The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Autonomous Planning for Mobile Manipulation Services Based on Multi-Level Robot Skills*

Martin Weser and Jianwei Zhang
TAMS - Technical Aspects of Multimodal Systems
Department of Informatics, Hamburg University
Vogt-Kölln-Strasse 30, 22527 Hamburg, Germany
{weser,zhang}@informatik.uni-hamburg.de

***Abstract*— General purpose service robots are expected to deal with many different tasks in unknown environments. The number of possible tasks and changing situations prevent developers from writing control programs for all tasks and possible situations. Complex robot tasks are thus accomplished by sequential execution of less complex robot actions that are triggered and configured by a task planner. The question of the appropriate abstraction level of robot actions is still being researched and not discussed conclusively. In this paper, we address the problem of atomicity of robot actions and provide some key properties that have to be considered while designing plan-based robot control systems. Based on these properties, we define and implement atomic skills of different abstraction level for the service robot TASER. The HTN planner JShop2 is used to complete the plan-based control architecture which is evaluated in a set of experiments.**

I. INTRODUCTION

Robots are often built for specific tasks in a controlled environment. In this case research aims at increasing robustness, accuracy, safety and execution speed. Service robots, in contrast, aim at flexibility and versatility. Instead of increasing the performance of single robot tasks, researchers investigate diverse robot behaviors that are to be executed on a single robot system. Robot mobility and manipulation services, for example, demand for completely different control strategies. Our investigations with the mobile service robot TASER, which is shown in Figure 1, strive for integrating these different application areas into one coherent system. One way to tackle this is simply to load several task-specific programs on the system that can be executed selectively. However, with an increasing number of tasks this method quickly hits the wall. Maintenance of control programs becomes laborious and development of new behaviors requires repetitive implementation of similar subroutines. Thus, researchers in multi-purpose service robotics rearranged robot tasks into more manageable units that we call robot *skills*. Robot skills specify either atomic robot *actions* or simple robot *behaviors* which are used synonymously in this paper. Each robot skill is implemented in a *control program* that directly accesses sensors and actuators to achieve the expected behavior. A *task-planner* combines control programs to achieve complex robot tasks.

*This work is founded by the DFG German Research Foundation (grant 1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems)

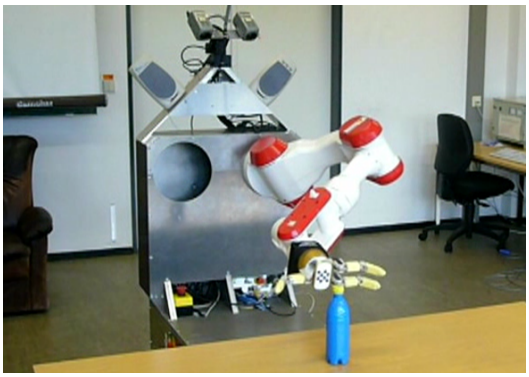

Fig. 1. The mobile robot TASER executing a manipulation task.

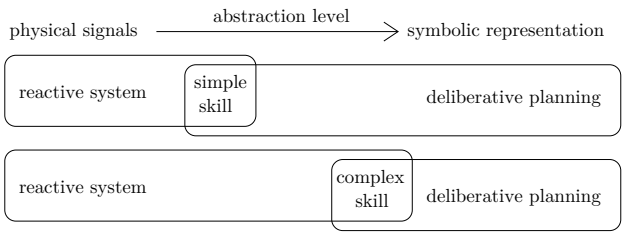

Fig. 2. Deliberative planning and reactive control communicate via atomic robot skills. The combined complexity of both layers is independent of the abstraction level of skills.

To ensure high reusability in a variety of robot tasks, the control programs have become more and more basic. Reduced complexity of robot control programs, in turn, leads to increased complexity in the planning component. The complexity of the overall system, which comprises both the control programs and the planning component, remains unchanged. Figure 2 shows the relation of robot planning and control using simple and complex skills respectively.

The complexity of robot skills defines the abstraction level at which the deliberative component interfaces the reactive execution. The question on the granularity of robot skills has to be considered in every hybrid robot architecture. Surprisingly, to the best of our knowledge it has not been addressed explicitly in the literature. To quote [1, p.207],

> the nature of the boundary between deliberation and reactive execution is not well understood at this time, leading to somewhat arbitrary architectural decisions.

We find a strong affinity to the symbol grounding problem (SGP) that deals with the right way to relate symbols with perception and action [2]. This paper, in contrast, focus on the question which symbols are suitable for application in a hybrid architecture. While assuming that the SGP has little relevance for applied robotics due to application-specific solutions, this paper aims at increasing understanding of the boundary between deliberative and executive control.

The contributions of this paper are 1) analyzing requirements for atomic skills that put constraints on their abstraction level, 2) redefining skills for a service robot scenario according to the constraints given earlier, and 3) evaluating the implemented robot actions in integrated experiments with the mobile manipulator TASER. We believe these contributions help bridge the gap between symbolic planning and reactive execution.

The rest of this paper will proceed as follows. The next section reviews the state of the art in plan-based robot control with focus on the boundary between deliberative and reactive parts. The granularity of atomic robot skills is analyzed in Section III. This section provides criteria to be considered in the design process of robot software systems. We define and implement a set of robot skills in Section IV. Section V evaluates the feasibility of our approach by integrating the symbolic planner JShop2 with the implemented skill library. We conclude with a summary and a discussion on future work in Section VI.

II. State of the art in plan-based robot control

The *plan based robot control* paradigm can be considered as state of the art in autonomous robot systems [3]. It combines reactive robot control [4], [5] and deliberative reasoning to symbolic planning systems that utilize robot skill libraries. Plan-based robot control systems are often realized in a layered architecture. A distinguishing feature of these architectures is the abstraction level at which the two layers communicate (Figure 2). The abstraction level of robot skills is also a decision of the appropriate representational modality for a given problem. Which problem should be solved in continuous space (e.g. sensor data processing, coordinate transformation, trajectory planning) and which should be solved in symbolic terms on the planning level? In the following we review selected hybrid robot systems with respect to the abstraction level on which the reactive and deliberative layer interface.

The authors of the interactive tour guide robot MINERVA [6] distinguish four layers in its control architecture. High-level control and human interaction constitute the deliberative layer. Their common purpose is planning and monitoring of museum tours and user interaction. Other parts such as localization, navigation, mapping and interactive routines that attempt to realize a "believable social agent" can be considered as the execution layer since they are optimized routines with a specific purpose. The high-level interface communicates with the rest of the software using HLI, a component of GOLEX [7]. MINERVA operated for two weeks in a natural and dynamic environment.

The autonomous mobile robot GRACE [8], developed at Carnegy Mellon University and others, has attended the AAAI robot challenge in 2002. As several research groups were involved in the development of GRACE, they utilize a kind of odd architecture that allows each group to use its own deliberative component. A common symbolic planner is not described in the literature. However, all task-dependent modules share the use of basic robot skills that are provided by dedicated modules. These modules together constitute the executive layer. Perception and action routines include localization in a dynamic environment, safe navigation in the presence of moving people, path planning, visual tracking of people, signs, and landmarks, gesture and face recognition, speech recognition and natural language understanding, speech generation, and social interaction with people. The complexity level of the actions that can be used by GRACE's deliberative components reaches from simple (e.g. speech generation) to highly complex (e.g. gesture recognition).

The planning layer and the sensor-actor layer in the robot ARMAR-III [9] is mediated via a synchronization and coordination layer. The planning layer decomposes abstract tasks into sets of subtasks. The middle layer then invokes the proper robot skills sequentially or in parallel to achieve the subtasks [10]. The reactive layer specifies control commands for the robot's devices as well as dedicated algorithms. The implemented robot skill library includes visual environment perception (3D hand and face detection and tracking), speech recognition and attempts to manipulate objects.

The University of Munich developed an early prototype of a mobile service robot for health care and domestic automation services named ROMAN [11]. In this semi-autonomous system, the robot receives commands by a human instructor. These are split up into basic task primitives that can be executed by control programs. The reported basic primitives, e.g. door opening, are rather coarse. However, the principle of executing a sequence of control programs in order to achieve complex goals is similar to the other approaches.

A discussion of all layered robot systems would clearly go beyond the scope of this paper. The selected systems, however, already show that the decision on the abstraction level of the boundary between layers is a matter of design choice, and the solutions proposed in the literature differ significantly. In the case of complex control programs, the planning component is rather simple [11]. If control programs become more basic, the planning component becomes an immanent part of the system [6], [8], [9].

The following section throws light on this issue by analysing arguments for different abstraction levels leading to some key considerations in designing layered robot architectures.

III. Granularity of atomic robot skills

Why do robot skills have to be atomic and how can atomic skills differ in their granularity? The answer is simple: They do not have to be atomic from the control perspective. A robot skill does naturally implement atomicity from the perspective of symbolic planning. At the point where the

particular robot controller is called to execute an action, the planner cannot decompose it further. If an action is composed of two or more simpler actions, the planner does not invoke the complex action, but the sequence of simpler actions that together achieve the aspired behavior. Admittedly, most actions and behaviors are parameterizable (positions, objects, speed etc.), still the generation of sequences of motor commands that lead to physical effects is invisible to symbolic processes. Thus, the question is not whether an action is atomic, but on what level of complexity the intersection of symbolic planning and physical execution should be.

The decision about which problems are to be handled using automated planning and which ones are to be managed by robot control programs is *ultimately a matter of design choice* [12]. Although the question of balancing the complexity is inherent to every plan-based robot architecture, it has not been addressed individually in the literature. We cannot provide a conclusive answer to that question either. Instead this paper aims at providing arguments that have to be considered in the design process of layered robot architectures.

A. Arguments for simple robot skills

1) **Reuse:** a robot skill should be applicable in as many scenarios as possible. The simpler the skills are, the higher the chance to be used in different tasks.
2) **Generalizability:** a generic skill description on symbol level is to be preferred to allow simple reuse of actions in the planning process.
3) **Unambiguity:** an action must be unambiguous in terms of its effect and executable without further interpretation. How the effect will be achieved may depend on the current external conditions.
4) **Consistency:** an action should provide information to keep the symbolic representation of the world consistent with reality. This requirement depends on the sensor capabilities of the robot and holds for successful execution as well as for all kinds of possible failures.
5) **Recoverability:** usually it is assumed that atomic actions are either completed successfully or the state of the world will be unchanged. In embodied robotics this is hard to achieve since uncompleted actions require opposed actions to restore the foregoing situation. For this reason, we propose that an action has to report its state of execution to the symbolic layer to ensure the consistency of the symbolic world representation.
6) **Verifiability:** the completion of an action's effect should be verifiable by robot sensors.
7) **Planning flexibility:** in case of too complex robot skills the planner will loose its flexibility to react to unforeseen situations and tasks. If only a couple of complex skills are stringed together there is no use for complex planning.

B. Arguments for complex robot skills

1) **Efficiency:** implementation of robot skills can be highly optimized. The larger the chunks of work to be done in a single skill, the larger the possibility for optimization.
2) **Hardware independence:** hardware independence can only be achieved to a certain degree. However, it becomes more difficult if skills are of low abstraction. Grasping an object, for example, can be done with different grippers by executing specific motion sequences. If the motion sequence is provided by the planning layer, it may not be applicable to other grippers.
3) **Task-specific algorithms:** a general purpose problem solver should only be used for problems that can not be solved efficiently by task-specific algorithms. Compared to task-specific algorithms, general purpose planning is rather slow. Inverse kinematics and geometric path planning are examples of problems that should not be solved at planning level.

To the argument (B2) one should add that the implementation of an action is of no relevance for the planning layer. An action should be defined as an interface that is implemented in robot control programs. While deciding the granularity of atomic actions, one should keep in mind that the physical execution may change.

C. Summary of properties for atomic robot actions

The previously discussed issues have to be considered while designing robot actions for a layered plan-based robot control architecture. We explicitly point out that these requirements are guidelines rather than hard constraints. Actually it is impossible to fully satisfy all of them, e.g. verifiability (A5) excludes independence of robot hardware (B2).

The relatively few arguments for complex robot skills give the impression that a general tendency to simple robot skills is desirable. In fact, if we assume a perfect planner that has the capability of coding the robot behaviors on the fly for every robot task in every situation, we could define native hardware commands as atomic robot skills. Obviously, such a planner is not available. The capabilities of the deliberative planning component are supposably the bottleneck of the overall system, thus the arguments on efficiency (B1) and task-specific algorithms (B3) are of major importance.

IV. Design and implementation of a robot control library

The first implementation of plan-based robot control on the robot TASER resulted from the need to combine several demo-applications (i.e. control programs) to more complex robot tasks. A simple symbolic description of the actions performed by the control programs was sufficient to let a symbolic planner generate sequences of actions that achieve complex goals. A list of initially implemented robot actions is shown in Table I.

The control programs that where implemented as initial prove of concept were not specifically designed to be used as atomic operations in a task planner. Hence, they potentially violated the properties of atomic robot actions that are listed in Section III. The demand for hardware independence (B2), for example, has been neglected in all sensor-specific `find`

TABLE I

INITIALLY IMPLEMENTED ROBOT ACTIONS. THE GRANULARITY OF THE ACTIONS WAS CHANGED LATER ACCORDING TO SECTION III.

focus on action	focus on perception
dock on table|door|corner	find table in laser
pick|place rubbish can	find table in camera
pick|place from surface	tactile sensing of surface
straight mobile motion	find cup on table
path planning	find door in laser
follow person	find door in omni. camera
open door	check if door is closed
use light switch	find rubbish can
detect and track people	detect and track people

actions. Reuse of skills (A1) is not considered, as several pick and place actions implement similar movements.

To provide a strong foundation for future research and experiments, we redefined the set of robot actions according to Section III, specifically we focus on reuse (A1) and generalizability (A2). Most of the criteria cannot be verified using numerical measurements, still we empirically follow them as much as possible in the design process.

The robot action `open_door` is used in the following to describe the process of task-decomposition into atomic robot skills. Skills are defined in symbolic terms using a PDDL-style syntax that does not need to be explained further.

A. Design of atomic actions for door opening

The first necessary step is to approach the door with the mobile platform. This skill is defined in a parameterized way `(approach ?obj ?actuator)`, thus it can be used by the planner in different contexts (A2) such as reaching for objects with a manipulator. Instantiated with `mobile` and $\texttt{door}_i$, this action uses a dedicated path planning algorithm in combination with the robot's self-localization to alleviate the symbolic planner (B1,3).

To allow fine manipulation, it is necessary to state the relative position of the door to the robot precisely. The generic action `(specify_pose ?obj ?sensor)` is used for both, finding the exact position of the door in the laser range data and finding the doorknob in camera images. The latter requires the camera to be focused to the expected position of the doorknob using the general action `(focus_on ?obj ?device)` instantiated with a pan-tilt unit as device.

If laser range scanners are used to find objects, an EM-based algorithm is applied that iteratively matches the 2D shape provided by the object to a subset of measured scan points. Figure 3 shows the estimated position of a door before and after the algorithm was applied. If cameras are used as sensors, several object detection methods can be applied. In our implementation we use a color-based approach to detect the object in image coordinates. Previous knowledge on the surface on which the object is located (table-top, wall, door) and the position and orientation of the calibrated camera enables computation of the three-dimensional object position while avoiding computationally expensive stereo analysis.

The position of the doorknob is sufficient to control the manipulator relative to it using the `(approach ?o ?a)`

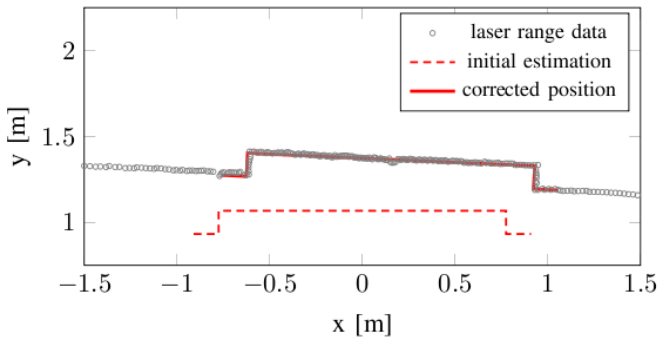

Fig. 3. Perceiving the door position: the initial estimation (dashed red) is matched to the laser range data (gray circles).

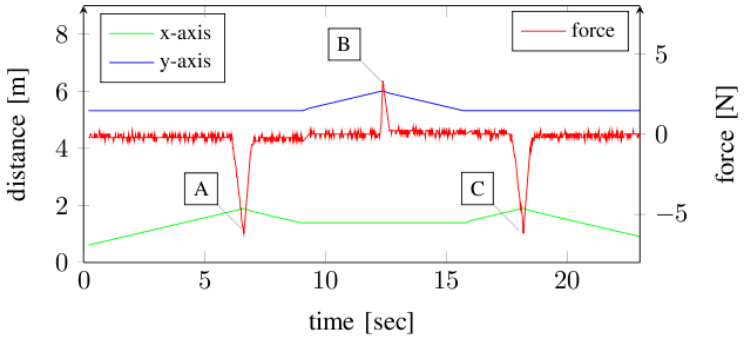

Fig. 4. Hand position and force values while the robot opens a door. First, the robot measures the distance of the door (A), then it pushes the doorknob until a force threshold is reached (B). The force measured in (C) signals that the door is still closed, thus it is assumed that the door is locked.

skill. Using the doorknob, as a specialized action, is defined as `(open ?doorknob ?manipulator)` without consideration of generalizability. We decided to use a task-specific definition rather than defining a general `use` action due to low expected code reuse in the implementation and – more importantly – to ensure unambiguity in the execution layer (A3). Transitions among sub-trajectories are triggered by force values measured at the end effector. The relation of force values and arm trajectory is shown in Figure 4. Success or failure is measured using force-sensors and a comparison of the expected and executed manipulator trajectory (A4,6). The opening task analysed here finishes with pushing the door to open it `(push_surface ?surf ?manipulator)`. Again, this definition is generic and can be used in other contexts, such as haptic detection of a table top or other furniture as done in [13].

The redesigned atomic robot actions can – except for `(open ?d ?m)` – used in several contexts. The strongest violation of the discussed properties from Section III may be hardware-independence. However, if the sensors for e.g. detecting a door are not available to the system it renders this particular example task meaningless.

B. Relating symbols for robot skills with reactive execution

Each defined robot skill has to be implemented in a control program, that integrates knowledge from a world model with current sensor readings, to generate executable control sequences for related robot actuators.

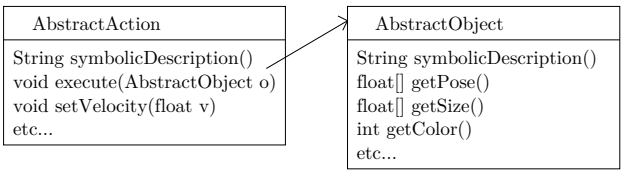

Fig. 5. Symbolic and continuous representation of skills and objects is done via shared data structures.

Since most robot skills are executed in relation to objects, these are represented in symbolic terms as well. The problem of relating symbols to external objects is known as the symbol anchoring problem [14]. In the work described here, a pragmatic approach is used that links symbols and analog descriptions of an object's physical properties in shared data structures. This is schematically shown in Figure 5.

The demand for hardware independence (B2) presupposes that the perception process for objects is bound to the robot control program, not to the object. How an agent can interact with a physical object is dependent on the agent's manipulative and perceptive capabilities rather than on the properties of the object. For example, the recognition process for `(specify_pose ?o ?s)` is defined in the sensor implementation rather than in the object's representation. In this way, we allow simple exchange of robot skills without changing the representations of the objects.

The bottom line is that physical properties attached to a symbol have to be either specifically designed for one kind of agent or they have to include all potentially useful information i.e. a complete physical model. The latter is obviously not possible. Thus, in practice symbols are enriched with just enough information to be perceived by the targeted robot platform.

V. Integrated experiments and results

In this section, we describe how we integrate the planning system JShop2 [15] with previously defined robot skills and report results from experiments.

A. Integrating planning and robot control

We chose JShop2 as planning component due to simple definition of problem and domain files, its provable good performance, and – most importantly – the possibility to define basic actions at multiple abstraction levels. The planning process performs task-decomposition according to a hierarchical task net, which is provided by the domain file. It may generate short sequences of complex actions or longer sequences of simple actions, depending on the given task. Figure 6 depicts the implemented overall architecture of action sequence generation and execution. In the symbolic layer, the planner generates plans based on the current state and an abstract task, which is provided by a human operator. The plan consists of robot skills taken from the skill library whose design and implementation has been discussed earlier. To execute the plan physically, control programs are invoked accordingly. Executed control programs access and modify the current state of the world in symbolic terms as well as continuous representations to ensure consistency.

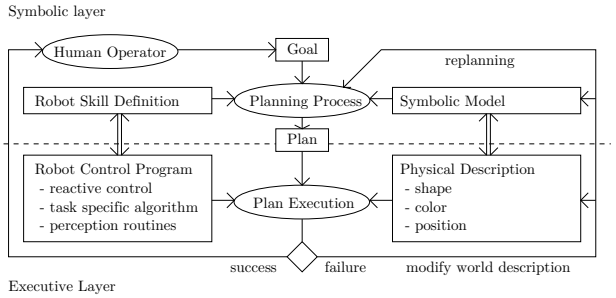

Fig. 6. Architecture of the layered control system.

TABLE II
Robot tasks and number of required atomic robot skills.

Robot task	# (skills)
pick up rubbish-can	5
open door	8
grasp cup from table	6
place cup on table	6
press light switch	5
change room	3

Experiments showed that failures in skill execution are mainly caused by world representations that are inconsistent with reality. For example, if the planner lets the robot pick up a cup that is actually not there, the skill will fail. In such cases, the control program will change the world state independently and cause replanning with the updated state.

The closed-world assumption is underlying every planning process in JShop2 but cannot be guaranteed in real-world applications. To overcome this, we introduced an atomic skill that causes replanning explicitly. This enables the planner to generate plans that include replanning based on deliberately acquired additional knowledge.

B. Results

In a first line of experiments, we analysed a set of prototypical tasks for autonomous service robots as proposed in the literature. We defined atomic actions for six tasks taking the arguments from Section III into account. The domain description for the planner described only the decomposition and execution of the prototypical tasks. To compare the complexity of task decomposition and execution, in the initial situation the robot was in the laboratory, clear of other objects, and its manipulator was empty in park position. The number of skills that are required to achieve the tasks are shown in Table II. The "change room" task assumed that the door is already open. This turned out to be the simplest task, the skill sequence is: `(approach` $\texttt{door}_\texttt{i}$ `mobile)` - `(specify_pose` $\texttt{door}_\texttt{i}$ `laser)` - `(approach` $\texttt{room}_\texttt{k}$ `mobile)`. Accomplishment of all tasks requires execution of 33 atomic skills. The domain description of the symbolic planner consists of 6 skills, that is a reuse factor of $\frac{33}{6} = 5.5$ in the symbolic domain. Admittedly, the implementation of `(approach ?o ?m)` and `(specify ?o ?m)` depends on the instantiated modality. Thus, the reuse factor in the executive layer is slightly lower.

Other criteria from Section III are hard to measure. Consistency (A4), for example, has to be implemented in control programs. Independence of robot hardware and control programs (B2) is supported as much as possible due to the representation of modalities as a symbolic variable. This initial experiment did not include ambitious tasks in complex environments. Symbolic planning was reduced to a minimum if present at all.

In a second line of experiments, we investigated the applicability of the overall layered architecture as shown in Figure 5. We expand the symbolic domain description to complex applications such as pick and place among rooms with closed doors, hidden objects, and acting under incomplete or false knowledge about the environment. The resulting plans for such tasks easily consist of more than hundred atomic actions of different complexity.

Experiments showed that the planned sequences of atomic actions could be executed without human intervention. Exceptions in control programs led to controlled abort of plan execution and replanning based on the changed state of the world. Especially in unknown environments, the robot's ability to actively cause replanning turned out to be fruitful. For example, if the robot should pick up an unknown cup it can search for it actively. If no cup was found, the planner can decide how to behave next, e.g. search for it in another room or change the light conditions by switching the light on or off.

VI. Conclusion and discussion

Every deliberative and reactive hybrid system has to decide which problems are to be solved in symbolic terms and which within reactive controllers. However, the level of complexity of basic robot actions has rarely been addressed in the literature. The gap between robot action execution on the one hand, and deliberative planning systems on the other is still an area of active research. We believe that our discussion on these issues is an important contribution to the development of versatile robots that possess symbolic reasoning capabilities. The discussion on the granularity of robot actions is certainly not conclusive, still our approach is of practical nature and provides guidelines to efficiently implement actions for autonomous robot systems.

We are aware that some people claim that a distinction between deliberative and executive layer is outdated and that integrated planning and control is to be preferred. For two reasons we still believe that a layered system is reasonable:

1) The separation of the two layers is not strict. The layers are coupled via object models that provide both, symbolic descriptions as well as rich continuous representations to be used for perception and physical manipulation. The reactive implementation of action and behavior is not separated from the symbolic world description. Actions modify the symbolic representation while the external world is perceived and manipulated. Furthermore, the two layers are not hierarchically organized in control of the overall system: an action can cause the planner to generate new solutions while the planner causes actions to be executed.
2) The focus of this work is applicability and practicability. Other approaches may be cognitively more plausible or more promising for advanced embodied intelligence, however we achieved good results with reasonable effort. The layered architecture effectively supports debugging, maintenance and extension. Robotics is not only a tool for AI research, it also focuses on practical and robust solutions that will lead to products accessible for a wide range of applications. The approach developed in this work promises advanced robot behavior and a short time to market with technologies that are both state of the art as well as sophisticated and robust.

The need for better understanding of the boundary between symbol and control level is part of a higher goal to build intelligent embodied robots. Our future work aims at endowing the robot with more atomic skills. We consider both hand crafted implementation and autonomous learning.

References

[1] Ronald C. Arkin. Behavior-based Robotics. *MIT Press*, 2 edition, 1998.

[2] Stevan Harnad. The symbol grounding problem. *Physika D*, 42:335–346, 1990.

[3] M. Ghallab, R. Alami, J. Hertzberg, M. Gini, M. Fox, B. Williams, B. Schattenberg, D. Borrajo, P. Doherty, J. M. Morina, A. Sanchis, P. Fabiani, and M. Pollack. A roadmap for research in robot planning, 2006.

[4] Valentino Braitenberg. Vehicles. Experiments in Synthetic Psychology. *MIT Press*, Cambridge, Mass., 1984.

[5] Rodney A. Brooks. A robust layered control system for a mobile robot. In *ICRA Int. Conf. on Robotics and Automation*, volume RA-2, pages 14–23, 1986.

[6] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. 1999.

[7] D. Hähnel, W. Burgard, and G. Lakemeyer. Golex - bridging the gap between logic (golog) and a real robot. In *KI '98: Annual German Conf. on Artificial Intelligence*, pages 165–176, 1998. Springer-Verlag.

[8] R. Simmons, D. Goldberg, A. Goode, M. Montemerlo, N. Roy, B. Sellner, C. Urmson, M. Bugajska, M. Coblenz, M. Macmahon, D. Perzanowski, I. Horswill, R. Zubek, D. Kortenkamp, B. Wolfe, T. Milam, M. Inc, and B. Maxwell. Grace: An autonomous robot for the aaai robot challenge. *AI Magazine*, 24:51–72, 2003.

[9] T. Asfour, K. Regenstein, P. Azad, J. Schrder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An integrated humanoid platform for sensory-motor control. In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006.

[10] T. Asfour, D.N. Ly, K. Regenstein, and R. Dillmann. Coordinated task execution for humanoid robots. In *Experimental Robotics IX*, volume 21, pages 259–267. STAR, 2005.

[11] U. D. Hanebeck, C. Fischer, and G. Schmidt. Roman: A mobile robotic assistant for indoor service applications. In *IROS Int. Conf. on Robotics and Systems*, 1997.

[12] E. Beaudry, F. Kabanza, and F. Michaud. Planning for a mobile robot to attend a conference. In *Canadian Conf. on Artificial Intelligence*, pages 48–52, 2005.

[13] Martin Weser and Jianwei Zhang. Proactive multimodal perception for feature based anchoring of complex objects. In *ROBIO Int. Conf. on Robotics and Biometrics*, 2007.

[14] Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003. Special issue on perceptual anchoring.

[15] D. Nau, T. C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *Journal on Artificial Intelligence Research*, 20, 2003.