A Framework for Mobile Manipulation

Lars Petersson and Henrik I. Christensen[⋆]

Computational Vision and Active Perception
Numerical Analysis and Computing Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Abstract. In this article we present a proposal for a control architecture suitable for mobile manipulation. The approach attempts to combine existing techniques for navigation and mobility with a flexible control system for the manipulator arm, hence a behavior-based system for controlling the platform is integrated with a hybrid dynamic system for the manipulator control. An analysis is made of the system characteristics, motivating these choices. The objective is to have a system level design where all aspects can be included as a natural part, e.g. mission planning, task analysis, path planning, control and integration. Also, using a hybrid dynamic systems paradigm, methods exist for stability, controllability and observability analysis. A language for specification of hybrid automata is developed to support a structured design.

1 Introduction

Mobile platforms are in general use today for a wide range of applications such as automated warehouses and commercial cleaning. In most applications the interaction with the environment has been engineered to make the task well-defined and easier. For tasks that are more general it might not be possible or it may be inconvenient to use engineering of the environment, i.e. for assistance to handicapped, for operation in homes and offices. To enable use of mobile platforms for such domains it is of interest to use “mobile manipulators”, as general purpose manipulators in diverse application domains. Mobile manipulation is expected to enable a leap in performance and diversity for robotics. This has recently lead to an increased interest in mobile manipulation.

The Stanford team, headed by Khatib [8, 9, 7], has reported significant new results in this area, in particular in terms of adaptive path planning for redundant systems and simple mobile interaction. Another notable effort is the work at Technical University of Munich [5].

Mobile manipulation involves mobility and manipulation. Traditionally the two aspects have been treated separately. In most of the currently available mobile systems the control is also primarily separated. I.e., in the ROMAN system from TU Munich, the manipulation is only carried out when the platform is stationary. To achieve full flexibility there is however a need to integrate the two systems. For the integration it is essential to carefully analyze the characteristics of the overall system to provide a reasonable balance between the two systems. To achieve a system level design it is essential to consider all aspects including mission planning, task analysis, path planning, control and integration. The implementation of all of these components requires a solid framework, that facilitates efficient implementation using theoretically well founded methods. A possible framework for mobile manipulation is presented in this paper.

The characteristics of manipulation and mobility/navigation is initially analyzed to provide a basis for design of an overall system. Based on the analysis a “mixed” architecture is suggested. The architecture uses a behavior based approach to navigation, while manipulation is implemented using a hybrid dynamic system. The integration of the two paradigms is achieved using an observer structure. The analysis and design is described in section 2. The “mixed” architecture has been implemented as an Object-Oriented Pseudo Real Time system, that provides a flexible basis for implementation of a rich variety of tasks. The implementation is described in section 3. An example task that illustrates the “implementation” of mobile manipulation tasks is described in section 4. Finally a summary and issues for future research are described in section 5.

[⋆] Email: {larsp,hic}@nada.kth.se

2 System Design

For the design of a suitable framework for mobile manipulation it is essential to consider the characteristics of the two systems. In a typical system the mobility system and manipulation may be used separately or together. It is for example convenient to be able to use the mobility system for regular navigation between work places with little consideration for the manipulation system, likewise is it of interest to be able to use the manipulator as a regular system, when the base is stationary. In addition it is of course necessary to have the possibility of joint task specifications to enable true mobile manipulation. An abstract view of the system is shown in figure 1.

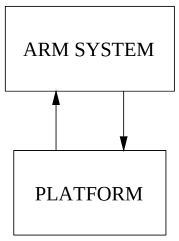

Fig. 1.: Abstract model of the mobile manipulation system

2.1 Analysis

We have chosen to consider the aspects of dynamics, dominant control paradigm, resource structure, and control specification for our design of the system.

The dynamics of mobile platform is typically limited due to the inertia of the platform. Typically the platforms are velocity or position controlled at a rate of 5-25 Hz. Lower-level PID controllers might operate a higher rates, but they are typically inaccessible to the programmer. This sampling rate is adequate for normal navigation tasks including obstacle avoidance, docking, and positioning. Manipulation on the other hand is typically carried out at sampling rates of 200 Hz - 5 kHz. Interaction with objects implies that even small perturbations can results in significant contact forces, which motivates the need for a significantly higher bandwidth of the closed loop system.

Mobile platforms are typically controlled in polar coordinates $\mathbf{p} = (r, \alpha)^T$, where r is the distance and α is the orientation. The control specification is typically velocity based, i.e. $\dot{\mathbf{p}} = (v, \dot{\alpha})^T$, in terms of translational and rotational velocity. Often the two dimensions are controlled independently or at least only loosely synchronized. There might be different behaviors in charge of control of each of the degrees of freedom and the loops might only be connected through sensory feedback. In contrast manipulation (typically at least 6 degrees of freedom) requires synchronized control of all joints, as the task typically is specified in Cartesian coordinates using a velocity screw $(T = (\dot{x}, \dot{y}, \dot{z}, \dot{\rho}, \dot{\delta}, \dot{\gamma})^T$. Using inverse kinematics the control specification is transformed into joint coordinates, $\mathbf{q}$. The implementation of the control specification requires synchronized control of all axes.

The control specifications outlined above implies that the resource allocation strategies are quite different. In manipulation the control system has exclusive access to the physical system. In most mobile robots, systems have shared access to the physical system, i.e. no single behavior is awarded exclusive access to the system (all the time).

The dominating control paradigms used by the two systems are also quite different. In mobile robotics one of the dominating paradigms used today is Behavior-Based Systems (BBS) [3]. Given the need for synchronized control of all degrees of freedom the dominating control paradigm for manipulation is multivariate control. Given a desire to be able to exploit existing control software the different paradigms poses a challenge.

The characteristics outlined above can be summarized by Table 2.

System characteristic	Arm system	Platform system
Response time	5-10 ms	50-100 ms
Paradigm	Multi variate control	Loosely synchronized control
Resource control	Exclusive	Shared
Control	HDS	Behavior based

Fig. 2.: Major characteristics of the mobility and manipulation systems

2.2 Integration

A framework for mobile manipulation should preferably retain the inherent characteristics for each of the two sub-systems while at the same time providing a basis for coordination.

From a task control point of view a noble manipulation system represents a redundant control system, in the sense that most combined control is specified in terms of motion of the end-effector. Given a three degree of freedom (holonomic platform - $\mathbf{p} = (x, y, \alpha)^T$) and a six degree of freedom manipulator ($\mathbf{q} = (q_1, \ldots, q_6)^T$, the system has up to three redundant degrees of freedom. The dynamics of the two systems are very different as pointed out above. A possible approach to the control is the provide separate control for each of the two systems, with an added coordination component.

For the mobility system it is possible to obtain information about manipulator state, $\mathbf{q}$, on-line due to the higher sampling rate. The control of the manipulator can thus be considered directly in the control of the platform, with no or little loss of generality. Given a desire to exploit existing software it is convenient to use a behavior based approach to control, in which the manipulator simply is considered yet another behavior that contributes to the control of the platform, or rather it constrains the possible motion of the platform. The control system can then be considered composed of four components: i) sensors, ii) actuators, iii) behaviors and iv) behavior fusion/scheduling. Sensors and actuators can easily be modeled as resources, that are serviced by servers. Behaviors are independent control methods that integrate sensing and action, i.e they are clients that utilize the resources. To ensure overall coherence, the output of different behaviors are combined using scheduling or fusion methods. The definition for such control system are widely described in the literature and a good overview can be found in [3].

For the manipulation system it is impossible to obtain timely information about the motion of the platform, due to the lower sampling rate for control of the platform. To enable compensation for platform motion it is possible to introduce a task oriented observer. The observer receives information about the control command executed by the platform and interpolates between samples to provide an estimate of the platform motion. One problem to consider in the design of a control system for the manipulator is that the dynamics of the system changes considerably based on the task. I.e. during pre-grasping the system has a dynamics determined by the mechanics of the arm, while manipulation requires inclusion of the object manipulated into the control and during constrained tasks like door or drawer opening the motion is constrained and the physical constraints must be considered. To enable easy handling of the different manipulation "modes" it is convenient to adopt the paradigm of Hybrid Dynamic Systems [12, 4]. A hybrid dynamic system (HDS) is here specified by a 7-tuple:

$$H = (Q, X, I, f, E, G, T)$$

where

- Q is the set of discrete variables, i.e. different states
- X is a set of continuous variables, i.e. the joint coordinates
- $I \subset Q \times X$ is the set of initial states of the system
- $f : Q \times X \rightarrow TX$ is a vector field that specified the control of the system. I.e. the state dependent control laws
- E is the set of discrete events
- G is a set of guards that trigger discrete transitions
- $T : Q \times (E \cup G) \rightarrow Q$ is the transition matrix that maps event in to state changes

The advantage of HDS is that existing methods for analysis of systems in terms of stability, controllability and observability can be used. These tools provides a suitable basis for analysis and design of systems.

A hybrid automata specification can also be used for specification of the observer. In that case the term f specifies the control law of the platform for a specific command specified by a particular command (in the set Q). For simple cases the observer can simply be an n-order Taylor approximation of the platform dynamics, where the operating point is specified by the motion parameters ($\dot{\mathbf{p}}$).

3 Implementation

The system outlined above has been implemented on a Nomadic XR4000 platform with a PUMA 560 manipulator mounted on top. The system is shown in figure 3.

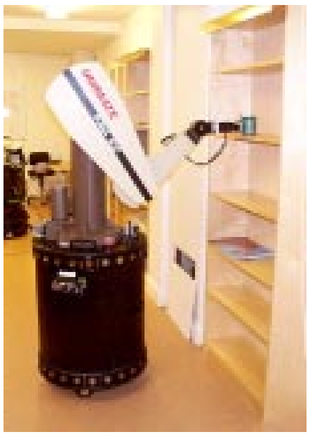
Fig. 3.: The XR4000 platform with a Puma560 manipulator.

The two systems (mobility and manipulation) are implemented on different computer systems to accommodate the quite different computational demands. In the system, there is also a third computer dedicated to computer vision (see Figure 4)

i) Computer #1 is dedicated to sensing in general and computer vision in particular. Computer vision generally consumes a lot of computing power which explains this choice. A typical response time for tasks residing on this computer is approximately 25ms, i.e. the operating system Linux will have sufficient timing capabilities.
ii) Computer #2 processes the data from the sonars, the laser range sensor, the infrared sensors etc. The localization of the platform and control of the mobility are the tasks of this computer. The system operates at a sampling rate of about 10 Hz. Given that the system runs under vanilla Linux the control is carried out in pseudo real-time.
iii) Computer #3 is dedicated to control of the Puma560 and processing of sensory data directly related to manipulation tasks (except vision). In this case, it is necessary to use a real real-time operating system in order to control the very fast dynamics. The choice of real-time operating system is QNX [14], which is a Unix-like environment.

The mobility system for control of the platform is implemented as a standard behavior-based system. The overall system has been described elsewhere, see [1]. The system is implemented on a standard Linux computer, which also provides the overall task specification for the mobile manipulation system, i.e. the specification of division of control between manipulation and mobility using redundancy. The result of this division is a task specification that provides input to the manipulation system.

The manipulation system is implemented on a regular PC. Due to the need for real-time response and a high sampling rate the software was implemented using the real-time operating system QNX [14].

The manipulation system implements the hybrid control system outlined above. Basic joint control is implemented by a set of gravity compensated PD-controllers. On top of the joint control a module carried out the inverse-kinematics transformation. For different tasks the PD parameters (K_p, K_d) can be changed. During normal operation the primary switching is carried out at the discrete level, and the system can here be considered a Discrete Event System [2]. The system is described by the 5-tuple: $A = (Q, E, \delta, T, \mu)$ Where

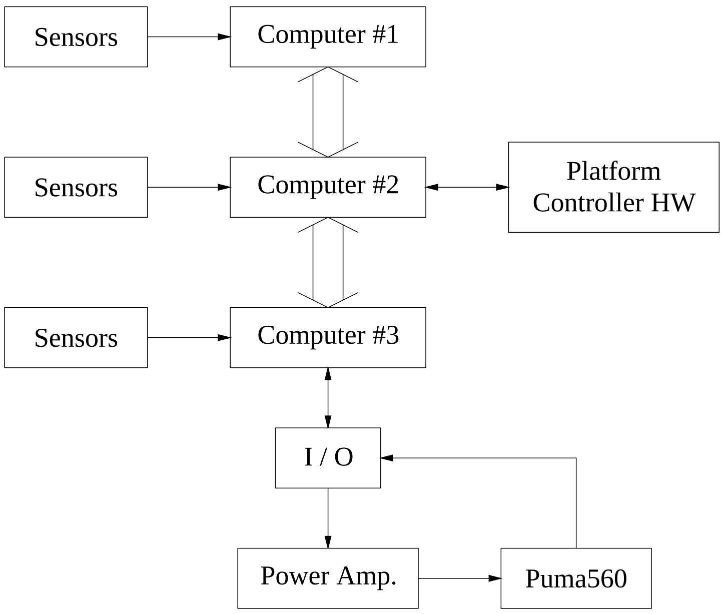

Fig. 4.: Hardware setup on the XR4000

- Q is the set of states, i.e. different actions
- E is the set of potential events
- δ is the initial state(s), $\delta \subset Q$
- T is the transition matrix $(Q \times E \rightarrow Q)$
- μ is a set of terminal states, $\mu \subset Q$

Each state Q has an associated set of control parameters for the PD controllers.

The framework enables easy specification of Hybrid Dynamic Automata or Discrete Event Systems. A dedicated language has been developed for the specification of such systems. Each state corresponds here to a particular action, like pre-grasping, manipulation, pick-up, etc.

The system architecture developed is shown in Figure 5.

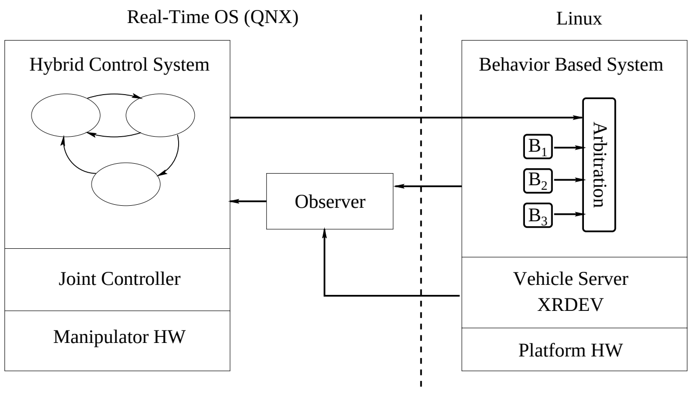

Fig. 5.: System architecture. Notice how the HDS comes in as just another behavior in the BBS, and how the BBS is treated as a “disturbance” that has to be observed and rejected. The observer also uses information from other sensors such as a gyroscope and accelerometers, but this is not shown in the figure. The operating systems used, QNX and Linux, are motivated in the text.

3.1 Specification of the HDS

An automaton defined in this language begins to specify which interface to use, i.e. which hardware server, and also the initial node. It may be of interest sometimes to exchange the hardware server to a simulated environment to perform off-line debugging. The initial node will contain the controller that will be the first one to execute. Thereafter, all the nodes in the automaton are listed. A node here corresponds to a typical action of a manipulation session that can be described with a continuous controller.

Program structure	Node structure
interface	*name*
initial node	*type*
nodes	*parameters*
	transitions

Each node is specified by name, type, parameters and transitions. The transitions refer to the other nodes, or itself, by using the specified name. The type of a node determines the functionality, such as the controller, and also defines which parameters or initial values that can be passed to the node. The designer of a node not only constructs the corresponding controller, but also defines the events that will cause a transition to occur.

During initialization, a lexical, syntactic and semantic check is performed, ensuring that only a well defined automaton can be executed. The language was implemented using the well known lexical and parsing utilities `lex` and `yacc` [6, 11], which makes it easy to extend the grammar if needed. More details of this language can also be found in [13].

3.2 Design of the Observer

We need to estimate, in real-time and at a high frequency, the pose of the platform to be able to take the motion of the platform into account when controlling the arm. The pose estimate is not for navigational purposes which means we only need good accuracy over short distances. However, it is very important that the estimate has a high resolution and is smooth, which is another characteristic difference from pose estimates for navigation. A non-smooth estimate of the platform pose will cause the arm controller to reject something that looks like a step-function resulting in a jerky motion, possibly damaging the arm or the object being manipulated.

Currently there are three sources of information from sensors available for the observer

- The platform odometry. Update frequency approximately 5-25 Hz and a delay of the order of 20ms. Angular resolution is 0.1 degree and positional resolution 1mm.
- A 3-axis optical gyro. Update frequency 100 Hz and a delay of about 10ms.
- A 3-axis accelerometer which is synchronized with the gyro.

Feed-forward information about the current platform motion command resulting from the arbitration/fusion mechanism of the behavior based system is also available.

The observer must consequently deal with data with different update rates, delays and noise levels. So far, the most promising algorithms seems to be a modified version of the extended Kalman filter. An overview of estimation techniques where different update rates and delays are discussed has been made by Larsen [10].

4 Experiments

To illustrate the utility of the framework outlined above a simple task is described in detail. The task chosen is drawing on a whiteboard. The task involved bringing the arm in contact with the whiteboard, drawing and finally retraction from the whiteboard. To make the task a little more interesting a circle is drawn. First the system draws a cross for the center (using two strokes) and then it attempts to draw a circle with the cross at the center. The task thus involves multiple state and interaction using feed-forward, force control for detection of contact, pressure control to ensure continued contact, and retraction of the arm. The task is specified in Cartesian coordinates to illustrate synchronized motion of all degrees of freedom. The task involved the following control modes

– Hold (keep the system stationary)
– Approach and detect contact
– Draw line having pressure control
– Draw circle having pressure control
– Goto pose (also used to retract)

The following events can be generated by the system

– Contact detected
– Whiteboard out of reach
– Line finished
– Circle finished
– Goto pose finished
– Timeout

Using these specifications the overall task definition is shown below. The nodes containing a "Hold" controller are used to let the system relax itself before continuing with the approaching motion. A small jerky motion would otherwise trigger the force threshold in the contact detection. Designing the trajectory generation for smooth transitions solves this problem (equal joint velocities on the borders).

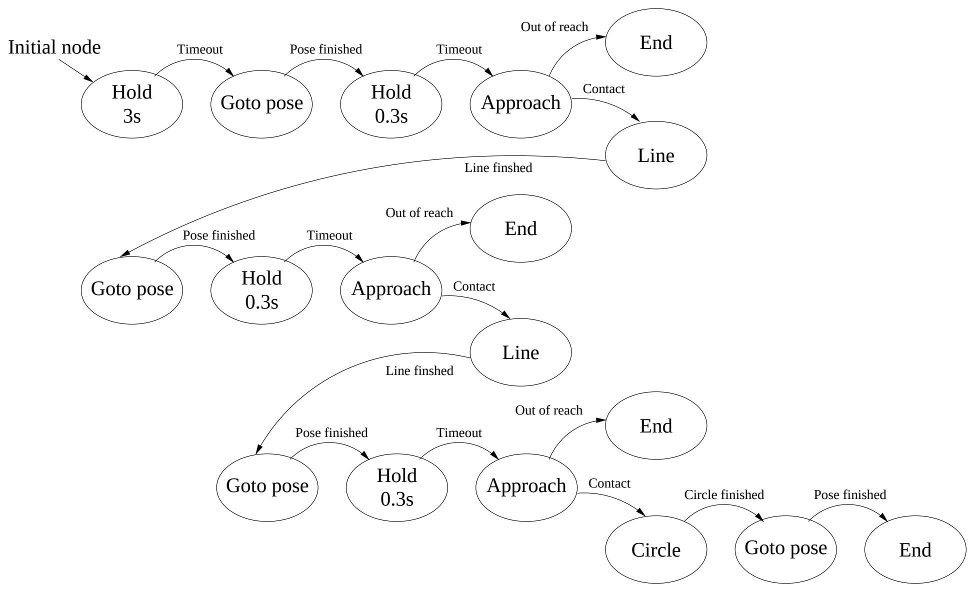

Fig. 6.: A graphic illustration of the hybrid automaton drawing a circle with a cross in the center.

A sample from the definition file that implements the above looks like

```
BEGIN
        NAME = Approach1;
        TYPE = Whiteboard_approach;

        Force = 200; % Threshold when contact
        Filter = 4;  % Select low-pass filter
        MaxX = 700;  % Define out of reach
        Speed = 20;  % Approach with 20mm/s

        TRANSITION[Contact] = Line1;
        TRANSITION[Out_of_Reach] = End;
END
```

```
BEGIN
        NAME = Line1;
        TYPE = Whiteboard_line;

        Length = 50;
        Direction = 225;
        Speed = 40;
        Force = 150;
        FGain = 0.00002;

        TRANSITION[Finished] = Retract1;
END
```

The result of the experiment is shown in Figure 7. Although the experiment did not explicitly use the possibility of simultaneously moving the platform (due to hardware problems), it illustrates how different control strategies are switched between. Measurements on the circle afterwards, showed that it actually was a

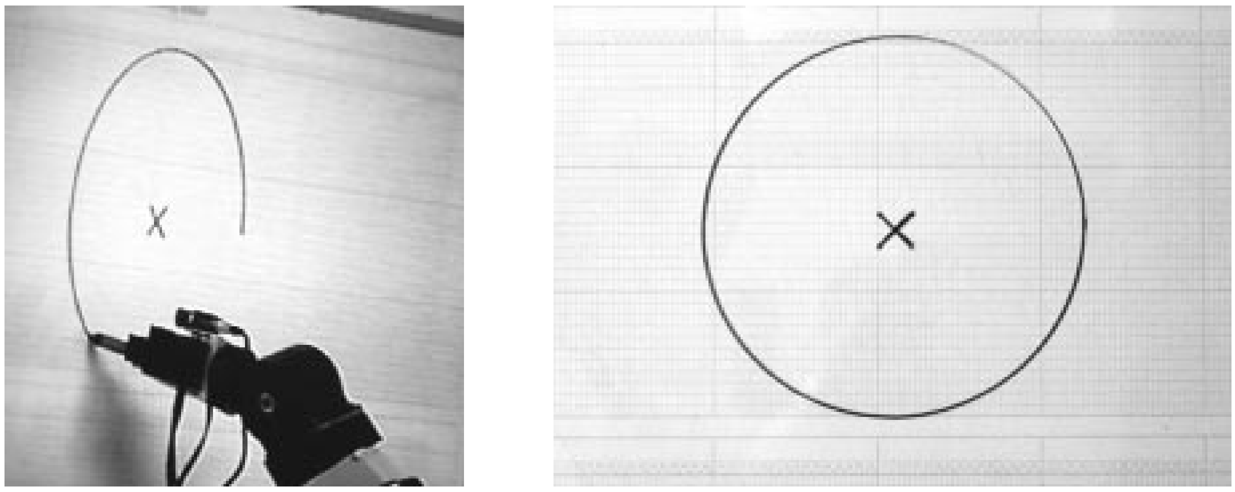

Fig. 7.: Left: Picture showing the robot drawing the circle with a cross in the center. Right: The final drawing.

circle and not an ellipse giving evidence that the control of the different joints where synchronized. Moreover, the specification language turned out to be a useful tool when adding new sub-tasks to the experiment, giving feed-back when something was not coherent in the automata.

5 Summary

The proposed architecture was motivated by a need to extend the skills of mobile platforms to also include advanced interaction with the environment, e.g. manipulation of objects, in an incremental way. Therefore, an approach to combine the existing techniques for navigation and mobility with a flexible control system for the manipulator arm was suggested. The objective was to find a system level design including mission planning, task analysis, path planning, control and integration. The focus of this paper was mainly control and integration whereas the other aspects are treated in earlier published research. An analysis was made of the different characteristics of the two cooperating systems, giving valuable insights when designing the architecture. Controlling the slow platform dynamics separately using loosely synchronized control through a behavior-based system was preferred, while the fast manipulator dynamics required synchronized control. A hybrid dynamic system was chosen to be the control paradigm of the arm due to the flexibility and the advantage of existing methods for analysis in terms of stability, controllability and observability.

The architecture has proved to be a successful way of structuring and designing a complex robotic system for mobile manipulation. The main contribution is the idea of combining two very different control paradigms, using the characteristics of each sub-system, in a structured and well founded way.

It became apparent that simultaneous motion of the platform and the manipulator requires timely information of the platform state, hence an observer is needed. The observer can provide estimates at a higher rate than sensor updates with the help of a model. The observer processes data from, in our case, the platform odometry, a 3-axis optical gyro and a 3-axis accelerometer.

Further research is needed on the design of the observer, taking into account that sensory information arrives at different frequencies with different delays. In addition, depending on the current task of the system, sensory data is being more or less corrupted with noise. The design of integrated control schemes is also a research issue, i.e. which set of behaviors and arm controller work well together.

Currently, the architecture is used to design controllers for visual servoing using an eye-in-hand camera, force/torque control etc. to perform tasks like opening doors, advanced manipulation of objects to fetch and carry and cooperation between robots. The multi-agent theme will also be investigated in more detail.

Acknowledgment

This work has been carried out within The Centre for Autonomous Systems (CAS) which is a research centre at the Royal Institute of Technology in Stockholm, sponsored by the Strategic Research Foundation.

References

1. M. Andersson and A. Oreback and M. Lindstrom and H.I. Christensen. : Intelligent Sensor Based Robot Systems *Lecture Notes in Artificial Intelligence.* Springer Verlag, Heidelberg, Germany, September 1999.
2. P. J. Ramadge and W. M. Wonham: The Control of Discrete Event Systems. *Proceedings of the IEEE*, vol. 77, no. 1, January 1989, pp. 81–97.
3. R.C. Arkin: *Behavior-Based Robotics.* The MIT Press, Cambridge, Massachusetts, 1998.
4. B.E. Bishop and M.W. Spong: Control of Redundant Manipulators Using Logic-Based Switching. Proceedings of the *37th IEEE Conference on Decision and Control*, Tampa, Florida, USA, December 1998.
5. U. D. Hanebeck, C. Fisher, G. Schmidt: ROMAN: A Mobile Robotic Assistant for Indoor Service Applications. *International Conference on Intelligent Robots and Systems.* Grenoble, France, 1997.
6. S. C. Johnson: *Yacc: Yet Another Compiler Compiler.* Computing Science Technical Report No. 32, Bell Laboratories, Murray Hill, New Jersey, 1975.
7. O. Khatib: Inertial Properties in Robotics Manipulation: An Object-Level Framework. *International Journal of Robotics Research*, vol. 14, no. 1, February 1995, pp. 19–36.
8. O. Khatib, K.Yokoi, K.Chang, D.Ruspini, R.Holmberg, A.Casal and A.Baader: Force Strategies for Cooperative Tasks in Multiple Mobile Manipulation Systems. *International Symposium of Robotics Research*, Munich, October 1995.
9. O. Khatib, O. Brock: Elastic strips: Real-Time Path Modification for Mobile Manipulation. *The Eighth International Symposium of Robotics Research*, Shonan, Japan, October 1997.
10. T. D. Larsen: *Optimal Fusion of Sensors.* Ph.D. Dissertation, Department of Automation, Technical University of Denmark, September, 1998.
11. M. E. Lesk and E. Schmidt: *Lex - A Lexical Analyzer Generator.* Computing Science Technical Report No. 39, Bell Laboratories, Murray Hill, New Jersey, 1975.
12. J. Lygeros, C. Tomlin, and S. Sastry: Controllers for Reachability Specifications for Hybrid Systems. *Automatica*, 1999. Accepted for publication, to appear in March 1999.
13. L. Petersson, M. Egerstedt, and H.I. Christensen: A Hybrid Control Architecture for Mobile Manipulation. Accepted for publication at the *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Korea.
14. *QNX Operating System: System Architecture.* Users Guide, QNX Software Systems Ltd. 1997.

This article was processed using the TEX macro package with SIRS99 style