A World Model for Multi-Robot Teams with 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 principle, a robot member of a multi-robot team with teammates that can communicate their own sensing can build a more accurate world model by incorporating shared information from its teammates than by relying only on its own sensing. However, in practice, building a consistent world model that combines a robot's own sensing with information communicated by teammate robots is a challenging task. In this paper, we present in detail our approach to constructing such 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. 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 and robust at tracking a dynamic object in its environment than a team that does not use shared information. The paper includes a comprehensive description of the data structures and algorithms, as implemented for our CM-Pack'02 team, which because the RoboCup 2002 world champion in the Sony legged-robot league.**

I. 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 shared portions of the environment, the need for a consistent view of the world is even greater. For systems like the ones described in [6] and [7], 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. However, these observations do not need to be merged in real-time, as the environment tends to be static.

In adversarial domains, such as robotic 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. An example of world model design without communication for is presented by [1] in their description of the 1999 RoboCup Agilo RoboCuppers mid-sized robot team. In the Sony legged-robot league, the hardware for communication was not available on the robots until 2002, so all teams were forced to rely entirely on local sensing to build their world models. [10] describes the pre-communication implementation of the CM-Pack'01 legged robot team.

The advantages of utilizing communication when it becomes available are obvious. 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 [4]. Another highly successful mid-sized robot team, CS Freiburg, designed a system where 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 [3], [2].

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

II. Sources of Knowledge for Building State

The 2002 AIBO 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. All relevant objects in the world are color-coded, allowing the unique recognition of an object by its color. The camera information is processed as described in [10], to produce output in the form of (x, y, θ), in the robot's local coordinate system, for all of the objects in the current field of view. The objects that the vision system 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.

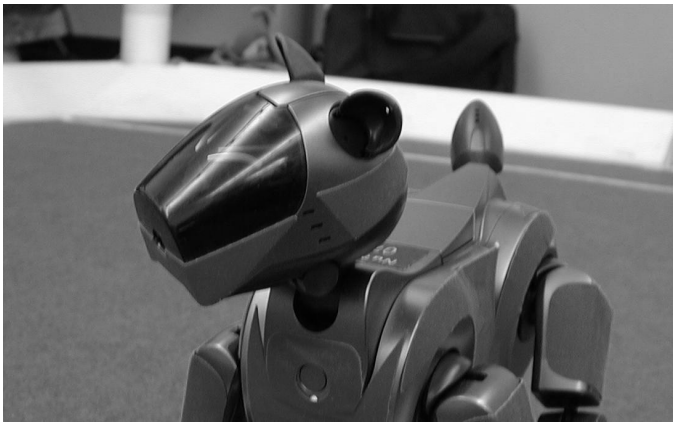

Fig. 1. This is one the Sony AIBO robots for which this world model was implemented. The round aperture at the tip of the robot's nose is the CCD camera that is used to capture visual sensor data.

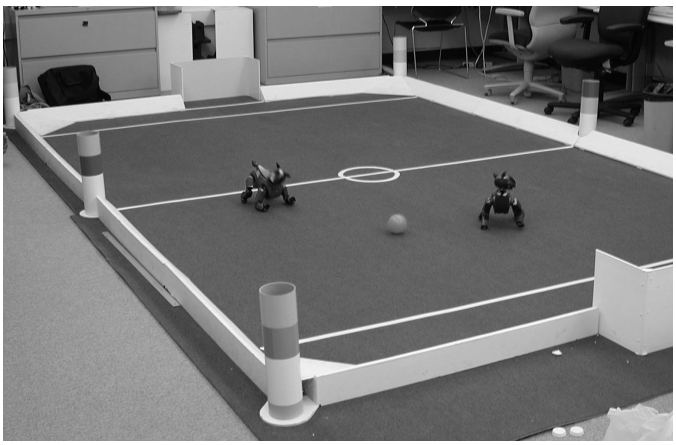

Fig. 2. This image shows two AIBO robots on a regulation-sized field. The vertical cylinders in the corners of the field are the color-coded markers that are used by the robots for localization.

The known locations of the markers observed by the vision module are used by each robot to compute its own location on the field, using the method detailed in [5] and [9]. The output of the localization module is two 2-dimensional Gaussian distributions, one for the robot's position and one for the robot's heading. Each Gaussian distribution is comprised of:

- 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 added by Sony as a standard feature of 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.

Using the information acquired by each robot through its own vision system and the information communicated between teammates, 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.

III. 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 data structure 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 is stored in global coordinates as a 2-dimensional Gaussian structured to contain the same format of information as the Gaussian parametric distributions described in Section II. We do this to ensure that data representation remains uniform across all the modules of the system. Each object also has associated with it a timestamp, τ.

Each element of the individual world model is updated by information either from the vision module, the communication module, or a combination of the two. The robot's own position, *wm_position* and *wm_heading* comes directly from the localization module, which in turn receives its input solely from vision. The opponent position vector, *wm_opponent*, is also determined entirely from vision information. The teammate position vector, *wm_teammate*, however, is determined entirely from shared information communicated by the teammate robots. Although the vision module returns positions for robots of both colors, making it possible to extract some teammate information from the vision, this information is so noisy that it is discarded entirely in favor of the more accurate shared information. The position of the ball, *wm_ball*, is calculated by combining its position as returned by the vision module with information that is shared between teammates.

The individual world model is updated by calling routine described in Table I.

Procedure UpdateWorldModel(*robot_position, robot_angle, ball_pos, op_pos*, $\tau_{current}$)
  UpdateLocalization(*robot_posn, robot_ang*)
    wm_position = robot_position
    wm_heading = robot_angle
  UpdateVision(*ball_pos, op_pos*, $\tau_{current}$)
  UpdateSharedInformation($\tau_{current}$)
  UpdateTime($\tau_{current}$)

TABLE I
Individual World Model Update Procedure

The procedure that updates the world model to account for new localization information simply copies the localization information into *wm_position* and *wm_heading*. This requires no processing, as the input data from the localization module, *robot_position* and *robot_angle*,is already in the format used by the world model. Additionally, because the objects in the individual world model are stored in global coordinates, they do not need to be shifted to account for the change in robot position.

A. Update from Vision

Both the ball position, *wm_ball*, and the opponent position vector, *wm_opponent*, are updated from the information re-

turned by the vision module, as described in Table II. The vision module returns the observed ball position, *ball_pos*, and a vector of observed opponent positions, *op_pos*. The update is comprised of two major steps: updating the ball position, and updating the position of opponents. Because the vision module 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 the vision module reports that the ball has been seen, it is merged with the old ball position, using the method detailed in [8]. 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, with their appropriate 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.

It should be pointed out that in earlier implementations of the individual world model, we experimented with updating the ball position without merging with old information. Instead, the position reported by vision was trusted immediately. Although this method allowed for faster response time when locating the ball, it was subject to noise due to sensor error, and was discarded after experimentation.

The vision module returns a vector of the positions of all opponent robots that were observed. However, as the robots are identical, there are no visual characteristics that distinguish one opponent robot from another. Instead, we developed a method that attempts to match a new observation of an opponent robot to a previously observed opponent and then updates its location. If no matching robot is found to be within OP_THRESHOLD, the maximum allowable distance, of the new observation, the oldest position in the opponent vector is merged with the new value.

Procedure UPDATEVISION(*ball_pos*, *op_pos*, $\tau_{current}$)
Update the ball position.
if *ball_pos* $\neq$ NIL
$\mu_{global} = \mu_{wm\_position} + \text{ROTATE}(\mu_{ball\_pos}, \mu_{wm\_heading})$
$\sigma_{global} =$ SMALL_ERROR
MERGE(*wm_ball*, $\{\mu_{global}, \sigma_{global}\}$)
$\tau_{wm\_ball} = \tau_{current}$
Update the opponent vector.
for $i = 1$ **to** SIZE(*op_pos*)
$\mu_{global} = \mu_{wm\_position} + \text{ROTATE}(\mu_{op\_pos_i}, \mu_{wm\_heading})$
$j = \arg\min_k \|\mu_{wm\_opponent_k} - \mu_{global}\|$
$dist = \|\mu_{wm\_opponent_j} - \mu_{global}\|$
if $(dist <$ OP_THRESHOLD$)$
$\sigma_{global} =$ SMALL_ERROR
MERGE($\{\mu_{global}, \sigma_{global}\}$, $wm\_opponent_j$)
$\tau_{wm\_opponent_j} = \tau_{current}$
else
$j = \arg\max_k(\tau_{current} - \tau_{wm\_opponent_k})$
$\sigma_{global} =$ SMALL_ERROR
MERGE($\{\mu_{global}, \sigma_{global}\}$, $wm\_opponent_j$)
$\tau_{wm\_opponent_j} = \tau_{current}$

TABLE II
PROCEDURE TO UPDATE FROM VISION

B. Update from Shared Information

The ball position and the teammate position vector are updated, as in Table III from information stored in the shared world model. The format of the shared world model is described in Section IV.

As explained in Section II, the communication latency between robots is extremely high. Each robot receives information from its teammates, on average, every .5 seconds, but the latency was 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, as was done in [4]. 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 described in Section IV.

Because the vision information that is returned for observations of robots, both teammates and opponents, is extremely noisy, it is always preferable to use the position provided by each teammate, rather than attempting to integrate the two sources of information. In the update, the position of each teammate is requested from the shared world model and stored in *wm_teammate*.

Procedure UPDATESHAREDINFORMATION($\tau_{current}$)
If the ball has not been seen in a long time, request its location from the shared world model.
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*)
$\tau_{wm\_ball} = \tau_{current}$
Get teammate location from the shared world model
for $i = 1$ **to** *n*
$wm\_teammate_i =$ GETTEAMMATELOCATION(*i*)

TABLE III
PROCEDURE TO UPDATE FROM SHARED INFORMATION

C. Update from Time

Because the robot soccer environment is dynamic, we expect objects to move over time from where the robot last observed them. However, we present here a position-only world model that does not attempt to track velocities, although we intend to investigate velocity-tracking in the future. 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.

Procedure UPDATETIME($\tau_{current}$)
If any object has not been updated this time period, add some error to its standard deviation.
if $\tau_{wm\_ball} \neq \tau_{current}$
$\sigma_{wm\_ball} = \sigma_{wm\_ball} +$ SMALL_ERROR
for $i = 1$ **to** *m*
$\sigma_{wm\_opponent_i} = \sigma_{wm\_opponent_i} +$ SMALL_ERROR

TABLE IV
PROCEDURE TO UPDATE FOR TIME

D. Accounting for Localization Changes

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 V was implemented to correct for this source of error.

Procedure SHIFTBALL()
Get *robot_position* and *robot_angle*, the current robot position and heading.
Shift the ball position into local coordinates:
$\mu_{to\_local} =$ 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:
$\mu_{to\_global} =$ ROTATE$(\mu_{wm_ball}, \mu_{robot_angle})$
$\mu_{wm\_ball} = \mu_{robot\_position} - \mu_{to\_local}$

TABLE V
CORRECTION FOR LOCALIZATION SHIFT

IV. 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
- *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.

The shared world model also contains two methods (Table VI and Table VII) that are relevant to this paper. These methods are used by the individual world model to access information stored in the shared world model. The GETTEAMMATELOCATION function is straightforward; it returns the position of the requested teammate as it 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 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 $\tau_{current} - \tau_{swm\_ball_i} > \tau_t hreshold$
return FALSE
if $\sigma_{swm\_ball_i} > \sigma_{threshold}$
return FALSE
if $swm\_sawball_i \neq$ **false**
return FALSE
return TRUE

TABLE VI
PROCEDURE TO GET THE BEST BALL LOCATION

Procedure GETTEAMMATELOCATION(*i*)
return $swm\_position_i$

TABLE VII
PROCEDURE TO GET THE LOCATION OF A TEAMMATE

V. EXPERIMENTAL RESULTS

The shared and individual world models presented in this paper were used by the CM-Pack'02 legged-robot team in the 2002 RoboCup competition that took place in Fukuoka, Japan. The team performed extremely well, winning the competition to become the world champion. In order to experimentally verify the efficacy of the world model separately from the overall performance of the team in competition, we compared the behavior of a robot team using this world model, constructed with both sensor and shared information, to a robot team using only sensor information for determining ball location. We had originally intended to use an overhead camera to record the position of the ball on the soccer field and compare it to the robot's estimate of the ball position. However, while running that experiment, we discovered that the robots themselves physically occlude the ball with their bodies, preventing it from being seen by the overhead camera for as many as 36% of the time steps recorded. Our current system for tracking the ball from overhead does not account for ball occlusion, thereby reducing its efficiency for data tracking. We will be enhancing our global vision system in the future to account for this problem.

The robot behaviors for this system are comprised of many behavior states, some of which can execute simultaneously. Each robot transitions between states due to the contents of its individual world model, the output of its localization module, and the output of several potential functions, described in [11]. 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. If its uncertainty about its position or the position of the ball grows above a certain threshold, the robot executes an active localization behavior, where it turns its head in the expected direction of markers or the ball, in the hope of observing relevant features. This behavior executes concurrently with other robot behaviors and does not cause interruption. However, if the ball has not been observed for a long time, and no other information has allowed the robot to reduce its uncertainty about ball position, the ball is considered "lost", and the robot transitions into a behavior called SEARCH_SPIN. 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 several soccer games against each other. The teams used identical software, and each had two attacker robots and a goal keeper. Each game lasted around 10 minutes, during which the ball was replaced in the center of the field if a goal was scored, but the robots were not moved back to their starting positions. After each 10 minute trial, the robot batteries were changed, and the robots were restarted from their initial configurations. Each attacker robot wrote to an on-board log file the time for which it was active and each instance when it transitioned into the SEARCH_SPIN behavior. We were only interested in the attacker's logs because the goal keeper robots are not permitted to execute the SEARCH_SPIN behavior. We ran two trials each of fully functional teams, using both sensors and shared information to construct its world model, and teams from which all instances of ball information-sharing was removed. Table VIII shows a summary of the data collected. SHARED refers to the teams that use shared information and NO_SHARED refers to the teams that did not share ball information. The "# SEARCH_SPIN" column gives the raw counts of how many times the SEARCH_SPIN behavior was triggered. This reflects only the number of times that the behavior began, and does not adequately represent the amount of time that the robots spent executing the SEARCH_SPIN behavior. Although we do not currently have data to support this observation, it is our belief, formed through long periods of observing the teams, that the robots utilizing shared information not only transition into the SEARCH_SPIN behavior less frequently than robots without shared information, but also spend considerably less time executing the behavior once it has begun. The final column in Table VIII represents the number of transitions into the SEARCH_SPIN behavior per minute.

	time (min)	# SEARCH_SPIN	SEARCH_SPIN per minute
SHARED	87.35	192	2.198
NO_SHARED	88.84	418	4.705

TABLE VIII
COMPARING HOW OFTEN THE BALL IS LOST BY COUNTING TRANSITIONS INTO 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 lost the ball 2.14 times more frequently than robots that did incorporate shared information. This causes them to interrupt their behaviors to search for the ball more frequently, reducing their effectiveness at accomplishing the task of playing soccer. By effectively integrating information that is shared between cooperative agents, as demonstrated by these results and the results shown in [11], 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.

VI. CONCLUSION

The results of our experiment 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 ball, and needed to interrupt their behaviors to search for the ball with lower frequency. Although the communication available for our use had high and variable latency, making it impossible to synchronize with sensor data that arrived predictably at 25 HZ, we were able to utilize shared information effectively by using it only on an as-needed basis.

As the AIBO hardware continues to evolve, we hope that lower latency communication will become available for our use. This will enable us to conduct future investigations such as the benefit of simultaneously observing an object from multiple locations and merging observations. These observations can be especially important for tracking of a moving object like the ball. Even without hardware improvements, the accuracy of opponent detection in our current model remains to be determined. We hope also to improve our ability to observe the environment using an overhead camera, both to enable us to compare our robots' perceptions of the world to the ground truth of the world state, and to investigate the integration of global information with local sensing and communication.

VII. Acknowledgements

The authors would like to thank the other members of the CM-Pack'02 legged-robot team, Scott Lenser, Ashley Stroupe, Sonia Chernova, and Jim Bruce, for their hard work during the development of our team for this year's RoboCup competition. The authors would also like to thank Brett Browning for his valuable suggestions during the writing of this paper.

This research was sponsored by Grants No. DABT63-99-1-0013, F30602-00-2-0549, and by generous support by Sony, Inc. This material was based upon work supported under a National Science Foundation Graduate Research Fellowship. The content of this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] Douglas Vail and Manuela Veloso. Multi-robot dynamic role assignment and coordination through shared potential fields. Proceedings of ICRA 2003 (submitted).