allow, for instance, to infer that people and mobile robots usually stay on the ground, information useful, for instance, to compute distances from images. When facing an application, it is then possible to complement the general ontology with application-specific information; for instance, we may already know that balls are spherical, but in a Robocup (Asada *et al.* 1999) robot soccer application we also know that the ball is red and has a given dimension.

During the robot activity, data coming from sensors are matched against general knowledge in the conceptual model, and, when enough evidence is collected, a concept instance is generated, thus inheriting by default properties eventually not detected by sensors. In case of uncertainty, it is often more reliable to instantiate a more general perceptual image. For instance, again in a Robocup application, robots belonging to different teams wear different markers, which may be detected with some uncertainty. Therefore, a set of features may be aggregated more reliably as an instance of *robot*, than as an instance of *opponent robot*.

From the design point of view, the presence of a reference model makes it possible a modular design, with people having different competence interacting on the same knowledge, if needed. So, people working on sensors know that they should produce certain information in a given format, and people working on control can rely on the presence of such data. Moreover, it is also possible to apply pre–defined libraries on the knowledge implemented for the specific application, thus improving software reliability, and reducing design efforts.

The formal model

The knowledge representation model we propose for anchoring is based on the notion of *concept* and its *properties*. We generate an instance of a concept from the perceptual image of a physical object. As already stated, the building of such instance and its tracking in time are the main activities of anchoring.

In a formal way, a property is a tuple

$$p \triangleq < label, \mathbb{D}, \rho >, \tag{1}$$

where *label* denotes the property, $\mathbb{D}$ is the set of all the possible values for that property given a specific representation code (e.g. for the colors we can use the set $\{red, green, blue, \ldots\}$ or the RGB space $\mathbb{N}^3_{[0,255]}$) and ρ represents a restriction of the domain $\mathbb{D}$ for that property in the specific concept:

$$\rho : \mathbb{D}_p \to \wp(\mathbb{D}_p). \tag{2}$$

Two properties p_1 and p_2 are compatible, $p_1 \sim p_2$, if they have the same label and domain or a mapping between the respective domains. A property p_1 includes p_2 if they are compatible and the restriction of domain $\rho_1(\mathbb{D})$ is included in the restriction of domain $\rho_2(\mathbb{D})$:

$$p_1 \subseteq p_2 \Leftrightarrow (p_1 \sim p_2) \wedge (\rho_1(\mathbb{D}) \subseteq \rho_2(\mathbb{D})) \tag{3}$$

A set of properties describes a *concept* C, which is used in our model to represent the knowledge about perceptual images of physical objects. Depending on the concept and on the specific domain a property can be classified as *substantial* or *accidental* (respectively S and A in equation 4). Substantial properties are those properties that characterize the immutable part of a concept; for a given object, their values do not change over time, and they can be used for object recognition since they explain the essence of the object they represent. Accidental properties are those properties that do not characterize a concept, their values for the specific instance can vary over time, they cannot be used for object recognition but are the basis of instance formation, tracking and model validation (see ahead). We admit that perceptions for accidental properties can vary over time.

$$C \triangleq \{< p, \mathtt{x} >\} : \mathtt{x} \in \{\mathtt{S}, \mathtt{A}\}, \tag{4}$$

For sake of ease we define the set of properties for a concept as

$$P_i \triangleq \{p : p \in C_i\}, \tag{5}$$

and the partial function ϕ, defining the type of property:

$$\phi : \bigcup_i (C_i \times P_i) \to \{\mathtt{A}, \mathtt{S}\}. \tag{6}$$

The *extension* $\epsilon(C)$ of the concept C is the set of objects to which the concept applies (i.e. all possible instances of that concept). The *intension* $\iota(C)$ of the concept C is the set of properties which a concept involves in itself and which cannot be modified without changing it (i.e. all substantial properties).

Example: concepts We have applied the described model to our robotic agents operating in the RoboCup domain, in the F-2000 league, where a team of four robots play soccer against another four (Asada *et al.* 1999). Each robot should recognize at least the ball, other robots, goals and walls; the established knowledge of an agent contains the concepts of: *object*, *ball*, *playground*, *wall*, *robot*, *ally*, *opponent*, *defensive–goal*, *attacking–goal*. For each concept we define the substantial and accidental properties (e.g., the concept *ball* is characterized by its substantial properties such as shape – round –, dimension – 20 cm –, color – red –, and its accidental properties, such as position and velocity).

The fundamental structural relationships between concepts are *specialization* and *generalization*. We say that a concept C_2 specializes a concept C_1 (we denote that with $C_2 = \sigma(C_1)$) when it is defined by a superset of C_1 properties and compatible properties are included. The formal definition of this specialization is

$$(\iota(C_2) \supseteq \iota(C_1)) \wedge (\epsilon(C_2) \subseteq \epsilon(C_1)), \tag{7}$$

and operatively it is translated in

$$(P_2 \supseteq P_1) \ \wedge (p_i \supseteq p_j) \quad \forall p_i \in P_1, \forall p_j \in P_2. \tag{8}$$

Generalization is the inverse of specialization, so a concept C_2 generalizes C_1 when it has a subset of C_2 properties and compatible properties are subsumed. Concepts C_2 and C_3 that specialize C_1 do that in a *partial* way or a *total* way; we have a total specialization iff

$$\epsilon(C_2) \cup \epsilon(C_3) = \epsilon(C_1), \tag{9}$$

otherwise we have a partial specialization. Concepts C_2 and C_3 that specialize C_1 do that in a *overlapping* way or a *exclusive* way; we have an exclusive specialization iff

$$\epsilon(C_2) \cap \epsilon(C_3) = \emptyset, \tag{10}$$

otherwise we have a overlapping specialization. In *object oriented* models and languages specialization is generally referred as *inheritance*.

Using concepts it is possible to describe the knowledge – domain specific and not – used by the agent during the anchoring process. To explain how this knowledge is used, we introduce the notion of *model*: given $\mathcal{D}$ as the set of the known domains, a model $\mathcal{M}_d$ is the set of all the concepts known by the agent referring to the specific domain $d \in \mathcal{D}$, linked by relationships – structural and domain specific. A relationship between concepts can represent:

1. *constraints*: they must be satisfied by concept instances in order to belong to the model
2. *functions*: they generate property values for a concepts from property values of another (inference function)
3. *structures*: structural constraint to be used while reasoning about classification and uncertainty

We have previously described an example of the third type of relationship presenting the specification of a concept

$$\sigma \subseteq \{C\} \times \{C\} \tag{11}$$

we will describe how this can be used reasoning about uncertainty in next section.

The whole *established knowledge* of an agent is thus defined as:

$$\mathcal{M} = \bigcup_{d \in \mathcal{D}} \mathcal{M}_d. \tag{12}$$

For instance, the model of the environment $\mathcal{M}_E \subseteq \mathcal{M}$ is the knowledge (concepts and relationships) concerning the perceivable objects in the environment.

In the RoboCup domain we exploit all kind of relationships proposed in our model: constraints (e.g., there is only one *ball* instance and its position is into the *playground*), functions (e.g, the position of an object can be computed from the relative position of that object with respect to another object at known position), and structures (e.g., *ally* and *opponent*, in this domain, are a total and exclusive specialization of *robot* that is a specialization of the generic concept *object*).

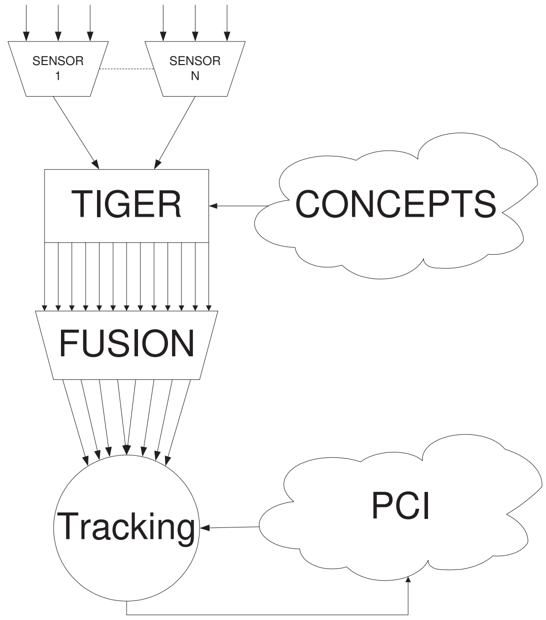

Figure 1: The anchoring process

The anchoring process

The environment is perceived by a situated agent as a collection of concept instances. The property values for these instances are sensed by means of *smart sensors*, which analyze percepts and give an interpretation of such percepts at a higher level of abstraction with the aid of basic domain knowledge. As in any embodied system, information extracted by sensors can be affected by noise; reasoning at an higher level of abstraction can be useful in order to reduce the influence of noisy data on the anchoring process by taking in account *sensorial uncertainty*.

Sensors produce a description of perceptual images in terms of sets of features: one set of features for each perceived object. Each feature f is represented as a pair

$$f \triangleq < label, value >, \tag{13}$$

where *label* is the symbolic name of the property the feature refers to, and *value* is the value of the feature belonging to an appropriate set of possible values $\mathbb{D}$.

This activity can be considered as the *symbol grounding* (Harnad 1990) phase in the anchoring process: percepts are interpreted as symbolic features to be classified as concept instances and maintained in time. The symbol grounding is demanded to smart sensors in order to confine the uncertainty due to sensorial acquisition as close as possible to the source and to enhance information about the percepts with a measure of reliability.

These sets of feature are processed by a module called TIGER (Temporary Instance GEneratoR). It exploits the knowledge contained in the model of the environment $MathcalM_E$ in a *classification* process *gamma* in order to detect a concept matching the perceived object, thus producing conceptual instances with the related degree of reliability:

$$\gamma : \wp(\{f\}) \rightarrow \{\overline{C}\} \times \Re_{[0,1]}. \quad (14)$$

In doing this, we use as classification criteria the *degree of matching* of the features perceived by sensors about the object and the substantial properties of the concepts. After classification, all knowledge associated to the concept, substantial and accidental properties, can be used to reason about the concept instance.

The matching degree among concepts and features describing an object in the environment can be computed by a *pattern matching* algorithm (Apostolico & Galil 1997) that takes into account that only partial descriptions of perceptual images may be available, that only substantial properties are relevant, and that not all properties have the same expressive power in describing a concept. Pattern matching for a concept C is done on the intension $\iota(C)$ of the concepts but only instances belonging to the extension $\epsilon(C)$ of the concepts can be accepted.

Pattern matching, as symbol grounding, has to deal with uncertainty, since the classification process can be affected by noisy features, partial descriptions of objects, and partial contradictions among concept properties or in relationships among instances. We define θ the function that associates a concept instance to its reliability value:

$$\theta : \{\overline{C}\} \rightarrow \Re_{[0,1]}. \quad (15)$$

Concept instances related to more intensive concepts are less reliable, since they require the matching of more features:

$$\iota(C_1) \subseteq \iota(C_2) \Rightarrow \theta(\overline{C_1}) \geq \theta(\overline{C_2}) \quad (16)$$

Reasoning about uncertainty can be useful to drive pattern matching and classification, since we can reduce the uncertainty associated to an instance of a concept by means of structural relationships properties such as generalization. Moreover, different objects in the environment, originating the same features, can be associated to the same instance of a concept, in this case we need to solve a *perceptual aliasing* problem by enhancing the sensorial apparatus of the agent.

More abstract are the input data, more general are the modules that use them. For this reason we introduced two layers at the beginning of the anchoring process: smart sensors and the TIGER. Smart sensors transform raw data in sets of features, thus giving a uniform interface, common to every kind of sensor: it doesn't matter if we have a sonar or a camera, since smart sensors return sets of features. The TIGER rises further the level of abstraction of input data: from sets of features to conceptual instances, so that the following modules can work with data that may come from different sources (e.g. communication with other agents, virtual sensing, etc.). This will be more evident in the following section, when we will extend the model to the multi–agent domain.

Since the same physical object may be perceived at the same time by distinct sensors, the output of the TIGER may contain several conceptual instances that are related to the same physical object. The FUSION module merges those conceptual instances that are supposed to be originated from the perception of the same object by different sensors, i.e. *sensor fusion* (Murphy 1996). The fusion of concept instances can be achieved through clustering techniques (Fisher & Langley 1985). In literature are present many approaches to the clustering problem (Baraldi & Blonda 1999). We define cluster a set of concept instances related to concepts whose extensions present a non-null intersection and have "similar" values for the accidental properties. The meaning of similar is embodied by the following partial function, defined only for compatible properties:

$$\delta : \rho_{p_1}(\mathbb{D}) \times \rho_{p_2}(\mathbb{D}) \mapsto \{\texttt{true}, \texttt{false}\}, \quad (17)$$

i.e., a function that given two values of compatible properties p_1 and p_2 returns true if the related concept instances can coexist in the same cluster, false otherwise.

Two concept instances $\overline{C}$s ($\overline{C}_1$ and $\overline{C}_2$) can belong to the same cluster if:

1. their accidental properties are similar:
$$\forall p_i \in P_1, \forall p_j \in P_2 : (p_i \sim p_j) \wedge$$
$$(\phi(C_1, p_i) = \texttt{A}) \wedge (\phi(C_2, p_j) = \texttt{A}) \Rightarrow \delta(\overline{p_i}, \overline{p_j}) = \texttt{true}$$
2. they are originated from different sources
3. the respective concepts are not mutually exclusive, i.e.
$$\epsilon(C_1) \cap \epsilon(C_2) \neq \emptyset.$$

For instance, *person* and *man* are two compatible concepts, while *man* and *woman* cannot belong to the same cluster; moreover instances of concepts like *woman* and *soldier* can belong to the same cluster since some women are soldiers

A new merged concept instance ($\overline{MC}$) is extracted for each cluster, and its accidental properties are deduced from the accidental properties of the cluster elements by a fusion process that takes into consideration also their reliability values. A $\overline{MC}$ is an instance of the most intensive concept among those relative to the $\overline{C}$s belonging to the cluster and its reliability is evaluated by combining their reliability values, so that more numerous clusters will produce more reliable $\overline{MC}$s.

From all instances of concepts $\overline{C}$s and the model $\mathcal{M}_E$ it is possible to infer new concept instances using relationships between concepts representing specific knowledge for the application domain. We define as instance of the environment model $\overline{\mathcal{M}}_E$ the set of all concept instances either derived from the anchoring process or from inference on concept instances that are compatible with the relationships contained in the model itself:

$$\overline{\mathcal{M}}_E \equiv \{\overline{C} : C \in \mathcal{M}_E\}. \quad (18)$$

The *state* of the system represented by the model instance $\overline{\mathcal{M}}_E$ is the set of all values of accidental properties – time variant and not – of concept instances belonging to the model itself.

The *tracking* phase of anchoring consists of maintaining in time a coherent state of the model and a correct classification of instances. This phase tries to match the conceptual instances perceived in the past with those generated by the latest perception. In doing that, accidental properties have to be monitored in time using state prediction techniques such as linear regression or Kalman filtering (Kalman 1960). During tracking three possible actions can take place in the $\overline{\mathcal{M}}_E$:

- *inserting*: the conceptual instance processed is new, and has to be inserted in the $\overline{\mathcal{M}}_E$ associated with an identifier
- *innovating*: the conceptual instance processed is the temporal evolution of one of those contained in the $\overline{\mathcal{M}}_E$; the accidental properties are updated and its reliability is increased
- *obsoleting*: when a conceptual instance in the $\overline{\mathcal{M}}_E$ has not been innovated by any conceptual instance among those processed; its reliability is decreased

Example In RoboCup domain many robots are equipped with sonar and vision systems. The smart sensor associated with the sonar produces features related to the position of perceived objects, while the smart sensor of the vision system produces features related to position, color and dimension of perceived objects. In a simplified example (without considering reliability values) the TIGER module could receive:

<sonar
<obj 1 < ρ 100> < θ 180> >
<obj 2 < ρ 230> < θ 315> > >

and

<vision
<obj 1 < ρ 250> < θ 313> <color red> <dimension 18> > >

Here ρ is a distance in centimeters and θ an angle in degrees. From this input data TIGER produces two instances of the concept *object* (*object* is a special concept that is not described by any substantial property) for the sets of features sent by the sonar, and an instance of the concept *ball* for the set of features sent by the vision system. Since *ball* is a specialization of *object*, the FUSION module merges the conceptual instances generated by *obj 2* from sonar and *obj 1* from vision, with 240 cm as ρ and 314 deg as θ.

As already stated, the model we are presenting in this section can be considered as the logical basis for anchoring, but it is also suitable for classical activities that an embodied agent has to accomplish:

- *sensor fusion*: features perceived by different sensors can be aggregated if they refer to the same object in the environment; this is done to collect as much as possible information about objects, to avoid perceptual aliasing, and to reduce noise using redundancy in sensorial perception
- *self–localization*: we consider *self–localization* as the process of instantiating the environment model, thus obtaining $\overline{\mathcal{M}}_E$. This definition is a generalization of the common notion of self–localization (Borenstein, Everett, & Feng 1996) since it enables reasoning about the own position not only in terms of geometrical model, but also in terms of more general knowledge (features)
- *virtual sensing*: the instantiation of a model of the environment can be used to produce new information applying *state estimation* techniques to theoretical knowledge about the model. This new information can be seen by the agent as new *virtual* features produced by sensors looking at the model of the environment instead than considering the environment itself

Extension to MAS

So far, we have dealt with world modelling processes in a single–agent architecture. It is expected that in a multi–agent context each agent could take advantage of data perceived by its teammates. Having the opportunity to combine different local representations, it is possible to build a shared viewpoint of the common environment, that we call *global representation*. In doing this we suppose that each agent shares the same ontology containing *global concepts* (GC).

The global representation builder receives as input the instances of models produced by the local processes. As we have already stated in previous section each model instance contains a set of instances of concepts (e.g., wall, robot, person, etc.). The agent having those instances in its $\overline{\mathcal{M}}_E$ is the *source* and specifies a reliability value associated to the anchoring process, considering reliability of sensors in the operating conditions, pattern matching, and so on. The anchoring process for the global representation is the same described in the previous section, starting from the FUSION module. In this case we have no sensors, or better we have special sensors (i.e. agents) producing conceptual instances instead of sets of features.

A global representation gives to the MAS some interesting qualities (that justify the flourishing of several recent works about this topic (Hugues 2000)(Nakamura *et al.* 2000)(Jung & Zelinsky 2000)):

- *robustness*: the information coming from several agents that are working together in a given environment can be referred to the same physical objects; the global representation exploits such redundancy to validate global concept instances
- *extensive sensing*: a MAS is comparable to a *super-agent* able to sense and act at the same time in different places. In this regard, the agents of a MAS can be considered as virtual sensors supplying the

super-agent with information at a high level of abstraction to be merged in the global representation. In this way, even if an agent cannot sense some physical objects, these may be perceived by other agents. By sharing such information, each agent is provided with an extensive (virtual) sensorial apparatus that enables it to get a more complete knowledge about its working environment

- *fault tolerance*: the global representation can be used to identify and correct sensorial faults; in fact, by comparing concept instances from different agents it is possible to notice discrepancies. For instance, an error in the localization of an agent affects the property "position" of its concept instances. By using the global representation it is possible to recover such an error, by inferring the robot position so that its local representation matches those of the other agents. The effectiveness of this process is influenced by the number of agents that share perception about the same physical objects
- *cooperation*: in a MAS it is easier to achieve coordination by reasoning on a global model shared by the agents. Since the global representation is the result of a merge phase executed on the local representations produced by the agents, a coordination process that works on it gets the authority for assigning activities to each agent, thus avoiding a complex negotiation phase (Bonarini & Restelli Submitted).

It is important to point out that the global representation is not built in order to substitute the local representation, but to supplement this by providing for lacking information and by recovering possible errors. Each agent weights the proper way of integrating the information coming from the global representation, just like it does with information coming from any sensor; for this reason we refer also to the global representation as a *virtual sensor*. In this way, we exploit both the accuracy and the autonomy supplied by the local representation (useful for execution purposes) and the completeness and robustness of the global one (useful for coordination purposes).

The overall architecture

What we have presented above is integrated in a behavior–based architecture augmented with a model of the environment merging the characteristics of behaviors – in terms of reactivity, autonomy, reliability and separation of concern at design time – with world modeling facilities such as those mentioned here below.

The framework we have presented concerns the *world modeling* part of a wider cognitive reference model (Radermacher 1996) that we adopted to design the architecture of our agents. It is organized in four layers each representing a different level of abstraction in information from *raw data* to *theories and models* (figure 2).

- 1^{st} layer (*data layer*): extracts basic features from raw data streams as a basis of eventual understanding; this layer is the only agent's cognitive interface with the environment; the other layers work on features extracted by this one or on abstract information elaborated by other agents when present
- 2^{nd} layer (*concept layer*): interprets and combines features to identify basic concepts at a higher level of abstraction (*symbolic concepts*) using previously established domain knowledge
- 3^{rd} layer (*knowledge processing layer*): processes symbolic concepts to obtain more complex information to be used for planning actions, communicating and other reasoning activities; on this layer *symbolic concepts* are used to select actions to be executed and to infer *structured knowledge*
- 4^{th} layer (*theory and models layer*): contains structured abstract models and theories to be used in information processing by the 2^{nd} and 4^{th} layers; this is the repository of established domain knowledge.

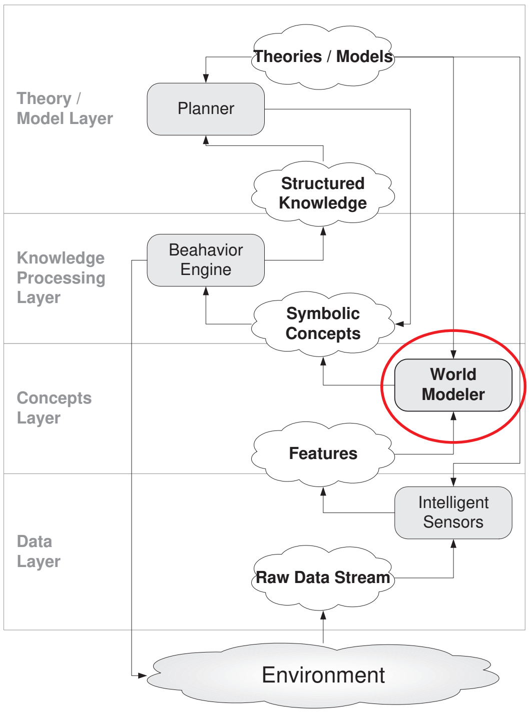

Figure 2: The overall architecture

Intelligent sensors implement the data layer extracting basic features from raw data streams. For instance, omnidirectional vision (Bonarini 2000)(Bonarini, Aliverti, & Lucioni 2000) provides relative positions and distance of colored objects or the angular position of vertical edges in the environment for autonomous robot applications.

The *World Modeler* operates at the 2^{nd} layer to instantiate and maintain a conceptual model of the environment, anchoring symbolic concepts to features extracted from data by means of theoretical knowledge belonging to the 4^{th} layer.

Executive modules in the behavioral component of the architecture belong to the *Behavior Engine* (Bonarini *et al.* Submitted), which uses symbolic predicates valued on concept instances to propose actions for the agent. Complex predicates are built from the ground ones by using logical operators to describe information at a higher level of abstraction, and to enable inference.

The *Planner* (Bonarini & Restelli Submitted) reasons on models in the 4^{th} layer to produce goals as symbolic input to the *Behavior Engine*; these goals are represented by complex predicates used in selecting and blending actions from different behavioral modules. Models in the 4^{th} layer can be abstract maps, graph representations, concept hierarchies, or concepts themselves, used to forecast or simulate the result of future actions.

In this architecture, the world modeler builds a local representation instancing a model of the environment $\overline{\mathcal{M}}_E$, realizing sensor fusion, anchoring, and self–localization. Moreover, it is possible to enhance $\overline{\mathcal{M}}_E$ with information coming from other agents in a multi–agent system obtaining, in such a way, global distributed sensing.

Answering the question ...

In this paper we have presented a formal model for anchoring that puts in evidence how the anchoring problem is neither a new problem nor needs for new solutions. Although a model for anchoring has been already proposed with some applications by Saffiotti and Coradeschi (Coradeschi & Saffiotti 2000)(Coradeschi & Saffiotti 1999), we propose an alternative modeling approach to support the implementation of a general framework for anchoring, also suitable for sensor fusion and multi–agent communication.

The main advantages of the novelties introduced by the proposed solution to the anchoring problem are

- *general unifying model*: our approach allows the comprehension of relationships between anchoring and classical AI notions such as symbol grounding, pattern recognition, state estimation, and clustering
- *separation of concerns*: we propose a framework on automatic instancing of a symbolic model of the environment from sensorial percepts given a correct formalization of knowledge about the environment thus separating the design of sensorial apparatus from the control architectures
- *integrability of domain knowledge*: the use of classic knowledge representation languages suitable for expert system, allows to insert domain knowledge deriving also from human experts

As we have already stated we have applied this model to the RoboCup domain. This domain can be classified as *loosely connected*, due to the sensorial capability of our agents and the particular environment. In fact, although all our robots are equipped with omnidirectional vision (Bonarini 2000), they cannot always perceive everything is happening on the field, because of image resolution, and partial occlusions due to other robots. Each robot maintains its local instance of the model, which provides enough information for reactive behaviors; moreover, it exchanges with teammates information aimed at the maintenance of a distributed global model instance. We have made experiments where a robot has been purposely blinded and it is localized by other robots, which inform it about its position and that of ball, goal and obstacles. Of course, the performance of the blind robot could not be optimal, but it was able to intercept the ball when playing as a goal keeper, and to kick the ball in goal when playing as an attacker.

We consider our model as a possible general formalization of the anchoring problem, putting in evidence how assessed AI techniques can support its solution. Moreover, the framework we are presenting is suitable for triggering special, active sensing behaviors, working as active anchoring processes (Saffiotti & LeBlanc 2000). Finally, we have presented how it can be easily applied to multi–agent system applications increasing robustness, reliability, and fault tolerance.

References

Apostolico, A., and Galil, Z. 1997. *Pattern Matching Algorithms.* New York , NY: Oxford University Press.

Asada, M.; Kitano, H.; Noda, I.; and Veloso, M. 1999. Robocup: today and tomorrow – what we have learned. *Artificial Intelligence Journal* 110:193–214.

Baraldi, A., and Blonda, P. 1999. A survey of fuzzy clustering algorithms for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 29:786–801.

Bonarini, A., and Restelli, M. Submitted. An architecture for multi-agent systems. *IEEE Transactions on Robotics and Automation.*

Bonarini, A.; Aliverti, P.; and Lucioni, M. 2000. An omnidirectional vision sensor for fast tracking for mobile robots. *IEEE Transactions on Instrumentation and Measuring* In press.

Bonarini, A.; Invernizzi, G.; Labella, T. H.; and Matteucci, M. Submitted. A fuzzy architecture to coordinate robot behaviors. *Fuzzy Sets and Systems.*

Bonarini, A.; Matteucci, M.; and Restelli, M. Submitted. A framework for robust sensing in multi–agent systems. In *Proceedings of Robocup 2001.*

Bonarini, A. 2000. The body, the mind or the eye, first? In M. Veloso, E. Pagello, M. A., ed., *RoboCup 98–Robot Soccer World Cup III*, 695–698. Berlin, D: Springer Verlag.

Borenstein, J.; Everett, H. R.; and Feng, L. 1996. *Navigating Mobile Robots: Systems and Techniques.* Wellesley, MA: A. K. Peters, Ltd.

Britannica. 2001. www.britannica.com.

Coradeschi, S., and Saffiotti, A. 1999. Anchoring symbols to vision data by fuzzy logic. *Lecture Notes in Computer Science* 1638:104–112.

Coradeschi, S., and Saffiotti, A. 2000. Anchoring symbols to sensor data: Preliminary report. In *AAAI/IAAI*, 129–135.

Coradeschi, S., and Saffiotti, A. to appear. Perceptual anchoring of symbols for action. In *Proccedings of the 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*.

Fisher, D., and Langley, P. 1985. Approaches to conceptual clustering. In *Proc. of the 9th IJCAI*, 691–697.

Harnad, S. 1990. The symbol grounding problem. *Phisica D* 42:335–346.

Hugues, L. 2000. Grounded representations for a robots team. In *in Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2248–2253.

Jung, D., and Zelinsky, A. 2000. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots* 8(3):269–292.

Kalman, R. 1960. A new approach to linear filtering and prediction problems. *Transactions ASME, Series D, Journal Basic Engineering* 82(3):35–45.

Minsky, M. 1975. A framework for representing knowledge. *The Psychology of Computer Vision* 211–277. McGraw-Hill.

Murphy, R. 1996. Biological and cognitive foundations of intelligent sensor fusion. *IEEE Transactions on Systems, Man, and Cybernetics* 26(1):42–51.

Nakamura, T.; Ebina, A.; Imai, M.; Ogasawara, T.; and Ishiguro, H. 2000. Real-time estimating spatial configuration between multiple robots by triangle and enumeration costraints. In *in Proceedings of 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2048–2054.

Radermacher, F. J. 1996. Cognition in systems. *Cybernetics and Systems* 1 – 41.

Saffiotti, A., and LeBlanc, K. 2000. Active perceptual anchoring of robot behavior in a dynamic environment. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 3796–3802.

Whitehead, S. D., and Ballard, D. H. 1991. Learning to perceive and act by trial and error. *Machine Learning* 7:45–83.