Autonomous Environment and Task Adaptation for Robotic Agents

Michael Beetz and **Thorsten Belker** [1]

Abstract. This paper investigates the problem of improving the performance of general state-of-the-art robot control systems by autonomously adapting them to specific tasks and environments. We propose ***model- and test-based transformational learning (*MTTL*)*** as a computational model for performing this task. MTTL uses abstract models of control systems and environments in order to propose promising adaptations. To account for model deficiencies resulting from abstraction, hypotheses are statistically tested based on experimentation in the physical world.

We describe XFRMLEARN, an implementation of MTTL, and apply it to the problem of indoor navigation. We present experiments in which XFRMLEARN improves the navigation performance of a state-of-the-art high-speed navigation system for a given set of navigation tasks by up to 44 percent.

1 Introduction

The use of general problem-solving mechanisms enables autonomous robots to deal with unexpected and new situations, which—sooner or later—occur in any complex and changing environment. Unfortunately, the application of general problem-solving mechanisms must often be paid for with a loss of performance compared to highly specialized mechanisms. A promising approach to achieve both the performance of specialized mechanisms and generality is to equip the agents with a learning component that enables the agent to improve its performance by autonomously adapting to its environments and typical tasks.

Indoor navigation for autonomous robots provides a good case in point. So far research on indoor robot navigation has mainly focussed on the development of general, reliable control methods that are optimized for average performance. Safe high-speed navigation systems have been developed by different research groups and tested over extended periods of time [**?**]. Architectures that have been successfully employed include those that combine decision-theoretic path planning techniques with reactive execution systems [**?**]. These architectures also have the advantage using general, concise, and well-understood control algorithms with proven properties that optimize average performance.

Still, various system parameters must be adapted for the general systems to exhibit satisfactory performance in new operating environments. Even worse, it is often necessary to adjust the parameters differently for different tasks and areas of the environment. So far this parameter tuning has been mainly performed by human programmers. Unfortunately detecting situations which are not handled appropriately, finding appropriate parameterizations, and testing alternative parameterizations is very tedious and requires an enormous amount of experimentation with the physical robot.

In this paper we investigate how a robotic agent using such general algorithms can autonomously adapt the parameterizations of its algorithms in order to significantly and substantially improve its performance for a given environment and a specific set of navigation tasks. We bias the adaptation process to avoid frequent parameter changes in order to keep the behavior more predictable and stable.

The contributions of this paper are threefold. First, we describe a computational method for model-based adaption of general robot navigation methods to a specific environment and task. Second, we describe a framework that allows for a concise specification of heuristic rules for detecting improvable behavior and revising behavior specifications to improve performance. Third, we investigate how a robotic agent can empirically decide whether or not adaptations improve its performance.

This paper presents XFRMLEARN, which is an implementation of MTTL for SRCs [**?**], a specific kind of robot control architecture and the RHINO navigation system [**?**]. We present experiments in which XFRMLEARN improves the performance of the RHINO navigation system, a state-of-the-art navigation system, for given navigation tasks by up to 44 percent within 6 to 7 hours. XFRMLEARN has autonomously operated an RWI B21 autonomous mobile robot for more than 100 hours in an unprepared office environment.

2 XFRMLEARN

This Section describes the operation of XFRMLEARN by applying it to the adaptation of the RHINO navigation system, which has shown impressive results in several longterm experiments (e.g., [**?**]). Conceptually, the RHINO navigation system works as follows. A navigation task is transformed into a Markov decision problem and solved by a path planner using a value iteration algorithm [**?**]. The solution is a policy that maps every possible location into the optimal heading to reach the target. This policy is then given to a reactive collision avoidance module that executes the policy taking the actual sensor readings (and thereby unmodelled obstacles) and the dynamics of the robot into account [**?**].

RHINO's navigation behavior can be improved because the path planner solves an idealized problem that does not take the desired velocity, the dynamics of the robot, and dynamic obstacles into account. The reactive collision avoidance component takes these aspects into account but makes only local decisions. Neither component deals with sensor crosstalk, which causes the robot's sonar sensors to hallucinate obstacles.

The RHINO navigation system can be parameterized in different

[1] University of Bonn, Dept. of Computer Science III, Roemerstr. 164, D-53117 Bonn, Germany, email: {beetz,belker}@cs.uni-bonn.de.

ways. One can constrain the navigation path to be taken for achieving a given navigation task by specifying a sequence of intermediate points which are to be visited in the specified order. In addition, a travelmode parameter determines how cautiously the robot should drive and how abruptly it is allowed to change direction. Finally, one can specify the sensors to be used for obstacle detection.

The robot controller with XFRMLEARN in operation works as follows. While the robot is busy, XFRMLEARN passively estimates and monitors the pose of the robot, the translational and rotational velocity and stores the measured values of these quantities together with a time stamp in a behavior trace. Figure **??** depicts such a behavior trace as a sequence of circles. The position of the circles represent the robot's position, the circle's size is proportional to the translational speed, and the circle's darkness to the rotational speed.

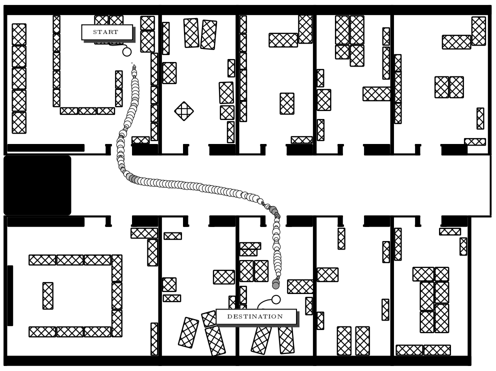

Figure 1. An exemplary behavior trace.

When RHINO is idle, XFRMLEARN is started and executes a loop with four steps: ***detect***, ***diagnose***, ***revise***, and ***test***. In the ***detect*** step XFRMLEARN selects an "interesting" behavior trace together with the navigation plan that has produced the trace. XFRMLEARN then looks for conspicious behavior patterns in this trace, which are often hints for behavior flaws. The ***diagnose*** step tries to generate explanations for the conspicious behavior patterns on the basis of XFRMLEARN's models. The result of the diagnosis step is used to index promising revisions. The ***revise*** step applies the revision that is expected to produce the highest performance gain. Finally, in the ***test*** step XFRMLEARN runs a series of experiments to test the hypothesis that the new plan generates behavior that is significantly better than that generated by the original one.

XFRMLEARN uses ***structured reactive parameterizations (SRPAs)***, concurrent reactive control plans that specify constraints for the robot's behavior and parameterize the low-level control system in a situation- and context-specific way. SRPAs specify how the navigation behavior is to be adapted when certain triggering conditions become true and which intermediate locations the robot should head to as it performs the task [**?**]. We will use the following syntax for SRPAs:

$$\frac{\underline{\text{with behavior adaptation}}}{\text{GO-PATH}\ (\ \textit{path}\ )}\quad \textit{beh-spec}^*$$

where *beh-spec* has the form <u>whenever</u> *c*: *p* ← *v*, where c is a triggering condition, p is a parameter of the subsymbolic navigation system, and v is the value p is to be set to. *path* is a sequence of points.

The SRPA specifies that the robot should take a path through the intermediate destinations given by *path*. Concurrently, the controller runs the guarded behavior adaptations *beh-spec*. Thus, whenever the state of the robot, in particular its position and orientation, changes such that a triggering condition *c* of an adaptation becomes true, then the navigation system is parameterized as specified by *p* ← *v*.

2.1 The Learning Step "Detect"

The "detect" step selects the SRPA that with respect to XFRMLEARN's experience promises the highest performance gain for the next learning iteration. XFRMLEARN then selects a behavior trace that has been generated by this SRPA. This trace is analyzed to detect stretches of conspicious behavior that we call *conspicious behavior modes*. A conspicious behavior mode is a subtrace of a behavior trace that substantially deviates from the expected behavior.

2.2 The Learning Step "Diagnose"

In the "diagnose" step the conspicious behavior modes are causally explained, if possible. Those conspicious behavior modes that can be causally explained are called *behavior flaws*. The performance gain that could be achieved by eliminating each flaw is estimated and stored in the description of the flaw. The information can be used to determine the utility of a revision step (see Section **??**). The flaws are diagnosed based on a world model together with a model of the low-level navigation system.

Models for the Diagnosis Step. Knowledge about the environment is a powerful resource for tailoring the navigation behavior of the robot to the particular navigation tasks at hand. In this paper we enrich the general environment model, which contains knowledge about obstacles etc., with additional knowledge about office environments. Environments such as office buildings are functionally designed. Within offices, the paths to the desks are usually kept free and so are hallways. Doors separate the working areas, like offices, from more crowded areas, like hallways.

XFRMLEARN's knowledge about RHINO's environment is represented in a symbolically annotated 3D model [**?**], which specifies the position and orientation of obstacles as well as their type and function. The world model enables XFRMLEARN to draw inferences like the following ones: what is the closest obstacle to the left of the robot? How would the collision avoidance module probably perceive the local surroundings at position $\langle x, y\rangle$?

To adapt the navigation behavior effectively, XFRMLEARN is also equipped with causal and functional knowledge about RHINO navigation system that is encoded into diagnostic rules. We will explain these rules next.

Diagnostic Rules. XFRMLEARN's diagnostic rules used in this paper are rules that classify conspicious navigation behavior as being caused by close obstacles (diagnostic rule **D-1**), narrow passages (**D-2**), low target velocity (**D-3**) and high target velocity (**D-4**), and sonar cross talk (**D-5**). Exploiting functional knowledge allows us to formulate more specific rules. For example, a conspicious behavior could be caused by a doorway, which is a specific narrow passage that often causes cross talk (**D-6**). In general, such specializations of diagnostic rules define taxonomies of behavior flaws. Note, the diagnosis rules are hand coded and heuristic. There are, of course, approaches to learning these classification rules such as learning from examples classified by experts or performing data mining in richer behavior traces. Learning the classification rules, however, is beyond the scope of this paper.

Let us discuss a diagnostic rule for the following conspicious behavior. The robot stops in the midst of carrying out a navigation task while traversing a doorway. A possible explanation that XFRMLEARN generates for this kind of behavior is that the sonar sensor data are corrupted by "cross talk," which causes the robot to hallucinate obstacles in front of itself. Because in narrow passages the robot cannot circumnavigate obstacles, the collision avoidance module stops the robot.

```
if      BehaviorMode(Stops,?srpa,?subtrace)
        ∧ InNarrowPassage(?subtrace,?percentage)
        ∧ ?percentage > 50%
then  infer  BehaviorFlaw with
                CATEGORY      = StoppingCausedByCrossTalk
                SRPA          = ?srpa
                SUBTRACE      = ?subtrace
                SEVERITY      = severity(?subtrace,?srpa,
                                         ?percentage,
                                         StoppingCausedByCrossTalk)
```

Figure 2. Diagnostic rule **D-5** for diagnosing behavior flaws as caused by sonar cross talk.

Figure **??** shows a simple and general rule for diagnosing these kinds of behavior flaws (rule **D-5** above). This rule explains the stopping of the robot during the traversal of a narrow passage (at least 50% of the conspicious subtrace has to lie in the narrow passage) with the occurrence of sonar cross talk. If the diagnostic conditions hold then a behavior flaw description is returned. The behavior flaw description also contains an estimate of the flaw's severity, which in the case of cross talk is the time spent standing at one place as opposed to traverse the narrow passage with the velocity specified in the SRPA.

The diagnostic rules are applied to all conspicious subtraces and the resulting behavior flaw descriptions are collected.

2.3 The Learning Step "Revise"

The "revise" step uses heuristic programming knowledge in the form of transformation rules to propose promising revisions of the given SRPA. XFRMLEARN selects revisions with a probability proportional to their expected utility to generate a new candidate SRPA. This selection mechanism trades off the exploitation of previous experiences and the exploration of alternative revision strategies.

Transformation Rules The programming knowledge is abstract and declarative knowledge about how SRPAs are to be revised to avoid specific kinds of behavior flaws.

```
transformation rule  switch-off-sonars-to-avoid-cross-talk
to eliminate  ?behavior-flaw with
                  CATEGORY    = StoppingCausedByCrossTalk
                  SRPA        = ?srpa
                  SUBTRACE    = ?subtrace
   if  CrossesNarrowPassageRegion (?subtrace,?region)
   then  with expected utility
             = expected-utility (switch-off-sonars-to-avoid-cross-talk,
                                 ?behavior-flaw)
         insert reactive parameterization
             WHENEVER entering-region?(?region) : SONAR ← OFF
```

Figure 3. Transformation rule **R-2**.

Fig. **??** shows a transformation rule indexed by a behavior flaw *"StoppingCausedByCrossTalk"*. The rule is applicable if the subtrace *?subtrace* traverses a narrow passage *?region*. The application inserts a reactive parameterization into the SRPA that upon entering the narrow passage will switch off the sonar sensors. The rule contains a term for estimating the expected utility of the revision (see below).

For the purpose of this paper, XFRMLEARN provides the following revisions:

R-1 If the behavior flaw is attributed to the traversal of a narrow passage (**D-2**) then insert additional intermediate points into the topological component of the SRPA that cause the robot to traverse a narrow passage orthogonally and with maximal clearance.

R-2 Switch off the sonar sensors in narrow passages to avoid hallucinating obstacles due to sonar cross talk (indexed by **D-5**).

R-3 Insert an additional intermediate point into the topological component of the SRPA in order to pass a closeby obstacle with more clearance (indexed by **D-1**).

R-4 Increase the target velocity in sparsely cluttered areas where the measured velocity almost reaches the current target velocity (indexed by **D-3**).

R-5 Insert an additional intermediate point into the topological component of the SRPA to avoid changes in the robot's direction that are too abrupt (indexed by **D-2**).

R-6 Apply **R-2** and **R-1** simultaneously (indexed by **D-6**).

Estimating the Expected Utility of Revisions. We have seen above that XFRMLEARN's transformation rules are fairly general. This has an important consequence. The expected utility of rules varies from environment to environment and task to task. Therefore an important aspect in behavior adaption is to learn the environment and task specific expected utility of rules based on experience.

The expected utility of a revision r given a behavior flaw b is computed by *EU(r|diagnosed(b)) = P(success(r)|correct(b)) * s(b)*, where b is a diagnosed behavior flaw, $s(b)$ is the severity of the behavior flaw (the performance gain, if the flaw would be completely eliminated), $P(success(r)|correct(b))$ is the probability that the application of revision r improves the robot's performance, given that the diagnosis of b is correct. For computing the probability $P(success(r)|correct(b))$ XFRMLEARN maintains a simple statistic about successful and unsuccessful rule applications.

In general, the first equation overestimates the expected utility of a revision but tends to select, with respect to our practical experience, the same revisions that programmers propose when visually inspecting the behavior traces. The estimate of the expected utility of the second most promising revision is used as an estimate of how interesting the current navigation task or the SRPA for this task respectively is for future examinations by XFRMLEARN.

2.4 The Learning Step "Test"

Because of the heuristic nature of the transformation rules, their success has to be empirically tested. To perform this test, XFRMLEARN repeatedly and alternatingly executes the original SRPA p and the new candidate SRPA $\hat{p}$ proposed in the revision step to compare their performance. After k experiments (we usually use $k = 7$) the test component decides whether or not to substitute the original SRPA by the new candidate SRPA in the library. The new candidate SRPA is accepted, if the *t-test* based on the experiments results in a 95% confidence that the new SRPA performs better than the original one. In order to perform the statistical significance test we have also applied a *bootstrapping t-test* (see [**?**], chapter 4) with a rejection level of 5%. Unlike the t-test, the bootstrap test does not assume that the sampling distributions are roughly normal and have the same variance. Surprisingly though our experiences show that the t-test produces almost identical results compared to the bootstrapping t-test.

3 Experimental Results

We have performed two extended learning sessions for one navigation task respectively. In both learning sessions XFRMLEARN improved the average navigation behavior for the given task successfully by 31% and 44%. There is a 95% probability that the performance gain has been at least 21% and 18%, respectively. In the second learning session, besides the average performance the standard deviation has been significantly reduced.

We consider these performance gains as being very impressive, for several reasons. First, XFRMLEARN has only used observable symptoms such as driving slowly or fast. It did not have access to the execution states of the collision avoidance module which would have provided more informative and reliable evidence for the diagnosis of behavior flaws. Second, the transformation rules are quite general and applicable to many indoor environments. They are also not specifically tailored for this particular system. Third, we have applied XFRMLEARN to a low-level navigation system which is well tuned and has been developed in the same environment. We expect the performance gains for insufficiently configured navigation systems to be much higher.

The Figure **??** summarizes one of the two learning sessions. In this learning session XFRMLEARN has tried six revisions, four of them were successful. Fig. **??** shows the original (empty) SRPA and the final SRPA for the navigation task and two typical behavior traces. A brief visual inspection shows that the learned SRPA runs smoother and much faster in the hallway. The destination office is entered in an obtruser angle. The transformations that were considered in the learning cycles are visualized in Fig. **??**(upper right).

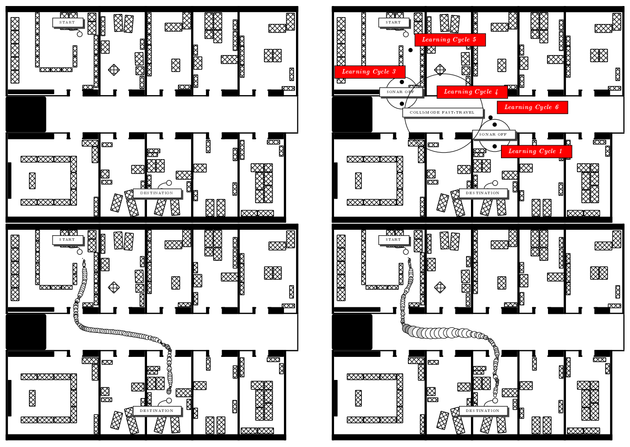

Figure 4. The original (left) and the learned plan and two behavior traces.

To compare the performance of the original and the learned SRPA we have performed an experiment in which we performed the navigation task with each SRPA eleven times. Fig. **??** contains the measured times (left) and the most important values of the descriptive statistics. All characteristic numbers of the learned SRPA are substantially lower. The average time needed for performing a navigation task was reduced by 93.57 seconds, which corresponds to a reduction of about 44 percent. The probability that the duration has been reduced is 0.9966. The t-test for reducing the navigation time by at least 39 seconds has a significance of 0.952.

DURATION			
ORIGINAL		LEARNED	
156.8	160.5	114.5	102.6
265.9	148.5	184.5	97.9
487.6	195.3	108.6	138.3
167.4	145.3	95.6	93.6
215.5	214.3	128.8	86.1
181.5		159.0	

STATISTICS	ORIG.	LRND.
Minimum	145.3	86.1
Maximum	487.6	184.5
Range	342.2	98.4
Median	181.5	108.6
Mean	212.6	119.0
Variance	9623.1	948.0
Std dev	98.0	30.7

Figure 5. The durations for carrying out the navigation task with the original and the learned SRPA (left). The descriptive statistics for the experiment (right).

In the second session we posed a navigation task to go from one room into an adjacent one, which means that no big performance gains could be obtained by simply increasing the speed in the hallway. For this task, XFRMLEARN has proposed and tested five plan revisions, three of them were successful. The average time needed for performing the navigation task was reduced by 35.3 seconds, which corresponds to a reduction of about 31 percent. The t-test for the learned plan being at least 24 seconds (21%) faster returns a significance of 0.956. A bootstrap test returns the probabilty of 0.956 that the variance of the performance has been reduced.

4 Discussion

In this section we discuss XFRMLEARN with respect to generality issues and its techniques to deal with high variances of robot behavior.

Generality. XFRMLEARN provides a framework in which heuristic rules for detecting improvable behavior and revising behavior specifications to improve performance can be specified concisely. We have presented a set of fairly general rules including ones for introducing additional points to constrain the robot's path, circumnavigating obstacles with greater clearance, and in-/decreasing the target velocity. Of course, such rules can only be of limited generality. They must be reformulated if the robot is not equipped with an omni-directional drive or has a very different sensory apparatus.

Dealing with the Variance in Robot Behavior In our view, one of the main hindrances in the application of machine learning techniques to autonomous robot control is the high variance in robot behavior that is inherent in most systems that control physical robots in complex environments. This variance makes it very difficult to decide whether one control strategy is better than another one. Therefore it is a main idea of MTTL to explicitly take this variance into account.

To find a good parameterization of the control system we perform a search in the space of possible parameterizations. As this search space is continuous we consider a greedy search strategy as promising. Because accepting worse parameterizations as improvements would cause a hill-climbing learning algorithm to search into the wrong direction, the risk of making such wrong decisions should be minimized. Therefore MTTL uses a stringent test: a new candidate parameterization is accepted only if it increases the performance significantly (in a statistical sense).

In our learning experiments we have to deal with various sources of variance. For example, the time for traversing a doorway depends on whether or not sonar cross talk occurs in a particular trial, how close a particular path passes the door frame, the amount of rotation upon entering the doorway, etc.

Given such high variances, passing the significance test requires a large number of time consuming experiments. To reduce the number of experiments we employed additional techniques: testing performance gains of adaptations locally instead of globally (with an envelope to account for side effects of the adaptation), test a two plans alternatingly (to minimize the effect of unmodelled influence factors like the process load of computers), and use macro revision rules, such as transformation rule **R-6**, that perform multiple revisions.

Given that we use such a stringent criterion for the acceptance of new candidate SRPA the time needed for tests is reasonable. To test one SRPA revision takes about 60 minutes and a complete learning session for one navigation task takes about 6-8 hours. These time requirements, however, can be drastically reduced, probably by a factor 3-4, if (1) a navigation task and its inverse task are learned together and (2) using local tests several hypotheses can be tested simultaneously. For example, two revisions for traversing different doors could be tested simultaneously without having to consider interferences between the two revisions. These optimizations would imply that an office delivery robot could autonomously adapt itself to its working environment within a weekend. Further reductions can be obtained by adequately generalizing learning results.

5 Related Work

Our adaptation problem differs from those tackled by other researchers [**?**, **?**, **?**] in that these researchers only investigate learning techniques for parameterizing reactive control systems and do not apply their techniques to control systems that are already tuned for average performance.

MTTL differs from other approaches to automatically generating task and environment specific controllers such as genetic algorithms [**?**, **?**] and reinforcement learning [**?**, **?**]. Floreano [**?**], for example, use evolutionary learning for learning the parameterization of a simple robot control system. Santamaria and Ram [**?**] describe an approach for learning a so-called adaptive policy, a mapping from perceived situations to the continuous space of possible configurations of a purely reactive navigation system. They consider the problem as a reinforcement learning problem and solve it using a hybrid learning algorithm that combines techniques from case-based learning and reinforcement learning. Both kinds of approaches are very elegant, do not require (sophisticated) models, and are therefore very general. They pay, however, for this generality with a larger number of hypotheses that must be considered in their search for better controllers. Unfortunately, in autonomous robotics these hypotheses have to be tested by physical experiments, which limits the complexity of the problems that can be tackled with these approaches. It would be very interesting and instructive to test these general and model-free approaches on hand-tuned and sophisticated controllers (such as the RHINO system [**?**]) where the parameterizations that improve the performance are very sparse in the space of possible parameterizations. For the adaptation of sophisticated controllers the use of models is a valuable resource to find parameterizations that are capable of improving the behavior. In addition, we only know of applications of these learning techniques to purely reactive controllers. It remains to be investigated whether pure mappings from perceptions to parameterizations are capable of producing satisfying global behavior. For example, in the context of service robotics certain parameter configurations should be associated with environment regions rather than perceived situations in order to achieve more coherent and predictable navigation behavior. Furthermore, learned controllers are in general not transparent for human programmers.

Goel et al. [**?**] introduce a framework that is, in some aspects, similar to ours: A model-based method monitors the behavior generated by a reactive robot control system, detects failures in the form of behavioral cycles, analyzes the processing trace, identifies potential transformations, and modifies the reactive controller. However, the method is used to reconfigure a reactive control systen when it is trapped in a cyclic behavior due to the lack of a global perspective on the task it has to perform. They only perform revisions to resolve conflicts between different reactive behaviors. In addition, their revisions are specific to particular problem episodes and cannot be reused for other episodes.

6 Conclusions

In this paper we have investigated the problem of improving the performance of general, parameterizable robot control systems by autonomously adapting them to specific tasks and environments. This includes control systems that are already tuned for average performance. The adaptation problem for those control systems is challenging because the parameterizations that improve their performance are very sparse in the space of possible parameterizations.

We have proposed ***model- and test-based transformational learning (*MTTL*)*** as a computational model for performing this task. The scope of robot control systems that MTTL is applicable to includes those that employ strategic decision as well as reactive execution components. MTTL can adapt sophisticated control systems successfully because it uses abstract ***models*** of environments and robot control systems to propose promising adaptations, performs a greedy search by ***transforming*** structured reactive parameterizations (SRPAs), and ***tests*** a transformed SRPA to accept it only if the SRPA improves the robot's behavior with statistical significance. Performing statistical significance tests based on experimentation in the physical world enables MTTL to account for model deficiencies resulting from abstraction and explicitly deals with the high variance of robot behavior in complex environments.

We have described **XFRMLEARN**, an implementation of MTTL, and applied it to the problem of indoor navigation. XFRMLEARN has been integrated into structured reactive controllers (SRCs) and has successfully learned better navigation plans for the robot RHINO. These navigation plans are also used by the planning system XFRM. XFRMLEARN has autonomously operated an RWI B21 autonomous mobile robot for more than 100 hours in an office environment. Our experiments have shown that XFRMLEARN is capable of improving the performance of the RHINO navigation system, a state-of-the-art navigation system, both significantly and substantially: in two extended learning sessions XFRMLEARN has improved RHINO's navigation behavior within 6 to 7 hours by 31% and 44% respectively.

Acknowledgements. This work is partly supported by the Deutsche Forschungsgemeinschaft (DFG) (Contract No. BE 2200/3-1).

REFERENCES

[1] M. Beetz. Structured reactive controllers — a computational model of everyday activity. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[2] M. Beetz, W. Burgard, D. Fox, and A. Cremers. Integrating active localization into high-level control systems. *Robotics and Autonomous Systems*, 23:205–220, 1998.

[3] M. Beetz, M. Giesenschlag, R. Englert, E. Gülch, and A. B. Cremers. Semi-automatic acquisition of symbolically-annotated 3d models of office environments. In *International Conference on Robotics and Automation (ICRA-99)*, 1999.

[4] P. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.

[5] D. Floreano. Robotics in artificial life and behavior engineering. In T. Gomi, editor, *Evolutionary Robotics*. AAI Books, Ontario, 1998.

[6] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 1997.

[7] A. Goel, E. Stroulia, Z. Chen, and P. Rowland. Model-based reconfiguration of schema-based reactive control architectures. In *Proceedings of the AAAI Fall Symposium on Model-Based Autonomy*, 1997.

[8] A. Howe and L. Pyeatt. Integrating pomdp and reinforcement learning for a two layer simulated robot architecture. In *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, WA, 1999.

[9] J. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[10] J. Santamaria and A. Ram. Learning of parameter-adaptive reactive controllers for robotic navigation. In *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics*, Caracas, Venezuela, 1997.

[11] R. Sutton and A. Barto. *Reinforcement Learning: an Introduction*. MIT Press, 1998.

[12] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99)*, 1999.

[13] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.