A distributed world model for robot soccer that supports the development of team skills

Frans C.A. Groen Jeroen Roodhart Matthijs Spaan
Raymond Donkervoort Nikos Vlassis

Computer Science Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{groen,roodhart,mtjspaan,rdkvoort,vlassis}@science.uva.nl

Abstract

In this paper a distributed dynamic world model is presented together with the team skills for the Dutch robot soccer team. It uses a shared world model which is locally maintained in each robot. Odometry and landmarks are used to calculate the robot's own position together with the time-stamped position of the moving objects present. Taking into account the uncertainty present, the time-stamped measurements of the whole team are fused in each robot and the position and velocity of moving objects are calculated. The main benefit of a distributed dynamic world model is that team coordination is greatly simplified. Sharing the same world model enables the team to reason not only about their own actions but they can also predict the next action a team member will take.

1 Introduction

Future robots will be intelligent on-line embedded systems, which are able to operate in human inhabited dynamic environments. Applications are in service robotics, intelligent transport systems, public safety systems and rescue scenarios to mention a few. Robot soccer has become a standard testbed for these types of Real-World Multi-Agent systems in the development and integration of the broad variety of technologies needed such as mechatronics, control theory, architecture of distributed systems, computer vision, self-learning systems, cooperative autonomous systems. Essential for these types of applications is that there is a team of autonomous systems working together. The systems perceive their environment, are able to move in dynamic environments, and communicate with each other, resulting in a shared world model of the environment. A distributed control technique must be used to obtain team behavior, instead of traditional central control systems. This makes the whole system robust for erroneous perception or malfunctioning of one or more robots.

Scientific challenges include: how to create distributed dynamic world models, how to learn and adapt effective multi-agent strategies in dynamic situations,

how to integrate the different subsystems and many more. In this paper we will focus on the distributed dynamic world model and the team skills for the robot soccer scenario. The corresponding Dutch research project involves the Dutch RoboSoccer team called 'Clockwork Orange' with three participating universities, the University of Amsterdam, the Delft University of Technology, and the University of Utrecht.

2 A distributed dynamic world model

Robot soccer constitutes a realization of a complex system where moving and static objects coexist and interfere with each other in a time-dependent fashion. For the analysis, it is convenient to assume two basic elements in the robot soccer world: (i) a static environment consisting of the soccer field, lines, goals, etc., and (ii) a set of moving objects consisting of two teams of mobile robots (teammates and opponents), and the ball.

In principle, any team strategy requires knowledge of the state (position, orientation, and velocity) of each robot, both for teammates and opponents, as well as the state (position and velocity) of the ball. Let $X^t = \{x_i^t\}$, $i = 1, \ldots, 4$, collectively denote the state of the four teammate robots at time step t, O^t the state of the four opponents, and B^t the state of the ball, combined in a state vector $S^t = [X^t, O^t, B^t]$. Moreover, at each time step t every teammate robot observes a sensor reading y_i^t, in our case an image snapshot, collectively denoted for all teammate robots by $Y^t = \{y_i^t\}$.

Modeling the system as a Markovian state-space model requires two stochastic models, a state transition model in the form $p(S^t|S^{t-1})$ that relates previous states with new ones according to the kinematics of the robots and the ball, and an observation model $p(Y^t|S^t)$ that assigns likelihood values to sensor profiles observed at certain states. With the above definitions, a generic state prediction scheme is given by the following iterative scheme (Bayes' rule)

$$p(S^t|Y^t) = \alpha \; p(Y^t|S^t) \int p(S^t|S^{t-1}) p(S^{t-1}|Y^{t-1}) dS^{t-1} \tag{1}$$

where $p(S^{t-1}|Y^{t-1})$ is the filtered state density at the previous time step, and α is a normalizing constant making $p(S^t|Y^t)$ integrate to one. This is a general probabilistic framework which has been successfully used in the context of single robot localization [9], where the main assumption is the existence of a single dynamic object (the robot) and a static environment. Known prediction mechanisms like the Kalman filter or the recent particle filter [4] are particular instances of the above iterative update equation (1).

2.1 Multi-robot localization

The central aspect that distinguishes a multi-robot system from a single-robot one is the fact that each robot can observe other teammate robots and then communicate this information around [5, 7]. This way, each robot can have an estimate of

its own state based on a self-localization mechanism like the one described above, but it can also have additional, fused estimates of its own state based on the communicated information from its teammates. Similarly, the state of the opponent robots and the ball can be collaboratively estimated by all teammate robots.

In formal terms we can assume that the observation vector y_i^t of a robot i at time t can be decomposed into $y_i^t = [y_i^t(e), y_i^t(x)]$, where $y_i^t(e)$ denotes the observed static environment, i.e., the unoccluded soccer field features, and $y_i^t(x, o, b)$ denotes the observed moving objects, i.e., other teammate robots, opponents, and the ball. Using the $y_i^t(e)$ part of its observation vector, the robot i can localize itself using (1) by computing the posterior density

$$p(x_i^t|y_i^t(e)) \propto p(y_i^t(e)|x_i^t) \int p(x_i^t|x_i^{t-1}) p(x_i^{t-1}|y_i^{t-1}) dx_i^{t-1}, \tag{2}$$

where the proportionality symbol implies normalization to unit integral.

Moreover, if $y_j^t(x_i)$ is an observation of another teammate robot $j \neq i$ that involves the robot i, the estimate of the state of the robot i can be further improved by applying Bayes' rule again and treating the above posterior as prior

$$p(x_i^t|y_j^t(x_i)) \propto p(y_j^t(x_i)|x_i^t) p(x_i^t|y_i^t(e)) = p(x_i^t|y_i^t(e)) \int p(y_j^t(x_i)|x_j^t, x_i^t) p(x_j^t) dx_j^t. \tag{3}$$

In the last formula, the quantity $p(y_j^t(x_i)|x_j^t, x_i^t)$ can be regarded as the likelihood that the robot j being at state x_j observes at time t the robot i at state x_i. Further observations from teammate robots can be incorporated in the above framework to get fused posterior state estimates.

In our robot team, the kinematics model is given by odometry and the observations are CCD camera images. In the odometry module the translation and/or rotation of the robot is calculated from the rotation of the wheels, where the uncertainty due to slip increases with time. The typical odometry induced error has been estimated by running an UMBBench procedure as described in [1]. For the observation model, a two-tiered approach is used for computing the likelihood of an observation based on a matching procedure of observed line segments to a model of the soccer field. Details about this method can be found in [3].

2.2 Lag filtering

A second characteristic of multi-robot systems is that the communicated information may arrive with delays due to limited network resources (latencies, low bandwidth, etc.), and the above inference cannot always be carried out instantaneously. For this reason, the robots must maintain a history of state posteriors and carry out a sort of *lag* filtering every time observations from other teammate robots arrive that correspond to previous time steps. To ensure the consistency of the communicated messages, a timestamp mechanism is used.

Since decision making must be based on the most recent information, the state posterior is continuously updated using (2), while action commands, observations, and state distributions, are stored in memory. If a message from another teammate

robot arrives at time t that corresponds to a lagged observation at time $t-\tau$, then (3) is applied to the state distribution at time $t-\tau$, and the filtered posterior is propagated through (2) and (3) repeatedly up to time t.

To simplify the implementation, in our system we have approximated the state posteriors with spherical Gaussian distributions, fully characterized by their means and variances. For each teammate robot i we maintain a history of state estimates (posterior means) $\{x_i^{t-T}, \ldots, x_i^{t-1}, x_i^t\}$ together with the corresponding variances, for a time horizon T. When an observation $\hat{y}_i$ arrives that is shifted back in time, we carry out the following steps:

1. We find the estimate $x_i^{t-\tau}$ with time-stamp $t-\tau$ equal (in practice closer) to the time-stamp of the observation $\hat{y}_i$.
2. We update the posterior estimate $x_i^{t-\tau}$ by applying (3) to get a fused estimate $\hat{x_i}^{t-\tau}$.
3. We compute the transformation matrix A that maps $x_i^{t-\tau}$ to $\hat{x_i}^{t-\tau}$.
4. We adjust all subsequent estimates $\{x_i^{t-\tau+1}, \ldots, x_i^t\}$ by applying the operator A to each of them.

The above discussion applies to the problem of maintaining a distributed shared world model that allows prediction of the state of each teammate robot. Similar considerations apply to the prediction of the other moving objects, i.e., opponent robots and the ball, whereas for the former, the additional problem of assignment (which robot corresponds to which observation) has to be addressed. Due to lack of space we defer detailed discussion of these issues herein.

3 Team skills

The main benefit of a distributed dynamic world model is that team coordination is greatly simplified. Sharing the same world model enables team robots to reason not only about their own actions but they can also predict the next action a teammate robot will take. Communication on the team coordination level is used to improve robustness and is almost a necessity since the team is heterogeneous both in hardware and software. The Utrecht Pioneer robots use a subsumption architecture while the Amsterdam/Delft Nomad Scouts operate on a hybrid architecture. We also hope to make our system more robust by having a backup mode when the world module cannot provide reliable data.

3.1 Team strategy

Our team uses a global team behavior (a strategy) from which each robot derives its individual behavior (also known as role). The team skills module determines the current team behavior using a simple finite state machine which takes as input the question "Who has the ball?". We have defined three team strategies which are shown in figure 1 together with the transitions between them.

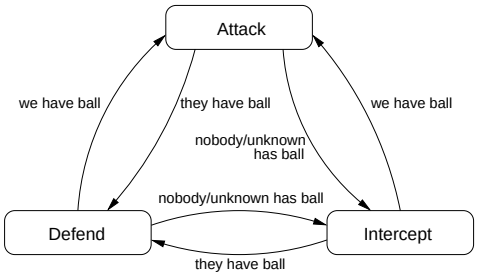

Figure 1: Finite state machine to determine the next team behavior.

3.2 Distributing individual behaviors

There are about half a dozen different individual behaviors available, all of which have an attacking, defending, or intercepting purpose. Utility functions based on potential fields [6] are used to distribute the roles among the team members. This technique is similar to the ones used by other participants in the middle size league [2, 10]. Each robot calculates its own scores for the individual behaviors useful at the moment, communicates them and then calculates the scores of all its teammates. Incoming scores from one of your teammates overwrite the scores you calculated for him. This introduces a level of redundancy which improves the team performance. When everybody knows the scores for the individual behaviors of every team member the roles can be easily distributed.

3.3 Choosing the next action

The exact specification of an individual behavior is not (and can not be) the same on both architectures. The following describes the Amsterdam/Delft approach which is similar to the one Tambe [8] described for use in the simulation league. A Markov decision process is used to search through the state space of a robot's future states. First we generate a number of possible actions and their parameters with respect to the current situation and individual behavior. For each of these actions the probability of success is calculated. The impact on the world of an action and the expected concurrent actions of teammates and enemy players is evaluated. Modeling teammate actions however is relatively easily compared to modeling your opponent with reasonable accuracy. Now the action with the maximum expected utility can be chosen and the process can be repeated to look further ahead. One of the challenges of this approach is calculating the probability a certain action will succeed, but after an estimation by hand a technique like reinforcement learning can be used to adjust them.

3.4 Evaluating the world

Another key problem of the method described above is the choice of evaluation criteria for rating the future world an action produces. We have chosen four criteria

based on real-life soccer experience. Each of them has its own weight associated with it, since some things are more important in soccer then others.

First criterion is a check whether the ball is in one of the goals. This criterion obviously has the highest weight of all, scoring a goal is the goal of the game and avoiding a goal for your opponents is also very important.

Next criterion is the concept of ball possession. This score is at its maximum when our team has the ball and at its minimum when one of the opponents has it. This information can be retrieved from the world model. When both teams do not possess the ball, the score depends on the ratio of the distance of our player closest to the ball and the distance of their closest player.

Third criterion is a role evaluation. For this criterion the potential fields from the role distribution process are used. Each individual behavior has its own attractors and repellers. A defender for instance likes to be in the area just in front of his own goal area.

The last criterion concerns obstacle avoidance and strategic positioning. This criterion also uses potential fields, but this time only static obstacles, moving objects and your own heading are being considered. Static obstacles can be modeled as repellers, these are the corners, walls and both goal areas[1]. The ball is the only attracting object in the game, and its force is amplified more than any of the other forces since the game revolves around it. Your own heading also influences the potential field, looking in the direction of the ball is a good thing. If the position of the ball is unknown however looking towards the enemy goal is rewarded.

3.5 Improving robustness

In a real world domain like the RoboCup middle-size league one has to take care to design a robust system. Common problems are temporary communication failures or vision based self-localization which is not able to provide position updates for some time. This is the reason our team skills module has a relative mode next to the absolute mode described above. In this relative mode no actions are generated which involve absolute coordinates (like "dribble to position x,y" or "shoot at angle α"). Instead more reactive actions of the underlying player skills module are being considered (like "chase ball" or "seek enemy goal"). The system goes into relative mode when it notices the error estimate of its own position has risen above a certain threshold or when the vision system notices its own position cannot be correct. The team skills module switches back to absolute mode when the vision system has reclaimed its position.

When a robot initiates a "chase ball" action it broadcasts to all team members an estimate of time it thinks it needs to reach the ball. When it wants to chase the ball itself it checks whether or not it will be the first to arrive there. If it is not going to be first and the other robot who is chasing the ball is only a few seconds away the first robot chooses another action. Using this mechanism we want to prevent two robots trying to chase the ball at the same time, even in relative mode.

[1] Current RoboCup regulations forbid a field player to stay more than 5 seconds in its own goal area and more than 10 seconds in the opponent's goal area.

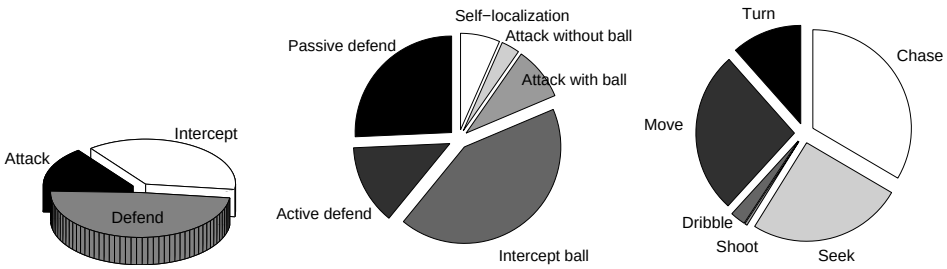

Figure 2: Statistics on the team behavior for a winning match.

4 Discussion and Results

We presented a shared world model which is locally maintained in each robot. Using time stamping, we can fuse observations measured at different times with unknown latencies in the network. Taking into account the uncertainty present, the time-stamped measurements of the whole team are fused in each robot and the position and velocity of moving objects are calculated. The system is able to optimally integrate the observations of the whole team, and to get an estimate of the position and velocity of objects when they are only perceived by other robots.

Our team skills module is able to dynamically distribute roles amongst the team members and choose actions based on these roles. It can handle situations where not all four players are active or when their positions are not known. The system is robust enough to deal with several kinds of failures as we experienced in RoboCup middle-size league games and it is able to recover after they have been resolved.

We used our world model and team skills in the Dutch RoboSoccer team at the RoboCup 2001 world championship in Seattle, USA. Clockwork Orange made it to the quarterfinals to be defeated by the world champion CS Freiburg. In total seven games were played, resulting in three victories, one draw and three losses. In Fig. 2 we show some basic statistics on the team skills module, obtained in a match against Artisti Veneti. The match resulted in a 3–0 victory which was our ticket for the quarterfinals. The results are the average of the three field players, gathered over 48 minutes.

The team behavior chart shows the robots are only small portion of the time occupied with attacking. In this team behavior one robot gets assigned to him the "attack with ball" role, which should result in a dribble towards the enemy goal followed by a shot. Passing the ball towards the auxiliary attacker (the one in "attack without ball") has not yet been implemented since the mechanical and lower level software make-up don't permit it yet.

A lot of time is spent chasing the ball: it is not only the action to obtain a ball lying in the open but also used to try to steal the ball away from an opponent. When the position of the ball is not known (in this game about one third of the time) every field player starts seeking it, which can be clearly observed in play when they all are turning circles. Even when not in "Defend" team behavior

there is still one field player in the role "Passive defend". He usually is moving in front of his own goal waiting for an enemy attack. The individual behavior "Self-localization" is chosen when the vision system or world model have lost track of our position and the robot has nothing urgent to do. This role facilitates the vision self-localization by turning to one of the goals and subsequently move towards it. This should help the vision system by encountering enough lines to reclaim our position estimate.

References

[1] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 5, Oct. 1996.

[2] C. Castelpietra, L. Iocchi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players. In *Proceedings of the 4th International Workshop on RoboCup*, pages 149–158, Melbourne, Australia, Aug. 2000.

[3] F. de Jong, J. Caarls, R. Bartelds, and P. P. Jonker. A two-tiered approach to self-localization. In *Proc. RoboCup 2001 Int. Symposium*, Seattle, USA, Aug. 2001.

[4] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000.

[6] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[7] S. I. Roumeliotis. *Robust Mobile Robot Localization: From Single-Robot Uncertainties to Multi-Robot Interdependencies*. PhD thesis, University of Southern California, May 2000.

[8] M. Tambe and W. Zhang. Towards flexible teamwork in persistent teams: Extended report. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(2):159–183, 2000.

[9] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[10] T. Weigel, W. Auerbach, M. Dietl, B. Dümler, J. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski, and M. Thiel. CS Freiburg: Doing the right thing in a group. To appear in P. Stone, G. Kraetzschmar, T. Balch, RoboCup 2000, Springer-Verlag, 2001.