Fundamenta Informaticae 75 (2007) 281–294
IOS Press

Cooperative World Modeling in Dynamic Multi-Robot Environments

Daniel Göhring* **and Hans-Dieter Burkhard**
Institut für Informatik
Humboldt-Universität
Unter den Linden 6, 10099 , Berlin, Germany
goehring@informatik.hu-berlin.de
hdb@informatik.hu-berlin.de

Abstract. In this paper we describe how a group of agents can commonly estimate the position of objects. Furthermore we will show how these modeled object positions can be used for an improved self localization. Modeling of moving objects is commonly done by a single agent and in a robocentric coordinate frame because this information is sufficient for most low level robot control and it is independent of the quality of the current robot localization. Especially when many robots cooperate with each other in a partially observable environment they have to share and to communicate information. For multiple robots to cooperate and share information, though, they need to agree on a global, allocentric frame of reference. But when transforming the egocentric object model into a global one, it inherits the localization error of the robot in addition to the error associated with the egocentric model.

We propose using the relation of objects detected in camera images to other objects in the same camera image as a basis for estimating the position of the object in a global coordinate system. The spacial relation of objects with respect to stationary objects (e.g., landmarks) offers several advantages: The information is independent of robot localization and odometry and it can easily be communicated. We present experimental evidence that shows how two robots are able to infer the position of an object within a global frame of reference, even though they are not localized themselves. We will also show, how to use this object information for self localization. A third aspect of this work will cope with the communication delay, therefore we will show how the Hidden Markov Model can be extended for distributed object tracking.

Keywords: Markov Localization, Multi-Agent Systems

*Address for correspondence: LFG Künstliche Intelligenz, Institut für Informatik, HU-Berlin, Unter den Linden 6, 10099 Berlin, Germany

1. Introduction

As all of our work is related to concurrent processes of perception and communication, we believe it to be a venerable contribution in memory of Zdzislaw Pawlak. We show how combination of individual knowledge can be used to improve the knowledge for the case of robots.

Creating a common world model is crucial in multi robot environments especially when sensor capacity is limited and when cooperation between robots comes into play. Usually a robot is surrounded by many objects with static or dynamic behavior. In the first case a key task of every multi robot system is to localize the agents relative to the given environment which is useful for path planning or efficient positioning. In case of dynamic objects very often the agent wants to know and to predict the state of the object, most commonly the position, speed, orientation, etc. The task of estimating an objects position as well as self localizing is made more difficult, whenever the robot has only a limited field of view like when using normal camera. We will use the scenario of soccer playing robots (RoboCup) [8] for our discussions.

In hybrid architectures [2] basic behaviors are often directly coupled to sensor data, e.g., the ball percept when following the ball. When sensor capacity is limited, it is important to maintain a model about the world. The world modeling task can be split into subtasks, one task, to estimate the robot's position is called *self localization*, another task, to estimate the position of an object of the environment is called *tracking*. Bayesian filters, especially Monte Carlo Localization approaches, have become very popular for robot localization [16] and tracking and whenever sensor data is imprecise and/or limited. In [9] the knowledge of the robot about the world state - usually referred to as *belief* - subsumes its knowledge about object positions and its knowledge about self localization in a Bayesian network [13][6]. The advantages of such a model are obvious. By having a combined model for self localization and object tracking the observer can to some extend infer from object positions to self localization information and vice versa. But due to the curse of dimensionality the approach of putting everything in one model is usually not tractable under realtime conditions. The state space becomes high-dimensional very quickly.

Usually for efficiency reasons the modeling task is split up into the object modeling and the self localization task. Now to model an object within its environment, self localization information has to be passed to the object modeling process. But when self localization is uncertain or jumpy, the object state becomes jumpy as well. To overcome this limitation of self localization accuracy, object modeling approaches as in [4] tend to use an egocentric model for tracking a moving object (a ball) and an allocentric model for self localization. Because egocentric models represent the object state from an agents point of view, they are often called local model. In contrast to local models, allocentric models do not use the perspective of certain agents, therefore being called global models.

Splitting up the state space into local and global models works quite nice for single agent systems, but fails when communication with other agents is necessary. The locally represented information has then to be transformed into global coordinates, to be communicated to the other agent and to be transformed again into local coordinates of the second agent. It is obvious that small errors in the self localization of one or both robots can have significant impacts on the usability of such communicated information.

Therefore, in this article we want to present a solution to the multi agent object tracking problem by modeling objects in global coordinates from the start, while trying to stay independent from self localization information and keeping the state space small. To achieve this we want to examine the sensing process more closely. In feature based belief modeling, features are extracted from the raw sen-

sor data. We call such features *percepts* and they correspond directly to objects in the environment detectable in the camera images. There are different levels of percepts. Shapes in the robot image can be named percepts, but at this stage it is not clear to the robot, where the corresponding objects can be found in the robot's environment. Therefore we use further sensor data of the robot's neck and leg angles to calculate the object positions in coordinates relative to the robot. For a better understanding, such a percept could have the form of, e.g., *ball(100,200)*, referring to a ball within $100mm$ in front and $200mm$ to the left of the robot.

In a typical camera image of a RoboCup environment, the image processing could, for example, extract the following percepts: *ball*, *opponent player*, and *goal*. Different percepts are then passed to different modules, e.g., the ball modeler receives (only) the ball percept. Percepts are commonly considered to be independent of each other to simplify computation, even if they are used for the same purpose, such as localization [14]. Using the distance of features detected within a single camera image to improve Monte Carlo Localization was proposed by [7]: when two landmarks are detected simultaneously, the distance between them yields information about the robot's whereabouts.

When modeling objects in relative coordinates, using only the respective percept is often sufficient. However, information that could help localize the object within the environment is not utilized. That is, if the ball was detected in the image right next to a goal, this helpful information is not used to estimate its position in global coordinates. We call seeing different percept within one image a *percept relation*. We define a percept relation as a tuple of two objects, containing the distance and angle of the objects to each other. Thus, for a percept relation, the observer's position is not important anymore. Using percept relations yields several advantages:

Sensing errors As the object of interest and the reference object are detected in the same image, the sensing error caused by joint slackness or robot motion becomes irrelevant as only the relations of the objects within the camera image matter. Thus the only remaining errors are those caused by image processing.

Global localization The object can be localized directly within the environment, independent of the quality of current robot localization.

Communication Using object relations offers an efficient way of communicating sensing information, which can then be used by other robots to update their belief by sensor fusion.

1.1. Outline

We will show how relations between objects in camera images can be used for estimating the object's position within a given map and present experimental results. Our robots don't have to be localized for estimating the global position of an object which is in stark contrast to other approaches. Furthermore, we will show how communication percepts can accelerate the modeling process in a group of robots. Finally we will present how self localization can be improved by communication of percept relations between different agents.

2. Markov Localization

Markov Localization is an approach for estimating an object state. It uses Hidden Markov Models (denoted HMM) - a statistical model where the system state being modeled is assumed to be a Markov process with hidden/unknown parameters. Thereby the hidden parameters have to be inferred from the observable parameters. It got its name by using the Markov assumption, i.e., for a current state estimation only the last sensor data, input data and the last state estimation are necessary. Thus it is probabilisticly independent from earlier sensor and input data. As the real world state is not observable but only some output variables from the sensor data, it is called *Hidden* Markov Model. Fig. 1 shows main parts of HMMs used for state estimation.

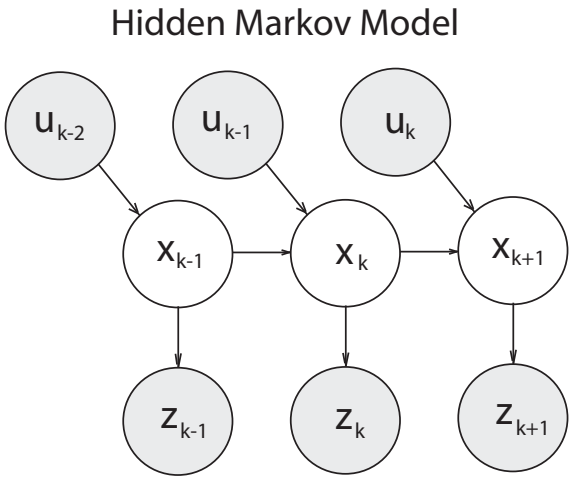

Figure 1. A Hidden Markov Model with input actions u and sensor data z. The object state x is hidden and must be estimated by u and z. It shows, that z_{k+1} depends on x_{k+1} and that x_{k+1} depends on u_k and x_k but not on any other variable.

Markov localization methods, in particular Monte-Carlo Localization (MCL), have proven their power in numerous robot navigation tasks, e.g., in office environments [5][3], in the museum tour guide Minerva [15], in the highly dynamic RoboCup environment [10], and outdoor applications in less structured environments [12]. MCL is widely used in RoboCup for object and self localization [14][11] because of its ability to model arbitrary distributions and its robustness towards noisy input data. It uses Bayes law and Markov assumption to estimate an object's position.

A Bayesian network is a form of probabilistic graphical model. It represents a set of variables together with a joint probability distribution with explicit independency assumptions. It is a directed, acyclic graph with nodes representing variables, arcs representing probabilistic dependencies between the variables and local probability distributions for each variable given values of its parents [1].

Terminology. We distinguish world model, world state, world belief, object model, object state and object belief. A world model contains all parameters, necessary for a description of a robot's environment. It also describes, how those parameters interact with each other and how they change over time. E.g., the position of the robot and the position and speed of the ball could subsume in a world model.

The model could also describe, that the speed of a rolling ball decreases over time. The world state is the vector, containing all state variables of the world, e.g., robot position. Unfortunately the robot cannot exactly know the world state, moreover it can only estimate the world state. The estimation of the world state is called world belief. It can contain fixed positions of objects of the environment or probability functions for representing uncertainty [16]. The definition for an object model are similar. An object model describes all state variables of an object and how they change over time or interact with each other, e.g., a velocity variable usually changes a position variable over time. The object state contains all necessary state variables of the object in the robot's environment. Again the robot can only estimate the real parameters, its estimation is represented by the belief function.

We will describe, how we implemented a model for a group of robots localizing ball. The probability distribution is represented by a set of samples, called particle set. Each particle represents a position hypothesis, i.e., a vector containing the coordinates of a possible ball position. The current belief about the object's position is represented by the particle density, i.e., by knowing the particle distribution the robot has knowledge about the most probable ball position. Thereby the belief function $Bel(s_t)$ describes the probability for the object (e.g. ball) state s_t at a given time t. Originally it depends on all sensor inputs[1] $z_1, .., z_t$ and all robot actions $u_1, .., u_t$ as the next equation shows:

$$Bel(s_t) = p(s_t|z_t, u_{t-1}, z_{t-1}, u_{t-2}, ..., z_1, u_0, z_0) \tag{1}$$

Bayes filters use the Markov assumption, which states that current measurements z_t and executed actions u_t are independent from earlier measurements or actions when object state s_t was given, as fig. 7 shows. With Bayes' law:

$$p(s|Z) = \frac{p(Z|s)P(s)}{p(Z)} \tag{2}$$

where s is the object state and Z is the set of measurements, equation (8) can be written as:

$$Bel(s_t) = \frac{p(z_t|s_t, u_{t-1}, ..., u_0, z_0)p(s_t|u_{t-1}, ..., u_0, z_0)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \tag{3}$$

We can further simplify this equation by using the Markov assumption, which makes the current sensor data z_t conditionally independent from earlier measurements z and actions u at a given object state s_t to:

$$Bel(s_t) = \frac{p(z_t|s_t)p(s_t|u_{t-1}, ..., u_0, z_0)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \tag{4}$$

By using the law of total probabilities the upper part of the fraction can be changed to:

$$Bel(s_t) = \frac{p(z_t|s_t)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \int p(s_t|s_{t-1}, u_{t-1}, ..., z_0)p(s_{t-1}|u_{t-2}, ..., z_0)ds_{t-1} \tag{5}$$

With the Markov assumption we get:

$$Bel(s_t) = \frac{p(z_t|s_t)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \int p(s_t|s_{t-1}, s_{u-1})Bel(s_{t-1})ds_{t-1} \tag{6}$$

[1] A sensor input z_i contains a measurement vector, e.g., the ball coordinates of a seen ball.

We use a normalizing constant η making sure that all probabilities sum up to 1 :

$$Bel(s_t) = \eta p(z_t|s_t) \int p(s_t|s_{t-1}, s_{u-1}) Bel(s_{t-1}) ds_{t-1} \tag{7}$$

Equation (7) is a recursive form of the Bayes filter, which enables us to recursively estimate an object's position. The equation can be split up into two equations, one for predicting the object state by using data about the robot's actions and another equation for incorporating sensor data into the estimation:

$$Bel^{-}(s_t) \longleftarrow \int \underbrace{p(s_t|s_{t-1}u_{t-1})}_{\text{process model}} Bel(s_{t-1}) ds_{t-1} \tag{8}$$

$$Bel(s_t) \longleftarrow \eta \underbrace{p(z_t|s_t)}_{\text{sensor model}} Bel^{-}(s_t) \tag{9}$$

$p(s_t|s_{t-1}u_{t-1})$ describes the probability function for object state s_t, after having been in state s_{t-1} and performing action u_{t-1}. E.g., when a robot is at position $s_{t-1} = (x, y)$ and it intended to walk a certain distance $u_{t-1} = (d_x, d_y)$, we get a probability for the next state $p(s_t = (x + d_x, y + s_y))$. Equations (8) and (9) are also the basic functions of a common Bayes filter. Equation (8) shows how the *a-priori* belief Bel^{-} is calculated from the previous belief $Bel^{-}(s_{t-1})$. It is called *a-priori belief* because it is the belief prior to the sensor data z_t, therefore it also often referred to as prediction. If we modeled the ball speed, in the prediction step we would calculate a new ball position, given the old position plus the current speed and the passed time since the last state estimation. Also actions of the robot, changing the ball state must be taken into account. But in our static situation nothing has to be propagated, because the ball position is static and the robot is not interacting with the ball. Furthermore, the ball position is modeled relatively to the field and not to the robot, which makes it independent from robot motions. In (9) the a-priori belief is updated by sensor data z_t, therefore called update step. The term $p(z_t|s_t)$ defines the probability for a measurement z_t given the object state s_t. In our case the probability function (sensor model) $p(z_t|s_t)$ is Gaussian, this means we will most likely measure the ball at the position, where it really is, but due to limited camera resolution our measurement is uncertain, so we could measure it at other positions with a lower probability as well. Our update information is information about object relations as described in section 3. Therefore the sensor model is needed, telling the filter how accurate the sensor data are. The particles are distributed equally at the beginning, then the filtering process begins. The main focus of our work is how the sensing process in a group of robots can become more distributed, so at this stage we abstract from how a particle filter works in detail and concentrate on how the sensor model of the particle filter can be extended.

3. Object Relation Sensor Modeling

Now we want to show, which kind of information can be gained from percept relations. Reference domain will be the play field of the RoboCup Sony Four Legged League which is a highly dynamic environment with moving and static objects, as fig. 2 shows. The whole field is color coded and includes

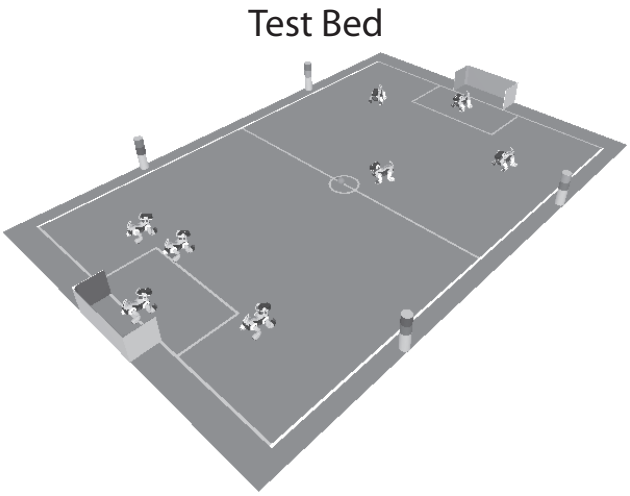

Figure 2. The play field of the Sony 4-Legged League.

flags and goals for localization and an orange ball. As a robot we used the Aibo ERS-7 from Sony which has a CCD camera with a horizontal opening angle of 55^o.

3.1. Landmark Information

Fig. 3 a) and b) show two example images and below the information that can be gained regarding possible ball positions (c and d). Seeing a ball and a flag at once results in a circular line for possible ball positions, in case of seeing a ball and a goal the resulting curve forms a spiral. The resulting percept relation includes the two angles to the flag or goal borders as well as the distance and angle to the ball.

In both cases one percept relation alone is not sufficient to exactly estimate the ball position on the field. As a consequence one robot could overcome this limitation by scanning for further landmarks standing in relation to the ball. But this could be time consuming, therefore communication with other agents comes into play. The key idea is that a whole bunch of agents perceives more than one robot alone. In the next section we will introduce an approach how to combine sensor readings from two or more robots into one object modeling framework.

3.2. Line Information

Another method to localize objects on the play field is by using field lines. Although their detection provides less information about the current positions because they are hardly distinguishable as long as there are no line crossings, they provide useful information because they often can be seen from every spot on the field. One possibility is, as fig. 4 shows, to use the distance of the ball to the line and compute all possible positions on the field. This can be extended to finding line crossing or the middle circle.

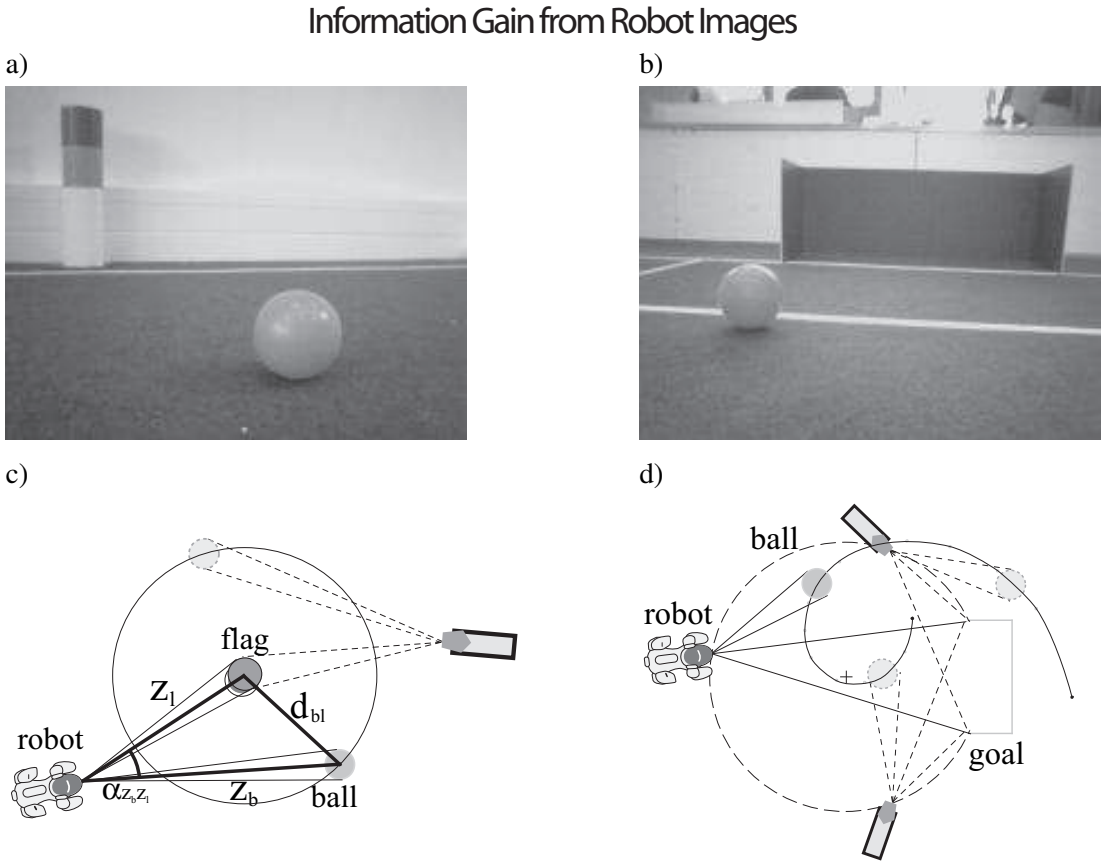

Figure 3. Single percept: a) a robot perceives a flag, i.e., the angle from left to right border of the flag in the image , b) perceiving a goal, i.e., the angle between left and right goalpost - and in both images the angle and distance to the ball
c) a flag and a ball let the robot determine the ball's distance relative to the flag d_{bl}; all possible positions of the ball relative to the flag form a circle, d) the same calculation for a goal and a ball. The circular arc determines all possible positions for the robot, the spiral arc represents all possible ball positions.

Now we have seen, which kind of information the robots perceive. In the next section we will describe, how these data can be combined.

4. Sensor Combination in Multi Robot Systems

There are several methods to combine the knowledge of two or more robots. One method is to combine the belief of the robots after the modeling (belief fusion) [4]. Another method is to communicate the sensor data, i.e., the percept relations before the modeling (sensor fusion). We decided for the latter approach because it yields several advantages as the next sections will show.

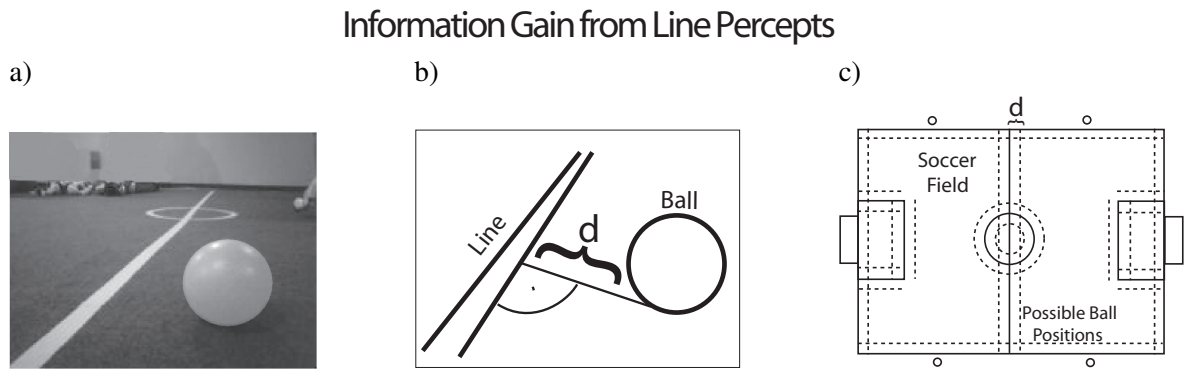

Figure 4. a) Shows a line and a ball. The robot can calculate the distance from the ball to the line d , as shown in b) . But as lines are not distinguishable from each other, all positions on the field lying in a distance of d to a field line are possible ball positions. c) Dotted points represent resulting possible ball positions in a distance of d to a field line.

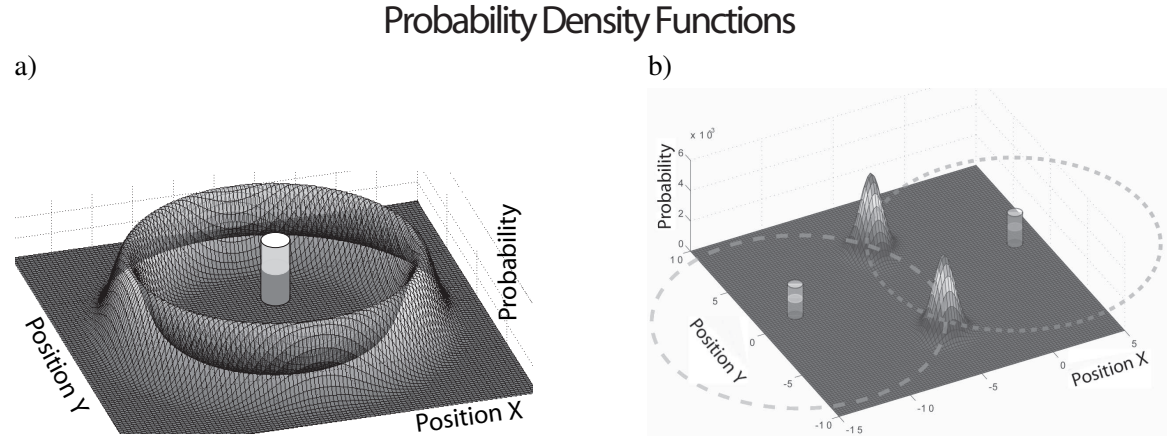

Figure 5. Probability distributions for the ball position - a) for one robot seeing a ball and a landmark. b) For two robots, each seeing one landmark and the ball.

4.1. Sensor Fusion

Sensor Fusion means in our case, that both robots are communicating the percept relations before the modeling. Later, all percept relations are incorporated into the modeling process equally, i.e., all percept relations, also the communicated ones are used for the likelihood calculation of the particle set during the update step of the particle filter. The advantage is, that percept relations are cheap to communicate - in contrast to communicating particle distributions. An example for a scenario in which two robots try to commonly model a ball position can be found in figure 5. As long as the situation is static, communication delays have not to be considered. Otherwise, when the situation becomes dynamic, one has to think about how to combine the sensor data of the different robots. In section 5 we will compare both methods in experiments.

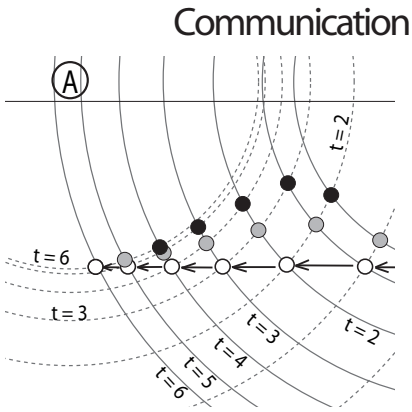

Figure 6. Effects of communication delay. White ball moving from right to left. Two agents (not in the diagram), agent A sees the ball and flag A, agent B communicates its percept relation consisting of the ball and flag B. Gray balls represent modeled ball positions with communication delay of 1, black balls represent modeled positions resulting from a delay of 2.

4.2. Handling Communication Delays

When there are communication delays in dynamic situations the communicated information must be assigned to the time when it has been perceived. Fig. 6 demonstrates the effect of to late arriving percept relations.

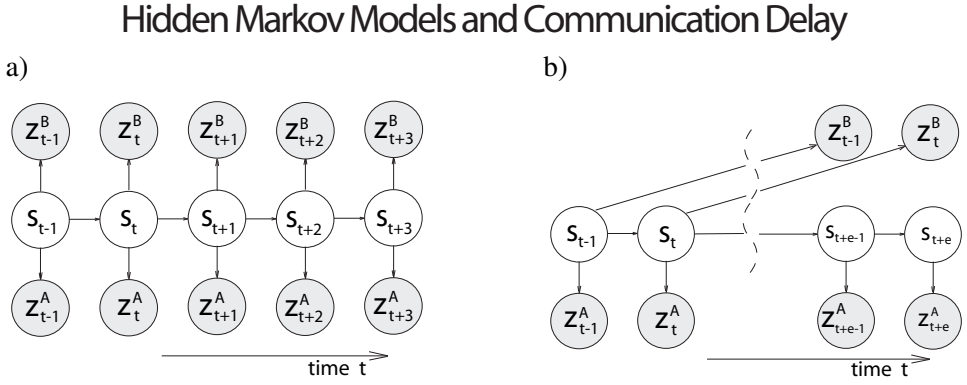

Figure 7. Robot b is sending its percept to robot A. a) no communication delays; b) communication delay $e > 0$.

We investigated, how delayed information can correctly be incorporated into the modeling process. Therefore assume that the time delay is constant and that it takes e time steps for the the percept to get from B to A. In case of $e = 0$ we get the Markov chain as in fig. 7. As a result, if one wants to model the object state s_t of time t, he has to wait until $t + e$ to get all sensor data from communicating agents.

As an example we want to model model state s_{t+e+1} with a given time delay of $e > 0$ time steps and two robots A and B. Our solution is, to use state s_{t+1}, where sensor information z^A_{t+1} and z^B_{t+1} is available, to remember the belief, e.g., particle distribution of s_{t+1} and then incrementally and a-priori model state s^*_{t+2}, s^*_{t+3}, ..., until s^*_{t+e+1} with just the sensor information z^A_{t+1} from A. In the next $t+e+2$ step we receive sensor data z^B_{t+2} from B, which we can use to revise the a-priori state estimation s^*_{t+2}. Therefore we remembered the belief function from s_{t+1}. Now we continue as already described. We call this method *history revision*. This approach allows incorporating communicated sensor data when time delays are given. It is feasible for small communication delays. But if the communication delay gets bigger the effort to revise the a-priori states from the past and to propagate them into the future gets linearly bigger.

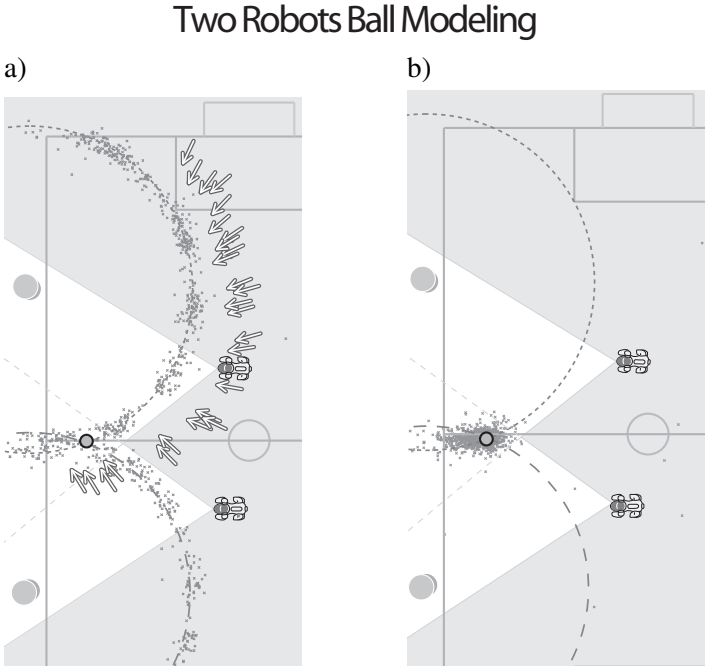

Figure 8. Particle distribution as seen from robot A: a) B is communicating its ball particle distribution to A, and A is adding it to its own distribution (small dots), the arrows indicate the self localization particle set from A; b) both robots are interchanging percept relations, which are then used for ball state estimation, the ball particle distribution from A converges to a small area.

5. Experiments

We conducted several experiments with the Aibo ERS-7 robot. As a representation of the object state we chose a Particle Filter because of its ability to represent arbitrary distributions. In the first experiment we placed both robots so that each one could see one flag only, thus being unable to self localize, and a ball. As a result (see fig. 8) we could see that communicating percept relations led to a quick convergence of the particle set, whereas just combining the two particle sets of the robots did not lead to any improvement. On the contrary, the entropy about the ball position on the field even increased.

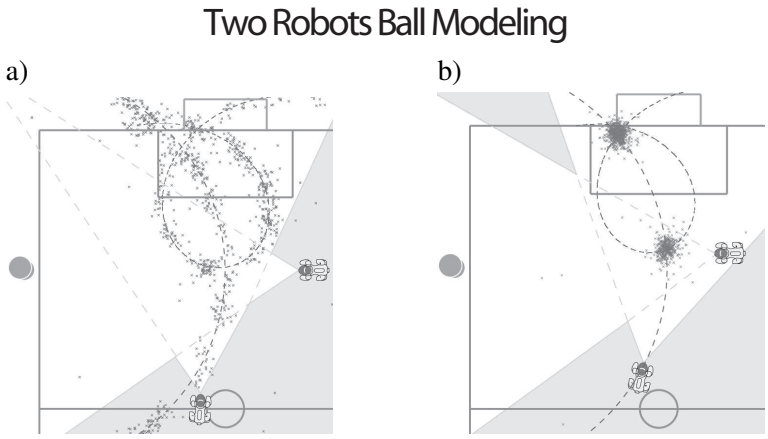

Figure 9. Same experiment as before, now one robot sees a flag and the ball, the other one is seeing the goal and the ball.

In experiment B we placed one robot in front of the goal, the other one in front of the flag (fig. 9). Again, interchanging percept relations helped to let the particle set of the ball converge to a certain area, whereas communicating particle distributions did not lead to any improvement of the state estimation.

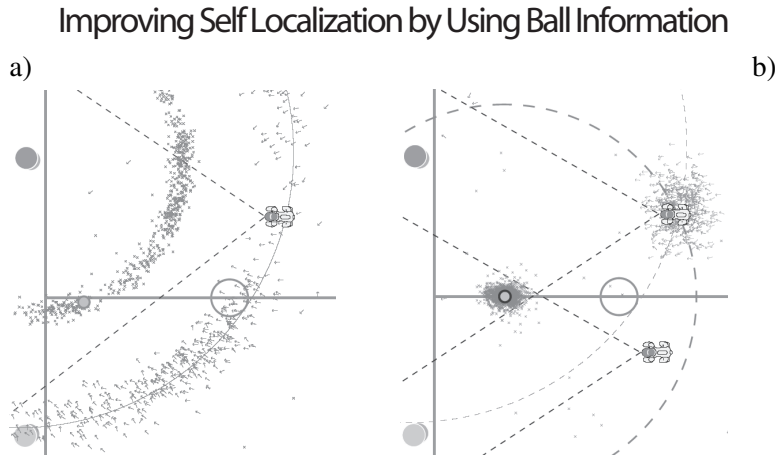

Figure 10. Same experiment as experiment A. Robot A uses its belief of the ball position to improve its self-localization a) no improvement of self-localization belief (small arrows around the robot) when only communication ball particle distribution; b) communicating percept relations of the ball leads to convergence of the localization particles (small arrows around the upper robot converged to one area)

In the last experiment we analyzed, how modeled objects could help a robot with its self localization. The situation is the same as in the first experiment, but now the robot A is using its new belief about the ball position for its own position estimation. Whereas in fig. 10 a) it was unable to accurately self localize, by using percept percept relations and its belief about the ball position, fig. 10 b) shows, that its localization belief also converged nicely.

6. Conclusion

We introduced a new approach for multi-agent world modeling in dynamic environments. Percept relations enable robots to commonly estimate and communicate object positions, even if the robots are badly self-localized. We presented an application example in the RoboCup environment. The given approach is extensible from one to n robots. The experimental data show the advantages of using percept relations. We also discussed how Markov modeling can be extended to overcome the limitations of delayed communication. Future work will investigate, how object relations can actively be searched for and how this approach can be applied to not just single-object but multi-object tracking.

References

[1] Wikipedia-bayesian network, September 2006.

[2] R. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, USA, 1998.

[3] J. Burlet, O. Aycard, and T. Fraichard. Robust navigation using markov models. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[4] M. Dietl, J. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01), Maui, Hawaii*, 2001.

[5] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI)*, pages 343–349. The AAAI Press/The MIT Press, 1999.

[6] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[7] K. Kaplan, B. Celik, T. Mericli, C. Mericli, and L. Akin. Practical extensions to vision-based monte carlo localization methods for robot soccer domain. In I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *9th International Workshop on RoboCup 2005 (Robot World Cup Soccer Games and Conference)*, Lecture Notes in Artificial Intelligence. Springer, 2006. To appear.

[8] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[9] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, volume 3276 of *Lecture Notes in Artificial Intelligence*, pages 18–33. Springer, 2005.

[10] S. Lenser, J. Bruce, and M. Veloso. CMPack: A complete software system for autonomous legged soccer robots. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 204–211. ACM Press, 2001.

[11] S. Lenser and M. M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*, pages 1225–1232. IEEE, 2000.

[12] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1985–1991. IEEE, 2003.

[13] J. Pearl. Fusion, propagation, and structuring in belief networks. In *Artificial Intelligence*, volume 29, pages 241–288, 1986.

[14] T. Röfer and M. Jüngel. Vision-based fast and reactive monte-carlo localization. In D. Polani, A. Bonarini, B. Browning, and K. Yoshida, editors, *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pages 856–861. IEEE, 2003.

[15] S. Thrun, D. Fox, and W. Burgard. Monte carlo localization with mixture proposal distribution. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 859–865, 2000.

[16] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. Technical Report CMU-CS-00-125, School of Comuputer Science, Carnegie Mellon University, April 2000.