1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLBLE BLUE TESTING SINCE 1964

METRO

1080 COUNTY ROAD 6, BARNESVILLE, MD 21015, USA
TEL. 410 485 7857 FAX 410 485 7857 TLE 0485857857

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**A Blackboard, Activity-Based Control Architecture
for the
Navigation of Autonomous Vehicles**

by

Reda Ezzat FAYEK, B.Eng.

**A thesis submitted to the
Faculty of Graduate Studies and Research
in Partial fulfilment of the requirements
for the degree of**

Master of Engineering

**Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering,
Carleton University,
Ottawa, Ontario, Canada K1S 5B6**

August 27, 1992

© Reda Ezzat FAYEK 1992

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file - Votre référence

Our file - Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-79786-X

Canada

The undersigned hereby recommend to the Faculty of Graduate Studies and Research acceptance of the thesis

*A Blackboard, Activity-Based Control Architecture
for the
Navigation of Autonomous Vehicles*

submitted by Reda Ezzat FAYEK, B.Eng., in partial fulfilment of the requirements for the degree of Master of Engineering.

Chair,
Department of Systems and
Computer Engineering

Thesis Supervisor

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
August 27, 1992

Abstract

This thesis presents a system architecture for autonomous navigation for a mobile platform designed on the view of activities. It uses a blackboard system to guide the different components of the architecture to perform the desired task. A control unit that acts upon information posted on the blackboard supervises the activities. Each activity is composed of several sensing modules connected to one or more perception modules, which are, in turn, connected to an appropriate action module so that the platform can exhibit a specified skill. The control unit interacting with the blackboard decides what activity should be active at any given time, based on information provided by the perception modules.

The approach followed in this architecture is based on the decoupling of two types of reasoning: reactive tactical reasoning to deal with unexpected events in a real-time fashion, and strategic reasoning based on common-sense. This leads to an activity-based system structure which is highly flexible and adaptive. Two such activities were implemented and used to evaluate the merits of this control architecture. Experimental results showed that this approach combines most of the merits of traditional control frameworks while avoiding their weaknesses.

Acknowledgements

I would first like to thank my two supervisors Gerald Karam and Ramiro Liscano for their invaluable guidance, both academic and practical. Without their cooperation, the research for my thesis would not have taken place at the National Research Council on the Mobile Robot Project. Their confidence in my work was a continuous encouragement throughout the last two years.

My special thanks go to Shadia Elgazzar and all the people of the Mobile Robot Project: Ramiro Liscano, Dave Green, Larry Korba, Sherman Lang, Allan Manz, and Doug Taylor. Without their continuous help, programming in Harmony, fixing hardware problems, and operating the vehicle would have been an insurmountable task. My thanks also extend to Bob Orchard and Aurora Diaz for the time they spent helping me. I would also like to thank Emil Petriu and Jaques Domey. I also thank my friend Sherif for his encouragement and support at the final stages of this thesis.

Finally, I would like to thank NSERC, the Department of Systems and Computer Engineering, and the National Research Council for the funding they provided.

CONTENTS

1. INTRODUCTION	1
1.1. Evolution of the Term Robot	1
1.2. Autonomous Mobile Vehicles	2
1.3. Motivation	3
1.4. Essence of the Solution	5
1.5. Essence of the Demonstration	7
1.6. Thesis Overview	8
2. BACKGROUND	10
2.1. Control of Autonomous Mobile Vehicles	10
2.2. Hierarchical Control Architectures	13
2.2.1. Hierarchical System Decomposition	13
2.2.2. Merits of Hierarchical Decomposition	15
2.2.3. Shortcomings of Hierarchical Controllers	16
2.2.4. Deviations from Hierarchical Decomposition	18
2.3. Behavioral Control Architectures	21
2.3.1. Behavioral Decomposition	21
2.3.2. Breakthroughs of Behavioral Systems	23
2.3.3. Weaknesses of Behavioral Architectures	25
2.3.4. Variations of Behavioral Architectures	26
2.4. Blackboard-Based Systems	30
2.4.1. Blackboard Control Architectures	31
2.4.2. Advantages of Blackboard Architectures	32
2.4.3. Limitations on Blackboard Control Systems	34
2.5. Summary	35
3. APPROACH	37
3.1. Current Impediments to Effective Design	37
3.2. Requirements of the Control Architecture	39
3.2.1. Behavior Requirements	39
3.2.2. Design Requirements	42
3.3. Control Architecture Decomposition	44
3.4. Analysis of the Design Approach	49
3.5. Summary	53
4. SOLUTION	56
4.1. Experimental System	56
4.2. Description of the Actual Control Structure	58
4.3. The Activities	61
4.3.1. Feature Extractors	62

4.3.2. Action Modules	64
4.4. The High-Level Decision-Making	67
4.4.1. The Knowledge Sources	69
4.4.2. The Production System	70
4.4.3. The Actions	71
4.5. Symbolic/Numeric Control Interface	72
4.6. Monitoring and Debugging	74
4.7. Physically Implemented System	76
4.7.1. The Passageway Activity	77
4.7.2. The Channel Activity	78
4.7.3. An Example of System Behavior	81
5. VALIDATION of the DESIGN	84
5.1. Overview	84
5.2. Experimental Setup	85
5.2.1. Support Tools	85
5.2.2. The Activities	86
5.2.3. The Blackboard	88
5.2.4. Observations	90
5.3. Meeting the Original Requirements	92
5.3.1. Reactivity	93
5.3.2. Intelligence	93
5.3.3. Multiple Sensors	95
5.3.4. Multiple Goals	95
5.3.5. Modularity	96
5.3.6. Flexibility	97
5.3.7. Adaptability	98
5.3.8. Expandability	98
5.4. Discussion	99
5.4.1. Granularity Level	100
5.4.2. Truth Maintenance	101
5.4.3. Failing Modes	102
6. CONCLUSIONS	104
6.1. Research Summary	104
6.2. Implementation and Experiments	108
6.3. Contributions	110
6.4. Future Work	111
References	114

LIST OF FIGURES

Figure 2.1 Classical hierarchical control structure	14
Figure 2.2 Hierarchical decomposition	16
Figure 2.3 Modified hierarchical control architecture	20
Figure 2.4 Twin parallel perception/control hierarchies	20
Figure 2.5 Behavioral decomposition of a mobile robot control system	22
Figure 2.6 The Subsumption architecture	24
Figure 2.7 The Independent Agents Architecture	27
Figure 2.8 Recursive Nested Behavior Control	28
Figure 2.9 Behavioral decomposition of the grasping task	29
Figure 2.10 Motor Schema-based control	30
Figure 2.11 Blackboard Control Architecture	31
Figure 2.12 Activation set of virtual sensors and reflexive behaviors	33
Figure 3.1 Passageway activity	46
Figure 3.2 Outline of the activity-based intelligent control architecture	48
Figure 4.1 Photo and block diagram of the experimental vehicle	57
Figure 4.2 Block diagram of the control framework	58
Figure 4.3 Package structure	61
Figure 4.4 Detailed activity decomposition	62
Figure 4.5 Feature extractor	63

Figure 4.6 Action module	64
Figure 4.7 A low-level control loop	66
Figure 4.8 The blackboard system	68
Figure 4.9 Basic structure of a high-level control loop	70
Figure 4.10 Symbolic/numeric control interface	74
Figure 4.11 The monitoring/debugging package	75
Figure 4.12 The passageway activity	78
Figure 4.13 Irregularities of a typical workshop floor	79
Figure 4.14 The channel activity	80
Figure 4.15 Possible scenario in a workshop	81

1. INTRODUCTION

1.1. Evolution of the Term Robot

Twenty years before Isaac Asimov and John Campbell coined the phrase *Robotics* to denote the study of robots, Karel Capek (1921) derived the term *Robot* from the Czech word *robata* which means "forced labor." Such imaginary robots played many important and central roles in science fiction stories and triggered a large number of technological advances. The concept of "capable devices" is, however, much older than the terminology currently used to describe it. Around 2000 B.C. an Egyptian toy dog with a lever-controlled jaw was fabricated, in the 1400s the first *android* clocks were developed in Germany and Switzerland, and in 1770 Pierre and Henri Jacquet-Droz constructed mechanical *automata* that could write, draw, and play musical instruments.

Nitzan [33] defines a robot as being "a general purpose machine system that may include effectors, sensors, computers, and auxiliary equipment and, like a human being, can perform a variety of tasks under unpredictable conditions." The Japanese classification of industrial robots has five categories, corresponding to increasing level of intelligence and autonomy, ranging from teleoperated manipulators to intelligent robots [39].

During the last decade, the number of researchers in the field of robotics increased tremendously, as can be observed from the growing literature. During the same period, a large number of robotic projects emerged, both in academic and industrial environments. The first attempts were mostly concerned with manipulators and their mechanical and spatial control problems. Autonomous mobile vehicles, which require higher levels of autonomy and reasoning, subsequently attracted a significant number of researchers who initially worked on problems like trajectory planning and collision avoidance.

1.2. Autonomous Mobile Vehicles

Numerous autonomous mobile platforms have been built throughout the world with different levels of success. Such intelligent vehicles proved to be of a great value for a variety of applications both in indoor and outdoor environments. Indoor applications include, but are not limited to, building security, automated delivery, factory/workshop automation, intelligent wheelchairs, routine and emergency maintenance in hazardous sites. On the other hand, outdoor autonomous vehicles can be used in terrain exploration, space and underwater navigation, mining, military unmanned armored vehicles, driving assistance, and on site inspection of natural catastrophes or accidents.

The research in this area originally focused on the solution of navigational problems. Most of the effort was directed towards the path planning and the collision avoidance algorithms and their computational issues. Initially, little work has been done on the selection of the system structure required to control the interaction of the different components. Traditional system configurations, relying on basic engineering techniques,

were typically used for this purpose. Recently, more attention has been drawn to the issues related to the system configuration, and more elaborate architectures were investigated. With the increased number of experimental implementations, it was observed that one major factor that seriously affects the practicality and usefulness of such systems seems to be the inherent properties of the control architecture. The use of new technologies, or more efficient implementations, help in improving the performance of these systems, but only at the level of individual components. These improvements, however, cannot overcome the upper limits imposed by the overall structure of the control architecture. Such architecture-dependent limitations generally influence the applicability, the adaptability, and the response time of the autonomous vehicles.

1.3. Motivation

A careful literature survey shows that the majority of robotic control systems can be categorized into three general types of design [22]: hierarchical controllers, behavioral systems, and blackboard systems. Traditionally, hierarchical control systems were used primarily because of their well established theoretical base and easy implementation. In such architectures, the control problem is divided along functional lines into progressive levels of abstraction of data. The responsibility of the lowest level is to interact with the environment, via sensing and action; the highest level's responsibilities are reasoning, planning, and decision-making.

The hierarchical decomposition of the control problem, while being a theoretically sound engineering approach, leads to unacceptable performance when applied to real-time

systems, in general, and to mobile vehicles, in particular. Their purely sequential nature results in long response times and poor reactivity in real-time unpredictable environments. They are also prone to other unavoidable implementation complications which reduce the possibility of adopting a strictly hierarchical design.

On the other extreme, fully parallel behavioral decomposition of the problem was investigated to avoid the basic shortcomings of hierarchical systems. In such architectures, the connection between sensing and action is assured by a set of capable agents running concurrently. Each of these agents is performing some simple and fundamental behavior. The distinctive feature of behavioral systems is the absence of any central reasoning, or coordinating, agent. The overall behavior exhibited by the system is the result of the interaction between the competing capable agents.

Behavioral architectures brought many breakthroughs in improving the expandability and response time of the control systems. On the other hand, they introduced new design and control problems, particularly in the arbitration of conflicting commands and the inflexibility of the design.

Some researchers realized the duality of this control problem and the need to combine some essential features of both hierarchical and behavioral systems. The decision-making subsystem is merely a sequential device, but the real-time interaction mechanisms are mostly parallel in nature. Therefore, the concept of using a common blackboard to coordinate the interaction of several independent components was borrowed from the field of Artificial Intelligence, and was applied to this control problem. In a similar system, the blackboard provides a means for a centralized decision-making module to guide a group of independent agents to the solution. However, a standard blackboard control system suffers

from serious implementation problems involving the required storage capacity and the processing time.

1.4. Essence of the Solution

Most of the problems associated with each of the previously mentioned control methods are inherent in the implementation. That is, they are the result of the selected decomposition of the control problem; they put an artificial and undesirable upper limit on the system's capabilities. Many of these weaknesses would not arise if the control architecture was defined and built after a careful consideration of the system's requirements, instead of modifying a strict control paradigm (such as pure hierarchical or behavioral systems) to suit the problem in hand. This is the objective of this thesis.

The purpose of this research is to (1) build a conceptual model for a control architecture for the navigation of an autonomous mobile platform that can exhibit reactive behavior to deal with unexpected events, while at the same time being able to make task-level planning decisions, and (2) physically implement a prototype of this control framework to validate the design. Contrary to traditional approaches for solving this control problem for a real-time environment, the proposed control architecture was not defined until after the identification of its required specifications and characteristics. The first step of the design process was to enumerate a list of important performance measures that need to be respected to ensure an acceptable operation of the mobile vehicle in the target environment.

This set of required general behavior specifications includes (1) reactivity to dynamic environments, (2) intelligent reasoning, (3) support for multiple sensors, (4) reconciling multiple goals, (5) robustness against imperfections and faulty situations, and (6) overall reliability. These fundamental requirements in the behavior of the sensor-based intelligent control have direct implications on the control framework and the system design. Design specifications have been identified and observed when choosing the different system components: (1) modularity, (2) flexibility, (3) expandability, (4) adaptability, and (5) centralized high-level decision-making capability.

These design requirements dictate the essence of the solution and give the general framework of the control architecture. The control problem has been divided into the different activities that the system is to perform in order to operate successfully in real-world environments. These activities achieve the basic skills which are required to be exhibited by the autonomous mobile vehicle. Examples of these skills are: passageway traversal, doorway traversal, following a person, and channel crossing. Each such activity is capable of successfully performing its task, and is implemented as an organizational unit comprising sensing, perception, and control modules. These building modules are designed such as to reduce their interdependence. Therefore, their interconnections can be easily changed with little modification which ensures flexibility and adaptability. The high-level decision-making capability requirement is satisfied by incorporating a rule-based symbolic reasoning element. The main function of the reasoning agent is to guide the capable activities towards the solution. This relieves the central agent from the long processing delays and the heavy computational requirements typically found in systems with centralized intelligence. This design approach leads to a system architecture which satisfies the previously stated design criteria and the behavior specifications. This control

framework also provides a testbed for different control strategies. It facilitates the experimentation with different interconnection patterns between the modules, without any necessary changes to the system structure, in order to investigate their relative performance.

1.5. Essence of the Demonstration

The results of this research were evaluated by applying them to develop a framework for the control of the experimental mobile vehicle, *eAVE*¹, and using it to ensure intelligent navigation in a partially known indoor environment. Since the time frame of this research was limited, and due to the large number of possible skills to implement, only two activities were considered. These are (1) the passageway activity responsible for collision free navigation in open-space, and (2) the channel crossing activity responsible for floor sensing and manoeuvring around level irregularities; e.g., grooves, trenches, and/or pipes. These two activities are particularly relevant for the autonomous navigation in an industrial workshop environment.

These activities were implemented and tested separately before they were integrated into the common control framework. The layout of the laboratory space was continuously modified to create different situations. The configuration of the static objects was expanded by the introduction of dynamic objects; these were mostly the people working in the laboratory. Due to the lack of an available floor sensor, a set of simulated channels in the floor was used to test the interaction of the control system with the dynamic situations that

¹ This is the "experimental Autonomous Vehicle" used as a testbed in the Autonomous Systems Laboratory (ASL) of the Institute for Information Technology (IIT) at the National Research Council (NRC).

arise from random navigation in this environment. The focus of attention of the navigation system was continuously attracted to the imminent threats, whether these were obstacles or floor relief. These were handled appropriately by the relevant activity invoked by the intelligent agent.

The proposed control framework is not restricted to this application. One of its significant advantages is its applicability to the control of a wide variety of complex real-time systems involving distributed sensing, decision-making, and action. Therefore, the conclusions of this research should be relevant for the intelligent control of real-time systems in general.

1.6. Thesis Overview

Chapter 2 gives a detailed analysis of the previous attempts to solve the control problem of autonomous mobile vehicles. It covers the traditional hierarchical systems and their recent modifications, as well as the pure behavioral architectures and their less rigid variations. The use of the blackboard-based control architectures is also analyzed.

The shortcomings of these strict control paradigms are discussed in chapter 3. The study of the different impediments for effective intelligent control are depicted. Consequently, a set of behavior specifications is formulated; these requirements are necessary in order for the sensor-based real-time control architecture to exhibit any intelligent behavior. The implications of these behavior specifications on the design approach are also enumerated. The design approach is then developed in order to satisfy the

stipulated characteristics. A discussion of the adopted control architecture, at the end of chapter 3, demonstrates the merits of this design approach.

Chapter 4, on the other hand, describes the details of the particular implementation used to demonstrate the validity and feasibility of this control framework. The hardware and software used for this implementation are both discussed. This chapter is organized according to the conceptual building blocks of this architecture, namely the real-time packages. Their different components and their interactions are also detailed. Chapter 4 concludes with a discussion of the original set of requirements against the proposed solution.

The experiments carried out on the implemented system are presented in chapter 5, together with the challenges encountered and the design decisions and tradeoffs. Finally, chapter 6 presents the general conclusions, a summary of the contributions, and the suggestions for future work on this problem.

2. BACKGROUND

2.1. Control of Autonomous Mobile Vehicles

Building intelligent robotic systems that reason while functioning in unstructured environments is a challenging task. Intelligent, self-sufficient inference systems are essential if robots are to operate continuously in unpredictable environments. In addition to an on-board knowledge to handle various operating environments, these systems require the ability to monitor and react to dynamic, unexpected events in real time to guarantee intelligent responses.

To achieve a reasonable operation of an autonomous mobile vehicle, it is necessary to combine the information acquired from multiple sensors to make decisions, while trying to satisfy simultaneous, and possibly conflicting, goals. The use of multiple sensors improves the reliability and the robustness of the system. This is because the failure of a particular sensor would not cause the total failure of the perception subsystem, and also because the malfunction of any particular sensor can be accounted for partially from the

sensory information provided by the remaining sensors. Additionally, the ability to satisfy several contradictory goals is necessary to make the appropriate compromises when such situations arise.

These two requirements, multiple sensors and multiple goals, have been identified in all mobile vehicle projects. However, the approach used to fulfil them has depended on the way the problem was decomposed and handled. The research in this field has gone through several phases of evolution. Most of the innovations have followed corresponding advances in the software techniques or the hardware implementations used to cope with the increasing computational demands of these control systems.

With the advances in parallel processing systems, many implementations have used distributed architectures to alleviate the processing bottleneck of raw sensory information. Similarly, the introduction of new software concepts, such as blackboard systems and object-oriented languages, had direct impact on some mobile robot research projects. These techniques were introduced to ease the integration of large amounts of sensory information into a central representation and to structure this knowledge to make it easily accessible by the different reasoning modules.

Recently, more attention has been directed towards the system organization used to control intelligent autonomous robots. There are several computational functions that need to be performed by any robot control architecture. However, the type of processes used to accomplish these functions and the organization of the data flow between these processes determine the characteristics of the control architecture. These computational functions can be summarized as: *perception, decision-making, and action*. These are described briefly.

The control architecture should include perception processes to transform the raw sensory data, coming from the physical sensors, into a more compact representation to be used by other processes. The perception processes may feed their output to other perception processes in order to increase the information content of the manipulated data; i.e., to increase its level of abstraction².

Similarly, the control architecture must include action processes to transform the output from the decision-making processes (to be described later) into continuous control signals for the actuators. If the output of the decision-making processes is high-content commands, more action processes may be required to decompose it to more detailed sequences of action.

Finally, each decision-making processes transforms the processed sensory information into desired commands according to a specific rule or control law. The characteristic function of the decision-making process is defined at the time of design and depends on the processes connected to its input and output.

The majority of the architectures found in the literature can be categorized into three general types of design [22]: hierarchical controllers, behavioral systems and blackboard systems. The main differences between these architectures arise from the type of information used for the computational processes and the interconnections between them. That is, the information (or level of data abstraction) used by each process, and the data paths between the different processes.

²Not to be confused with the Software Engineering notions of abstraction.

2.2. Hierarchical Control Architectures

Conventional architectures identified and related three major subsystems (sensing, planning, and control), in a "sense-think-act" cycle. Traditionally, hierarchical system decompositions were used primarily because of their well established theoretical base and relatively easy implementation. In such architectures, the control problem is divided along functional lines into progressive "levels of abstraction" of data [1, 2, 34, 40]. That is, the computational functions that need to be performed are used as the basis for system decomposition. Since the processes corresponding to these computational functions handle data with different information content, hierarchical systems may be viewed as sequences of data transformations. Data flows from the sensors to the series of perception processes, to the decision-making process, and from there through the series of action processes to the actuators.

2.2.1. Hierarchical System Decomposition

The common distinctive feature of hierarchical controllers is that they have been designed with two or more layers of abstraction in mind. Such control systems consist of a central "intelligent" subsystem, with perceptual modules as inputs and action modules as outputs. Information flows through a hierarchical series of assimilation stages, then through a series of planning stages, until finally it results in the issuance of the control signals [37]. At any particular layer in the hierarchy, the information processed has a certain level of abstraction which is lower than that of the next higher layer. Furthermore, the information available to, and used by, a given layer is unique to the function performed at that level [40]. Consequently, each layer has to communicate with lower and higher

layers via well defined interfaces. Figure 2.1 gives a typical example of such a decomposition of the control problem.

The responsibility of the lowest layer is to interact with the environment via sensing and action; the highest layer's responsibilities are reasoning, planning, and decision-making. Therefore, raw sensory data flows upwards from the sensors at the bottom of the hierarchy and is processed and abstracted into more symbolic information until it reaches the top layer. The reasoning/planning module then uses the appropriate inference mechanisms to guide the operation of the whole system. It acts as a symbolic information processor that tries to satisfy the predefined goals while taking into account the perceived state of the world. The commands are issued from this highest level module and decomposed into more specific and detailed sequences of actions. At the lowest layer, these commands result in the issuance of direct actuator control signals.

Figure 2.1 Classical hierarchical control structure
Adapted from [31], fig. 8.

The system performs as a classical closed-loop feedback controller. The layers form a chain through which information flows from the environment, via sensing, through the hierarchy and back to the environment, via action, closing the feedback control loop [11]. The complexity of the behavior generated by the control hierarchy depends on four factors: the number of layers, the number of feedback variables which enter each layer, the sophistication of each layer function, and the sophistication of the sensory processing modules which extract feedback variables [1].

2.2.2. Merits of Hierarchical Decomposition

Hierarchical systems are based on the intuitive decomposition of complex systems into smaller subsystems which are easier to design. They allow the designer to decompose the control problem into precise levels of control and sensory abstraction. This well structured functional decomposition is important during the realization phase, and, therefore, a hierarchical organization is needed in defining the logical structure of the system [40]. Figure 2.2 gives a common functional decomposition which can be applied, in theory, to any control problem.

The modular structure of hierarchical control systems makes them extendable. Changes during the implementation phase are mostly additions or deletions of entire modules within a particular level. Additionally, sufficient modularity allows the complexity of any level to be kept within tolerable limits regardless of the complexity of the overall system [1].

In a hierarchical system, the response time is relatively short at the lower levels and the complexity of the control computations is low. The response time is longer at the higher levels and the complexity of the computations is large. Therefore, the total computational power required at each level of the hierarchy is more or less constant [3]. This helps in balancing the computational load throughout the system.

Figure 2.2 Hierarchical decomposition

The highest level in the hierarchy is always in control. This guards effectively against control being lost in a low level routine which might be performing previously commanded operation that needs to be interrupted [8].

2.2.3. Shortcomings of Hierarchical Controllers

The hierarchical decomposition based on progressive levels of data abstractions, while being a theoretically sound engineering approach, leads to unacceptable performance when applied to real-time control systems, in general, and to autonomous mobile vehicles, in particular. Furthermore, hierarchical controllers do not provide the essential adaptability necessary for coping with unexpected real-time events [5, 6].

The series of stages through which sensor data flows place unavoidable delays in the loop between sensing and action [37]. Furthermore, the sequential structuring of the system can cause any particular error in a single module to totally disorient the robot [36].

Moreover, although planning systems relying on progressive levels of abstraction provide analytic and provable results, they are highly deterministic structures whose only way to deal with ambiguity and/or uncertainty is to increase the precision of their knowledge bases. This results in a strict structured problem decomposition, poor use of sensory information, and a lack of flexibility.

They are also prone to other fundamental implementation complications which reduce the possibility of adopting a strictly hierarchical design. In such systems, a long chain of modules is needed to connect perception to action, and an instance of each module must be complete and working before the system can be operational at all. That is, a full system must be built before any of these modules can be tested. But, since the optimal subsystem partitioning can not be known at the time of the initial design, advance identification of the best interaction paths between these subsystems is not possible. Additionally, changes to a particular piece must either be done in such a way that the interfaces do not change, or the effects of the change must be propagated to neighboring modules, thus, changing their functionality [11].

The strict functional decomposition approach creates an intricate design pitfall, namely, the need for early interface specification. This imposes serious limitations on the system development. For the modules to be adequately designed, the interfaces governing the information pathways between them should be identified and established first. The specification of these interfaces can only be done after the identification of the inputs and

outputs of the modules, i.e., after these modules are settled. The easy way out of this problem is the early specification of the interfaces. That, however, places serious constraints on the modules before their potential capabilities are completely understood. Development of each module is thus restricted by interface limitations that may discourage some desirable enhancements if they cannot conform to these specifications [37].

2.2.4. Deviations from Hierarchical Decomposition

Several other system organizations were developed trying to avoid the shortcomings of strict hierarchical systems. Some of these were augmentations to the basic hierarchies by adding more connections between the modules, while some others used different configurations and grouping of the functional units. These were attempts to develop systems that exploit the merits of hierarchical systems while trying to alleviate their shortcomings.

For instance, the *activity structure* [14] is a dynamic tree with a parent-child relationship. The activity hierarchy is not a predefined one; at any particular time, its configuration only depends on the current task and the functions required to perform it. This notion of activity facilitates the management of multiple actions and avoids the fixed number of layers imposed by rigid hierarchical systems. Moreover, it treats all activities in the same way regardless of their level. This allows the control system to respond to asynchronous events at any moment, since all it needs to do is to initiate the corresponding chain of activities. In this configuration, however, there is no guarantee that the lower level activities will return control to the higher-level ones in cases of failure. This is due to the lack of a centralized reasoning and decision-making capability.

In other implementations, such as the one shown in figure 2.3, the organizational hierarchy is not a strict tree structure. It is split into processing and control hierarchies, and it supports horizontal information flow between computational modules at the same level in the two hierarchies. Nevertheless, the primary flow of data and control is vertical as in [1].

At the lowest level of the processing hierarchy, the feedback consists of raw sensory data. This level is fast acting and has very short loop delays. Each ascending level processes this input further, relaying the processed data to the corresponding level in the control hierarchy. Together with horizontal flow of information, this causes the feedback to close a real-time control loop at each level in the hierarchy. Similarly, the sensory processing hierarchy can receive input at various levels from the control hierarchy to ensure a two-way exchange between the two hierarchies at all levels. Together with the nested control loops, this feature adds more flexibility and reactivity to the originally rigid hierarchical structure. Furthermore, the decisions each level has to make at any time are relatively simple, because only the correct action for the next instant needs to be planned [8]. This *cross-coupled processing-generating hierarchy* may be a fundamentally correct approach to make the robot responsive to real-time sensory input [1]. However, it does not solve the problem of dependence of high-level layers on the low-level ones.

Twin parallel hierarchies, illustrated in figure 2.4, [17] for navigation and for perception can also be used. They are organized according to the information abstraction level and the required response time. The top level is a form of symbolic servo loop operating at the action level; it monitors the execution of each task and dynamically generates the actions required to accomplish its goals. In the same time, some "reflexes" (which are unplanned commands based on raw sensory data) are implemented to improve the system reactivity to unexpected events.

Figure 2.3 Modified hierarchical control architecture

Figure 2.4 Twin parallel perception/control hierarchies

2.3. Behavioral Control Architectures

Behavioral systems were introduced [5, 7, 10, 11, 24, 37] to cope with the main shortcomings of hierarchical systems, especially issues regarding reactivity, reliability, and robustness. Unlike hierarchical systems, the dominant idea in behavior-based control systems is to break up the control problem into the behaviors that should be exhibited instead of the stages of information flow within the controller. Their distinctive feature is the absence of any central intelligent reasoning agent. The emergent behavior demonstrated by the system is the result of the interaction between its competing constituents. The data flows from the sensors through parallel paths to the actuators. Each path is a compound module performing the main control computational functions: perception, decision-making, and action. Since different actions arise from the different paths, any behavioral system requires an action arbitration mechanism. This is used to resolve the possible conflicts between the different actions and to provide a coherent set of physical commands to the actuators.

2.3.1. Behavioral Decomposition

The control problem is decomposed into "levels of competence" or "task-achieving behaviors" [11] where lower levels perform simple and general tasks, while higher levels perform more complex and situation-specific tasks. The building block of behavioral architectures is a "behavior". A behavior is a module which accepts sensory information as input and generates actuator control signals as output. Figure 2.5 shows a tentative

behavioral decomposition of the control problem of a mobile robot. Although higher levels are expected to exert influence on the lower level ones, the lower layers can still perform reasonable control actions on the robot without waiting for the assimilation of sensory data to be completed by the upper levels. To ensure this independency, each behavior has access to the robot's sensors and can manipulate its actuators connecting sensing to action [7].

Furthermore, the part of the system doing perception is spread over many units which are not necessarily interconnected by data paths or related by function [13]. Sensory information coming from the multiple sensors is integrated only when required by a certain behavior [7].

Figure 2.5 Behavioral decomposition of a mobile robot control system

Since all the levels are continuously exposed to the sensory information, and since each level employs a representation derived from only a small portion of the available sensory data, distinct, and possibly conflicting, commands may be issued by different behaviors. This replaces the typical sensor fusion requirement with the need for command arbitration and/or command fusion. Consequently, the role of a central representation, and hence a central reasoning module, decreases since the conflict resolution happens at the action level rather than at the perception level [12].

In such systems, the complexity of the robot's behavior is not necessarily inherent in the complexity of the architecture, but can also be due to the complexity of the environment [13]. That is, a simple behavioral system with a relatively small number of behaviors can exhibit a relatively complex overall behavior as it reacts to the dynamic events in a complex environment.

2.3.2. Breakthroughs of Behavioral Systems

In the *subsumption architecture* [11], the system is divided on the basis of the desired external manifestation of the robot into levels of competence, called "behaviors." It uses a rigidly layered decomposition that emulates multiple behaviors [5]. Each layer consists of a fixed topology network of asynchronous modules communicating over low-bandwidth channels. The output of the arbitrator between any two successive behaviors is the output of the high priority level. The illustration in figure 2.6 shows that any particular level of competence includes as a subset each lower level of competence. This is guaranteed since higher layers "subsume" the effect of lower ones by "inhibiting" their inputs and/or "suppressing" their outputs (at the arbitrator). The failure of one or more levels results in a performance degradation instead of a complete fail-out. When higher levels can not produce results in a timely fashion, lower levels will still produce sensible behavior - albeit at a lower level of competence.

Generally, in behavior-based control systems, perception is action-oriented; that is, sensory data is only integrated when required by a certain level. This provides an easy and efficient mechanism for the integration of multiple sensors. The command fusion allows

pathways for primitive sensory data to be exploited at the lower levels for immediate responses [37]. This also allows many individual goals to be matched concurrently at no extra cost which eliminates the need for an early decision on which goal to pursue first. In this way, the computation necessary to support quite complex behavior is highly simplified.

Figure 2.6 The Subsumption architecture

Such architectures exploit the advantages of a wide variety of immediacy/assimilation tradeoffs without loss of flexibility. With the absence of any central representation, there is little chance that any given change in the properties of the world can cause total collapse of the system. Rather a given change would at most incapacitate some but not all the control levels. This, together with the distributed modelling of the environment over the different levels, adds to their robustness and reliability.

There is an incremental path from very simple behavioral systems to complex ones, as high level competence can be achieved by adding new layers "on top of" existing ones leaving them intact. This allows for early experimentation in the real world. The idea of using the world as its own model, instead of building simplified world models that might affect the solution, improves the system robustness and guards against typical pitfalls of earlier implementations [13].

Experiments with the behavioral control architectures [24] illustrated the advantages of having direct connection between sensing and action at a low level. Tying sensing to action modules improves their competence level and reactivity, while tying action to sensing reduces the computational demands on the perception subsystem.

2.3.3. Weaknesses of Behavioral Architectures

Pure behavioral systems, though easily extendable, do not provide enough design flexibility and modularity. Their fixed topologies and/or strict interfaces limit the number of possible interaction patterns between the different behaviors, which reduces the overall system flexibility. For instance, during the design of the lower levels of the subsumption architecture, the possible realizations of higher levels must be taken into account to prepare the proper interfaces. These difficulties are similar to the limitations imposed by the early interface specification in hierarchical architectures.

Furthermore, human level intelligence is too complex and little understood to be correctly decomposed into the right subpieces, and even if we knew the subpieces we still would not know the right interfaces between them [13]. Therefore, in most of the existing

behavioral implementations, the choice of the order in which layers are classified is completely artificial and questionable. There is no formal method to divide a given task into an optimum set of behaviors. Additionally, a more general and flexible relationship than strict hierarchy between behaviors is required.

The absence of any form of central representation in pure behavioral systems causes them to lack common-sense. Thus, the emergent behavior of the system is not the result of a global understanding of the situation, but it is more a reflexive reaction to the latest dynamic events. It is doubtful that highly complex behaviors can be achieved without the aid of any central representation [13]. This limitation eliminates the possibility of using a strictly behavioral decomposition in complex systems.

Among the new design and control problems introduced by these architectures is the arbitration of the conflicting commands generated by the competing behaviors. A more intelligent and situation-specific behavior cooperation or arbitration scheme is required instead of simple fixed-priority mechanisms since behavior priority lists may change with respect to specific situations and/or actuator [24].

2.3.4. Variations of Behavioral Architectures

The *Independent Agents Architecture* [42] is also a behavior-based architecture that decomposes the control system vertically into perception, behavior, and actuator control subsystems. The behavior subsystem is decomposed horizontally into independent *sensorimotor behaviors*, as shown in figure 2.7. Complex tasks are decomposed into subtasks that can be implemented by separate parallel sensorimotors [42].

Figure 2.7 The Independent Agents Architecture

In this architecture, however, a common perception subsystem provides the processed sensory information for the behavior agents which reduces redundancy in the data processing. This reduces the need for inter-behavior symbolic communications and restricts it to their interactions with the environment, which increases the overall system modularity [16].

The *Recursive Nested Behavior Control* [7], shown in figure 2.8, is a strictly recursive layered hierarchy of behaviors featuring feedback and feedforward. Low-level behaviors report to high-level ones through feedback connections. Unlike the subsumption architecture, the output of the higher-level behaviors is directed, as feedforward signal, to the lower-level behaviors to modify their output. Therefore, the feedforward mechanism provides the necessary arbitration; it avoids conflicting commands at the bottom level. The

actuators are connected to the output of the lowest-level behavior. The different behaviors use different representations according to their level of data abstraction. In this architecture, both hardware and software can be configured as "Functional Units" consisting each of several closely related functions which require intensive data exchange. The boundaries of these functional units do not need to match with the boundaries of the behavior levels.

Figure 2.8 Recursive Nested Behavior Control

Another system decomposition [10], similar to the subsumption architecture, was proposed. In this approach, any specific task is divided into a set of consecutive subtasks operating concurrently but independently - each monitors conditions for its execution and overrides the action of its predecessor. Instead of the competing agents of the subsumption architecture, the subtasks generate cooperative actions directed toward the completion and maintenance of the global task. This is achieved by the appropriate handling of the output

from different behaviors instead of the fixed priority mechanism of the subsumption architecture. In this case, a careful choice of the subtasks is required to ensure this cooperative behavior. For simple tasks, a time-sequence of the subtasks may prove to be an adequate partition. Figure 2.9 describes one such example which is the task of grasping an object by a manipulator arm/gripper setup.

Figure 2.9 Behavioral decomposition of the grasping task

The *Motor Schema-based* navigation [4] integrates both planning and sensor-driven dynamic replanning. The concept of using Motor Schemas is illustrated in figure 2.10. A perceptual module, called "perceptual schema", is associated with each action module, called "motor schema" [6], and a communication mechanism is provided to reduce conflicting actions. The output of several such combinations is fed to an arbitrator which generates the control signal to the actuators. Command arbitration is accounted for by a superposition mechanism (vector summation) to achieve an overall consistent behavior. Additionally, the distributed processing in such architecture allows for modular design.

Figure 2.10 Motor Schema-based control
Adapted from [6], fig. 3.

2.4. Blackboard-Based Systems

With the increased number of research projects in this field, some researchers realized the duality of the autonomous vehicle control problem and the need to combine some essential features of both hierarchical and behavioral architectures. While the decision-making subsystem is merely a sequential device, the real-time interaction mechanisms are mostly parallel in nature. Therefore, the concept of using a common blackboard to coordinate the interaction of several independent components was borrowed from the field of Artificial Intelligence, and was applied to this control problem. It provides a means for parallel execution by the concurrent independent components, while allowing sequential reasoning and decision-making. In these architectures, each module contains a perception process, an action process, and a partial decision-making process. A centralized decision-making process interacts with all the modules in the architecture.

2.4.1. Blackboard Control Architectures

Blackboard control systems [21, 23, 25, 35] are based on the idea of using a number of distributed sensing, acting, and reasoning expert modules communicating via a common memory block in a coordinator, called the blackboard, which guides these agents to a solution. When used for intelligent control, the distributed nature of such systems exploits the advantages of parallelism (of behavior-based systems) and centralized high-level reasoning (of hierarchical systems) simultaneously. Figure 2.11 gives a schematic diagram of a typical blackboard control architecture.

Figure 2.11 Blackboard Control Architecture

One of the original implementations of a blackboard system for autonomous navigation was by Harmon [21] who used it to share information between tightly coupled sensing, control, and planning modules. Expanded "object-attribute-value" tuples were used for sensor data fusion by including measures of accuracy, confidence, and timestamps.

Payton [37] proposed a multi-behavior organization based on hierarchical control. It consists of low level modules performing tasks requiring a high degree of immediacy and high level ones executing tasks involving more assimilation of sensory data. With all levels operating in parallel, higher level modules exert selective activation and deactivation of lower level responses. During execution, the behavior choice is based on a fixed-priority system, without any mechanism to combine the results of concurrent behaviors.

The so-called "lateral decomposition" incorporates a high degree of flexibility and parallelism through the division of each module into a set of expert agents communicating with others over a common blackboard within the same module. The output of each module is then formulated by a local arbitrator which monitors the blackboard and is responsible for conflict resolution. Moreover, each expert sub-module has a perceptual component called "virtual sensor" and an action component called "reflexive behavior". This is illustrated in figure 2.12. This allows for the sharing of assimilated perceptual information by different behaviors. Several modules run in parallel, communicating through a common blackboard, and feed their control decisions to a command arbitration unit that selects the highest priority commands to control the vehicle actuators [37].

2.4.2. Advantages of Blackboard Architectures

The blackboard model provides a powerful and flexible paradigm for coordinating the interactions between capable subsystems. The advantage of such approach is that each subsystem contains local expertise in solving a specific problem, which allows the

coordinator to delegate much of the local work to these agents. This minimizes inter-component communication and leads to a system that can be implemented in a highly parallel manner.

Figure 2.12 Activation set of virtual sensors and reflexive behaviors
 Adapted from [37], fig. 2.

During the design process, it also provides a means for coordinating the programming activity associated with system development since subsystem developers have a well defined mechanism through which to exchange information with other subsystems. This, however, does not imply that a unique common representation should necessarily be used, as multiple representations can be handled by the same blackboard. Moreover, the constructive reasoning in the blackboard formalism is useful in reducing the effects of errors and uncertainty in a system with distributed knowledge sources [22].

2.4.3. Limitations on Blackboard Control Systems

Although the standard blackboard interaction paradigm provides many system programming benefits, it does not provide a realistic implementation model. Several restrictions have been outlined [43] on the use of standard blackboard systems. In general, it is required that all state information, no matter how trivial, be kept on the blackboard and that all such data be accessible to all knowledge sources. This "free-for-all" requirement means that there is no formal way of restricting data access to ensure its consistency. The lack of responsibility for data on the blackboard makes it difficult to decide when it is no longer required. This is undesirable since it leads to the accumulation of large quantities of useless information. Harmon [22] proposed using active processes to maintain the consistency of the data on the blackboard.

In systems that use any form of centralized intelligence, a crucial design factor is the decision about how much of the processing the central coordinator should delegate to the other agents and how much it should do itself [35]. The topology of inter-module connections is equally important because they determine how the components interact with each other.

Another dilemma to be solved is to provide support for "reasoning against time"; that is deciding on the appropriate timing for taking actions in the presence of uncertainties. Blackboard systems should be constrained to provide results in limited periods of time and are required to reason with continually changing input data [43].

2.5. Summary

As can be seen from the foregoing analysis, the choice of the control architecture has direct effects on the overall system performance. Any particular architecture seems to impose upper limits on crucial performance measures. While technological advances can improve the performance of specific design components, they cannot overcome such architecture-dependent limitations.

For instance, the series of stages through which sensory data flows in hierarchical architectures creates unavoidable processing delays. This, in turn, sets an upper limit on the reactivity of the control architecture to dynamic and unexpected events when operating in real-time environment, and restricts, therefore, the applicability of such system to certain control problems. Additionally, the need to implement and debug all modules in a hierarchical architecture before testing the whole system is a major handicap during the design phase.

In behavior-based architectures, the absence of any central form of intelligence causes such systems to lack common-sense. Despite all their breakthroughs in response time and reactivity to dynamic events, behavioral control architectures applications are limited to relatively simple tasks. Furthermore, the decomposition of the overall behavior into separate levels of competence is completely artificial and questionable. That seriously affects the system's overall behavior. Therefore, a more general and flexible relationship than strict hierarchy between behaviors is required.

Both types of decompositions possess their advantages and disadvantages: the behavioral decomposition is essential in the design of the control strategy while the functional decomposition, required at every behavior level, is important in the implementation phase. Therefore, one decomposition scheme does not completely outclass and/or replace the other.

Standard blackboards involve physical limitations on the response time and the storage capacity required by the reasoning module. However, blackboard-based systems provide enough design flexibility to use any appropriate system components and any suitable interconnection mechanism. They can also be efficiently used in this control problem to combine features from both hierarchical and behavioral system decompositions while avoiding their major shortcomings.

3. APPROACH

3.1. Current Impediments to Effective Design

The present state of computing technology requires complex robots to be implemented as concurrent systems. From the discussion in chapter 2, it can be seen that the currently available generalized concepts for robot control systems suffer from various flaws. The existing research experience has not yet produced a satisfactory paradigm for the distribution and/or coordination of the functionalities of autonomous robotic systems.

The literature survey from chapter 2 depicts various deficiencies in existing theories and support tools. While considerable experience has been gained in designing computing systems for robots, much of it is very configuration-specific and most of it is not rigorous or systematic enough to be applicable to different implementations. Additionally, while robots need to be reliable and predictable, there is not sufficient knowledge to design and implement reliable and predictable computing systems for such complex robots [22]. This situation leads to the use of control architectures which have been developed more from intuition than from a methodical analysis in most robot implementations.

The robot's inherent complexity and the lack of application experience make modifications during design a certainty [22]. Consequently, a need for flexible integration schemes arises in order to enable incremental development approaches and to permit easy system modification and expansion. Such schemes allow the integration of additional capabilities and/or the enhancement of the existing ones after some experience is gained while debugging an operational version of the system. Furthermore, the robot control architecture needs to be both conceptually and operationally flexible in order to allow for incremental and continuous modifications. This, however, is not typically found in the state-of-the-art systems. Also, the lack of software development tools for mobile robots seriously impedes the research in this field.

Similarly, very little performance evaluation has been done on mobile robots control strategies. The lack of any quantitative performance analysis tools inhibits efficient development because the significance of the resultant performance improvements cannot be accurately gauged. Consequently, comparisons between mobile robots are very difficult. Similarly, the lack of any common underlying theory prevents proper evaluation of the design options and restrains effective design simulation and analysis.

These factors not only make systematic design and implementation challenging, but they also make any meaningful testing of complex robot computing system performance very difficult. These problems were addressed separately in different implementations and were handled with different degrees of success in the architectures reported in chapter 2. Nevertheless, the identification of these deficiencies and impediments helps in setting the requirements of the control architecture and the performance measures to be used in assessing its design. This leads to an analytical and unbiased design approach and facilitates the investigation of different theories and control strategies.

3.2. Requirements of the Control Architecture

The complexity of the individual design components and their mutual interactions require a system-oriented design approach. A control framework must be formulated first, before the components are identified and built to implement the required capabilities within this framework. It is necessary to follow this sequence because, otherwise, an early choice of the components would strongly affect the overall design decisions, which is undesirable.

The specific problem in hand is the design of a sensor-based real-time control architecture to allow a vehicle to navigate autonomously through an indoor environment cluttered with both fixed and dynamic objects. For the sensor-based control system to exhibit any kind of intelligent behavior in a real-world environment, it has to meet a set of requirements. Therefore, it is important to identify the general behavior specifications needed for any type of intelligent control. Once these are specified and formulated, their design implications provide the outline for the implementation requirements. This is discussed in more detail in the following subsections.

3.2.1. Behavior Requirements

Reactivity: Since the vehicle is intended to navigate in an unstructured real-world environment, there is very little chance that it would be able to expect the dynamics of the world's configuration. Such an environment consists of fixed objects such as walls, doorways, and hallways, and other moving objects such as human beings or other mobile vehicles. Therefore, the control system must ensure that the vehicle reacts to unexpected events in a "reasonable" time while actively seeking the desired goal. Additionally, with the

continuous motion of the autonomous vehicle, sensory information becomes obsolete more rapidly, and consequently necessitates short reaction times.

Intelligence: At times when the vehicle is avoiding collision with fixed or dynamic objects that happen to be on its trajectory, it has to determine an alternative course to the goal. In most cases, there will be more than one such alternative. The vehicle's navigation system has to favor actions that optimize the path to the destination according to the acquired knowledge about the environment and based on the rules of basic common-sense. This requires the control system to provide for a global understanding of the situation in order to discard alternatives that might lead to dead-ends or local traps. Such a strategic high-level decision-making capability is contrasted with other situation-specific tactical decision-making capabilities. To ensure versatility and flexibility, the world image has to be independent of any predefined representations or world models. Therefore, this knowledge has to be incrementally built from the acquired sensory information in real-time.

Multiple Sensors: A robot control system that relies on a unique type of sensory information is bound by the limitations imposed by that sensor's physical characteristics and reliability. Therefore, an intelligent control system must integrate multiple sensors that cover a reasonably wide range of accuracy, reliability, and applicability. Multiple sensors are also required to allow the extraction of different and complementary features of the environment. This is necessary to avoid biasing the reasoning agents by the information of one particular sensor and to improve the overall interpretation of the sensory information into a sufficiently accurate world image.

Multiple goals: In a real-world environment, the control system of the autonomous vehicle should be able to handle concurrent, and possibly conflicting, goals. This is

essential to cope with situations where the ultimate goal cannot be within immediate reach. In such cases, short-term goals must be pursued first. The same requirement holds in other situations where the reasoning agent needs to reconcile conflicting goals that require contradicting actions in order to perform a complex task (for instance, passing through a narrow doorway while avoiding collisions, or approaching a target point near an object while avoiding contact with that object). The control strategy should identify which goals are more pertinent to the global solution and assess their relative priorities according to the situation in hand. This requires the support of multiple behaviors. These behaviors would be given different priorities for the different functionalities. An intelligent agent should decide on which behavior to follow based on the current knowledge about the world.

Robustness: The robustness of a system is perceived in its ability to deal with imperfect inputs, unexpected events, uncertainties, and sudden failures and/or malfunctions. Robustness is a crucial characteristic of a system intended to operate in a real-world real-time environment. The integration of multiple sensors is the first step towards increasing the system's robustness. This can also be further improved by using additional techniques for error prevention, detection, and recovery.

Reliability: The reliability of a system is measured by its ability to operate without failures or performance degradation over a certain period of time. Apart from the components' reliability in a compound system, there are issues concerning the overall system reliability. The control system should maintain a certain level of competence independently of environment-specific information; that is, the performance should not be influenced by the particularities of each situation. The mobile vehicle should exhibit equally sensible behaviors when exposed to different configurations and sets of goals.

3.2.2. Design Requirements

Section 3.2.1 outlines a small set of requirements to be met by the behavior of a sensor-based intelligent control system in general. The physical implementation of such a system is subject to another set of design requirements in order to satisfy the required behavior specifications. These are discussed in this section.

Modularity: The multiple sensors requirement, among other factors, increases the complexity of an autonomous robot control system. Consequently, the control problem has to be divided into smaller subsystems that can be designed separately. Since the optimal subsystem partitioning can not be known at the time of the initial design, advance identification of the best interaction paths between the system elements is not possible. Therefore, the complexity involved in the design of a robotic system and the large number of individual components therefore require a modular design approach. Using such a strategy is needed to allow the elements to be designed, implemented, and debugged independently. This is not only necessary during the design phase, but it is also crucial for the incremental enhancement of the overall system. As new technological solutions become available, the individual components can be replaced by more advanced, efficient, or reliable ones in a transparent manner without affecting the overall system architecture. Additionally, for maintenance purposes, failure detection and correction can be greatly facilitated when the faulty modules are easily identified and isolated.

Flexibility: Unlike other more theoretical research fields, experimental robotics, in general, and sensor-based intelligent control, in particular, do not have a strong supporting theory nor a large cumulative experimental knowledge. Consequently, it is not known, at

the current stage, what are the best design elements to use or which approach is best suited to combine these elements. This situation results in continuous changes in the design during the implementation phase. A flexible control structure would allow the design process to be guided by the demonstrated success or failure of one, or some, of the individual design elements. Fixed-structure control architectures impose their intrinsic characteristics on the problem solution and prevent the application of useful design components if they do not fit in the adopted control structure. A flexible architecture, on the other hand, would be easily modified based on the experimental results during the design, implementation, and debugging phases by changing the interconnection patterns between the different system components. Such an approach leads to better overall system structure and more reliable design decisions.

Expandability: Besides the system's design flexibility requirement, there is an issue concerning its implementation flexibility. A long time is required to design, build, and test the individual components of the control system independently. Therefore, it is desirable to be able to build the control structure incrementally. This would (1) allow smooth addition of components to the system as they become available, (2) provide a means for adaptable system integration, (3) reduces overall delays if the design of one module takes longer than anticipated, and (4) enable subsequent improvement of the system performance and versatility by adding more capabilities besides existing ones without disturbing the general control structure.

Adaptability: There is a well-defined control strategy associated with each skill that the system should exhibit. Such a strategy focuses the attention of the reasoning component on particular aspects of the environment. It identifies the prerequisites to perform the task, matches them with the current state of the world, and eventually carries out the task if these

conditions are met. In a real-time environment, a large number of separate capabilities is required to cope with the diversity of the possible situations. Consequently, the control system should be equipped with a reasonable set of control strategies to support these various capabilities. Furthermore, since the state of the world changes very rapidly and unexpectedly, the control system must support a flexible mechanism to switch smoothly between the different control strategies according to the current situation. This is necessary to satisfy the reactivity and intelligence requirements.

Centralized high-level decision-making capability: The knowledge acquired by any perception subsystem can only be partial and incomplete due to the physical limitations of the sensors and the possible bias in interpretation. The use of multiple sensors partly solves this problem by increasing the amount and diversity of the acquired knowledge. However, a global reasoning agent, responsible for the understanding of the overall situation, can account for the errors introduced by misinterpretation of the sensory data. Such a centralized agent would gather high-level knowledge from the different perception modules, and merge them into a global image of the actual state of the world. This global understanding is fundamental in assessing conflicting actions arising from dynamic world parameters and to fulfil the intelligent behavior requirement.

3.3. Control Architecture Decomposition

Now that the required characteristics of the real-time intelligent control architecture have been stipulated, the suitable design elements can be selected. These characteristics necessitate a highly parallel modular architecture to facilitate the use of multiple sensors and

the support of multiple concurrent goals. Additionally, adopting a modular design approach satisfies both the expandability and the flexibility requirements. It is also required to decouple the "high-level" strategic reasoning actions from "low-level" tactical reactive responses. It was observed [30] that sometimes the physical computing environment and the choice of the components have a strong and undesirable influence on the system architecture. Therefore, in order to avoid such a pitfall, the choice of the physical implementations of the components is postponed until the logical design is complete.

To this end, and unlike hierarchical and behavioral structures, the control problem is decomposed according to the type of basic activities that the system is to perform and the skills it is intended to exhibit. The design should allow enough flexibility to establish the proper interconnections between the sensing, perception, and control modules to perform those activities. That is, the system should be able to switch its focus of attention according to the changes in the environment, and to select the appropriate set of modules to activate in an intelligent fashion.

An activity is an organizational unit capable of successfully performing a specific function and to exhibit a basic skill. Examples of such activities are: following a person, passageway traversal, hallway navigation, crossing a doorway, wall following, and channel crossing. Each activity includes the sensors, perception modules, and decision-making modules necessary for it to carry out its task. For the vehicle to be capable of performing a complete task, such as "navigate from one room to another", it should be able to combine the desired activities to perform this task in the proper manner. Furthermore, it must be able to monitor the proper operation of these activities and to react to any unexpected events in the environment during their execution. Figure 3.1 shows the details of an example of such an activity as a stand alone system. The navigation algorithm used in

this activity is based on previous and independent work on collision avoidance [32]. It was adapted to be used as a passageway activity in the experimental analysis of the proposed control architecture.

Figure 3.1 Passageway activity
From [29], fig. 1.

To ensure system reactivity and responsiveness to changes in the real-time environment, the traffic of information between the perception agents and the reasoning agents should be kept to a minimum in order to avoid communication delays. This suggests that the information being exchanged should be of a high level of abstraction, that is, processed data as opposed to raw sensory information. This, in turn, requires the perception modules to process the raw data and to extract the relevant environmental features in the observed world state. These feature extractors should provide the high-level information in real-time to guarantee that the control decisions are based on the most recent

knowledge about the environment. Additionally, they should have relatively short processing times to ensure the reactivity to dynamic events.

A reasonable set of feature extractors, connected to multiple sensors, together with an appropriate number of action modules would provide the mobile vehicle with the basic skills necessary for indoor navigation. The performance of such a control system would be comparable to that of a pure behavioral architecture in terms of the response time and the overall behavior exhibited. Although such a system can navigate the vehicle in a cluttered environment with no collisions, it would be vulnerable to several pitfalls such as local traps, dead-ends, and limit cycles. The main reason is the lack of common-sense and of a global understanding of dynamic events.

Therefore, a centralized global high-level reasoning element is needed to ensure an intelligent overall behavior. This strategic decision-making agent must have access to all the available high-level knowledge in order to reason according to the global state of the environment. Furthermore, the intelligent control unit must be independent of the other system components' organization and interactions. It should act more as an intelligent symbolic information processor rather than as a command generator. This advocates that the generation of the required tactical control actions should be delegated to the activities. Only the output of one such activity is selected, according to the control strategy adopted by the decision element, and is routed to the actuators to physically drive the vehicle. Such an organization decouples the high-level assimilation and reasoning actions from the low-level reactive responses.

The blackboard concept meets these specifications and provides the required support for symbolic information manipulation and decision-making. Additionally, it can

accommodate multiple control strategies, which simplifies the support for multiple goals. Therefore, it is used as the high-level decision-making element in the proposed architecture. It is used to assimilate the global situation based on the available abstracted sensory information and the user-specified task, and to guide the system accordingly to the desired goal by appropriate sequencing of the activities. The general outline of the proposed Blackboard Activity-Based control framework is shown schematically in figure 3.2. A detailed description and analysis of this architecture can be found in chapter 4.

Figure 3.2 Outline of the activity-based intelligent control architecture
From [29], fig. 2.

Table 3.1 gives a comparison of the analysis of these requirements in the traditional system structures as well as in the proposed control framework. The relative advantages and weaknesses of these architectures can be observed.

	Hierarchical Architectures	Behavioral Architectures	Blackboard Systems	Blackboard, Activity-based Framework
Reactivity	Low	High	Medium	High
Intelligence	High	Minimum, or None	High	High
Multiple Sensors	Yes, with Difficulties	Yes	Yes	Yes
Multiple Goals	Yes	Yes, with Difficulties	Yes	Yes
Robustness	Very Low	Very High	Medium	High
Reliability	Low	High	Medium	Medium-High
Modularity	Yes	Yes	Yes	Yes
Flexibility	No	No	Yes	Yes
Expandability	Yes	Yes	Yes	Yes
Adaptability	No	No	Yes	Yes
Centralized Decision-Making	Yes	No	No	Yes

Table 3.1 Comparison of the different control structures

3.4. Analysis of the Design Approach

The use of activities representing the basic skills that the vehicle is intended to exhibit is, to some extent, similar to the behavioral decomposition of the control problem. It provides multiple pathways for sensory information to guide the control: short pathways between sensing and action, from the feature extractors to the action modules, and longer ones through the high-level reasoning element. This ensures the coverage of a wide range of the immediacy/assimilation spectrum. Nevertheless, activities in this architecture differ from the behaviors proposed by Brooks [11] in that a behavior is a response of an

individual or group, the mobile vehicle in this case, to the environment. Examples of such behaviors are: "decelerate if an object is ahead" and "turn ± 90 degrees and move if stopping". An activity, on the other hand, is a goal-oriented function which is subject to situation-specific restrictions. Examples of activities are: "navigation in an unknown open-space" and "safely crossing a channel on the floor". For instance, traversing a hallway with the intention of keeping to the right of the hallway cannot be implemented only as a simple reactive response to the environment; it is an action based on the understanding that generally most of the traffic flows in this manner.

In this control architecture, the reactive tight control loops, involving only the sensors, the feature extractors and the action modules, provide reasonable tactical control decisions based on the available sensory information. This satisfies the reactivity requirement and facilitates the integration of multiple sensors. On the other hand, the longer control loops, including the blackboard and its associated control unit, provide a clean way to merge the accumulated knowledge about the environment in order to have a global understanding of the world state. The intelligent purposeful behavior requirement could not be achieved without such a centralized decision-making component. Additionally, the feature extractors are operating at all times, irrespective of which activity is enabled, which ensures that all the relevant features in the environment are recorded as facts posted on the blackboard. This increases the system reactivity to dynamic events by alerting the blackboard of any changes. It also modifies the system's focus of attention, which fulfils the adaptability requirement.

There is an issue, however, concerning the choice of the appropriate set of basic skills to be implemented in order to achieve a certain task. The choice strongly depends on the personal understanding and interpretation of the human behavior, and, therefore, it is

prone to error. Compared to behavioral architectures, it seems that it is better to consider the type of independent skills that the system is to exhibit rather than the progressive levels of competence it should exhibit. Although the number of activities that we can imagine is quite large, it is probably easier to produce a purposeful and conscious "behavior" from such activities than from an equal number of behavior layers. Another important advantage of this system structure over behavioral decomposition is the independent and complementary nature of the different activities. These two features are essential in the implementation and debugging phases during which the best interaction paths are not known in advance causing continuous changes in the design. Furthermore, such a structure is inherently flexible and easily modifiable, that is, it satisfies both the flexibility and expandability requirements.

This system architecture has some similarities with the "activation sets" proposed by Payton [37]. These are primarily in the use of features extractors - analogous to his "virtual sensors" - and in the flexibility in switching the mode of operation in real-time. Unlike the virtual sensors which detect very specialized environmental attributes, the feature extractors supply high-content information. Additionally, the virtual sensors are part of a layered perception hierarchy, which is not the case with the feature extractors. The notion of using independent feature extractors has the advantage of allowing the multiple sensors to feed into any or several of the perception modules. This reduces the redundancy in the sensory processing and facilitates sensory information integration. It also maximizes the use of the available sensory data by allowing the sharing of the assimilated information between a variety of activities.

The main difference between this control architecture and standard blackboard-based architectures is the granularity level of the agents interacting through the blackboard.

In other blackboard-based architectures, the agents react to the information available on the blackboard and provide their contribution to the global solution by posting it on the blackboard. The role of the blackboard consists of enabling those agents that are more likely to contribute to the solution. The global solution is incrementally built on the blackboard. In this Blackboard Activity-Based control architecture, however, the blackboard selects the appropriate activity to handle the situation in hand, or affects the parameters governing the action module of the enabled activity. This is done based on the information provided by two types of knowledge sources: (1) contributors, which are the feature extractors providing environmental information, and (2) deciders, which are the action modules providing state information. Each of the available activities can be thought of as a stand-alone system which achieves a complete task. The information exchange between the activities and the blackboard in this architecture is limited to high-content symbolic information. The global solution to the navigation problem is the emergent behavior of the vehicle in response to its environment.

In order to reduce unnecessary information processing overhead, perception should be action-directed. During operation, only one activity is enabled to process data and issue corresponding vehicle control commands. Therefore, only those feature extractors which support this activity need to connect to it when it is enabled. Since the decision on which activity should be enabled requires a global understanding of the states of both the environment and the vehicle, the high-level reasoning/decision-making module should perform the selective activation/deactivation of the appropriate modules. This does not contradict with the requirement of keeping the sensing modules active at all times to alert the reasoning agent about the changes in the dynamic environment.

Unlike traditional hierarchical control system architectures, the high-level reasoning agent in this architecture does not act as planning module, nor as a servo-controller. In this context, it only integrates symbolic information about the environment to develop a global understanding of the situation.

The major contribution of this high-level reasoning element is that it provides the possibility to incorporate common-sense in the decision-making process. It does not affect the reactivity and responsiveness of the lower-level reactive components, namely, the action modules. It only performs selective activation or deactivation based on its acquired and synthesized knowledge.

Table 3.2 summarizes the relevance of each of the design elements to the proposed architecture and their contributions to its overall performance and merits.

3.5. Summary

To avoid the undesirable influence of the early implementation specifications on the design of a control system, the structure of the control architecture was defined independently from any implementation issues. An explicit design approach was set, its elements defined, and its performance characteristics specified. The analysis of section 3.4 above shows that this results in a system structure that satisfies most of the general requirements initially speculated for any sensor-based intelligent control system. Additionally, it avoids many of the shortcomings of the rigid-structure control architectures previously used in other projects.

Component Specification	Sensing	Feature Extractors	Action Modules	Blackboard
Reactivity	+	+	+	
Intelligence		+	+	+
Multiple Sensors	+	+		+
Multiple Goals			+	+
Robustness	+		+	+
Reliability		+		+
Modularity	+	+	+	+
Flexibility	+	+	+	+
Expandability	+	+	+	+
Adaptability		+		+
Centralized Decision- Making		+		+

A (+) means this component contributes to this specification.

Table 3.2 Analysis of the design and the requirements

The main advantage of this control framework is that it provides an extremely flexible way to integrate different sensors, control strategies, and capabilities. Furthermore, it combines both the required reactivity to dynamic environments and the high-level reasoning capability necessary for intelligent and purposeful behavior. Its inherent flexibility is of great advantage in the design phase, and allows easy adaptability during operation.

The robustness of this control architecture can only be evaluated after the implementation of a reasonable number of activities in order to be able to circumvent the

effects of sudden failures and or imperfect inputs. The problems of error prevention, detection, and recovery were not handled in this architecture. They may be incorporated in the reasoning agent's functionality. These issues need to be addressed in order to satisfy the robustness and the reliability requirements.

4. SOLUTION

4.1. Experimental System

In chapter 3, it was argued that the early choice of the physical components of the control system may have undesirable influences and biases on the design. Therefore, the actual implementation was postponed until the completion of the logical design of the whole control framework.

Before discussing the implementation of this control architecture, it is important to describe the control testbed used for the experiments. Figure 4.1 is a photo of the experimental autonomous vehicle "eAVe" used at the Autonomous Systems Laboratory of the National Research Council. It also shows a simple block diagram of the system configuration. The platform is commercially available from Cybermotion Inc. [26]. It allows the use of the on-board odometer for position measurement. The vehicle's motors are controlled by a manufacturer-implemented algorithm using a Z-80 microprocessor. Communication with this controller uses a serial line and is synchronized to the 10Hz vehicle servo rate. The vehicle is equipped with a ring of 24 Polaroid ultrasound transducers [41] which are controlled via a custom built command interpreter [28].

Figure 4.1 Photo and block diagram of the experimental vehicle

Both the vehicle controller and the ultrasound command interpreter are interfaced using separate serial lines to an on-board 5-processor MC68020 VME-based multiprocessor system which runs under the Harmony³ operating system. This multiprocessor is the computing testbed used for the proposed architecture. Harmony is a multi-tasking multi-processing operating system designed for real-time applications. It exploits the concept of multiple tasks communicating information and commands through a message passing mechanism and has been previously used to control robots in a variety of contexts [19, 20].

³ Harmony is a mark reserved for the exclusive use of Her Majesty the Queen in right of Canada by the National Research Council of Canada.

4.2. Description of the Actual Control Structure

Now that the logical structure of the control architecture and the testbed are described, the details of the implementation can be discussed. Figure 4.2 shows the interconnections between the different components of the Blackboard, Activity-Based control framework. It was mentioned in chapter 3 that this architecture consists of two main control streams. These are namely (1) low-level reactive numeric control loops, each of which involving several sensors, feature extractors, an action module, and the vehicle actuators, and (2) a high-level symbolic control loop which controls the interconnections of the lower level control loops.

Figure 4.2 Block diagram of the control framework

Each low-level numeric control loop depends on one or more sensors. These are continuously operating and feed their raw sensory information to the appropriate perception modules called feature extractors. The sensory data is processed in these perception modules and is stored in suitable representations. The relevant attributes of the environment

are expressed in a symbolic form to support the high-level symbolic control loop. Some control-dependent features are supplied to the only enabled action module in order to provide the necessary control signals, namely direction and speed, to steer the vehicle.

The reasoning of the high-level symbolic control loop relies on the symbolic information extracted by the perception modules to develop a global understanding of the state of the world and that of the vehicle. These relevant features are stored on the blackboard, and are used to drive the high-level reasoning module. The rules forming the blackboard's production system have the general form "*IF condition THEN action*." They are fired (triggered) if the symbolic features satisfy the conditions of their left-hand-side (LHS). These rules enforce the control strategy adopted by the reasoning module. Their right-hand-side (RHS) consists of actions which manipulate the states and the interconnection of the modules of the lower level loops. State information also flows from the action modules, through a reporting mechanism, to the blackboard, in order to feedback the progress of the actions being carried on at the lower level control loops. In this manner, the reasoning module guides the system to perform the task in hand based on environmental information, from the feature extractors, and progress information (success or failure), from the action modules.

Due to the lack of efficient standard supporting tools for real-time systems, an expert system shell was adapted to build the high-level reasoning agent. It was modified twice: first to incorporate the blackboard concepts, and then to operate in, and interact with, an asynchronous real-time environment. A numeric/symbolic interface was implemented to link the blackboard, which handles symbolic information, and the real-time activities, which process numeric information. This interface uses a serial communication protocol of packaged messages.

In order to allow easy debugging of the components' interaction during real-time operation, relevant information about the states and the operation of the different system modules are transmitted to a common monitoring module. The accessibility and use of this monitor is not limited to debugging the system during the implementation phase, but it is also instrumental during normal operation to selectively display and examine the behavior and interaction of the different modules.

In the following sections, the implementation of the logical system components are described in more detail. To ensure portability and modularity, the different components, with the exception of the blackboard system, are implemented as real-time packages. A package is essentially a stand-alone module having a well defined function. The software building blocks of the system are Harmony tasks which are program objects that can run concurrently and exchange information via message passing. Although tasks are the actual building blocks, packages are considered to be the conceptual building blocks of the control architecture. That is, when a package is created all its constituent tasks are created, and they are all destroyed when it is destroyed. A package typically consists of an active task, called a "server", and a set of auxiliary tasks. A schematic diagram of the structure of such a typical package is shown in figure 4.3. These secondary tasks are meant to free the server from long processing and/or communication delays: they may perform heavy computations or may be used as "courier tasks" that connect to servers of other packages. Servers were originally given this name because they were intended to control the access to a shared resource and ensure the mutual exclusion of its use. They are, therefore, designed to operate in such a fashion as to be continuously ready and waiting for calls for service from different - and possibly unknown - clients, and to ensure that they are never blocked. Although packages are disjoint by definition, they can communicate through clean interfaces by passing messages between their servers [18].

Figure 4.3 Package structure

4.3. The Activities

Each low-level numeric control loop is based on a specific behavior-generating action package, and comprises three other general packages. These general packages run continuously and interact with whichever action package that is enabled. They are namely the reporting, monitoring packages, together with the feature extractor package corresponding to the current activity. The basic components of any given activity are illustrated in figure 4.4.

Figure 4.4 Detailed activity decomposition

4.3.1. Feature Extractors

Although a feature extractor may depend on more than one sensor, it provides information on some specific attribute in the environment. Such a feature extractor is depicted in figure 4.5. Sensors, and their associated servers, are continuously operating and feed their raw sensory information to the servers of the perception modules, i.e., the feature extractors, via appropriate courier tasks. The sensory data is manipulated, processed, and then expressed in a suitable representation. The choice of the representation is local to the feature extractor, i.e., the adopted representation is not required to fit in a general centralized world-model. The selection is made in such a way as to ease the detection and expression of the pertinent attributes.

Figure 4.5 Feature extractor

The relevant attributes of the environment are derived by the symbolic information extractor task and are expressed in a symbolic format. The output of these modules includes some parameters describing the feature of interest as well as some indication on the importance of the issued signal. It is then relayed to the blackboard, via the interface package, in order to support the high-level symbolic control loop. Since the feature extractors are continuously operating, irrespective of which activity is enabled, the blackboard reasoning module is notified about the dynamic changes in the environment as soon as they occur, and is, therefore, always up-to-date. It is the responsibility of each feature extractor to keep record of the information it sends to the blackboard to ensure its consistency. The truth maintenance task, a typical problem in standard blackboard systems, is distributed on the agents in this design. This relieves the reasoning module from the time-consuming task of continuously checking the consistency of its knowledge.

4.3.2. Action Modules

At any time during navigation, the enabled action module determines the actual behavior of the vehicle in terms of the direction and speed. In order to generate conscious control commands to the vehicle actuators, the server of this action module typically creates an associated control information extractor task, as shown in figure 4.6. The responsibility of this task is to open a connection to the server of the corresponding feature extractor and to use its representation to derive relevant control-specific information. This ensures that the low-level decisions are sensor-driven in order to achieve tight and reactive control.

Figure 4.6 Action module

For the high-level reasoning module to intelligently influence and guide the real-time operation, the decision of the action module also takes into consideration a set of high-level commands issued by the blackboard. These commands are the desired direction and speed which are based on the global understanding of the situation. Nonetheless, it is the responsibility of the action module to provide the actual control signals and to manoeuvre the vehicle to eventually meet the recommendations of the reasoning module. The high-level commands may also consist of constraints on the operation of the action modules in order to restrict their operation to within predefined limits. This is particularly useful for the smooth transition from one activity to another.

Another function of the action package is to provide a collection of set points for the control signals of the vehicle actuators. A separate task may be used for this purpose. These computations should reconcile the desired settings, coming from the blackboard reasoning module, and the sensory-based information, coming from the feature extractor. The control set points task ensures that these signals are fed to the vehicle steer and drive motors in a synchronized fashion.

Consequently, a typical low-level control loop originates at one or more sensors with their respective servers and terminates at the vehicle actuators and their corresponding servers. It links these two ends, namely sensing and action, through a sequence of tasks and servers consisting of a feature extractor server, a control information extractor, and an action server. The feedback loop is closed through the interaction of the vehicle with the environment. The behavior of such a control loop achieves one "elementary" navigational activity. Figure 4.7 combines the different components of typical low-level control loop and indicates their interconnections.

Figure 4.7 A low-level control loop

In order to achieve effective navigation, the state (success or failure) of the action module together with its important local decisions are reported to the reasoning module to be taken into consideration when issuing subsequent high-level commands. This is done through a reporting mechanism that closes the high-level feedback control loop. Both the reported and monitored information are updated whenever a change occurs in the state of the action module server. This ensures that the performance of the low-level control loops can be closely observed by the reasoning module and the operator during operation.

4.4. The High-Level Decision-Making

The responsibility of the reasoning module is to assimilate the global situation in real time and to generate conscious decisions accordingly in order to guide the system to accomplish the user-specified task. The operation of this reasoning module should only deal with high-content symbolic information in order to achieve global reasoning. A rule-based production system using the BB_CLIPS⁴ blackboard system is used as the decision-making mechanism. BB_CLIPS is an extended version of CLIPS⁵, an expert system tool written in the C language to ease portability and integration with external systems. The main advantage of CLIPS is that it supports two kinds of integration mechanisms: (1) it can be embedded in a larger system if it needs to share data with other subsystems, and (2) it supports calls to external functions by its rules. This easy addition of external functions allows CLIPS to be extended or customized in different ways.

A version of BB_CLIPS, running on a Macintosh IIfx computer, was augmented to interact in real time with the Harmony operating system using a serial communication protocol. The important information to be shared by the different system components is stored on the blackboard in a symbolic format. Such information includes the sequence of goals to be achieved, the currently used control strategy, and the acquired high-level knowledge about both the state of the world and that of the robot.

⁴ BB_CLIPS was developed in the Knowledge Systems Laboratory, the Institute for Information Technology, the National Research Council of Canada. It implements some features of the blackboard control architecture.

⁵ CLIPS was developed by the Artificial Intelligence section, Lyndon B. Johnson Space Center, NASA, and is available from COSMIC, the University of Georgia.

The tasks specified by the user in a symbolic form are stored and processed by the task decomposition module⁶. They are analyzed and decomposed into their primitive components. That is, they are split into sequences of successive activities that need to be carried out in the appropriate order to achieve the desired goals. The sequence of intermediate goals is fed to the agenda of BB_CLIPS with the appropriate priority levels. They are acted upon by the inference engine which attempts to match them with the facts asserted on the blackboard. A simple block diagram of the logical structure of this reasoning module is given in figure 4.8.

Figure 4.8 The blackboard system

⁶ The task decomposition module was not implemented since it is beyond the scope of this work.

The Harmony interface of the blackboard system is responsible for the two-way asynchronous serial communication with vehicle. Incoming symbolic information is asserted as facts to the blackboard to be acted upon by the rules. Similarly, the rules on the agenda assert symbolic commands on the blackboard as facts and invoke the Harmony interface to send these to the action modules.

4.4.1. The Knowledge Sources

While the original set of goals that needs to be satisfied is provided by the decomposition of the user specifications, the actual execution of these tasks depends mostly on the interaction of the reasoning module with the environment through the activities. Therefore, although the static knowledge of the reasoning system is task-dependent, its dynamic decisions depend on the feature extractors which act as dynamic knowledge sources. They are continuously notifying the blackboard system of facts corresponding to relevant changes in the environment. These facts trigger the production system and fire its rules; this results in the issuance of the control commands necessary to drive the low-level control loops and to guide them towards the achievement of the desired goals. Figure 4.9 describes the interconnections between the modules of a typical high-level control loop.

This two-way communication between the reasoning module and the low-level control loop uses the symbolic/numeric control interface. It consists of a serial communication protocol server on the Harmony operating system, and a similar module on BB_CLIPS side. The feature extractors send their symbolic information about the environment to the interface server which relays them to the blackboard. Similarly, the

commands issued by the blackboard are fed to the same server which includes a task to dispatch these commands on the corresponding system components. The symbolic nature of the information exchanged over this serial link ensures that the communication delays are short compared to the processing times on both ends and that they do not represent an overhead on the system's performance.

Figure 4.9 Basic structure of a high-level control loop

4.4.2. The Production System

The primary representation method is a forward chaining rule language based on the Rete algorithm [15]. The LHS of the blackboard's rules denoted as "condition" is a set of boolean expressions, also known as clauses or predicates, that need to be satisfied

simultaneously in order for the rules to be fired. It may also include calls to external "predicate" functions. If a rule is fired its RHS denoted as "action" is executed. It may consist of one or more blackboard manipulations and/or control related commands.

The rules of the production system determine its behavior in response to the different and continuously changing situations. The main function of this reasoning system is the arbitration between the incoming notifications based on the acquired knowledge in order to provide sensitive decisions. Therefore, the selection of these rules requires special care, and usually experience, in order to ensure acceptable performance under the various circumstances. They should include enough knowledge to provide decisions that comply with basic common-sense regardless of the vehicle's situation. Since different activities may require distinct control strategies, the set of rules governing the system's operation may be changed at run-time. BB_CLIPS provides a means to control the reasoning action through a "control blackboard." This option would incorporate rules which affect the rules' priorities and/or the strategy used to handle rules on the agenda.

The flexibility of the set of rules which drive the reasoning module allows the use of multiple control strategies in response to the different situations. This provides an easy means to dynamically draw the focus of attention of the reasoning module to the attributes of the environment which are most relevant to the activity in hand.

4.4.3. The Actions

The RHS of any rule on the agenda of the blackboard is not limited to blackboard oriented manipulations, such as assertion of new facts or retraction of existing ones, to

guide the reasoning mechanism. It may also include some control-oriented decisions in the form of activation or deactivation of the action modules of the low-level control loops or changing their parameters and their constraints. Furthermore, the action side of the rules can invoke calls to software functions to perform any necessary computations.

The relevance of the RHS of these rules is manifested in the control-oriented commands which manipulate the operation of the low-level control loops. These control decisions are required to adapt the parameters of the enabled activity in order to exhibit the desired behavior, or to deactivate it and enable another suitable activity, while maintaining a smooth and uninterrupted operation. This latter situation requires the adjustment of the parameters of the two successive activities in such a manner as to make the transition transparent to the user. These control commands, which are symbolic in nature, are sent to the action servers.

4.5. Symbolic/Numeric Control Interface

A two-way communication is needed between the two control loops in order to achieve a conscious and reactive overall behavior. The perceptions and the states of the low-level loops should be carried upwards, or fed back, to ensure the reactivity of the high-level loop. Similarly, the intelligent decisions of the high-level control loop should be propagated downwards, or fed forward, to manoeuvre the operation of the low-level control loops in order to provide a sensible overall behavior. The information to be exchanged should be dense in order to minimize communication overhead, and should, therefore, be limited to high-content symbolic information.

Since the main components of the two control loops reside on two different hardware bases, namely the multiprocessor running Harmony and the Macintosh computer, a communication protocol was implemented to support the required information exchange. A serial communication software application was built by the author as a real-time package at the Institute for Information Technology of the National Research Council. Figure 4.10 describes the structure of this interface package. Its server is used as the heart of this symbolic/numeric control interface. The interface is transparent to the operation of both loops. That is, as far as the numeric control loops are concerned, the interface is considered as a normal package: symbolic information is sent to it, or received from it, using the regular Harmony message passing mechanism. Similarly, as far as the symbolic control loop is concerned, the symbolic information coming from the numeric control loops is asserted on the blackboard as regular facts. And similarly, among other regular blackboard oriented actions in the RHS of the fired rules, there may be some calls to functions to send symbolic control commands to the numeric control loops via the interface.

The upwards flow of symbolic information consists of two main streams: (1) perception information from the feature extractors describing the detected attributes in the environment and their relevance, and (2) progress reports from the action modules of the numeric control loops about their success, progress, or failure, in achieving the desired goals. These two types of information are essential in keeping the high-level decision-making system aware of the continuous evolution of the state of the world and the behavior of the action modules, in order to develop a global and dynamic understanding of the situation. This, in turn, is required to generate sensible and conscious decisions. On the other hand, the downwards information flow consists of control commands to the action modules to enable/disable them or to change their operating parameters. These commands are dispatched to the appropriate action modules via an intelligent "command processor"

courier task in the interface package. This command processor task, which has access to the contents of the messages, routes the control commands to the server of the appropriate action module.

Figure 4.10 Symbolic/numeric control interface

4.6. Monitoring and Debugging

During the implementation phase of such a complex system, there is a need to closely observe the interaction of the different modules in real time, and to study their mutual effects. Similarly, during the iterative design phase, this is a crucial function for it ensures that the design decisions are feasible and realistic ones. It is particularly relevant to the timing of the changes in the internal states of the different modules. Such a monitoring

facility can be used effectively as a nucleus for a real-time debugging tool to analyze situations of failure and success, which was not typically found in existing mobile robot control architectures. A collection of cases may be grouped in a library that could be used subsequently to incorporate a case-based learning mechanism.

Moreover, during normal real-time operation, the global behavior of the vehicle is the outcome of the interaction of several activities which are orchestrated by the blackboard decisions. Therefore, it is necessary to provide some means to observe the progressive changes of the internal states and local decisions of the action modules in order to understand the reasons for its behaviour. Figure 4.11 is a simple diagram of this package.

Figure 4.11 The monitoring/debugging package

For these reasons, a separate package is added for monitoring and debugging purposes. It asynchronously receives (1) information on the extracted features from the

symbolic extractors, (2) state and decision information from the action servers of the different activities, (3) vehicle drive and steering control set points, and finally, (4) blackboard control decisions. This information is used to update a display panel in real time. It is also used to build a log of the interactions between the different modules and saved for later processing and analysis.

4.7. Physically Implemented System

In order to facilitate the physical verification of the merits and weaknesses of this control framework, it was decided to implement a basic experimental control system. Therefore, a sample problem was addressed, and this approach was applied in an attempt to provide a possible solution. Among the factors that affected the choice of the problem to consider were: (1) a desire to experiment with a novel and resourceful problem, and (2) a strong emphasis on dealing with a realistic problem and a realistic environment.

The general area of autonomous navigation in a partially known workshop environment provides a wide variety of specific problems to be addressed. It also requires a set of basic skills to be exhibited by the vehicle during navigation in order to interact intelligently with the unstructured environment of the workshop floor. These problems need to be solved individually but their actions should be integrated in real time to provide a reasonably intelligent overall behavior.

Among the basic skills that we thought essential for the autonomous navigation in such an environment are the passageway navigation, doorway traversal, channel crossing,

and following a moving object (maybe a person). Due to time limitations only the solutions of two of these problems were integrated; these are namely the passageway activity and the channel activity. Therefore, the resulting control system is only considered as a concrete example of the ideas developed in this research. However, it is far from being a complete one, and should only be viewed as one instantiation of this control framework.

4.7.1. The Passageway Activity

The passageway activity, shown in figure 4.12, is responsible for the navigation of the vehicle in any open-space. Its feature extractor gathers sensory information from several sources and integrates them in some internal representation. In the currently implemented system, these sensors consist of the ultrasonic transducers and the vehicle's local odometry. The representation used in the actual system is a polar histogram where the range and certainty information about the closest object in each of 48 angular sectors is stored and continuously updated. From this data, the feature extractor identifies the open-spaces, extracts and monitors their relevant attributes, and provides the action module with control specific information.

The action module, on the other hand, drives and steers the vehicle intelligently in these open-spaces to reach the goal locations, while avoiding obstacles. It is based on previous work on local collision avoidance algorithms [32]; this work was modified and augmented to fit this open control architecture and to interact with its key modules. The current algorithms manipulate the linear and angular velocities of the vehicle to find the closest opening to the desired direction.

Figure 4.12 The passageway activity

4.7.2. The Channel Activity

The floor of a typical workshop would not be expected to be a particularly even or tidy one. Most probably, it would have some irregular-shaped objects, such as heavy machines and bulky tools. Furthermore, there may be some obstructing grooves, channels

or pipes in the floor. Figure 4.13 gives a schematic diagram of the different possible irregularities which can be typically found in a workshop environment. It is the responsibility of the channel activity to manoeuvre the vehicle to safely cross such floor level irregularities. The vehicle is not expected to avoid these channels; it is required to identify the feasibility of crossing them, and eventually take the necessary actions to do that safely.

Figure 4.13 Irregularities of a typical workshop floor

Such a channel activity was created for this purpose; its global structure is shown in figure 4.14. The feature extractor of this activity relies on the available sensory information to identify the channels present in the vicinity of the vehicle. It extracts the relevant attributes to describe them and stores these in a suitable local representation. Such attributes may include: the depth (or height), width, orientation, and distance of the detected channels. In addition to these control-specific attributes, the symbolic feature extractor task derives some other attributes for the symbolic reasoning loop; these may include a measure of relevance of the individual channels detected and a priority level to be attributed by the blackboard. In this system, the sensory information is simulated since developing an algorithm to extract the channel information was beyond the scope of this work.

Figure 4.14 The channel activity

Meanwhile, the channel feature extractor notifies the action module by sending the control-specific attributes. The action module, in turn, uses its local intelligence to operate the vehicle and successfully manoeuvre it across the channel. This requires mechanisms to avoid the trapping of the wheels, assuming the case of a wheeled vehicle, in the groove, or to avoid the sliding of the wheels along a pipe or a hose on the floor. This requires that the vehicle temporarily abandons its original trajectory to handle the obstructing channel, before it eventually regains its normal course. The exact details of the transitional trajectory

are calculated with respect to the actual parameters of the situation such as: (1) the angle between the vehicle's trajectory and the groove's orientation, (2) the vehicle's speed, (3) the depth or height of the channel, and (4) the orthogonal distance to the channel.

4.7.3. An Example of System Behavior

In order to illustrate the interaction of the different system modules which achieves the successful navigation in such an environment, a possible scenario was developed and depicted in figure 4.15.

Figure 4.15 Possible scenario in a workshop

Referring to figure 4.15, the vehicle is originally at location (0) and is required, by the blackboard, to navigate autonomously to the destination point (14). The goal location is

only given in terms of its coordinates, absolute or relative. The vehicle, perceiving no obstructions proceeds in the direction of point (14) until it reaches point (1) where the open-space extractor notifies the blackboard and the passageway action module of the presence of an obstacle. It also realizes that the space between object (II) and the south wall is not wide enough, and therefore, it decides to seek an open-space in another direction. The blackboard, realizing the situation, directs the passageway activity to proceed in that direction temporarily. The vehicle reaches point (2) where the channel feature extractor detects the channel (AB). The blackboard commands the channel activity to take over, temporarily, to manoeuvre around this channel. The vehicle crosses the channel safely. At point (3), the blackboard releases the passageway activity after being notified by the channel activity of the successful completion of its task. The open-space feature extractor detects the object (I) and the open-space between objects (II) and (III). The blackboard realizes that going through this passageway would bring the vehicle closer to the original goal point, and commands the passageway activity accordingly.

At point (4) the channel feature extractor identifies the second portion of channel (AB) and the symbolic feature extractor notifies the blackboard of the fact that it is a dangerous one and that it could be crossed if handled appropriately. Consequently, the blackboard issues the commands to switch from the passageway activity to the channel activity to handle this unexpected situation. However, it soon realizes that there is no room for safe crossing of the channel on the other side because of the presence of object (II). Therefore, the blackboard delays crossing the channel for a later time, and seeks a safe crossing point; it commands the channel activity to go along the channel away from object (II). At point (5), the channel activity is commanded to cross; it traverses the groove perpendicularly. The blackboard decides to switch activities to reach the goal location.

At point (6), the passageway activity reorients the vehicle in the appropriate direction specified by the blackboard. At point (7), the open-space feature extractor detects the obstacle (IV) and the action module drives the vehicle to point (8) where the blackboard modifies the desired direction again. When the vehicle reaches point (9), the channel feature extractor detects channel (CD) and realizes that it is impossible to cross it for its large width and depth. The blackboard is notified of this fact which decides to try to avoid the channel instead of crossing it. Knowing that the original goal is towards point (14), the blackboard commands the channel activity to follow the channel (CD) while keeping a certain safety distance from it. The open-space feature extractor identifies the openings at both sides of point (9), but realizes that the one towards the wall would bring it closer to the desired direction. The blackboard decides to proceed in this way, which drives the vehicle to point (10) where a dead-end is detected. The blackboard, then, decides to try the other opening and commands the channel activity to consider the opposite direction. This drives the vehicle on a path through points (11) and (12). At point (12), where the channel (CD) ends, the passageway activity is commanded to take over while heading towards point (13), and there again towards the final destination, point (14).

5. VALIDATION of the DESIGN

5.1. Overview

It was observed, in chapter 2, that a large number of control architectures for navigation of autonomous vehicles proposed by researchers in robotics were not physically implemented and used for practical analysis. Many of these relied on simulations to test the feasibility of the ideas. Despite the importance of such simulations for the early design phases, it seems that the unavoidable simplifications involved in simulating the real world usually affect the applicability of the results. Therefore, in order to avoid this traditional pitfall, after the theoretical analysis of the Blackboard Activity-Based control architecture in chapter 4, experimental evidence on its feasibility was sought through physical implementation.

In order to assess the feasibility of the Blackboard Activity-Based control architecture for the specific problem of the navigation of an autonomous vehicle, a

prototype was implemented and used for experimental analysis. This experimental system consisted of two basic activities (described in section 4.7): the passageway activity and the channel activity. The choice of those specific activities was based on their importance to the navigation of any mobile vehicle, and particularly in an unstructured industrial environment such as a workshop floor. Although several robot navigation projects concentrated on the collision avoidance aspect of the problem, which is accounted for in the passageway activity, much less research effort was directed to floor sensing and navigation to manoeuvre around such floor obstructions as pipes, cables, trenches, and grooves. These are handled by the channel activity.

5.2. Experimental Setup

The physical implementation of the control architecture, and the experimentation with its components, provides reasonable indications on the practical limitations of the design. It highlights the weaknesses of the architecture and its bottlenecks. The experience gained in the implementation is a source for well-justified decisions during the iterative design phase.

5.2.1. Support Tools

A prototype of the control framework was physically implemented. Two preparatory steps were needed before a full system could be operational. The first step was the development of a communication package between the activities running on the Harmony-based multiprocessor (on-board of the autonomous vehicle) and the blackboard

system running on the Macintosh computer. This communication package was built, on both machines, such as to support general-purpose packetized synchronous and/or asynchronous full duplex serial communication in real time. Error detection is performed on both ends of the communication line. A normal RS-232 cable was used in the original implementation. During the testing phase, this was changed to an RF modem link with a data rate of 9600 baud.

The second step was the modification of the blackboard system, BB_CLIPS, to support real-time operation. The normal "event loop" of BB_CLIPS was modified to perform continuous checks on the serial port linking the Macintosh computer to the vehicle. A special event handler was built to process the incoming serial messages and to perform simple syntax checking. In case of acceptable messages, this event handler parses the message and asserts the corresponding facts on the blackboard which would trigger the appropriate rules. In this manner, the blackboard becomes sensitive to the messages sent from the vehicle during real-time navigation.

5.2.2. The Activities

These two preliminary steps were required because of the unavailability of any such products for real-time systems. Subsequently, the software components (packages) of a typical activity were coded in C language using Harmony message-passing primitives. An instance of such an activity, the channel activity, was built. Some details of the channel activity were not physically built because of the lack of a ready-to-use floor sensor. Therefore, the required floor sensor/feature extractor combination was replaced by a real-time simulator generating continuous information about channels in the experimental

environment. The information generated by this simulated sensor/feature extractor is of the same type that would be expected from the channel feature extractor that would be used with a real sensor. Therefore, this should not affect the validity of the results obtained in these experiments. Additionally, the feature extractor of the channel activity generated the channel information only when the vehicle was in the vicinity of the channel; this ensured that both the action module and the blackboard rely exclusively on the simulated real-time sensory information.

The performance of the feature extractor and the action module of the channel activity was tested independently from the other system components. The vehicle was placed at different locations, and different orientations, in the vicinity of a simulated channel and was commanded manually to cross the channel. The feature extractor of this activity was continuously feeding the action module with numerical information about the channel: the distance to the channel, its orientation, its depth, and its width. The reaction of the action module to this dynamic information was observed and its decision-making mechanism was incrementally enhanced until acceptable performance was achieved. The computations done by the action module included: finding the best angle to cross the channel, calculating reasonable angular and linear velocities (steer and drive), and deciding on an adequate crossing point. When the channel activity was commanded to follow a channel (to go parallel to the channel without crossing it), the action module also computed the required orientation. It should also compute an acceptable distance (to maintain) away from the channel. This last feature, however, was not implemented as of the time of writing this thesis.

The action module of the passageway activity was based on an existing module running an algorithm for collision avoidance. The feature extractor of this activity generated

21 of/de 2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

15800 COUNTY ROAD 3 BURNINGWELL, OK 73007 USA
TEL: 817-428-4887 FAX: 817-428-7887 TUL: 817-428-4888

symbolic information describing the neighbourhood of the vehicle. This feature extractor, together with the action module, were included in the global structure of the Blackboard Activity-Based architecture.

During the experimentation phase, the feature extractors of the two activities were running continuously feeding their information to their respective action modules. Manual commands were used to switch back and forth between the activities in order to study the "transient behavior" of the vehicle, that is its reaction to a sudden change in the environment requiring a change of its focus of attention. It was originally planned that the blackboard assume the responsibility of changing the parameters of the activities in order to ensure smooth transitions between them. Although this was required, it was observed that a faster and more robust technique was also necessary. The reason was that if the control signals of the two consecutive activities did not overlap, the exact state of the vehicle could not be accurately estimated due to the uncontrolled motion. In the case of non-overlapping activities, a "dead-man" time-out would generate a vehicle hardware error between activities causing it to stop until the error is manually cleared. This was accounted for by introducing overlapping periods of constant values of the linear and angular velocities of the vehicle during transition times. This mechanism guarantees seamless transitions, steady behavior, and continuous position control.

5.2.3. The Blackboard

After the two low-level control loops were implemented and tested, both individually and jointly, the blackboard system was incorporated into the navigation system to provide the high-level strategic decision-making. This was accomplished by allowing the

symbolic information coming from the feature extractors of both activities to feed the fact base of BB_CLIPS. These features were continuously sent through the communication package from the vehicle to the Macintosh computer. The event handler of BB_CLIPS asserted the corresponding facts on the blackboard in real time.

The symbolic information consisted of (1) environmental features extracted from the sensory information (or simulated sensory information) by the feature extractors and (2) state information coming from the action modules. They were formulated in a form which is readable by CLIPS rules; i.e., TRUE/FALSE clauses. In the case of the passageway activity, the features were limited to: clear way ahead, object to the right, and object to the left. Those extracted by the channel feature extractor included: threatening channel in the vicinity, approaching a channel, and going parallel to a channel. The states of the passageway activity consisted of: "ENABLED" and "DISABLED." Those of the channel activity were: "ENABLED," "DISABLED," "STARTING," "ORIENTING," "PROCEEDING," "LEAVING," "FINISHING," and "PARALLELING." They indicate the current action carried out by the channel activity with respect to the channel in the vicinity of the vehicle.

The control strategy, responsible for the high-level decision-making, was expressed as a set of BB_CLIPS rules. The rules used in this implementation relied on the facts asserted on the blackboard by the feature extractors and the action modules of the two activities. There were 15 rules having an average of 5 clauses each in their condition part (LHS). Each rule had only one command in its action part (RHS), which consisted of switching to one of the two activities. An example of a BB_CLIPS rule is:

```

(defrule Command_10_1 ""
  (declare (salience 5000))
  (CH_ACT (state PARALLELING))
  (CH_EXT (manvr TRUE) (closer ?clsr) (prll ?p))
  (OS_EXT (clear TRUE) (right TRUE) (left TRUE))
  (Vehicle Initialized)
  ?f1 <- (COMMAND ?cmd&~"10")
  =>
  (retract ?f1)
  (assert (COMMAND "10"))
  (Send_Command "10")
)

```

This rule commands the vehicle to switch to the channel crossing activity (*assert (COMMAND "10")*), if the current command is other than crossing *?f1 <- (COMMAND ?cmd&~"10")*, while going parallel to a channel (*CH_ACT (state PARALLELING)*), and it is safe to cross the channel at the current location (*OS_EXT (clear TRUE) (right TRUE) (left TRUE)*).

5.2.4. Observations

In order to adequately test the performance of this control architecture and to provide continuous operating conditions, the vehicle was to wander around the open space randomly avoiding any obstacles, stationary or moving, and crossing any existing channels safely. Although point-to-point motion would be the actual intended goal of the vehicle in an industrial environment, the "wander mode" provides all the necessary conditions for the purpose of this experimental validation. It also exposes the vehicle to continuously changing surrounding conditions which are a reasonable source for a variety of unexpected events and sudden changes such as moving objects blocking its trajectory randomly and/or obstructing channels with different orientations.

During the incremental design of the blackboard rules, several obstacles were introduced to complicate the blackboard decision-making process and to push it to its limits in order to analyze its failing modes. The vehicle occasionally hit objects or crossed channels from sharp angles. These failures were due to two main reasons: (1) the use of partial knowledge for decision-making (due to incremental development), and (2) the congestion of the serial link between the vehicle and the blackboard (in cases of a large number of simultaneous obstructions).

During experimentation, the vehicle was switching back and forth between the two activities in response to the information dynamically posted on the blackboard. The "default" situation was that of the vehicle moving along a straight line segment until an object or a channel entered its "visibility range" and caused it to change its path; i.e., modified its focus of attention. The feature extractors assumed the responsibility of judging the objects and channels perceived by the sensors; they notified the blackboard about these items only when they constituted a "significant" obstruction. In cases when objects obstructed the vehicle intended path, the passageway action module decided on the corrective action with no intervention of the blackboard. On the other hand, in cases of threatening channels, the blackboard commanded the channel action module to take over the vehicle control, until it reported successful crossing of the channel.

In other cases, where the crossing the channel was dangerous because of objects on the other side of the channel (these were usually people), the blackboard commanded the channel action module to go parallel to the channel until a safe crossing point became reachable. The combination of these three types of commands provided a continuous mode of operation and produced an uninterrupted vehicle motion. The vehicle autonomously

approached (and avoided) the objects from different angles and crossed the simulated channel at different points and from different orientations. The angular and linear velocities of the vehicle were adjusted by the appropriate action module in order to avoid abrupt changes in orientation. This was necessary because of the vehicle's high inertia.

The configuration of the laboratory space used for experimentation was intentionally and unintentionally modified several times with no effect on the system's performance. These modifications consisted mainly of changing the organization of the different movable objects such as desks, tables, boxes, chairs, panels, etc. Similarly, during testing periods, people working in the laboratory were free to move around safely, thus, obstructing the way of the vehicle at different times and different locations.

5.3. Meeting the Original Requirements

The main goal of the experimental work reported here was to validate the Blackboard Activity-Based control architecture and to assess its feasibility for the problem of real-time navigation of sensory-based autonomous vehicle. Some of the behavior specifications, introduced in chapter 3, could not be experimentally verified due to limited number of implemented activities. These are mainly the robustness and reliability requirements. However, experimentation with this architecture showed positive indications about its feasibility. The main characteristics observed during this experimental work are described.

5.3.1. Reactivity

The reactivity of the control architecture was ensured by the conceptual separation of two types of decision-making: strategic and tactical. This was validated when the system switched the *control context* but not the *control level* in response to dynamic and unexpected events. That is, the occurrence of unexpected events incurred smooth switching of the system's focus of attention (control context) while maintaining a continuous tight control (at the same level) on the actuators. The sudden changes in the environment, such as people obstructing the vehicle's path, did not cause the low-level loops (activities) to relinquish control to the high-level loop (the blackboard) and wait for lengthy situation analysis before an appropriate action could be taken. On the other hand, the passageway activity maintained tight control over the vehicle with no intervention from the blackboard. The vehicle's response time (approximately 500 millisecond) was adequate for objects obstructing its path at reasonable speed.

5.3.2. Intelligence

The use of the blackboard as the high-level decision-making mechanism provides an easy way to incorporate progressive levels of situation understanding. This could be achieved by including corresponding rules to describe those situations. In this manner, common-sense could be allowed to guide the reasoning process.

One such example is a rule included in this implementation; it was called the "I got stuck" rule.

```

(defrule Command_00_0 ""
  (declare (salience 5000))
  (CH_EXT (manvr TRUE) (closer ?anything) (prll ?p))
  (CH_ACT (state PARALLELING))
  (OS_EXT (clear FALSE) (right ?r) (left ?l))
  (Vehicle Initialized)
  ?f1 <- (COMMAND ?cmd&~"01")
  =>
  (retract ?f1)
  (assert (COMMAND "01"))
  (assert (! GOT STUCK !))
  (Send_Command "01")
)

```

This rule ensured continuous operation when the vehicle was trapped in impasses or dead-ends. The blackboard realized the problem, ignored the presence of the channel, and continued open-space navigation (and collision avoidance). Although, this would not be the appropriate decision in a workshop floor environment, it was a feasible way to take other actions, since the blackboard "understood" the situation. This reaction was a reasonable validation of the intelligent behavior provided by the blackboard system. Possibly, a more sensible action would be to stop the vehicle temporarily, before backing in the opposite direction to try an alternative path.

BB_CLIPS can provide several ways to add more sensible and intelligent overall behavior with little modification to the blackboard and minor additions to the activities. It provides the programmer with access to rules that affect the behavior of the blackboard control unit. Therefore, there is a possibility to incorporate some situation-specific blackboard manipulation rules to modify the control strategy in response to different configurations of the environment.

This architecture also provides the blackboard with the capability of changing the parameters of the different activities in order to exhibit different behaviors. Although this feature was not used extensively in this implementation, the blackboard was allowed to command the channel activity in one of two ways: crossing the channel or following the channel. A possible addition, would have been to allow the blackboard to restrict the action of the passageway activity to confine the vehicle motion into a certain area; e.g., a room or a hallway.

5.3.3. Multiple Sensors

Three different sensors were used in this experimental system: (1) the vehicle's odometry information, (2) the ultrasound transducers, and (3) the simulated channel sensor. Although the ultrasound sensors fed the passageway activity only, and the simulated channel sensor fed the channel activity only, the odometer was continuously feeding position and orientation information to both activities. This was facilitated by the use of the multitasking multiprocessing Harmony operating system. The asynchronous nature of its message passing mechanism allowed real-time sensory information to be processed by the appropriate servers without any overhead due to sequential processing.

5.3.4. Multiple Goals

The control algorithm of each of the activities was exposed to the continuous information coming from the corresponding feature extractor. Therefore, the action modules had suggestions for the control signals to be sent to the vehicle actuators. The blackboard, however, performed selective activation and deactivation of the activities, thus

handling those multiple goals provided by the different activities. The blackboard was continuously assessing the relevance and the applicability of each activity to the situation in hand. This was particularly visible in situations when the blackboard switched temporarily from the passageway activity to the channel activity in order to cross an obstructing channel. The arbitration between the two goals, following the intended path and crossing the channel safely, was performed by proper sequencing of the two corresponding activities.

5.3.5. Modularity

The simple interconnections between the different components of the activities allowed independent design of the modules before they were integrated to form a complete activity. For instance, the feature extractor of the channel activity was developed and tested independently from the action module of the same activity. Similarly, at a higher granularity level, the activities were developed separately from the blackboard system. Furthermore, the two activities were designed by different people. The modularity of this architecture allowed such components to be independent from each other which facilitated both the design phase and the debugging phase. The passageway activity was previously tested extensively. The channel activity, on the other hand, was incrementally designed and developed for this research. It was tested both individually and through its interaction with the passageway activity. Additionally, due to the modularity of this system, the simulated sensor/feature extractor of the channel activity can be easily replaced by a physical sensor and a corresponding feature extractor. Research is currently underway independently from the development of this experimental setup.

5.3.6. Flexibility

During the implementation phase of this architecture, the components were added incrementally, due to the complexity of the individual components and their interactions. The organization of this architecture proved to be adequate for design changes; that was a result of the flexibility of the control framework. Activities were built and tested individually before they were incorporated into this general system structure. This integration required minor modifications to the existing system and to the new activities. Allowing for such an evolving and iterative design satisfies the flexibility requirement postulated in chapter 3.

The interconnection patterns between the components of this control framework do not impose any restrictions on the information exchanged between the modules. For instance, there are no limitations on the messages coming from the feature extractors. Therefore, the symbolic information generated at the feature extractors, especially that of the passageway activity, were changed several times during the experimentation in order to improve the decision-making capability of the blackboard. This, however, was totally transparent to the system architecture and organization. Having such system flexibility allowed the design decisions to be based on experimental evidence and practical performance measures.

Similarly, during the design of the algorithms used in the action modules, some design decisions had to be made regarding the relative responsibilities of the blackboard and the action module. These, however, were changed several times in order to assess the

performance of different algorithms. The blackboard control mechanism, using rules to command the action modules, provided a flexible way to handle different algorithms with simple modifications to the rules defining the control strategy. Therefore, new control algorithms can be tested and included to enhance the overall performance without affecting the system structure.

5.3.7. Adaptability

During experimentation, the blackboard decision-making mechanism and its control strategy were exposed to various obstacles and channels configurations. The diversity of these cases did not affect the performance of the navigation system which demonstrates its reliability.

The high-level blackboard control over the activities was continuously changing the focus of attention of the system to respond to the changing world configuration. The blackboard chose the activity which could handle the situation best: the channel activity in the vicinity of threatening channels, and the passageway activity for the rest of the time. The blackboard rules included implicit knowledge about the capabilities of the action modules of both activities.

5.3.8. Expandability

As mentioned earlier, the activities used in this implementation were developed independently and added one at a time. The passageway activity was built around a existing module that was previously used for collision avoidance, while the channel activity was

created and tested for the purpose of this architecture. Similarly, the blackboard system was based on an available expert system shell, BB_CLIPS, that was modified separately to fit this architecture. This possibility of incremental design and independent implementation of the different activities adds to the system expandability. Several developers would be able to work separately on skill achieving activities, before these are integrated into the control structure. This feature is required in the design of large systems due to the complexity involved in the design of the individual components. The modifications to the control architecture, in such cases, are limited to the addition of the appropriate rules in the blackboard control unit to take advantage of the added capabilities.

5.4. Discussion

The essence of this control framework is decoupling the two conceptual levels of decision-making: strategic and tactical. This approach proved to be practically useful. Tactical decision-making, which requires very little computational overhead, is required for immediate responses to dynamic events and should be based on automatic reactions rather than on lengthy deductive reasoning. The strategic decisions, on the other hand, need to comply with basic common-sense, and therefore, they should rely on a global understanding of the situation and on a sensible assessment of the plausible alternatives. The passageway activity and the channel activity, both, have individual tactical decision-making capabilities. The blackboard was used to integrate these two capabilities into a single navigation system. This, however, introduced several design challenges.

5.4.1. Granularity Level

While designing the feature extractors, it was required to identify suitable granularity levels for the symbolic information generated and sent to the blackboard. There were many environmental attributes which could be extracted by appropriate processing of the sensory data. Some of these attributes were control oriented; these were mostly numerical parameters describing physical quantities such as depth, distance, orientation, range, height, etc. Although these attributes were essential for the low-level control loops, they were of insignificant use for the high-level reasoning. The use of such information on the blackboard would have complicated its decision-making process causing it to be too situation-specific. Therefore, it was desirable to extract more symbolic features to support facts asserted on the blackboard and to be used by the rules of the inference mechanism.

On the other hand, special care was taken in order not to congest the symbolic feature extractor module with unnecessary processing with the intention to over-simplify the time-consuming task of the blackboard. Otherwise, the blackboard would have delegated most of its high-level decision-making, based on global understanding, to modules in the low-level loops which are vulnerable to incomplete and/or inconsistent knowledge. A suitable compromise for this design tradeoff could only be achieved by experimental analysis. Such a design compromise is application-specific.

An analogous concern was related to the design of the action modules of the low-level loops. In order to achieve high reliability, the performance of the system needs to be independent on the particularities of any specific environment; that is, the behavior of the

vehicle should rely on the sensory information as much as possible for its tactical control. Therefore, we favored simple and robust numerical control algorithms rather than sophisticated versatile ones to manoeuvre the vehicle. A realistic example is that of the action module of the channel activity; a possible algorithm would have been to use a predetermined trajectory, a sinusoid for instance, and to determine its parameters to cross the channel perpendicularly at the desired crossing point. We chose to use a more reliable algorithm which monitored the orientation of the obstructing channel, based on information supplied by the control feature extractor, and continuously maintained a heading normal to that orientation. The second algorithm was computationally less expensive, faster, and more reliable. Unlike the first algorithm, it handles complex and odd-shaped nonlinear channels easily.

5.4.2. Truth Maintenance

Among the issues that need to be addressed in blackboard systems is the problem of truth maintenance. That is, a mechanism is required to continuously monitor the information posted on the blackboard by the different agents in order to ensure that they do not contradict. Without such a mechanism, the decisions taken by the blackboard system would suffer from serious problems. In this implementation of the Blackboard Activity-Based control architecture, the problem of truth maintenance was solved in the event handler incorporated with the blackboard. Although this is not the intended module for this task, it was the fastest way to test the system. In this instance of the architecture, the parser of the event handler retracts any existing facts from the blackboard fact base if they contradict with the incoming up-to-date messages.

However, this task could be accomplished more efficiently by distributing the responsibility on the different agents. In such an implementation, the feature extractors and the action modules would keep track of the last information they posted on the blackboard, and would retract it before sending the new information. This relieves the blackboard event handler from continuously checking the fact base against the incoming messages and reduces the processing on the Macintosh computer running BB_CLIPS.

5.4.3. Failing Modes

During the experimentation with the full implemented system; i.e., with the two activities, the vehicle was exposed to changing conditions. Several times, objects were put dynamically in strategic locations to study and enhance the behavior of the rules forming the blackboard control strategy. Occasionally, the vehicle hit objects or crossed channels from sharp angles. These failures were used to guide the incremental modifications of the blackboard rules. During this phase, two failure modes were repeatedly observed.

The first failure mode was encountered when the rules governing one of the activities were not fully implemented and tested, the vehicle seemed to hesitate before taking the appropriate decision. This was due to the fact that the blackboard relied on incomplete information during those short periods, before the vehicle reached situations where proper decisions could be taken by the blackboard. This failure mode is specific to the development phase, and should not be manifested in normal operation. It should be corrected by appropriate rules on the blackboard.

Another failure mode that was encountered at the later stages of the experimental work was characterized by observable delays between the vehicle's reactions and the changes in the environment that caused them. After several experiments, it was found that these delays were introduced by the transmission of the serial messages and their processing by the blackboard. A first approach to solve this problem could be solved by increasing the data rate on the communication link between the vehicle and the Macintosh computer. More rigorous solutions would involve the handling of the truth maintenance task by the modules communicating with the blackboard, or eventually by implementing the blackboard on the on-board multiprocessor to save the serial communication overhead.

6. CONCLUSIONS

6.1. Research Summary

This thesis addressed the problem of formulating a framework for intelligent sensory-based control of experimental autonomous vehicles. Such robots should be able to interact coherently with their environment, both by adequately interpreting the available sensor data so as to reach appropriate conclusions about the real world for tactical decisions, and by acquiring and manipulating rich and substantially complete knowledge about the environment for strategic decision-making. The control architecture is required to:

- record and handle large quantities of complex real-world data in real-time,
- understand scenes, and
- respond promptly and intelligently to external stimuli.

A literature survey conducted during the first phase of this work depicted the major weaknesses in the current state of the art solutions to the control problem addressed here. It was observed that little attention was paid to the effects of the control architecture on the actual performance of the autonomous system. These deficiencies constitute severe impediments for efficient integration of the different capabilities required in a real-time system intended to operate in a realistic and unpredictable environment.

Based on the analysis of the previously developed control architectures, and their relative merits and shortcomings, a set of requirements to be met by a successful control framework was identified. These requirements were classified into two groups, namely, behavior specifications and design specifications. The behavior specifications are:

- Reactivity to unexpected events,
- Intelligence and common-sense,
- Multiple sensors,
- Multiple, and possibly conflicting, concurrent goals,
- Robustness against imperfect information, uncertainties, and sudden failure, and
- Reliability regardless of the environment-specific information.

The design requirements, which are dictated by the above mentioned behavior specifications, are concerned with the conceptual design of any particular sensory-based intelligent control system. They are necessary to select the most appropriate components for any particular implementation. These can be summarized as follows:

- Modularity,
- Flexibility,
- Expandability,
- Adaptability, and
- Centralized high-level global understanding and decision-making.

A control architecture was developed and iteratively modified and enhanced to satisfy the aforementioned specifications.

In this architecture, the control problem is divided into independent activities that the system is required to have in order to perform intelligently in a real-world environment. An activity is an organizational unit capable of performing a complete task; i.e., exhibiting a certain basic skill. Examples of activities are: follow a person, cross a doorway, cross a channel. Each activity consists of tightly coupled perception and action modules which perform closed-loop control to accomplish their intended activity. The action modules are capable of making the appropriate low-level control decisions to perform the activity successfully.

The perception modules, called feature extractors, are continuously monitoring the environment to extract information about relevant features. They merge numeric sensory data into symbolic information before it is supplied to a reasoning agent, a blackboard in this case. The blackboard combines the high-level information with a suitable control strategy, expressed as a set of production rules, to generate strategic commands to guide the different activities towards the completion of the overall goal. The role of the blackboard in this architecture is similar to that of the operator in a telerobotic control system. It manipulates high-level symbolic information for decision-making in the same

way the operator makes his decisions based on a global understanding of the situation. The activities, on the other hand, have the required autonomy to make tactical decisions to accomplish the commanded task. The blackboard can change the focus of attention of the overall system to handle dynamic situations by selectively enabling/disabling the appropriate activities. The transitions between successive modes of operation can be made seamless by appropriate manipulation of the parameters of the successive activities. This would ensure smooth switching of the control context. By stringing together sequences of activities, the system can be made to execute scripts to perform high-level tasks.

Although it was not originally intended to mimic human behavior, which is not completely well-understood, the behavior of this control architecture is similar to human performance in dynamic situations. The activities represent those tasks that humans perform automatically with very little attention, such as walking in a corridor or driving in the same lane. These tasks are carried out successfully with little energy until they are modified, or eventually interrupted, by higher level decisions in response to external stimuli; e.g., avoiding a person coming to the corridor from an open door, or changing lanes to overtake a slower car ahead. It is necessary to decouple these two levels of intelligence (tactical and strategic), in order to ensure both reactivity to dynamic environments and sensible overall behavior. This similarity to human behavior increases the confidence in the feasibility and the validity of this approach.

This control framework provides viable answers to the problem addressed in this research; i.e., it satisfies the expected behavior requirements. Furthermore, this system structure should be applicable to any sensory-based experimental control system requiring both high adaptability and reliability, because it is independent of the design elements used for any particular implementation.

6.2. Implementation and Experiments

In order to demonstrate the practicality and applicability of this control framework to sensory-based robotics in general, and autonomous mobile vehicles in particular, a prototype was implemented and used for testing. The selected problem was the integration of two basic skills required for autonomous navigation of a mobile vehicle in a workshop environment: (1) the passageway navigation in open-space, and (2) the channel crossing capability. The prototype was implemented on two computing bases: a multiprocessor running a real-time multitasking operating system, and a Macintosh computer running a real-time blackboard-based expert system shell. A real-time serial communication package was implemented to transfer high-level information from the activities to the blackboard, and vice versa.

The passageway activity implementation is based on previous work on collision avoidance which was reorganized to fit the control framework. The channel activity was fully implemented except for the floor sensing capability. The sensor was simulated by a real-time task. Further research is being done in the Autonomous Systems Laboratory on floor sensing.

During the implementation phase, several issues have been raised; they constitute a set of implementation challenges that need to be considered for each application. There is no definite answer for these questions. They depend on the stage of the design and on the particularities of the problem at hand.

The first challenge encountered was to decide on the granularity level to adopt in the perception and action modules. That is to say, the responsibility of the modules of the low-level control loops can be minimized at the expense of increased complexity in the reasoning agent. It is desirable to include sufficient intelligence in the lower modules in order to allow the reasoning agent to delegate the tactical operations. This frees the centralized decision-making module to the more intricate task of strategic control. The decisional capability of the low-level modules should not be made too complex in order to guarantee tight sensory-based closed-loop control, which, in turn, ensures robustness. Otherwise, the reasoning agent would have inaccurate instantaneous information about the decisions of the lower loops, and, consequently, weak immediate effects on their actions.

Therefore, there is a tradeoff between the robustness of the individual activities and the simplicity of the centralized reasoning agent. This is a time-dependent problem. Since the overall system is incrementally built, it often becomes necessary to modify the relative dependence of the reasoning module on the actually implemented activities. For instance, in this experimental setup, the blackboard relies entirely on the passageway activity and the channel activity. At a later stage, when more activities become available, the system dependence on those particular activities would decrease to be complemented by the added capabilities. The adaptability of this control framework is beneficial in such situations, since there is no need to modify the overall structure to incorporate those changes.

Another design issue is the selection of the relevant features to extract from the environment in order to support the high-level reasoning. Another intricate problem is the extraction of those features and their attributes from the available sensory information. Similarly, the Blackboard Activity-Based Control Architecture offers a simple solution for

this problem. Simulated sensors and experimental perception modules are built and used to investigate the relevant symbolic attributes. The design is reiterated and enhanced several times, and the final specifications determine the required sensors which can provide those features. This situation reveals the practicality of this control framework for flexible experimental work.

6.3. Contributions

The flexibility, adaptability, and expandability of this open architecture make it a valuable design tool for sensory-based intelligent control of autonomous systems. It efficiently combines the reactivity of behavioral systems, the perception of hierarchical architectures, and the intelligence of blackboard mechanisms.

Unlike behavioral and hierarchical system structures, the control strategy used in this architecture is confined in the rules of the production system in the reasoning module. It is, therefore, completely decoupled from the system architecture. This allows different strategies to be investigated and compared in order to evaluate their relative performance without any changes to the interconnection pattern.

Finally, this control framework should be applicable to systems requiring real-time handling of large amounts of sensory information and which are intended to respond to unpredictable stimuli. Such systems are the critical components of applications, not only in industrial environments, such as robotics and inspection, but also in medical imaging, environment monitoring, and systems to aid the handicapped.

6.4. Future Work

From the experience gained in developing this control architecture and the specific activities, it seems that it would be useful to allow the high-level reasoning module (the blackboard) to slow down the vehicle's motion if it realizes that it can not provide the required strategic decisions in a timely fashion. Although the vehicle would exhibit a sensible behavior in such situations, due to the performance of the enabled activity, this would prevent the temporary accumulation of contradictory information on the blackboard, and would save the computational overhead of sorting these contradictions.

Some situations may require the control architecture to rely on information coming from one sensor, and the corresponding feature extractor, rather than another. In such cases, the low-demand sensor could be treated differently with modified parameters to provide more relevant information. For instance, the passageway activity relies strongly on the ultrasonic range transducers. For this purpose, the ultrasonic sensors have a short useful "visibility" range to ensure high accuracy. When the channel activity is enabled, however, the sonar information only helps detecting objects. Therefore, during those periods, it would be desirable to increase the range of these transducers, at the expense of lower accuracy, in order to improve the detection capability of the overall system and to increase its "visibility". This would possibly require complex operations for mode switching at the sensor level, but may be a feasible and inexpensive mechanism to enhance the perception of the system.

In order to reduce the communication delays in the symbolic/numeric information interface, the blackboard may be reconfigured and coded as a Harmony real-time task. This would result in a significant increase in the overall reactivity. Another intermediate step towards the same goal would be the use of a faster communication link; e.g., an Ethernet communication.

Due to the limited time frame of this research, some issues were not addressed deliberately. Although these were considered beyond the scope of this work, they are closely related.

The work presented here dealt mostly with the development and the definition of the control framework rather than with the physical implementation of a full control system. Therefore, it was always assumed that the global goals to be achieved by the autonomous vehicle were stored on the blackboard. However, in the general case of an integrated intelligent control system, the user specification of the goal should be at the highest possible level of abstraction, e.g., "deliver this mail to room 317" or "empty this bin in the waste container." For such a level of human-machine interaction a "User Specification Language" should be developed and continuously enhanced and enriched. The concepts of "Natural Language Understanding" techniques may be a promising approach for this purpose.

A closely related question is that of the "Task Decomposition Heuristics" used to express the user-defined specifications in the form of alternative sequences of intermediate goals to be performed by the control architecture. These are the goals that are handled by the reasoning module, namely the blackboard. The heuristics used for this decomposition

are application-dependent. Task-level planning techniques and control scripts may facilitate the solution of this problem. The use of this control framework eases the experimentation with several such heuristics and allows for case-based learning to refine those heuristics.

The robustness and the reliability of such architecture can only be tested with long experiments involving a large number of activities. This could not be achieved, however, in the context of this work. Therefore, it is necessary to pursue further research along those lines to investigate the effects of increased system complexity on the overall performance, and to study its implications on the selection of the design elements.

The experiments carried out in this research showed that the performance of the implemented navigational system can be further improved by the addition of more sophisticated sensors. A long range object detection sensor is a good example. The BIRIS⁷ laser range sensor [9] can be used for this purpose in order to provide higher accuracy, shorter delays, and longer ranges. Some effort has been put at the Autonomous Systems Laboratory of the National research Council in the extraction of geometric features from BIRIS data [27]. Geometric feature extraction can be further enhanced using a set of efficient packages using genetic algorithms [38] also developed in the Autonomous Systems Laboratory. This work, if integrated in the Blackboard Activity-Based Control Architecture, could be a valuable contribution to its perception capability.

⁷BIRIS is a very compact optical range sensor developed in the Institute for Information Technology of the National Research Council.

References

- [1] J. S. Albus, A. J. Barbera, and R. N. Nagel, "Theory and Practice of Hierarchical Control," *Twenty Third IEEE Computer Society International Conference*, pp. 18-39, Washington, D.C., September 1981.
- [2] J. S. Albus, R. Lumia, and H. McCain, "Hierarchical Control of Intelligent Machines Applied to Space Station Telerobot," *Proceedings of the IEEE Symposium on Intelligent Control*, pp. 20-27, Philadelphia, Pennsylvania, January 1987.
- [3] J. S. Albus, C. R. McLean, A. J. Barbera, and M. L. Fitzgerald, "Hierarchical Control for Robots and Telerobots," *Proceedings of the IEEE International Workshop on Intelligent Control*, pp. 39-49, Troy, New York, August 1985.
- [4] R. C. Arkin, "Motor Schema-Based Mobile Robot Navigation," *International Journal of Robotics Research*, Vol. 8, No. 4, pp. 92-112, 1987.
- [5] R. C. Arkin, "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior," *Proceedings of the International Conference on Robotics and Automation*, pp. 264-271, April 1987.
- [6] R. C. Arkin, "Dynamic Replanning for a Mobile Robot Based on Internal Sensing," *Proceedings of the International Conference on Robotics and Automation*, pp. 1416-1421, Scottsdale, Arizona, May 1989.
- [7] E. Badreddin, "Recursive Nested Behavior Control Structure for Mobile Robots," *Proceedings of the International Conference on Intelligent Autonomous Systems 2*, pp. 586-596, Amsterdam, December 1989.
- [8] A. J. Barbera, J. S. Albus, M. L. Fitzgerald, "Hierarchical Control of Robots using Microcomputers," *Ninth International Symposium on Industrial Robotics*, pp. 405-422, March 1979.
- [9] F. Blais, M. Rioux, and J. Domey, "Optical Range Image Acquisition for the Navigation of a Mobile Robot," *Proceedings of the International Conference on Robotics and Automation*, pp. 2574-2580, Sacramento, California, April 1991.

- [10] D. L. Brock, and J. K. Salisbury, "Implementation of a Behavioral Control on a Robot Hand/Arm System," *International Symposium on Experimental Robotics*, Toulouse, France, June 1991.
- [11] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23, March 1986.
- [12] R. A. Brooks, "A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network," *Proceedings of the International Conference on Robotics and Automation*, pp. 692-696, Scottsdale, Arizona, May 1989.
- [13] R. A. Brooks, "Intelligence without Representation," *Artificial Intelligence*, vol. 47, pp. 139-159, 1991.
- [14] R. Chatila, A. Alami, B. Degallaix, V. Pérébaskine, P. Gaborit, and P. Moutarlier, "An Architecture for Task Refinement and Execution: Control for Intervention Robots: Preliminary Experiments," *International Symposium on Experimental Robotics*, Toulouse, France, June 1991.
- [15] The CLIPS v4.2 Reference Manual, Artificial Intelligence Section, Lyndon B. Johnson Space Center, April 1988.
- [16] J. H. Connell, "Creature Design with the Subsumption Architecture," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1124-1126, Milan, Italy, August 1987.
- [17] J. L. Crowley, P. Reignier, O. Causse, and F. Wallner, "Action Level Behaviors for Locomotion and Perception," *International Symposium on Experimental Robotics*, Toulouse, France, June 1991.
- [18] W. M. Gentleman, "Message Passing between Sequential Processes: The Reply Primitive and the Administrator Concept," *Software Practice and Experience*, Vol. 11, No. 5, pp. 435-466, May 1981.
- [19] W. M. Gentleman, C. Archibald, S. Elgazzar, D. Green, and R. Liscano, "Case studies of realtime multiprocessors in robotics," *Proceedings of the Second International Specialist Seminar on the Design and Applications of Parallel Digital Processors*, Lisbon, Portugal, April 1991, NRC 31813.
- [20] D. Green and R. Liscano, "Real-Time Control of an Autonomous Mobile Platform using the Harmony Operating System," *Proceedings IEEE International Symposium on Intelligent Control*, pp. 374-378, Albany, New York, September 1989, NRC 30554.
- [21] S. Y. Harmon, "Practical Implementation of Autonomous Systems: Problems and Solutions," *Proceedings of the International Conference on Intelligent Autonomous Systems*, pp. 47-59, Amsterdam, Netherlands, December 1986.

- [22] S. Y. Harmon, "Current Design Limitations of Computing Architectures for Complex Robot Systems," *IEEE International Symposium on Intelligent Control*, pp. 116-121, Philadelphia, Pennsylvania, January 1987.
- [23] S. Y. Harmon, B. E. Bianchini, and B. E. Pinz, "Sensor Data Fusion through a Distributed Blackboard," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1449-1454, San Francisco, California, April 1986.
- [24] R. Hartley, and F. Pipitone, "Experiments with the Subsumption Architecture," *Proceedings of the International Conference on Robotics and Automation*, pp. 1652-1658, Sacramento, California, April 1991.
- [25] B. Hayes-Roth, and M. Hewett, "BB1: An Implementation of the Blackboard Control Architecture," in R. Englemore, and T. Morgan (Eds.), *Blackboard Systems*, Addison-Wesley, 1988.
- [26] K2A Mobile Platform, Cybermotion, Roanoke, VA, 1987.
- [27] S.T. Lang, A Structural Approach to Map Making for an Autonomous Mobile Robot, Ph.D. thesis, Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 1991.
- [28] S. T. Lang, L. W. Korba, and A. K. C. Wong, "Characterizing and Modelling a Sonar Ring," *Proceedings of SPIE Advances in Intelligent Robotics Systems: Mobile Robots IV*, pp. 291-304, Philadelphia, Pennsylvania, November 1989 NRC 30928.
- [29] R. Liscano, R. Fayek, and G. M. Karam, "A Blackboard, Activity-Based Control Architecture for a Mobile Platform," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 333-338, Raleigh, NC, July 1992.
- [30] R. Liscano, and D. Green, "REDY, An Architecture for a Sensor-Based Mobile Platform," *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Ottawa, Canada, September 1990, NRC 31746.
- [31] R. C. Luo, and M. G. Kay, "Multisensor Integration and Fusion in Intelligent Systems," *IEEE transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5, pp. 901-931, September/October 1989.
- [32] A. Manz, R. Liscano, and D. Green, "A Comparison of Realtime Obstacle Avoidance Methods for Mobile Robots," *International Symposium on Experimental Robotics*, Toulouse, France, June 1991, NRC 31826.
- [33] D. Nitzan, "Development of Intelligent Robots: Achievements and Issues," *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, pp. 3-13, March 1985.
- [34] F. R. Noreils, and R. G. Chatila, "Control of Mobile Robot Action," *Proceedings of the International Conference on Robotics and Automation*, pp. 701-707, Scottsdale, Arizona, May 1989.

- [35] G. K. H. Pang, "A Framework for Intelligent Control," *Journal of Intelligent and Robotic Systems*, Vol. 4, pp. 109-127, 1991.
- [36] R. P. Paul, H. F. Durrant-Whyte, and M. Mintz, "A Robust, Distributed Sensor and Actuation Robot Control System," *Third International Symposium on Robotics Research*, pp. 93-100, Gouvieux, France, October 1991.
- [37] D. W. Payton, "An Architecture for Reflexive Autonomous Vehicle Control," *Proceedings of the International Conference on Robotics and Automation*, pp. 1838-1845, San Francisco, April 1986.
- [38] G. Roth, and M. D. Levine, "Random Sampling for Primitive Extraction," *International Workshop on Robust Computer Vision*, Seattle, Washington, October 1990.
- [39] K. Sadamoto, *Robots in the Japanese Economy*, Tokyo: Survey Japan, 1981.
- [40] K. G. Shin, and S. B. Malin, "A Hierarchical System Architecture for Coordinated Control of Industrial Manipulators," *IEEE International Conference on Robotics*, pp. 609-619, 1984.
- [41] Ultrasonic Ranging System, POLAROID Corporation, Ultrasonic Components Group, Cambridge, MA.
- [42] B. Yamauchi, and R. Nelson, "A Behavior-Based Architecture for Robots Using Real-Time Vision," *Proceedings of the International Conference on Robotics and Automation*, pp. 1822-1827, Sacramento, California, April 1991.
- [43] R. Zanconato, "BLOBS - An Object-Oriented Blackboard System Framework for Reasoning in Time," in R. Englemore and T. Morgan, Eds., *Blackboard Systems*, Addison-Wesley, 1988.

END

24 05 93

FIN