

Interactive visualization, information sharing, planning and learning for a team of robots

M.T.J. Spaan[§] M. Koutek* B. Terwijn[§] J.R. Kok[§] H. E. Bal*
 M. Boasson[§] F.C.A. Groen[§] N. Vlassis[§]

[§]Informatics Institute, University of Amsterdam
 Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

*Faculty of Sciences, Vrije Universiteit
 De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

Abstract— In this paper we are interested in intelligent real-world multiagent systems that try to cooperatively solve a task. A multiagent (or multi-robot) system is a collection of agents that coexist in an environment and interact with each other. In our case, we are interested in fully cooperative multi-robot systems in which all robots share a common goal.

We report in this work on research on the following topics: First, we developed a shared world model which estimates the global positions of objects in order to reduce the uncertainty in the environment. It allows us to fuse observations made by different agents and improve position estimates of each agent. Second, we extended our visualization of a robot soccer game by incorporating sensor data generated by a set of virtual sensors and by enabling remote human interaction. Third, we developed a new POMDP algorithm for agent planning in uncertain environments, in which the agent only receives partial information (through its sensors) regarding the true state of environment. Fourth, we have proposed a multiagent Q-learning technique, that allows a group of agents to learn how to jointly solve a task given the global coordination requirements of the system.

I. INTRODUCTION

In the future we expect to see intelligent multiagent systems being deployed in real-world situations, for instance, rescuing in disaster scenarios. The societal and economical benefits of building such systems are huge, while at the same time there are important research questions to be answered. Our project involves the study of such ‘intelligent’ real-world multiagent systems in which a team of robotic agents carry out a joint task. A central topic in our research is how the agents can build and maintain shared models of the environment. We have been studying the problem of robot localization, that is, how a robot can find its position in the environment under conditions of uncertainty in its motion and sensor measurements. We are using particle filters for this problem. We have also been studying the

problem of multiagent coordination in the case of a large number of agents. Here we use the framework of coordination graphs. We are also interested in planning under uncertainty, and here we are investigating solution techniques for partially observable Markov decision processes.

In parallel, we are developing software that allows for real-time visualization of the behavior of a multiagent system. We use robot soccer as testbed. The visualization takes place on the ICWall (at the Vrije Universiteit Amsterdam), a scalable tiled display that can be used for 3D (stereo) graphics. We have implemented software libraries in Splice for connecting the robots at the University of Amsterdam with the ICWall. Splice is an efficient publish-subscribe model of computation where production and consumption of data are decoupled. The visualization allows a human to interact with the video wall and affect the behavior of the robots in real-time. It displays the shared model of the environment as perceived by a robot.

The setup of the paper is as follows. In section II we describe our visualization and interaction environment, together with the shared world model which it visualizes. Section III reports on our algorithm for planning in uncertain environments, and focuses on planning for robots. In section IV we extend our previous work on multiagent coordination by letting the agents learn how to solve a joint task, given the global coordination requirements which are captured in a coordination graph. Finally, section V presents some conclusions and future work.

II. VISUALIZATION ON THE ICWALL

We use robot soccer as a useful case study for research on real-time visualization of multiagent systems and human interaction using virtual environments. Although each of the robots is autonomous, we need a centralized (possibly remote) visualization and monitoring system allowing a human observer to interfere and control the robots



Fig. 1. RoboCup visualization on the ICWall.

during the game. This is especially useful while developing new algorithms for the robots, their evaluation, and even during the real deployment of the robots. The ICWall tiled display has been chosen as the remote visualization site (at Vrije Universiteit Amsterdam); the robots and the soccer field are located at the Informatics Institute of the University of Amsterdam. The ICWall has been implemented using off-the-shelf components, which reduces the costs substantially. It provides functionality of a so called ‘tiled display’, see Fig. 1. The display is scalable and offers high screen resolution (the sum of the resolutions of each tile).

The large format allows a wide variety of different applications. An advantage is the large field of view and the possibility for simultaneous projection of different types of information. The ICWall is successfully used for education and research purposes, for which the room should be as multifunctional as possible. In the current design, several components are present, such as the large projection screen, a plasma panel with touch screen (Smartboard), and a wireless network. The lecturer uses the touch screen as interface for control of and interaction with the tiled display. The room is capable of seating around 40 people in front of a projection screen of roughly 5.25 by 2.0 meters. The high resolution display allows detailed scientific visualization, and overcomes the limited screen resolution of standard monitors. The large field of view and stereo projection significantly improve immersion of the user and the audience into the displayed virtual environment.

A. System overview

We have developed a distributed visualization and control environment for autonomous robots, capable of playing soccer, on top of the Clockwork Orange robot soccer software [3], interfaced through the Splice communication

layer [1], and the Aura scene-graph library [13].

The visualization client of our robot soccer software is part of the shared data space which is provided by the Splice system. Based on the sensor data our system builds a shared model of the world with the robots. The visualization client is intended to run on ICWall, driven by a cluster of 8 rendering PCs and one server. It is implemented on top of the Aura API, which is a portable 3D graphics and scene graph library for Virtual Reality Applications [13]. A parallel implementation of Aura has been developed to run on the cluster which drives the tiled display. The sequential Aura runs on desktop PCs (Linux or Windows) or SGI’s (IRIX), and needs OpenGL. It is therefore possible to use the visualization client also on the desktop systems.

Visualization of a robot soccer game on the tiled display (as shown in Fig. 1) gives the human observer an opportunity to be present in a sort of ‘virtual command center’ and to remotely monitor the state of the each robot, the team, and the whole game. The large screen and wide field of view offer enough space for visualization and display of additional data, such as live camera images from the soccer field for correlation with the virtual state of the game, and many other data that are made available in the shared data space via Splice. At the current stage we are mainly focused on visualization and ‘tuning’ of robot localization. However we also see benefits in the use of the visualization for the monitoring of the strategic and tactical information of the robots and the team. An important research topic is the problem of latency which influences the human feedback to the robots, hence we plan to study this problem as future work. From the Virtual Reality point of view we consider to develop and employ intuitive interaction techniques and interfaces. We also plan to incorporate into our system some higher-level commands of the human observer (coach) for improvement of tactics and team behavior. The virtual environment provided by the ICWall seems to have the right potential for this kind of applications. To simplify the design of concurrent software modules we use Splice [1] (product of Thales Nederland B.V.) which implements a virtual shared data space in a network of processes each running possibly on remote machines. Splice uses publish-subscribe protocol in which all data that is produced by a process is immediately communicated to all processes that have once consumed it. Although this increases the amount of traffic on the network, it reduces the time for the consumer processes to access the data when they actually need it, offering significant speedups when real-time behavior is in order.

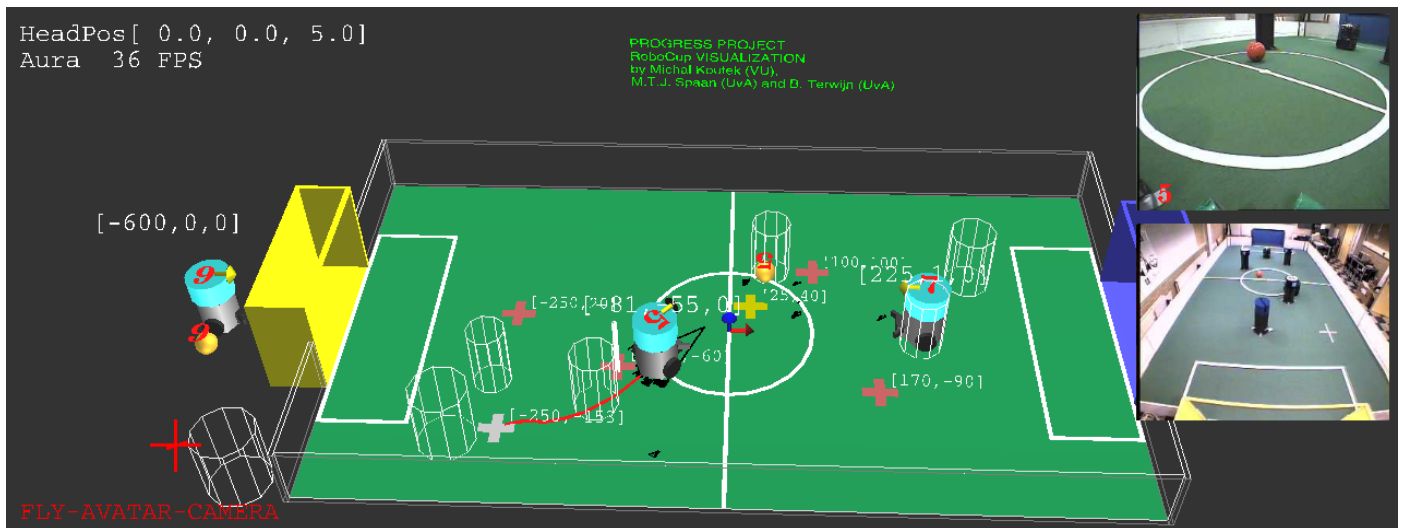


Fig. 2. Visualization of the shared world model. On display is the shared world model of robot #5 consisting of black obstacles (the white cylinders) and the ball. To the right its camera image (top) and the video stream from the overview camera (bottom) are shown.

B. Building a shared world model

We developed a Shared World Model (SWM) estimating the global positions of objects in order to reduce the uncertainty in the environment. This allows us to fuse observations made by different agents and estimate new attributes such as the speed of objects for extrapolation. This requires each robot to accurately localize itself, i.e., to find its most likely position given a sequence of robot displacements and observations. In our case the observations consist of camera images which are compared to a model of the environment. We have taken an iterative approach in which after each displacement and observation the probability distribution representing the uncertainty over the current position of the robot is updated. This probability distribution is represented by a set of candidate positions of the robot which are called particles [12], [10].

The SWM is realized by first building a local world model on each agent which tracks the position of objects relative to the agent. Second, the local world model is sent to all agents, including itself, together with the agent's multiple position hypotheses: its particles. Each agent then updates its SWM by projecting the local world models it receives on the global position of the agent which supplied it. For this it needs to select the most likely position hypothesis of the other agent and the certainty associated with this hypothesis. When no acceptable match is found the local world model is discarded.

Additionally, the SWM is used to improve the position estimate of each agent. This is realized by building on each agent an additional SWM composed only of local world models received from other agents. By matching the local

world model of the agent with this SWM, new position candidates can be deduced. These position candidates are used, together with the position candidates derived from the static objects (the lines in the environments and the goals), to improve the position estimate.

Since each agent computes the SWM by itself there is no central control which makes the system more robust in case of failure. As a disadvantage the SWM on each agent can differ due to different order and time of arrival of the local world models. We do not expect problems because of these differences but this will be investigated in experiments together with the correctness of the SWM and the improvement in the position estimation.

C. Visualizing the shared world model

We have extended our visualization and interaction environment to incorporate the shared world model of each robot, see Fig. 2. The user can select the robot he is interested in. The visualization will show the objects in this robot's world model alongside the camera images and localization uncertainty of a robot. The visualization environment allows for human interaction and even direct control of the robot. The interactive robot 3D visualization gives a nice overview and provides a great experimental environment. It has provided new insights into the behavior and limitations of the current Clockwork Orange software base [7].

III. ROBOTIC PLANNING IN UNCERTAIN ENVIRONMENTS

As autonomous robots are being applied in more and more complex domains the need grows for tractable ways

of planning under uncertainty. In order for a robot to execute its task well in a real-world scenario it has to deal properly with different types of uncertainty: a robot is unsure about the exact consequence of executing a certain action and its sensor observations are noisy. Robotic planning becomes even harder when different parts of the environment appear similar to the sensor system of the robot. In these partially observable domains a robot needs to explicitly reason with uncertainty in order to successfully carry out a given task.

Partially observable Markov decision processes (POMDPs) provide a rich mathematical framework for solving such planning problems [4]. The POMDP defines a sensor model specifying the probability of observing a particular sensor reading in a specific state, and a stochastic transition model which captures the uncertain outcome of executing an action. In many situations a single sensor reading does not provide enough evidence to determine the complete and true state of the system. The POMDP framework allows for successfully handling such situations by defining and operating on the *belief state* of a robot. A belief is a probability distribution over all states and summarizes all information regarding the past. Solving a POMDP now means computing a policy—i.e., a mapping from belief states to actions—that maximizes the average collected reward of the robot in the task at hand. Such a policy prescribes for every belief state the action that maximizes the expected reward a robot can obtain in the future.

Unfortunately, solving a POMDP in an exact fashion is an intractable problem. Intuitively speaking, looking one time step deeper into the future requires considering each possible action and each possible observation. A recent line of research on approximate POMDP algorithms involves the use of a sampled set of *belief points* on which planning is performed (see e.g. [8], [9]). The idea is that instead of planning over the complete belief space of the robot (which is intractable for large state spaces), planning is carried out only on a limited set of prototype beliefs that have been sampled by letting the robot interact with the environment. We have developed along this line a simple randomized approximate algorithm which exhibits excellent performance, being orders of magnitude faster than state-of-the-art methods in many benchmark problems in the literature [11], [14]. We will demonstrate its application on an office delivery task involving a mobile robot with omnidirectional vision in a highly perceptually aliased office environment, where the number of possible robot locations is in the order of hundreds.

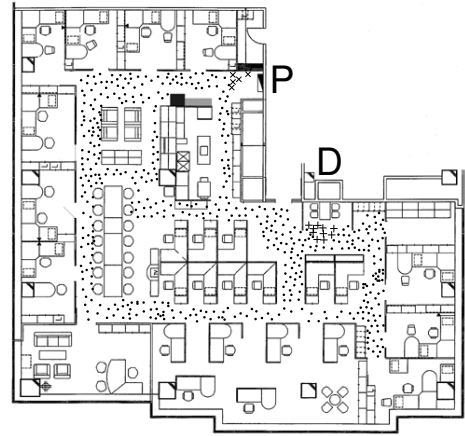


Fig. 3. The office environment. The markers $\{\cdot, +, \times\}$ denote the grid positions of the problem. The positions \times (close to P) are the pickup locations, the ones marked by $+$ (below D) are the delivery (goal) positions.

A. Experimental results

We applied our algorithm on a problem which is based on data obtained in a realistic setting: a delivery task in an office environment with 1000 states. The task is to pick up mail at the entrance of the office (P , see Fig. 3) and deliver it to a certain room (D). At the start of the task the robot is uncertain about its location, and whether it has picked up the mail. The observation model is based on panoramic images taken by an omnidirectional camera mounted on the robot¹. We clustered these images to create a discrete set of 10 observations and subsequently computed an observation model from them. Fig. 4 shows some example observations with their observation model. The robot can execute four basic motion commands $\{north, east, south, west\}$ which transports it according to a Gaussian distribution centered on the expected resulting position. In order to accomplish its task the robot must first execute the *pickup* action in one of the five pickup states near the entrance of the office (marked by P in Fig. 3). Only in one of these states the action results in flipping the pickup bit, after which delivering the mail has become possible. To complete the task and receive positive reward the robot has to execute the *delivery* in one of the ten delivery states (indicated by D in Fig. 3).

Our algorithm can successfully solve the resulting POMDP model. To visualize the policies it computes, we plotted some example trajectories in Fig. 5. They show the computed policy directs the robot to first move to the pickup states, pick up the mail, and then move to the delivery locations in order to deliver the mail. Running the

¹We used the MEMORABLE database, provided by the Tsukuba Research Center in Japan, for the Real World Computing project.

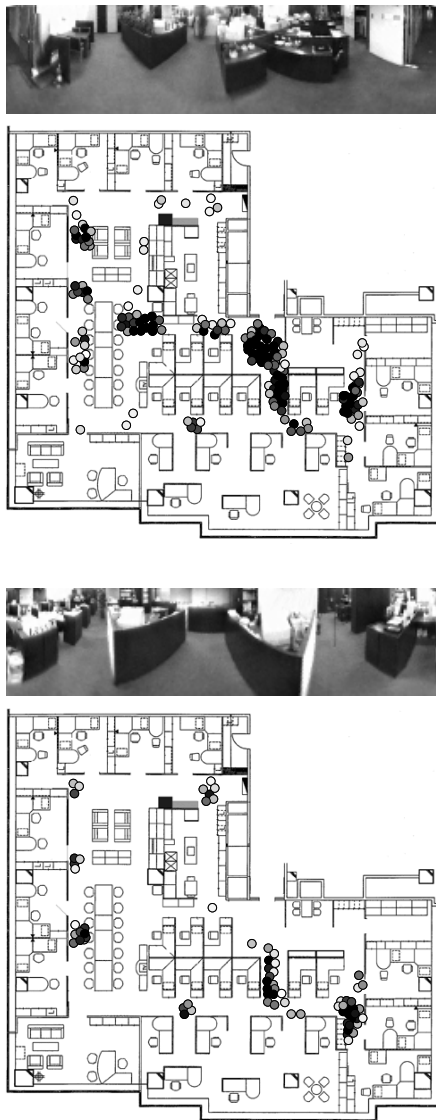


Fig. 4. Panoramic images corresponding to two possible sensor observations, and below each one its induced sensor model. The darker the dot, the higher the probability.

algorithm on some benchmark problems from the literature indicated very competitive performance to similar algorithms: in most cases it is at least one order of magnitude faster than state-of-the-art algorithms, and equally good or better in terms of the quality of the computed policy.

IV. LEARNING TO COORDINATE A MULTI-ROBOT TEAM

In this project we are interested in intelligent real-world multiagent systems that try to cooperatively solve a task. A multiagent (or multi-robot) system is a collection of agents that coexist in an environment and interact with each other. In our case, we are interested in fully cooperative multi-robot systems in which all robots share a common goal. Previously, we have shown how to coordi-

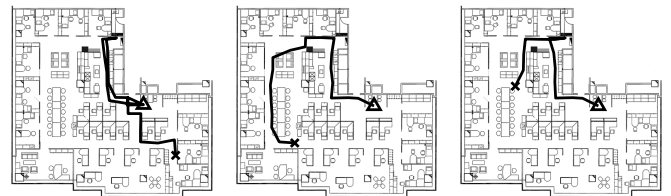


Fig. 5. Some example trajectories in the TRC environment. Start positions are marked with \times and the last state of each trajectory is denoted by a \triangle .

nate the actions of a multi-robot team by assigning roles to the robots and applying a coordination graph to the problem [10], [5]. Roles are a natural way of introducing domain prior knowledge to a multiagent problem and provide a flexible solution to the problem of distributing the global task of a team among its members. In the soccer domain for instance one can easily identify several roles ranging from ‘active’ or ‘passive’ depending on whether an agent is in control of the ball or not, to more specialized ones like ‘striker’, ‘defender’, ‘goalkeeper’, etc. Such an assignment of roles provides a natural way to parametrize a coordination structure over a continuous domain. The intuition is that, instead of directly coordinating the agents in a particular situation, we assign roles to the agents based on this situation and subsequently try to ‘coordinate’ the set of roles. Recently we extended this work by allowing the agents to *learn* the value of the different coordination rules [6]. We have demonstrated how Q-learning, a well known reinforcement learning technique, can be efficiently applied to such multiagent coordination problems. In many problems agents only have to coordinate with a subset of the agents when in a certain state (e.g., two cleaning robots cleaning the same room). We have proposed a multiagent Q-learning technique, *Sparse Cooperative Q-learning*, that allows a group of agents to learn how to jointly solve a task given the global coordination requirements of the system.

A. Sparse Cooperative Q-Learning

In [6] we first examine a compact representation of the state-action space in which the agents learn Q-values based on full joint actions in a predefined set of states. In all other (uncoordinated) states, the agents learn based on their individual action. Then we generalize this approach using a context-specific coordination graph (CG) [2]. In a CG each node represents an agent, while an edge defines a dependency between two agents. Only interconnected agents have to coordinate their actions. The global coordination problem is now decomposed into a number of local prob-

lems that involve fewer agents. In order to compute the optimal joint action (with maximum total payoff), a variable elimination algorithm can be used.

In a CG, value rules can be used to specify the dependencies between the agents. These rules define a (local) payoff for a subset of all state and action variables. In our method, the global Q-value for a state equals the sum of the payoffs of all applicable value rules. After every state transition, the payoff of every applicable rule is updated based on a Q-learning rule that adds the contribution of all involved agents. Effectively, each agent learns to coordinate only with its neighbors in a dynamically changing CG. This allows for a sparse representation of the joint state-action space of the agents, resulting in large computational savings.

B. Results

We demonstrate the proposed technique on the ‘predator-prey’ domain in which two predators have to coordinate to capture a single prey in a 10×10 world.

As is seen in Fig. 6, both the IL approach and our proposed method learn quickly in the beginning with respect to the MDP learners since learning is based on fewer state-action pairs. However, the IL approach does not converge to a single policy since the agents do not model the action of the other agent in the coordinated states. These dependencies are explicitly taken into account for the other two methods. For the MDP learners, they are modeled in every state which results in a slowly decreasing learning curve. For the context-specific approach they are considered only for the coordinated states, resulting in a quicker decreasing learning curve with comparable performance to the optimal policy. Our method thus achieves a good trade-off between speed and solution quality.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have reported on research on several aspects of cooperative real-world multiagent systems. We presented our visualization environment of a robot soccer game, a useful case study for research on real-time visualization of multiagent systems and human interaction using virtual environments [7]. The visualization incorporates the shared world model of the robot, which fuses observations from different robots and improves their position estimates. We described our algorithm for planning in environment in which a robot is unsure about the exact consequence of executing a certain action and in which its sensor observations are noisy [11]. We extended our research on multiagent coordination by allowing the agents to learn the value of coordination rules [6].

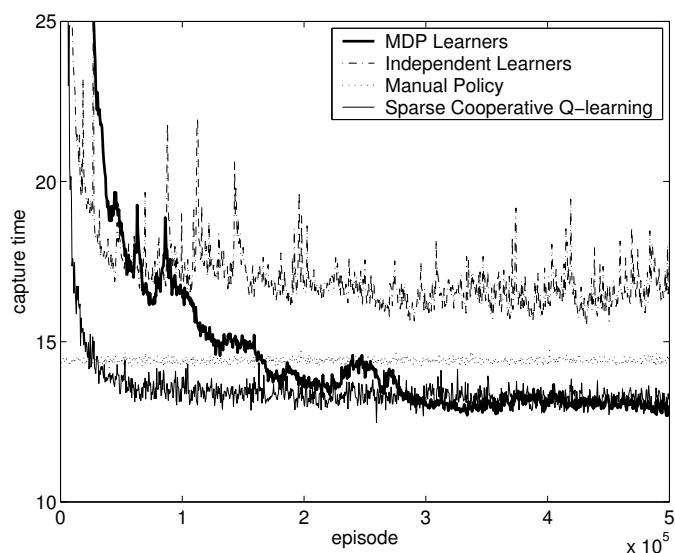


Fig. 6. Capture times during the first 500,000 episodes (averaged over 10 runs).

At the moment we are working on the following directions: First, we study anytime algorithms for multiagent action selection in coordination graphs, in particular algorithms that involve distributed message-passing techniques. Second, we conduct experiments to determine the usefulness of humans interacting with the team of robots through the visualization in improving the accuracy of the shared world model.

Acknowledgments

This work is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

REFERENCES

- [1] M. Boasson. Control systems software. *IEEE Trans. Automatic Control*, 38(7):1094–1107, 1993.
- [2] C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored MDPs. In *Proc. 8th Nation. Conf. on Artificial Intelligence*, Edmonton, Canada, July 2002.
- [3] Pieter Jonker, Bas Terwijn, and Bram van Driel. The Clockwork Orange team 2004. In *Proceedings CD RoboCup 2004*. Springer-Verlag, Lisbon, Portugal, 2004.
- [4] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
- [5] Jelle R. Kok, Matthijs T. J. Spaan, and Nikos Vlassis. Noncommunicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 2004. In press.
- [6] Jelle R. Kok and Nikos Vlassis. Sparse cooperative Q-learning. In Russ Greiner and Dale Schuurmans, editors, *Proc. of the 21st*

- Int. Conf. on Machine Learning*, pages 481–488, Banff, Canada, July 2004. ACM.
- [7] M. Koutek, M. T. J. Spaan, and B. Terwijn. Construction and 3d visualization of the shared world model in the robosoccer context. Technical Report NIFEWVU20082004, Faculty of Sciences, Vrije Universiteit, 2004. In preparation.
- [8] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, August 2003.
- [9] N. Roy and G. Gordon. Exponential family PCA for belief compression in POMDPs. In *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.
- [10] M. T. J. Spaan, M. Koutek, B. Terwijn, J. R. Kok, H. E. Bal, M. Boasson, F. C. A. Groen, and N. Vlassis. Coordination, data sharing, and remote visualization of a team of autonomous robots. In *Proc. 4th PROGRESS Workshop on Embedded Systems*, Nieuwegein, The Netherlands, 2003.
- [11] Matthijs T. J. Spaan and Nikos Vlassis. A point-based POMDP algorithm for robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2399–2404, 2004.
- [12] B. Terwijn, J.M. Porta, and B.J.A. Kröse. A particle filter to estimate non-markovian states. In F.C.A. Groen, editor, *International Conference on Intelligent Autonomous Systems, IAS'04*, pages 1062–1069. IOS Press, March 2004. ISBN 1-58603-414-6.
- [13] T. van der Schaaf, L. Renambot, D. Germans, H. Spoelder, and H. Bal. Retained mode parallel rendering for scalable tiled displays. In *Proc. 6th Ann. Immersive Projection Technology (IPT) Symposium*, Orlando, Florida, 2002.
- [14] Nikos Vlassis and Matthijs T. J. Spaan. A fast point-based algorithm for POMDPs. In *Benelearn 2004: Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands*, pages 170–176, Brussels, Belgium, January 2004. (Also presented at the NIPS 16 workshop ‘Planning for the Real-World’, Whistler, Canada, Dec 2003).