Proc of the IEEE Int. Conf. on Robotics and Automation (ICRA-2000)
San Francisco, CA, April 2000, pp. 3796-3802

Active Perceptual Anchoring of Robot Behavior in a Dynamic Environment

A. Saffiotti
alessandro.saffiotti@aass.oru.se

K. LeBlanc
kevin.leblanc@aass.oru.se

Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University
S-70182 Örebro, Sweden

Abstract

Perceptual anchoring is the process of linking action to the appropriate objects in the environment via perception. The pivot of anchoring is the inclusion of micro-models of the world, or *anchors*, into a controller. In this paper, we propose to use anchors to focus the perceptual effort according to the current needs of the controller. We describe an active gaze control strategy able to maintain anchoring of several objects in a dynamic environment, and show how we have used it in a team of legged robots in the RoboCup'99 international robot soccer competition.

1 Introduction

In order to perform reliably in a dynamic environment, an autonomous robot needs to link, or *anchor*, its actions to the appropriate objects in the environment via perception. The environment being dynamic, the properties of these objects may change over time, and thus require constant perceptual attention. However, a robot typically has limited perceptual resources, both in terms of sensing and of processing power. The design of effective allocation strategies for these resources is therefore of paramount importance in dynamic and partially unpredictable environments.

The literature in the field of autonomous robotics has reported many solutions to the problem of controlling perceptual effort. One common approach is to pack both the perceptual and action processes into one module (or *behavior*), in order to focus the perceptual effort exclusively on those features in the environment which are *relevant* to the current task. Early examples of this approach can be found in Arkin's concept of affordances [1], in Malcom and Smithers' concept of encapsulation [10], and in Brooks' subsumption architecture [4]. Another common approach is to use information about the current task to perform *active* control of the sensors, in order to search for specific features. Active control may imply selecting a specific algorithm, or physically pointing a sensor in a specific direction. This approach has been pioneered by Bajcsy [2] and Ballard [3], and is an active research area in computer vision (e.g., [6, 9]).

In this paper, we present a technique for active perceptual anchoring in dynamic environments which combines the two approaches above. Following [14], we introduce *perception-based behaviors* as basic units of control which use *anchors* to connect behavior to the environment. An anchor is a data structure which stores information about a physical object, which is needed by the behavior, e.g., the position of a specific door to cross. We then extend the notion of an anchor, in order to use it to focus the perceptual effort. In particular, we include in each anchor a measure of how much the corresponding object is actually *needed* for the execution of the current task. This measure is used to selectively direct the gaze by deciding which anchor we should try to acquire at any given moment.

To illustrate our technique, we explain how we have used active perceptual anchoring to control a team of legged robots in the RoboCup[1] domain, as part of the Swedish national entry at the 1999 edition of the competition. The robots we used were dog-like robots produced by Sony [15], which were equipped with several sensors, as illustrated in Fig. 1. The task was to play soccer in an engineered field where all the objects were distinguishable by their color. The need for selective gaze control arose because the robot only has one sensor with which it can detect the objects: a camera and infra-red range finder combination mounted on the head. This sensor had to be oriented toward

[1] Annual international robot soccer competition [11].

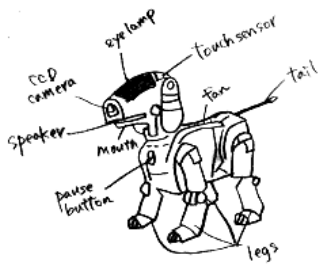

Fig. 1: Sony legged robot (© 1999 Sony Corporation).

an object for reliable detection; however, there were many objects of interest in the environment.

The rest of this paper is organized as follows. In the next two sections, we present perception-based behaviors and active perceptual anchoring in general terms. Sec. 4 describes how we implemented these techniques on the Sony robot platform. Sec. 5 reports three illustrative experiments, and Sec. 6 concludes.

2 Perception-based behavior

In autonomous robotics, one often distinguishes between so-called reactive, or sensor-driven behavior, and proactive, or goal-driven behavior. One way to draw this distinction is to say that these two types of behavior use different types of input data. Proactive behavior typically uses as input a combination of prior information and "proprioceptive" data, i.e., data from sensors that observe the state of the robot's body. Reactive behavior, by contrast, typically takes as input "exteroceptive" data, i.e., data from sensors that observe the environment.

This distinction can be seen in the example shown in Fig. 2. On the left, the robot is engaged in a proactive "go-to-net" behavior: the expected position of the net in some reference frame is given as prior information, and the robot relies on its odometry to estimate and reduce its distance from this position. On the right, the robot is using visual-feedback to reactively move in the direction in which the net is observed. Although the two behaviors may produce the same trajectory, there are important differences. The proactive

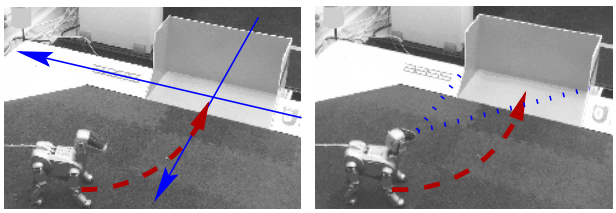

Fig. 2: Proactive (left) vs. reactive (right) behavior.

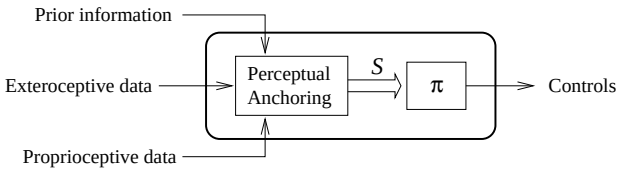

Fig. 3: A perception-based behavior.

behavior ignores any real-time data about the net: if this is moved, or if the odometry is wrong, the robot will reach a position which is not related to the actual goal. The reactive behavior does not suffer from this limitation, but it will be lost if the net is not perceived, e.g., owing to sensor noise or to occlusion.

As it appears, proactive and reactive strategies have merits and drawbacks which are somehow complementary; it is therefore reasonable to combine them into one type of behavior which integrates both goal-based and sensor-based information. To this aim, we define a *perception-based behavior* as a unit of control organized as shown in Fig. 3. The behavior takes exteroceptive data, proprioceptive data, and prior information, and integrates them into an internal state S. In general, S integrates information about the part of the physical world which is relevant to the behavior, and which is to be used by the control law π. The pivot of this integration is the notion of an *anchor*.

Anchors were originally introduced in [13] as "object descriptors," and have been further investigated in [14, 5]. An anchor is a data structure in S which collects information about a specific physical object, such as the net in the above example. This information is used by the control law π to generate a control value $\vec{u}$. Perceptual anchoring is responsible for initializing and updating the anchors in S as follows.[2]

Creation: build an anchor in S for each object in the environment used by the behavior, and initialize it from prior information; set the criteria against which future percepts should be matched (e.g., the color of the net).

Prediction: predict the properties of the anchor after some time has elapsed since it was last observed; this phase takes proprioceptive data, and possibly a dynamic model of the object into account.

Anchoring: match newly acquired percepts (exteroceptive data) against the anchor, and update it accordingly; a consistency check with the prior information may also be performed in this phase.

[2] This prediction-update cycle is typical of recursive estimation [8]. Anchoring, however, is peculiar in its use of prior, abstract information for initialization and verification [5].

It is easy to see that proactive and reactive behaviors are special cases of perception-based behaviors. The difference between these types of behavior does not reside in the control law, but in the different way in which the anchors relate to the environment. For example, the two go-to-net behaviors above can be produced by a simple control law that minimizes the distance to an anchor associated with the net. In the proactive behavior, this anchor is initialized from prior information and updated by odometry; in the reactive one, it is updated by visual data.

The most interesting case, however, is when we use all three types of information to update the anchor. In this case, the anchor acts as a "micro-model" of one object in the environment, which is relevant to the behavior. Through the anchor, the behavior operates in a closed loop with the environment whenever we have perceptual data, and relies on plausible assumptions when these are not available. This is particularly important for the robots used in this work, since looking at one object usually results in losing sight of others.

It is useful to know how much an anchor is supported by recent perceptual data, rather than being based on prior information or predictions. Therefore we associate each anchor with a number in the $[0, 1]$ interval, called "anchored," which measures the level of current perceptual support. This number is set to 1 each time the anchor is updated from perceptual data, and it exponentially decreases when the anchor is updated by prediction only. The amount of the decrease depends on the reliability of the prediction. This in turns depends on the type of object (e.g., fixed or movable) and on the displacement of the robot. For the legged robots considered in this work, the decrease was rather dramatic since odometry is extremely unreliable and since most objects are movable.

3 Active perceptual anchoring

Anchors provide the key to making the perceptual processes more relevant to the needs of the behavior. Since the anchors in S contain all the variables used by the π control law, the perceptual processes only need to extract the values of these variables. Moreover, since anchors are models of the needed objects, the perceptual routines can use the information in the anchors to narrow the search space; for example, in order to anchor the ball, we only need to check the camera image for orange blobs close to the ground.

Focusing the perceptual processes on the anchors in S greatly simplifies perception. Execution of complex tasks, however, may require information about

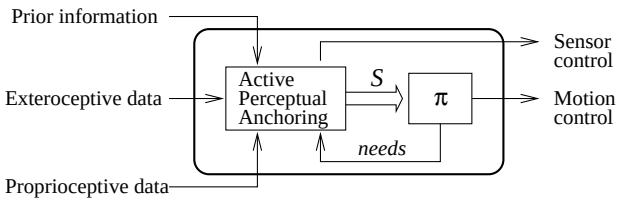

Fig. 4: Feedback of perceptual needs.

many objects, and therefore S may contain many anchors. The key observation here is that the control law typically needs to access different anchors at different stages of execution. For example, in the RoboCup domain the robot only needs the position of the ball and of the opponent net when it must kick the ball into the net, and it only needs the position of the landmarks when it must return to its home position in the field.

To exploit this fact, we use the organization summarized in Fig.4. Anchors are extended to include a value that measures, on a $[0, 1]$ scale, how much that specific anchor is currently *needed* by the control law π. This value is fed by π back into S. The perceptual processes use the "needed" values to decide for which anchors they should try to extract information.

As marked in the figure, the anchoring module may also generate sensor controls, e.g., a pan-tilt control to change the fixation point of a camera. In this respect, the "needed" measure can be effectively used to address an important problem in active gaze control: how to decide where to move the fixation point. For each anchor a in S, we measure the importance of a as follows:

$$\text{important}(a) = \text{needed}(a) \cdot (1 - \text{anchored}(a)). \quad (1)$$

The anchor f with the highest importance is selected as the current focus of attention; the fixation point is then determined by the expected position of the object which is associated to f. It is easy to see that if there are n anchors which are needed, this selection mechanism will select and anchor all of them in a round-robin fashion.

Equation (1) has a logical reading. The "anchored" and "needed" measures can be interpreted as fuzzy predicates. Then, (1) gives the truth value, according to the rules and connectives of fuzzy logic [12], of the fuzzy predicate "important" defined as

$$\text{important}(a) \equiv \text{needed}(a) \wedge \neg\text{anchored}(a).$$

In other words, re-anchoring a becomes important if a is needed, and if it has not been anchored recently.

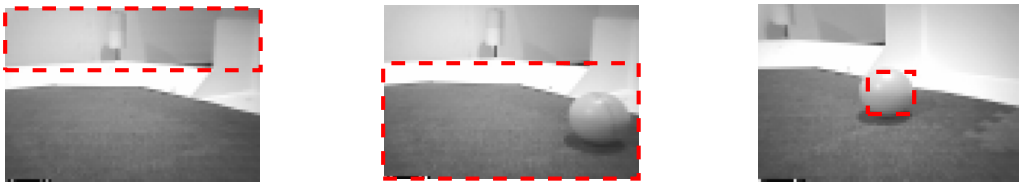

Fig. 5: A ball blob in the astray zone (left), tracking zone (center), and anchoring zone (right).

4 Anchoring in the RoboCup domain

In order to illustrate the use of active perceptual anchoring, we briefly discuss how we implemented it on the Sony robots for the RoboCup domain.

At an abstract level, the control architecture that we have used in each robot is similar to the one sketched in Fig. 4. The S state contains an anchor for each object of interest in the RoboCup domain. (The number and type of objects are fixed in this domain.) Each anchor contains the type of object, its (ρ, θ) position in robot's polar coordinates, and the value of the "anchored" and "needed" fuzzy predicates. The updating of the position and "anchored" fields is done as discussed above. The "needed" field is set by the π control law.

The π control law is implemented by a set of fuzzy behaviors organized in a hierarchical way.[3] These behaviors include fuzzy rules of the form

```
IF condition NEED(object);
```

whose effect is to assert the need for an *object* at a degree that depends on the truth value of *condition*. For example, consider the rules

```
ALWAYS          NEED(Ball);
IF(NearBall)    NEED(Net1);
```

where `ALWAYS` is a condition which is always true. In a situation where the ball is at 400 mm from the robot, the truth value of "NearBall" is 0.7, and these rules assert a value of needed of 1.0 for the anchor Ball and of 0.7 for Net1. Behaviors are dynamically activated and deactivated according to the current task and situation, and several behaviors can be active at the same time. The needed values stored in the S state are those asserted by the active behaviors, combined by the max operator. This guarantees that perceptual anchoring only depends on the currently active behaviors, hence on the current task.

[3]The details of fuzzy hierarchical behaviors are not relevant here, and can be found in [14, 7].

Perceptual anchoring is implemented as follows. Since most objects in our domain are uniquely identified by their color, we use the color information from the camera as our main cue. We use a model-based approach to combine regions of a given color into "blobs" of some type; for instance, an orange region of a reasonable size is taken to be a ball blob, and a green region over a pink one forms a landmark blob. We then check blobs against the following three object-dependent criteria, illustrated in Fig. 5.

First, we reject any blob which is "astray" for its type: in the leftmost picture in Fig. 5, the pink band of a landmark has been incorrectly classified as a ball (orange) blob, but rejected because the ball is assumed to lay on the ground. Second, we select blobs which are in the "anchoring zone" (AZ) for their type. This is the area in the image where we can reliably measure the object's position. For instance, a landmark blob is in its AZ if it is fully inside the image (i.e., it does not touch the image border) since we use the blob's size to estimate distance. On the other hand, the AZ for the ball is a small fovea at the center of the image, since we use the IR sensor, which has a field of view of about $\pm 2.5°$, to estimate distance. Finally, we consider the remaining blobs to be in "tracking zone" (TZ); these should be brought into the anchoring zone using gaze control.

Gaze control is described by the finite state automaton sketched in Fig. 6, where rectangles denote head motion states whose transitions are activated by events, and ovals denote purely computational states whose transitions depend on the result of the computation. The meaning of each state is as follows.

Select: choose an anchor f on which to focus using (1); if the "anchored" level of f is greater than a given threshold, then set the fixation point to this position and exit via the **G**ood **E**stimate transition; otherwise exit via the **not-GE** transition.

Scan: perform a visual scan to explore the part of the space where f could be located, e.g., the field ground for the ball; exit when one of the following events occurs:

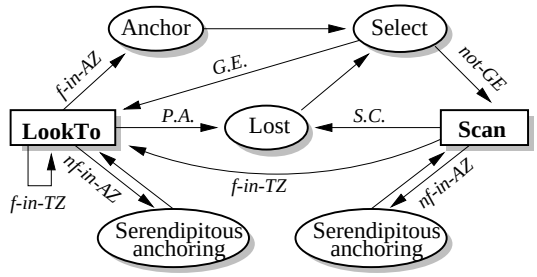

Fig. 6: State transitions for perceptual anchoring.

- a blob that matches f appears in the TZ; set the fixation point to that blob's position and exit via the ***f*-in-TZ** transition;
- a blob that matches an anchor $a \neq f$ appears in the AZ for a; exit via the ***nf*-in-AZ** transition;
- the physical scan is completed; exit via the **S**can **C**omplete transition.

LookTo: move the camera to the current fixation point; when:

- a blob that matches f appears in the AZ, exit via the ***f*-in-AZ** transition;
- a blob that matches f appears in the TZ, set the new fixation point to that blob's position and stay in this state;
- a blob that matches an anchor $a \neq f$ appears in the AZ for a, exit via the ***nf*-in-AZ** transition;
- the fixation point has been achieved, and no blob that matches f is in the image, exit via the **P**osition **A**chieved transition.

Anchor: measure the position of the object and update the f anchor; select a new focus.

Serendipitous anchor: an object other than f is in the AZ: measure its position and update the corresponding anchor; continue to search for f.

Lost: either "Scan" or "LookTo" have been completed without the current f being found; mark f as lost and go select a new focus.[4]

There is a subtlety in the "a blob that matches" conditions above. In general, how to associate regions to objects is a complex issue, which may require the use of extra-perceptual information. For example, the three opponent robots all have the same perceptual signature, and can only be distinguished by using information about their past positions. In our case, we simply associate a blob to an anchor based on its type, i.e., its color. (See [5] for a more elaborated solution to the association problem.)

[4] A lost object will not be selected as focus as long as there are other objects which are important according to (1).

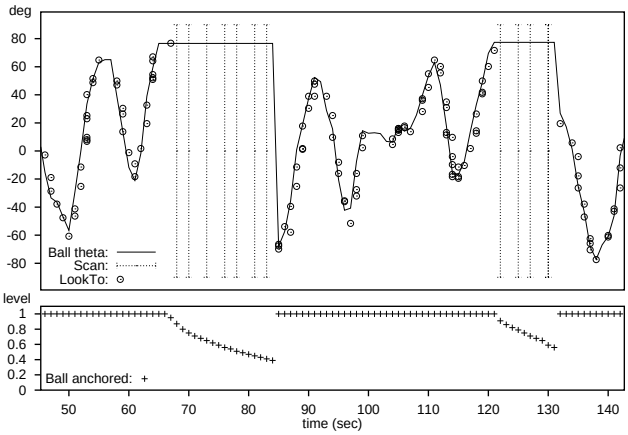

Fig. 7: Tracking a moving ball.

5 Experiments

We present three simple experiments run on a Sony legged robot in the RoboCup domain. In the first experiment, the robot simply had to stand still and track the ball, which was moved by hand. This was achieved by the following rules:

```
ALWAYS GO(STAY);
ALWAYS TURN(AHEAD);
ALWAYS NEED(Ball);
```

The first two rules set the linear and rotational velocity to zero; the third one sets the "needed" value for the ball to 1. Fig 7 shows the result of ball tracking. The upper part plots the temporal evolution of the estimate of θ in the anchor for the ball. The circles mark the "LookTo" commands sent to the head whenever a new fixation point was determined; the dotted vertical lines mark "Scan" commands. The lower part of the figure shows the corresponding evolution of the "anchored" value. From the beginning till time 66 the ball was moved between $\pm 60°$, at a fixed distance in front of the robot. The robot was able to track and anchor the ball at all times. At time 66 the ball was removed from the field; the robot then started a visual scan, while maintaining the previous θ value and decreasing the degree of anchoring. At time 86 the ball was laid down again, and the robot could rapidly

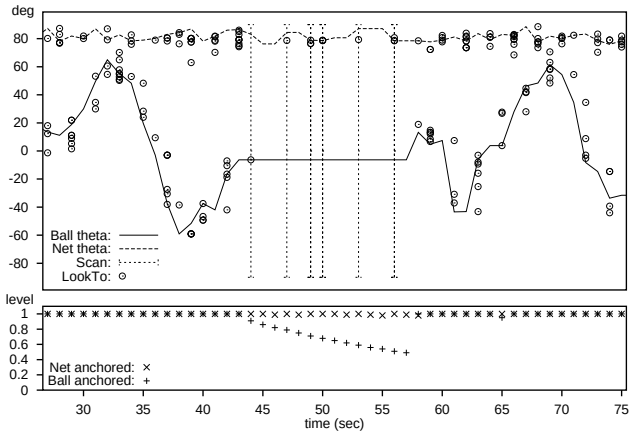

Fig. 8: Tracking a moving ball *and* the net.

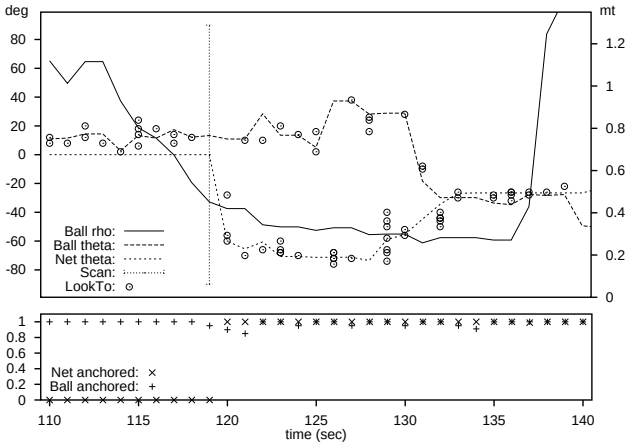

Fig. 9: Switching between tracking only the ball, and tracking both ball and net.

identify and re-anchor it. The rest of the plot shows another sequence of moving the ball, stealing it, and putting it down again. The robot consistently kept the ball anchored when it was in view, and searched for it when it was out of view.

In the second experiment the robot was placed in the middle of the field and heading at 80° from a net (Net1), and it had to track both the moving ball and the net. This was achieved by adding the rule:

```
ALWAYS NEED(Net1);
```

to the previous ones. The plot in Fig. 8 shows the temporal evolution of the estimate of θ for the ball and net anchors, and the corresponding values for the "anchored" predicates. The ball was first moved between $\pm 60°$ in front of the robot, then taken out of the field at time 43. At time 57, it was put back down and moved from left to right in front of the robot. The robot alternatively looked at the ball and at the net using the **Select** function discussed above, and kept both of them constantly anchored when the ball was in view. When the ball was removed from the field, the robot alternatively made a scan for the ball, and re-anchored the net.

The third and final experiment illustrates how active perceptual anchoring is guided by the current perceptual needs expressed by the behaviors during execution. The robot first has to approach the ball, then align with the net and finally kick the ball. This behavior is implemented by the following rules:

```
IF (NOT(NearBall))                USE (Reach,Ball);
IF (AND(NearBall,NOT(Aligned))) USE (Align,Net1);
IF (AND(NearBall,Aligned))       USE (Kick,Net1);
ALWAYS                            NEED (Ball);
IF (NearBall)                     NEED (Net1);
```

where the USE keyword denotes a recursive call to a sub-behavior. The plot in Fig. 9 shows the evolution of (θ, ρ) for the ball anchor, and of θ for the net anchor, together with the two values for "anchored." Until time 118 the robot was approaching the ball (as testified by the decreasing ρ) while tracking it. Since the "NearBall" predicate was false, the "needed" value for the net was kept at 0, and the net was never selected as the focus of attention; as a consequence, the camera was never pointed at the net, which remained fully unanchored, with θ at its default value of 0. At time 118, the truth value of "NearBall," and hence the net's "needed" value, was high enough for the net to become the focus of attention. The robot then started a visual scan to search for the net, momentarily losing sight of the ball. The net was soon seen and anchored, and the robot then pointed its camera to the expected position of the ball and re-anchored it. The robot then got close to the ball, and at time 127 it executed the alignment maneuver while using the camera to alternatively anchor the ball and the net. The alignment maneuver brought the θ of both the ball and the net close to zero. Finally, at time 137, the robot kicked the ball, which headed away from the robot and toward the net.

The above behaviors are extremely simple, but they illustrate the main ingredients of active perceptual anchoring. A set of significantly more complex behaviors were implemented for the RoboCup competition using the mechanisms described in this paper. During the games, the robots successfully tracked all and only the objects which were relevant to the task being performed.

6 Conclusions

Active perceptual anchoring has proven to be an effective method of maintaining a consistent local picture of the environment. The central ingredient is the notion of an "anchor", a local model of an object in the environment which is relevant to behavior. Anchors perform two main functions: (i) they integrate prior information, past perception, and current perception in the service of the control law; and (ii) they direct the perceptual processes to be relevant to the current task. The first point was already investigated in [14]. The second point was the main focus of this paper. The pivot of this point is the ability to put into the anchors information about the current perceptual needs of the controller. This information is used to determine the fixation point for active camera control. We have shown how this technique has been successfully used in the robot soccer domain. In the near future, we also expect to apply this technique on service robots in indoor and outdoor domains.

The use of anchors as a means to integrate perception and action has not received much attention in the robotics literature. One notable exception is the system described in [16], which uses "markers" to model objects in the environment which are important to the current task. Markers are very close to the object descriptors used in [14], and to the anchors used in the work described here. The main difference is that a marker embodies an indexical, rather than a definite reference: the same marker can be associated to different objects depending on the task, while an anchor is uniquely associated to one physical object. (This one-to-one association is important for using anchors to connect symbolic names to perceived objects [5].) A second difference is that in the system in [16] the information about the current perceptual needs is provided by the higher decision layers, while in our system this information is provided by the controller *inside* the perception/action layer.

Acknowledgments

The work reported in this paper was partly supported by the Swedish KK Foundation. This work greatly benefit from discussions with Silvia Coradeschi, Dimiter Driankov, and the colleagues of Team Sweden '99 listed at http://aass.oru.se/Living/RoboCup/.

References

[1] R.C. Arkin. The impact of cybernetics on the design of a mobile robot system: a case study. *IEEE T. on Sys., Man, and Cybernetics*, 20(6):1245–1257, 1990.

[2] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[3] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–87, 1991.

[4] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[5] S. Coradeschi and A. Saffiotti. Anchoring symbols to vision data by fuzzy logic. In A. Hunter and S. Parsons, eds, *Proc. of the ECSQARU'99 Conf.* LNCS 1638, 104–115. Springer-Verlag, DE, 1999. Online at http://aass.oru.se/Research/Robots/asd.html.

[6] J.L. Crowley and H.I. Christensen, editors. *Vision as Process*. Springer-Verlag, 1995.

[7] D. Driankov and A. Saffiotti, eds. *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*. Springer-Verlag, DE, forthcoming.

[8] A. Gelb, J.F. Kasper, R.A. Nash, C.F. Price, and A.A. Sutherland. *Applied Optimal Estimation*. MIT Press, 1974.

[9] R. Howarth. Interpreting a dynamic and uncertain world: task-based control. *Artificial Intelligence*, 100:5–85, 1998.

[10] C. Malcom and T. Smithers. Symbol grounding via a hybrid architecture in an autonomous assembly system. In P. Maes, editor, *Designing autonomous agents*, 123–144. MIT Press, 1990.

[11] The RoboCup Federation. Robocup official site. http://www.robocup.org/.

[12] E. Ruspini, P. Bonissone and W. Pedrycz, eds. *IEEE Handbook of Fuzzy Computation*. Oxford Univ. and IOP Press, 1998.

[13] A. Saffiotti. Pick-up what? In C. Bäckström and E. Sandewall, editors, *Current Trends in AI Planning*, pages 166–177. IOS Press, 1994.

[14] A. Saffiotti, K. Konolige, and E.H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.

[15] Sony Corporation. Entertainement Robot AIBO. http://www.world.sony.com/robot/.

[16] G. Wasson, D. Kortenkamp, and E. Huber. Integrating active perception with an autonomous robot architecture. Robotics and Automation J. (to appear).