Integrated Plan-based Control of Autonomous Service Robots in Human Environments

Michael Beetz[*], Tom Arbuckle[†], Thorsten Belker[†], Maren Bennewitz[‡], Wolfram Burgard[‡]
Armin B. Cremers[†], Dieter Fox[§], Henrik Grosskreutz[¶], Dirk Hähnel[‡], and Dirk Schulz[†]

Abstract

In this article we present an overview of recent extensions to the RHINO system, a system for controlling autonomous robots. We identify computational principles that enable RHINO to accomplish complex, diverse, and dynamically changing tasks in human working environments. These principles include plan-based high-level control, probabilistic reasoning, plan transformation, and context and resource-adaptive reasoning. We show how these principles are incorporated into RHINO's software modules. Long-term experiments demonstrate the practicality of our approach.

1 Introduction

Controlling an autonomous robot entails specifying how the robot is to react to sensory input in order to accomplish its goals. AI-based robotics researches specialisations of this control problem. One of these specialisations' key challenges is the creation of an autonomous robot which can accomplish prolonged, complex and dynamically changing tasks while operating in real-world environments. To ensure continuing utility, the robot must be able to learn task-specific environment models and autonomously tailor its control routines. We take, as a prototypical example, a robot office courier.

Real-world environments have several typical characteristics. They are usually not prepared for the robots' benefit and they change independently of the robots' actions since the robots are not the sole inhabitants of their worlds. Moreover, the robots have no sources of knowledge about their environments other than their own sensors. In such environments, there is a strong requirement that robots have rich perceptual capabilities to carry out the diverse tasks assigned to them. In particular, they must at least be able to recognise objects, and places and be able to interactively communicate with people.

[*] Institute of Computer Science IX, Munich University of Technology, Arcisstr. 21, D-80333 Munich, Germany.

[†] Department of Computer Science III, University of Bonn, Roemerstr. 164, D-53117 Bonn, Germany.

[‡] Department of Computer Science, University of Freiburg, Am Flughafen 17, D-79110, Freiburg, Germany.

[§] Department of Computer Science, University of Washington, Box 352350 Seattle, WA 98195-2350, USA.

[¶] Department of Computer Science V, Aachen University of Technology, D-52056 Aachen, Germany.

Requirements are also imposed on the control systems for these robots. Sets of complex and possibly interacting tasks require task management involving predictive action planning. This includes being able to reason about and manage concurrent sub-tasks during their execution. The reasoning processes must succeed despite the boundedness of computational resources. Moreover, prolonged successful operation or adaptation to new environments require the controllers to modify themselves on the bases of their experiences and improve their control routines as their environments and tasks change.

Recently, autonomous robots have operated successfully as museum tourguides [8, 24] and performed simple cleaning [9] and delivery tasks [23, 14, 26]. The tasks were carefully chosen: they made great demands on the robots' navigation skills and on safe and reliable operation in unstructured, dynamic environments but their structure was kept deliberately simple.

In recent years, the RHINO research group has been investigating computational principles for autonomous control systems that are capable of accomplishing complex and diverse tasks in human working environments. Our research has therefore focused on developing advanced perceptual capabilities, learning and adaptation capabilities, and planning mechanisms for robotic agents.

In this article, we will give an overview of recent extensions to the RHINO system that represent our initial steps on the way to achieving some of the challenges of AI-based robotics stated above. After presenting a typical activity scenario to be used as a descriptive example throughout the paper, we proceed to describe the key design principles underlying our robot control system. We continue with a description of the software architecture and the functions of its components. We close with short summaries of the long-term experiments we have performed with our control system and with our conclusions.

2 An Integrated Plan-based Controller for a Robot Office Courier

Let us begin our discussion of integrated plan-based control by presenting a problem-solving episode that is to be performed by the robot RHINO (see figure 10(c)) acting here in the capacity of an autonomous robot office courier. RHINO, an RWI B21 mobile robot, is controlled by a self-adapting, plan-based robot controller called ***Structured Reactive Controller***

(SRC) [3].

The SRC performs the activity management for the robot courier and enables the robot to carry out schedules of multiple jobs both reliably and efficiently. The SRC consists of a collection of concurrent control routines that specify routine activities and can adapt themselves to non-standard situations. More specifically, the SRC employs three kinds of control processes: routine activities that handle standard jobs in standard situations, monitoring processes that detect non-standard situations, and planning processes that are responsible for the adaptation of standard routines to non-standard situations.

Consider the following delivery episode which illustrates the kinds of behaviour that robots can exhibit using SRCs for activity management. In this episode, RHINO receives the following two commands (see figure 1): "put the red letter on the meeting table in room A-111 on the desk in A-120." and "deliver the green book from the librarian's desk in room A-110 to the desk in A-113."

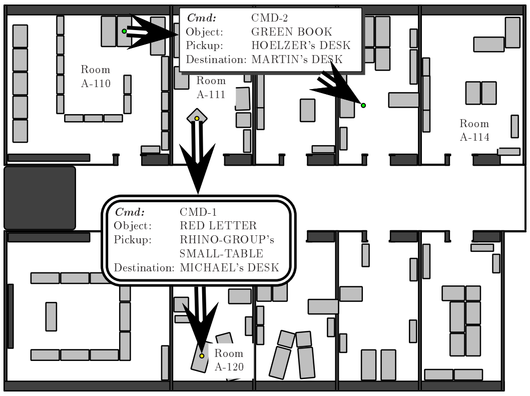


Figure 1: The figure shows RHINO's operating environment for the robot office courier application. The environment consists of a small hallway with adjacent offices, a library, and a classroom. The figure also depicts two delivery requests.

To accomplish its commands RHINO uses a library of ***routine plans***. The routine plan for deliveries specifies that RHINO is to navigate to the pickup point, wait until the letter is loaded, navigate to the destination of the delivery, and wait for the letter to be unloaded.

Whenever RHINO is to carry out a set of delivery jobs, it quickly computes an appropriate schedule. RHINO's initial schedule is to deliver the red letter first and the green book afterwards. Proposing a schedule implies making assumptions about whether doors are open and closed. In our experiment RHINO assumes that the rooms A-110, A-111, A-113, and A-120 are open. To adapt its schedule flexibly, RHINO monitors the scheduling assumptions while performing its deliveries: whenever it passes a door it estimates the opening angle of that door and revises the schedule if necessary.

Figure 2, 3(a), and 3(b) picture three qualitatively different event traces that the SRC has produced while carrying out the delivery jobs under different circumstances.

Let us explain how the SRC produces the event trace depicted in figure 2. Initially, all doors in the environment are open. RHINO starts with the delivery of the red letter and heads to the meeting table in A-111 where the letter is loaded **(step 1)**. Then the door of A-120 is closed **(step 2)**. Thus when RHINO enters the hallway to deliver the red letter at Michael's desk it estimates the opening angle of the door of room A-120 and detects that the door has been closed **(step 3)**. A failure is signalled.

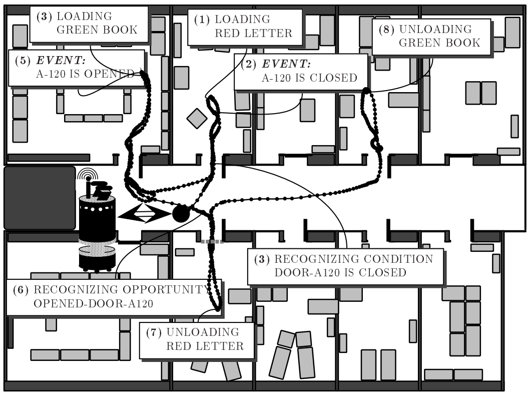
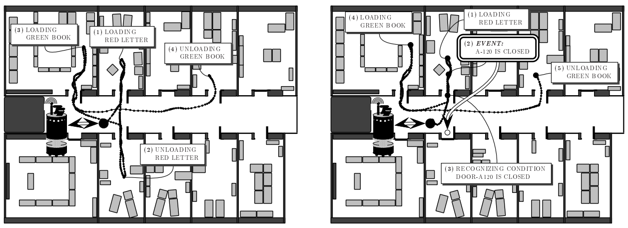


Figure 2: Complete trajectory and event trace for the two deliveries if A-120 is closed while the robot is in A-111 and opened again while it is in A-110.

Since RHINO does not know when room A-120 will be open again, it revises the schedule so that it now delivers the green book first and accomplishes the failed delivery as an opportunity. Thus RHINO navigates to the librarian's desk in A-110 to pick up the green book to room A-113 **(step 4)**. At this moment room A-120 is opened again **(step 5)**. As RHINO heads towards A-113 to deliver the green book it passes room A-120 **(step 6)**. At this point the door estimation process signals an opportunity: A-120 is open! Therefore, RHINO interrupts its current delivery to complete the delivery of the red letter. After the delivery of the red letter is completed **(step 7)**, RHINO continues the delivery of the green book **(step 8)**.



(a) (b)

Figure 3: Complete trajectory and event trace for the two deliveries if all doors are open (a) and if A-120 is closed but not opened again (b).

Figures 3(a) and (b) show the behavior generated by the same SRC if all doors stay open (figures 3(a)) and if A-120 is closed but not opened again (figures 3(b)).

The self-adapting plan that accomplishes these behaviors consists of the *primary activities* that specify the course of action for satisfying the user requests — the delivery tour. The plan also contains two plan adaptors. The first one makes sure that the intended course of action contains delivery tasks for every actual user request and no unnecessary delivery tasks. Thus, whenever a user request is added, deleted, or revised, the plan adaptor performs the necessary changes on the intended primary activities. The second plan adaptor makes sure that the collection and delivery steps of the tour are ordered so that there are no unwanted interactions between the steps and that the tour can be accomplished quickly.

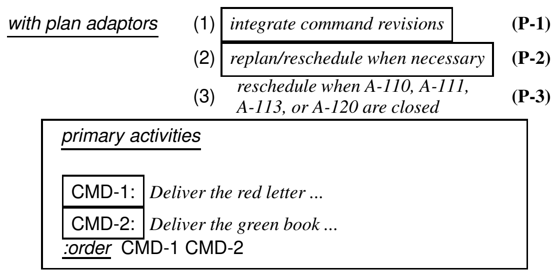


Figure 4: The figure shows the toplevel structure of the SRC after the plans for the two delivery requests have been inserted. The SRC also contains a plan adaptor that triggers a rescheduling process when one of the doors that have to be passed on the delivery tour is detected to be closed.

During the course of the episode, the SRC modifies itself as follows. RHINO initially performs no primary activities. Its plan adaptors ensure that new commands are received and processed. Upon receiving the two jobs (event “start” in figure 2) the plan adaptor **P-1** puts plans for accomplishing them into the primary activities. The insertion of the commands triggers the scheduler of the plan adaptor **P-2**. The plan adaptor for rescheduling also adds an additional plan adaptor **P-3** that monitors the assumptions underlying the schedule, that is that the rooms A-110, A-111, A-113, and A-120 are open. The SRC after this revision is shown in figure 4.

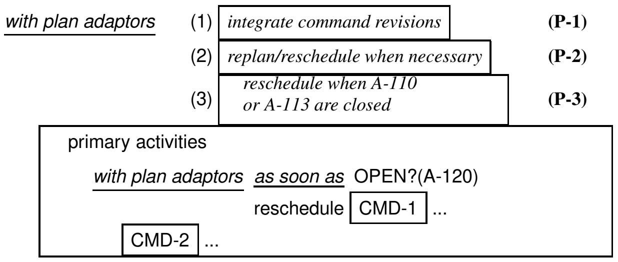


Figure 5: Structure of the SRC after A-120 is detected to be closed. The completion of the delivery for room A-120 is handled as an opportunity. As soon as RHINO learns that A-120 is open, the plan for the request is inserted in the actual schedule.

After RHINO has picked up the red letter and left room A-111, it notices that room A-120 has been closed in the meantime (event 3 in figure 2). Since RHINO cannot complete the delivery of the red letter the corresponding command fails. This failure triggers the replanning plan adaptor **P-2** which transforms the completion of the delivery into an opportunity (see figure 5). Thus, as soon as RHINO notices that room A-120 is open (event 6), it interrupts its current mission, completes the delivery of the red letter (event 7), and continues with the remaining jobs after the red letter has been delivered.

3 Design Principles

Let us now detail the key design principles embodied in the plan-based controller that we have just seen in action: ***dynamic system perspective, plan-based high-level control, probabilistic reasoning, plan transformation,*** and ***context and resource-adaptive reasoning.***

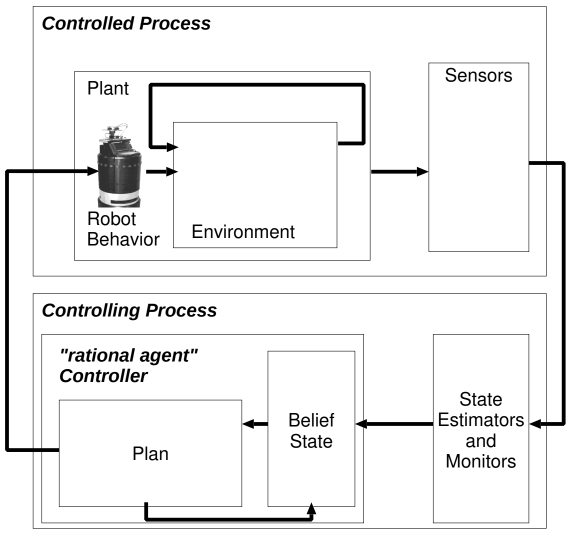


Figure 6: Block diagram of our dynamic system model for autonomous robot control. Processes are depicted as boxes and interactions as arcs. The dynamic system consists of the controller (or controlling process) and the plant (or controlled process). The plant is decomposed into the environment and sensor process and the controller into the state estimation and the rational agent process.

3.1 Dynamic System Perspective

Since flexibility and responsiveness to changing situations are important characteristics of the robot behaviour, we use ***dynamic systems*** as the primary abstract model for programming the operation of the integrated plan-based controller (see figure 6). 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 that are supplied to the controlled process as a response to the estimated system state.

The main consequence of this model is that robots must control concurrent and continuous processes both flexibly and reliably. In the RHINO system, this capability is achieved through control routines that specify and synchronise ***concurrent percept-driven*** behaviour. The routines are implemented in RPL (Reactive Plan Language) [16], a control language providing traditional control abstractions as well as high-level constructs (interrupts, monitors) for the synchronisation of parallel actions. RPL also allows for the specification of interactive behaviour, concurrent control processes, failure recovery methods, and the temporal coordination of processes. We employ these features to make plans reactive and robust by incorporating within them both sensing and monitoring actions as well as reactions triggered by observed events.

3.2 Plan-based High-level Control

To be reliable and efficient, our robot courier must flexibly interleave its tasks, exploit opportunities, quickly plan its courses of action, and, if necessary, revise its intended activities. Recomputing the best possible course of action whenever some detail of the robot's situation changes is not computationally tractable but can often be made so if the robots' controllers explicitly manage the robots' beliefs and current goals, revising their plans accordingly.

In RHINO, plans are symbolic representations of the future activity of the robot that 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 synthesised and revised by the robot to meet the robot's criterion of utility. To support plan formation and adaptation operations, RPL comes 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 [7]; (2) infer what might be wrong with a robot controller given an execution scenario [17]; and (3) perform complex revisions on robot controllers [17].

3.3 Probabilistic State Estimation

The plan-based high-level control of the robot courier is founded on abstract perceptions of the current state of objects, the robot, and its environment. In order to derive such abstract perceptions from RHINO's local and inaccurate sensors robustly, the RHINO system employs probabilistic state estimation techniques [21]. 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.

3.4 Plan Transformation

Since office environments are dynamic and the robot is uncertain about the state of its environment, a robot office courier must also be able to form and adapt plans quickly. 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. 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.

Plan transformation is not only used for runtime plan adaptation, but also to learn canned plans from experience [6].

3.5 Context and Resource-adaptive Operation

To make its control decisions in a timely manner the plan-based controller applies various resource-adaptive inference methods. Probabilistic state estimation, for example, is performed through sampling-based inference methods [20]. 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 [5].

As an example, consider the case where the robot is required to find a letter on a desk for future delivery. The robot must employ visual sensing to detect the letter but the post-processing of this sensing will depend heavily on the current context. The controlling process is aware of such internal and external factors as lighting, distance to object, object colour, visual clutter, current processing load and employs them to alter the way in which the envelopes are detected.

4 Control System Architecture

Figure 7 depicts the software architecture of the current RHINO-control system. The perceptual subsystem is shown on the right and the action subsystem on the left. Both are parts of the low-level control system (LLS). The central box is the high-level controller, which we call a *Structured Reactive Controller (SRC)*. The interface between the low-level and high-level control system is constituted by HLI, the High-Level Interface. We now describe the functions and operation of these modules in greater detail.

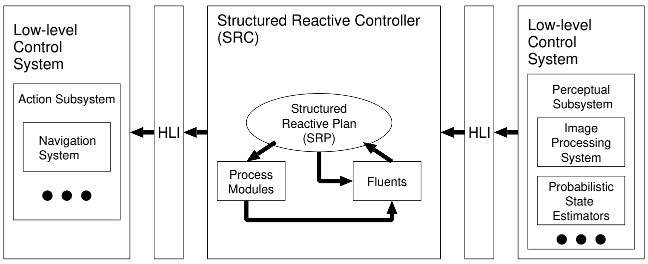


Figure 7: Block diagram of the RHINO control system. The diagram shows the control and communication links for plan-based control. The control and communication links within the low-level control system are omitted.

4.1 The Low-level Control System

The low-level control system (LLS) consists of approximately 20 distributed modules, each of which is designed to monitor or control dedicated aspects of the robot and to provide information to other modules. The modules communicate using an asynchronous message-passing library.

The LLS contains various server modules that interface directly with the robot's sensors and effectors (lasers, sonars, cameras, motors, pan/tilt unit, speech unit, buttons or touch-sensitive display, and so on). It also contains various navigation modules that perform functions like mapping, path planning, localisation and collision avoidance.

The modules of the LLS share important design characteristics. Many of them solve computational problems that can be compactly described in mathematical terms. The localisation module, for example, computes the probability distribution over the robot's position, or the path planner computes the shortest paths from all locations to a given destination. The LLS modules employ iterative algorithms with simple update rules and can therefore run many iterations per second. Such fast iterative algorithms are particularly well suited to the LLS because they are able to react almost immediately, can promptly accommodate asynchronously arriving information and often perform anytime computations which produce initial results very quickly and can continue to improve upon these results given more computation time.

4.1.1 Perceptual Subsystem

The main components of the perceptual subsystem are the state estimators and the image processing subsystem.

Probabilistic State Estimators Besides the localisation module [11], RHINO currently uses two different state estimators for estimating the dynamic state of objects in its environment. The first one handles objects such as doors, chairs, or wastebaskets which are permanent parts of the dynamic world model and which are subject to infrequent state changes [20]. The state density of these objects is estimated using a template matching approach which compares the real sensor measurements against ideal measurements for each sample state. These ideal measurements are obtained by a simulation of the sensor within a 3D world model.

The second state estimator used in the RHINO system is employed to estimate the trajectory of other robots or of people walking by. In contrast to the static objects handled by the first estimator, these objects are not permanent parts of the environment model and they may enter and leave the scene frequently. The state estimation method for these objects is therefore based on the detection and tracking of dedicated features which indicate their presence. Often several people have to be tracked simultaneously and sometimes their paths cross. In such a situation, the detected features need to be assigned to the objects being tracked. The estimator relies on a probabilistic data association algorithm for this purpose.

Image Processing Subsystem Making image processing dynamically configurable and run-time controlled is a necessary requirement for it to be successfully integrated within the RHINO control system. The RHINO system's visual and image processing components are constructed using a programming framework called RECIPE: Reconfigurable, Extensible, Capture and Image Processing Environment [2]. RECIPE provides a standardised, multi-threaded, environment in which modules – for image processing or other tasks – are dynamically loaded at run-time to take advantage of the robot's context. The framework provides standardised facilities for resource management and image capture to improve system robustness and to promote code re-use. RECIPE modules, as well as being run-time loadable, are scriptable, active, entities well able to mirror the requirements of the SRC. The SRC can

```
From: peters@cs.uni-bonn.de)
Date: Fri, 24 Oct 1997 12:03:57
To: rhino+tcx@cs.uni-bonn.de
Subject: Command

Could you please bring
the yellow book on the desk
in room a-120
to the library before 12:30
```

(a)

```
(REQUEST
   :SENDER (THE PERSON (FIRST-NAME Hanno)
                       (LAST-NAME Peters))
   :RECEIVER (THE ROBOT (NAME RHINO))
   :TIME (THE TIME-INSTANT 10:24:12:03)
   :REPLY-WITH (YOUR COMMAND FROM 12.03)
   :CONTENT (ACHIEVE
              (LOC (THE BOOK
                       (COLOR YELLOW)
                       (ON (THE DESK
                              (IN (THE ROOM A-120)))))
                   (THE LOC (IN (THE ROOM LIBRARY))))
   :DEADLINE (A TIME-INSTANT (BEFORE (DATE 12.30))))
```

(b)

Figure 8: The figure shows an electronic mail sent to RHINO's computer account (a) and its translation into the internal speech act representation (b).

reason about its image processing plans in the same way as any other plans. Contextual modifications calculated by the SRC are reflected in the operations delegated to the RECIPE image processing server. Creating RECIPE modules is very simple and the content of the modules is not restricted in any way. So far, RECIPE modules have been created for object recognition, gesture recognition, image segmentation, motion-based object recognition and the encapsulation of standard image processing operations.

Communication The ability to perform natural language conversation significantly enhances robots' capabilities, augmenting both their abilities to perceive their environment and to effect changes within it. For instance, using natural language a robot can ask a person to open a door which it cannot open by itself. Being able to perform natural language communication also enables robots to both receive a wider range of job specifications and also to acquire information that cannot be perceived using their sensors.

Therefore, we have extended the RHINO control system with a module that enables RHINO to send and receive electronic mails written in a very constrained subset of English. Incoming electronic mails are parsed using a simple Definite Clause grammar and transformed into an internal speech act representations (see figure 8). Object descriptions, such as the right desk in room A-120 are then interpreted using RHINO's symbolic world model, which makes task specifications, such as *pick up the red letter from the right desk in room A-120* effective. The computational processes for communication, including parsing, interpretation, sentence construction are then integrated into the plan language RPL, which provides the robot with means for controlling conversational actions. Using these plan language extensions we were able to concisely specify a wide spectrum of communication behaviour.

4.1.2 Navigation System

The RHINO navigation system as currently used is described in [8]. It works as follows. A navigation problem (a pair of locations $\langle$s,d$\rangle$ where s is the start location and d the destination) is transformed into a Markov decision problem, MDP, to be solved by a path planner using a value iteration algorithm. The solution is a mapping from every possible location to the optimal heading to reach the target. This mapping is then given to a reactive collision avoidance module which, taking the robot's actual sensor readings and the dynamics into account, employs this mapping to reach its target.

4.2 High-level Interface

While the LLS provides the basic functionalities of the robot control system in the form of continuous control processes that can be activated and deactivated, it does not provide effective means for combining the processes into coherent task-directed behaviour. Without additional functionality it would be tedious and error-prone to specify robust and efficient control programs to perform non-trivial and composed control tasks.

Accordingly, the RHINO software architecture employs HLI [12] as a high-level interface to the LLS. From the LLS point of view, HLI can be seen as a module that functions as a resource for performing non-trivial tasks more reliably, effectively, and efficiently. It does so by synchronising the operations at the LLS level in task specific ways and by recovering from local execution failures.

From the SRC point of view, HLI provides "tasks" as a convenient programming abstraction for the low-level control processes and as a reliable execution component. Using HLI, the SRC can start and terminate tasks and wait for their completion. In addition, HLI maintains the robot's current operational status as well as relevant feedback from the control processes – such as task completion signals and so on – for immediate access by the SRC.

4.3 Structured Reactive Controllers

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 anticipate 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 our discussion of 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 the 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 of concurrent reactive plans [17, 4].

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 RHINO to successfully plan its delivery tours. To be able to project their plans, robots must have causal models that represent the effects of their actions. RHINO uses causal models for predicting the behaviour generated by modern autonomous robot controllers accurately enough to foresee a significant range of realistic execution problems. This is possible because RHINO's 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.

Therefore, RHINO employs probabilistic sampling-based temporal projection methods that can infer information from such complex distributions quickly and with bounded risk [11]. RHINO's temporal projection methods solve the following prediction problem: given a set of possible plan failure modes and a risk that the robot is willing to take that its prediction is wrong, then infer whether the plan is likely to produce one of the failure modes with a probability greater than some given threshold percentage.

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. RHINO's navigation system [8], for example, considers navigation as a Markov decision problem. It models the navigation behaviour 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 [6], 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 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

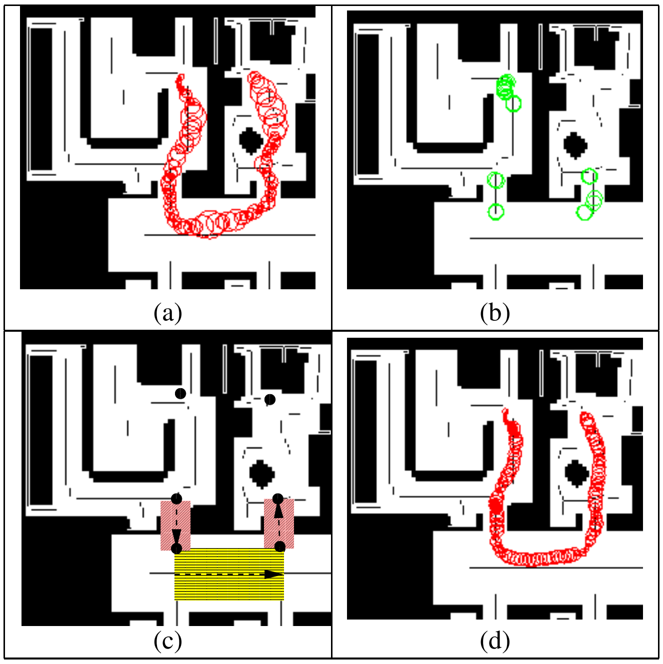


Figure 9: The figure visualizes a summary of a learning session: A behaviour trace of the default plan (a); behavior stretches where the robot moves conspiciously slowly (b); the added subplans in the learned navigation plan (c); and a behaviour trace of the learned plan, which is on average 29% faster than the default plan (d).

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 exemplary learning session is shown in figure 9.

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) [3], and (3) allow for compact and realistic symbolic predictions of continuous, sensor-driven behaviour [7].

5 Long-term Demonstrations

This section describes several experiments 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 10(a)) operated in the "Smithsonian Museum" in Washington for a period of thirteen days [25]. 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.

Figure 10: The mobile robots MINERVA (a) and the RWI B21 robot RHINO (c) that are used in the experiments.

In two other experiments we have evaluated more specific capabilities of the plan-based controller. In one experiment, we have shown how predictive plan transformation can improve the performance by outperforming controllers without predictive transformations in situations that require foresight while at the same time retaining its performance in situations that require no foresight. In a second experiment, we tested RHINO's ability to learn symbolic navigation plans. In these experiments, which involved more than 100 hours of autonomous experimentation, RHINO's learning component significantly and substantially improved the performance of the RHINO navigation system (see [6] for details).

6 Related Work

In recent years, there have been a number of longterm real-world demonstrations, which have impressively shown the potential impact of this technology for future applications of plan-based control. In NASA's Deep Space program a plan-based robot controller, called the *Remote Agent* [18], has autonomously controlled scientific experiments in the space. In the *Martha* project [1], fleets of robots have been effectively controlled and coordinated. *Xavier* [22], an autonomous mobile robot with a plan-based controller, has navigated through an office environment for more than a year, allowing people to issue navigation commands and monitor their execution via the Internet. CHIP [10] is an autonomous mobile robot that is intended to serve as a general-purpose robotic assistant, which has been applied to a variety of tasks including clean-up tasks.

All these approaches to plan-based control differ with respect to the aspects of the control problem they focus on and the assumptions they make. XAVIER employs ROGUE, a system for interleaved task planning, execution, and situation-dependent learning [13]. In XAVIER action plans are sequences of actions and reactive behaviour is generated by XAVIER's plan execution system ROGUE. In RHINO, on the other hand, reactive behaviour is specified explicitly and transparently by the plans. One of the main research goals tackled by the CHIP control system is the development of a general library of low-level and situation-driven high-level plan sketches. In contrast to CHIP's plans the plans of SRCs are additionally designed to be reasoned about and revised. Our approach to plan-based control of robotic agents differs from the FLAKEY approach [15] in several ways. FLAKEY dispenses with execution time deliberation [19] and can therefore not predict and forestall execution failures in the way Structured Reactive Controllers do. The emphasis in the development of the REMOTE AGENT is the design of a plan-based control system that controls a very complex physical system with extreme reliability.

7 Conclusions

The principles and experimental results above have provided us with a set of promising design principles for autonomous robot control systems that are to accomplish complex and changing tasks. Having control systems that dynamically adapt themselves to real-world environments and changing tasks is a very powerful approach. The additional flexibility provided by adaptive systems permits the solution of problems that cannot easily be solved by more traditional architectures. These techniques enable autonomous robots to make better informed control decisions based on reasoned predictions while carrying out their actions and they additionally ensure the continuing utility of the robot's control routines for long-term robot operation.

We have identified design principles that enable robots to achieve the desired level of adaptivity. Firstly, the robot controllers should be represented as concurrent reactive plans that specify how the robot is to react to changes in its belief state in order to accomplish its tasks. Secondly, the belief state of the robot has to be computed by a more powerful perceptual apparatus. We believe that probabilistic state estimation and dynamically configurable image processing routines are key components of such an apparatus. Thirdly, most reasoning tasks should be performed in a resource-adaptive manner. Probabilistic sampling provides a promising framework which allows control systems to balance the risks of making false inferences against the computational resources needed to arrive at decisions. Finally, to further simplify reasoning tasks, the robots should better exploit their knowledge of the task context. Modular, dynamically configurable systems like RECIPE permit us to fully exploit the contextual information.

Even seemingly simple problem-solving situations show that these capabilities may be crucial for future service robots. Our system can make a better informed decision based on a reasoned prediction about an action's possible consequences. Allocating some of its computational resources to the calculation of a new plan, the controller can continue to execute the current plan and, if it later arrives at a better one, it can then smoothly integrate the new sequence of actions into the old plan. The size of the computational resource allotted to the calculation can easily be adjusted by changing the sample size of the probabilistic plan projection.

The remarkable successes of the RHINO system are well demonstrated by the experimental results. Future work will focus on improving the robustness of the system in the face of unknown situations or hardware or software failures and on further exploiting the expressive power of the system's probabilistic models and context-sensitive calculation abilities.

Acknowledgements. We wish to acknowledge the continuing contributions of Sebastian Thrun to the RHINO system. We further wish to thank the anonymous reviewers for their valuable suggestions for improving this article.

References

[1] 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.

[2] T. Arbuckle and M. Beetz. Extensible, runtime-configurable image processing on robots — the RECIPE system. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.

[3] 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.

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

[5] 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.

[6] 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.

[7] M. Beetz and H. Grosskreutz. Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior. In

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

[8] 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.

[9] H. Endres, W. Feiten, and G. Lawitzky. Field test of a navigation system: Autonomous cleaning in supermarkets. In *Proc. of the 1998 IEEE International Conference on Robotics & Automation (ICRA 98)*, 1998.

[10] J. Firby, P. Prokopowicz, M. Swain, R. Kahn, and F. Franklin. Programming CHIP for the IJCAI-95 robot competition. *AI Magazine*, 1996.

[11] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL, 1999.

[12] D. Hähnel, W. Burgard, and G. Lakemeyer. Golex - bridging the gap between logic (GOLOG) and a real robot. In *Proceedings of the 22nd German Conference on Artificial Intelligence (KI 98)*, 1998.

[13] K. Haigh and M. Veloso. High-level planning and low-level execution: Towards a complete robotic agent. In L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 363–370, Marina del Rey, CA, February 1997. (New York, NY: ACM Press).

[14] S. King and C. Weiman. Helpmate autonomous mobile robot navigation system. In *Proceedings of the SPIE Conference on Mobile Robots*, pages 190–198, Boston, MA, November 1990. Volume 2352.

[15] K. Konolige, K. Myers, E. Ruspini, and A. Saffiotti. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

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

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

[18] 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.

[19] K. Myers. A procedural knowledge approach to task-level control. In B. Drabble, editor, *Proceedings of the Third International Conference on AI Planning Systems*, pages 158–165, Edinburgh, GB, 1996. AAAI Press.

[20] D. Schulz. Template-based state estimation of dynamic objects. In *Procs. of the EUROBOT Workshop*, 1999.

[21] 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.

[22] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A modular architecture for office delivery robots. In *Proceedings of the First International Conference on Autonomous Agents*, pages 245–252, 1997.

[23] 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.

[24] 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.

[25] 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.

[26] N. Tschichold-Gürman, S.J. Vestli, and G. Schweitzer. The service robot MOPS: First operating experiences. *Robotics and Autonomous Systems*, 34(2-3).