Perceptual Anchoring via Conceptual Spaces

Antonio Chella
University of Palermo, Italy
chella@unipa.it

Silvia Coradeschi
Örebro University, Sweden
silvia.coradeschi@aass.oru.se

Marcello Frixione
University of Salerno, Italy
frix@dist.unige.it

Alessandro Saf£otti
Örebro University, Sweden
asaf£o@aass.oru.se

Abstract

Perceptual anchoring is the problem of creating and maintaining in time the connection between symbols and sensor data that refer to the same physical objects. This is one of the facets of the general problem of integrating symbolic and non-symbolic processes in an intelligent system. Gärdenfors' *conceptual spaces* provide a geometric treatment of knowledge which bridges the gap between the symbolic and sub-symbolic approaches. As such, they can be used for the study of the anchoring problem. In this paper, we propose a computational framework for anchoring based on conceptual spaces. Our framework exploits the geometric structure of conceptual spaces for many of the crucial tasks of anchoring, like matching percepts to symbolic descriptions or tracking the evolution of objects over time.

Introduction

Perceptual anchoring is the problem of how to create and maintain in time the right correspondence between symbols and sensor data that refer to the same physical objects (Coradeschi & Saf£otti 2000; 2003). Perceptual anchoring is an important aspect of the connection between symbolic and sensory based processes in an embedded intelligent system, e.g., an autonomous robot. An example is the problem of connecting the symbol used by a planner to refer to an object needed for an action, say 'cup-22', to the data that correspond to that object in the sensori-motoric system. This connection must be dynamic since the same symbol must be associated to new entities in the perceptual stream in order to track the object over time or to re-acquire it at a later moment. Anchoring can be seen as an important special case of symbol grounding (Harnard 1990) where the symbols denote individual physical objects.

Perceptual anchoring is but one of the facets of the general problem of integrating symbolic and non-symbolic processes in an intelligent system. *Conceptual spaces* have been recently introduced as a way to bridge the gap between symbolic and sub-symbolic AI (Gärdenfors 2000) by providing a geometric treatment of concepts and knowledge representation. A conceptual space has dimensions that are related with the concepts managed at the symbol level as well as with the quantities processed by the sensors. Conceptual spaces allow us to represent discrete concepts, which are the main entities manipulated at the symbol level, inside a structure where we can place continuous observable quantities, which are the main entities provided by the perceptual system. Conceptual spaces therefore provide an intermediate representation in which both symbolic and sensor-based information can be integrated. In addition, conceptual spaces are endowed with a geometric structure that allows to perform topology- and similarity-based reasoning inside the space itself. They are therefore well suited to formalize the types of reasoning needed for perception, as demonstrated by their use in several vision applications (Chella, Frixione, & Gaglio 1997). Because of these reasons, it was pointed out in (Chella, Frixione, & Gaglio 2003) that conceptual spaces could offer a fruitful setting for the study, formalization and implementation of perceptual anchoring.

In this paper, we develop a computational framework for perceptual anchoring based on conceptual spaces. This framework builds on the one proposed in (Coradeschi & Saf£otti 2000), reformulated in a conceptual space setting. As it turns out, the new framework brings a number of advantages over the original one. First, it clari£es the integration between perceptual and symbolic information since both types of information are represented in the same formal structure. Second, it clari£es the dynamic aspect of anchoring by modeling objects as trajectories in the conceptual space. Third it allows us to replace most of the domain-dependent functions used in the original framework by generic functions that exploit the geometric structure of the conceptual space. For example, in a conceptual space, the problem of matching a perceived object to a symbolic description becomes a simple test for set inclusion between the point representing the perception and the region representing the symbolic description.

In the next section we outline the proposed framework in general terms. We then translate it in formal terms, discuss its use in the larger perspective of an embedded intelligent system, and illustrate it by a simple example run on a mobile robot equipped with a symbolic planner and a vision system.

The Conceptual Framework

Conceptual spaces

Gärdenfors has introduced *conceptual spaces* in (Gärdenfors 2000). Roughly, a conceptual space is a metric space whose

dimensions, called *qualities*, are related with the quantities processed by the robot sensors. Examples of dimensions are color coordinates (HSV) and spatial coordinates.

Points in a conceptual space, called *knoxels*, represent the epistemologically primitive elements at the considered level of analysis. For instance, a knoxel can represent an individual object, which is characterized by a given value for each dimension of the conceptual space. The distance $d(k_1, k_2)$ between two knoxels k_1 and k_2, according to the given metric, is interpreted as a measure of similarity between the entities represented by k_1 and k_2.

Concepts are represented by regions in a conceptual space: a concept corresponds to the region of the space in which are located the points that are considered instances of that concept. A special role is played by so called natural concepts, which correspond to convex regions in a conceptual space. For a natural concept c, we can select a knoxel k_c as a prototype: points closer to k_c correspond to "more typical" instances of the concept.

We said that individual *objects* can be represented by knoxels in the conceptual space. However, in a dynamic perspective, objects can be more pro£tably seen as *trajectories* in the conceptual space indexed by time. The properties of objects usually change with time: objects may move, an object can alter its shape or color, and so on. As the properties of an object are modi£ed, the point representing it in a conceptual space moves, and describes a certain trajectory. Several assumptions can be made on this trajectory, e.g., smoothness, and obedience to physical laws.

The interest of conceptual spaces for our purposes is that they constitute an intermediate level between the symbolic system and the perceptual system. From the perceptual side, knoxels in a conceptual space can represent the entities coming from the perceptual system, together with their measured attributes. These knoxels are abstractions of the sensor data, since they represent a summary of the information regarding a certain object coming from different sensors. For instance, a knoxel can represent the information about the position, size and color of a given door as measured by a laser system plus a vision system. From the symbolic side, the conceptual space can be seen as an internal semantics for the symbol system. Predicates in the symbol system are mapped to region of the conceptual space, and individual constants are mapped to knoxels. This semantics is perceptually grounded, since the elements of the conceptual space are directly related to perception.

Anchors

The task of anchoring is to create and maintain in time the correspondence between symbols and sensor data that refer to the same physical objects. In terms of conceptual spaces, this correspondence can be seen as a link between a symbol that denotes a given physical object in the symbol system, and a knoxel that represents the entity provided by the perceptual system when observing that object. We call such a link an *anchor*.

Fig. 1 illustrates this connection. The conceptual space in this example only has the two qualities Hue and Size. The meaning of the g and h functions will become clear later

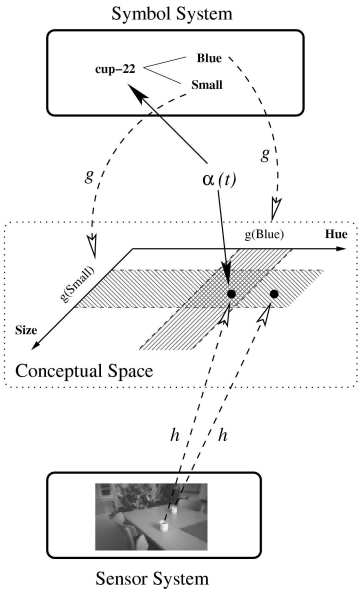

Figure 1: Anchoring the symbol 'cup-22'.

in this paper. The anchor, denoted by $\alpha(t)$, connects the symbol 'cup-22' to a knoxel derived by the observation of a given cup. The concepts Blue and Small constrain the region in the conceptual space where this knoxel can be found.

Once we have an anchor, we must update it in order to keep the symbol aligned to the corresponding perceptual data as those data change with time. The anchor should therefore account for the object persistence in face of a ¤ow of different knoxels from the perceptual system that all originate from the same object, and in face of changes in the properties of the object, e.g., its position. To do so, we need to include a *state* in the anchor. We take then an anchor to be triple $\langle x, k, p \rangle$ where x is an individual constant, p is a knoxel coming from the perceptual system, and k is a knoxel that represents the system's knowledge of the current state (properties) of the object.

The anchor's state can be seen as an internal model, at the conceptual level, of the physical object. This model summarizes both symbolic information and perceptual information about the object. As the anchor evolves in time, its state describes a trajectory in the conceptual space. It is the task of the anchoring process to update this state so that it approximates as well as possible the actual properties of the object.

Figuring out the evolution of an object (its future position, or the way in which its features are going to change) can be seen as the extrapolation of a trajectory in a conceptual space. To identify again an object that has been occluded for a certain time interval amounts to interpolate its past and present trajectories. In both cases, the anchoring process can make hypotheses concerning the evolution of the object using the geometric properties of the corresponding trajectory. Several sources of information can be used to constrain the trajectories in the conceptual space, including symbolic knowledge, smoothness hypotheses, and physical laws.

Fig. 2 illustrates the update of the anchor shown in Fig. 1. The new state of the anchor (empty circle) is predicted from

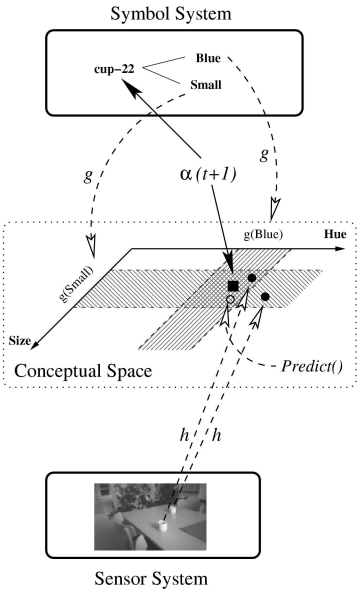

Figure 2: Updating an anchor over time.

the previous one by assuming no change in size and color. Two new knoxels (£lled circles) are generated by the perceptual system. These may be different from the ones generated at the previous time step because of sensor noise. Both percepts are rather close to the predicted state. However, only one of them is acceptable given the symbolic constraints "Blue" and "Small". This is used to update the prediction, leading to the new state shown by a black square.

The information contained in the anchor integrates symbolic and perceptual knowledge over time. This information can be used by the different components of the intelligent system to perform reasoning and action. For instance, the position information can be used by a controller to steer the robot toward an intended object. Interestingly, this information is available even when the object is not in view.

The Formal Framework

We now translate in formal terms the conceptual framework outlined above. We assume to have a *symbol system* Σ and a *sensor system* S that the anchoring system should connect. The details of these systems are not important, but we presuppose two things:

- Σ includes a set $\mathcal{X}$ of individual symbols, like 'cup-22', and a set $\mathcal{P}$ of predicate symbols, like 'blue'. We call a set σ of unary predicates a *symbolic descriptor*. Symbolic descriptors can be used to qualify individuals, like a {large, red, cup}.
- S produces at each time t a measurement vector $\mathbf{v}_t \in V$, e.g., a color image or a laser scan.

Ingredients

Anchoring uses the following three main ingredients.

1 A *conceptual space* K including qualities $\{q_1, \ldots, q_m\}$ with values in the domains $K_1, \ldots, K_m$ respectively. We call any point $\mathbf{k} = \langle k_1, \ldots, k_m \rangle$ in K a *knoxel*.

Examples of qualities are HSV values, shape parameters, and spatial coordinates.

The conceptual space is meant to provide an intermediate space where we can represent both the information coming from S and the one coming from Σ. To do so, we need two additional ingredients that establish the relation between K and S and between K and Σ, respectively.

2 A *conceptual sensor model* $h : V \to 2^K$ that associates each measurement in S with a set of knoxels.

The h function transforms a measurement vector from the sensor system into a set of knoxels in the conceptual space. Each knoxel corresponds to an object that has been observed, together with the observed values of its qualities. For instance, if $\mathbf{v}_t$ is a color image containing a red box and a white door, then $h(\mathbf{v}_t)$ may return a pair of knoxel whose HSV qualities correspond to red and white, respectively.

Intuitively, h processes the raw data coming from the sensory system in order to extract and summarize the data to be available to the anchoring system[1]. For example, in (Chella, Frixione, & Gaglio 1997) h performs a 3D reconstruction of the perceived objects in a 11-dimensional conceptual space.

3 A *predicate grounding function* $g : \mathcal{P} \to 2^K$ that associates unary predicates in Σ to areas in the conceptual space.

The g function gives semantics to symbolic predicates in terms of observable qualities in the conceptual space. For example, in Fig. 1 above, the predicate 'blue' is associated to the area of the conceptual space consisting of all the knoxels that have the Hue value within a given range, and no restriction for the other qualities.

The g function can be extended to symbolic descriptors in the obvious way: if $\sigma \in 2^{\mathcal{P}}$ is a symbolic descriptor, then $g(\sigma)$ is the set of all the knoxels that satisfy all the properties in σ, i.e., $g(\sigma) = \bigcap_{p \in \sigma} g(p)$.We also denote by $[\sigma]_g$ the knoxel in the conceptual space that represents the *prototype* of the descriptor σ. In the simplest case, $[\sigma]_g$ can be computed by taking the center of mass of the set $g(\sigma)$.

We now de£ne the central notion of an *anchor*, a data structure connecting Σ and S via the conceptual space K.

De£nition 1 *An* anchor *is a partial function α from time to triples in $\mathcal{X} \times K \times K$.*

Thus, an anchor α associates at every time t a triple $\alpha(t) = \langle x, k, p \rangle$ where: $x \in \mathcal{X}$ is an individual symbol denoting a given object in the symbol system; $p \in h(\mathbf{v}_t)$ is a knoxel in K corresponding to an object perceived by the sensor system at time t; and k is a knoxel in K that represents the current (estimated) state of the physical object denoted by x and perceived as p. We denote by $\langle \alpha^x, \alpha^k, \alpha^p \rangle$ the three components of an anchor α. The reason why α is a partial function is that anchors do not live forever, but they can be created and destroyed. Fig. 1 above illustrates all the given ingredients.

An anchor effectively formalizes a time-dependent link between a symbol in Σ and the perceptual data from S that

[1] Currently, we do not represent the uncertainty in the measurements in h. This will be part of our future work.

refer to the same physical object. The core of this link, which guarantees temporal consistency, is the state information about that object, stored in the conceptual space as α^k. As time evolves, $\alpha^k(t)$ describes a trajectory in the conceptual space K. The task of the anchoring process is to maintain this trajectory as close as possible to the trajectory that describes the actual physical object inside K, as described above. The anchoring process does so via the following functionalities.

Functionalities

Their use will be illustrated and clari£ed in the next section. The **Find** functionality takes as arguments a symbol that needs to be anchored, together with a symbolic descriptor, and returns an anchor for that symbol. If a matching anchor with no symbol is already present, it uses that anchor, otherwise it creates a new one.

Find Take a symbol x and a symbolic descriptor σ, and return an anchor α de£ned at t (and possibly unde£ned elsewhere). This functionality is summarized by the following pseudo-code.

procedure Find (x, σ, t)
  $A \leftarrow \{\alpha \mid \alpha^x(t) = \perp \ \wedge \ \alpha^k(t) \in g(\sigma)\}$
  if $A \neq \emptyset$
    then $\alpha \leftarrow \arg\min_{\alpha \in A} d(\alpha^k(t), [\sigma]_g)$
      $\alpha(t) \leftarrow \langle x, \alpha^k(t), \alpha^p(t) \rangle$
    else $\alpha \leftarrow$ *NewAnchor*()
      $P \leftarrow \{p \in h(\mathbf{v}_t) \mid p \in g(\sigma)\}$
      if $P \neq \emptyset$
        then $p \leftarrow \arg\min_{p \in P} d(p, [\sigma]_g)$
          $\alpha(t) \leftarrow \langle x, p, p \rangle$
        else $\alpha(t) \leftarrow \langle x, [\sigma]_g, \perp \rangle$
  return α

First, we check if there is any anchor that is not currently associated to a symbol ($\alpha^x(t) = \perp$) and whose state satis£es the symbolic descriptor ($\alpha^k(t) \in g(\sigma)$). If so, we select the one closest to the prototype $[\sigma]_g$ of σ according to the distance function d, associate it to symbol x, and return. Otherwise, we create a new anchor, and collect all the percepts currently generated by the sensor system ($h(\mathbf{v}_t)$) that satisfy the symbolic descriptor σ. We then select the percept p closest to $[\sigma]_g$ and put it in the anchor. We use p to initialize the state information. If no satisfying percept was found, we initialize the state information using the prototype of σ.

Note that situations of ambiguity, like multiple matching percepts, are solved by exploiting the metric structure of the conceptual space. Alternatively, Find might return one distinct anchor for each matching option, and leave the choice to the symbolic system.

Our second functionality is somehow the dual of Find: it creates a new anchor for a currently perceived object.

Acquire Take a knoxel p, assuming $p \in h(\mathbf{v}_t)$, and return an anchor α de£ned at t and unde£ned elsewhere.

procedure Acquire (p, t)
  $\alpha \leftarrow$ *NewAnchor*()
  $\alpha(t) \leftarrow \langle x, p, p \rangle$
  return α

The **Acquire** functionality allows us to initiate a new anchor whenever we receive an "interesting" percept which does not match any existing anchor. What constitutes an interesting percept is decided by the program calling this functionality: typically, these are percepts that might be later anchored to a symbol for use in the reasoning system.

The £rst two functionalities can create new anchors. In order to continuously update an existing anchor, we use the following functionality.

Track Take an anchor α de£ned for $t-1$ and extend its de£nition to t.

procedure Track (α, t)
  $\sigma \leftarrow$ symbolic descriptor for $\alpha^x(t-1)$
  $\hat{K} \leftarrow$ *Predict*(α^k, t)
  $P \leftarrow \{p \in h(\mathbf{v}_t) \mid p \in \hat{K} \ \wedge \ p \in g(\sigma)\}$
  if $P = \{p\}$
    then $k \leftarrow$ *Update*$(\hat{K}, p)$
      $\alpha(t) \leftarrow \langle x, k, p \rangle$
    else $\alpha(t) \leftarrow \langle x, \hat{k}, \perp \rangle$
  return α

We £rst extrapolate the previous state α^k through the *Predict* function. This returns an area $\hat{K}$ in the conceptual space where the new state is expected to be. Then, we consider those percepts p currently generated by the sensor system which are compatible with both this prediction and the symbolic descriptor σ. If there is just one such percept, we set the new state $\alpha^k(t)$ to a combination of the predicted states $\hat{K}$ and the observation p. Otherwise, we set $\alpha^k(t)$ to the prototype $\hat{k}$ of the prediction $\hat{K}$.

Both the *Predict* and *Update* functions can exploit the metric structure of the conceptual space. For instance, *Predict* can generate a neighborhood of the point computed by linear extrapolation, and *Update* can compute the center of mass of its arguments. Finally, the **Reacquire** functionality is used to re-establish a lost symbol-percept connection, e.g., because the object could not be observed for some time.

Reacquire Take an anchor α de£ned for $t-k$ and extend its de£nition to t.

procedure Reacquire (α, t)
  $\sigma \leftarrow$ symbolic descriptor for $\alpha^x(t)$
  $P \leftarrow \{p \in h(\mathbf{v}_t) \mid p \in g(\sigma)\}$
  $P' \leftarrow \{p \in P \mid$ *Verify*$(p, \alpha^k, g(\sigma))\}$
  if $P' \neq \emptyset$
    then $p \leftarrow \arg\min_{p \in P'} d(p, [\sigma]_g)$
      $\alpha(t) \leftarrow \langle x, p, p \rangle$
    else fail
  return α

This functionality is somehow a combination of the **Track** and the **Find** ones. We £rst select all the current percepts that satisfy the symbolic descriptor. For each such percept p, then, we check whether or not p is a plausible continuation of the previous state α^k by the *Verify* function. This function tries to interpolate a trajectory between α^k and p, which lays inside $g(\sigma)$, also considering any physical constraints on the

trajectory. Finally, the percept closest to $[\sigma]_g$ is selected and used to update the anchor. If no satisfactory percept is found, the **Reacquire** functionality fails.

As in the case of the **Find**, ambiguities are solved inside this functionality by selecting the closest candidate according to the metric structure of the conceptual space. Alternatively, we could return multiple anchors and leave the selection problem to the calling system.

Using the Framework

The above framework gives us the basic ingredients and functionalities needed for anchoring. In this section, we discuss how to use this framework and present the implementation of the system that we have used in our experiments.

Anchors are created using two main mechanisms. The £rst mechanism is Top-down, or goal-driven. This happens, for instance, when the symbolic system needs to anchor a symbol to perceptual data, e.g., in order to make an action executable. Suppose that the symbol system decides to execute the action to cross "door-22". The symbol "door-22" needs to be connected to the perceptual knowledge about the door in order to successfully do the crossing. The anchor can be created either when the connection with the perceptual knowledge is acquired or previously using the prior knowledge about the object. In the case of the door the anchor is £rst created when the planner decides to go to that door. The prior knowledge stored in a map about the door is put in the anchor and it is used by the controller to approach the door. When the door is perceived and recognized as "door-22" the actual perceptual data of the door are put in the anchor and used by the controller for executing the crossing door behavior.

The second mechanism to create an anchor is bottom-up, or event-driven. When the perceptual system perceives an object that is or could be of interest, it creates an anchor using the Acquire functionality. The aim is to keep in memory perceptual information about objects that can be used later on in the anchoring process.

Currently each anchor is considered independently. A better approach that we are going to investigate would consist in £nding the connection simultaneously of all active anchors and available percepts considering the possible dependencies among the anchors.

The process to update an anchor in time to account for the expected changes in properties and to include new observations is performed by the Track functionality. There might be several anchors simultaneously active, one for each object of interest: Track updates all of them.

The anchoring module is calling the Track functionality every time that new perceptual data are received. All perceptual data matching existing anchors are used by the Track for updating the anchors. The Acquire functionality is called for each of the percepts that are not matched to any existing anchor and that represent interesting objects. In the Acquire currently implemented an anchor is created for all new recognizable objects. This simple solution is working because we hard-code in our vision routines which objects can be recognized. However a more general solution would require a mechanism with which the objects of interest are dynamically decided.

In general, Track may have a very dif£cult job. For instance, it is not clear how to perform tracking if there are several matching percepts, or if there is a large amount of uncertainty in the perceived properties. The strategy we currently use is to let the Track functionality recognize the problem, but not try to solve these situations. Instead, a trace of the problems is left in the anchor itself by not updating its percept. If the symbolic system is interested in an anchor and this anchor is not updated, then the symbolic system can decide to take actions in order to be able to have an updated anchor again. For instance a planning system can perform informative actions and call the Reacquire functionality to recover the anchor (Broxvall, Karlsson, & Saf£otti 2004).

The Track and Acquire functionalities are called inside the anchoring module. The Find and the Reacquire functionalities are called by the symbolic system and are creating and maintaining the actual connection between the symbols and the perceptual data.

The Find functionality is used when the symbolic system wants to create the connection between a symbol and a percept. The Find can select one of the already existing anchor or create a new anchor. In principle also anchors already connected to a symbol could be selected, for instance “my cup” and “a red cup” could denote the same object. However, we have not considered this case in our experiments yet and we therefore leave it as an open problem.

The Reacquire functionality is called when the symbolic system needs to have updated information about an object. For the Reacquire to succeed the object needs to be currently perceived while in the Find the object could have been perceived also previously. The two functionalities can be combined together, for instance the Find could be £rst used to £nd an object appropriate for an action and the Reacquire to be sure that the object is currently perceived.

If we consider the anchoring module as part of a larger cognitive system, it can be interpreted as part of the short term memory where perception and prior information (included the one used for prediction) are integrated over time, and which provides the necessary information for action and immediate decision. It interacts with the long term memory by accessing the prior information contained there. The dual problem is how to transfer the information stored in the anchors to the long term memory. Anchors that are not used should be removed from the anchoring module and stored in the long term memory if they can be interesting for future tasks. Prior information could also be revised given the perceptual information stored in the anchoring module. Currently we do not have a mechanism to transfer information to the long term memory. We simply keep all anchors created during the performance of a task in the anchoring module and we remove them when the task is £nished. We intend to explore the relation between long and short term memory in our future work.

An additional interesting issue in the anchoring process is the handling of ambiguous cases. In another paper in this proceedings (Broxvall *et al.* 2004) we propose a system handling anchoring ambiguities using planning. We intend to

Figure 3: Our anchoring scenario.

explore in our future work how Conceptual Spaces can be used to help resolving ambiguities.

A Simple Example

The ideas presented in this paper have been tested in a mobile robot system. The system includes: a conditional progressive planner, PTLplan (Karlsson 2001), that has the capacity to reason about incomplete and uncertain information, and the Thinking Cap (Saf£otti, Konolige, & Ruspini 1995), a fuzzy behavior-based controller including a navigation planner. The system uses sonars to navigate and detect obstacles, and vision to perceive objects. The conceptual space used in our system includes axes for color parameters (HSV), for position, and for shape parameters.

In this example a rescue robot is given the task to go into a room in a building (for which it has a map) and look for victims — see Fig. 3. If there is none, it checks for dangers, including red gas bottles containing explosive gases.

The system generates a plan including the action "Cross(Door3)". It creates an anchor top-down (through Find) for Door3 from the map information and uses it to approach the door. When the door appears in the image a knoxel is created corresponding to the door. The Track functionality realizes that this knoxel can be used to updated the anchor thus completing it with actual perceptual information. The re£ned and more precise information is used to perform the crossing operation. When it is inside the room, the robot looks for victims while exploring around. During this, it sees two gas bottles, and creates two anchors bottom-up (through Acquire). No victim is found. The robot has the prior information that two red gas bottles should be present in the room, Gasbottle1 and Gasbottle2. It does a Find of Gasbottle1 whose symbolic description is {red gas-bottle}. The area of the conceptual space determined by the predicates red and gas-bottle contains the knoxels associated to the two anchors created previously bottom-up. One of the anchors is selected and it is then completed by attributing it the symbol Gasbottle1. Similarly the Find of Gasbottle2 is executed. The £nal task of the robot is to go near Gasbottle1. The controller turns the robot toward the position of the gas bottle. The position information is in the knoxel contained in the anchor. The Reacquire functionality is then executed to make sure that the gas bottle is in view before starting the go near action. During the go near action the Track functionality keeps the anchor updated.

Conclusions

According to the spirit of Gärdenfors' proposal, various forms of reasoning can be pro£tably seen as forms of geometric reasoning performed at the conceptual level. Such a geometric approach to knowledge representation and reasoning is believed to be fruitful for the design of embedded intelligent systems.

In this paper, we have shown that there is much to gain by integrating Gärdenfors' conceptual spaces in the anchoring framework. Conceptual spaces offer an appropriate intermediate level in which both symbolic and sensor-based information can be integrated; they clarify the dynamic aspect of anchoring by modeling objects as trajectories in the conceptual space; and they provide a metrics over concepts that can be used inside the anchoring functionalities. The proposed integration also contributes to the study of conceptual spaces, by investigating how to represent individual objects inside a conceptual space and how to account for their evolution in time. This problem had not been paid much attention until now.

Acknowledgments: we would like to thank Lars Karlsson and Mathias Broxvall for their contribution to the ideas reported in this paper. This work has been funded by the swedish KK foundation and by Vetenskapsrådet.

References

Broxvall, M.; Coradeschi, S.; Karlsson, L.; and Saf£otti, A. 2004. Have another look: On failures and recovery planning in perceptual anchoring. In *Proc. of the AAAI04 Workhsop on Anchoring Symbols to Sensor Data*.

Broxvall, M.; Karlsson, L.; and Saf£otti, A. 2004. Steps toward detecting and recovering from perceptual failures. In *Proc of the 8th Int Conf on Intellgent Autonomous Systems*.

Chella, A.; Frixione, M.; and Gaglio, S. 1997. A cognitive architecture for arti£cial vision. *Artif. Intell.* 89:73–111.

Chella, A.; Frixione, M.; and Gaglio, S. 2003. Conceptual spaces for anchoring. *Robotics and Autonomous Systems* 43(2-3):193–195. Special issue on perceptual anchoring.

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

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

Gärdenfors, P. 2000. *Conceptual Spaces*. Cambridge, MA: MIT Press, Bradford Books.

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

Karlsson, L. 2001. Conditional progressive planning under uncertainty. In *Proc. of the 17th IJCAI Conf.*, 431–438.

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