Towards Comprehensive Computational Models for Plan-based Control of Autonomous Robots

Michael Beetz
Munich University of Technology
Department of Computer Science IX
Orleanstr. 34
D-81667 Munich, Germany

Abstract

In this paper we present an overview of recent developments in the plan-based control of autonomous robots. We identify computational principles that enable autonomous robots to accomplish complex, diverse, and dynamically changing tasks in challenging environments. These principles include plan-based high-level control, probabilistic reasoning, plan transformation, and context and resource-adaptive reasoning. We will argue that the development of comprehensive and integrated computational models of plan-based control requires us to consider different aspects of plan-based control — plan representation, reasoning, execution, and learning — together and not in isolation. This integrated approach enables us to exploit synergies between the different aspects and thereby come up with simpler and more powerful computational models.

In the second part of the paper we describe *Structured Reactive Controllers (SRCs)*, our own approach to the development of a comprehensive computational model for the plan-based control of robotic agents. We show how the principles, described in the first part of the paper, are incorporated into the SRCs and summarize results of several long-term experiments that demonstrate the practicality of SRCs.

Introduction

In recent years, autonomous robots, including XAVIER, MARTHA (AFH+98), RHINO (BCF+00; BAB+01), MINERVA, and REMOTE AGENT, have shown impressive performance in longterm demonstrations. In NASA's Deep Space program, for example, an autonomous spacecraft controller, called the *Remote Agent* (MNPW98), has autonomously performed a scientific experiment in space. At Carnegie Mellon University XAVIER (SGH+97), another autonomous mobile robot, has navigated through an office environment for more than a year, allowing people to issue navigation commands and monitor their execution via the Internet. In 1998, MINERVA (TBB+00) acted for thirteen days as a museum tourguide in the Smithsonian Museum, and led several thousand people through an exhibition.

These autonomous robots have in common that they perform plan-based control in order to achieve better problem-solving competence. In the plan-based approach robots generate control actions by maintaining and executing a plan that is effective and has a high expected utility with respect to the robots' current goals and beliefs. Plans are robot control programs that a robot cannot only execute but also reason about and manipulate (McD92a). Thus a plan-based controller is able to manage and adapt the robot's intended course of action — the plan — while executing it and can thereby better achieve complex and changing tasks. The plans used for autonomous robot control are often reactive plans, that is they specify how the robots are to respond in terms of low-level control actions to continually arriving sensory data in order to accomplish their objectives. The use of plans enables these robots to flexibly interleave complex and interacting tasks, exploit opportunities, quickly plan their courses of action, and, if necessary, revise their intended activities.

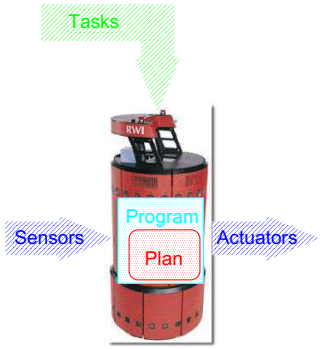

Figure 1: Plan-based control of robotic agents. The control program specifies how the robot is to respond to sensory input to accomplish its task. The plan is the part of the control program that the robot explicitly reasons about and manipulates.

To be reliable and efficient, autonomous robots must flexibly interleave their tasks and quickly adapt their courses of action to changing circumstances. Recomputing the best possible course of action whenever some aspect of the robot's situation changes is not feasible but can often be made so if the robots' controllers explicitly manage the robots' beliefs and current goals and revise their plans accordingly. The use of plans helps to mitigate this situation in at least two ways. First, it decouples computationally intensive control decisions from the time pressure that dom-

inates the feedback loops. Precomputed control decisions need to be reconsidered only if the conditions that justify the decisons change. Second, plans can be used to focus the search for appropriate control decisions. The can neglect control decisions that are incompatible with its intended plan of action.

In the remainder of this paper we proceed as follows. In the first part, we describe principles and building blocks of computational models for plan-based control. In the second part, we will then outline our initial steps towards such a comprehensive computational model that contains the building blocks introduced in the first part.

Principles of Plan-based Control

Plans in plan-based control have two roles. They are both executable prescriptions that can be interpreted by the robot to accomplish its jobs and syntactic objects that can be synthesized and revised by the robot to meet the robot's criterion of utility. Besides having means for representing plans, plan-based controllers must also be equipped with tools that enable planning processes to (1) project what might happen when a robot controller gets executed and return the result as an execution scenario; (2) infer what might be wrong with a robot controller given an execution scenario; and (3) perform complex revisions on robot controllers.

Let us now consider some of the key issues in the plan-based control of robotic agents: *dynamic system perspective, probabilistic reasoning, symbol grounding*, and *context and resource-adaptive reasoning.*

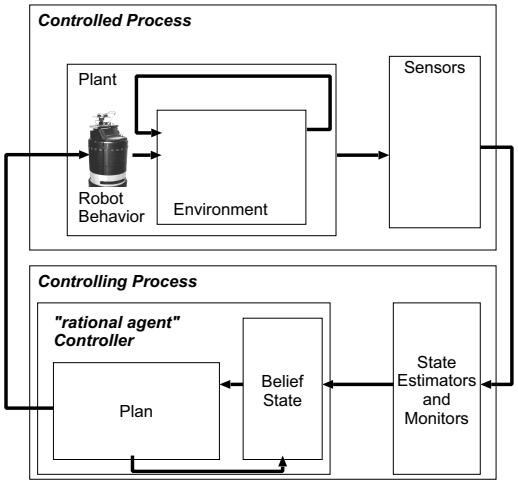

Figure 2: Block diagram of our dynamic system model for autonomous robot control. Processes are depicted as boxes and interactions as arcs.

Dynamic System Perspective. Since flexibility and responsiveness to changing situations are important characteristics of the robot behavior, we use ***dynamic systems*** as the primary abstract model for programming the operation of the integrated plan-based controller (see figure 2). In this model, the state of the world evolves through the interaction of two processes: the controlling process – the robot's control system – and the controlled process, which comprises events in the environment, physical movements of the robot and sensing operations. For complex dynamic systems, it is often useful to further decompose the controlled process into an environment and a sensing process. The environment process changes the world state and the sensing process maps the world state into the sensor data received by the robot. This suggests making a similar decomposition of the controlling process into state estimation and action generation processes. State estimation processes compute the robot's beliefs about the state of the controlled system. Auxiliary monitoring processes signal system states for which the controlling process is waiting. An action generation process specifies the control signals supplied to the controlled process as a response to the estimated system state.

The main consequence of this model is that robot action plans must control concurrent and continuous processes both flexibly and reliably.

Probabilistic Reasoning. Probabilistic reasoning is a key technique employed in the control of autonomous robots. Probabilistic reasoning is employed in a number of different ways.

First, plan generation and revision methods compute plans that have a probability of achieving a given goal with a probability higher than a specified threshold or they compute plans with the best expected cost benefit trade-off (BDH98; KHW95; DHW94). To employ such probabilistic planning techniques actions are represented through probabilistic effect models and the planning techniques consider probability distributions over world states rather than the states themselves. The advantage of these techniques is that they can properly handle the uncertainty caused by incomplete knowledge and inaccurate and unreliable sensors and the uncertainty resulting from non-deterministic action effects. The main problem associated with these techniques are the computational cost associated with the application of these techniques.

The second area in plan-based control where probabilistic reasoning techniques are heavily used is the interpretation of sensor data acquired by the robots' sensors (Thr00). The plan-based high-level control of robotic agents is founded on abstract perceptions of the current state of objects, the robot, and its environment. In order to derive such abstract perceptions from local and inaccurate sensors robustly, plan-based controllers often employ probabilistic state estimation techniques (SB01). The state estimators maintain the probability densities for the states of objects over time. Whenever state information is requested by the planning component, they provide the most likely state of the objects.

The probability density of an object's state conditioned on the sensor measurements received so far contains all the information which is available about the object. Based on this density, one is not only able to determine the most likely state of the object, but one can also derive even more meaningful statistics like the variance and entropy of the current estimate. In this way, the high-level system is able to reason about the reliability of an estimate.

A third application field of probabilistic reasoning is learning. Probabilistic reasoning techniques enable robots

to learn symbolic actions, probabilistic action models, and competent action selection strategies from experience.

Symbol Grounding. One of the key difficulties in the application of plan-based control techniques to object manipulation tasks is the symbol grounding or anchoring problem. In most plan representations constants used in the instantiations of plan steps denote objects in the world. This is a crude oversimplification because robots often do not have direct physical access to the objects themselves. Rather the control systems must use object descriptions that are computed from sensor data in order to manipulate objects. The use of object descriptions rather than objects to instantiate manipulation actions yields problems such as ambiguous, inaccurate, and invalid object descriptions. Powerful computational models of plan-based control must therefore have much more expressive representational means to make these problems transparent to the planning techniques. Several researchers (Fir89; McD90; CS00) have developed techniques to incorporate object descriptions into plan-based control.

Plan Transformation. Another key issue in the plan-based control of robots, in particular for those robots that are to act in dynamic and complex environments, is the fast formation and adaptation of plans. A very common idea for achieving fast plan formation is the idea of a *plan library*, a collection of canned plans for achieving standard tasks in standard situations (McD92b). However, such plans cannot be assumed to execute optimally. In a situation where an unexpected opportunity presents itself during the execution of the robot's tasks, for example, a canned plan will have trouble testing for the subtle consequences that might be implied by an alteration to its current plan. The decision criteria to take or ignore such opportunities must typically be hardwired into the canned plans when the plan library is built.

An alternative is to equip a robot with *self-adapting plans*, which carry out plans with the constraint that, whenever a specific belief of the robot changes, a runtime plan adaptation process is triggered. Upon being triggered, the adaptors decide whether plan revisions are necessary and, if so, perform them. Plan adaptation processes are specified explicitly, modularly, and transparently and are implemented using declarative plan transformation rules.

Context and Resource-adaptive Operation. To make its control decisions in a timely manner the plan-based controller applies various resource-adaptive inference methods (Zil96). These enable the controller to trade off accuracy and the risk of making wrong decisions against the computational resources consumed to arrive at those decisions. Moreover, the results of the resource-adaptive reasoning are employed to adapt the execution modes of the process in response to the robot's context (BACM98).

Building Blocks of Plan-based Control

The building blocks of plan-based control are the representation of plans, the execution of plans, various forms of automatic learning, and reasoning about plans, including plan generation and transformation, and teleological, causal, and temporal reasoning.

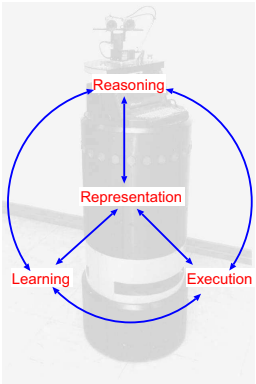

Figure 3: The main components of plan-based control are plan representation, execution, learning, and reasoning and their interactions.

But before we dive in and discuss the building blocks of modern plan-based control models let us first get an intuition of how traditional robot planning techniques function. Most of these techniques are based on the problem-space hypothesis (New90): they assume problems can be adequately stated using a state space and a set of discrete and atomic actions that transform states to successor states. A solution is an action sequence that transforms any situation satisfying a given initial state description into another state that satisfies the given goal. Plan generation is the key inferential task in this problem-solving paradigm.

As a consequence, representational means are primarily designed to simplify plan generation from first principles. Problem space plans are typically used in layered architectures (BFG+97), which run planning and execution at different levels of abstraction and time scales. In these approaches planning processes use models that are too abstract for predicting all consequences of the decisions they make and planning processes cannot exploit the control structures provided by the lower layer. Therefore they lack appropriate means for specifying flexible and reliable behavior and plans can only provide guidelines for task achievement.

Contrary to the plan space approach, plan-based control of robotic agents takes the stand that there is a number of inference tasks necessary for the control of an autonomous robot that are equally important. These inference tasks include ones that enable the competent execution of given plans, ones that allow for learning plans and other aspects of plan-based control, and various reasoning tasks, which comprise the generation and assessment of alternative plans, monitoring the execution of a plan, and failure recovery.

These different inference tasks are performed on a common data structure: the plan. Consequently, the key design issues of plan-based control techniques are representational and inferential adequacy and inferential and acquisitional efficiency as key criteria for designing domain knowledge representations (RK91). Transferring these notions to plan-based control, we consider the representational adequacy of plan representations to be their ability to specify the neces-

sary control patterns and the intentions of the robots. Inferential adequacy is the ability to infer information necessary for dynamically managing, adjusting, and adapting the intended plan during its execution. Inferential efficiency is concerned with the time resources that are required for plan management. Finally, acquisitional efficiency systems is the degree to which they support the acquisition of new plan schemata and planning knowledge.

To perform the necessary reasoning tasks the plan management mechanisms must be equipped with inference techniques to infer the purpose of subplans, find subplans with a particular purpose, automatically generate a plan that can achieve some goal, determine flaws in the behavior that is caused by subplans, and estimate how good the behavior caused by a subplan is with respect to the robot's utility model. Pollack and Horty (PH99) stress the point that maintaining an appropriate and working plan requires the robot to perform various kinds of plan management operations including plan generation, plan elaboration, commitment management, environment monitoring, model- and diagnosis-based plan repair, and plan failure prediction.

It does not suffice that plan management mechanisms can merely perform these inference techniques but they have to perform them fast. The generation of effective goal-directed behavior in settings where the robots lack perfect knowledge about the environment and the outcomes of actions and environments are complex and dynamic, requires robots to maintain appropriate plans during their activity. They cannot afford to entirely replan their intended course of action every time their beliefs change.

To specify competent problem-solving behavior the plans that are reasoned about and manipulated must have the expressiveness of reactive plan languages. In addition to being capable of producing flexible and reliable behavior, the syntactic structure of plans should mirror the control patterns that cause the robot's behavior — they should be realistic models of how the robot achieves its intentions. Plans cannot abstract away from the fact that they generate concurrent, event-driven control processes without the robot losing the capability to predict and forestall many kinds of plan execution failures. A representationally adequate plan representation for robotic agents must also support the control and proper use of the robot's different mechanisms for perception, deliberation, action, and communication. The full exploitation of the robot's different mechanisms requires mechanism-specific control patterns. Control patterns that allow for effective image processing differ from those needed for flexible communication, which in turn differ from those that enable reliable and fast navigation. To fully exploit the robot's different mechanisms, their control must be transparently and explicitly represented as part of the robot's plans. The explicit representation of mechanism control enables the robot to apply the same kinds of planning and learning techniques to all mechanisms and their interaction.

The defining characteristic of plan-based control is that these issues are considered together: plan representation and the different inference tasks are not studied in isolation but in conjunction with the other inference tasks. The advantage of this approach is that we can exploit synergies between the different aspects of plan-based control.

Plan management capabilities simplify the plan execution problem because programmers do not have to design plans that deal with all contingencies. Rather plans can be automatically adapted at execution time when the particular circumstances under which the plan has to work are known. Plan execution mechanisms can also employ reasoning mechanisms in order to get a broader coverage of problem-solving situations. The REMOTE AGENT, for example, employs propositional reasoning to derive the most appropriate actions to achieve the respective immediate goals (WN97; NW97). On the other side, competent plan execution capabilities free the plan management mechanism from reasoning through all details. Reasoning techniques such as diagnostic and teleological reasoning are employed in transformational learning techniques in order to perform better informed learning decisions and thereby speed up the learning process (BB00). Skill learning mechanisms have also been applied to the problem of learning effective plan revision methods (Sus77). There is also a strong interaction between the learning and execution mechanisms in plan-based control. Learning mechanisms are used to adapt execution plans in order to increase their performance . Competent execution mechanisms enable the learning mechanisms to focus on strategical aspects of problem-solving tasks.

Structured Reactive Controllers: a Computational Model of Plan-based Control

After having described the general components of computational models of plan-based control I want to give you now a brief overview of our own approach to the development of such integrated computational models. The robot controllers that realize this computational model are called ***Structured Reactive Controllers (SRCs)*** (Bee01). Structured Reactive Controllers are self-adapting plans that specify concurrent reactive behavior. They revise themselves during the execution of specified user commands in order to exploit opportunities and avoid predictable problems. They are also capable of experience-based learning.

Structured Reactive Controllers use a very expressive plan language, called RPL (McD91), and a number of software tools for predicting the effects of executing plans, for teleological and causal reasoning about plans, for revising plans during their execution, and for automatically learning routine plans.

Given a set of jobs, an SRC concurrently executes the default routines for each individual job. These routine activities are general and flexible and work well in standard situations. They can cope well with partly unknown and changing environments, run concurrently, handle interrupts, and control robots without assistance over extended periods. For standard situations, the execution of these routine activities causes the robot to exhibit an appropriate behaviour while achieving its purpose. While it executes routine activities, the SRC also tries to determine whether its routines might interfere with each other and monitors robot operation for non-standard situations. If one is found, it will try to antici-

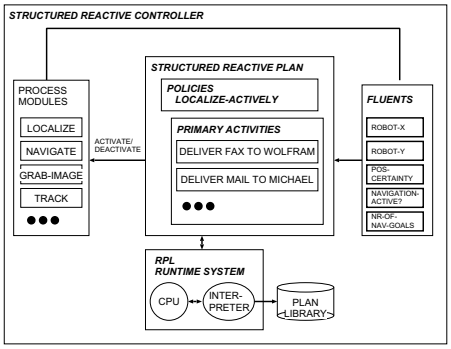

Figure 4: Components of a structured reactive controller. The structured reactive plan specifies how the robot responds to changes of its fluents, registers that are asynchronously set by the sensing processes. The interpretation of the structured reactive plan results in the activation, parameterization, and deactivation of process modules that execute and monitor the physical continuous control processes.

pate behaviour flaws by predicting how its routine activities might work in these non-standard situations. If necessary, it revises its routines to make them robust for this kind of situation. Finally, it integrates the proposed revisions into the activities it is pursuing.

Transformational Planning of Concurrent Reactive Plans. Consider the following plan adaptor, which illustrates the planning techniques employed by SRCs.

With plan adaptor Whenever the robot detects an open door that was assumed to be closed
if this situation ***is an opportunity***
then it ***changes its course of action*** to make use of the opportunity

Concurrent reactive plan

The plan adaptor is triggered by a change of its belief about an door being open or closed. Upon being triggered the adaptor decides whether a change in the intended course of activity is suitable and if so performs it. The process of plan adaptation is realized through transformational planning (McD92b; Bee00).

Transformational planning is implemented as a search in plan space. A node in the space is a proposed plan; the initial node is the default plan created using the plan library. A step in the space requires three phases. First, a plan adaptor *projects* a plan to generate sample execution scenarios for it. Then, in the *criticism* phase, a plan adaptor examines these execution scenarios to estimate how good the plan is and to predict possible plan failures. It diagnoses the projected plan failures by classifying them in a taxonomy of failure models. The failure models serve as indices into a set of transformation rules that are applied in the third phase, *revision*, to produce new versions of the plan that are, we hope, improvements.

Prediction in Structured Reactive Controllers Temporal projection, the process of predicting what will happen when a robot executes its plan, is essential for many robots to successfully plan courses of action. To be able to project their plans, robots must have causal models that represent the effects of their actions. These causal models should be sufficiently realistic to predict the behavior generated by modern autonomous robot controllers accurately enough to foresee a significant range of real execution problems. This can be achieved if action models reflect the facts that physical robot actions cause continuous change; that controllers are reactive systems; that the robot is executing multiple physical and sensing actions; and that the robot is uncertain about the effects of its actions and the state of the environment.

The problem of using such realistic action models is obvious. Nontrivial concurrent plans for controlling robots reliably are very complex. There are usually several control processes active. Many more are dormant, waiting for conditions that trigger their execution. The branching factors for possible future states — not to mention the distribution of execution scenarios that they might generate — are immense. The accurate computation of this probability distribution is prohibitively expensive in terms of computational resources.

Learning Symbolic Robot Plans. We have already stressed the importance of representing the plans that the robot has committed to execute explicitly as a means of economically using the limited computational resources for flexible task execution and effective action planning. However, this raises the question of how such plans can be obtained. Many autonomous mobile robots consider navigation as a Markov decision problem. They model the navigation behavior as a finite state automaton in which navigation actions cause stochastic state transitions. The robot is rewarded for reaching its destination quickly and reliably. A solution for such problems is a mapping from states to actions that maximises the accumulated reward. Such state-action mappings are inappropriate for teleological and diagnostic reasoning, which are necessary to adapt quickly to changing circumstances and quickly respond to exceptional situations.

We have therefore developed XFRMLEARN (BB00), a learning component that builds up explicit symbolic navigation plans automatically. Given a navigation task, XFRMLEARN learns to structure continuous navigation behaviour and represents the learned structure as compact and transparent plans. The structured plans are obtained by starting with monolithic default plans that are optimized for average performance and adding subplans to improve the navigation performance for the given task.

XFRMLEARN's learning algorithm works as follows. XFRMLEARN starts with a default plan that transforms a navigation problem into an MDP problem and passes the MDP problem to RHINO's navigation system. After RHINO's path planner has determined the navigation policy the navigation system activates the collision avoidance module for the execution of the resulting policy. XFRMLEARN records the resulting navigation behaviour and looks for stretches of behaviour that could be possibly improved. XFRMLEARN

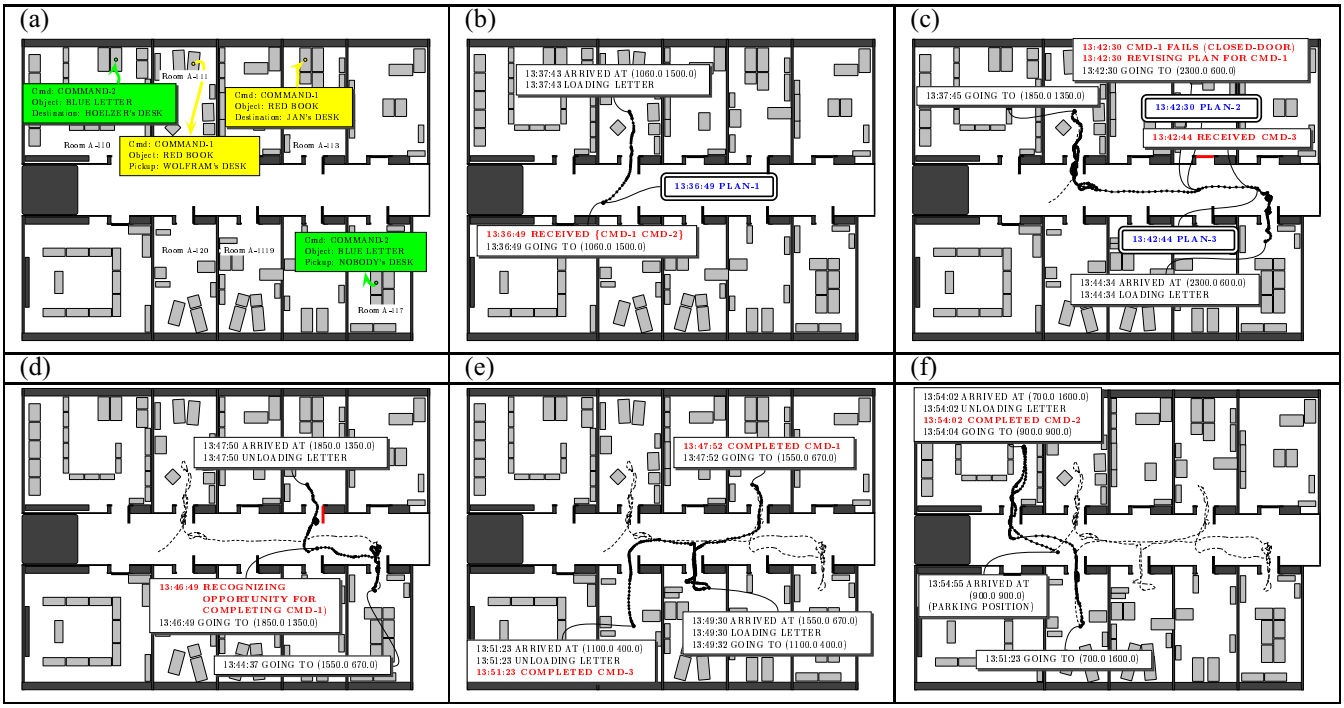

Figure 6: Execution trace for a delivery tour. RHINO receives two commands 6(a). Upon receiving the two commands the SRC puts plans for the commands into the plan, computes an appropriate schedule, and installs it. It also adds a control process that monitors that the rooms it must enter are open. The order of the delivery steps are that RHINO starts with picking up the book (Fig. 6(b)) and delivering it in A-113. After RHINO has left room A-111, it notices that room A-113 is closed (Fig. 6(c)). Because RHINO cannot complete the delivery of the book the SRC revises the plan by transforming the completion of the delivery into an opportunity. RHINO receives a third command which is integrated into the current schedule (Fig. 6(d)). As it passes room A-113 on its way to A-119 it notices that the door is now open and takes the opportunity to complete the first command (Fig. 6(d)). After that it completes the remaining steps as planned (Fig. 6(e-f).

then tries to explain the improvable behaviour stretches using causal knowledge and its knowledge about the environment. These explanations are then used to index promising plan revision methods that introduce and modify subplans. The revisions are subsequently tested in a series of experiments to decide whether they are likely to improve the navigation behaviour. Successful subplans are incorporated into the symbolic plan. An learning session is shown in figure 5.

Using this algorithm can autonomously learn compact and well-structured symbolic navigation plans by using MDP navigation policies as default plans and repeatedly inserting subplans into the plans that significantly improve the navigation performance. The plans learned by XFRMLEARN support action planning and opportunistic task execution by providing plan-based controllers with subplans such as traverse a particular narrow passage or an open area. More specifically, navigation plans (1) can generate qualitative events from continuous behaviour, such as entering a narrow passage; (2) support online adaptation of the navigation behaviour (drive more carefully while traversing a particular narrow passage) (Bee99), and (3) allow for compact and realistic symbolic predictions of continuous, sensor-driven behaviour (BG00).

Long-term Demonstrations

This section describes several experiments (figure 7) that evaluate the reliability and flexibility of the RHINO system and the possible performance gains that it can achieve.

The flexibility and reliability of runtime plan management and plan transformation has been extensively tested in a museum tourguide application. The robot's purpose was to guide people through a museum, explaining the exhibits to be seen along the robot's route. MINERVA (figure 7(a)) operated in the "Smithsonian Museum" in Washington for a period of thirteen days (TBB$^+$99). It employed an SRC as its high-level controller. During its period of operation, it was in service for more than 94 hours, completed 620 tours, showed 2668 exhibits, and travelled over a distance of more than 44 kilometers. The SRC directed MINERVA's course of action in a feedback loop that was carried out more than three times a second. MINERVA used plan adaptors for the installment of new commands, the deletion of completed plans, and for tour scheduling. MINERVA made about 3200 execution time plan transformations while performing its tourguide job. MINERVA's plan-based controller differs from RHINO's only with respect to its top-level plans in plan library and some of the plan adaptors that are used.

In another experiment we have evaluated the capabilities

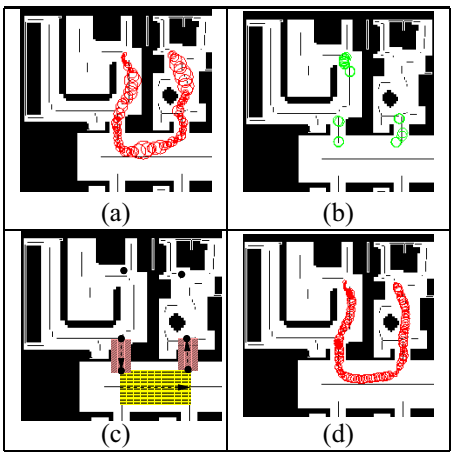

Figure 5: The figure visualizes a summary of a learning session: A behaviour trace of the default plan (a); behavior stretches where the robot moves conspiciously slowly (b); the added subplans in the learned navigation plan (c); and a behaviour trace of the learned plan, which is on average 29% faster than the default plan (d).

Figure 7: The mobile robots MINERVA (a) and the RWI B21 robot RHINO (c) that are used in the experiments.

of plan-based controllers to perform predictive plan management. This experiment has shown that predictive plan transformation can improve the performance by outperforming controllers without predictive capabilities in situations which require foresight while at the same time retaining their performance in situations that require no foresight. Figure 6 pictures an execution trace for a sample problem-solving scenario.

Conclusions

Our longterm research goal is to understand and build autonomous robot controllers that can carry out daily jobs in offices and factories with a reliability and efficiency comparable to people. We believe that many behavior patterns, such as exploiting opportunities, making appropriate assumptions, and acting reliably while making assumptions, that make everyday activity efficient and reliable require plan-based control and the specification of concurrent, reactive plans.

In this paper we have presented an overview of recent developments in the area of plan-based control of autonomous robots. Computational principles including plan-based high-level control, probabilistic reasoning, symbol anchoring, plan transformation, and context and resource-adaptive reasoning are incorporated in a number of state-of-the-art systems.

We believe that a necessary step towards more powerful plan-based robot controllers is the development of comprehensive and integrated computational models that address issues plan representation, reasoning, execution, and learning at the same time. A key component of such a computation model is the design of the plan representation language such that it allows for flexible and reliable behavior specifications, computationally feasible inference, stability in the case of runtime plan revisions, and automatic learning of symbolic plans for robot control.

Comprehensive computational models will enable us to tackle new application areas, such as the plan-based control of robot soccer teams, and longterm application challenges, for example, the robotic assistance of elderly people and the plan-based control of robotic rescue teams after disasters such as earthquakes.

References

R. Alami, S. Fleury, M. Herb, F. Ingrand, and F. Robert. Multi robot cooperation in the Martha project. *IEEE Robotics and Automation Magazine*, 5(1), 1998.

M. Beetz, T. Arbuckle, M. Bennewitz, W. Burgard, A. Cremers, D. Fox, H. Grosskreutz, D. Hähnel, and D. Schulz. Integrated plan-based control of autonomous service robots in human environments. *IEEE Intelligent Systems*, 16(5):56–65, 2001.

M. Beetz, T. Arbuckle, A. Cremers, and M. Mann. Transparent, flexible, and resource-adaptive image processing for autonomous service robots. In H. Prade, editor, *Procs. of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 632–636, 1998.

M. Beetz and T. Belker. Environment and task adaptation for robotic agents. In W. Horn, editor, *Procs. of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, 2000.

W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.

C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of AI research*, 1998.

M. Beetz. Structured reactive controllers — a computational model of everyday activity. In O. Etzioni, J. Müller, and J. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents*, pages 228–235, 1999.

M. Beetz. *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*, volume LNAI 1772 of *Lecture Notes in Artificial Intelligence*. Springer Publishers, 2000.

M. Beetz. Structured Reactive Controllers. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:25–55, March/June 2001.

P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1), 1997.

M. Beetz and H. Grosskreutz. Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior. In *Pro-*

ceedings of the Fifth International Conference on AI Planning Systems, Breckenridge, CO, 2000. AAAI Press.

S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In *Proc. of the 17th AAAI Conf.*, pages 129–135, Menlo Park, CA, 2000. AAAI Press.

D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. In K. Hammond, editor, *Proc. 2nd. Int. Conf. on AI Planning Systems*. Morgan Kaufmann, 1994.

J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. Technical report 672, Yale University, Department of Computer Science, 1989.

N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76:239–286, 1995.

D. McDermott. Planning reactive behavior: A progress report. In K. Sycara, editor, *Innovative Approaches to Planning, Scheduling and Control*, pages 450–458, San Mateo, CA, 1990. Kaufmann.

D. McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

D. McDermott. Robot planning. *AI Magazine*, 13(2):55–79, 1992.

D. McDermott. Transformational planning of reactive behavior. Research Report YALEU/DCS/RR-941, Yale University, 1992.

N. Muscettola, P. Nayak, B. Pell, and B. Williams. Remote agent: to go boldly where no AI system has gone before. *Artificial Intelligence*, 103(1–2):5–47, 1998.

A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.

P. Nayak and B. Williams. Fast context switching in real-time propositional reasoning. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 50–56, Menlo Park, 1997. AAAI Press.

M. Pollack and J. Horty. There's more to life than making plans: Plan management in dynamic, multi-agent environments. *AI Magazine*, 20(4):71–84, 1999.

E. Rich and K. Knight. *Artificial Intelligence*. McGraw Hill, New York, 1991.

D. Schulz and W. Burgard. Probabilistic state estimation of dynamic objects with a moving mobile robot. *Robotics and Autonomous Systems*, 34(2-3):107–115, 2001.

R. Simmons, R. Goodwin, K. Haigh, S. Koenig, J. O'Sullivan, and M. Veloso. Xavier: Experience with a layered robot architecture. *ACM magazine* Intelligence, 1997.

G. Sussman. *A Computer Model of Skill Acquisition*, volume 1 of *Aritficial Intelligence Series*. American Elsevier, New York, NY, 1977.

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.

S. Thrun, M. Beetz, M. Bennewitz, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 2000. to appear.

S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

B. Williams and P. Nayak. A reactive planner for a model-based executive. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1178–1185, San Francisco, 1997. Morgan Kaufmann Publishers.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.