

# M-ROSE: A Multi Robot Simulation Environment for Learning Cooperative Behavior

Sebastian Buck, Michael Beetz, and Thorsten Schmitt

Munich University of Technology, Department of Computer Science IX,  
Orleansstr. 34, D-81667 Munich, Germany

**Abstract.** The development of high-performance autonomous multi robot control systems requires intensive experimentation in controllable, repeatable, and realistic robot settings. The need for experimentation is even higher in applications where the robots should automatically learn substantial parts of their controllers. We propose to solve such learning tasks as a three step process. First, we learn a simulator of the robots' dynamics. Second, we perform the learning tasks using the learned simulator. Third, we port the learned controller to the real robot and cross validate the performance gains obtained by the learned controllers. In this paper, we describe M-ROSE, our learning simulator, and provide empirical evidence that it is a powerful tool for learning of sophisticated control modules for real robots.

## 1 Introduction

The development of high-performance autonomous multi robot systems requires intensive experimentation in controllable, repeatable, and realistic robot settings. The need for experimentation is even higher in applications where the robots should automatically learn substantial parts of their controllers. For example, to improve the competence of our robot soccer team we have learned a model for predicting the time needed to perform a given single robot navigation task. To learn such a model with an expected inaccuracy of less than three percent we had to collect data from more than 4600 navigation episodes. Assuming that setting up and executing a navigation task takes only two minutes, we would have had to spend more than 150 hours of experimentation with the real robots. Obviously, this is not feasible.

We therefore propose to solve such learning tasks as a three step process. First, we learn a simulator of the robots' dynamics. Second, we perform the learning tasks using the learned simulator. Third, we port the learned controller to the real robot and cross validate the performance gains obtained by the learned controllers. Using this method we can perform the learning task mentioned above by collecting data from the real robots only for 6 hours and then perform the learning task in 3 hours of simulation.

In this paper, we describe M-ROSE, our learning simulator, and provide empirical evidence that it is a powerful tool for the learning of complex control modules for real robots.

Robot simulation in general is a powerful tool for the development of autonomous robot control systems because it allows for fast and cheap prediction and makes experiments controllable and repeatable. Simulation allows for the quick detection and diagnosis of software errors.

The use of robot simulators also yields a number of problems. For simulation purpose time must often be discretized. Also, simulators typically work in an abstract feature space and might therefore ignore key factors for the robot behavior [11]. Others argue that simulated controllers are doomed to succeed because of the design of the simulators [3]. As a consequence, software that succeeds in simulation may fail on a real robot [4]. Accurate and numeric simulations are typically extremely complex and expensive in terms of computational resources and are thus performed by parallel and distributed simulation [9,13]. Despite these problems a number of simulators have been proven to be valuable resources in the development of robot controllers [8].

Multi robot simulation includes the simulation of sensing as well as the simulation of motion dynamics. The simulation of motion maps the dynamic state of the robot and a low level robot command into a successor state. Sensor simulation maps the local surroundings and the sensor model into the sensed data. A substantial amount of work has been done on either one or both of these aspects: Lee et al. [12] generates an artificial neural network based model of the environment within a simulator of a Nomad robot, to learn an action model in a MDP framework simulators have been used [2], and many experiments of the successful well known rhino robot [18] were done in simulation. Furthermore the soccerserver [14] used in the RoboCup Simulation League mimics some sensory and motion abilities of a human-like soccer player. Another RoboCup simulator developed [10] is able to simulate a large number of different sensors (infrared, bumper, camera, and laser) while motion is simulated by directly using the values of translational and rotational control commands to compute a new state.

While most of the above work concentrates on the simulation of sensors while more or less neglecting the motion our main goal in this paper is an accurate simulation of the robots' dynamical behavior which becomes very important in high-speed environments such as autonomous robot soccer. In the RoboCup MidSize League abrupt changes in speed and direction are as common as they are in a real soccer game. Our approach to simulating changes in state in robotics involves neural learning from real robot data and is, as we will show, easy extendable and highly qualified because of its brilliant accuracy.

The remainder of this paper is organized as follows: In section 2 we will describe our motion (2.1) and sensory model (2.3) followed by statistics to document the accuracy (2.5). Section 3 shows the results of some successful experiments done with real soccer robots whose behavior has been implemented and learned in the simulator. Finally section 4 concludes.

## 2 Building a Multi Robot Simulation Environment

In addition to an accurate modeling of sensors and motion a multi robot simulator should be able to simulate different kinds of robots with their respective motion profile acting at the same time. A model of the robots' static environment as well as models of dynamic objects should be easy to integrate. Furthermore it is essential that a high number of learning data can be obtained in a reasonable period of time.

The main difference between our simulator and the RoboCup soccerserver [14] is that we simulate a team of real robots while soccerserver simulates a hybrid, in some aspects human-like, soccer player. Our team (the *AGILO RoboCuppers* [1]) consists of four Pioneer I robots [15] who obviously have several disadvantages compared with an agent of the soccerserver: They cannot hold the ball if it does not come directly into their ball-guiding device and path planning with real robots becomes more complicated than in the soccerserver where the field is about 105 times 70 meters and a player has a diameter of only 0.3 meters. Moreover the ball can go through a player in soccerserver (if it is fast enough) and can be given a velocity vector by a player having it in its kicking range. Teams like the *Karlsruhe Brainstormers* [16] have shown that in the soccerserver environment learning algorithms perform excellent and a lot better than a human being controlling a player with a joystick.

In addition to our Pioneer robots we started to extend our simulator by integrating a model of our B21 robots which we use for indoor exploration tasks. Before we describe the components of our simulator in detail we briefly point out the major advantages of M-ROSE:

- The individual dynamic motion profile of a robot can be learned.
- The simulator is easy extendable and can combine the simulation of robots with different dynamic motion profiles.
- Models of other objects (static and dynamic) or humans can easily be integrated.
- Powers are modeled according to their physical rules.
- Step by step analysis and monitoring supports the development of control software.
- Time lapse allows performing a great number of experiments in exiguous time.
- The control software can either be linked with the simulator core or the real robot.

### 2.1 Learning Models of Dynamic Behavior

In this section we describe how to learn a dynamic model of the motion behavior of a Pioneer I robot. Models can similarly be learned for other robots with the respective data at hand.

The dynamic state of the robot at a certain time  $i$  is given by the quintuple

$$\zeta_i = \langle x_i, y_i, \varphi_i, V_{tr_i}, V_{rot_i} \rangle \quad (1)$$

where  $x_i$  and  $y_i$  are coordinates in a global system,  $\varphi_i$  is the orientation of the robot and  $V_{tr_i}$  ( $V_{rot_i}$ ) are the translational (rotational) velocities. The robot control system issues driving commands  $\xi = \langle V_{tr}, V_{rot} \rangle$ . The dynamic model for the change in state from the current state  $\zeta_c$  to the successor state  $\zeta_{succ}$  used by the simulator is acquired by learning the mapping

$$\Delta : \zeta_c \times \xi \mapsto \zeta_{succ} \quad (2)$$

from experience, that is recorded data from real robot runs. Our simulator learns this mapping using a standard multi layer neural network [7] with one hidden layer and supervised learning with the RPROP algorithm [17]. Considering the current state  $\zeta_c$  at  $x_c = 0, y_c = 0$ , and  $\varphi_c = 0$  in a local system we can reduce the input dimension to 4 by converting the successor state's ( $\zeta_{succ}$ 's)  $x_{succ}, y_{succ}, \varphi_{succ}$  into that local system (that means we regard  $\Delta x, \Delta y, \Delta \varphi$  instead of  $x_{succ}, y_{succ}, \varphi_{succ}$ ):

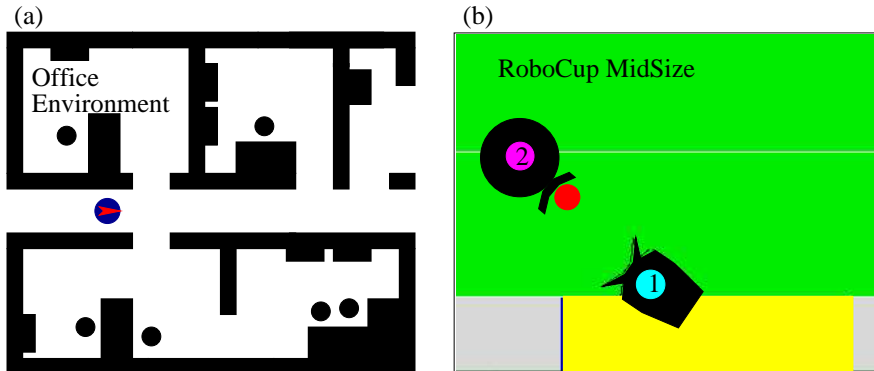
$$\Delta : \langle V_{tr_c}, V_{rot_c}, V_{tr}, V_{rot} \rangle \mapsto \langle \Delta x, \Delta y, \Delta \varphi, V_{tr_{succ}}, V_{rot_{succ}} \rangle \quad (3)$$

Using this mapping we have learned the dynamics of Pioneer I robots. During data acquisition we have executed a wide variety of navigation scenarios covering even abrupt changes in velocity and orientation to comply with the requirements of high-speed navigation. We have collected a total of more than 100.000 training patterns from runs with a real Pioneer I robot. Although the learning time took a few hours on a 800 MHz machine the computational amount of the neural network while running the simulation is infinitesimal.

## 2.2 Considering Physical Properties

All robots are dealing more or less with a dead time which means that at the moment a robot is performing a low level control command it has already sent control commands (e.g. translational and rotational target velocity) for further movements. In the case of Pioneer I robots the dead time is around 300 ms while the controller accepts ten commands per second. To consider the dead time M-ROSE writes low level control commands in a queue and waits a certain time before executing the commands.

**Shapes of Objects and Environment** The shapes of robots and any other objects are modeled as polygons or circles (see fig. 1). The corners of the polygon (radius of the circle) according to the objects reference point are read from a shape-file. A Pioneer I robot is described by a polygon while a B21 robot or a ball are regarded as a circle. The advantage of polygons and



**Fig. 1.** Subfigure (a): Simulation of a B21-robot. All objects are drawn as filled polygons or circles. The direction of the robot is indicated by an arrow. Subfigure (b): Simulation of a RoboCup game with a Pioneer and a Nomad-shaped robot.

circles is that physical laws of nature can easily be applied to simulate for instance the collision with a ball. However, one can go far deep into physical simulation but we just consider the basic laws for collision and friction with a tunable factor of random.

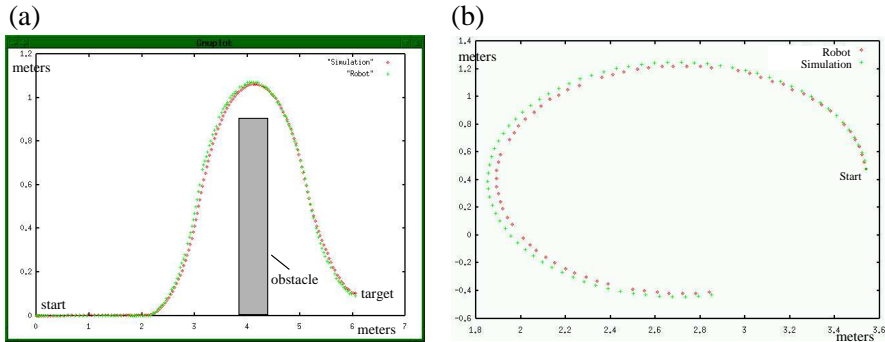
Furthermore M-ROSE can simulate a manipulator on a robot interacting with an object. Currently this is done by analytic computation (position, direction, and power of the manipulator must be specified in the respective configuration file). But similar to the robots' dynamics the movements of the manipulator can be learned. For the description of the environment another configuration file similar to the shape-file for robots exists.

### 2.3 Simulation of Perception

The sensor model used in our RoboCup simulations so far is a very simple one: The robot receives his current state data with some simulated noise. Doing so we assume that the sensors of our robots are almost perfect which obviously is a wrong assumption. Therefore we are currently working on a more realistic approach: In several RoboCup games we have recorded the “*true*” state of a robot  $\zeta$  (given by a camera mounted at the ceiling of the room) and the sensory data vector  $\mathcal{Y}$  in parallel. To achieve a realistic simulation of the sensors we will learn the mapping

$$\mathcal{I} : \zeta_c \mapsto \mathcal{Y}_c \quad (4)$$

from a current robot state  $\zeta_c$  in the simulator to the current sensory data vector  $\mathcal{Y}_c$ . Experiments will show if  $\mathcal{I}$  is best represented by an interpolating lookup-table, decision trees or neural networks trained with the recorded data.



**Fig. 2.** A comparison between the position data of a real robot and the data simulated by M-ROSE. Subfigure(a): A robot driving a jink around an obstacle. Subfigure(b): A robot driving an ellipse.

## 2.4 Extensibility

The M-ROSE system in general is able to simulate different kinds of robots. Each robot to be simulated can be specified in a configuration file which contains information about the names of the shape-file and the network-file (which simulates the motion) of the robot. In future information about the robots sensors (and  $\mathcal{I}$  in particular) will be included. The number of robots possible to be simulated is infinite but, however, a large number of robots (dependent on the computational power) will degrade the systems performance with respect to real time simulation and time lapse.

## 2.5 Accuracy

Evaluating around 100 trajectories each ten seconds in length driven with a real Pioneer I robot and *not* used for learning we found an average error of 2% in the simulation of position and orientation. We believe that this error largely results from the inherent indeterministic behavior of the robot controller. This indeterministic behavior itself usually leads to an average error close to one percent. After extreme navigation situations where the translational velocity was set from full to zero and the rotational velocity was set from zero to full (or both the other way around) sometimes an error of up to 10% in orientation (position) occurred. These cases are probably the hardest to predict at all and not very common in real robot navigation in general. Figure 2 shows real robot and predicted data of the simulator in two prevalent trajectories. Obviously the match of predicted and real robot data looks satisfying. The remaining small errors in simulation are balanced by the control software running at a frequency of 10Hz in real time.

### 3 Simulation Based Behavior Learning

After describing the simulation environment and regarding its accuracy we now want to present some demonstrations of successful robot behavior which was implemented in M-ROSE and works well on real robots. The examples cover cooperative tasks (coordination of actions in robot soccer, path planning) as well as a difficult manipulation task.

To demonstrate the successful behaviors we give statistical results and additionally provide robot videos on our web page. So far all examples are related to RoboCup.

#### 3.1 Demonstration 1: Learning coordination

The first example shows how a cooperative behavior learned in M-ROSE works on real robots. One of the key issues in RoboCup is the cooperative coordination of the robots' actions. Part of it is the decision which robot of the team is to go for the ball. Considering the current dynamic states of all robots of one team as well as orientation and velocity of the target state near the ball, and additionally, the dynamic configuration of obstacles this is no simple decision. To support this decision we have implemented a neural projection  $\mathcal{P}$  which maps start ( $\zeta_s$ ) and target state ( $\zeta_t$ ) of a robot to the estimated time need to reach the target state considering the configuration of obstacles and given a dynamic model of the robots motion (see [5] for details):

$$\mathcal{P} : \zeta_s \times \zeta_t \mapsto \textit{time} \quad (5)$$

The robot  $r_f$  assumed to be the fastest at the ball (of all robots  $\mathcal{R}$ ) is chosen to go there:

$$r_f = \operatorname{argmin}_{r_i \in \mathcal{R}} \mathcal{P}(\zeta_s(r_i), \zeta_t) \quad (6)$$

This estimation is done by every single robot and consistent (1) in the simulator (because the robots got information about their *true* states) and (2) to a high extent in the real robot environment (because of our accurate visual localization system). As mentioned in the introduction the amount for data acquisition and learning was 9 hours while learning without simulation would have taken more than 150 hours.

**Results in a real robot environment** To demonstrate the quality of the coordination method implemented in M-ROSE we measure the number of robots going for the ball at the same time. Further we observe how long the same robot goes for the ball without any other robot going for the ball. 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 went for the ball.

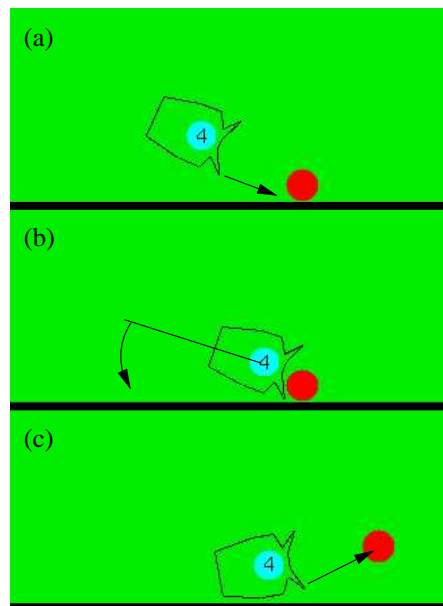
#robots going for ball at the same time	quota in relation to the whole time played
0	00.34%
1	98.64%
> 1	01.02%

The average time one robot goes for the ball or handles the ball without being interrupted by a decision of a fellow robot is 3.35 seconds. Videos showing typical decision situations are available at <http://www9.in.tum.de/people/buck/videos.html>.

### 3.2 Demonstration 2: Learning to get the ball

Another challenging task in RoboCup so far is to remove a ball from the wall. According to the rules of the RoboCup MidSize League it is not permitted to grip or fix the ball. Thus the ball has to be removed from the wall by carefully touching it without maneuvering the robot in a stuck situation at the wall. Most of the teams participating in the RoboCup MidSize League were not able to remove a ball from the wall in a controlled way by 2001. Our approach to that task is hand-coded and implemented in M-ROSE.

All experiments and tests to improve the performance and perfect the ball handling at the wall were done in the simulation environment. The software that we have developed in M-ROSE worked instantly on a real robot without any further adjustments. The hand-coded method to remove the ball from the wall is based on the simple idea to first approach the ball (see figure 3(a)) and then stop directly before it (see figure 3(b)) and turn rapidly towards the center of the field (see figure 3(c)). Thereby the ball should be touched by the outer part of the robots kicking device to move it inside the field.



**Fig. 3.** A simple hand-coded method to remove a ball from a wall. The ball is moved by an impulse of a rapid turn.

**Real robot Behavior** So far there are no detailed statistics on removing the ball from the wall but in general it works on real robots. There are videos available at <http://www9.in.tum.de/people/buck/videos.html> showing typical situations where real robots take a ball from the wall in a controlled way.

### 3.3 Demonstration 3: Learning to choose the best planning method

Another software module developed in M-ROSE is the path planner of our RoboCup team. In this work, we propose to select problem-adequate navigation planning methods based on empirical investigations, that is the robots should learn by experimentation (which is done in the simulator) to use the fastest planning methods. The robots have learned predictive models for the performance of different navigation planning methods in a given application domain. A hybrid planning method that selects planning methods based on a learned predictive model outperforms the individual planning methods (statistics and detailed description can be found in [6]). All the learning data was generated in M-ROSE and the hybrid planning method was implemented in the simulator as well.

## 4 Conclusions

In this paper we propose a simulation environment for multi robot application domains. The main contributions are an accurate, robot specific neural simulation of robot motion and an easy extendable software environment. The dynamic behavior is simulated by a neural prediction of changes in state according to certain low level control commands.

The M-ROSE simulator addresses many key aspects of real robot behavior including accurate motion models, dead times, collisions of objects, powers of manipulators, shapes of objects, and extensibility. In contrast to most of the previous work the main focus is set on the simulation of motion which is most critical in high-speed scenarios such as RoboCup games. An accurate simulation of the sensors is considered in our ongoing research and is expected to greatly broaden the learning tasks that can be tackled and to improve the learning competence of M-ROSE for real robots. The examples in section 3.1, 3.2, and 3.3 have shown that control software hand-coded as well as learned in M-ROSE already leads to more than acceptable results on real robots which we believe is to a large extent founded in the accuracy of the simulator. While so far extensive simulations have been running with Pioneer I robots only the motion behavior of other robots can and will be learned analogously.

## References

1. 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 (AAMAS) 2002.
2. T. Belker and M. Beetz: *Learning To Execute Navigation Plans* in F. Baader, G. Brewka and T. Eiter (eds): *Lecture Notes in Artificial Intelligence*, vol. 2174.
  3. R.A. Brooks and M.J. Mataric: *Real Robots, Real Learning Problems*, in *Robot Learning*, Jonathan H. Connell and Sridhar Mahadevan, eds., Kluwer Academic Press, 193-213, 1993.
  4. R.A. Brooks: *Artificial Life and Real Robots*, in F. J. Varela and P. Bourguine, editors, *Proceedings of the First European Conference on Artificial Life*, pp.3-10, 1992.
  5. S. Buck, T. Schmitt, and M. Beetz: *Reliable Multi Robot Coordination Using Minimal Communication and Neural Prediction*, Seminar on Plan-based Control of Robotic Agents 2001, Schloss Dagstuhl, *Lecture Notes in Artificial Intelligence*, 2001, Springer Verlag.
  6. 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.
  7. J. Hertz, A. Krogh, and R. G. Palmer: *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.
  8. N. Jakobi, P. Husbands and I. Harvey: *Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics*, *Third European Conference on Artificial Life (ECAL95)*, pp.704-720, Springer Verlag, 1995.
  9. M.L. Jugel and A. Sydow: *Parallelity in High-Level Simulation Architectures*, in *Transaction of the Society for Computer Simulation International*, Vol. 15, No. 3, pp.101-103, 1998.
  10. H.-U. Kobialka, P. Schoell, and A. Bredenfeld: *Tools for Assessing RoboCup Behavior* *RoboCup Workshop, RoboCup-Euro 2000*, Amsterdam, May 28th - June 2nd, 2000
  11. T. Lee, U. Nehmzow, and R. Hubbard: *Computer Simulation of Learning Experiments with Autonomous Mobile Robots*, *Proceedings of TIMR 99, Towards Intelligent Mobile Robots*, Bristol, 1999.
  12. T. Lee, U. Nehmzow, and R. Hubbard: *Mobile Robot Simulation by Means of Acquired Neural Network Models*, *European Simulation Multiconference*, Manchester 1998.
  13. U. Mehlhaus and W. Rausch: *Distributed simulation of robot tasks*, *ESS'93 European Simulation Symposium*, pages 433-438, Delft, Holland, October 1993.
  14. 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.
  15. Pioneer Mobile Robots, *Operation Manual*, 2nd edition, Active Media, 1998.
  16. M. Riedmiller and A. Merke: *Karlsruhe Brainstormers - a reinforcement learning approach to robotic soccer II*. In *5th International Workshop on RoboCup*, *Lecture Notes in Artificial Intelligence*, 2001, Springer Verlag.
  17. M. Riedmiller and H. Braun: *A direct adaptive method for faster backpropagation learning: the Rprop algorithm*, *Proceedings of the ICNN*, San Francisco, 1993.
  18. S. Thrun, A. Bucken, W. Burgard, D. Fox, T. Frohlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt: *Map learning and high-speed navigation in RHINO* MIT/AAAI Press, Cambridge, MA, 1998.