

# Multi-Humanoid World Modeling in Standard Platform Robot Soccer

Brian Coltin, Somchaya Liemhetcharat, Çetin Meriçli, Junyun Tay, and Manuela Veloso

**Abstract**—In the RoboCup Standard Platform League (SPL), the robot platform is the same humanoid NAO robot for all the competing teams. The NAO humanoids are fully autonomous with two onboard directional cameras, computation, multi-joint body, and wireless communication among them. One of the main opportunities of having a team of robots is to have robots share information and coordinate. We address the problem of each humanoid building a model of the world in real-time, given a combination of its own limited sensing, known models of actuation, and the communicated information from its teammates. Such multi-humanoid world modeling is challenging due to the biped motion, the limited perception, and the tight coupling between behaviors, sensing, localization, and communication. We describe the real-world opportunities, constraints and limitations imposed by the NAO humanoid robots. We contribute a modeling approach that differentiates among the motion model of different objects, in terms of their dynamics, namely the static landmarks (e.g., goal posts, lines, corners), the passive moving ball, and the controlled moving robots, both teammates and adversaries. We present experimental results with the NAO humanoid robots to illustrate the impact of our multi-humanoid world modeling approach. The challenges and approaches we present are relevant to the general problem of assessing and sharing information among multiple humanoid robots acting in a world with multiple types of objects.

## I. INTRODUCTION

For several years, we have witnessed and experienced the robot soccer challenge towards having a team of robots autonomously perform a “scoring” task (pushing a ball into a goal location) on a predefined space in the presence of an opponent robot team. We are focused on the teams of robots with onboard perception, control, actuation, and communication capabilities. While many complete robot soccer teams have been devised with varied levels of success, one of the main challenges is still the “world modeling” problem for such robot teams, where robots have limited, directional perception. Each robot needs to build a model of the state of the world, e.g., the positioning of all the objects in the world, in order to be able to make decisions

The first author was partially sponsored by the Office of Naval Research under grant number N00014-09-1-1031. The second author was partially supported by Lockheed Martin, Inc. under subcontract 8100001629/1041062, and the Agency for Science, Technology, and Research (A\*STAR), Singapore. The third author was supported by The Scientific and Technological Research Council of Turkey Programme 2214. The views and conclusions contained in this document are those of the authors only.

B. Coltin and S. Liemhetcharat are with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, USA [bcoltin@cs.cmu.edu](mailto:bcoltin@cs.cmu.edu), [som@ri.cmu.edu](mailto:som@ri.cmu.edu)

Ç. Meriçli, J. Tay, and M. Veloso are with Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, USA [cetin, junyunt}@cmu.edu](mailto:{cetin, junyunt}@cmu.edu), [veloso@cs.cmu.edu](mailto:veloso@cs.cmu.edu)

towards achieving its goals. World modeling is the result of the robot’s own perception, the robot’s models of the objects, and the communicated information from its teammates. This world modeling problem is complicated by the fact we consider that the robot relies only on visual perception of the objects in the environment, which is typically noisy and inaccurate. In addition, the robots have a limited field-of-view which allows the robot to detect only a small subset of objects at a particular time. Interestingly, we note that the primary goal of the robots is not to track the multiple objects in the world, but to accomplish some other task, e.g., scoring a goal. However, effectively performing the task directly depends on an accurate model of world objects. We view this world modeling problem of a group of robots with limited perception and communication capabilities relevant to a very general future environment when robots will naturally need to perform tasks involving identifying and manipulating objects in a world with other moving robots and towards achieving specific goals.

World modeling clearly includes object tracking and there is extensive previous related work. Multi-model motion trackers incorporate the robot’s actions as well as its teammates’ actions (e.g., [1]), to allow a robot to track an object even if its view is obscured, and if teammates take actions on the object. Rao-Blackwellised particle filters have been extensively used to effectively track a ball in the robot soccer domain (e.g., [2]). In the presence of multiple robots communicating among themselves, a variety of approaches have been developed to fuse the information from multiple sources, using subjective maps [3], in high-latency scenarios [4], with heterogeneous robots [5], and using a prioritizing function [6].

In this paper, we drive our presentation using the RoboCup Standard Platform League [7] [8] with the Nao humanoid robots [9] in detail, to carefully present the general world modeling problem. We identify different classes of objects in the world in terms of their motion models. We discuss and contribute the world model updating approaches for each of the identified object classes, and demonstrate the effectiveness of these approaches experimentally.

## II. PROBLEM STATEMENT

We are interested in modeling objects in the world, such that the humanoid robot has an accurate estimate of the location of the objects, even if the objects are not currently visible to the humanoid. The world model maintains hypotheses of the positions of the objects in the world, given the sensor readings of the humanoid and the models of the objects.

**Definition 1:** A **World Model** is a tuple  $\{O, X, S, M, H, U\}$ , where:

- $O$  is the set of labels of objects that are modeled
- $X$  is the set of possible object states, i.e.,  $x \in X$  is a tuple representing the state of an object, such as its position in egocentric coordinates, velocity, and confidence
- $S$  is the set of possible sensor readings, i.e.,  $s \in S$  is a tuple representing all currently sensed objects, and the internal state of the robot
- $M$  is the set of models of the objects, where  $m_o \in M$  is the model of object  $o$
- $H : M \times O \rightarrow X$  is a hypothesis function that returns the state of an object given its model
- $U : M \times O \times S \rightarrow M$  is the model update function, i.e.,  $m'_o = U(m_o, o, s)$

In multi-robot scenarios, such as RoboCup, communication between teammates, e.g., sharing of ball information, can be viewed as a sensor reading of the receiving robot. Also, at any point in time, there can be some objects that are not sensed by the robot. As such, the update function  $U$  must be capable of updating the models of objects that are not currently sensed.

#### A. Objects in the World

There are multiple types of objects in the world, which have been organized into static, passive, actively-controlled, and foreign-controlled [10] (see Fig. 1). Static objects, as their name implies, are stationary objects. Passive objects are objects that do not move on their own, but can be actuated by other objects, e.g., a ball in the RoboCup domain. Models of passive objects include a motion model for tracking their velocity and trajectory, as well as the effects of other robots' actions on the object, for example, when a teammate kicks the ball. Actively-controlled and foreign-controlled objects are those that move on their own, and are differentiated by whether we have full knowledge of the actions taken by the objects. In the robot soccer domain, the robot's teammates are actively-controlled and the opponents are foreign-controlled.

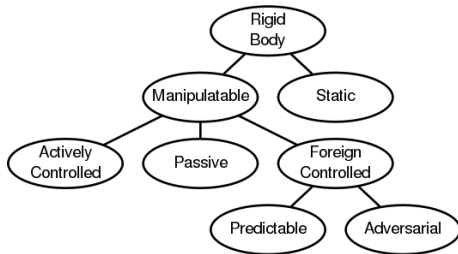


Fig. 1. Types of objects, as introduced by [10].

**Definition 2:** Let  $O$  be the set of all objects in the world model.  $O_s, O_p, O_a, O_f$  are static, passive, actively-controlled, and foreign-controlled objects, respectively, where  $O_s, O_p, O_a, O_f \subseteq O$ .

In RoboCup domain,  $O_s$  is comprised of the goals (yellow and blue) and other fixed landmarks on the field, such as

field lines and corners.  $O_p$  contains the ball, and  $O_a$  and  $O_f$  consist of teammates and opponent robots respectively.

For each object in the world model, we maintain a model of its position in egocentric coordinates. The model of the object is updated according to the category of that object. For example, static objects are updated only based on visual cues (e.g., a goal post is detected in the camera image), and by the robot's movement, as they do not move. The models of such objects do not include velocity, since static objects do not move in the environment. In contrast, passive objects have an associated velocity model, which is updated based on both visual cues and actions taken by the robot and its teammates, e.g., kicking the ball.

#### B. Challenges in Modeling the World

Creating an accurate world model for the RoboCup domain is a challenging problem. Firstly, the Nao humanoid robot used in the RoboCup Standard Platform League has limited sensing capabilities (see Fig. 2). The internal sensors of the Nao, i.e., accelerometers and gyroscopes, are useful to determine the robot's state, but are unable to sense external objects in the world. Ultrasonic sensors are used to detect obstacles in front of the robot, but the obstacle information is not incorporated into the world model. Perception of external objects is performed using computer vision on the images from the on-board cameras located in the Nao's head. Due to the narrow field-of-view of the cameras, the robots are only able to sense a subset of the objects in the world at any one time, and must actively choose which objects to perceive. Also, the field is  $6m \times 4m$ , while the robot is only 30cm wide, and so the robot is typically unable to perceive some objects in the world without moving around.

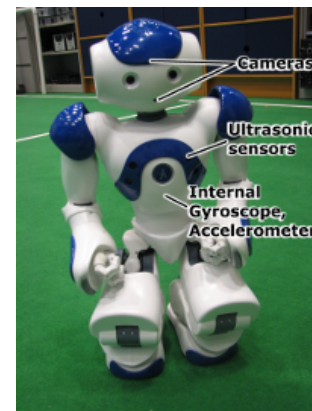


Fig. 2. Aldebaran Nao humanoid robot used in the Standard Platform League of RoboCup, and its on-board sensors.

Secondly, the environment is highly dynamic and adversarial. The position of the ball varies over time, as the robots on the field interact with it. Furthermore, the robots are constantly moving across the field, limiting line-of-sight to the ball and other landmarks. The actions of teammates are shared across the team, therefore modeling teammates is relatively easier than modeling opponents, whose actions are unknown and are difficult to track. The goal of the robot

team is to kick the ball into the opponent’s goal, and as such, modeling objects is not the primary objective of the robots. The robots typically maximize the amount of time perceiving the ball (as it is the most important object in the domain), but have to maintain an accurate model of other objects in order to carry out the high-level goal of scoring.



Fig. 3. The field setup of RoboCup SPL 2009

Thirdly, landmarks in the environment are ambiguous. Although goals are distinguished by their colors, blue and yellow, it is difficult to differentiate between the left and right goal posts of the same color, especially when the robot is standing close to the goal. In addition, the soccer field is marked by non-unique lines and corners (see Fig. 3), which are impossible to differentiate based on a single camera image, e.g., a straight line looks identical when the robot stands on either side of it. Fig. 4 shows a yellow goal, an ambiguous (left/right) yellow goal post, and an ambiguous corner.

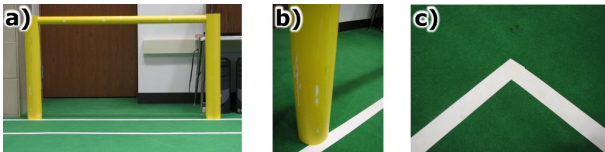


Fig. 4. a) A yellow goal. b) An ambiguous yellow goal post. c) An ambiguous corner.

### III. ROLE OF THE WORLD MODEL

The world model, which contains the positions of objects in the environment, is only a part of a larger system. To fully understand the design and function of the world model in the RoboCup domain, a brief explanation of the other components and their interactions with the world model is necessary.

On a low level, the vision component processes images from the camera and returns the positions of visible objects. Since the objects on the field are color-coded (the field is green, goals are yellow and blue, the ball is orange), vision uses color segmentation and blob formation to identify objects. All of the robot’s motions are controlled by a motion component. The motion component receives motion commands such as walk forward, turn, or kick, and executes them by manipulating the joint angles. A walk algorithm based on the Zero Moment Point (ZMP) approach is employed [11]. The motion component outputs odometry information (i.e., the displacement of the robot).

The world model maintains the positions of objects in egocentric coordinates. However, to make sense of the world models of their teammates or execute cooperative behaviors, they must communicate using global coordinates. The self-localization algorithm takes the observations of goals, lines and corners from vision along with odometry information from motion as input, and estimates the robot’s global position using a particle filter [12]. The localization problem is especially challenging for humanoid robots due to noisy odometry and a limited field of view. Ambiguous landmarks also make localization difficult — for lines, we use an algorithm presented in [13] to update the particle weights.

The referees communicate their rulings through a wireless network with the robots. The information received from referees is processed to determine the current state of the game, such as when a goal is scored, when kickoff occurs, or when a penalty is called.

At the highest level are the Nao’s behaviors, which decide the robot’s actions. These include skills, tactics, and plays, which model low-level abilities (such as kicking the ball), the behavior of a single robot, and the behaviors of multiple robots, respectively [14]. Behaviors issue motion commands to the motion component. They retrieve information about the environment from the world model, and the robot’s own global position from localization.

In this architecture, the world model fills the essential role of determining the positions of objects on the field, merging observations from vision, messages from teammates, and odometry information from the motion component. The world model’s position estimates are then used by the behaviors to decide the robot’s actions.

### IV. MODELING THE WORLD

The algorithms for modeling objects vary widely depending on the object category, yet several fundamental algorithms are utilized by all object types.

Firstly, all objects are updated based on the robot’s odometry. Odometry information is passed to the update function  $U(m, o, s)$  as values  $\Delta x$ ,  $\Delta y$ , and  $\Delta\theta$  in  $s$ . For all object types, the function  $U$  first updates the estimated position with an odometry function, i.e.,  $(x', y') = \text{odom}(x, y, \Delta x, \Delta y, \Delta\theta)$ , where  $(x, y)$  and  $(x', y')$  are the original and transformed coordinates of the object respectively.

Secondly, each observation of an object  $o$  in  $s$  from some sensor includes the position and confidence of the observation. These observations are integrated into the position estimate  $\vec{x}$  of  $m_o$ , via a filter, e.g., a Kalman filter. The filter reduces the model’s sensitivity to noise, and weights the observations according to their confidence.

Finally, the objects will not always be sensed. The world model must track the objects’ positions even when they are not currently sensed. It measures the confidence  $c \in [0, 1]$  of its estimates so that the robot doesn’t act on outdated or incorrect information. When the object is sensed (either through a physical sensor or teammate communication),  $c$  is set to the confidence given in  $s$ . Otherwise, the confidence

decays according to a function  $N(c, s)$  which is specific to the object being modeled.

The updated confidence  $c$  in an object’s position is thresholded into three states:

- *Valid*: The robot currently senses the object or sensed it recently. The robot’s behaviors should assume the position is correct.
- *Suspicious*: It has been some time since the object was sensed. The robot’s behaviors should look at the object before it becomes invalid.
- *Invalid*: The object’s position is unknown.

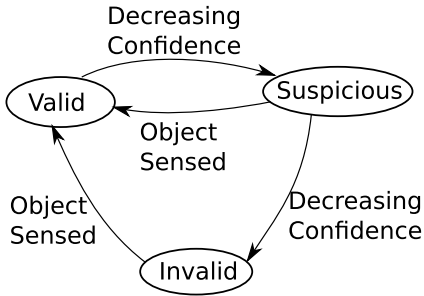


Fig. 5. Transitions between object confidence states.

The threshold levels  $l_{\text{suspicious}}$  and  $l_{\text{invalid}}$ , specific to each object type, are determined through experimentation. See Fig. 5 for the transitions between confidence levels. The suspicious state is an active feedback mechanism, which serves as a request from the world model to look at the object. The behaviors set a boolean flag in  $s$  when the robot is currently looking at the object’s estimated position.  $N$  will typically accelerate the decay of the confidence when this flag is set. This active feedback mechanism ensures that false positives from sensors and objects which have moved are invalidated more quickly so that the robot does not act on incorrect information.

Using these general algorithms applicable to all object types, we will discuss how each category of object is modeled, particularly in the RoboCup domain.

#### A. Static Landmarks

The RoboCup Standard Platform League (SPL) uses a field setup closely resembling a real soccer field. The only unique landmarks are two colored goals and the center circle (see Fig. 3). The landmarks on the field (both unique and non-unique) are categorized as *static* objects ( $O_s$ ) because their positions on the field do not change.

1) *Goal Posts*: Due to the large size of the goal relative to the field of view of the robot, the two goal posts are treated separately. This raises the problem of uniquely identifying the goal posts. One straightforward approach is to use spatial relations between the left and right posts and the top goal bar. However, this is especially difficult, if not impossible, in cases where the robot looks at a post where the top goal bar is not seen (see Fig. 4b for an example). Uncertainty associated with the vision component, such as changes in the lighting or misclassifications during the color segmentation phase,

might lead the goal post perception algorithm to incorrectly identify a left or right goal post.

2) *Field Lines*: The SPL soccer field contains a set of markings for visually emphasizing the special regions and boundaries of the field. These are non-unique, and we do not include them in the world model. However, they are used in the robot’s self-localization process to compute its own position.

3) *Updating the Confidence of Static Objects*: In addition to partially visible goal posts, all of the field markings except the center circle are non-unique landmarks. A landmark should be identified uniquely before updating its confidence. Different methods can be used to disambiguate non-unique landmarks. Taking advantage of known global landmark positions, constraints imposed by the relative positions of landmarks with respect to each other can be used to associate perceived landmarks with existing objects in the world model. Another way of associating non-unique landmarks with known ones is using the proximity of its global position to the real positions of known landmarks.

The major distinction separating static landmarks from other objects is that they are subjected to a decay function based on the motion of the robot instead of time. The vision component computes a confidence value  $c \in [0, 1]$  for each visible static landmark. That value is used by the model as long as the object is currently sensed by the robot. The confidence value remains unchanged if the object is no longer sensed but the robot is stationary. If the robot is moving,  $U$  sets  $c_{t+1} \leftarrow N(c_t)$  where  $N$  is a decay function dependent on the rate of the robot’s motion.

#### B. Passive Objects

In RoboCup, the ball is the most important object and therefore the world model needs an accurate position estimate for the ball at all times. The ball requires a more complex model than static landmarks because it moves across the field based on the actions of robots. It belongs to a more general class of *passive* objects ( $O_p$ ), i.e., objects which do not move of their own accord, but will move when acted on by external forces. A passive object can be free, or controlled by a robot— each state requires a different model. We will specifically study the problem of modelling the ball, but the techniques used are applicable to general passive objects.

Recall that  $m_{\text{ball}} \in M$  is the model of  $o_{\text{ball}} \in O_p$ . This model is updated based on the sensor readings  $s$  by an update function  $U(m_{\text{ball}}, o_{\text{ball}}, s)$ . The hypothesis function  $H(m_{\text{ball}}, o_{\text{ball}})$  returns an estimate of the ball state,  $x_{\text{ball}}$ .

1) *Tracking the Ball*: In every update of the ball’s model, the position and confidence of the ball are updated according to the odometry function  $odom$  and a filter  $f$ .

Since the ball is a passive object, unlike the goal posts, we must model its motion. The ball has a velocity  $\vec{v}$  (an element of  $m_{\text{ball}}$ ) which decays over time at a rate  $\alpha$ , such that  $\vec{x}_{t+1} \leftarrow \vec{x}_t + \vec{v}_t \Delta t$  and  $\vec{v}_{t+1} \leftarrow \max(0, \vec{v}_t - \alpha \Delta t)$ . The decay rate  $\alpha$  depends on the properties of the surface and the ball. Other motion models may be used in the more general case of other types of passive objects.

$\vec{v} = 0$  unless a robot acts on it— the question is, then, how can the actions of the other robots be modeled to predict when the ball will be kicked. In [1], a probabilistic tracking algorithm is introduced based on the actions of the robots. The ball transitions between free and kicked states based on the actions of the robot and its teammates, which are communicated wirelessly (and listed in  $s$ ). When the ball transitions to a kicked state, the update function  $U$  sets  $\vec{v} \leftarrow \vec{d}v_i$ , where  $\vec{d}$  is a unit vector representing the direction the robot is facing (communicated by the teammate) and  $v_i$  depends on the strength of the robot’s kick.

Modeling the actions of the opponents is more challenging. In this case,  $U$  resorts to estimating a velocity based on the changes in the ball’s position over time. This velocity estimation also serves to detect the unintentional actions on the ball which are common in the Standard Platform League, such as falling down on the ball or bumping into it.

2) *Updating the Ball Confidence:* All objects are modeled with a confidence value  $c$ , which is thresholded to a valid, suspicious, or invalid state. How this confidence is updated varies with each type of object. In the case of the ball, when it is visible,  $c$  is simply the confidence computed by the vision component. If vision does not detect a ball,  $U$  updates the confidence according to a decay function  $N$ .  $N$  is dependent on the time elapsed and the movement of the robot. If  $c$  is thresholded as suspicious, and the robot is looking at the estimated ball position,  $N$  causes the confidence to decay more rapidly. This increased decay rate is an active feedback mechanism which ensures that false sightings and balls which have moved are invalidated more quickly so that the robot can begin to search for the ball.

3) *Multiple Hypotheses:* We have described an effective model of the ball for a single robot, if the hypothesis function simply returns the estimated ball position and its confidence level. However, it does not incorporate information from the robot’s teammates. To do this, we include a list of hypotheses  $h$  in  $m_{\text{ball}}$  containing the ball position estimates and confidence values from the robot and its teammates.

Each robot estimates the ball position in its own local coordinate frame. To understand each other’s position estimates, the robots must convert the positions to global coordinates, using self-localization, to send to their teammates. The robots receiving these global position estimates convert them to their own local coordinate frame with self-localization. This process introduces the error present in the self-localization of both robots into the estimate of the ball’s position, so we factor the localization error into the confidence level for teammate ball estimates  $h$  in  $m_{\text{ball}}$ . This causes the robot to favor its own estimates over those of its teammates.

The hypothesis function  $H$  returns the position estimate with the highest confidence, and  $U$  decays the confidence of the hypothesis with the highest confidence. Fig. 6 shows how the ball confidence returned by  $H$  varies over time.

### C. Controlled Objects

Along with static and passive objects, the third type of object is controlled objects, which have the ability to

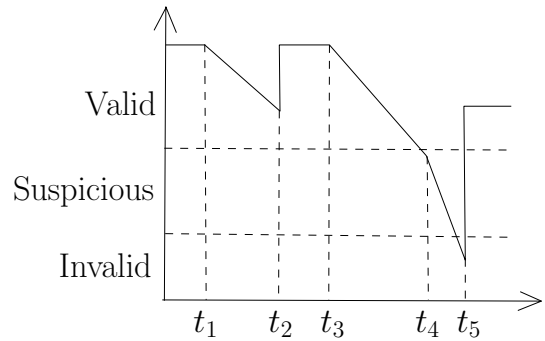


Fig. 6. An example scenario showing the ball confidence returned by  $H$ . Initially the ball is visible, but at  $t_1$  it leaves the field of view. The ball is seen again at  $t_2$ , but lost once more at  $t_3$ . At  $t_4$ , the ball becomes suspicious, and the behaviors look at where the ball is supposed to be. It is not present, so the confidence decays more rapidly. At  $t_5$ , after the ball becomes invalid, a position estimate is received from a teammate.

move themselves without external forces. The world model includes two types of controlled objects:  $O_a$  (actively-controlled) and  $O_f$  (foreign-controlled). We have full knowledge of the actions of actively-controlled objects, while foreign-controlled objects are controlled by others, i.e., their actions are unknown. In RoboCup, each robot on our team is an actively-controlled object, and the opposing team’s robots are adversarial foreign-controlled objects.

The most essential actively-controlled object for the robot to model is itself. In the egocentric coordinate frame, the robot is always at the origin, so its position is not stored explicitly in the world model. Instead, the relative positions of the other objects are updated according to the robot’s odometry by the update function  $U$ . The robot’s global position on the field is determined by localization.

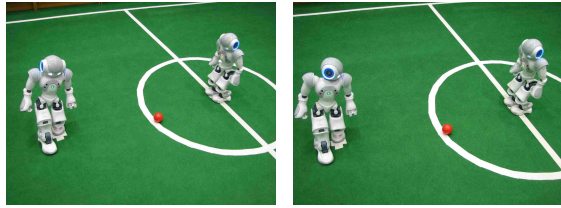
The other actively-controlled objects are the robot’s teammates. Each robot’s global position, computed using localization, is shared wirelessly with teammates. This information is used for team behaviors, such as passing to a robot upfield or backing up an attacker. Although the localization information is prone to error, communicating positions wirelessly has the advantage of uniquely identifying the robots. Furthermore, the robot will know the positions of teammates which are occluded or not in the line of sight.

The opposing robots are detected visually and treated in the world model as if they are static objects. So  $U$  and  $H$  behave similarly for foreign-controlled and static objects. This approximation is reasonable because the Nao’s motion is slow, although bipedal motion algorithms are steadily improving. The behaviors use the positions of opposing robots to attempt to kick away from them, particularly when shooting past the goalie and into the goal.

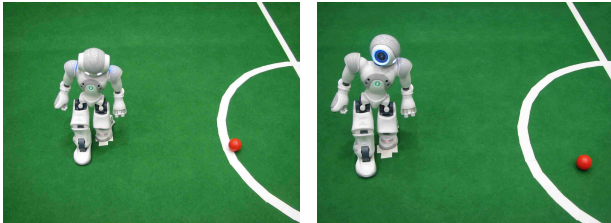
## V. EXPERIMENTAL RESULTS

We ran experiments to test the effectiveness of the modeling of the ball’s position. Specifically, we tested the effect of sharing information between teammates and adding hypotheses for the ball’s position after a kick. We placed two robots two and a half meters apart, one with a ball directly in front.





(a) Ball leaves field of view after kick. (b) Robot looks at teammate hypothesis, sees ball.



(c) Ball leaves field of view after kick. (d) Robot looks at hypothesis from kick, sees ball.

Fig. 7. The robot kicks the ball left, out of its field of view, and we measure the time until it finds the ball again. In (a), the robot does not see the ball after it is kicked, but the robot locates the ball in (b) through a hypothesis from its teammate. In (c), the robot loses the ball after kicking, but locates it again in (d) by searching where a hypothesis was placed based on the properties of the kick.

Both robots tracked the ball with their cameras, performing a scan when the ball was not visible. The robot near the ball kicked the ball to its left towards the second robot. After each kick, we measured the time it took for the ball to enter the robot’s field of view again. We conducted this experiment ten times each for three different world models: one with only a single hypothesis for the ball position, one which incorporated the hypotheses of the teammate, and a third which only included hypotheses based on the predicted strength and direction of the kick (see Fig. 7).

The results are shown in Table I. The robot generally loses sight of the ball during a side kick because the ball moves quickly and is partially obscured by the shoulder while moving. The robot finds the ball again in approximately 6 seconds while performing a scan. Performing a scan is a costly operation due to the limited field of view of the camera, and the elevated position of the head in humanoids.

Using the position of the ball generated by the teammate and a hypothesis based on the properties of the kick both reduce the time spent search for a ball significantly. However, using the kick hypothesis is faster than a teammate estimate. This is partly due to a delay in communications, but mainly occurs because kick hypotheses are proactive rather than reactive — the robot anticipates the ball’s position and moves its head before the ball arrives, rather than waiting for the other robot to sense the ball at its new position.

## VI. CONCLUSION

In the RoboCup Standard Platform League, a highly dynamic and adversarial domain, the humanoid Nao robots must know the positions of the objects on the field in order to win the game. The main contributions of this paper are:

Scenario	Time to See Ball (s)
No Teammate or Kick Hypotheses	$6.13 \pm 2.35$
Teammate Hypothesis Only	$3.99 \pm 1.73$
Kick Hypothesis Only	$1.63 \pm 0.38$

TABLE I

MEAN TIME AND STD. DEV. TO SEE THE BALL AFTER A SIDE KICK

formalization of the general world modeling problem, and a solution to the problem based on categorizing objects as static, passive, actively-controlled, and foreign-controlled. We classify the confidence of modeled objects as valid, suspicious and invalid. A suspicious object is an active feedback mechanism, which serves as a request by the world model to look at the object. Similarly, when the robot looks at an object but is unable to sense it, the object’s confidence decays quickly (making it invalid) to prevent inaccurate information from being used in the robot’s behaviors. Predictions based on a robot’s own actions and sensory input from teammates are incorporated into the world model, and their effectiveness is verified experimentally. Although the presented solution is tailored to the RoboCup domain, it is applicable to general world modeling problems.

## REFERENCES

- [1] Y. Gu and M. Veloso, “Effective Multi-Model Motion Tracking using Action Models,” *Int. Journal of Robotics Research*, vol. 28, pp. 3–19, 2009.
- [2] C. Kwok and D. Fox, “Map-based multiple model tracking of a moving object,” in *Proc. of RoboCup Symposium*, 2005, pp. 18–23.
- [3] N. Mitsunaga, T. Izumi, and M. Asada, “Cooperative Behavior based on a Subjective Map with Shared Information in a Dynamic Environment,” in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 291–296.
- [4] M. Roth, D. Vaill, and M. Veloso, “A real-time world model for multi-robot teams with high-latency communication,” in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 2494–2499.
- [5] H. Utz, F. Stulp, and A. Muhlenfeld, “Sharing Belief in Teams of Heterogeneous Robots,” in *Proc. of RoboCup Symposium*, 2005, pp. 508–515.
- [6] P. Rybski and M. Veloso, “Prioritized Multi-hypothesis Tracking by a Robot with Limited Sensing,” *EURASIP Journal on Advances in Signal Processing*, 2009.
- [7] RoboCup, “RoboCup International Robot Soccer Competition,” 2010, <http://www.robocup.org>.
- [8] RoboCup SPL, “The RoboCup Standard Platform League,” 2010, <http://www.tzi.de/spl>.
- [9] Aldebaran, “Aldebaran Robotics - Nao Humanoid Robot,” 2010, <http://www.aldebaran-robotics.com/pageProjetsNao.php>.
- [10] S. Zickler and M. Veloso, “Efficient Physics-Based Planning: Sampling Search Via Non-Deterministic Tactics and Skills,” in *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2009, pp. 27–34.
- [11] J. Liu, X. Chen, and M. Veloso, “Simplified Walking: A New Way to Generate Flexible Biped Patterns,” in *Proc. of 12th Int. Conf on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, 2009.
- [12] S. Lenser and M. Veloso, “Sensor resetting localization for poorly modelled mobile robots,” in *Proceedings of ICRA-2000, the International Conference on Robotics and Automation*, April 2000.
- [13] T. Hester and P. Stone, “Negative information and line observations for monte carlo localization,” in *Proceedings of ICRA-2008, the International Conference on Robotics and Automation*, May 2008.
- [14] B. Browning, J. Bruce, M. Bowling, and M. Veloso, “STP: Skills, tactics and plays for multi-robot control in adversarial environments,” *IEEE Journal of Controls and Systems Engineering*, vol. 219, pp. 33–52, 2005.