

Reliable Multi Robot Coordination Using Minimal Communication and Neural Prediction

Sebastian Buck, Thorsten Schmitt, and Michael Beetz

Munich, University of Technology, Germany
{buck,schmittt,beetz}@in.tum.de

Abstract. In many multi robot applications, such as robot soccer, robot rescue, and exploration, a reliable coordination of robots is required. Robot teams in these applications should therefore be equipped with coordination mechanisms that work robustly despite communication capabilities being corrupted.

In this paper we propose a coordination mechanism in which each robot first computes a global task assignment for the team that minimizes the cost of achieving all tasks, and then executes the task assigned to itself. In this coordination mechanism a robot can infer the intentions of its team mates given their belief states. Lack of information caused by communication failures causes an increase of uncertainty with respect to the belief states of team mates. The cost of task achievement is estimated by a sophisticated temporal projection module that exploits learned dynamical models of the robots. We will show in experiments, both on real and simulated robots, that our coordination mechanism produces well coordinated behavior and that the coherence of task assignments gracefully degrades with communication failures.

1 Introduction

Multiple collaborating robots with a common goal usually have to coordinate their actions in order to avoid physical interferences and to achieve a maximum of speed-up. Reasonably in cooperative multi robot systems the common goal is decomposed into several tasks related to the individual robots of the system. The decomposition in terms of tasks is unique and changes depending on the current world state. Assuming such tasks can be performed by a single *action* each the question is how to assign tasks (actions) to the robots. Actually a sequence of actions is required for each robot to achieve the common goal. The complexity of this combinatorial problem increases exponential with the number of robots. Moreover the *cost* of a robot executing a certain action must be well known. This requires an accurate prediction.

Typical multi robot applications dealing with this problem include robot soccer, exploration, mine sweeping and messenger systems. The main advantages of multi robot systems over single robot systems are speed-up and fault tolerance [9, 11]. But without reasonable coordination a multi robot system can even be less efficient than a single robot system (e.g. two robots block each other). Therefore

Mataric [18, 19] suggests that control in multi robot systems must be addressed as a separate, novel, and unified problem, not an additional 'module' within a single-robot approach.

Information acquisition of robots in general occurs by sensors or communication. To achieve consistency and cooperation in multi robot behavior some kind of common basis for situation assessment is necessary (e.g. synchronization or a common world model) [30]. Young et al. [32] distinguishes between leader-following and behavioral schemes: While in the first approach one robot organizes the whole coordination and assigns actions to the other robots in the latter approach the robots autonomously decide supported by more or less communication. An example for a leader-robot system is given in [10]. While in the leader-following approach success strongly depends on the leader-robot and therefore fault tolerance is poor the behavioral approach requires intelligent software and computational resources on *all* robots. Behavioral systems can be categorized in those with a shared model of the environment, those with a shared abstract description of the environment's state (e.g. at the planning level), and those without shared information. Examples for systems using global maps are the collaborative exploration systems described in [8, 27]. Alur et al. [1] periodically exchanges information at discrete time intervals. Many other systems rely on communication too [14, 17, 23]. The system MAPS [31] uses globally shared information remodeled in an abstract virtual space. Multi robot systems with a shared model of the environment require the computation of such a model which is nontrivial. Systems with synchronization on an abstract description level oblige to change the implementation whenever anything changes on that abstract level (e.g. priorities in planning). Furthermore it is argued that communication may lead to delays in information acquisition and can increase complexity and degrade multi robot systems [12, 16, 26]. However, using no shared information means to be dependent on an accurate state estimation of the environment.

Assuming there is a common basis for action selection that characterizes the current state of the whole multi robot system there is still the question of how to best coordinate the robots' actions. Agents must reason about expected team utilities of future team states [29]. Forestalling interferences and estimating the team utility requires in general the prediction of the consequences of a particular action assignment for the robot team.

The approach proposed in this paper engages a minimum of communication while primarily relying on a robot's local information. The robots in our system share no global model of the environment but individually estimate the state of the environment with such an accuracy that a periodically exchange of some key features (e.g. distances and angles) of the local estimations among the robots suffices to ensure a cooperative action selection. Moreover these key features enable one robot to put itself in each others robot's situation and thereby to compute the action selected by the other robot. The action selection process employs a state-projection system based on neural networks that is able to project the robots' states into the future. This Projector is supported by a hybrid multi

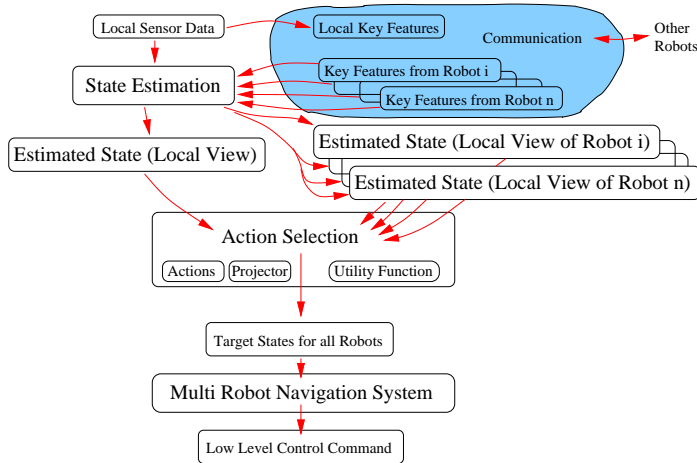


Fig. 1. A robot architecture in a multi robot system to facilitate cooperative behavior: The robot's *state estimation* is based on *local sensor data* and *key features* from the other robots of the system (if available via communication). Local key features are sent to the other robots. The robot estimates the environment's state not only from its local view but also from the views of the other robots. Thus cooperative *action selection* is done by a *utility function* which can rely on all of the robot's *local views*. A neural projector supports the utility function by estimating the time need for desired state changes. Thereafter the target states of the actions are given to the *multi robot navigation system* which considers path planning and computes the appropriate *low level control command* for the robot.

robot navigation system. The whole system is implemented and tested in highly dynamic robot soccer environment.

The remainder of the Paper is organized as follows: In Section 2 we give an overview on the proposed multi robot coordination architecture and the action selection algorithm and discuss advantages and disadvantages. Section 3.1 shortly describes how a single robot localizes itself and other objects and generates a model of its environment. Section 3.2 introduces the neural projection system and describes how it works in conjunction with the employed multi robot navigation system and the action selection algorithm. In Section 4 we put our approach in relation with two concrete examples of robot soccer ((a) which robot is to go for the ball and (b) playing double pass without explicitly learning it) and give results of empirical investigations. Finally in section 5 we close with a conclusion.

2 Overview

To coordinate a team of multiple robots with a common goal in a shared environment we use a layered hybrid system [2] containing a state estimation module, an action selection unit, and a multi robot navigation system. The hybrid architecture works as follows (see fig. 1).

The robot receives sensor data from its camera and some key features from its team mates. Out of that information a model of its environment is built. Thereby principally local information is used while the other robots' key features (if available) are used for evidence. Additionally the robot constructs a model of the environment from each other robot's local view by primarily relying on the key features of that robot. If those features are not available due to communication problems only local information is used.

This technique enables the action selection unit not only to choose an action from a local point of view but to put the robot in another robot's situation and thereby to consider the choice of that robot. This consistency allows the robot to behave cooperative and avoids robots interfering one another. Action selection is done by a simple utility function which is based on a priority list of actions. It is supported by a sophisticated neural projection system which estimates the time need for a requested change in state. Actions are assigned at a frequency of 10Hz and may change depending on the world state even if the respective task was not completed. There is no explicit synchronization between the robots but each robot deciding every 100ms allows only very short times of double assignments.

Each action can be mapped to a certain target state for the robot. All of those target states (consisting of position, orientation etc.) are given to the multi robot navigation system whose task it is to compute a low level control command leading the robot towards its target state considering obstacle avoidance and motion dynamics.

A Cooperative Action Selection Algorithm Action selection from a set of actions \mathcal{A} in the context of a multi robot system \mathcal{R} with a common goal means to find a mapping $\mathcal{S} : \mathcal{R} \rightarrow \mathcal{A}$ which assigns each robot of the team $r_i \in \mathcal{R}$ a different action (task) $a_j \in \mathcal{A}$ in a way that the common goal can be performed with minimal costs ($c(r_i, a_j)$ is the cost for robot r_i performing action a_j).

$$\min \sum_{i=1}^{|\mathcal{R}|} c(r_i, \mathcal{S}(r_i)) \quad \mathcal{S}(r_i) \neq \mathcal{S}(r_j) \Leftrightarrow i \neq j \quad (1)$$

Since this combinatorial problem is NP-hard usually heuristics are applied for minimizing the cost function. Assuming there are only four robots and four actions given this problem can be solved exactly with the amount of comparing all 24 options.

All the above considerations are based on the assumption that each robot can perform all actions and most notably that the performance of all actions is equally important. But in mine sweeping disabling a mine directly near a civil institution is prior and in robot soccer shooting the ball is more important than any other action in an offensive situation. Furthermore not all robots can execute all actions: A soccer robot without ball cannot shoot the ball and a robot that is stuck cannot move at all. Recapitulatory this means for certain applications the use of a priority list as well as a feasibility list of actions is necessary. Hence we define a feasibility function

$$\mathcal{F} : \mathcal{R} \times \mathcal{A} \rightarrow \{true, false\} \quad (2)$$

mapping a robot and an action to a boolean value (estimating the probability of success \mathcal{F} can map to $[0, 1]$ too [7]. In this case the following algorithm has to be modified and thresholds are used). In the following algorithm $\mathcal{A}(\hat{\mathcal{R}})$ denotes the priority list of actions (the set of idle robots). This algorithm takes into account that the actions are not equally important and not necessarily feasible for all robots.

$$\begin{aligned}
&\hat{\mathcal{R}} = \mathcal{R} \\
&\text{for } i = 1 \text{ to } |\hat{\mathcal{R}}| \\
&\quad \mathcal{S}(r_i) = no_operation \\
&\text{for } i = 1 \text{ to } |\mathcal{A}| \\
&\quad \text{if } \hat{\mathcal{R}} \neq \emptyset \wedge \min_{r \in \hat{\mathcal{R}}} (c(r, a_i)) < \infty \\
&\quad\quad r_j = \arg \min_{r \in \hat{\mathcal{R}}} (c(r, a_i)) \\
&\quad\quad \mathcal{S}(r_j) = a_i \\
&\quad\quad \hat{\mathcal{R}} = \hat{\mathcal{R}} \setminus r_j
\end{aligned} \quad (3)$$

The algorithm initializes all robots with the action *no_operation* and then loops for all actions in the order of priority.

$$priority(a_i) > priority(a_j) \Leftrightarrow i < j \quad (4)$$

If there is an idle robot left which can perform the current action the robot with minimal cost to perform this action is chosen to do so. The costs $c(r, a)$ are set to infinity if a robot r cannot perform the action a and to a predicted value $\mathcal{P}(r, a)$ otherwise. The prediction \mathcal{P} depends on the application dependent criterion of optimization.

$$c(r, a) = \begin{cases} \infty & \text{if } \mathcal{F}(r, a) = false \\ \mathcal{P}(r, a) & \text{otherwise} \end{cases} \quad (5)$$

Using the above algorithm each robot employs \mathcal{P} not only for predicting its own cost but for comparing it with the values computed for its team mates. Relying on identical software and the same local environment data the action selection of this algorithm is unique and consistent. By predicting the cost for robot r_i performing a less important action a_j \mathcal{P} can already rely on the knowledge which robot is to perform a more important action and in case there might be any interference $\mathcal{P}(r_i, a_j)$ will increase. This mechanism forces cooperative behavior and informs all robots on the action selection of their team mates. The major advantages of this approach so far are

- A robot can put itself into the situation of a team mate and thereby will behave cooperative by considering the team mates' decisions.
- All sequences of chosen actions are locally known and toggling situations in action selection can easily be detected and avoided.

- The algorithm is resistant against a crash of a single robot which will be detected (if not moving *and* not communicating for a certain time). Likewise a new robot can easily be integrated in the system. In systems using a high-accuracy state estimation there is even no communication necessary.
- The concept works fine with homogeneous and heterogenous robots. For heterogenous robots \mathcal{P} must be implemented for each different robot.
- The laborious computation of a global map of the environment is not necessary. There is no loss of time for broadcasting a global map.

3 Cooperative Action Selection

3.1 State Estimation

Since action selection strongly depends on a robots view of its environment we first describe how a robot estimates its state.

Probabilistic State Estimation We employ a state estimation module for individual autonomous robots [25] that enables a team of robots to estimate their joint positions in a known environment (such as a soccer field or an office building) and track the positions of autonomously moving objects. The state estimation modules of different robots cooperate to increase the accuracy and reliability of the estimation process. In particular, the cooperation between the robots enables them to track temporarily occluded objects and to faster recover their position after they have lost track of it.

The state estimation module of a single robot is decomposed into subcomponents for self-localization [13] and for tracking different kinds of objects. This decomposition reduces the overall complexity of the state estimation process and enables the robots to exploit the structures and assumptions underlying the different subtasks of the complete estimation task. Accuracy and reliability is further increased through the cooperation of these subcomponents. In this cooperation the estimated state of one subcomponent is used as evidence by the other subcomponents.

Considering further Physical Properties for State Estimation So far physical properties like inertia and acceleration of robots are not considered for state estimation. All robots are dealing more or less with a dead time which means that at the moment a robot is performing visual localization it has already sent control commands (e.g. translational and rotational velocity) for further movements. These sent commands can be used to predict the robot's state a little time ahead. This requires a mapping

$$\Delta : \zeta_c \times \xi \mapsto \zeta_{succ} \quad (6)$$

from a current state ζ_c and a sent command ξ to the successor state ζ_{succ} . Δ is learned from experience, that is recorded data from real robot runs. We

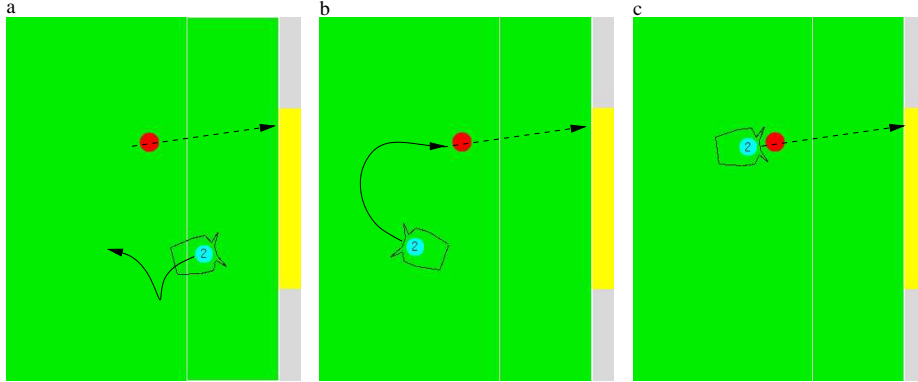


Fig. 2. A training scenario in the multi robot simulation environment (M-ROSE): To acquire training patterns for the neural projector \mathcal{P} a robot is set to a randomly defined start state ζ_s (position of the robot in subfigure a) and has to drive to a randomly defined target state ζ_t indicated by the dashed arrow. The direction and length of this arrow indicate the target state's orientation and velocity. The time the robot needs to reach its target state (subfigure b & c) is taken to complete the training pattern $\langle\langle\zeta_s, \zeta_t\rangle, time\rangle$.

propose to learn this mapping using a simple multi layer neural network and supervised learning with the RPROP algorithm [22]. Depending on the number of command cycles equating with the dead time Δ has to be applied repeatedly to the current state. This enables a robot to perform action selection and path planning considering its future state which is predetermined anyway.

In many mobile robot applications robots can get stuck or blocked by other objects. It is very useful to detect such a situation because it may strongly change the process of action selection not only of the affected robot but of all team mates too. To detect such a situation we employ Δ once again. Recording the low level commands ξ sent to the robot we can predict the robot's changes in state. If they differ significantly from the measured changes in state we assume the robot is not able to move correctly. This information is posted among the robots and will be considered in action selection. If communication fails and a robot does not move a certain time it will be considered dead anyway. Moving again it will be considered alive again.

3.2 A Neural Projection System to Estimate the Time Need for Changes in State in Robotics

As mentioned in section 2 the costs $c(r_i, a_j)$ of a robot r_i performing an action a_j depend on the specific application. In traveling it may be the way, in mine sweeping the time use per mine and in robot soccer the time to score or prevent a goal.

As written above we decompose a goal into several tasks which can be performed by an action each. In our case the costs $c(r_i, a_j)$ of a robot r_i can be

described as the time to complete an action a_j . Assuming that to complete an action means to reach a certain target state we try to predict the time a robot needs to reach that target state. This will give us an information how suitable it is that a robot performs a proposed task. That means we have to estimate the time need of a robot to complete an action considering the following:

- Knowledge of the dynamic behavior of the robot itself must be acquired.
- Actions of team mates must be taken into account.
- Other dynamic objects must be regarded as moving obstacles.

Due to the physical complexity of this problem it seems impossible to estimate the amount of time without learning algorithms. But learning the projection

$$\mathcal{P} : \mathcal{R} \times \mathcal{A} \rightarrow \text{time} \quad (7)$$

with data from real robot runs would require an impractical huge amount of time for data acquisition. Less expensive is to construct a robot simulator which mimics the physical behavior of the robots. Our multi robot simulation environment (M-ROSE,[4]) is based on the state change function Δ (equation 6). Not only the development of a low level controller but the implementation of an action selection unit are substantially simplified by this. Once an accurate level of simulation is achieved one can obtain unlimited training data for learning from such a multi robot simulator.

Learning a Neural Projection Neural Networks have been shown to be an accurate means for the prediction of run-times (see Smith et al.[28] for example). Hence we choose neural learning to obtain \mathcal{P} . We apply multi layer networks and the backpropagation derivative RPROP [22] because of its adaptive step-size computation in gradient descent. The data we use to train the neural projector \mathcal{P} is completely obtained from the also learned simulator in minimal time. The training patterns are of the form $\langle\langle\zeta_s, \zeta_t\rangle, \text{time}\rangle$ where ζ_s is the randomly chosen start state of a robot and ζ_t its also randomly chosen target state in the simulation. We get the necessary value of *time* by simulating a robot driving from ζ_s to ζ_t (see fig. 2). \mathcal{P} was trained with around 300.000 patterns using a network with input, output, and two hidden layers. At learning time there are no other objects or team mates taken into consideration. Using validation patterns for early stopping [24] the trained network achieved an average error of 0.13 seconds per prediction on a test set not used for learning. Due to the inherent indeterministic robot behavior and noise this is an acceptable result.

Taking the Actions of Team Mates into Account Using the proposed algorithm (3) to select a robot to execute an action a_j we can consider the behavior of all robots executing actions a_i under the condition that $i < j$ (a_i prior to a_j). Thus we can use this knowledge to compute $\mathcal{P}(r, a_j)$ for any robot $r \in \hat{\mathcal{R}}$. Since we know all start and target states of all robots executing actions a_k ($k \leq j$) a

set of start and target states as well as the priority of each target state (action) is given. A multi robot navigation system receives this data and computes the paths for all concerned robots including r . The multi robot navigation system (described more in detail in [5, 6]) consists of three components: an artificial neural network controller, a library of software tools for planning and plan merging, and a decision module that selects the appropriate planning and execution methods in a situation-specific way. The system has learned predictive models for the performance of different navigation planning methods by experimentation. The decision module uses the learned predictive models to select the most promising planning methods for the given navigation task. If the multi robot navigation system proposes to apply a path planning method for r to get from its start state ζ_s to its target state ζ_t and the first intermediate state on the computed path is ζ_i the time need is set to

$$\mathcal{P}(\zeta_s, \zeta_t) = \mathcal{P}(\zeta_s, \zeta_i) + \mathcal{P}(\zeta_i, \zeta_t) \quad (8)$$

This equation may be used recursively for the computation of $\mathcal{P}(\zeta_i, \zeta_t)$ if ζ_i is not the last intermediate state on the computed path.

Considering Moving Obstacles The multi robot navigation system considers moving obstacles as well. The objective of the navigation system is to achieve a state where each robot is at its target state as fast as possible. A set of representative features considering all obstacles is computed to characterize the navigation task. These features are then tested by a learned decision tree (see [5, 6] for more details) that chooses the most promising single robot path planning and plan merging method provided by a toolbox of algorithms. The decision tree was trained using moving obstacles with random direction. The toolbox extracts sequences of target states from the repaired plans and sends those sequences to the neural network controllers of the respective robots.

4 Empirical Investigations

The algorithm described in section 2 has been implemented and extensively tested in simulation and real robot environment. Being highly reliant on cooperation soccer playing robots are a suitable appliance for the approach described above. We observed (a) the behavior of a team of 11 autonomous soccer robots in the RoboCup simulation league environment and (b) the behavior of 4 real robots belonging to the RoboCup MidSize league.

4.1 Simulation Experiments

In the RoboCup simulation environment a soccer team consists of 11 autonomous software agents communicating with a server program [15, 20]. The agents receive sensory data and send low level control commands. Using the *Karlsruhe*

Brainstormers agent of RoboCup 1999 [21] as a basis for our experiments each robot can avail itself of a set of actions

$$\mathcal{A} = \{shoot2goal, pass2player, dribble, receive_pass, go2ball, offer4pass, gohome\}$$

which are mapped to low level commands by the agent. The feasibility function \mathcal{F} is instantiated by hand-coded functions based on situation dependent features. Action selection is done at the frequency of 10Hz. There is no direct communication between the agents but a limited communication via the soccer server simulation program.

We played several games and recorded the locally selected actions of all agents and measured how many times the agent chosen to receive a pass by the agent playing the pass was identic with the agent planning to receive a pass in relation to all passes played. At this we rated a match within a temporal difference of 0.1 seconds (1 simulation cycle) positive. This quotient gives us a measure of cooperation since pass play is a paradigm of cooperation. The experiments were performed with working communication and temporary disordered communication. In the following table one can see the above explained quotient with working communication and with a communication breakdown every 30, 60, and 120 seconds which lasts t_{bd} seconds with t_{bd} uniformly distributed over $[0, 30]$ seconds. Furthermore we played games without any communication. Each result is based on 10 games respectively.

Communication	Matches (pass player / receiver)	Average goals per game
full	82.1%	8.1
breakdown each 120 s	76.2%	7.4
breakdown each 60 s	70.1%	6.5
breakdown each 30 s	66.1%	6.1
no communication	63.9%	5.7

One can see that an increasing frequency of communication breakdown accompanies with a decreasing number of goals scored per game. Moreover less communication means less successful planning of pass play and therewith less coordination. Nevertheless a team using no communication is able to score around 70% of the goals a team with full communication scores. Additionally 77.8% of the matching quotient of a team using full communication is reached by a team without any communication. These results document that the chosen approach for action selection and coordination is robust and can deal very well with temporary failures.

Double Pass Play Without ever explicitly learning to play a double pass and no special double-pass-plan specified not seldom double pass play was observed. Analyzing some cases of double pass play we found a reasonable pattern to explain this effect (see fig. 3). Robot number 3 holds the ball and decides to pass to player number 2. Meanwhile player number 2 puts itself in the situation of player number 3 and recognizes that it has to receive a pass:

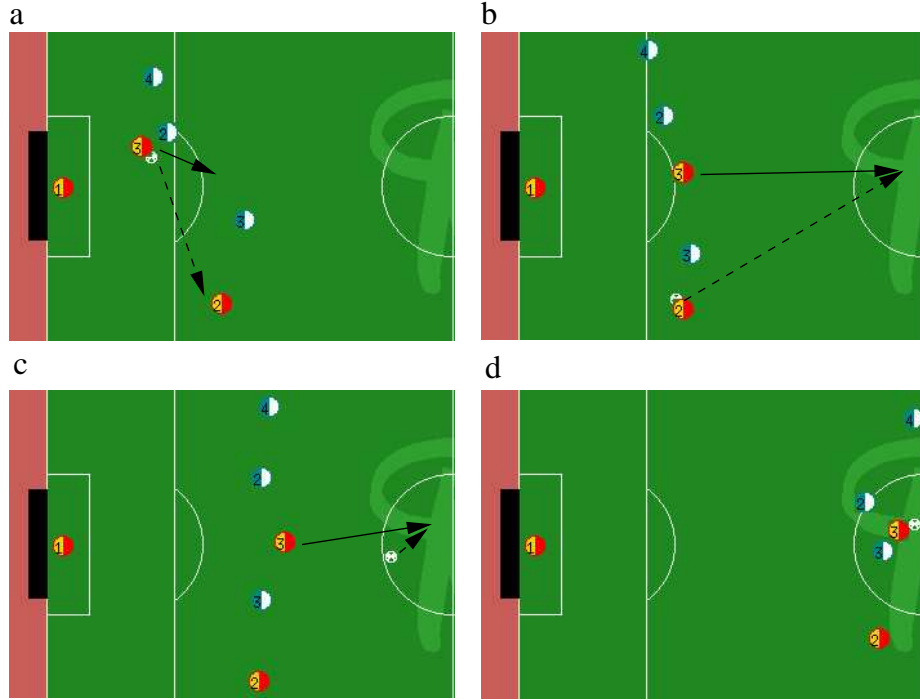


Fig. 3. A double pass scenario in the RoboCup soccer server environment. Player 3 chooses to play a pass to player 2 while at the same time player 2 awaits the ball (subfigure a). After player 3 gets rid of the ball he starts to offer itself for a pass play. As player 2 gets the ball player 3 computes that player 2 will pass the ball again (subfigure b). Finally player 3 gets back the ball (subfigure c & d).

$$\mathcal{S}(r_3) = \text{pass2player}_2$$

$$\mathcal{S}(r_2) = \text{receive_pass}_3$$

Immediately after having played the ball player 3 performs $\mathcal{S}(r_3) = \text{offer}_4\text{pass}$ to offer itself for a receipt of a pass (fig. 3a). As player 2 receives the ball it chooses to play a pass to player 3. Meanwhile player 3 puts itself in the situation of player number 2 and recognizes that it has to receive a pass (fig. 3b):

$$\mathcal{S}(r_2) = \text{pass2player}_3$$

$$\mathcal{S}(r_3) = \text{receive_pass}_2$$

Further player 3 performs $\mathcal{S}(r_3) = \text{receive_pass}$ while player 2 performs $\mathcal{S}(r_2) = \text{offer}_4\text{pass}$ after playing the pass back to player 3 (fig. 3c). Receiving the ball again player 3 is to choose from $\{\text{shoot2goal}, \text{pass2player}, \text{dribble}\}$ again (fig. 3d).

4.2 Real Robot Experiments

To evaluate our approach in a real robot environment we choose the RoboCup MidSize league. Two teams of 4 players compete on a field of about 9 meters in

length and 5 meters in width. Compared with the RoboCup simulation league there are some new challenging problems: The sensory data is not provided by a server program but must be acquired by the robot itself. Furthermore most robots are not able to receive a pass from any direction (like in simulation league) and path planning is more important on a comparably small field (the field in simulation league is 105 meters long). For our experiments we use the Agilo RoboCuppers team [3] as a basis. The available list of actions looks as follows:

$$\mathcal{A} = \{shoot2goal, dribble, clear_ball, go2ball, gohome, get_unstuck\}$$

To demonstrate the coordination of the team we measure the number of robots performing *go2ball* at the same time. Further we observe how long the same robot performs *go2ball* without being interrupted by another robot. The data was acquired from five real robot games against different opponent teams at the international robot soccer world cup 2001. The following table depicts in how much percent of the whole time played none, one, and more than one robot performed *go2ball*. The frequency of action selection is around 10Hz.

#robots performing <i>go2ball</i> at the same time	quota in relation to the whole time played
0	00.34%
1	98.64%
> 1	01.02%

The average time one robot performs *go2ball* or handles the ball without being interrupted by a decision of a team mate is 3.35 seconds. In only 0.25% of the time a robot that is stuck is determined to go for the ball by the other robots.

These results show that, in the context of robot soccer, coordination is warranted to a great extent. There are hardly ever situations where no robot or more than one robot approaches the ball at a time and the situations in which it is not clear which robot is to go for the ball are just a few.

Solving Situations with Stuck Robots In a highly dynamic environment like robot soccer not seldom a robot gets stuck due to another robot blocking it. The following example shows how such incidents were handled by the robots in our experiments (see fig. 4). Robot number 2 is supposed to be the fastest to get the ball and therefore approaches the ball (fig. 4a): $\mathcal{S}(r_2) = go2ball$. Near the ball robot 2 collides with an opponent robot. Robot 2 is in a deadlock situation and cannot move forward anymore. The only action feasible to execute remains *get_unstuck*. Thus robot 3 approaches the ball now (fig. 4b):

$$\mathcal{F}(r_2, a) = 0 \quad \forall a \in \mathcal{A} \setminus get_unstuck \quad \mathcal{S}(r_2) = get_unstuck \quad \mathcal{S}(r_3) = go2ball$$

Having reached the ball robot 3 dribbles towards the opponent goal while robot 2 is moving backwards (fig. 4c):

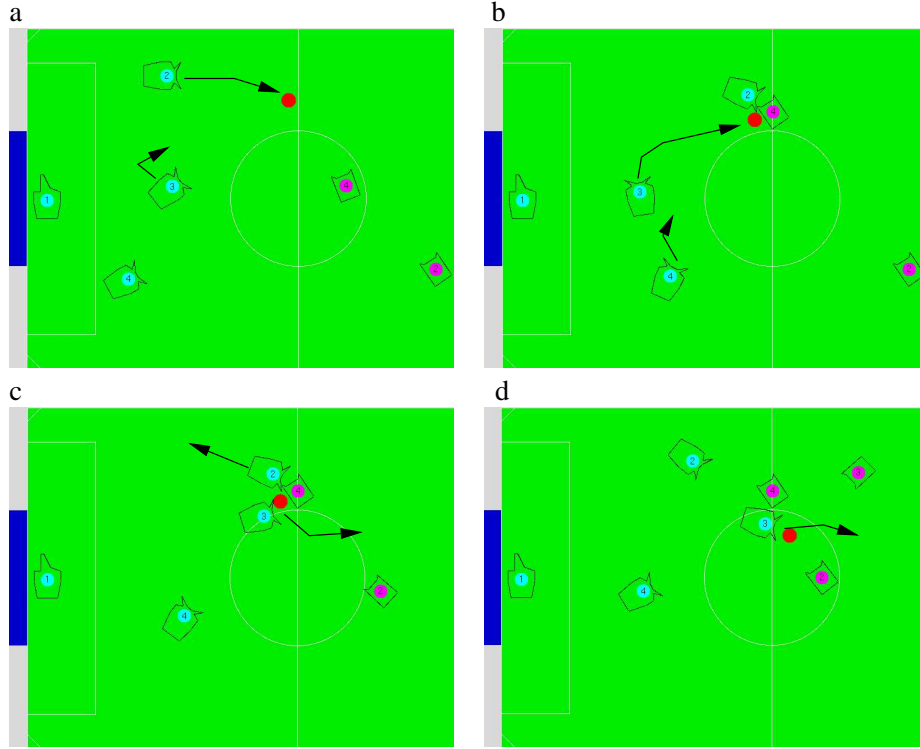


Fig. 4. An example for intelligent cooperation in a real robot soccer environment. Robot 2 approaches the ball (subfigure a) and thereby collides with a robot of the opponent team (subfigure b). As the opponent robot constantly pushes robot 2 is stuck and temporary not regarded by the other robots. Thus robot 3 moves towards the ball while robot 2 tries to get unstuck (subfigure c). Finally robot 3 dribbles towards the opponent goal while robot 2 is staying back in its own half (subfigure d).

$$\mathcal{F}(r_2, a) = 0 \quad \forall a \in \mathcal{A} \setminus \text{get_unstuck} \quad \mathcal{S}(r_2) = \text{get_unstuck} \quad \mathcal{S}(r_3) = \text{dribble}$$

Further on robot 2 is no more stuck and robot 3 is still dribbling:

$$\mathcal{F}(r_2, \text{go2ball}) = 1 \quad \mathcal{S}(r_2) = \text{gohome} \quad \mathcal{S}(r_3) = \text{dribble}$$

5 Conclusions

In this paper we propose an autonomous approach to collaborative action selection for multi robot environments. Action selection directly depends on the model of the environment. Our state estimation process of a single robot is mainly based on local sensor data while information from other robots is used for evidence only. Decision making is supported by the ability of a robot to put

itself in its team mates' situation. This mechanism used by humans and apes allows cooperative behavior among the robots. Moreover employing a neural projection system each robot can estimate the time need for itself or other robots to reach a certain state. Our technique has been implemented and tested in the highly dynamic and cooperation-dependent RoboCup environment both in simulation and in real robot runs. The results show that the proposed approach is reliable and, to a high extent, fault tolerant even if there is no communication between the robots but a reliable localization.

Compared to most previous methods our concept is neither leader-following nor dependent on a shared global map of the environment. Each robot decides completely autonomous supported by neural prediction. In action selection we consider the feasibility of an action as well as its costs and its priority. The main advantages of our method are extensibility, the applicability to heterogenous robots, stability in situations of failures of single robots, the consistency in action assignment, and that the laborious computation of a shared global map is not necessary.

Multi robot action selection in common is assumed to be a combinatorial problem that requires an exponential computational amount to be solved exactly. However we have to consider feasibility and priority of actions as well. To improve performance in the aspect of time sophisticated heuristics are required. We believe that a reasonable set of executable actions as well as the optimization in performance of every single action will substantially support the effectiveness of our algorithm. These extensions are subject of our future investigations as well as the integration of long term plans and the extension to other applications.

Acknowledgements

We would like to thank our students (Maximilian Fischer and Andreas Hofhauser) for their contributions to our RoboCup team. Further thanks go to Artur Merke and Martin Riedmiller (members of the RoboCup Simulation Team *Brainstormers*) for supporting the simulation experiments and providing the *n++* neural network library. Finally we would like to thank the staff and the organizers of the *Schloss Dagstuhl Seminar* for the event and the *RoboCup Federation* and the *DFG* for supporting our work.

References

1. R. Alur, J. Esposito, M. Kim, V. Kumar, and I. Lee: *Formal modeling and analysis of hybrid systems: A case study in multirobot coordination*. FM'99: Proceedings of the World Congress on Formal Methods, LNCS 1708, pp. 212–232, Springer, 1999.
2. R.C. Arkin: *Towards the Unification of Navigational Planning and Reactive Control*. AAAI Spring Symposium on Robot Navigation, pp. 1-5, 1989.
3. M. Beetz, S. Buck, R. Hanek, T. Schmitt, and B. Radig: *The Agilo Autonomous Robot Soccer Team: Computational Principles, Experiences, and Perspectives*. International Joint Conference on Autonomous Agents and Multi Agent Systems (AA-MAS) 2002.

4. S. Buck, M. Beetz, and T. Schmitt: *M-ROSE: A Multi Robot Simulation Environment for Learning Cooperative Behavior*. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa (eds.): *Distributed Autonomous Robotic Systems 5*, 2002, Springer Verlag.
5. S. Buck, U. Weber, M. Beetz, and T. Schmitt: *Multi Robot Path Planning for Dynamic Environments: A Case Study*. Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2001.
6. S. Buck, M. Beetz, and T. Schmitt: *Planning and Executing Joint Navigation Tasks in Autonomous Robot Soccer*. 5th International Workshop on RoboCup, Lecture Notes in Artificial Intelligence, Springer Verlag, 2001.
7. S. Buck and M. Riedmiller: *Learning Situation Dependent Success Rates Of Actions In A RoboCup Scenario*. Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, Springer 2000.
8. W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun: *Collaborative multi-robot localization*. Proceedings of the IEEE International Conference on Robotics and Automation, 2000.
9. Y.U. Cao, A.S. Fukunaga, and A.B. Khang: *Cooperative mobile robotics: Antecedents and directions*. *Autonomous Robots*, 4, 1997.
10. Q. Chen and J.Y.S. Luh: *Coordination and control of a group of small mobile robots*. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2315-2320, 1994.
11. G. Dudek, M. Jenkin, E.E. Milios, and D. Wilkes: *A taxonomy for multi-agent robotics*. *Autonomous Robots*, 3(4), 1996.
12. A. Garland and R. Alterman: *Multiagent Learning through Collective Memory*. AAAI Spring Symposium Series 1996: *Adaption, Co-evolution and Learning in Multiagent Systems*, 1996.
13. R. Hanek and T. Schmitt: *Vision-Based Localization and Data Fusion in a System of Cooperating Mobile Robots*. Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pages 1199-1204, 2000.
14. N. Jennings: *Controlling cooperative problem solving in industrial multi-agent systems using joint intentions*. *Artificial Intelligence*, 75, 1995.
15. H. Kitano, M. Tambe, P. Stone, S. Coradeschi, H. Matsubara, M. Veloso, I. Noda, E. Osawa, and M. Asada: *The robocup synthetic agents' challenge*. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.
16. R. Kube and H. Zhang: *Collective Robotics: From Social Insects to Robots*. *Adaptive Behaviour*, Vol. 2, No. 2, pp. 189-218, 1994.
17. H.J. Levesque, P.R. Cohen, and J. Nunes: *On acting together*. In Proceedings of the National Conference on Artificial Intelligence, AAAI press, 1990.
18. M.J. Mataric: *Coordination and Learning in Multi-Robot Systems*. *IEEE Intelligent Systems*, Mar/Apr 1998, 6-8.
19. M.J. Mataric: *Using Communication to Reduce Locality in Distributed Multi-Agent Learning*. *Journal of Experimental and Theoretical Artificial Intelligence*, special issue on Learning in DAI Systems, Gerhard Weiss, ed., 10(3), Jul-Sep, 357-369, 1998.
20. I. Noda, H. Matsubara, K. Hiraki, and I. Frank: *Soccer Server: A Tool for Research on Multiagent Systems*. *Applied Artificial Intelligence*, 12, 2-3, pp.233-250, 1998.
21. M. Riedmiller, S. Buck, A. Merke, R. Ehrmann, O. Thate, S. Dilger, A. Sinner, A. Hofmann, and L. Frommberger: *Karlsruhe Brainstormers - Design Principles*. In M. Veloso, E. Pagello, H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Lecture Notes in Artificial Intelligence, pp 588-591, Springer Verlag, 1999.

22. M. Riedmiller and H. Braun: *A direct adaptive method for faster backpropagation learning: the Rprop algorithm*, Proceedings of the ICNN, San Francisco, 1993.
23. A. Saffiotti, N. B. Zumel, and E. H. Ruspini: *Multi-robot Team Coordination using Desirabilities*. Proceedings of the Sixth International Conference on Intelligent Autonomous Systems, 2000.
24. W.S. Sarle: *Stopped training and other remedies for overfitting*. In Proceedings of the 27th Symposium on Interface, 1995.
25. T. Schmitt, R. Hanek, S. Buck, and M. Beetz: *Cooperative Probabilistic State Estimation for Vision-based Autonomous Mobile Robots*. Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2001.
26. S. Sen, S. Mahendra, and J. Hale: *Learning to Coordinate Without Sharing Information*. Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 426-431, 1994.
27. R.G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes: *Coordination for multi-robot exploration and mapping*. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 852-858, 2000.
28. W. Smith, V.E. Taylor, and I. Foster: *Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance*. JSSPP, pp. 202-219, 1999.
29. M. Tambe and W. Zhang: *Towards flexible teamwork in persistent teams*. Proceedings of the International conference on multi-agent systems (ICMAS), 1998.
30. A. Tews and G. Wyeth: *Thinking as One: Coordination of Multiple Robots by Shared Representations*. Proceedings of the IEEE International Conference on Intelligent Robots and Systems, vol. 2, pp. 1391-1396, 2000.
31. A. Tews and G. Wyeth: *Multi-Robot Coordination in the Robot Soccer Environment*. Proceedings of the Australian Conference on Robotics and Automation (ACRA '99), March 30 - April 1, Brisbane, pp. 90-95, 1999.
32. B.J. Young, R.W. Beard, J.M. Kelsey: *Coordinated Control of Multiple Robots using Formation Feedback*. IEEE Transactions on Robotics and Automation, In Review.