A Real-time World Model for Multi-Robot Teams with High-Latency Communication

Maayan Roth, Douglas Vail, and Manuela Veloso

School of Computer Science
Carnegie Mellon University
Pittsburgh PA, 15213-3891
{mroth, dvail2, mmv}@cs.cmu.edu

Abstract

In this paper, we present in detail our approach to constructing a world model in a multi-robot team. We introduce two separate world models, namely an *individual world model* that stores one robot's state, and a *shared world model* that stores the state of the team. We present procedures to effectively merge information in these two world models in real-time. We overcome the problem of high communication latency by using shared information on an as-needed basis. The success of our world model approach is validated by experimentation in the robot soccer domain. The results show that a team using a world model that incorporates shared information is more successful at tracking a dynamic object in its environment than a team that does not use shared information.

1 Introduction

The need to operate under partial observability and interact with objects in the environment makes the creation of a world model a necessity for most robotic systems. In a multi-robot system, where several agents interact simultaneously with each other and also with shared portions of the environment, the need for a consistent view of the world is even greater. For systems where the primary goal of the system is to cooperatively map an area using several robots, the challenge is to merge information from several agents coherently. (e.g. [7, 8]) However, these observations do not need to be merged in real-time, as the environment tends to be static.

In adversarial domains, such as robot soccer, the environment is dynamic. In addition to knowing the positions of its teammates to facilitate cooperation, the robot must be able to quickly locate the ball and avoid adversarial agents. When using local vision as the primary sensor, soccer-playing agents are usually unable to observe their entire environment. Unless communication between teammates is available, each robot must model its environment without input from other agents. Until recently, it was common for teams competing in RoboCup to build their world models without using shared information. The 1999 RoboCup Agilo RoboCuppers mid-sized robot team provides an example of world model design without communication [1]. In the Sony legged-robot league, the hardware for communication was not available on the robots until 2002. In the absence of communication, teams relied on local sensing to build their world models, and fixed roles for team behavior. (e.g. [11])

The advantages of utilizing communication when it becomes available are obvious. In the RoboCup middle-sized robot competition, each team is able to design their own hardware platform. From the beginning, teams have taken advantage of this freedom by incorporating communication hardware into their team design. For example, the Agilo RoboCuppers added communication to their system for the RoboCup 2000 competition. By using a Kalman filter to fuse information about the locations of objects in the environment, they enabled each robot on their team to use a global world model as if it were its own local model [5]. In another highly successful mid-sized robot team, CS Freiburg, each robot maintains a local world model, but contributes information to a global world model on a single off-board server. This server then sends global world model information back to the individual teammates, allowing them to update their state of the world [4, 3]. However, the presence of communication presents additional challenges. High communication latency can easily prevent teammate agents from forming a synchronized world-view in real-time. Trusting delayed information can introduce more error into an agent's world model than is removed by integrating shared information.

The focus of this paper is to present our solution to the problem of building a real-time world model for a multi-robot team with high-latency communication within the context of the RoboCup Sony AIBO competition. We assume for the purposes of this paper that the robots are able to sense task-relevant objects such as the soccer ball, teammate robots, and opponent robots. The techniques that we describe are applicable to any domain where a

Figure 1: *This is one the Sony AIBO robots for which this world model was implemented. The round aperture at the tip of the robot's nose is the CCD camera that is used to capture visual sensor data.*

robot interacts with a combination of passive objects that can be sensed and manipulated, intelligent agents that can be detected but with whom the robot cannot communicate, and intelligent agents that can communicate with the robot for the purpose of sharing information.

2 Sources of Knowledge for Building State

The 2002 AIBO robots perform all computation and sensing on-board, with the team forming a fully distributed system. The robots have two sources of information that are used to build state: vision and communication. Each robot is equipped with a CCD camera located at the front of its head. (See Figure 1) All relevant objects in the world are color-coded, allowing the identification of an object by its color. The camera information is processed to produce output in the form of (x, y, θ), oriented in the robot's local coordinate system, for all of the objects in the current field of view [11]. The objects that the vision is able to recognize are six color-coded markers at known locations around the field, two goals at either end of the field, the orange ball, and the other robots, which are either blue or red. Figure 2 shows an image of the field.

The robots have an a priori map of the field that gives locations for each of the markers. The locations of the markers as observed by vision are compared to the map and are used by each robot to compute its own location on the field [6, 10]. The output of the localization is two 2-dimensional Gaussian distributions, one for the robot's position and one for the robot's heading. Each Gaussian distribution is characterized by:

- μ, the mean, a 2-d vector of (x, y) position
- σ, the standard deviation, a 2-d vector of (σ_x, σ_y)

This year, wireless communication, in the form of 802.11 Ethernet, was available on the AIBO robots. This communication, although it has low bandwidth and high latency, allows the sharing of state information between teammates. This paper presents our solution to utilizing communicated information effectively, despite being unable to synchronize data streams from different robots due to high latency, and without relying on an external server for centralized information processing.

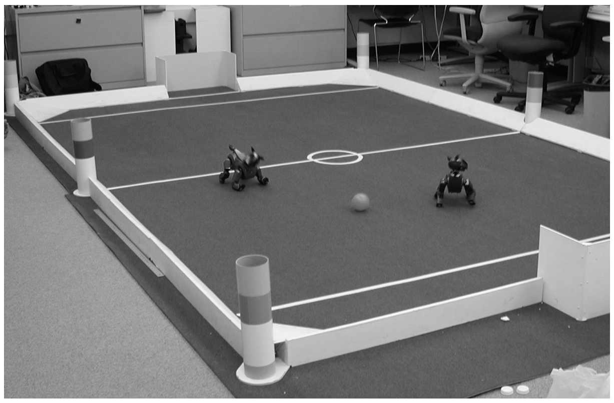

Figure 2: *This image shows two AIBO robots on a regulation-sized field. The vertical cylinders in the corners of the field are the color-coded markers that are used by the robots for localization.*

Using the information acquired by each robot through its own vision and integrating the information communicated between teammates on an as-needed basis, we introduce an approach for representing the world with two separate world models: an individual world model that describes the state of one robot, and a shared world model that describes the state of the team.

3 Individual World Model

Each robot maintains for itself an individual world model that contains its perception of the state of the world. The individual world model is a vector containing the positions of all the task-relevant objects in the robot's environment. In our robot soccer task, it is comprised of:

- *wm_position*, the robot's position
- *wm_heading*, the robot's heading
- *wm_ball*, the location of the ball
- *wm_teammate*, a vector of *n* teammate positions
- *wm_opponent*, a vector of *m* opponent positions

Each element of the individual world model contains two components: a 2-dimensional position stored in global coordinates, and a timestamp, τ. The position is stored as a Gaussian structured to contain the same format of information as the Gaussian parametric distributions described in Section 2.

We identify three distinct classes of objects that are stored in a world model. These are objects that are updated either entirely from vision information, entirely from communicated information, or by using a combination of the two. It is necessary to choose among these three policies to determine an appropriate update procedure for each component of the world model. In our task, a vision-only update is used for the robot's own position *wm_position* and *wm_heading*, and for the opponent position vector, *wm_opponent*. The teammate position vector, *wm_teammate*, relies entirely upon shared information broadcast by each robot. Only the ball position, *wm_ball*, relies on both sources of information, combining its position as returned by vision with information that is shared between teammates.

The updates to the robot's location, *wm_position* and *wm_heading*, come directly from localization, which in turn relies entirely on vision and the robot's internal map of the field. Although the vision returns positions for robots of both colors, making it theoretically possible for teammates to observe each other, the robots are physically identical, making it impossible to distinguish visually between robots on the same team. Therefore, a robot will not be able to improve its own localization estimate by integrating estimates of its position broadcasted by its teammates. For this same reason, it is difficult to accurately update the opponent position vector, *wm_opponent*. The vision module returns a vector of the positions of all opponent robots that were observed. Again, there are no visual characteristics that distinguish one opponent from another. In solving this data association problem, we utilize a greedy approach. The world model attempts to match a new observation of an opponent to a previous observation already stored in the world model by updating the stored position that is physically closest to the newly observed position.

For the reasons detailed above that indicate that a robot's own localization is more accurate than estimates of its position broadcasted by teammates, it is preferable to use the position provided shared by each robot to its teammates when updating the teammate position vector, *wm_teammate*, rather than attempting to integrate both sources of information. When updating, the position of each teammate is requested from the shared world model and stored directly in *wm_teammate*.

The position of the ball, *wm_ball*, is the only element of our world model that benefits from combining sensing information returned by vision with shared information transmitted between teammate. The procedure to update ball position can be broken down into two steps: an update from vision and an update from shared information.

3.1 Update from Vision

The most accurate estimate of ball position, *wm_ball*, is extracted from the information returned by vision and updated as described in Table 1. Because vision returns the locations of objects in coordinates local to the robot, whereas the positions are stored in global coordinates in the world model, it is necessary to convert all object positions into global coordinates. If vision reports that the ball has been seen, it is merged with the old ball position [9, 2]. The merge method takes advantage of the property of Gaussian distributions that states that the product of two Gaussians is also a Gaussian. By multiplying the two position estimates, taking into account their standard deviations, we end up with an estimate that is a weighted average of the old position and the new observation. Because we grow uncertainty with time, old information is given less weight than new information that starts with the default small standard deviation, SMALL_ERROR, allowing us to converge to the correct ball position with relatively few observations. However, by not discarding the old ball position out of hand, we are able to maintain a smoother estimate of ball position that does not fluctuate drastically as a result of spurious sensor readings. When merging the ball positions, it is important to limit the standard deviation, σ_{wm_ball}, to no less than the default minimum confidence value, SMALL_ERROR, to prevent it from becoming vanishingly small.

```
Procedure UPDATEVISION(ball_pos, op_pos, $\tau_{current}$)
  Update the ball position.
    if ball_pos $\neq$ NIL
      $\mu_{global} = \mu_{wm\_position} +$
        $\text{ROTATE}(\mu_{ball\_pos}, \mu_{wm\_heading})$
      $\sigma_{global} = \text{SMALL\_ERROR}$
      MERGE(wm_ball, $\{\mu_{global}, \sigma_{global}\}$)
      $\tau_{wm\_ball} = \tau_{current}$
```

Table 1: *Procedure to Update from Vision*

3.2 Update from Shared Information

When vision has failed for a certain period of time to provide information about the position of the ball, the ball position is updated, as in Table 2, from information stored in the shared world model. The format of the shared world model is described in Section 4.

As explained in Section 2, the communication latency between robots is extremely high. Each robot receives information from its teammates, on average, every .5 seconds, but the latency was occasionally observed to be as high as 5 seconds. Additionally, because timestamps associated with the data are local to each robot and cannot be matched between robots, it is impossible to integrate shared information via a Kalman filter, unlike other implementations that did allow this approach [5]. Because of these restrictions, we use the shared ball information sparsely, and only when the ball cannot be easily located by an individual robot. If the ball has not been observed by the robot for a period of time greater than

$\tau_{threshold}$, the best available ball location is requested from the shared world model, using the GETBALLLOCATION function. (See Table 4 and the accompanying description in Section 4.)

```
Procedure UPDATESHAREDINFORMATION($\tau_{current}$)
    if $\tau_{current} - \tau_{wm\_ball} > \tau_{threshold}$
        shared_ball = GETBALLLOCATION($\tau_{current}$, robot_id)
        if shared_ball $\neq$ NIL
            wm_ball = MERGE(wm_ball, shared_ball)
            $\tau_{wm\_ball} = \tau_{current}$
```

Table 2: *Procedure to Update from Shared Information*

3.3 Accounting for Aging Information and Localization Changes

Because the robot soccer environment is dynamic, we expect objects to move over time from where the robot last observed them. Although we intend to investigate velocity-tracking in the future, we present here a position-only world model that does not attempt to track velocities. To account for unobserved motion of objects without knowing their velocities, we grow our uncertainty for any object in the individual world model that was not observed in the last time step by adding SMALL_ERROR to each object's standard deviation.

The localization module and the individual world model are updated at different times during the system execution, making it necessary to correct the world model to account for changes in localization information. When the robot executes a localization update due to seeing a marker, its estimate of its own position changes, even though its physical position has not changed. To ensure consistency between the individual and the shared world models, objects in both world models are stored in global coordinates. However, this means that changes in the robot's knowledge of its position caused by seeing a marker also make it appear to the robot as if the other objects in its environment have suddenly changed position with respect to itself. Because we need to know the position of the ball with high accuracy at all times, it is necessary to correct the position of the ball to account for this shift immediately. The procedure in Table 3 was implemented to correct for this source of error.

4 Shared World Model

The shared world model is a fully distributed data structure, with each robot maintaining its own on-board copy. The contents of each robot's shared world model are:

- *swm_position*, a vector of *n* teammate positions
- *swm_ball*, a vector containing each teammate's estimate of the ball position

```
Procedure SHIFTBALL()
    Get robot_position and robot_angle
    Shift the ball position into local coordinates:
        $\mu_{to\_local} = \text{ROTATE}(\mu_{robot\_position}, -\mu_{robot\_angle})$
        $\mu_{wm\_ball} = \mu_{wm\_ball} - \mu_{to\_local}$
    Do the localization update from the sensor reading.
    Get the updated robot position and heading.
    Shift the ball back into global coordinates:
        $\mu_{to\_global} = \text{ROTATE}(\mu_{wm\_ball}, \mu_{robot\_angle})$
        $\mu_{wm\_ball} = \mu_{robot\_position} - \mu_{to\_local}$
```

Table 3: *Correction for Localization Shift*

- *swm_goalie*, a vector containing a flag for each teammate, indicating whether or not that robot is the goal keeper
- *swm_sawball*, a vector containing a flag for each teammate, indicating whether or not that robot saw the ball in the last time step

The last flag, *swm_sawball* is important because it prevents other robots from incorporating old or second-hand information into their individual world models when they receive an update from this robot. Each element in *swm_position* and *swm_ball* is made up of a 2-dimensional Gaussian and a timestamp, τ, as in the individual world model.

Updates to the shared world model occur asynchronously, with each robot updating its model whenever it receives a broadcast from a teammate. This means that communication latency or dropped messages may cause the shared world model contents to differ among robots. By not requiring synchronization between teammates, we avoid the communication overhead required to synchronize. Each robot broadcasts its own shared information at a rate of 2 Hz. Although this seems slow, it is due in part to bandwidth limitations. Additionally, because the high and variable latency prevents us from using the shared information for fine-grained control, there is no reason to broadcast at a higher rate.

Table 4 shows the procedure that is used by the individual world model to access information stored in the shared world model. The GETBALLLOCATION procedure determines which, among all the ball positions estimates reported by the team members, is the 'best' estimate of the true ball position. In the future, we may find it worthwhile to attempt to merge ball estimates as they are reported by different teammates. However, in the current implementation, we attempt to select the ball estimate that has the lowest uncertainty, and which has been observed within a reasonable period of time, $\tau_{threshold}$. We do not allow a robot to retrieve its own reported estimate from the shared world model, as this would only reinforce the robot's belief without adding new information. Additionally, we require the uncertainty to be below $\sigma_{threshold}$, a maximum allowable uncertainty.

```
Procedure GETBALLLOCATION($\tau_{current}$, robot_id)
    ball = NIL
    best_confidence = $\sigma_{threshold}$
    for $i = 1 \ldots n$
        if $i \neq$ robot_id
            if ISVALID($i$, $\tau_{current}$)
                if $\sigma_{swm\_ball_i} <$ best_confidence
                    best_confidence = $\sigma_{swm\_ball_i}$
                    ball = $swm\_ball_i$
    return ball

Procedure ISVALID($i$, $\tau_{current}$)
    if $i < 0$ or $i > n$
        return FALSE
    if $\tau_{current} - \tau_{swm\_ball_i} > \tau_{threshold}$
        return FALSE
    if $\sigma_{swm\_ball_i} > \sigma_{threshold}$
        return FALSE
    if $swm\_sawball_i \neq$ false
        return FALSE
    return TRUE
```

Table 4: *Procedure to Get the Best Ball Location*

5 Experimental Results

The shared and individual world models contributed in this paper are fully implemented and were used by the CM-Pack'02 legged-robot team in the 2002 RoboCup competition. The team performed extremely well, winning the competition to become the world champion.

In order to experimentally verify the efficacy of the world model separate from the overall performance of the team in competition, we designed a controlled experiment in the lab. We compare the behavior of a robot team using our world model, constructed with both sensor and shared information, to an identical robot team using only local sensor information for determining ball location.

The robot behaviors are comprised of many behavior states, some of which can execute simultaneously. Each robot transitions between states as a function of its individual world model, its localization, and some teamwork objective functions [12]. A new behavior is chosen 20 times per second. During the execution of most behaviors, such as positioning itself on the field or walking towards the ball, the robot opportunistically observes the world, updating its world model and localization as it sees markers or objects. As the robot's uncertainty about the position of the ball grows, it transitions between the states illustrated in Figure 3.

The robot's certainty about the position of the ball is highest in State 1, when the ball has been seen within the last time step. If the ball has been seen recently,

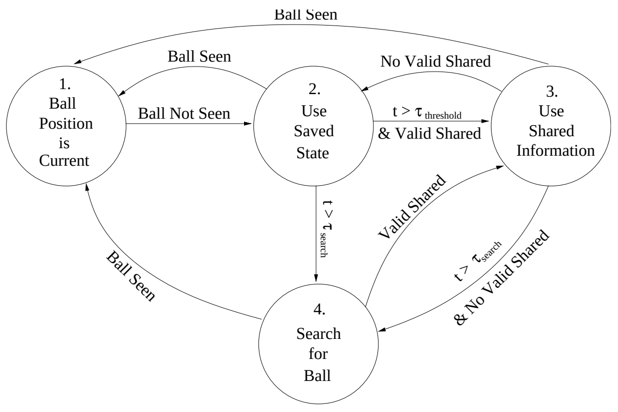

Figure 3: *Different degrees of certainty about ball position through which the robot transitions during execution.*

the robot operates in State 2, using the position of the ball saved in its individual world model. State 3 is triggered when the robot chooses to request information from its shared world model, as described in Section 3, and valid shared information is returned. If there is no valid shared information available, and the age of the ball position information in the individual world model grows beyond a threshold, τ_{search}, the ball is considered "lost", and the robot transitions into State 4, where the behavior SEARCH_SPIN is executed. The threshold of time without knowing the position of the ball that triggers a transition into the SEARCH_SPIN behavior was chosen to be approximately 5 seconds. In this behavior, which interrupts the robot's previous behavior, the robot spins in place, attempting to locate the ball on the field. Because this behavior interrupts other behaviors, we seek to minimize its occurrence.

In the experiment that we conducted, we ran two teams, each comprised of three robots, in two soccer games against each other. The teams are identical, with three attacker robots each. However, in one team, the robots were not permitted to use shared information to update the ball position. This is equivalent to forbidding transitions into State 3 as it is depicted in Figure 3. Each game lasted around 10 minutes, during which the ball was replaced in the center of the field whenever a goal was scored, but the robots were not moved back to their starting positions. Each attacker robot wrote to an on-board log file every frame, and indicated whether or not it was currently executing the SEARCH_SPIN behavior, and if this cycle was a new transition into the SEARCH_SPIN behavior. Table 5 shows the results of our experiment. In this table, SHARED refers to the teams that used shared information and NO_SHARED refers to the teams that did not share ball information. The "# starts" column indicates the number of times that the robots transitioned into the SEARCH_SPIN behavior. The "cycles in search" column gives the total number of behavior cycles in which the SEARCH_SPIN behavior was executed.

	total cycles	cycles in search	% cycles searching	# starts
SHARED	95437	1754	1.84 %	74
NO_SHARED	101076	19683	19.47 %	330

Table 5: *Comparing how often the ball is lost by counting behavior cycles spent in the SEARCH_SPIN behavior, with and without shared information*

The robots use the confidence and timestamp values stored in the individual world model to determine when to transition into the SEARCH_SPIN behavior. Therefore, we consider the SEARCH_SPIN behavior to provide an accurate estimate of how frequently the individual world model considers the ball to be lost. Without shared information from their teammates, the robots considered the ball to be lost for 19.47% of the game time, over 10 times more than the robots that did incorporate shared information. They were forced to interrupt their game-playing behavior to search for the ball on average every 306 frames, 4.2 times more often than the team incorporating shared information. By effectively integrating information that is shared between cooperative agents, as demonstrated by these results and the results shown in [12], we are able to minimize the instances in which the robots are unable to locate the ball, thus improving the performance of our robots over what they would be able to achieve without cooperation.

6 Conclusion

In this paper, we presented our approach for building a real-time world model for a fully distributed multi-robot team. Communication between teammates on our hardware platform had high and variable latency, making it impossible to synchronize with sensor data that arrived predictably at 20 Hz. Despite this challenge, we were able to utilize shared information effectively by building both an individual and a shared world model for each agent, and using shared information only when appropriate and on an as-needed basis.

Our experimental results clearly show that sharing information about the state of the world with teammates helps robots to overcome the problem of partial observability when locating relevant objects in their environment. By using both the individual and the shared world models, the robots were more aware of the position of the task relevant dynamic object (e.g the ball), and needed to interrupt their goal-directed behaviors to re-acquire its position with lower frequency.

Communication, even with lower latency, and the need to merge state information gathered by local sensing with collaborative information shared between robots on the same team, are ongoing challenges for the multi-robot domain. Our approach, detailed in this paper, contributes a step towards meeting this challenge by providing a method that is applicable to any multi-robot team that operates under equivalent real-time and high latency communication conditions.

References

[1] Thorsten Bandlow, Michael Klupsch, Robert Hanek, and Thorsten Schmitt. Fast image segmentation, object regocnition and localization in a RoboCup scenario. In *RoboCup-99: Robot Soccer World Cup III*, pages 174–185, 2000.

[2] Silvia Coradeschi and Alessandro Saffiotti. Anchoring symbols to vision data by fuzzy logic. *Lecture Notes in Computer Science*, 1638:104–115, 1999.

[3] Markus Dietl, Jens-Steffen Gutmann, and Bernhard Nebel. CS Freiburg: Global view by cooperative sensing. In *RoboCup 2001 International Symposium*, 2001.

[4] Jens-Steffen Guttman, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Rittinger, Augustinus Topor, and Thilo Weigel. The CS Freiburg team: Playing robotic soccer based on an explicit world model. *The AI Magazine*, 2000.

[5] R. Hanek, T. Schmitt, M. Klupsch, and S. Buck. From multiple images to a consistent view. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 169–178, 2001.

[6] Scott Lenser and Manuela Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of ICRA-2000*, 2000.

[7] Lynne E. Parker, Kingsley Fregene, Yi Guo, and Raj Madhavan. Distributed heterogeneous sensing for outdoor multi-robot localization, mapping, and path planning. In Alan C. Schultz and Lynne E. Parker, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer Academic Publishers, 2002.

[8] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001.

[9] Ashley W. Stroupe, Martin C. Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of ICRA 2001*, 2001.

[10] Ashley W. Stroupe, Kevin Sikorski, and Tucker Balch. Constraint-based landmark localization. In *Proceedings of 2002 RoboCup Symposium*, 2002.

[11] William Uther, Scott Lenser, James Bruce, Martin Hock, and Manuela Veloso. CM-Pack'01: Fast legged robot walking, robust localization, and team behaviors. In *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*, 2001.

[12] Douglas Vail and Manuela Veloso. Dynamic multi-robot coordination. In *Multi-Robot Systems*. Kluwer, 2003.