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.

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.

AUTONOMOUS UNDERWATER VEHICLE CONTROL THROUGH SITUATION IDENTIFICATION

by

Darrin K. Wolter

B.A.Sc. (Systems Design) University of Waterloo, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
November 10, 1993

ISBN 0-612-01023-6

Canada

APPROVAL

Name: Darrin Wolter

Degree: Master of Applied Science

Title of thesis : Autonomous Underwater Vehicle Control Through Situation Identification

Examining Committee: Dr. Jacques Vaisey
Assistant Professor, Engineering Science, Chairman

Dr. John S. Bird
Associate Professor, Engineering Science
Senior Supervisor

Dr. Shahram Payandeh
Assistant Professor, Engineering Science
Senior Supervisor

Dr. John D. Jones
Associate Professor, Engineering Science
Supervisor

Dr. William A. Gruver
Professor, Engineering Science
Internal Examiner

Date Approved: November 17, 1993

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Autonomous Underwater Vehicle Control through Situation Identification"

Author:

(signature)

Darrin Wolter
(name)

November 10, 1993
(date)

Abstract

This thesis sets the stage for development of an autonomous-vehicle control theory. Control of autonomous vehicles is not an artificial intelligence dilemma, but is a control problem where the control environment is often under-sensed. In this thesis we hypothesize that machines are only capable of applying a law given to them by their designers. As the possessors of a law, machines do not understand the environment in which they exist, but simply react to that environment as they were built, or programmed to react. We present the new ideas of sensor and actuator space as a mathematical framework for describing under-sensed control problems in terms of environmental situations that sensors are able to differentiate. We also show that computer-based control systems have a lookup-table equivalent, referred to as a Q-SAM. Each sensor-differentiable situation is associated with a single location in the Q-SAM.

We use Q-SAMs to explore the potential of situation-based control and show that they can accept many different forms of control laws. When control environments are under-sensed, we have found that the definitions associated with differentiable environmental situations are a function of the entire vehicle as well as the environment in which the vehicle exists, which is not the case for critically sensed control environments.

We describe a design methodology for autonomous systems that results in systems which are robust to disturbances in their environments. We use the methodology to determine the sonar parameters required by an autonomous underwater vehicle for the task of obstacle avoidance in an unknown obstacle field. Limitations of the vehicle are discussed.

for Jennifer

Acknowledgements

There are two people to whom I owe much of the credit for this thesis: John, my supervisor, for supporting my struggle through the black periods, for it is through that struggle that we learn what we truly believe, and Jennifer, my wife, for without her support this thesis would never have been. I would also like to thank my other supervisor, Shahram, for his contributions, and everybody at the Underwater Research Lab, in particular, Bill, with whom I could always discuss my ideas, and Paul, who listens.

Contents

List of Tables

List of Figures

Glossary

actuator mapping the function that describes how the actuators affect the environment as well as the natural changes of the environment

actuator space the n-dimensional space that represents all possible actuator responses

adequacy refers to a controller that is able to complete its task regardless of initial conditions, but not necessarily in the most efficient manner

AI Artificial Intelligence

Allen is an autonomous vehicle developed by the MIT mobot lab that uses only behaviour based control techniques

ALV Autonomous Land Vehicle, an AV used by the Hughes AI centre to demonstrate a hybrid control architecture

ARDF Actuator Response Distribution Function. This function is used to increase the rate at which a Q-SAM is filled.

AV Autonomous Vehicle

AUV Autonomous Underwater Vehicle

behaviour the manifestation of activity in the environment that results from applying specific responses to specific inputs. The inputs are generally not associated with internal representations.

behaviour trapping a limit-cycle type of effect that occurs when control oscillates among multiple independent behaviours

brittle a term used to describe systems that are not robust to changes in their environments

complete used with reference to sensor and actuator space. Sensor space is complete when all environmental situations requiring differentiation by the control mapping are associated with different regions of sensor space. Actuator space is complete when all actions of the actuators required by the control mapping are associated with independent locations in actuator space.

consistent refers to partitions. Partition A is consistent with partition B when all regions in the space differentiated by partition B are also differentiated by partition A

control mapping the function that transforms sensor inputs into actuator responses

critically sensed refers to control environments that are described with only recently sensed sensor values. For example, the environment of a position control system is critically sensed by a position sensor.

disturbance a change in the environment for which a system was not specifically designed

downloading a method of filling a Q-SAM by copying the responses determined by another controller

environment space the portion of the control cycle that represents the physical world. Environment space includes computer memory used for internal representations.

goal region the region of sensor space associated with a completed task.

global adaptation adapting the control surface by changing the actuator responses associated with many or all locations in sensor space.

hybrid architecture a control system that uses both behaviour-based and internal representation-based control techniques

internal representation a representation in computer memory that is used to generate control. For AVs, a common internal representation is a world map.

local adaptation adaptation of the control surface by changing the responses associated with one, or very few, differentiable situations

partition a division of sensor or actuator space by function. For a complete space, the quantization partition must be consistent with all other partitions

Q-SAM Quantized Sensor to Actuator Map. Q-SAMs are the lookup table equivalent of computer programs.

recently sensed sensor values The most recent values returned by sensors. These are not values returned from internal representations.

robust refers to a control system that responds well to disturbances in its environment

ROV Remotely Operated Vehicle

sensor mapping the function that transforms environmental situations into locations in sensor space

sensor space the space that represents all situations the sensors can sense

SHAKEY an AV developed at Stanford Research Institute which relies heavily on internal representation-based control techniques

situation a specific scenario in the environment

situation differentiation refers to which environmental situations are differentiated by the sensor transformation

situation identification refers to specification of the meaning of each location in sensor space

STRIPS STanford Research Institute Planning System

sufficient with reference to sensors that are able to differentiate all situations requiring differentiation by the control mapping.

under-sensed refers to control environments that are not described with only recently sensed sensor information. For example, a single sonar beam cannot sense all the obstacles in an environment, so the obstacle control environment is under-sensed.

world map an internal representation commonly used by AVs

Nomenclature

α width of sonar beam pattern

∞ infinity, meaning no objects are presently sensed by a sonar sensor

∞' not infinity, meaning an object is sensed by a sonar sensor

θ_v angular orientation of AUV with respect to the world-based coordinate system ($\theta_v = 0°$ means the vehicle is pointing along the world's positive y-axis)

$-$ means that the refenced item is irrelevant

$\bar{a}$ actuator state vector, represents present location in actuator space

$\bar{a}_n$ actuator state vector at time index n, represents present location in actuator space

$\bar{a}_r$ portion of the actuator state vector associated with real actuators

$\bar{a}_{wm}$ portion of the actuator state vector associated with internal representations

b parameter used to determine the width and length of sonar beam patterns

c minimum desired clearance between the vehicle and obstacles

d distance the sonars are placed in front of the centre of the vehicle

d damping coefficient of the AUV in the altitude-keeping example

d_l distance to the closest object detected in the left sonar beam pattern

d_r distance to the closest object detected in the right sonar beam pattern

$\bar{e}$ environment state vector, represents present location in environment space

$\bar{e}_n$ environment state vector at time index n, represents present location in environment space

$\bar{e}_r$ portion of the environment state vector associated with the real environment

$\bar{e}_{wm}$ portion of the environment state vector associated with internal representations

F_n force applied at time index n in the AUV altitude-keeping example

g sensor mapping, represents the transformation of the environment into sensor space

g_r portion of the sensor mapping that represents the effects of real sensors

g_{wm} portion of the sensor mapping that represents reading internal representations

h actuator mapping, represents the effects of the actuators on the environment and the continual changes of the environment

h_e actuator mapping, represents the continual changes of the environment

h_r portion of the actuator mapping that represents the effects of the real actuators on the real envrionment and the continual changes of the real environment

h_v actuator mapping, represents how the actuators effect the environment

h_{wm} portion of the actuator mapping that represents storing internal representations

h' a composite function that shows that a vehicle in the environment is a modelable part of that environment

l length of a sonar beam pattern

M mass of the AUV in altitude-keeping example

m control mapping, specifies which actuator responses are associated with which situations differentiated in sensor space

m_r portion of the control mapping that assigns actuator responses to real actuators (the responses can be determined from both internal representations and real sensors)

m_{wm} portion of the control mapping that assigns actuator responses to internal representations (the responses can be determined from both internal representations and real sensors)

Mode mode command sent to the low-level control algorithm

P_A actuator partion of sensor space (determined by the actuator responses assigned to different situations under control law m)

P_G goal partition of sensor space (defines which regions of sensor space are associated with a complete task and which are not)

P_Q quantization partition of sensor space (caused by quantization of sensor values)

r minimum turning radius of the AUV

$\bar{s}$ sensor state vector, represents present location in sensor space

$\bar{s}_n$ sensor state space at time index n, represents present location in sensor space

$\bar{s}_r$ portion of the sensor state vector associated with real sensors

$\bar{s}_{wm}$ portion of the sensor state vector associated with internal representations

t time

t_n indexed time

x x-position of the next vehicle waypoint sent to the low-level control algorithm

x_g x-position of the vehicle's endpoint (goal) location

x_o x-position of an object endpoint

x_v x-position of the vehicle in the world coordinate system

x_w x-position of the vehicle's next waypoint

y y-position of the next vehicle waypoint sent to the low-level control algorithm

y position of the AUV in the altitude-keeping example

$\dot{y}$ velocity of the AUV in the altitude-keeping example

y_g y-position of the vehicle's endpoint (goal) location

y_o y-position of an object endpoint

y_v y-postition of the vehicle in the world coordinate system

y_w y-position of the vehicle's next waypoint

Chapter 1

Introduction

1.1 Motivation

Advances in computational power over the last few decades have allowed our interest in control to expand beyond controlling simple equipment, like motors and conveyor belts, which operate in completely structured environments, to attempting to control autonomous vehicles that operate in the real world, which is an unstructured environment. Biological creatures have operated autonomously in the real world for many millennia, and of them, it is thought that humans are the most able to conceptualize and abstract about that real world. The human abilities of conceptualization, abstraction and reasoning, coupled with an ability to react to immediate situations, allow people to perform complex tasks in an ever-changing world. With the computational power of computers pressing towards that of the human mind, researchers are attempting to build computer-controlled machines to perform tasks that to date only humans have been able to perform (Wallich 1991).

Yet, with all this computational power, researchers are still unable to endow a machine with the cognitive abilities of a small child. The reason for this inability is that there is more to humans than the complex collection of biological components of which humans physically consist, and computers are only the sum of the electrical components of which they are comprised. This "more" we refer to may be unique to humans and has been labeled as gestalt and fringe consciousness (Dreyfus 1992). Unfortunately, there is no proof of a human synergistic existence short of the fact that all attempts at artificial intelligence, and more specifically creating artificial creatures,

have been very disappointing (Wallich 1991).

An illustration of this unique facet of humanity, relative to lack of it displayed by computers, is how people learn to play chess (Dreyfus 1992). When a person first learns to play chess, they learn the rules for moving each piece on the board in rote fashion. After several games however, the pieces and the chess board take on new meanings for the player, who no longer considers every possible move all the time. Instead s/he only considers moves that are significant. As more experience is gained, the player rises above the rules and begins to look at the game in terms of flows and directions of attack. Rising above the rules in this manner is conceptualization and abstraction. Computers on the other hand, are not able to rise above the rules. They are capable of only applying the rules which they have been given in rote fashion. The success of computer chess programs lies in the fact that computers are able to calculate the outcome of several thousand moves every second, something that people cannot do.

The fact of the matter is that computers are able to perform tasks in only rote fashion. The "intelligence" they possess is in the form of rules, which were given to them by their designer, that they apply to different environmental stimuli. Computers do not understand the environment in which they exist but simply react to the environment as they were built to react. More precisely, computers are simply the possessors of a law that is handed down to them from a person that understands the environment in which the computer exists, and the intelligence resides in the person that determined the rules.

Computers used to control equipment in the real world have an added constraint of requiring sensors capable of differentiating environmental stimuli to which the computer provides responses. Computers cannot respond to environmental stimuli which they cannot sense. Also, computers can respond to environmental stimuli with only responses associated with the actuators available to the computer. Computers cannot provide responses that their actuators cannot effect. From an overall perspective, computers are limited to supplying responses associated with their actuators to environmental stimuli that their sensors are able to differentiate. To the author's knowledge, this thesis is a first look at viewing autonomous vehicle control as a problem in differentiating environmental stimuli and providing appropriate actuator responses to those stimuli once they have been differentiated.

1.2 Principles

At the outset of this thesis, we feel that it is important to clearly define the opinions of the author. It is our opinion that machines, which includes computers and autonomous vehicles, are fundamentally different than human beings. This contrasts with the opinions of many artificial intelligence researchers who do not observe this distinction between man and machine and who believe that if a person can do something, so can a machine. Consequently, we do not believe that machines, including computers, are intelligent or can be programmed to be intelligent. It is our opinion that machines simply implement a law which was designed for them by their designers and that anything that can only implement a law is not intelligent and does not understand what it is doing.

With our opinions, it is reasonable to expect autonomous vehicle designers to extend the ideas of control theory for the analysis and exploration of the capabilities and limitations of autonomous vehicles because they are, like the equipment that control theory was developed to analyse, machines. However, it is our opinion that this has not been the case. Instead of continuing to develop control theoretic approaches, many autonomous vehicle researchers have chosen to base their designs on the concepts of artificial intelligence. We illustrate this change in development strategy in figure 1.1. It is our opinion that one reason for this change is that many of the sensors used on autonomous vehicles, and many of the environments in which autonomous vehicles operate, are difficult, if not impossible, to describe in the mathematical framework of feedback control theory (Van de Vegte 1986). A second, and more persuasive argument, is that many of the sensors used by autonomous vehicles have similar scopes, and limitations, as human sensors. For example, cameras and human eyes provide very similar information about the environment. Consequently, the ideas of artificial intelligence are very appealing to the designers of autonomous vehicles. However, it is our opinion that the use of artificial intelligence techniques has unreasonably raised expectations of autonomous vehicles and has exposed many of the limitations of machines that are not apparent in humans. In this thesis, we move back to a control-theoretic perspective of autonomous vehicle development, as shown in figure 1.1.

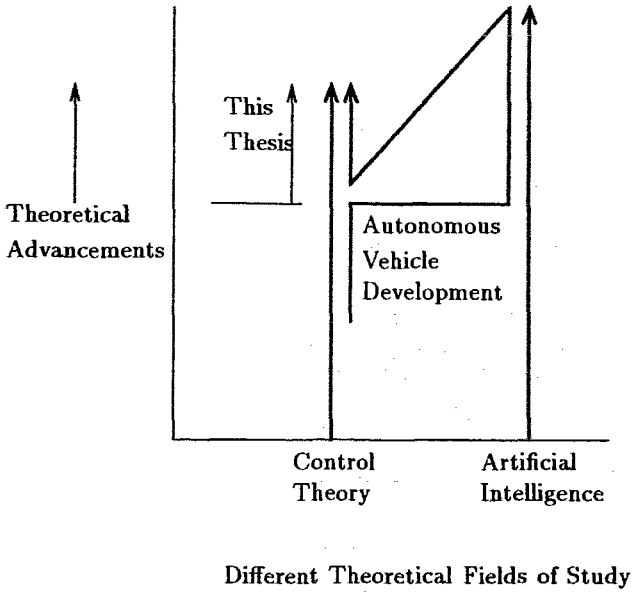


Figure 1.1: Autonomous vehicle development strategy

1.3 Contributions

The major contribution we make in this thesis is presenting a different set of initial assumptions on which to base development of autonomous vehicles. We assume that computers, and computer-controlled equipment, are not intelligent and do not understand the environment in which they exist. In accordance with this assumption, we show that computers, and computer controlled equipment, simply implement a law that is determined for them by their designers. This task is accomplished using the concept of Quantized Sensor to Actuator Maps, or Q-SAMs, which are lookup-table equivalents of computer-based control algorithms.

Another contribution we make in this thesis is introducing the concept of under-sensed and critically sensed control environments. Under-sensed control environments are ones in which sensors do not sense the entire control environment simultaneously, and critically sensed control environments are ones in which sensors do sense the entire control environment simultaneously. We show that autonomous vehicle development is a control problem where the control environment is often under-sensed. In doing so, we set the foundation of a control theory for systems operating in under-sensed control environments.

The foundation of this control theory consists of several concepts introduced in

this thesis. These concepts are sensor space, actuator space, situations, situation differentiation and situation identification, which are all related. Sensor space is the conceptual interface between sensors and control laws and actuator space is the conceptual interface between control laws and actuators. By "conceptual" we mean that there are no physical components associated with these spaces, not that they are abstract ideas. Situations are specific environmental scenarios. Sensors transform each situation into one location in sensor space. Situation differentiation is determining which situations are transformed into different regions of sensor space. Situation identification is assigning a meaning to each differentiable region in sensor space.

We also introduce the concept of Quantized Sensor to Actuator Maps, or Q-SAMs, which are lookup-table equivalents of computer-based control algorithms. Q-SAMs are used to explore the potential of situation-based control.

The final contribution we make in this thesis is showing that situations in the environment that are differentiated in sensor space by sensors operating in under-sensed control environments are a function of the entire autonomous system. This is in contrast with critically sensed control environments whose situations are a function of only the sensor transformation. This concept is the basis of a design methodology that we introduce for developing autonomous vehicles that operate in under-sensed control environments. This methodology develops systems that are robust to disturbances in their environments.

1.4 Thesis Outline

In chapter 1, this chapter, we have provided motivation for our research and stated the contributions we have made in this thesis. We now conclude chapter 1 by providing an outline of the remainder of this thesis.

In chapter 2, we present a brief history of Autonomous Vehicle (AV) development. The reader is introduced to the two basic paradigms of autonomous control: behaviour-based control and world map, or internal representation, -based control. To present a clear picture of the state of the research field, several "key" autonomous vehicles are described and their accomplishments and limitations discussed with reference to situation differentiation.

In chapter 3, we describe the autonomous vehicle control cycle and show that Q-SAMs are functionally equivalent to computer-based control systems. The concepts of sensor space, actuator space, situations, situation differentiation and situation identification are described in this chapter. We also show how both autonomous vehicle control paradigms are represented in the Q-SAM representation and the autonomous vehicle control cycle.

In chapter 4, we explore the potential of situation-based control, a new method of control law description, by using the Q-SAM representation of a control law. Q-SAMs are shown to support many forms of control laws by using them to implement algorithmic as well as expert-based control laws. The potential of situation-based adaptive control is examined.

In chapter 5, we outline a design methodology for developing autonomous vehicles that operate in under-sensed control environments. The methodology specifies the appropriate use of internal representations in autonomous vehicle control systems to ensure that the autonomous vehicles are robust to disturbances in their environments. In this chapter, we also show that the situations differentiated in sensor space are a function of the entire autonomous vehicle when the control environment is under-sensed. Through the course of this chapter, we examine the potential of vehicle based sonar systems for obstacle avoidance.

Finally, in chapter 6, we provide a general discussion of our results, summarize the significant ideas presented in this thesis and indicate areas of future work.

Chapter 2

Background

This chapter provides the reader with a background in autonomous vehicle research and a foundation for the research described in this thesis. To accomplish this task, we first present a general discussion of the philosophy and activity of the autonomous vehicle community. We then describe the fundamental assumption that makes our research different from previous research. To clearly illustrate the research field, we describe the philosophy of, and systems developed by, several key research groups. The progress of the implementation phase of each group is discussed with reference to this thesis. The chapter is concluded by summarizing the present state of autonomous vehicle research.

At this point, it is important to note that there is no general consensus about the meaning of the terms intelligence, learning, understanding and reasoning. It is our opinion that this circumstance exists because there are no direct methods of observing these human characteristics. Instead, these attributes are often determined indirectly by a system's ability to provide appropriate responses to stimuli. Unfortunately, appropriate responses can also be generated by systems that are, in our opinion, not intelligent, which is the case for autonomous vehicles or, as they are sometimes referred to, artificial creatures. For example, we do not consider a light switch intelligent simply because it can turn a light on, even though turning a light on is sometimes an appropriate response. This lack of definitions has left much leeway for use of these terms in the literature. Consequently, the terms intelligence, learning, understanding and reasoning are used in many different contexts in this chapter to properly present the ideas of different research groups. Where it is not obvious, we have noted the

form or definition of intelligence being described.

In section 2.1, we provide a general background of AV research, and describe the two basic paradigms of autonomous vehicle control: behaviour-based control and world map or internal representation-based control. In section 2.2, we describe the fundamental assumption that makes our research different from previous research, and discuss the implications of that assumption. In section 2.3, we outline many limitations of autonomous vehicle control strategies. In section 2.4, we describe the standard autonomous vehicle control architecture. In section 2.5, we describe SHAKEY, an AV that demonstrates many of the ideas of traditional artificial intelligence through its use of internal representations. We discuss the success of the SHAKEY project in terms of situation differentiation. In section 2.6, we describe Allen, a behaviour-based AV. We also describe the subsumption architecture, which is the basis of most behaviour-based control architectures, as well as some of the limitations of behaviour-based control. In section 2.7, we discuss the Autonomous Land Vehicle (ALV), a hybrid system that uses both behaviour-based and world-map-based control paradigms. We also discuss the limitations of this system in terms of situation differentiation. Finally, in section 2.8 we summarize the significant ideas discussed in this chapter and in section 2.9 we discuss some of the directions taken by autonomous vehicle researchers.

2.1 General Background

Much of the work on autonomous vehicles is centred in the artificial intelligence community, even though the work is really a controls problem. By the term "controls problem", we mean that autonomous vehicles operate in a control cycle in which they continually read sensors and perform some calculations on those sensor values to set actuators. The reason for this state of affairs is that autonomous vehicles face similar sensing impediments as human beings: namely sensors that are incapable of sensing the entire environment simultaneously. For example, human eyes cannot see an entire city and yet people are able to plan and navigate a path through a city. It is the author's opinion that a person's ability to accomplish this navigational task relies on some form of mental internal representation of the city (ie. a world map) that is sufficient for navigational purposes. Another reason that much autonomous vehicle work is done in the artificial intelligence community is that many researchers believe

they are creating artificial creatures and not just machines (Brooks 1991). Their fundamental assumption is that the intelligence which humans possess is biologically-based and therefore they should be able to replicate it with a machine. Consequently, many of the designs of autonomous vehicles have their roots in psychology and not in engineering. The work done by the MIT Mobot Lab (Brooks 1990) is based on behaviourist psychology. The work done at the Georgia Institute of Technology (Arkin 1990) is based on the psychological concept of schemas. The work done by Laird and Rosenbloom (1990) is based on an artificial intelligence system called SOAR, whose ideas have been expanded into a unified theory of cognition (Newell 1990).

The ideas of psychology have led to the development of two basic components in autonomous vehicle control systems, namely: situated reactive elements, known as behaviours, and internal representations, generally known as world maps to the autonomous vehicle community. Behaviours are a desired manifestation of activity in the world (Brooks 1986). They are created through a tight coupling between sensing and actuation (Brooks 1991). That is, only recently sensed sensor values are used to determine actuator responses. For example, a mobile robot developed by Brooks (1986) has an obstacle-avoidance behaviour controlling the robot. The robot uses the most recent distance values, returned by a ring of sonars mounted around the robot's chasis, to determine the direction and distance of the robot's next motion. The intelligence of the robot is displayed when the robot moves away from objects. In general, the intelligence of behaviour-based robots is evident from the interaction between the robot and the environment in which the robot exists. That is, the intelligence is observed indirectly through activity. It should be noted that Agre and Chapman (1990) state that the robot itself does not need to be intelligent for intelligence to exist, but that intelligence is a result of the interaction between the robot and the environment in which the robot exists. For behaviour-based systems, we can say that intelligence is in the eyes of the beholder and is displayed through activity.

Though behavioural systems are able to perform tasks in the world, in our opinion they are not intelligent. Behavioural systems are simply able to produce appropriate responses to the stimuli which they can sense. That is, in our opinion, they do not understand the environment in which they exist, but simply respond to it.

On the other hand, internal representations, the backbone of the artificial intelligence community, are based on models of how people view the world. For example, Simmons and Krotkov (1991) use range finders to construct internal representations of terrain geometry surrounding a vehicle from which their system reasons. Another example is STRIPS, a reasoning system based on first-order logic, which reasons about the blocks world (Nilsson 1980). The blocks world defines every object in the world and all potential relationships between the objects. The general approach to generate solutions with internal representations is as follows. Determine the representations (objects) and relationships among the representations for the domain of interest. Then, set up a search space and search[1] through all possible combinations of relations and representations until one is found that satisfies the system goals (Nilsson 1980). The intelligence of systems using internal representations is based on the fact that they follow a logical flow of thought and is demonstrated when the system determines the correct solution to a problem. For autonomous vehicles, an example problem is that of determining the "best path" between two locations in the world.

Unfortunately, most work done by the artificial intelligence community is faced with an insurmountable hurdle that prevents the adoption of their ideas by autonomous vehicle designers. Many AI researchers assume that sensing systems will one day be able to uniquely identify all the objects in the world with which their systems reason. Unfortunately, this is an unreasonable assumption. Work done by Ullman (1984) on visual routines suggests that to recognize an object being sensed requires a priori knowledge about the object being sensed. Observation of this hurdle led to the beginnings of behavioural intelligence (Brooks 1986) and also provides a basis for our research.

Though computers using internal representations "reason" about objects, they are not intelligent in the sense that people are intelligent. They do not, in our opinion, understand the environment in which they exist. Instead, computers using internal representations simply apply a set of algorithms determined by their designer, in a specific manner also determined by their designer, to the scenario or problem with which they are faced. More specifically, these systems are reacting to stimuli in the same manner as behavioural systems, though the stimuli are different. Consequently the two components of autonomous vehicle control systems are functionally equivalent

[1]search strategies are often referred to as control strategies in the artificial intelligence community.

and, in our opinion, possess the same amount of intelligence: none.

2.2 Fundamental Assumption

The fundamental assumption of our research is radically different from those of previous groups. We assume that human beings are more than just a collection of biological components and that computers are just a collection of electrical components. This implies that intelligence, learning, understanding and reasoning do not have the same meaning for autonomous vehicles as they do for humans. It also means that the task performed by an autonomous vehicle control system is similar to the task performed by control systems of simpler machines, like motors, and that autonomous vehicle development is one facet of control theory.

In feedback control theory (Van de Vegte 1986), mathematics is used to describe control problems. However, much of the work with AVs has not been described with the mathematical tools of control theory because AVs often operate in under-sensed control environments. By "under-sensed" we mean that sensors are not able to sense the entire control environment simultaneously. For example, to control an AV to navigate through an obstacle field, the control environment is described by the position and size of all the obstacles in that obstacle field. In all but very restricted cases, sensors are unable to sense this control environment simultaneously, so the control environment is under-sensed. On the other hand, when sensors are able to sense the entire control environment simultaneously, we say that the control environment is "critically sensed". For example, to control the position of a motor shaft, the control environment is described by the position of the shaft, which is critically sensed by a position sensor.

2.3 Limitations of Different Control Strategies

In the framework of control theory, we accept that tangible limits on autonomous vehicle functionality exist. In fact, we can even look for those limits which, to the author's knowledge, is something that has not been done by anybody in the autonomous vehicle community. Unfortunately, many researchers do not detail the limitations they

experience because they do not view AV development as a control problem, but instead compare their work with humans, who are more than machines. For example, Daily et al. (1988) describe successful experiments with an autonomous land vehicle, though careful analysis of later publications, (Payton 1990), (Olin and Tseng 1991), reveals that the experiments were not so successful. Consequently, many of the ideas discussed in this section are not referenced because they have been obtained through the author's experience and by piecing together comments made by various researchers. Most of the ideas in this section are described through examples because, as mentioned earlier, suitable mathematical constructs are not available and because examples provide a clear picture of the concepts.

We refer to some of the limitations of AVs as brittleness associated with the use of internal representations. Brittle refers to a lack of robustness on the part of the vehicle to disturbances in its environment. Brittle is a vague term that encompasses the many odd ways that systems using internal representations fail. We discuss three forms of brittleness related to autonomous underwater vehicles, for more, the interested reader is referred to Malcom and Smithers (1990).

The first form of brittleness we discuss results from using internal representations to provide inputs for control algorithms when the system is operating in a dynamic environment. Consider using a world map to determine a path through the world. If any object in the world moves after the path is generated, it is possible that the vehicle will pass through the moved object. Consequently, internal representations are not robust to changes in the environment and therefore can only guarantee reliable control in undisturbed environments, in our case, a static world. This also implies that the validity of any internal variable is effectively unknown any length of time after it is sensed, and the likelihood of the variable being valid diminishes with time since the variable was sensed.

A second form of brittleness in control generated from world maps results from cumulative sensor errors. Errors do not accumulate in traditional control systems because traditional sensors sense their values relative to a fixed reference. Vehicle based positioning systems however, do not have any fixed reference with which they are associated. If an onboard positioning system has a 5% error, then after traveling $3m$ the vehicle can be up to $0.15m$ from its sensed location. Or, from the vehicle's perspective, everything in the world map is now out by up to $0.15m$. As the vehicle

continues to travel, the position of objects in the map becomes increasingly erroneous. Therefore, as time goes on, the likelihood of a collision with an object increases because the vehicle's knowledge of obstacle locations decreases. Cumulative errors are disturbances for which control systems that use internal representations are not robust, unless the internal representations can be updated periodically.

A third form of brittleness results from computational explosions of which, all world maps are susceptible. Systems operating in dynamic worlds have strict timing constraints on their control cycles to maintain stability. Unfortunately, the computational time required to use a world map grows exponentially with the number of elements in the map and can quickly grow beyond the limits of the control cycle. Consequently, the use of world maps can result in unstable systems because of excessive computational requirements.

Brittleness also results from excessive extrapolation of sensor data. This is not related to internal representations specifically, but instead is related to how humans interpret data and the internal representations used by humans. This form of brittleness is very evident when vision systems are employed as sensors in control systems. For example, the vision systems on both the Autonomous Land Vehicle (ALV) (Thorpe 1991) and the Stanford Cart (Moravec 1983) have been noted to be sensitive to both shadows and changes in illumination, a common vision problem. In Thorpe (1991), the vision system Vomors tracks lines on the edge of roads the ALV wishes to follow. However, if the wrong edge is detected by the vision system, the vehicle will follow the new edge, which might take the vehicle off the road. This brittleness results when parameters used by control algorithms are not sensed directly, but instead are extrapolated from other sensor data. Though errors with vision systems might be infrequent, their occurrence is unpredictable and can be catastrophic. A direct method of sensing the line on the edge of the road, if it existed, would be much more robust than using a vision system. Consequently, great care must be exercised when sensor data must be extrapolated for the control algorithm.

A problem common to behavioural systems has been independently labeled as as a command arbitration problem (Payton 1990) and behaviour conflict trapping (Bellingham 1990). Both terms describe a sequence of actions associated with independent behaviours, on a vehicle that is controlled by multiple independent behaviours, frustrating each other to the point that the vehicle is placed in a limit cycle.

By "independent behaviours", we mean that the vehicle is controlled by different control algorithms, depending on the present sensor values. The example Bellingham provides is that of an AUV which is confronted with an obstacle on its left and shallow water on its right. The two independent behaviours controlling the vehicle are called avoid obstacles and avoid shallow water. If obstacles are detected, the obstacle avoidance behaviour controls the vehicle. If no obstacles are detected, the avoid shallow water behaviour controls the vehicle. As we can see, the obstacle avoidance behaviour has the highest priority. Initially, the obstacle avoidance behaviour first turns the vehicle away from the obstacle and into shallow water. Once clear of the obstacle, the avoid shallow water behaviour turns the vehicle away from shallow water and back towards the obstacle. When the obstacle is again sensed, the avoid obstacle behaviour turns the vehicle back to shallow water. The vehicle oscillates between the obstacle and shallow water until it slips between the two or runs aground. This limit cycle may be temporary or permanent, depending on the situation. In this thesis, we call behaviour trapping "behaviour fusion" because the vehicle exhibits a behaviour that is a composite of both independent behaviours controlling the vehicle. As will become apparent, the effects of behaviour fusion must be incorporated into autonomous vehicle design methodologies.

A limitation of behavioural systems results from the fact that behavioural systems do not respond to stimuli which they cannot sense. Observing this fact with the limited sensing capabilities of today's vehicles, we see that today's behaviour-based control systems are unable to plan paths through environments in the way people plan paths. Consequently, today's behaviour-based systems are limited to bumbling through the environment like a person with no recollection of how to travel to their destination.

2.4 Control Architecture

All control systems for mechanical devices can be decomposed into a hierarchical organization of feedback loops like those described by Albus (1981). However, it is often difficult to discern the hierarchical loops when controllers are developed using the concepts of behaviours and reasoning systems. Instead, it is easier to view autonomous vehicle control structures as the two level hierarchy shown in figure 2.1. The low-level

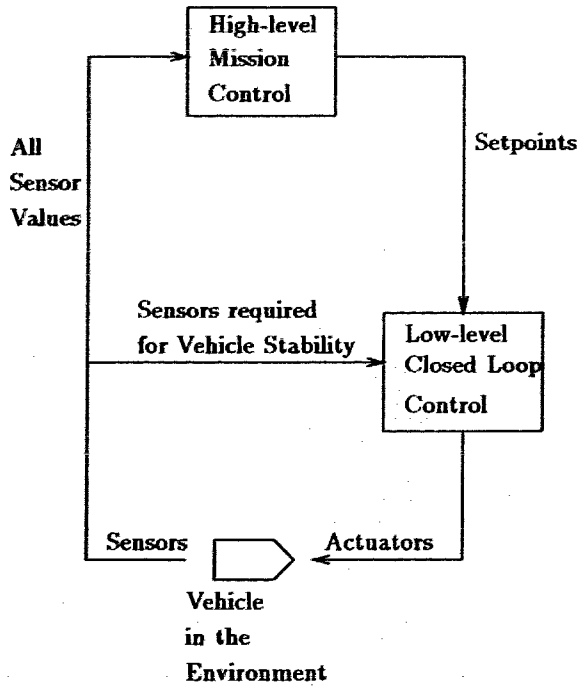


Figure 2.1: Standard autonomous vehicle control architecture

control system is responsible for maintaining vehicle stability in the world and is generally developed using feedback control theory. The control environment of the low-level control system is usually critically sensed. The high-level control system provides setpoints to the low-level control system, and is often associated with an under-sensed control environment. That is, control tasks which require "cognitive" type capabilities. For example, the high-level control system for an AUV might be concerned with obstacle avoidance, a generally under-sensed control environment, and provide setpoints in the form of waypoints or attitude commands to the low-level control system, which operates in the critically sensed control environment described by the vehicle's position and velocity.

All the systems described in this chapter adhere to this general architecture. The values used as setpoints for each system are different because the choice of setpoints is generally application specific. In the case of autonomous underwater vehicles, Bellingham (1990) found that waypoints provide better control than attitude setpoints because of the dynamical properties associated with underwater vehicles.

2.5 SHAKEY: SRI International

SHAKEY (Nilsson 1984)[2] is the first system we describe because it is one of the first autonomous vehicles developed and it is a complete system that demonstrates the ideas of traditional artificial intelligence in a real robot. SHAKEY is a mobile cart that has a camera and two range finders which it can pan and tilt. SHAKEY determines its position and orientation from two shaft encoders connected to its two drive wheels.

SHAKEY's environment consists of a few rooms that are connected with doorways. The walls are light and their edges highlighted with thick dark lines. Inside the rooms are a few blocks and wedges, each painted a distinct colour for easy identification by the vision system under proper lighting, which is provided.

The general architecture of SHAKEY's higher-level control system is shown in figure 2.2, which is the decomposition by function organization that is used by many

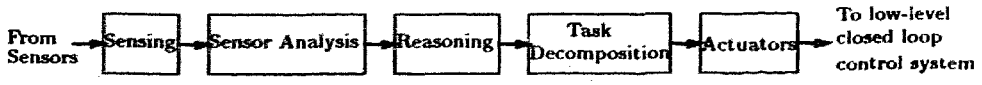


Figure 2.2: Decomposition by function type of controller organization

autonomous systems. The sensing and sensor analysis subsystems update internal representations that are stored in the reasoning subsystem. The reasoning subsystem uses the internal representations to develop a plan that accomplishes system goals, which are determined by an operator external to SHAKEY. A typical goal for SHAKEY is to organize the blocks in a certain fashion. The task decomposition subsystem decomposes the plan generated by the reasoning subsystem into physical actuator commands that are then executed by the actuator subsystem.

SHAKEY is intelligent because it reasons about the environment in which it exists. Reasoning for SHAKEY means that it generates and compares different plans to determine the best plan for accomplishing the system goal. The internal representations and the rules which SHAKEY applies to those internal representations are implemented with first order logic in the reasoning system STRIPS (STanford

[2]This is the classical SHAKEY reference, unfortunately we were unable to obtain it. Instead, our information comes from personal knowledge and secondary sources like (Nilsson 1980), (Brooks 1991) and (Shapiro and Eckroth 1987).

Research Institute Problem Solver) (Nilsson 1980). First order logic is one mathematical formalism that represents how humans deduce new facts from old facts. For a more indepth discussion about SHAKEY, the reader is referred to appendix A.

Though SHAKEY's success is said to stem from its ability to reason, its success really stems form the fact that SHAKEY's physical sensors, coupled with internal representations, are able to differentiate all situations requiring differentiation by the control algorithm. SHAKEY's reasoning system is part of that control algorithm. The situations requiring differentiation by SHAKEY pertain to the position and orientation of SHAKEY and the objects in its environment. This feat is accomplished through the meticulous engineering performed by SHAKEY's designers.

This meticulous engineering overcame the hurdle described in section 2.1 and mitigated the brittleness associated with internal representations. The hurdle is overcome by making everything in SHAKEY's environment uniquely identifiable. For example, all mobile objects, blocks and wedges, are uniquely identifiable through colour coding, and all static objects, walls, doors and corners, are uniquely identifiable by using the vision system in conjunction with the vehicle's position and orientation. Unfortunately, this approach is generally not possible for systems operating in the real world because of the vast number of different objects in the world. The brittleness associated with a dynamic world is avoided by having a relatively static world. That is, very little changes unless SHAKEY changes it. The brittleness associated with cumulative errors is mitigated by frequently calibrating the positioning system. For example, when SHAKEY's vision system detects a corner, whose location is known, SHAKEY can determine its own location by using the vision system in conjunction with the range finders. The brittleness associated with computational explosions is mitigated by not requiring SHAKEY to move at a fixed rate. SHAKEY can take as much time as is necessary to make a move. Finally, the brittleness resulting from excessive extrapolation of sensor data, namely vision, is virtually eliminated by the well constrained surface properties of all the objects in the environment. The meticulous engineering by SHAKEY's designers enables SHAKEY's sensing system and internal representations to differentiate all situations requiring differentiation by SHAKEY's control mapping.

Note that despite the fact that SHAKEY reasons, it always determines the same actuator response for specific situations differentiated by the physical sensors and

the internal representations. More specifically, SHAKEY's reasoning system can be viewed as a many-to-one mapping from sensor values, which include the values stored in internal representations, to actuator values. This also means that SHAKEY simply applies the rules it was given. If SHAKEY is given incorrect rules, it performs incorrectly.

2.6 Allen: MIT Mobot Lab

Allen (Brooks 1986) is the second system we describe because it is the first autonomous vehicle built using only behaviour-based control techniques. Allen is a mobile robot whose drive unit can rotate a specified number of degrees or move forward a specified distance. Allen has 12 sonar transducers evenly distributed around its circular chasis, one of which faces forward, that return values that are proportional to the distance to the nearest obstacle in front of the transducer. Allen operates in regular office environments.

Allen's higher-level control system is an implementation of the first three levels of the subsumption architecture (Brooks 1986). In the subsumption architecture, robots are decomposed into levels of task-achieving behaviours, as shown in figure 2.3. The higher levels (larger numbers in figure 2.3) subsume the levels beneath them,

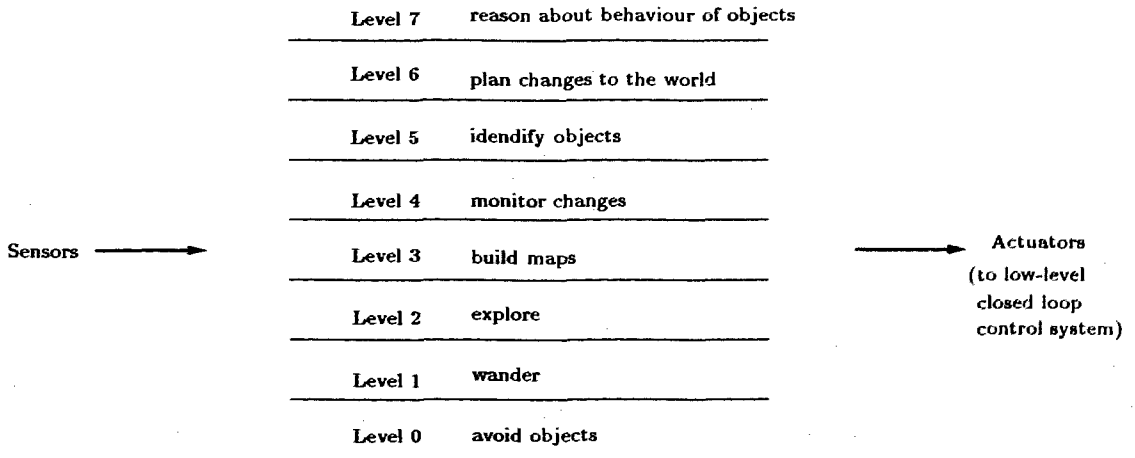


Figure 2.3: Subsumption architecture

thus ensuring that the robot always incorporates its lower-level functionality into its actions. The lowest level behaviour is avoid objects, which enables the robot

to survive. A robot that uses only a level 0 controller moves as far away from all objects in the world as possible. The level 1 behaviour, wander, makes the robot move randomly around the environment. The robot wanders in such a way that it avoids objects, and thus the level 1 behaviour subsumes the level 0 behaviour. The level 2 behaviour, explore, determines interesting objects in the environment, based on sensor values, and moves the robot towards those objects. The explore behaviour subsumes the wander behaviour by supplying the wander behaviour with a constant wandering direction. When the explore behaviour subsumes the wander behaviour, it also subsumes the avoid objects behaviour.

Allen is intelligent because it exhibits characteristics often attributed to biological creatures. The explore behaviour gives Allen the appearance of curiousity. The obstacle avoidance behaviour makes Allen appear to be aware of obstacles in its environment. The intelligence of behavioural systems is thought to be more on the level of an insect's intelligence than on the level of a human intelligence (Brooks 1989). For a more detailed discussion of Allen and the subsumption architecture, the interested reader is referred to appendix B.

Though Allen is said to be a demonstration of insect level intelligence, it is really a demonstration of the limited number of situations sensors are able to differentiate in an unstructured environment. The situations Allen is able to differentiate pertain to the size and shape of the free space surrounding Allen. By "free space" we mean areas in the environment that are free of obstacles. Note that these situations do not pertain to any object characteristics, as SHAKEY's do, because object characteristics cannot be differentiated by Allen's sensors. Interesting objects for Allen are the farthest points in sensed free space, a situation that can be sensed. One aspect of Allen and its control system, which determines actuator responses using only recently sensed sensor values, is that Allen is robust to changes in its environment. For example, Allen moves away from objects that move towards it. However, Allen is not capable of performing any practical tasks, it simply moves toward the farthest point in sensed free space which, if Allen is in a hall, causes Allen to move down that hall or, if Allen is in a large room, causes Allen to wander aimlessly in that room.

This lack of functionality was addressed in Herbert (Connell 1989), Allen's successor, which has the added capability of being able to follow walls. However, Connell found that wall following, with the intent to return to an initial location, without

any internal representations is only useful in extremely limited domains, and that in those domains wall following produces inefficient and circuitous routes. Other work by Connell (1992) shows that sensors are able to differentiate the difference between halls and the intersections of halls with doorways or other halls. This work also shows the increased functionality that can be obtained by adding internal representation to behaviour-based control architectures.

In summary, Allen, like SHAKEY, determines actuator responses from sensor values and some predefined set of rules specified by Allen's designer. The difference between Allen and SHAKEY is that Allen does not use any form internal representations.

2.7 Autonomous Land Vehicle (ALV): Hughes AI Centre

The final system we discuss, the Autonomous Land Vehicle (ALV), is a hybrid system that uses both internal-representation-based and behaviour-based control paradigms. The ALV is well described in the literature, (Payton 1986), (Daily 1988), (Payton 1990), (Olin and Tseng 1991), (Thorpe 1991) and illustrates the folly of using internal representations that cannot be sensed by sensors.

The ALV is an 8-wheeled vehicle designed to navigate over mildly rough terrain. It uses an onboard navigation system to determine the vehicle's position, orientation, pitch and roll relative to the world. The ALV also has a range scanner that scans an 80° horizontal and 30° vertical swath in front of the vehicle. Experiments with the ALV were conducted in a grassy field that contained gullies and rocks.

The control architecture of the ALV is shown in figure 2.4. The low-level control system of figure 2.1 is incorporated into the Motion Controllers block of this architecture. For autonomous vehicles, the ALV's architecture is relatively standard in the sense that it has a sensing leg (the left side of the figure) and an actuation leg (the right side of the figure). The bottom portion of the architecture is connected to real sensors and real actuators. As we move vertically in the perception system, data is assimilated producing a more complete picture of the environment. As we move down the actuation leg, tasks are decomposed into increasingly smaller subtasks to

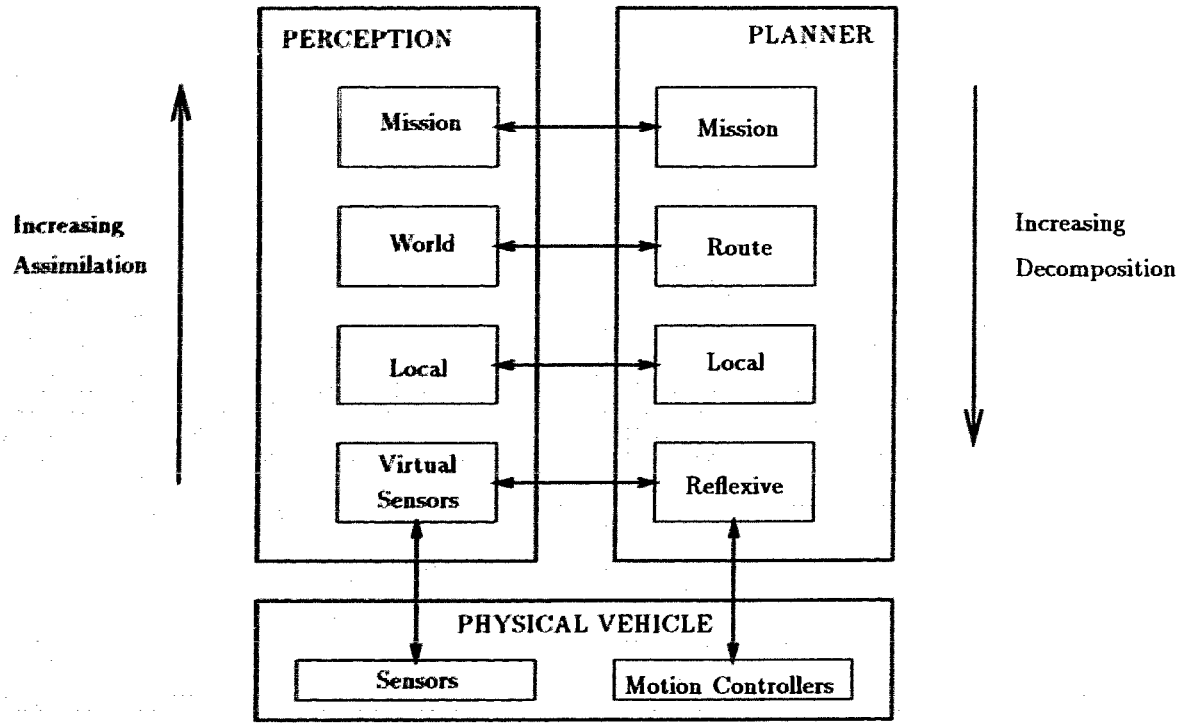


Figure 2.4: ALV control architecture

the point that they are actual motion commands. The different levels of the perception leg provide information which is pertinent to the tasks being decomposed in the respective levels of the actuation leg. The structure of figure 2.4 is described in (Daily 1988) but the ideas supporting the architecture are described in (Payton 1986).

The mission-planning module defines system goals and constraints, and instructs the mission-sensing module to configure the sensors to look for specific landmarks in the environment. This level of the architecture is designed to interact extensively with human mission planners. The world perception module maintains a world map of the environment that includes a list of landmarks indicating which have and have not been sensed. The route planner module uses the world map and the constraints of the mission planner module to determine a satisfactory route through the environment. The local perception module performs sensor fusion. It identifies landmarks and passes this information to the world perception module, and it identifies obstacles and environmental conditions and passes that information to the local planning module. The local planning module uses route information and environmental information to determine which reflexive behaviour will control the vehicle in the reflexive

planning module. The virtual sensor module detects specific environmental features as requested by the local perception module or the reflexive behaviour module. The term "virtual sensor" is used because the sensor values might not correspond to a single sensor, but might result from the processing of several sensor values. The reflexive behaviour module implements the currently active behaviour as specified by the route planning module. For a more indepth discussion of the ALV, the interested reader is referred to appendix C.

The ALV is considered to be intelligent because it uses both reasoning systems, at the higher-levels in the control architecture, and behaviours, at the lower-levels in the control architecture. Humans also use reasoning and behaviours. However, the ALV, like SHAKEY and Allen, simply applies actuator responses to situations that its sensors and internal representations are able to differentiate. The ALV is more functional than Allen because it uses internal representations. The ALV can traverse to distant locations in its environment, something that Allen cannot do. However, to accomplish this feat, the ALV uses a world map which is supplied to it by its designers. The sensors onboard the ALV are not able to sense the features stored in that world map and consequently, the ALV cannot sense changes in the environment, and is therefore brittle. Another reason the ALV is brittle is that the positioning system accumulates errors that cannot be mitigated by any other vehicle sensors. Consequently, the vehicle is limited to a range that is proportional to the rate at which errors accumulate in the positioning system. Though not directly stated in the literature, this limitation is evident from the fact that the ALV's longest trip was only 735m. The situations the ALV's sensors are able to reliably differentiate are those pertaining to terrain obstacles in front of the vehicle, based on range sensor data. Consequently, the ALV is able to successfully navigate local obstacles.

This work shows that despite complex internal representations, system functionality is limited by the actuator responses associated with situations that can be reliably differentiated by sensors.

2.8 Summary

This chapter provides a background and critical analysis of progress in autonomous vehicle research to date. The autonomous vehicles described in this chapter illustrate

that systems that do not use internal representations are robust to changes in their environments, and that systems that use internal representations can be brittle. These systems also illustrate that the underlying principle of operation of all autonomous vehicles is that they take inputs, which include sensor data as well as things like initial program parameters, and through some predefined set of rules and calculations produce a set of outputs that are used to set actuators. The ability of these systems to operate in a real world depends more on how well the sensing system is able to differentiate different situations in the world and the merit of the mapping that assigns actuator responses to each different situation than it does on the ability of the processor. The remainder of this thesis puts a concrete form to the ideas of situation differentiation and its applicability to autonomous vehicle control.

2.9 Discussion

In this section, we focus on how well researchers have followed through with their ideas. In the case of Nilsson, he returned to strictly computer work after the SHAKEY project. It is our opinion that the reason for this return is that the immense amount of pure engineering required to make AI operate in a wholly contrived world daunted the development of a system that would operate in the real world. More specifically, the inability of sensors to sufficiently differentiate objects in general environments prevents researchers from extending the SHAKEY paradigm to systems that operate in completely unstructured environments.

A similar trend is noted with Brooks, who has not constructed a vehicle that functions at level 3 of the subsumption architecture, which is the build maps level. However, a robot that does construct maps of its environment, TJ (Connell 1992), is able to identify only the office landmarks of the intersection between hallways and other hallways or doors, which are very specific landmarks. It is our opinion that the inability of sensing systems to sufficiently differentiate appropriate landmarks in general environments prevents the extension of the behaviour-based control paradigm to more complex tasks.

And finally, with regard to the ALV, research also seems to have halted. The researchers have not presented any work related to implementing the higher levels of their architecture. The work presented by Payton (1990) simply describes better

methods of using the sensor information available to their vehicle. In conclusion, it is our opinion that the work of the groups described in this chapter is limited by a lack of sensors able to sufficiently differentiate appropriate environmental stimuli.

Chapter 3

Sensor Actuator Mapping Theory

This chapter provides a mathematical framework for the concepts associated with situation differentiation and under-sensed control environments. Some elements of this framework are the ideas of sensor space and actuator space, which are part of a generalized control cycle described in this chapter. Sensor space is the conceptual interface between sensors and control laws and actuator space is the conceptual link between control laws and actuators. By "conceptual" we mean that there are no physical components associated with these spaces, not that they are abstract ideas because they are representable.

Using the generalized control cycle, we show that control laws can be represented as a many-to-one mapping from sensor space to actuator space. When the mapping is implemented on a digital computer, it is equivalent to a simple lookup-table, which we refer to as a Quantized Sensor to Actuator Map, or Q-SAM for short. By viewing control laws implemented on a computer as lookup-tables, it becomes obvious that computers simply implement a law, in our case a control law, and, in our opinion, do not themselves understand the environment in which they exist.

In section 3.1, we describe an example that we use to illustrate the ideas presented in this chapter. In section 3.2, we describe the generalized control cycle of an autonomous vehicle. In section 3.3, we expand the autonomous vehicle control cycle to show the role of internal representations in that cycle. In section 3.4, we describe sensor space, actuator space and environment space of the autonomous vehicle control cycle. In section 3.5, we describe the sensor transformation, control law and actuator transformation of the autonomous vehicle control cycle. In section 3.6, we introduce

Q-SAMs, which are lookup-table equivalents to computer-based control algorithms. Finally, in section 3.7, we summarize this chapter and in section 3.8 we provide a general discussion of some of the ideas presented in this chapter.

3.1 Example

Many of the ideas presented in this chapter are very general. To illustrate these ideas we use a simple example, that of an AUV whose task is to move to and then maintain a specific altitude in the water column. The dynamics of this simplistic altitude-keeping task are described by the linear differential equation

$$M\ddot{y}(t) + d\dot{y}(t) = F(t) \tag{3.1}$$

where M is the mass of the vehicle, d the drag, $F(t)$ the applied vertical force, $y(t)$ the altitude off the bottom and t is time. The desired altitude of the vehicle, y_o, is specified by an agent that is external to the AUV system. Since the controller is computer-based, we assume that the force $F(t)$ is applied in steps that have a time interval of Δt. This lets us represent the system in discrete time so that the parameters become

$$\begin{aligned} y_n &= y(t_n) \\ \dot{y}_n &= \dot{y}_n(t_n) \\ F_n &= F(t_n) \\ t_n &= t_{n-1} + \Delta t \end{aligned}$$

where n is the time index. For the control system discussions that follow, system error, err_n, is the difference between the desired altitude and the present altitude. More specifically

$$err_n = y_o - y_n.$$

3.2 Autonomous Vehicle Control Cycle

The control cycle of an autonomous vehicle is shown in figure 3.1. It consists of

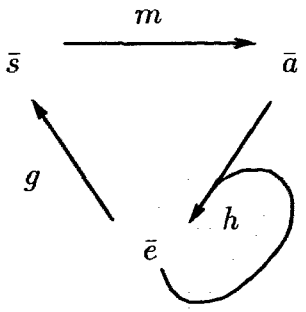


Figure 3.1: Autonomous vehicle control cycle

three vectors, $\bar{s}, \bar{a}, \bar{e}$, that represent the states of the sensors, actuators and environment respectively. Each vector is associated with a corresponding space, the sensor state vector with sensor space, the actuator state vector with actuator space and the environment state vector with environment space. The vectors can represent one position in their respective spaces at any given time. Connecting these spaces are three functions, g, m, h, that are the transformations made by the sensors, control law and actuators/environment respectively. A control cycle for the vehicle consists of the following: The present state of the environment is transformed by the sensors (transformation g) into the sensor state vector $\bar{s}$. The control mapping m takes the sensor values and through some predefined set of rules and/or calculations determines the actuator response $\bar{a}$. The transformation h uses the actuator commands to determine the new state of the environment $\bar{e}$. This cycle then repeats. The split arrow associated with transformation h signifies that h uses both the actuator commands and the present state of the environment to determine the new state of the environment. Mathematically the system is represented as

$$\bar{s}_n = g(\bar{e}_n) \tag{3.2}$$

$$\bar{a}_n = m(\bar{s}_n) \tag{3.3}$$

$$\bar{e}_{n+1} = h(\bar{e}_n, \bar{a}_n). \tag{3.4}$$

where

$$h(\bar{e}_n, \bar{a}_n) = h_e(\bar{e}_n) \circ h_v(\bar{a}_n) \tag{3.5}$$

is a composite function of the effects of the environment h_e and the effects of the vehicle h_v. By combining equations 3.2-3.4 we see that

$$\bar{e}_{n+1} = h(\bar{e}_n, m(g(\bar{e}_n))) = h'(\bar{e}_n) \tag{3.6}$$

where h' is a new function that incorporates all three transformations g, m and h. This function is interpreted to mean that once the vehicle is placed in an environment, it becomes part of that environment. That is, h' describes the vehicle's activity in the environment. Equation 3.6 also means that the more that is known about the environment h_e, the more the vehicle m, g and h_v can be designed to make h' reflect the wishes of the designer.

Describing the altitude-keeping example in the terms of the autonomous vehicle control cycle, the three state vectors are

$$\bar{s}_n = [err_n] \tag{3.7}$$

$$\bar{a}_n = [F_n] \tag{3.8}$$

$$\bar{e}_n = \begin{bmatrix} y_o \\ y_n \\ \dot{y}_n \end{bmatrix}. \tag{3.9}$$

y_o, the goal altitude, is part of the environment because it is supplied to the vehicle by an agent that is external to the vehicle. The transformation h is

$$\bar{e}_n = \begin{bmatrix} y_o \\ y_n \\ \dot{y}_n \end{bmatrix} \tag{3.10}$$

$$\stackrel{\mathbf{h}}{=} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{M}{d}(1 - e^{-\frac{d}{M}\Delta t}) \\ 0 & 0 & e^{-\frac{d}{M}\Delta t} \end{bmatrix} \begin{bmatrix} y_o \\ y_{n-1} \\ \dot{y}_{n-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{d} - \frac{M}{d^2}(1 - e^{-\frac{d}{M}\Delta t}) \\ \frac{1}{d}(1 - e^{-\frac{d}{M}\Delta t}) \end{bmatrix} [F_{n-1}] \tag{3.11}$$

$$= A[e_{n-1}] + B[a_{n-1}] \tag{3.12}$$

where A represents the natural changes of the environment, h_e, and B represents the effects of the vehicle's actuators on the environment, h_v. In general these effects are not necessarily decouplable, as they are for our linear example. The transformation g is

$$\bar{s}_n = [y_n] \tag{3.13}$$

$$\stackrel{g}{=} \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} y_o \\ y_n \\ \dot{y}_n \end{bmatrix} \tag{3.14}$$

$$= C[\bar{e}_n] \tag{3.15}$$

where C represents the sensor transformation from the environment. In this example, calculating the error err_n is part of the sensor transformation g, though it could easily have been part of the control mapping m. We have taken this approach because it provides a clearer picture of the ideas presented in this thesis.

3.3 Expanded Autonomous Vehicle Control Cycle

The expanded autonomous vehicle is shown in figure 3.2. We present the autonomous

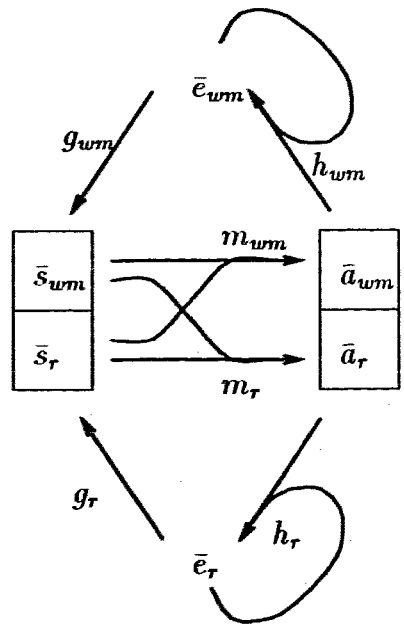


Figure 3.2: Expanded autonomous vehicle control cycle

vehicle control cycle in this manner to clearly illustrate the role of internal representations. In figure 3.2, the subscript *wm* indicates elements of the control cycle associated with internal representations (World Maps), and the subscript r indicates elements of the control cycle associated with real sensors and real actuators.

The ordinates of $\bar{e}_{wm}$ are the specific values of internal representations and are stored between control periods. In the example, the error sensor value err_n might be stored in an internal representation to be used the following control period with the new error sensor value. The sensor state vector for this scenario is

$$\bar{s} = \left[\frac{\bar{s}_{wm}}{\bar{s}_r}\right] = \left[\frac{err_{n-1}}{err_n}\right] \tag{3.16}$$

and the control mapping m can use the difference between the two error values as an estimate of the derivative of the error. The internal representations of $\bar{e}_{wm}$ are generally stored in computer memory. Therefore, some computer memory in an autonomous vehicle is part of the environment, from the perspective of the autonomous vehicle control cycle. The other portion of memory is that which is used to store the program code that implements the control law (mapping) m. The transformations h_{wm} and g_{wm} respectively write to and read from the internal representations $\bar{e}_{wm}$. Generally, they are inverse transformations whose net effect is a unity gain transformation. That is, the value written to memory is generally the value read from memory. Each control period m_{wm} updates the internal representations by deriving new values based on $\bar{s}_{wm}$ and $\bar{s}_r$ or simply copies values directly from $\bar{s}_{wm}$ to $\bar{a}_{wm}$. The former process changes the values of the internal representations and corresponds to updating the internal representations based on relevant sensor information and the latter process corresponds to simply maintaining the present values stored in memory.

Figure 3.2 provides visual clarification of some of the properties associated with internal representations. First, internal representations do not necessarily reflect any portion of the real environment. They can be a complete figment of the designer's imagination. Secondly, internal representations are a minimum of one control period old. That is, they must first be read from a real sensor value in $\bar{s}_r$ and stored in memory $\bar{e}_{wm}$ before they can be used as an internal representation from $\bar{s}_{wm}$, and this process requires one control period. Generally however, internal representations are older than a single control period, which is another reason that internal representations often do not correspond to the dynamic environment that they represent because the representation represents the state of the environment a long time ago. This lack of correspondence is one cause of system brittleness.

3.4 The Spaces

The three spaces in the control cycle, represented by the three vectors $\bar{s}, \bar{a}$ and $\bar{e}$, are the conceptual links between the physical components of the system and are illustrated in figure 3.3. The dimensionality of these spaces is not necessarily the

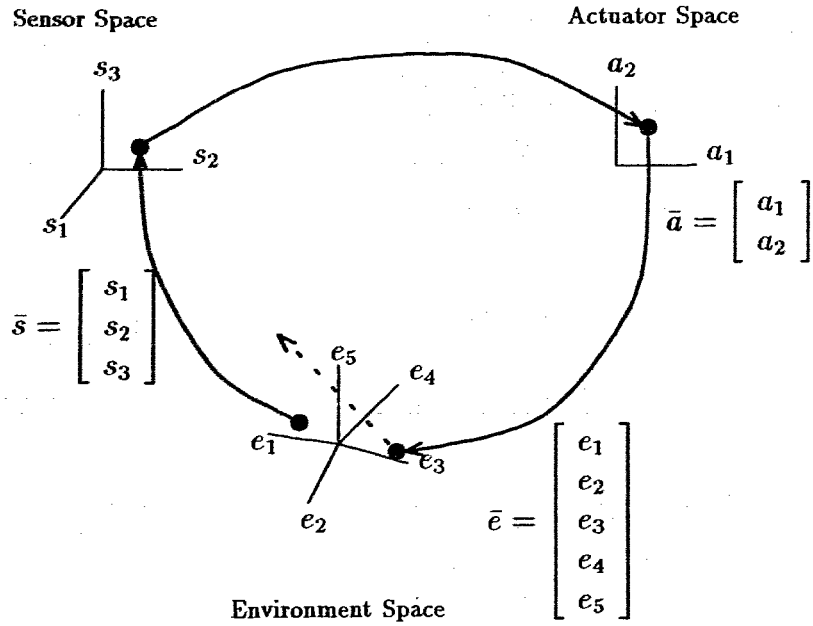


Figure 3.3: Spaces of the autonomous vehicle control cycle

same. By "dimensionality" we mean the number of axes in the space, which is equal to the number of ordinates in the respective state vector. Of these spaces, sensor space is our focus because we have found that sensor space requires the most effort to understand and design. One reason for this fact is that there are many different sensors available for autonomous vehicles, each able to sense different aspects of the environment. On the other hand, there are very few choices of actuators. For an AUV we are generally restricted to thrusters that operate in a single direction and rudders. Also, environment space is usually so large that designers often consider only the elements of it that can be sensed and the definition of those elements is usually obvious.

It is important to note, as mentioned in the previous section, that the ordinates of the sensor state vectors do not necessarily correspond to the output of a specific sensor, though this is often the case. Some sensor state vector ordinates might be associated

with internal representations stored in computer memory. Likewise, some ordinates of the actuator state vector and the environment state vector might represent internal representations stored in computer memory.

Computers used for control see the world through their sensors. Consequently sensor space represents a "computer's eye" view of the world. Every situation in the environment is transformed into one location in sensor space. These locations need not be unique for unique situations. That is $g : e_n \longrightarrow s_n$ is a many-to-one transformation. For the controller to recognize situations in the world as being different, however, they must correspond to different locations in sensor space. In the example, sensor space is the one dimensional continuous space represented by the error signal.

We say that sensor space is "complete" for a given task if all situations in the world requiring differentiation by the control mapping are associated with different locations in sensor space. This condition is satisfied for the example. An example of an incomplete sensor space is if we were to place a velocity limit on the altitude-keeping task. The sensor space of the example is then incomplete because situations with velocities in excess of the limit are not differentiable from situations with velocities below the limit. A complete sensor space for this case has an axis for the error signal and another axis for system velocity or an estimate of system velocity.

Associated with completeness, is the concept of situation identification. Situation identification refers to the process of identifying the meaning associated with each differentiable region in sensor space. In the example, the meaning of each region in sensor space is a specific altitude error. This process is relatively straightforward when the control environment is critically sensed, as it is for the altitude example, because the meanings are a function of only the sensor transformation g. However, when the control environment is under-sensed, the meaning associated with differentiable regions of sensor space are a function of the entire autonomous vehicle, and these meanings are not so straightforward, as will become apparent in chapter 5

Another concept in the design of sensor space is the goal region. The goal region is an area of sensor space that corresponds to the completed task of the vehicle. That is, if the vehicle has completed its task, then the sensor state vector represents a point in the goal region of sensor space. For the example, the goal region is the point $err = 0$ in sensor space. Though a point is acceptable it is common to specify a region like $-0.05m \leq err \leq 0.05m$ when being within $5cm$ of zero error is sufficient. In general,

multidimensional sensor spaces have goal regions that are also multidimensional. Depending on how the goal is specified, the dimensionality of the goal region can vary, but its dimensionality is always less than or equal to the dimensionality of the sensor space. For the point $err = 0$ the dimensionality of the goal region is zero and for the line segment $-0.05m \leq err \leq 0.05m$ the dimensionality of the goal region is one, which is equal to the dimensionality of the sensor space.

Associated with goal regions are subgoal regions which are supersets of the sensor space goal region, but subsets of sensor space. Subgoal regions can be used to represent progress towards the goal region or to prioritize different routes through sensor space to the goal region. When the vehicle travels to the goal region, it follows a trajectory through sensor space, and some trajectories are preferable to others. For the example, a subgoal is the region $-0.5m \leq err \leq 0.5m$ which indicates that the system is approaching the goal region. For multidimensional sensor spaces, subgoals are any volume of sensor space that is larger than and encompasses the goal region. A common subgoal in a multidimensional sensor space is when one of the ordinates of the sensor state vector is equal to that of the goal region. Note that vehicles complete tasks in sensor space, whereas designers view tasks in environmental space. Therefore, one reason that systems do not complete the tasks for which they were designed is that the transformation g maps situations that do not correspond to the completed task (system goal) into a region of sensor space that the designer believes corresponds to the completed task (goal region). That is, the designer did not understand the situations differentiated by the sensors, and consequently the system's sensor space is incomplete.

Before we discuss the final concept associated with sensor space, partitioning, we must consider the effects on sensor space when control laws are implemented on digital computers. All values used by a computer are digital, or quantized. Consequently, sensor space, and also actuator space, are quantized. When each sensor value is quantized independently, which is usually the case, the sensor space is broken into a countable number of multidimensional rectangles, as shown in figure 3.4. Each rectangle represents several situations and can be thought of as a single equivalence class in a partition of sensor space caused by quantization.

Partitioning is the final concept we discuss in sensor space design. The partition caused by quantization, P_Q, is the most significant because it must be consistent

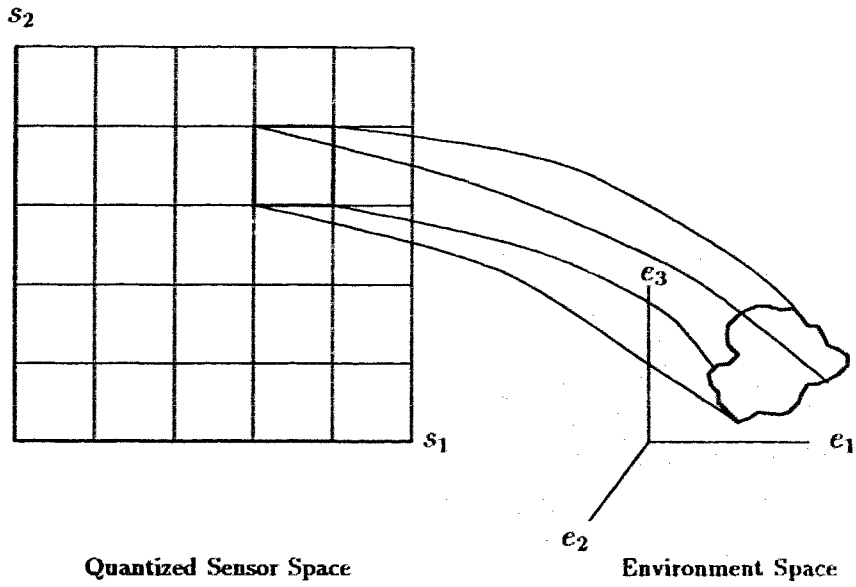
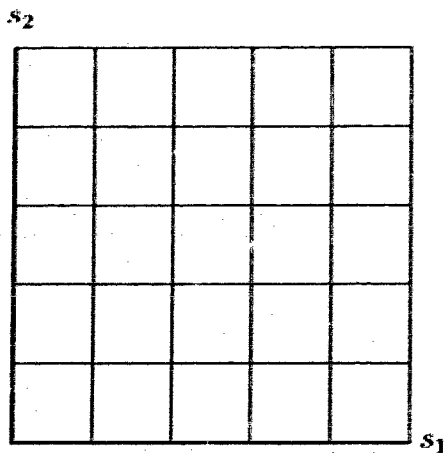


Figure 3.4: Quantized sensor space

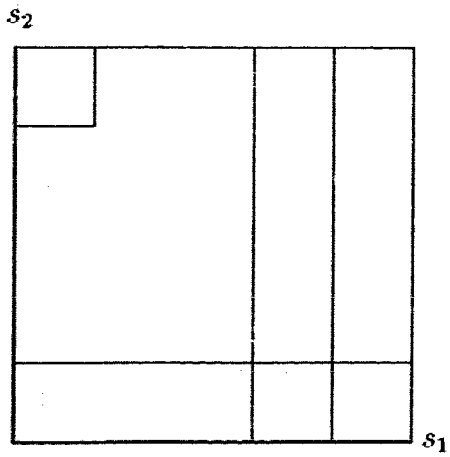
with all other partitions on sensor space. We say that P_Q is consistent with another partition, say P_A, when each equivalence class of P_Q is a subset of only one equivalence class of P_A, as shown in figure 3.5. More specifically, the borders of the equivalence classes of P_A are colinear with the borders of the equivalence classes of P_Q though the converse is not necessarily true. If the converse is true then the two partitions are mutually consistent and they are the same partition. Let P_A be the partition whose equivalence classes represent regions of sensor space that are associated with the same actuator responses under the mapping m. If P_Q is not consistent with P_A then at least one equivalence class of P_Q overlaps two equivalence classes of P_A. In other words, different portions of the same quantization region require different actuator responses, which is impossible to implement with any mapping m. Therefore to be complete, a quantized sensor space has an additional constraint that is: P_Q must be consistent with all other partitions placed on sensor space. Another partition that is always present is the goal partition, P_G. P_G has at least two equivalence classes; one that is the goal region and one that is not.

Actuator space is very similar to sensor space with the only exception being the lack of a goal region. Actuator space represents every effect that the actuators can have on the environment. Different locations in actuator space correspond to different



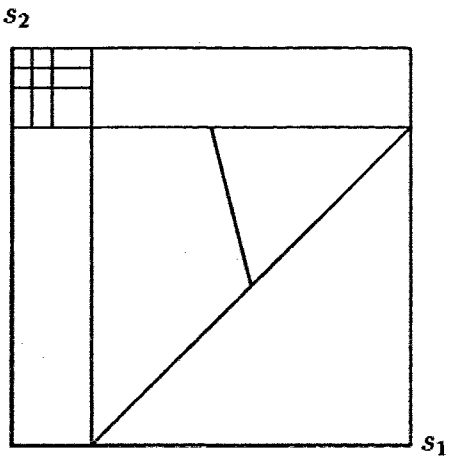
Quantization
Partition

P_Q



Consistent
Actuator Partition

P_A



Inconsistent
Actuator Partion

P_A

Figure 3.5: Consistent sensor space partitions

actions in the environment. In our example, actuator space is the one dimensional axis representing the force applied to the system. The actuator transformation h transfers every location in actuator space into its corresponding action in the world.

Like sensor space, actuator space is complete if each unique actuator response in the environment required by the control mapping m is represented by a different location in actuator space. The actuator space associated with the vertical thruster of our example is complete. An example of an incomplete actuator space is that associated with a horizontally mounted thruster because it has no actuator response that moves the vehicle vertically. That is, the vehicle cannot physically move to the goal region y_o.

Quantization in actuator space replaces a group of actuator responses in the environment with a single response. Like sensor space, the partition caused by quantization must be consistent with every other partition placed on actuator space.

Environment space represents everything about the environment. In our example, environment space is the altitude, velocity and desired altitude of the AUV. For a general AUV, environment space includes things like the AUV's position, objects and their positions, ocean currents, temperature and computer hardware, many of which cannot be sensed by the mapping g. One aspect of environment space is that it does not consist of only things that are external to the physical vehicle. Environment space also includes things like computer memory used for internal representations. Note that a vehicle's functionality is demonstrated in that part of environment space that is external to the physical vehicle, and not in any internal representations. Consequently, designers that rely heavily on the use of internal representations must be aware that their vehicle's performance is not judged on the values stored in those internal representations, but is judged on the actions taken in the world.

3.5 The Transformations

The three transformations, m, g, and h are the physical connections between the three conceptual spaces of the autonomous vehicle control cycle of figure 3.1. By "physical" we mean that there is generally some sort of hardware associated with the transformations. Though each transformation is independent of the rest, they are tightly coupled by the spaces, and consequently must be designed together. Specifically, the output

of the sensor transformation g must correspond to the inputs required by the control mapping m and likewise the output of the control mapping m must correspond to the abilities of the actuators.

By stating that the transformations must be designed together, we mean that autonomous vehicles must be designed as a whole. Decomposing autonomous vehicles into the manageable subcomponents of figure 2.2 has lead to important details slipping between the cracks of the interfaces (Malcom and Smithers 1990) which, in the autonomous vehicle control cycle, are sensor space, actuator space and environment space. That is, when the different subcomponents of figure 2.2 are developed independently, assumptions are often made about the other subcomponents that might be unreasonable. For example, a team designing a reasoning system for an autonomous vehicle might assume that the sensing system will be able to reliably detect every object in the environment, which might not be possible. Unfortunately, it is not until the system is finally assembled that the lack of attention paid to the interfaces becomes apparent in the system's poor performance or brittleness. Consequently, sensor space, actuator space and environment space must be clearly defined before the components of m, g and h are designed and built.

The first transformation we discuss is the sensor transformation g, which takes situations in the environment and converts them into locations in sensor space. The requirements on the transformation are specified by situations in the environment that the control mapping m needs differentiated. We say that g is "sufficient" if the sensors transform each situation in the world that requires differentiation into a different location in sensor space. For the example (altitude-keeping), the sensor transformation simply converts small ranges of errors (the ranges are due to quantization) in the environment into unique locations in sensor space. The example sensor transformation is sufficient for a proportional control law. For under-sensed control environments, the identification of situations differentiated in sensor space is a function of the entire vehicle. Consequently, the sufficiency of the sensor transformation is also a function of the entire vehicle. The quantization of sensor values into the quantized sensor space of a computer is also part of sensor transformation g.

The second transformation we discuss is the control mapping m. m is the control law of an autonomous vehicle, and m defines which actuator responses are applied to which differentiable situations. More specifically, m maps each location in sensor space

into one location in actuator space. That is, $m : \bar{s}_n \longrightarrow \bar{a}_n$, like g, is a many-to-one mapping. The reason that m is a many-to-one mapping is that machines, which are driven by rules, can apply only one set of rules to a given set of inputs. If situations are associated with the same actuator response, they do not need to be differentiated from one another in sensor space. Note that the "intelligence" of an autonomous vehicle is displayed by its ability to apply the appropriate actuator responses to appropriate environmental situations, and it is m that does this assignment.

Another concept associated with the control mapping m is that m defines how the vehicle's position in sensor space changes. The reason for this fact is that m is responsible for assigning actuator responses, which change the environment, which in turn change the sensor values and consequently, the vehicle's location in sensor space. More specifically, the control mapping m determines the vehicle's trajectory through sensor space, from its present location to the goal region. For the example, the trajectory is a straight line because sensor space is one dimensional. We say that a control mapping is "adequate" if all trajectories through sensor space lead to the goal region. That is, the system is able to move to the goal region from anywhere in sensor space. It is again noted that an autonomous vehicle views its progress in sensor space, whereas designers view the vehicle's progress in environment space.

The concept of adequacy is important when working with under-sensed control environments because optimal control, in the sense of feedback control theory (Van de Vegte 1986), might not be possible. For systems operating in critically sensed control environments, optimal control is possible because the system's progress can be quantitatively analysed from its sensor values. However, in under-sensed control environments this analysis is not possible. For example, in all but very restricted cases, it is impossible for a vehicle to determine an optimal path through an unknown obstacle field that it cannot critically sense.

Finally, the mapping h models the effects of the actuators on the environment. h is different than g and m in two major ways. First, it is in h that the time step Δt takes its effect in the control cycle. g and m, unlike h, are generally assumed to be instantaneous. Secondly, h is really a composite of two transformations, as shown in equation 3.5. h models the effects of the vehicle's actuators, h_v, and the effects of the environment on itself, h_e. For the example, h_v represents the effects of applying a force to the system for one control period, and h_e represents the effects of the vehicle's

momentum. Note that h_v might not have any effect on many of the elements of the environment $\bar{e}$. For instance, a vehicle cannot control the ocean currents but instead must move with them.

3.6 Quantized Sensor to Actuator Maps (Q-SAMs)

The mapping m, in conjunction with the autonomous vehicle control cycle, can implement any form of analog or digital control law. In this thesis, we are mainly interested in digital control mappings because most autonomous vehicle control systems are implemented with computers. Consequently, the discussion in this section focuses on digital control mappings, though many of the ideas presented also apply to analog control mappings.

It is not hard to envision the low-level control laws of figure 2.1 as a many-to-one mapping from sensor space to actuator space, but the high-level control laws of figure 2.1 have rarely been described in these terms. In fact, Badreddin (1991) simply said that we cannot even assume that these functions are one-to-one, which is the case. In the next few paragraphs, we will show how m can be represented as a simple lookup-table and how that lookup-table, as part of the autonomous vehicle control cycle, can implement the higher-level control mappings of figure 2.1.

As a digital mapping, m maps each region of quantized sensor space, which we refer to as a quantum, into one region of quantized actuator space. Since there are a finite number of quanta in quantized sensor space, we can use each quantum to index a unique location in a lookup-table, in which is stored the appropriate actuator response defined under m. We refer to this lookup-table as a Quantized Sensor to Actuator Map, or Q-SAM for short.

To show that Q-SAMs, in conjunction with the autonomous vehicle control cycle, can implement higher-level control mappings, consider some facilities of higher-level controllers generally though unimplementable with a lookup-table representation: random numbers, clocks and program variables. With respect to the autonomous vehicle control cycle, random number generators and computer clocks are part of the environment $\bar{e}$. That is, random number generators and computer clocks are each associated with one ordinate of the sensor state vector $\bar{s}$ and one axis of sensor space.

Higher-level control laws are generally implemented on a computer by a computer

program, which uses variables, that we refer to as "program variables". These programs have two types of variables: those which are used only during the control cycle, called temporary variables, and those which are stored between control cycles, called permanent variables. Temporary variables like i in the code segment

```
outvar = proc(invar)
        {
          for i = 1 to 10
          {
            temp = temp + i * invar
          }
          outvar = temp
        }
```

are part of the control mapping m. If m is represented in its Q-SAM form, i is incorporated into the Q-SAM when it is filled. By "fill" we mean the process of storing an actuator response in each location (quantum) in the lookup-table. Permanent variables, like *last_pos* in the code segment

```
outvar = proc(invar)
        {
          static last_pos
          outvar = invar + last_pos
          last_pos = invar
        }
```

are internal representations, and are part of the environment $\bar{e}_{wm}$ of figure 3.2.

Despite the fact that higher-level autonomous vehicle control systems have a Q-SAM equivalent, people rarely thing of them in such terms. The reason for this state of affairs is that programs, the form of most higher-level controllers, obscure the Q-SAM representation. Consider the line of code

$$y = 5x + 3. \tag{3.17}$$

To a casual observer it is not obvious that this line of code implemented on an 8-bit computer is equivalent to the Q-SAM in figure 3.6, where x is the sensor input and y

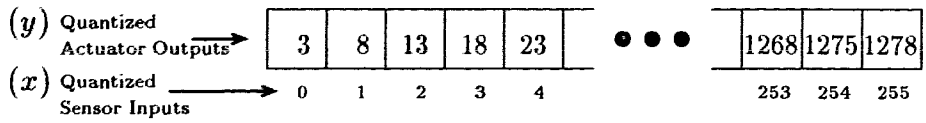


Figure 3.6: Q-SAM equivalent of the line of code: $y = 5x + 3$

is the actuator output. However this is the case.

There are several advantages to using the computer program implementation of higher-level control laws instead of the Q-SAM implementation. First, programs are easier to understand than lookup-tables. The line of code 3.17 is much easier to understand than the functionally equivalent Q-SAM in figure 3.6. Also, programs generally require significantly less memory than their Q-SAM counterparts. The Q-SAM of figure 3.6 requires 512 bytes to implement with 2 byte integers, whereas the line of code 3.17 can be represented in as few as 5 bytes.

On the other hand, there are several advantages to using the Q-SAM implementation of a control law. First, Q-SAMs are fast. The response time of the entire control algorithm is reduced to that of a memory access, which is why people often use lookup-tables to replace time-critical algorithms. Secondly and more importantly, the Q-SAM representation allows us to apply any actuator response to any situation. That is, the Q-SAM is an explicit representation of the mapping m, and can support any form of control law.

3.7 Summary

This chapter provides a mathematical framework for the concepts associated with situation differentiation. The interfaces between system components, namely sensor space, actuator space and environment space, must be clearly defined and understood before work on the system subcomponents can begin. This requirement is accomplished by constructing complete sensor and actuator spaces. Both high-level and low-level control mappings have a lookup-table equivalent called a Q-SAM. In the next chapter, we explore several ways that control laws can be developed using Q-SAMs and the concepts of situation differentiation.

3.8 Discussion

In section 3.6, we showed that both high-level and low-level controllers have a lookup-table equivalent called a Q-SAM. Coupling that discussion with the discussion of section 3.3, we see that computers running a program have a Q-SAM equivalent in terms of the autonomous vehicle control cycle. Note that the sensor state vector might be a time series of all values encountered by the computer. For example, the sensor state vector of a learning algorithm includes all combinations of inputs and outputs supplied to that algorithm. In doing so, we have shown that computers, and autonomous vehicles, simply implement the set of rules that was used to fill the Q-SAM, or equivalently, write the program. It is our opinion that the intelligence resides in the person that designed the rules used to fill the Q-SAM and not in the computer or autonomous vehicle.

As a final note, consider removing g_τ, $\bar{e}_\tau$ and h_τ from the expanded autonomous vehicle control cycle of figure 3.2. This corresponds to removing the physical components of our vehicle and leaves us with a dynamic model of a computer simulation where $\bar{s}_\tau$ is the simulation's inputs and $\bar{a}_\tau$ is the simulation's outputs. From the discussion of section 3.3, it is clear that the results of a simulation are completely contrived. Therefore, results obtained through simulation might have little, or no meaning with regard to systems operating in a real world.

Chapter 4

Situation-Based Control

This chapter demonstrates how situations can be used to develop and implement control laws. This task is accomplished by exploring various methods of filling Q-SAMs with control laws. By "filling" we mean storing specific actuator responses in each quantum of the Q-SAM. In the first few sections, we show that Q-SAMs are an explicit representation of a control law by filling the Q-SAM first with a proportional control law and then with the control law of an expert control system. In the latter section, we present evidence that adaptive situation-based control is feasible but not, as yet, practical.

Through the course of our work, we found that filling Q-SAMs with control laws can be time consuming due to the large number of quanta that are generally associated with a single Q-SAM. To reduce the number of situations that must be encountered during the filling process, we employ the use of a function, that we refer to as an actuator response distribution function, which distributes actuator responses among similar situations (similar Q-SAM quanta).

It is important to note that we are not trying to develop optimal controllers in this chapter because that is always application dependent, and there is a lot of control theory for any particular application. Instead, our purpose is to illustrate some of the facilities of situation-based control and the Q-SAM representation, and show that Q-SAMs can be used to implement many forms of control laws.

In section 4.1, we describe the system with which we conduct our experiments in this chapter. In section 4.2, we describe the actuator response distribution function that we use to increase the rate at which Q-SAMs are filled. In section 4.3, we

show a method of filling Q-SAMs with rule-based control laws by filling a Q-SAM with a proportional control law. In section 4.4, we fill a Q-SAM with the control law of an expert controller, whose control law we cannot describe mathematically, by a technique we refer to as "downloading". In section 4.5, we use the Q-SAM to experiment with adaptive situation-based control. In section 4.6, we discuss some of the philosophical implications of downloading control laws. Finally, in section 4.7, we summarize the results described in this chapter, and in section 4.8 we provide a general discussion of those results.

4.1 Example

In this chapter, we again use the altitude-keeping example described in chapter 3. We set the parameters of equation 3.1 to have a mass (M) of $35.0kg$, a drag coefficient (d) of $20.0kg/s$ and a control period (Δt) of $2.0s$. These parameters are estimates of the parameters governing the vertical motion of an AUV under development at the Underwater Research Lab. The control laws are implemented with a 65 word Q-SAM whose range of operation is $[-2.0m, 2.0m]$. That is, the sensing system is capable of differentiating 65 different error situations in the range $[-2.0m, 2.0m]$ as shown in figure 4.1. It should be noted that errors greater than $2.0m$ are associated with the

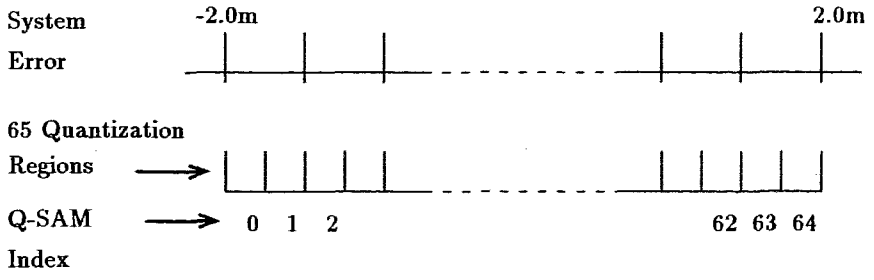


Figure 4.1: 65 quanta Q-SAM

situation of quantum 64, and errors less than $-2.0m$ are associated with the situation of quantum 0.

The control diagram for this example is shown in figure 4.2. The desired altitude, y_o, is set externally with respect to the control system. y_o is used to generate an error signal, which is quantized into one of 65 situations in sensor space. The response to

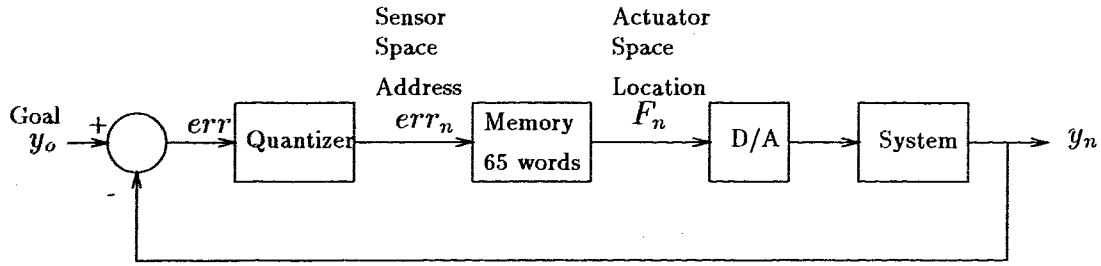


Figure 4.2: Control diagram for altitude-keeping with a 65 quanta Q-SAM

each of the 65 situations are stored in the 65 Q-SAM quanta. During the control cycle, err_n is used to access the appropriate response stored in the Q-SAM, which is converted to an analog signal and applied to the AUV.

At this point, we define the term control surface, and discuss some of the ideas that are associated with control surfaces. The control surface is the curve generated when sensor space is plotted versus its corresponding actuator responses. For this example, a typical control surface is shown in figure 4.3. Two control surface features

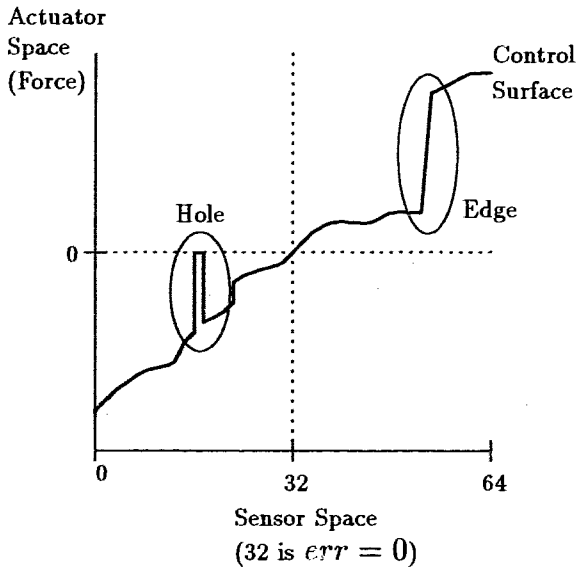


Figure 4.3: Control surface showing control surface features

illustrated in figure 4.3 are edges and holes. An edge is a drastic change in actuator responses associated with similar situations. Situation similarity is an application specific definition. For this example, similar situations are those associated with similar errors. A control surface hole is a region of the Q-SAM whose actuator responses

are default values. In our example, the default value is zero.

4.2 Actuator Response Distribution Function

We have found that filling Q-SAMs can be a tedious and time consuming task because there are many locations in Q-SAMs, and each requires an individual actuator response. To speed up this process, we implement a function that distributes learned responses among situations that are similar and expected to have similar responses. By "learned response" we mean responses that have been determined to be appropriate for a particular situation. We call this function an Actuator Response Distribution Function, or ARDF for short. Actuator response distribution functions should have greater effects on situations that are more similar to the situation for which the response was learned than on situations which are less similar. To satisfy this requirement, we have chosen the following exponential distribution function for the altitude-keeping example. We chose this function because it produces a smoother control surface than the other distribution functions that we tested. Similarity of situation is determined by proximity in sensor space. The function is

$$f_{j(new)} = w_{ij} f_i + (1 - w_{ij}) f_{j(old)} \tag{4.1}$$

where

$$w_{ij} = be^{(-c|i-j|)} \tag{4.2}$$

and i is the quantum associated with the learned actuator response, j is the quantum being updated, f_i is the learned response, $f_{j(old)}$ is the old response stored in quantum j, $f_{j(new)}$ is the updated response to be stored in quantum j, b is the averager weight for quantum i, and c is the diffusion constant. b is called the averager weight for quantum i because when quantum i is being updated, the exponential portion of equation 4.2 is equal to 1.0. c is called the diffusion constant because it effects the width of the distribution function, which will be discussed in the next paragraphs. For this actuator response distribution function, learning a response for one quantum alters the responses associated with all other quanta.

To envision the operation of this actuator response distribution function, consider the Q-SAM of the example filled with zeros. This corresponds to a flat control surface

with all the actuator responses being 0.0. Suppose the system determines that an actuator response of 1.0 is an appropriate response for the situation associated with quantum i. Storing this response in the Q-SAM leaves a Q-SAM that is filled with zeros, except in quantum i, which has a response of 1.0. The Q-SAM is still relatively empty. By applying the ARDF, more situations than that associated with quantum i will have non-zero actuator responses. Figure 4.4 shows the repetitive application of

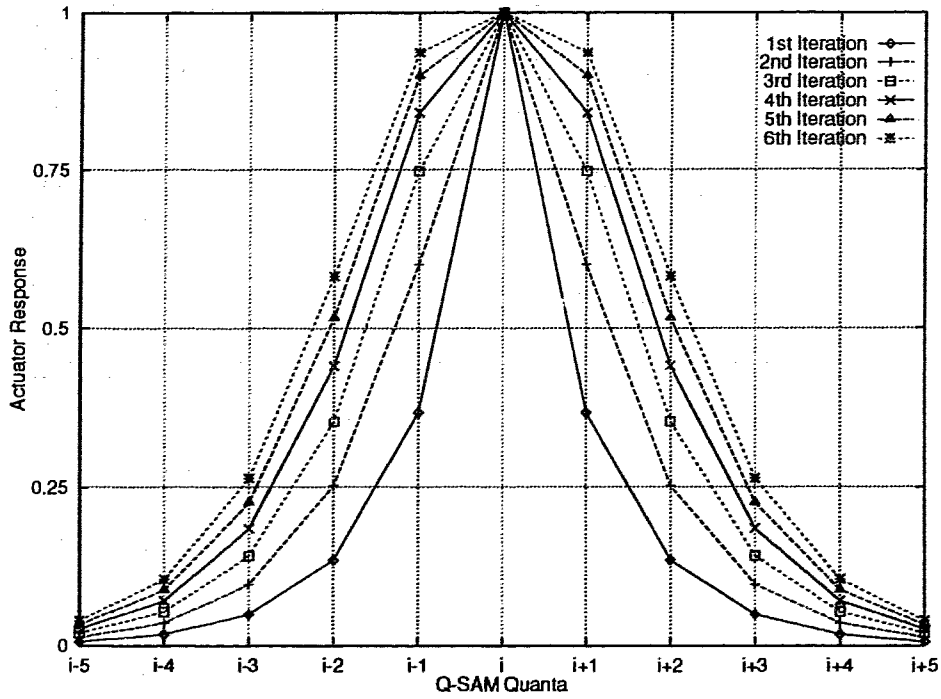


Figure 4.4: Repetitive application of the actuator response distribution function

the actuator response distribution function, with $b = 1.0$ and $c = 1.0$, to the Q-SAM. The width of the function is controlled by the parameter c. The line labeled 1^{st} iteration shows the control surface after the ARDF is applied once. After application of the ARDF, many situations are associate with non-zero actuator responses. As figure 4.4 shows, repetitive application of the ARDF does not change the response associated with quantum i, but makes the responses associated with other quanta increasingly similar to that of quantum i. It should be noted that the effects of the ARDF are less significant when the responses are similar, which is illustrated by the decreasing effects of the repetitive application of the ARDF. The repetitive application of the ARDF also shows the exponential convergence of the ARDF. Note also that the ARDF has a greater effect on situations which are more similar to the situation

for which the response was learned (i) than situations which are less similar. For the parameters chosen, the ARDF has little effect 4 sensor space quanta away ($i-4$) from the quantum associated with the learned response.

It should be noted that actuator response distribution functions like this one have a smoothing effect on the control surface in the areas of the Q-SAM in which they are applied. Smoothing is inherent when trying to make the actuator responses of similar situations similar. Consequently, care should be taken when using ARDFs on Q-SAMs so as not to remove desirable edges in the control surface. As a precaution, ARDFs should not be used in regions of a Q-SAM expected to have dissimilar responses. For the altitude keeping example, all situations in sensor space are expected to have similar actuator responses to those of neighbouring sensor space quantum. Consequently this ARDF can be used throughout the Q-SAM of the altitude-keeping task.

4.3 Filling a Q-SAM with Proportional Control

Most control theory requires a system model on which to base the development of the control law. The model is used to determine a control law and to analyse the effects of altering different design parameters. In this section, we show how to fill Q-SAMs with control laws developed from system models by determining a proportional control law for the example system and filling the Q-SAM with that control law.

The proportional control law is represented by the equation

$$F_n = k_p(y_o - y_n) \tag{4.3}$$

where k_p is the proportionality constant to be determined by the control engineer. To determine the appropriate value of k_p, the engineer uses standard digital control theory (Jacquot 1981) and puts equations 3.1 and 4.3 together and takes the Z-transform of the system. This process yields a characteristic equation of the form

$$z^2 + c_1 z + c_2 = 0 \tag{4.4}$$

where

$$\mathbf{cl} = \frac{\Delta t k_p}{d} - (1 + e^{-\frac{d}{M}\Delta t}) - \frac{M k_p}{d^2}(1 - e^{-\frac{d}{M}\Delta t}) \tag{4.5}$$

and

$$c_2 = e^{-\frac{d}{M}\Delta t}(1 - \frac{k_p \Delta t}{d}) + \frac{M k_p}{d^2}(1 - e^{-\frac{d}{M}\Delta t}) \quad (4.6)$$

for our example. The locations of the system poles in the Z-plane's unit circle determines the system's responses, and are a function of the proportionality constant k_p. The designer often chooses k_p so the system is critically damped because this produces the fastest movement to the goal without any overshoot. For the example, $k_p = 2.1694$ produces a critically damped system, and if $k_p > 24.58$ the system is unstable.

To fill the Q-SAM with the proportional control law, we apply equation 4.3, with $k_p = 2.1694$, to the central value of each quantization region in the Q-SAM and store the calculated actuator responses in their appropriate locations in the Q-SAM. By "central value" we mean the middle value of the error range represented by the quantum in question. For error signals outside the range of the Q-SAM, the force applied to the system is that which corresponds to the end quanta of the Q-SAM. The control surfaces associated with the control law and the Q-SAM implementation of the control law are shown in figure 4.5. Note that the limited range of the Q-SAM limits the magnitude of forces applied to the system when the error signal is outside the range $[-2.0m, 2.0m]$. The unit step response of both control laws is shown in figure 4.6. The responses are similar because the initial error is within the Q-SAM range. The difference in the steady states results from the coarse quantization of the Q-SAM. Increasing the number of quantization levels near the goal region reduces this bias. For a step of $8m$, whose initial error is outside the the Q-SAM range, the force limitations of the Q-SAM cause a slower response, as illustrated in figure 4.7.

This example has illustrated filling a Q-SAM with a proportional control law. This technique can be used with any rule-based control law like those generated using fuzzy logic theory or neural networks.

4.4 Downloading from an Expert

To illustrate some of the unique facilities of Q-SAMs and situation-based control, consider the problem of trying to convert a Remotely Operated Vehicle (ROV) into an AUV. Through years of experience, the ROV operator, an expert, has developed a

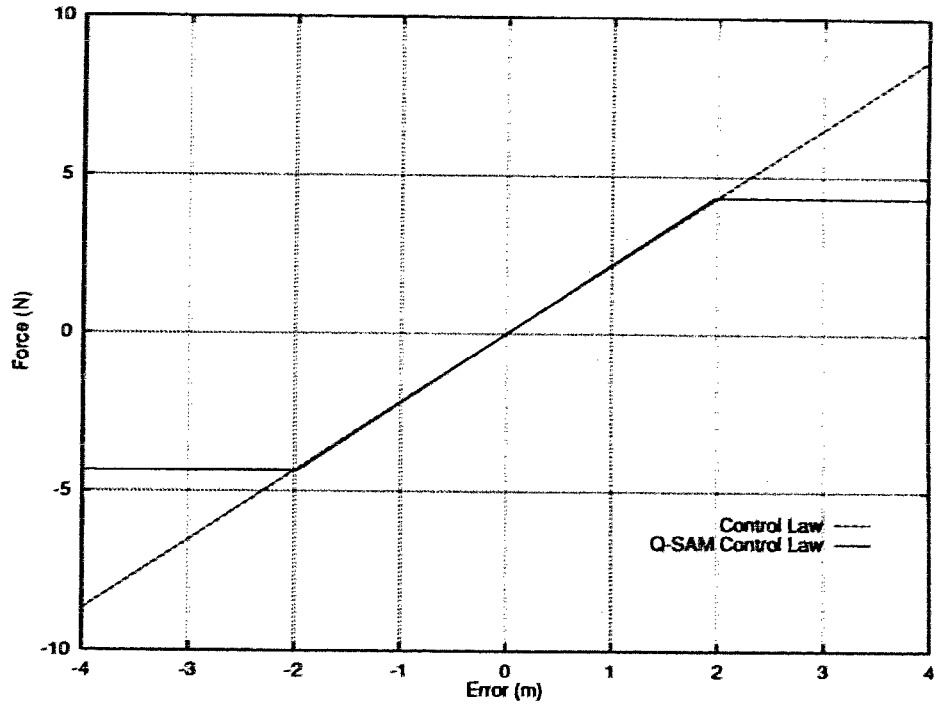


Figure 4.5: Proportional control law and its Q-SAM implementation

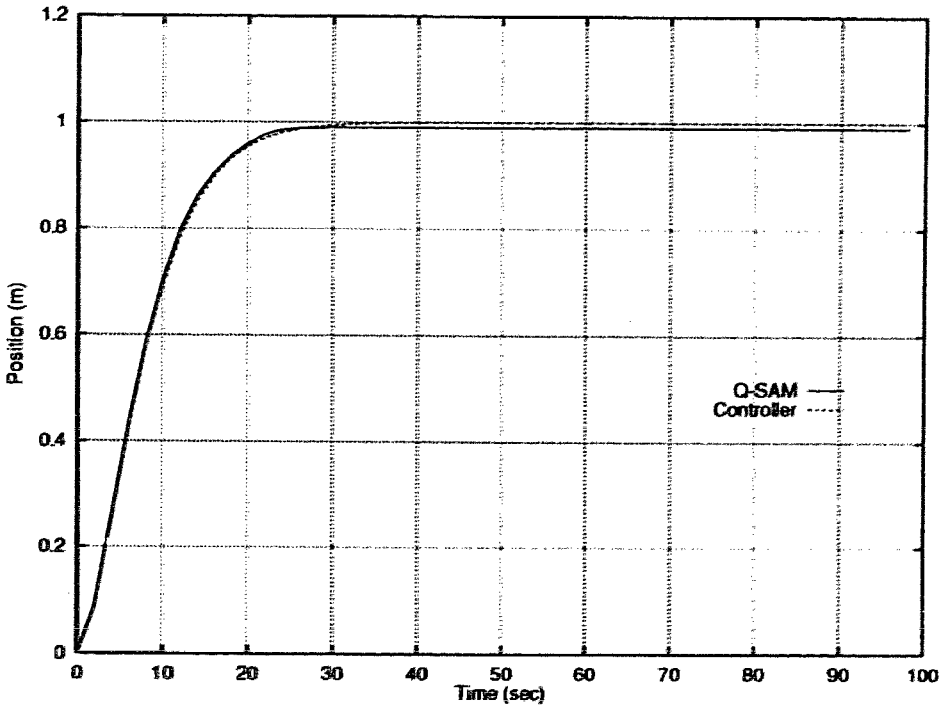


Figure 4.6: Step responses for the Q-SAM and continuous control laws (Step is within Q-SAM range)

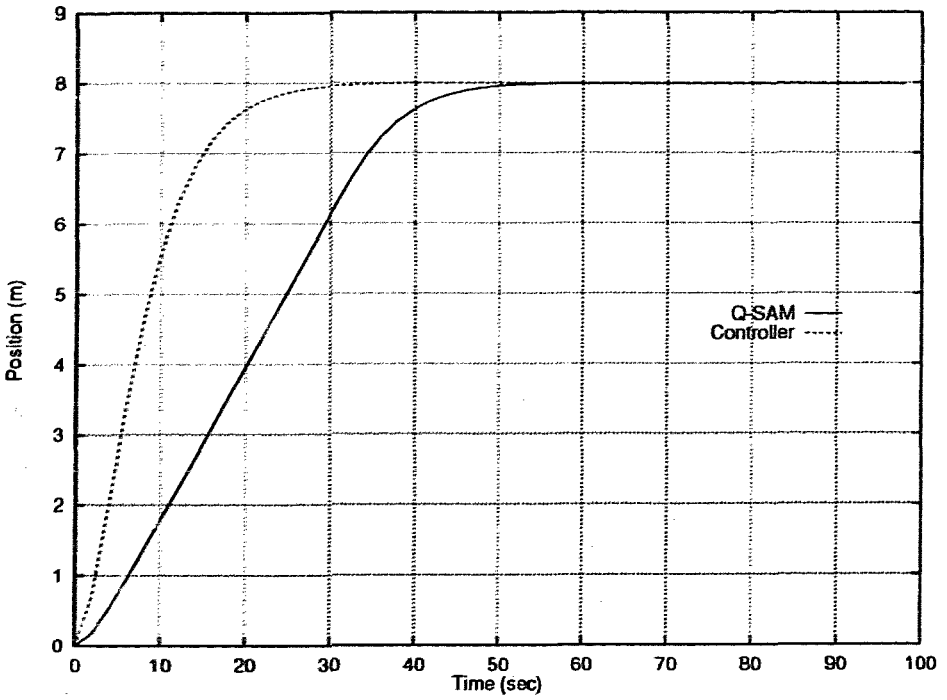


Figure 4.7: Step responses for the Q-SAM and continuous control laws (Step is outside of Q-SAM range)

control law that is sufficient to control the vehicle, however, that control law must be transferred to a computer to make the vehicle autonomous. Unfortunately, operators are generally unable to describe the control laws they have developed in terms that control engineers can exploit to develop computer-based controllers. In this section, we use a technique that we call "downloading" to record the responses of an expert controller into a Q-SAM. The Q-SAM, when implemented as the controller, mimics the responses of the operator, and the vehicle is autonomous. Note that this technique does not require any knowledge about the plant being controlled.

Experiments with the downloading transfer technique were conducted with a graphical simulation of the altitude-keeping task on a Sun workstation. The vehicle moved up and down the screen as a function of the force applied to the system by the operator through the mouse input device. The force applied to the system was proportional to the distance from the mouse to a central location on the mouse pad. The operator's task was to simply move the AUV to a specific location from several different initial locations. To download the operator's control law, the altitude error was sensed at a rate of $1/\Delta t$ Hz while the operator was controlling the vehicle.

The sensed error was used to address locations in the Q-SAM where the concurrently applied force was stored. This process is shown in figure 4.8

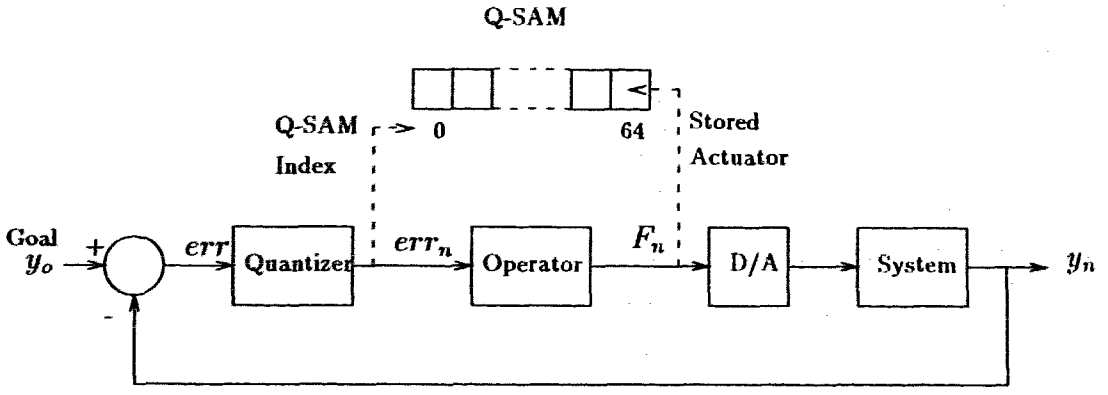
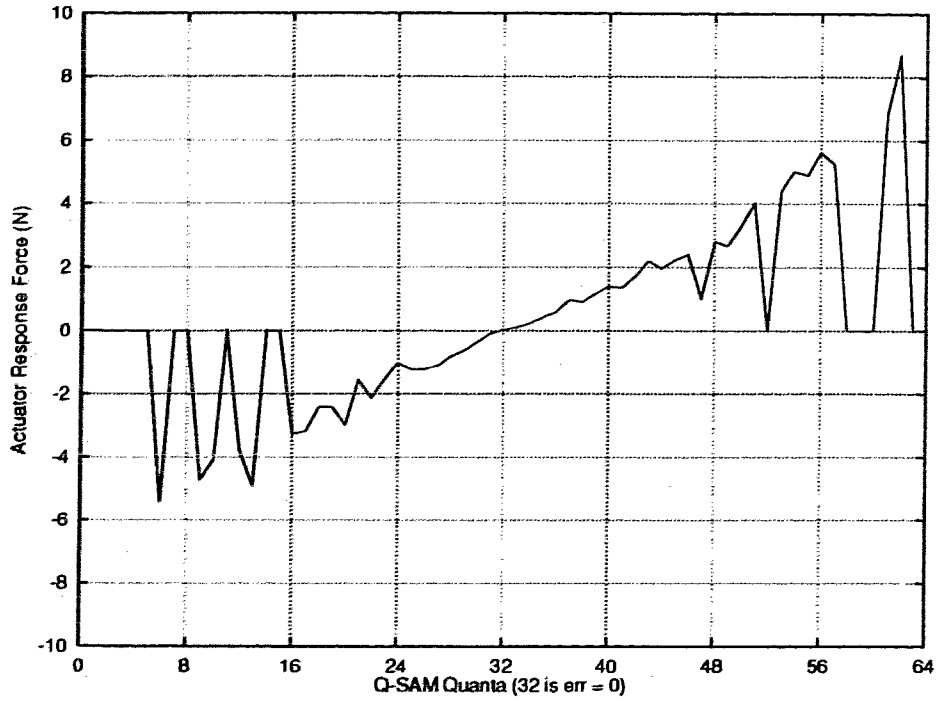


Figure 4.8: Downloading transfer technique

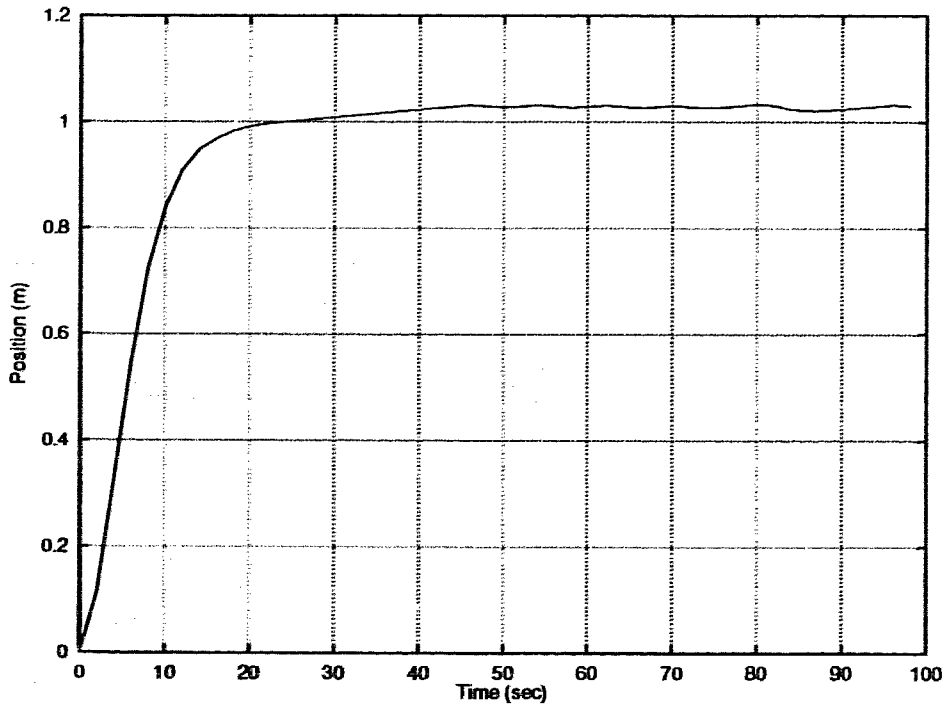
One requirement of this downloading technique is that the sensors must differentiate situations that correspond to the situations differentiated by the expert when the expert controls the vehicle. These situations must also be sufficient for the task at hand. For the altitude-keeping example, the altitude error sensor is sufficient, though vehicle velocity can also be sensed because the expert has some idea of the velocity of the vehicle he is observing. Using a velocity sensor means that sensor space is two dimensional and has one sensor axis for the system error, err_n, and another for the derivative of the error, $\dot{err}_n$.

Figure 4.9 a) shows a control law developed by this transfer technique using 30 randomly chosen initial locations during development. During the downloading process, when a quantum of sensor space is encountered more than once, the newly recorded response is averaged in with all previously recorded responses for that quantum. Though the step response of this technique, shown in figure 4.9 b), is adequate, the control surface is rough and has holes in it. Holes in the control surface are associated with regions of sensor space that are not accessed during downloading and therefore hold the default response of zero. For example, there is a hole at quanta 14 and 15 in figure 4.9 a). If the system is placed in one of these locations with zero velocity, it will not move.

The steady state error in figure 4.9 b) results from the flat section in the control surface near the goal region of $err = 0.0$. This flat section is caused by the expert,



a)



b)

Figure 4.9: Control law and step response from expert transfer technique without actuator response distribution

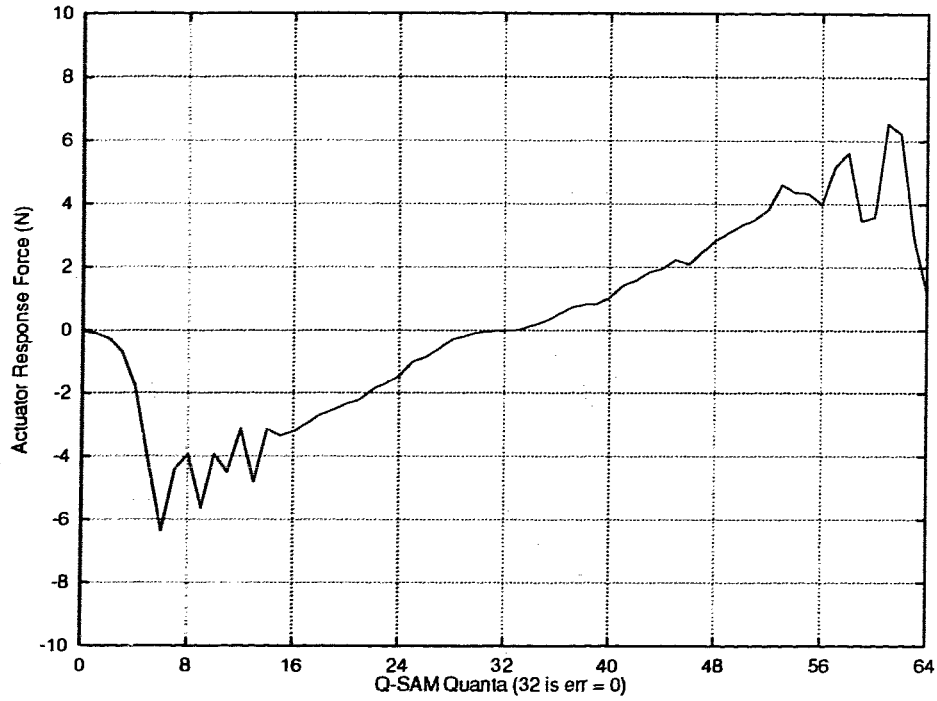
who had difficulty in maintaining the vehicle at the goal region. Consequently, the responses associated with the quanta near the goal region were continually adjusted, causing the flat section.

To reduce the number of holes in the Q-SAM, we can increase the number of randomly chosen initial locations or implement some mechanism that chooses initial locations from the group of sensor space locations that have not yet been accessed. The second approach guarantees no holes in the control surface and is reasonable for our one dimensional example. Unfortunately, it is not reasonable for most multidimensional systems because there are simply too many locations in the Q-SAM.

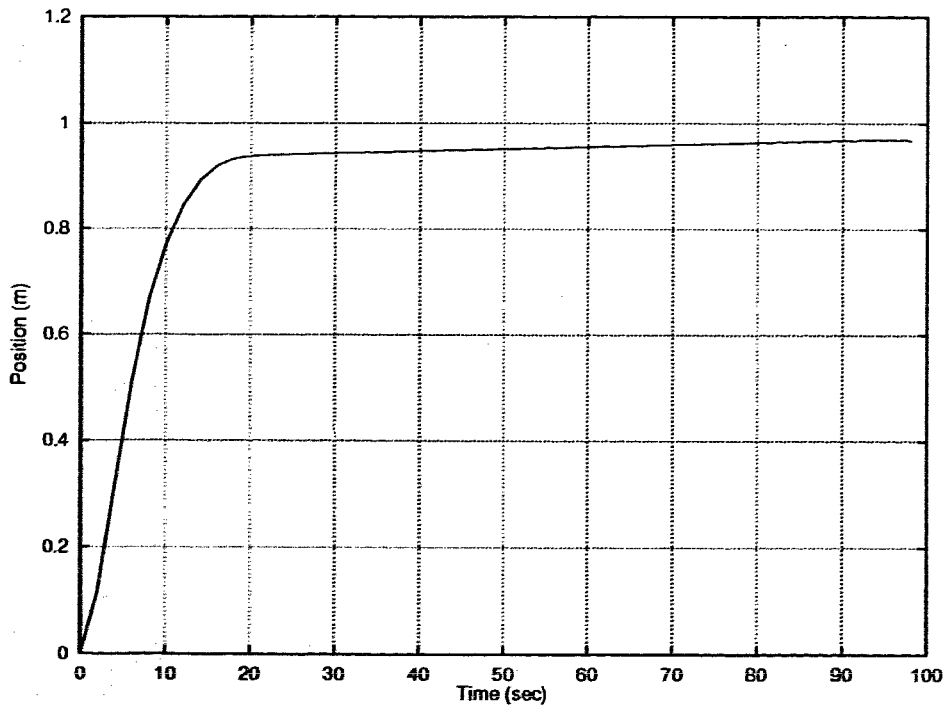
Therefore, we use the actuator response distribution function to fill in the holes during downloading so that the total number of trials is kept small. Since the actuator response of the expert is a desirable response, we average it into its respective quantum in the Q-SAM. Then, this newly averaged actuator response is distributed through the Q-SAM using equations 4.1 and 4.2. In equation 4.1, f_i represents this newly averaged actuator response which is then distributed with parameter b in equation 4.2 always set to 1.0. Parameter b is set smaller than 1.0 when the designer wishes to cumulatively develop a control surface, which is not the case in this section (see section 4.5 for an application with b not set to 1.0).

Figure 4.10 a) shows a control surface developed with the use of actuator response distribution. Again 30 randomly chosen initial locations were used during the downloading process. Parameter c of equation 4.2 was set to 1.0. This value of c causes the ARDF to have less than a 2% effect four sensor space locations away from the newly recorded actuator response, as can be seen in the 1^{st} iteration of figure 4.4. c was chosen to be 1.0 because it provides sufficient actuator response distribution for the example. The other system parameters were the same as the previous example. The resulting control surface is smoother than that of figure 4.9 a) and the holes are significantly reduced. The only locations in this Q-SAM that hold zero values are at the outer limits of sensor space. This scenario cannot be avoided when random locations are chosen during downloading, as we have done, because there is a low probability of randomly choosing the extreme locations of sensor space. However, it is not difficult to force the development phase to include the outer limits of sensor space.

The flat section of the control surface in figure 4.10 a) and the large steady state



a)



b)

Figure 4.10: Control law and step response from expert transfer technique with actuator response distribution

error are again caused by the poor performance of our expert

This experiment shows that it is possible to download control responses from expert controllers whose functionality is unknown and also demonstrates the use of actuator response distribution functions to speed up the Q-SAM filling process by reducing the number of holes in the Q-SAM.

4.5 Adaptive Situation-Based Control

Adaptive, or self-learning, control algorithms are used when neither an expert nor a sufficient system model are available to the control engineer. Situation-based adaptation, specifically adaptation with a Q-SAM implementation of the control law, permits an added degree of flexibility, relative to many adaptive techniques, because the control surface is adapted locally with respect to sensor space. By "local" we mean that adaptation is done one differentiable situation at a time. This means that the control surface can be altered every control period based on the effects of a particular actuator response to a specific differentiable situation. Figure 4.11 a) shows changes

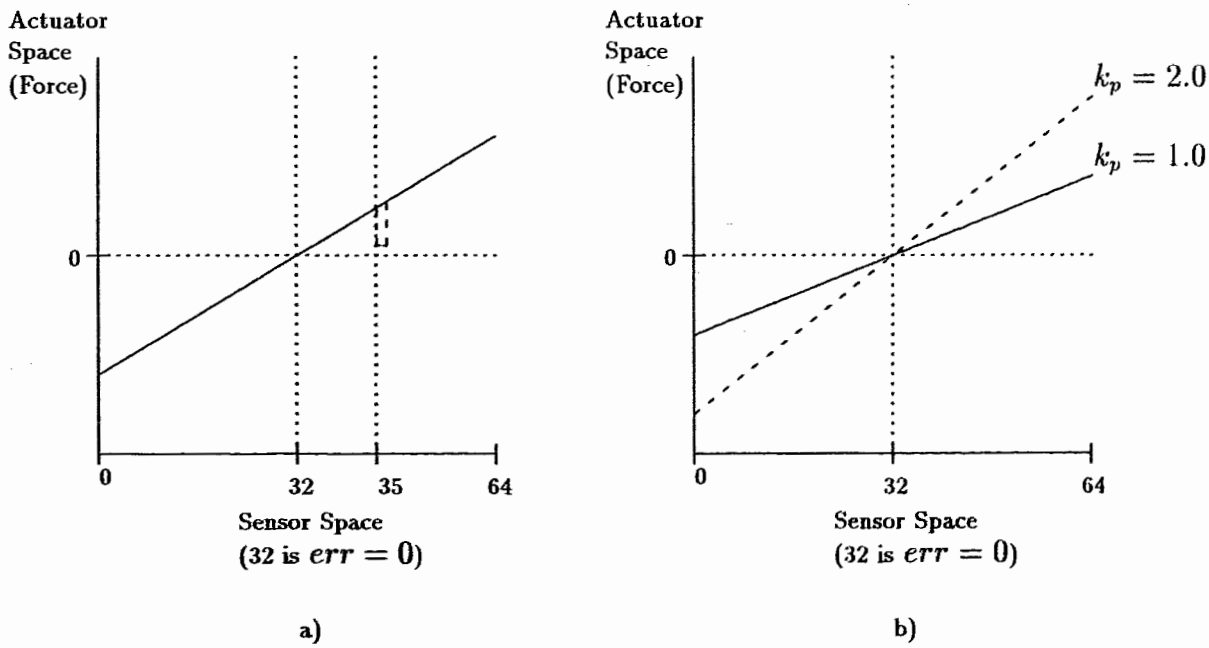


Figure 4.11: Comparison of local and global adaptation

in a control surface caused by altering the response associated with one situation,

specifically the situation associated with quantum 35 in the figure. Relative to sensor space, most adaptive techniques are global in nature and require several control periods of analysis prior to adaptation. By "global" we mean that adaptation alters the actuator responses associated with many differentiable situations. An example of a globally adaptive control law is an adaptive proportional controller. When the parameters of a proportional controller are adjusted, the actuator responses associated with most situations (all but the situation associated with zero error) are altered. Figure 4.11 b) shows the changes in a control surface caused by changing the constant of proportionality, k_p, of a proportional control law from 1.0 to 2.0. Most responses in the Q-SAM are changed. Examples of global analysis techniques are those associated with system step responses because their analysis requires several control periods, and consequently involves the actuator responses associated with several situations.

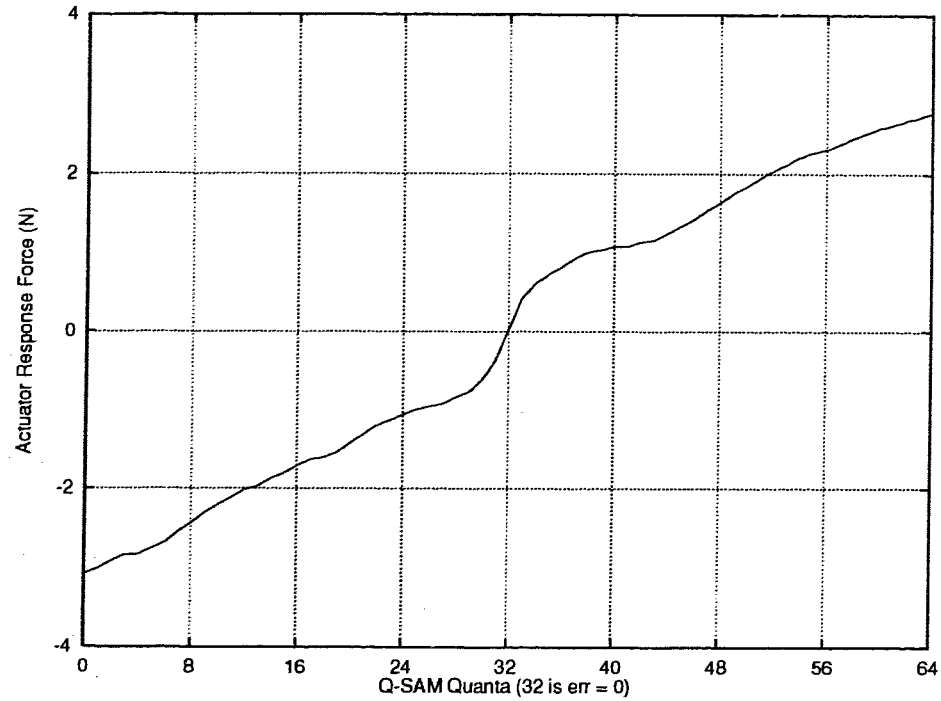
There are several reasons to examine situation-based adaptation. First, when the control environment is under-sensed, global adaptation is not possible because global analysis techniques compare their results relative to a standard, which in the case of an under-sensed control environment, cannot be sensed. For example, there is no known standard for navigating through an unknown obstacle field, and the vehicle's performance cannot be measured using only the vehicle's sensor values. Also, AVs generally have only a single mission that is usually very long, which means that global analysis techniques are of little value because they provide analysis after the vehicle has moved to the goal region. Finally, with situation-based adaptation, the general form of the control surface does not need to be specified a priori as it does with global adaptive techniques. For example, the use of an adaptive proportional controller presupposes that a control surface which is a flat plane in sensor space is desirable, as is shown in figure 4.11 b).

The adaptive algorithm we use in this section operates as follows. The Q-SAM is initially empty (ie. filled with zeros). The vehicle assigns a new altitude every 20 control periods. The new altitude is within a $4m$ window of the previous altitude assignment so that the system is operating within the $\pm 2m$ Q-SAM range during adaptation. Every control period, a new actuator response is generated by taking the appropriate response, based on the present error, from the Q-SAM and adding to it a random perturbation which is uniformly distributed in the range $[-2.0N, 2.0N]$. The random perturbation is the system's way of experimenting with new responses.

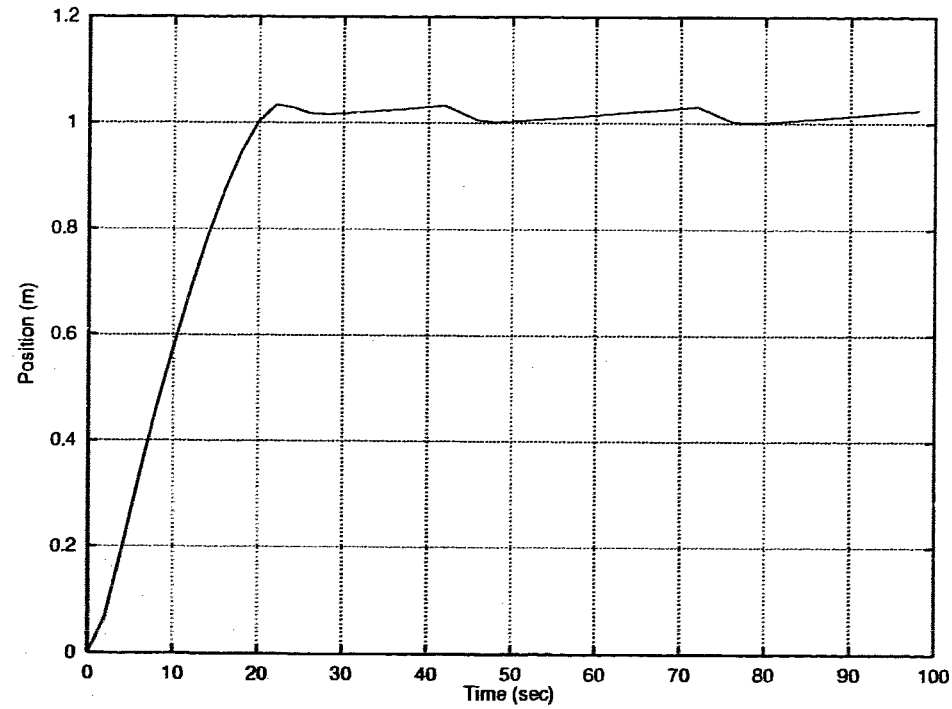
The effects of this new response are analysed after it has been applied for one control period. If the new response moved the system closer to the goal, but not too much closer, then it is distributed through the Q-SAM using the actuator response distribution function described in equations 4.1 and 4.2, which is how the system adapts. f_i in equation 4.1 is the successful new actuator response and parameter b of equation 4.2 controls the influence f_i has on the response presently stored in quantum i and what portion of f_i is distributed through the Q-SAM. This adaptive algorithm cumulatively generates a control surface.

Figure 4.12 a) shows a control law generated using this technique and running the model through 100 000 new altitude assignments. 100 000 altitude assignments are required because the Q-SAM is initially empty. If the Q-SAM is initially filled with a functional control law, then the number of iterations can be reduced. This reduction depends on the initial control law in the Q-SAM, though for most reasonable control surfaces 10 000 altitude assignments are required. The ARDF parameters are $b = 0.01$ and $c = 0.5$. That is, the newly learned force is averaged in with a 1% effect on the quantum with which it is associated and it is distributed so that it has less than a 0.02% effect eight sensor space locations from the quantum with which it is associated. These parameters were chosen because they provide a relatively stable control surface. By "stable" we mean that the control surface does not change drastically after the control surface has reached steady state. With this adaptation algorithm, the steady state control surface is continually changing during adaptation, though its deviation is probabalistically bounded. That is, there is a variance (σ) associated with the steady state control surface, as shown in figure 4.13. Increasing b increases the variance in the steady state control surface but, the steady state is approached much faster. If c is decreased, variance in the steady state control surface is reduced, but so is the magnitude of the surface.

The parameters b and c are sensitive to the number of elements in the Q-SAM. Consequently, their values should be determined relative to sensor space distances and not indexed quantization regions. For this system, any value of $b < 0.1$ will generate a relatively stable control surface. It should be noted that unless extreme values of b and c are chosen, this algorithm generates an adequate control surface. For figure 4.12, "Not too close" is defined as 25%. In other words, commands are distributed into the Q-SAM if they move the system up to, but not more than 25%



a)



b)

Figure 4.12: Control law and step response from adaptation algorithm (25% rule)

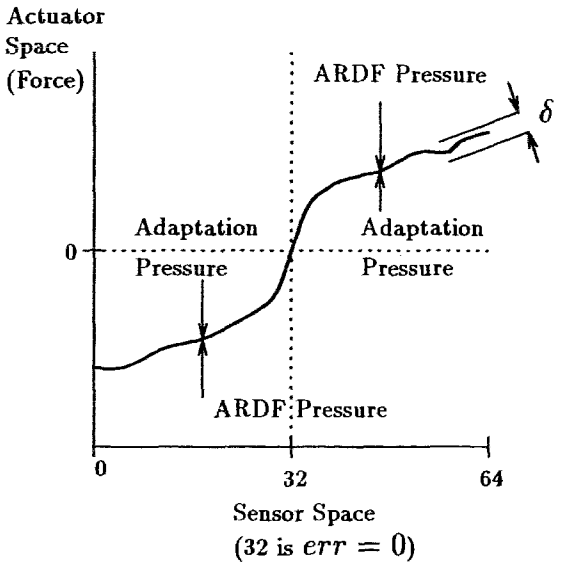
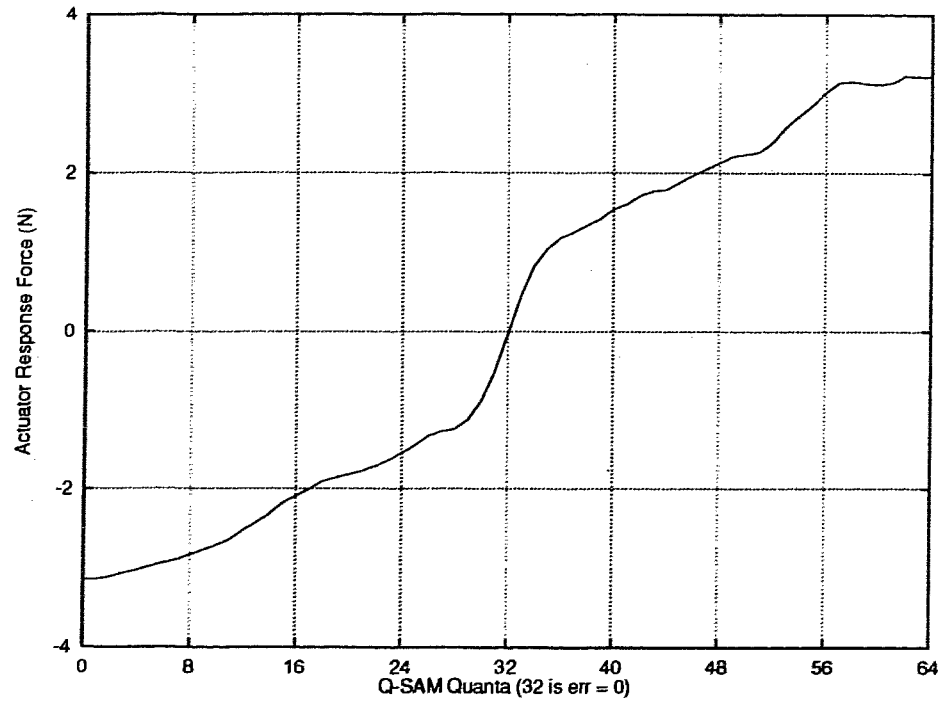


Figure 4.13: Parameters associated with the adapted control surface

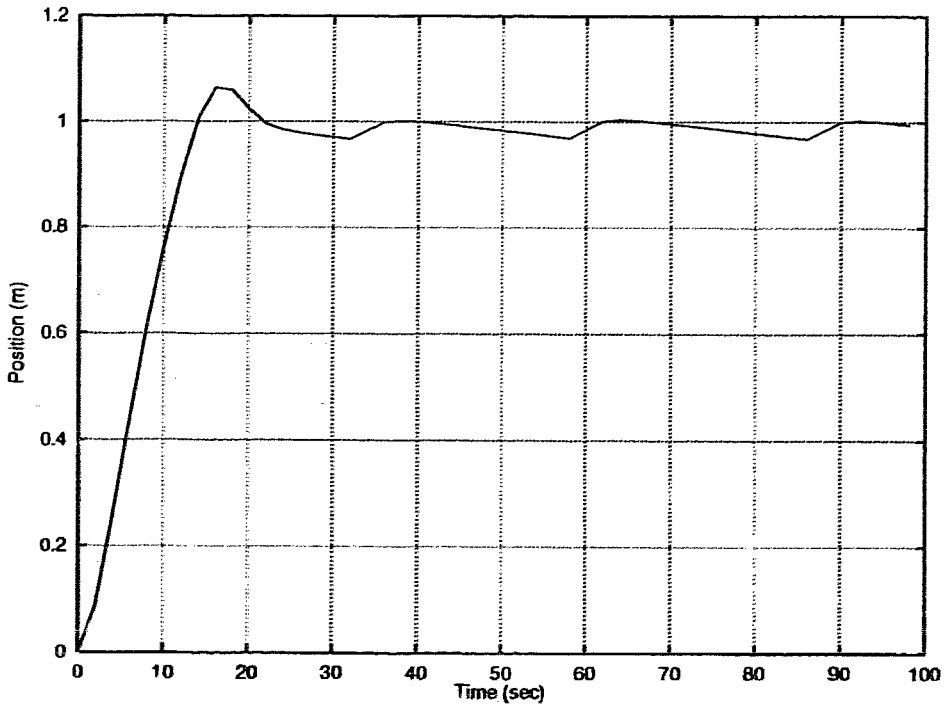
closer to the goal after one control period.

The 25% rule corresponds to a relatively slow approach to the goal. This can be seen by the step response of figure 4.12 b) that requires 20s to reach the goal. Figure 4.14 a) illustrates the control surface and figure 4.14 b) the step response of the system with "not too close" being defined as 75%. This control surface moves the system to the goal in 13s but has some overshoot. The roughness of the steady state portion of both step responses results from the low quantization of the Q-SAM. When the number of quantization levels is increased near the goal region, the steady state portion of the step responses is smooth. The drooping at the end of the control surface in figure 4.14 a) results from a combination of control surface deviation, and the system not yet having fully exercised the far reaches of the sensor space.

The control surface generated by this algorithm is a balance between two opposing pressures. The adaptation algorithm tends to increase the magnitude of the response stored in each quantum. The strength of this increase decreases with the magnitude of the response. The ARDF tends to decrease the magnitude of each quantum by trying to make all the responses equal, in this case equal to zero. The greater the difference between nearby quanta the larger the effective decrease. The steep section of the control surface near the origin ($err = 0$) is an artifact of the random perturbations continually being applied to the system, which cause the system to oscillate around the



a)



b)

Figure 4.14: Control law and step response from adaptation algorithm (75% rule)

goal region during adaptation. This oscillation reduces the average velocity of system near the goal region, which requires a proportionately higher force, on average, to compensate. A more detailed explanation of the shape of the control surface is given in appendix D.

This example shows that situation-based adaptation is possible though it is not as practical as initially desired. For the simple altitude-keeping task, at least 10 000 altitude assignments, though 100 000 is preferable, are required to generate a control surface that resembles the steady state control surface. After 200 altitude assignments, the control surface is generally very poor. Also, as mentioned earlier, we are interested in adapting the control surface during a single movement to the goal region, which is a single altitude assignment for the example. This experiment suggests that adaptive control, locally or globally, is not possible in a single movement towards the goal region which, in retrospect, is reasonable. Therefore, adaptive control techniques can only be used on autonomous vehicles when the vehicle moves to a goal region many times during a mission.

There are several positive results obtained from this experiment. First, for situation-based adaptive control, actuator response distribution functions are required to give the control surface form. When ARDFs are not used during adaptation, the actuator values stored in each sensor space quantum vary in a bounded random walk. That is, the responses stored in each quantum vary independent of the values stored in any other quantum. This results in the entire control surface having no steady state form. Despite this fact, the control surface generated without actuator response distribution is adequate because it moves the system to the goal region. Finally, this experiment has provided further evidence that the Q-SAM representation supports any form of control law because the control law generated by this algorithm has a very unique shape.

4.6 Implications of Control Law Downloading

At this point, we diverge from the engineering discussion to consider some of the philosophical implications of the Q-SAM representation and the ability to download control laws to that representation. This discussion is the opinion of the author and is used to clarify the disparity between man and machine. Imagine that someone designs

an expert system to control an AUV, but the expert system is too slow to calculate actuator responses in sufficient time to respond to the dynamic world in which the AUV exists. As a solution to this dilemma, they implement a Q-SAM on the AUV with the expert system operating above the Q-SAM, as shown in figure 4.15. The Q-

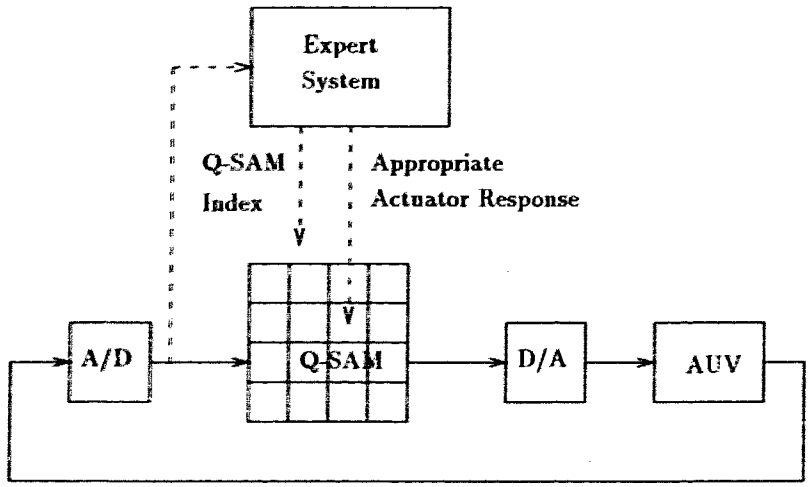


Figure 4.15: Downloading from an expert system

SAM is filled with random responses when the AUV is released into the environment. When the expert system is able, it generates responses and places them in their correct location in the Q-SAM. Initially, control is poor and the vehicle thrashes around the environment. But, as time goes on, the Q-SAM gradually takes on the control law generated by the expert system. Eventually, the Q-SAM mimics the expert system and provides the control that the expert system would have provided had it been fast enough to operate in the world.

The situation we have just described is one where the intelligence of a person is used to generate a control law in the form of an expert system. The expert system then passes that control law to a Q-SAM, whose form is a sensor to actuator mapping. The understanding of the environment resides only in the person and the law, chosen by them, is equivalently represented in both the expert system and the Q-SAM. Neither the expert system nor the Q-SAM have any understanding of their environment. They simply react as they were programmed to react.

This idea can be further extended by considering that the person who developed the expert system might have learned the rules from another person in rote fashion

and that the other person might have been given the rules from a predecessor in their field. In fact, the rules implemented in the Q-SAM might have been handed down from as far back as the beginning of time. This raises an important issue: Where is the true intelligence, and understanding, of the situations differentiated in sensor space to which the Q-SAM applies responses? If we assume that none of the people who learned the rules understood them, then either the rules were present at the beginning of time, which is unlikely, or the rules were developed by random chance, which is also unlikely, especially for tasks that are a mere convenience in the society of today.

Therefore, some of the people along the way must have understood what they were doing. It is this group of people that initially created the rules and later modified the rules. In the limit, only the creators of the rules required an understanding of what they were doing. Note that many of the people along the line did not need to understand what they were doing for the rules to work or to repeat the rules to their descendents.

This discussion has focussed on several ideas. First, understanding is not required to apply rules and appear intelligent. Secondly, on many occasions people are not necessarily exhibiting the intelligence they possess when they are performing tasks which appear to require intelligence. Finally, computers performing tasks that appear intelligent are only applying rules in rote fashion, and simply appear intelligent: they are not intelligent and do not understand the environment in which they exist.

4.7 Summary

In this chapter, we explore the potential of situation-based control using the Q-SAM representation of control laws. The Q-SAM representation of the control law m accepts many forms of control laws because of its unique ability to handle each situation in the environment individually. First, we fill the Q-SAM with a proportional control law to show that Q-SAMs are equivalent to rule-based control systems. Second, we download the control law of an expert controller, whose functionality is unknown, by recording the actuator response of the expert while s/he is controlling an ROV. One requirement of the downloading procedure is that situations used by the expert must

be differentiated in sensor space. Our experiments with adaptive situation based control found that adaptation during a single movement to the goal region is not, as yet, practical. Through the course of experimentation, we required the use of an actuator response distribution function to increase the rate at which the Q-SAM is filled and to provide form to the control surfaces developed adaptively. Finally, an analysis of the downloading procedure has suggested that people do not use their intelligence for every task they perform. Indeed, many people perform tasks in rote fashion, which is the only method by which computer-controlled equipment can perform tasks.

4.8 Discussion

In this section, we discuss two opinions not discussed in this chapter. First, using actuator response distribution functions when filling Q-SAMs with control laws has a smoothing effect on the control surface that is similar to the smoothing effects on control surfaces generated by control laws that are implemented with fuzzy logic (Smith and Comer 1991). Secondly, when recording a human expert, we observed the fact that humans have a very small control period when controlling equipment. Though we used a control period of $2.0s$, we could have explored the effects of decreasing the control period during recording. It should be noted that Q-SAMs can have the shortest control period of any digital control law implementation.

Chapter 5

Design of an Autonomous Underwater Vehicle

This chapter describes a design methodology based on situation identification and differentiation that results in vehicles that are robust to disturbances in their environments. By "robust" we mean that the performance of the vehicle degrades in a predictable manner that is proportional to the size of the disturbance. By "disturbance" we mean changes in the environment for which the vehicle was not specifically designed or cannot sense. The design methodology has three phases. In the first phase, experiments are conducted with the vehicle to identify the meaning of each differentiable situation in sensor space. In the second phase, the entire vehicle is designed using only recently sensed sensor values because systems developed in this fashion can be robust to disturbances the their environment. By "recently sensed sensor values" we mean the most recently acquired sensor values. In the third phase, internal representations are added to the second phase system to augment system performance without making the system brittle.

One aspect of this methodology is that sensors are developed together with vehicle dynamics. In the case of AUVs, this means that sensors must be designed for specific vehicles because vehicle dynamics are generally fixed for specific environments and missions. This contrasts with the traditional approach of choosing sensors a priori and expecting design engineers to develop sufficient control systems to accomplish the desired task. The traditional approach is acceptable for critically sensed control

environments, but is unacceptable when the control environment is under sensed because the meanings of differentiable situations are a function of the entire autonomous vehicle control cycle.

In this chapter, we use our design methodology to determine the sensor requirements and control mapping for an AUV that uses forward-looking sonars to avoid obstacles in an unknown obstacle field. The work supporting this chapter has brought to light the fact that the definitions associated with differentiable regions in sensor space are a function of the entire system (ie. m, g and h in figure 3.1) when the control environment is under-sensed.

In section 5.1, we describe the vehicle that we will use to illustrate our design methodology, which we describe in section 5.2 along with the motivation for the methodology. In section 5.3, we describe the experimentation phase of the design methodology for the vehicle. In section 5.4, we describe the second phase of the design methodology as it pertains to the vehicle, and show that the vehicle is robust to disturbances in its environment. In section 5.5, we describe the third phase of the methodology with respect to the vehicle, and show the performance improvements obtained by using internal representations. In section 5.6, we describe an improperly designed vehicle operating in an under-sensed control environment and show the difficulty in determining system brittleness. In section 5.7, we summarize the significant results presented in this chapter and in section 5.8 we provide a general discussion of those results.

5.1 Vehicle

The vehicle used in this chapter is a two-dimensional version of the torpedo-shaped AUV under development at International Submarine Engineering Research. The vehicle's attitude is controlled by planes and it has a single, rear-mounted thruster for propulsion. The vehicle travels at $1.9m/s$ and has a maximum turning rate of $0.9599rad/s$ ($5.5deg/s$), which corresponds to a minimum turning radius of $19.8m$. The vehicle is $4m$ long and has a clearance of $3.5m$, which means that we do not wish any obstacles to be closer than $3.5m$ from the vehicle's centreline. The vehicle is equipped with two sonars, mounted $2.0m$ in front of the vehicle's centre point, both facing forward with one facing to the left and the other to the right. Each sonar

returns a value that corresponds to the distance to the nearest object in the sonar's beam pattern.

The task of the vehicle is to transit to a distant location, called the endpoint (x_g, y_g), without colliding with any objects. Therefore, the vehicle's primary subgoal is obstacle avoidance. An on-board positioning system is used to determine the position (x_v, y_v) and orientation (θ_v) of the vehicle relative to the world. The vehicle is placed in an environment for which there is no previously recorded knowledge. This vehicle is shown in figure 5.1.

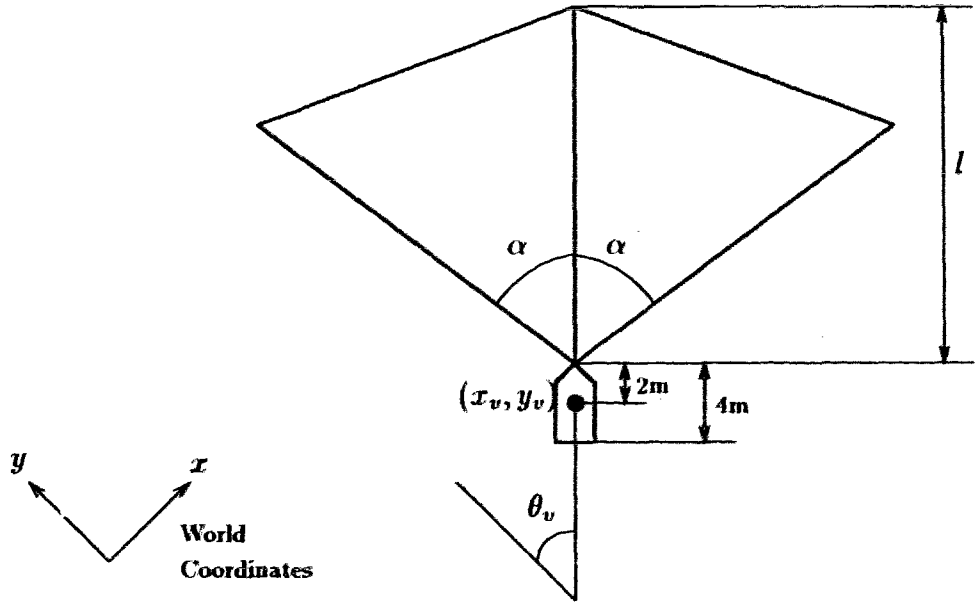


Figure 5.1: Autonomous underwater vehicle

For reasons of simplicity and to prevent clouding the issue with mathematics, the beam patterns of the sonars used by the vehicle are assumed to be isosceles triangles. The unequal angle of the triangle is the beam width (α) of the sonar beam pattern and is attached to the front of the vehicle. The beam length (l) is the distance in front of the vehicle that the sonar beam pattern extends. Describing sonar beam patterns in this manner preserves their function and therefore the essence of the problem. The beam patterns are also shown in figure 5.1.

The control system for the vehicle is shown in figure 5.2. It is the same diagram as figure 2.1 updated with the particulars of this vehicle. The low-level controller

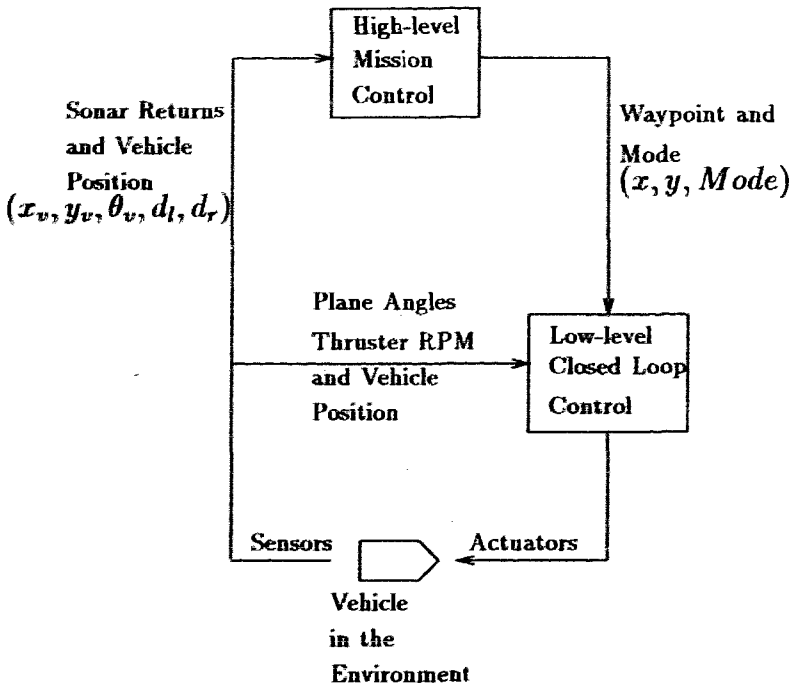


Figure 5.2: Autonomous underwater vehicle controller

is designed using feedback control theory. In this chapter, we develop the high-level controller whose actuator responses are waypoints and turning commands for the low-level controller. The actuator space for the high-level controller has the form $(x, y, Mode)$ where (x, y) is the position of the next waypoint in the world, and $(Mode)$ can hold one of three values: 0 meaning hard left turn, 1 meaning drive towards the specified waypoint, and 2 meaning hard right turn.

The sensor space for this vehicle is represented by the sensor state vector $(x_v, y_v, \theta_v, d_l, d_r)$ where (x_v, y_v, θ_v) are the position and orientation of the vehicle in the world and d_l and d_r are the distances to the nearest object in the left and right sonar beam patterns respectively. $(d_l, d_r) = (\infty, \infty)$ represents the situation that no objects are in either sonar beam pattern. The sensor and actuator spaces as well as the control mapping are illustrated in figure 5.3.

The experiments with this vehicle were conducted with a graphical simulation of the AUV on a Sun workstation.

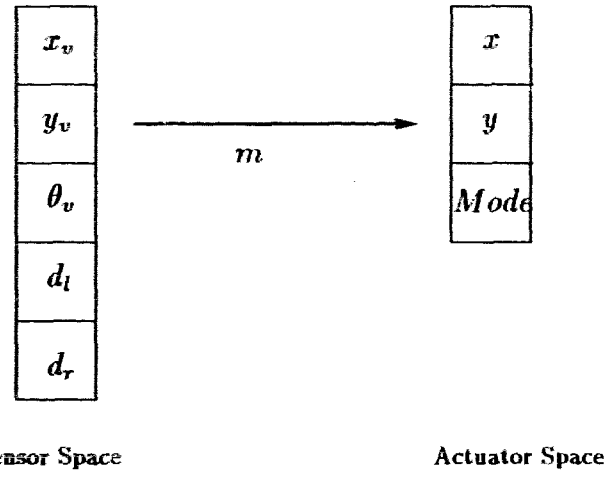


Figure 5.3: Control mapping with sensor space and actuator space

5.2 Design Methodology

The development of autonomous vehicles and their control systems is broken into three phases. Each phase represents one step in an incremental process of developing vehicles as a whole by identifying the situations differentiated in sensor space. The three phases of autonomous vehicle development are:

I Experimentation

II Robust Controller Development

III Internal Representation Development

The first phase of development, Experimentation, provides the designer with an understanding of the vehicle's capabilities and an idea of what situations particular sensors can differentiate. More specifically, the phase I experiments are conducted to gain insight into the identification of each equivalence class of the sensor space quantization partition P_Q. This phase of development is required by systems operating in under-sensed control environments because the meaning associated with each quantization equivalence class is generally not obvious since the meanings are a function of the entire vehicle. The control and actuator mappings, m and h, must be similar, or identical, to those to be used in phase II to ensure that appropriate meanings are assigned to each quantization equivalence class.

Using the knowledge obtained in phase I, the second phase of development, Robust Controller Development, results in a complete autonomous vehicle whose actuator responses are determined from only recently sensed sensor values. Using recently sensed sensor values to determine actuator responses means that the system has the potential to be robust to disturbances in its environment because the system is continually sensing that environment. In terms of sensor actuator mapping theory, in phase II the sensor, actuator and control mappings (systems) are designed and each equivalence class of P_Q is clearly defined. The actuator partition P_A, which is a function of the control mapping m, is designed so that P_Q is consistent with P_A. The system's goal and subgoal regions are defined and incorporated into the mapping m. This phase develops a system with complete sensor and actuator spaces.

The system developed in phase II is robust to disturbances in its environment. In phase III, internal representations, which are brittle, are added to the phase II system. Therefore, much effort should be expended in phase II to make the vehicle as functional as possible so that the final system is as robust as possible. In short, the system developed in phase II defines the system's robust functionality.

In the third phase, Internal Representation Development, internal representations are added to the phase II vehicle to augment vehicle functionality. Internal representations are necessary if the vehicle is to perform tasks which are more complex than simply bumbling around the environment like Allen (Brooks 1986). The internal representations are added in such a way that the brittleness with which they are associated does not affect the robustness of the system. This task is accomplished by not altering the control mapping associated with regions of sensor space deemed critical. For our vehicle, critical regions of sensor space are those associated with the presence of obstacles. That is, internal representations are not used to control the vehicle when sensors sense obstacles. Adding internal representations in this manner is in concordance with the philosophy of the subsumption architecture (Brooks 1986) in the sense that higher-level functions, those obtained with internal representations, cannot adversely affect the survival of the system.

It is important to note that each phase of development in this methodology requires experimentation with the vehicle to validate designers' beliefs and hypotheses. Requiring experimentation in each phase is consistent with Brooks' physical grounding hypothesis (Brooks 1991) which states that systems must be connected to real

sensors and real actuators if they are ever to function reliably in a real world.

5.3 Phase I: Experimentation

For initial experimentation with the vehicle, consider using sonar beam patterns which are each 10° wide and $30m$ long. The front of these beam patterns are each more than $5m$ wide and therefore they should be able to sense a $10m$ wide path in front of the vehicle. Since the primary vehicle goal is safety, the vehicle will turn away from obstacles when they are sensed and towards the endpoint when they are not sensed. For this task, the sensor space quantization partition has four coarse equivalence classes with which it is associated. They are $(d_l, d_r) = \{(\infty, \infty), (\infty', \infty), (\infty, \infty'), (\infty', \infty')\}$ where ∞ corresponds to the absence of obstacles in the respective sonar beam pattern and ∞', meaning "not infinity", corresponds to the presence of at least one obstacle in the respective sonar beam pattern.

A reasonable control mapping for this sensor space quantization partition is shown in table 5.1 where (x_g, y_g) are part of the control mapping and "$-$" means irrelevant.

Quantization Equivalence Class (d_l, d_r)	Actuator Response $(x, y, Mode)$
(∞, ∞)	$(x_g, y_g, 1)$
(∞, ∞')	$(-, -, 0)$
(∞', ∞)	$(-, -, 2)$
(∞', ∞')	$(-, -, 2)$

Table 5.1: Control mapping for sensor space quantization partition

This control mapping causes the vehicle to drive towards the endpoint when no obstacles are sensed and to turn left when obstacles are sensed in only the right sonar beam pattern, and to turn right when obstacles are sensed in only the left sonar beam pattern. When obstacles are sensed in both beam patterns, the vehicle arbitrarily turns right. The region $(d_l, d_r) = (\infty, \infty)$ corresponds to the primary subgoal of the vehicle whose goal region in sensor space is $(x, y, d_l, d_r) = (x_g, y_g, \infty, \infty)$.

When the vehicle is placed in the environment with an obstacle between the endpoint and the vehicle, the following scenario occurs. The vehicle moves toward the endpoint location until the sonar beam patterns move into the obstacle. Then, the controller applies the Mode 2 command, which turns the vehicle to the right until both sonar beam patterns move away from the obstacle. No longer sensing the obstacle, the vehicle moves toward the endpoint again by applying the Mode 1 command which, in this case, turns the vehicle left toward the endpoint location and also toward the obstacle. Consequently, the left sonar beam pattern moves into the obstacle which causes the vehicle to turn right again. This cycle continues until the vehicle collides with, or just scrapes by, the obstacle. The net effect of this scenario is that the vehicle moves the left edge of the left sonar beam pattern down the right edge of the obstacle until the vehicle is incapable of turning the beam pattern back into the obstacle.

The scenario just described is behaviour fusion, which was discussed in section 2.3. In this case, the behaviours being combined are move towards the endpoint and move away from obstacles.

5.4 Phase II: Robust Controller Development

The behaviour fusion noted in the previous section results from the vehicle's sensors and the under-sensed control environment in which the vehicle exists. To incorporate this knowledge into the vehicle's design means that the beam patterns must be designed so that the vehicle cannot collide with obstacles that the edges of its beam patterns move along. This means that the beam patterns must be wide enough so that motion of the obstacle, relative to the vehicle, cannot violate the system's clearance requirements after moving down the edge of the beam pattern. The minimum beam width that satisfies this requirement is

$$\alpha = \arcsin(\frac{d}{b}) \tag{5.1}$$

where α is the width of each sonar beam pattern, d is the distance the sonars are placed in front of the centre of the vehicle and b is one solution to the equation

$$c(2r-c)b^2 + 2d^2(r-c)b - d^2(r^2+d^2) = 0 \tag{5.2}$$

where c is the vehicle's clearance and r is the minimum turning radius of the vehicle. The derivation of this beam width is given in appendix E.

With sonars of beam width α in the previous scenario, the vehicle moves the left edge of the left sonar beam pattern down the right edge of the obstacle until the release point (R_l), shown in figure 5.4, reaches the obstacle. The release point is

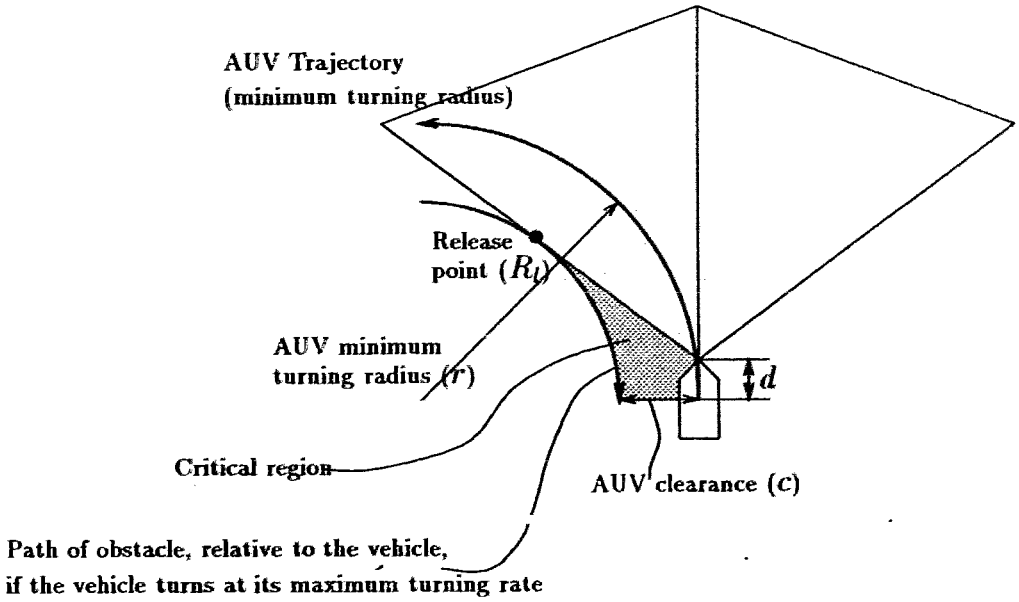


Figure 5.4: Sensor critical region

the nearest intersection, to the vehicle, of the edge of the sonar beam pattern and the circle inscribed by the vehicle's minimum clearance when the vehicle is driven at its maximum turning rate. The significance of the release point is that when a point obstacle is at the release point and the vehicle turns at its maximum rate toward the obstacle, the vehicle will miss the obstacle by a distance equal to the defined vehicle clearance. In a vehicle-based coordinate system, with the origin at the vehicle's turning centre, the positive y-axis extending forward from the origin and the positive x-axis extending to the right of the vehicle in figure 5.4, the left release point is

$$R_l = (-r + (r - c)\cos\alpha, d + (r - c)\sin\alpha) \tag{5.3}$$

and the right release point is

$$R_r = (r - (r - c)\cos\alpha, d + (r - c)\sin\alpha). \tag{5.4}$$

Once the release point of the left sonar beam pattern contacts the obstacle, the vehicle is incapable of turning into the obstacle. In fact, the vehicle is also incapable of turning

the left sonar beam pattern back into the obstacle. Therefore, associated with α and R_l is a critical region, defined by the shaded area in figure 5.4, that cannot contact an obstacle. There is also a critical region associated with α and R_r. If the critical region contacts an obstacle then the vehicle can pass the obstacle by a distance that is less than the defined clearance and therefore may collide with the obstacle without ever sensing it.

Assuming no objects are initially present in the critical region, the meaning associated with each coarse equivalence class of the sensor space quantization partition, as determined by α, r, c and d, are summarized in table 5.2.

Equivalence Class (d_l, d_r)	Situation Description
(∞, ∞)	There are no objects in the world with which the vehicle can collide before they are detected
(∞, ∞')	There is an object ahead of and to the right of the vehicle with which the vehicle can collide
(∞', ∞)	There is an object ahead of and to the left of the vehicle with which the vehicle can collide
(∞', ∞')	There are objects ahead of and on both sides of the vehicle with which the vehicle can collide

Table 5.2: Meanings associated with the coarse sensor space quantization equivalence classes

To guarantee that the critical regions do not contact any obstacles requires that obstacles are sensed in sufficient time to turn the critical regions away from the obstacles. That is, the beam patterns must leave the obstacles ahead of the release points. For the vehicle, this fact means that assumptions about the type of obstacles in the environment must be made. If we assume that the most complex obstacle in the environment is a flat wall that can be detected from any angle then the minimum required length of the sonar beam patterns to guarantee that the critical regions do not contact an obstacle is

$$l = 2r - d - (r - c)\cos\alpha \tag{5.5}$$

where l is the length of each sonar beam pattern and b is the same solution to equation 5.2 that was used in equation 5.1. The derivation of this equation is also given in appendix E.

For a flat wall environment, situations requiring differentiation by the control mapping are those pertaining to the slant of the wall relative to the vehicle. These situations can be differentiated with the values returned by the sonar sensors. If the vehicle encounters a right-slanted wall, as shown in figure 5.5, then $d_l > d_r$ and if the

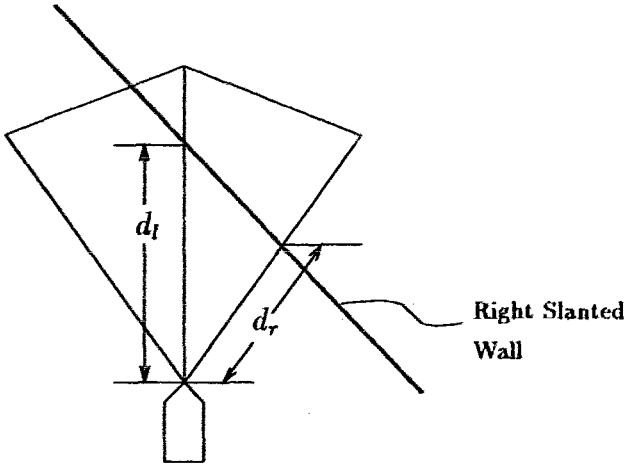


Figure 5.5: Vehicle encountering a right slanted wall

vehicle encounters a left-slanted wall then $d_l < d_r$. If the vehicle encounters a wall head-on, the sonar returns are equal, $d_l = d_r$.

With this knowledge we can construct the control mapping and actuator partion P_A appropriately, based on the sensor mapping defined by α and l. The equivalence classes of the actuator partition and the control mapping are summarized in table 5.3.

It should be noted that the sensor space of the system we have just developed is not complete as defined by section 3.4 because the sensor mapping defined by α and l maps the situation shown in figure 5.6 into the region in sensor space associated with a hard left turn, even though this situation does not require an obstacle avoidance maneuver. The beam patterns shown in figure 5.6 also define a sufficient sensor mapping for our vehicle. The sensor space defined by α and l is acceptably incomplete because it errs on the side of caution.

For our vehicle $\alpha = 40.8°$ and $l = 25.3m$. The performance of this vehicle when it

Actuator Equivalence Class	Sensor Space Region	Situation Description	Actuator Response $(x, y, Mode)$
0	$d_l > d_r$	There is at least one object in the environment that requires a left turn to avoid	$(-, -, 0)$
1	$d_l = d_r = \infty$	There are no objects in the environment with which the vehicle can collide before they are detected	$(x_g, y_g, 1)$
2	$d_l \leq d_r \cap d_l \neq \infty$	There is at least one object in the environment that requires a right turn to avoid	$(-, -, 2)$

Table 5.3: Control mapping and sensor space definitions for the vehicle

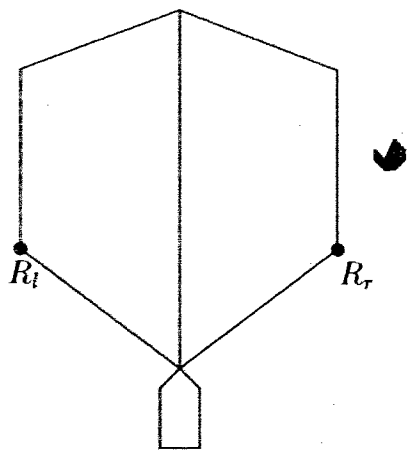


Figure 5.6: Another sufficient sensor mapping

encounters a flat wall is shown in figure 5.7. This figure was generated with our vehicle simulation. The initial position of the vehicle is $(x_v, y_v, \theta_v) = (0.0, 0.0, 0.0)$ and the endpoint location is $(0.0, 80.0)$. The wall extends between the points $(-30.0, 40.0)$ and $(30.0, 40.0)$. The trajectory of the vehicle does not contact the wall and the minimum distance between the wall and the vehicle centre line is $3.552m$ which is just larger than the defined vehicle clearance of $3.5m$.

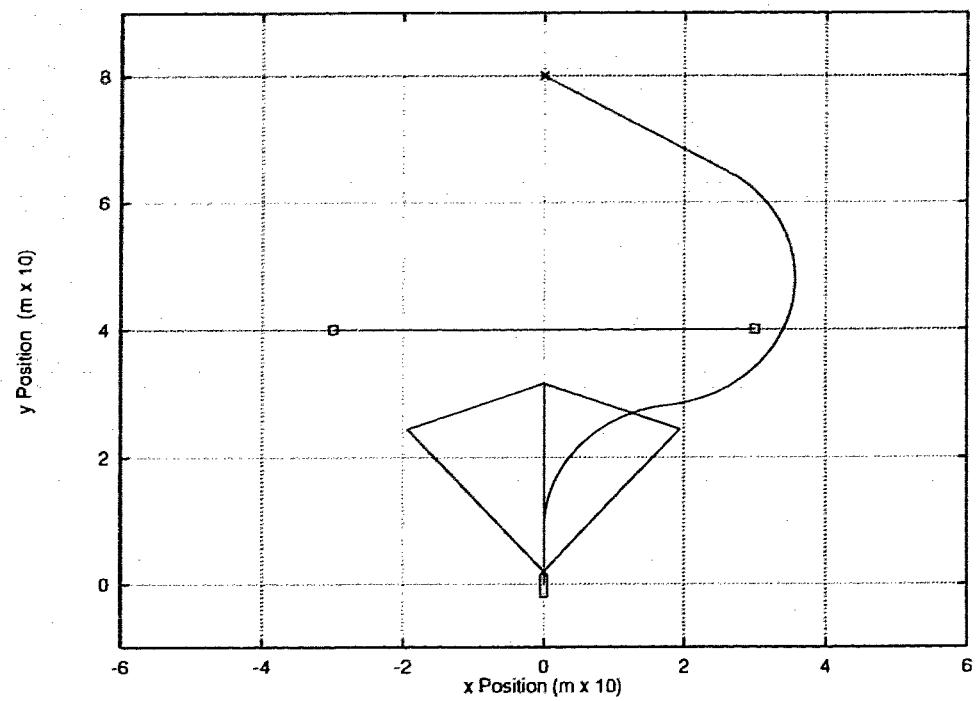


Figure 5.7: Phase II vehicle encountering a flat wall

When the vehicle is faced with a point object located at $(-10.0, 17.2)$ and an endpoint location of $(-50.0, 50.0)$, the vehicle passes the object with a minimum distance of $3.581m$, which is also just larger than the defined vehicle clearance. This trajectory is shown in figure 5.8.

This vehicle always passes objects that are between itself and the endpoint at a distance that is just larger than the defined vehicle clearance because of the situations that sonar sensors sense. When using only recently sensed sensor values and sensors like sonars, vehicles are limited to moving the edge of one beam pattern down one edge of the obstacles they encounter. Consequently, vehicles with limited range sensors, like sonars, are relegated to wall following if only recently sensed sensor values are used for control, which was also demonstrated by (Connell 1989).

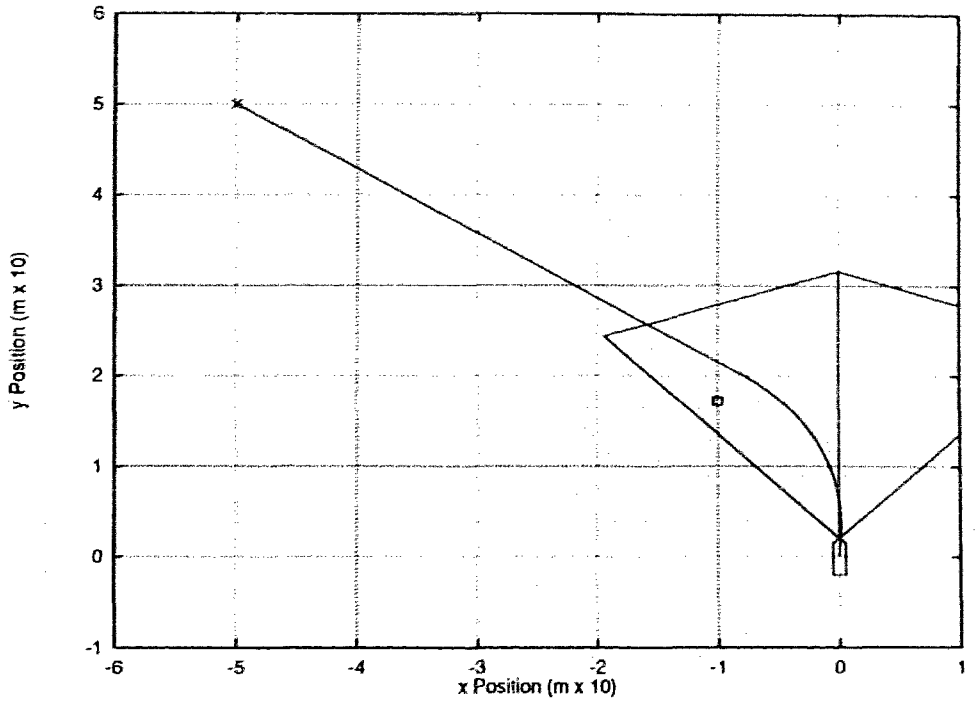


Figure 5.8: Phase II vehicle encountering a single object

To demonstrate the robustness of this system, consider the vehicle encountering a wall that is moving in the positive x direction. When the beam patterns first contact the wall, the wall extends from the point $(-30.0, 40.0)$ to $(5.0, 40.0)$. We call this scenario our "robustness test". This environmental disturbance simulates a dynamic environment and/or cumulative positioning system errors, as discussed in section 2.3. However, the phase II vehicle does not use position sensors for obstacle avoidance maneuvers, so for this vehicle we are only illustrating robustness to a dynamic environment.

The clearance of the vehicle as it passes the wall at different velocities is shown in figure 5.9. The performance degrades in a predictable, relatively linear fashion that is inversely proportional to the size of the disturbance. This is very similar to the degradation of a proportional controller to plant disturbances in the sense that performance degradation is proportional to the size of the disturbance.

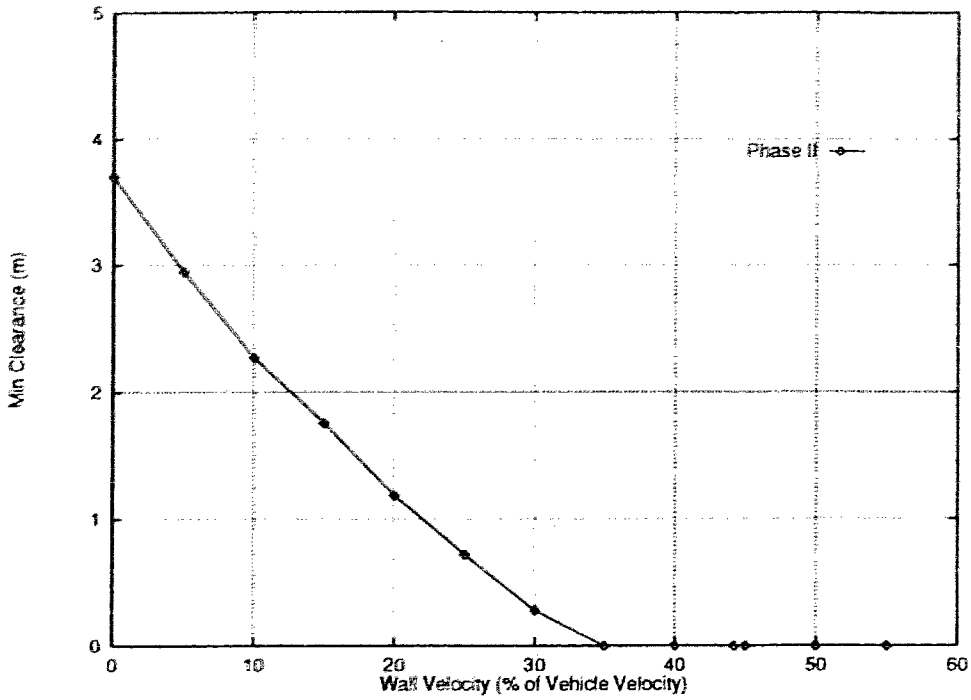


Figure 5.9: Robustness of the phase II vehicle to a dynamic environment

5.5 Phase III: Internal Representation Development

Internal representations only augment the control of the phase II system. That is, they alter only the responses associated with non-critical environmental situations under the mapping m. For our vehicle, the non-critical responses are those not associated with the primary subgoal of obstacle avoidance, which are the responses associated with actuator equivalence class 1 of table 5.3.

The internal representation we chose is an estimate of the endpoint of the object around which the vehicle must travel. This internal representation is relatively reliable because object endpoints are easily detected with sonar sensors. This is in contrast to internal representations that pertain to the size and location of objects in the environment, which are not easily detectable with sonar sensors. To illustrate the meaning of this internal representation, consider moving a sonar beam pattern across an object. When the beam pattern leaves the object the value returned by the sonar sensor increases to ∞. Given the distance value previously returned by the sonar and knowing the size and shape of the sonar beam pattern we estimate the location of

the endpoint of the object. If the object was last detected by the left sonar beam pattern, the vehicle must travel to the right of the object, and if the object was last seen in the right sonar beam pattern, the vehicle must travel to the left of the object. These situations can be represented in sensor space with three internal values $(x_{obj}, y_{obj}, beam)$ where (x_{obj}, y_{obj}) is the estimate of the location of the object endpoint in the world and $(beam)$ is a binary description of the last sonar beam pattern to contact the object. *beam* can have the value *left* or *right*, which indicates on which side of the object the vehicle must pass. The sensor and actuator spaces are expanded to $(x_{obj}, y_{obj}, beam, x_v, y_v, \theta_v, d_l, d_r)$ and $(x_{obj}, y_{obj}, beam, x, y, Mode)$ respectively, from the spaces of the phase II controller. These spaces and the control mapping are illustrated in figure 5.10.

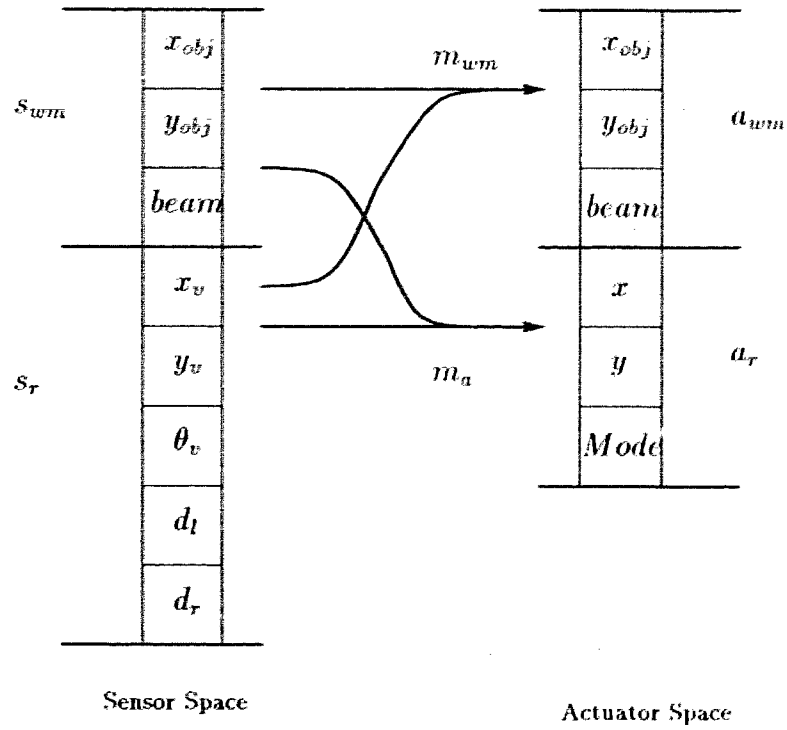


Figure 5.10: Phase III control mapping, with internal representations, showing sensor space and actuator space

An efficient method of using this internal representation is to store the object endpoint estimate for every control period in which the sonar beam patterns are in contact with an obstacle. Then, when the sonar beams no longer contact obstacles (ie. $(d_l, d_r) = (\infty, \infty)$), the estimate of the object endpoint is already stored in the internal

representation. This is an acceptable approach because the internal representations are not used to determine real actuator responses while the sonar beam patterns are in contact with an obstacle.

When neither sonar senses any obstacles (actuator equivalence class 1), the vehicle uses the information stored in the world map to plan a path around the obstacle with a greater clearance than the defined vehicle clearance. The control mapping uses an algorithm, which we call the path planning routine, to supply the low-level control system with intermediate waypoints that guide the vehicle around the obstacle. Once the vehicle is clear of the obstacle, the control mapping supplies the low-level control system with the mission endpoint as its waypoint. The path planning routine is part of the control mapping m_a.

The waypoint generated by the path planning routine is shown in figure 5.11, where the left sonar beam pattern was the last to contact the object. If the vehicle

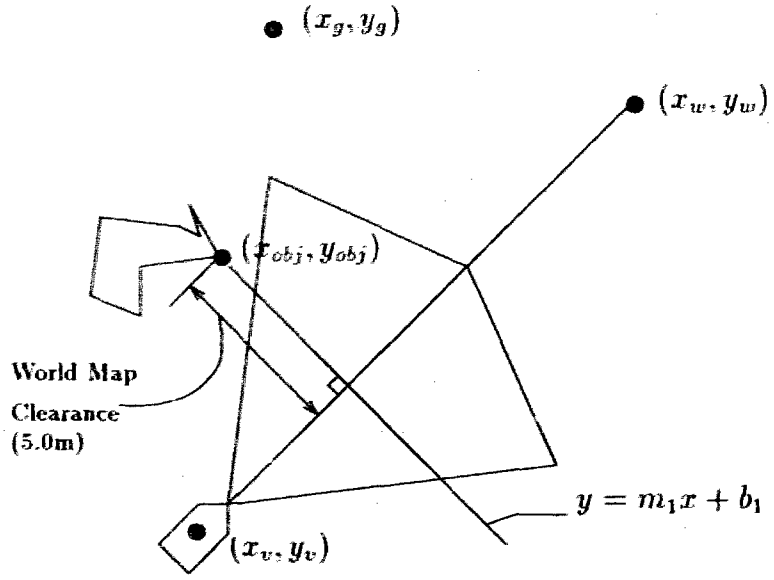


Figure 5.11: Waypoints generated by the path planning algorithm

has not crossed the line $y = m_1 x + b_1$ in figure 5.11, then the mapping m_a assigns the response $(x_w, y_w, 1)$ to the low-level control system. If the vehicle has crossed the line $y = m_1 x + b_1$ then the mapping m_a assigns the response $(x_g, y_g, 1)$ to the low-level control system. The vehicle determines if it has passed the line $y = m_1 x + b_1$ based on the sensor values of the positioning system, the goal location and the estimate of the obstacle endpoint location. The algorithm is described in appendix F.

The trajectories of the vehicle under this augmented control system are shown in figures 5.12 and 5.13 for the same scenarios experienced by the phase II vehicle. The

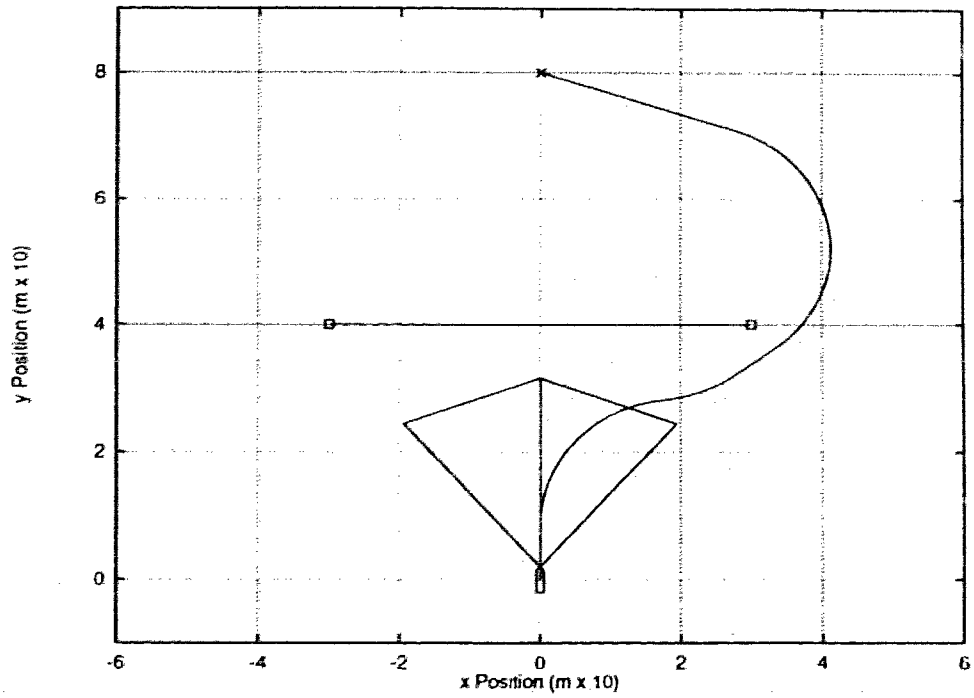


Figure 5.12: Phase III vehicle encountering a flat wall

clearance of the path planning routine is set to $5.0m$, which is how close each of the trajectories came to their respective obstacles.

The results of the robustness test for the phase III vehicle are shown in figure 5.14. The same linear trend in controller degradation is evident, however the vehicle has a larger clearance for equivalent disturbances, which results from the augmentation with internal representations. In the limit (ie. when there is always obstacles in at least one sonar beam pattern) the performance of the phase III vehicle is equal to that of the phase II system.

5.6 Brittle Design

In this section, we explore the effects of using internal representations before identifying the situations differentiated in sensor space. The initial instinct of many designers after the phase I experiments is to use internal representations to mitigate the shortcomings of the sensing system. However, as we will see, this is a short-sighted solution

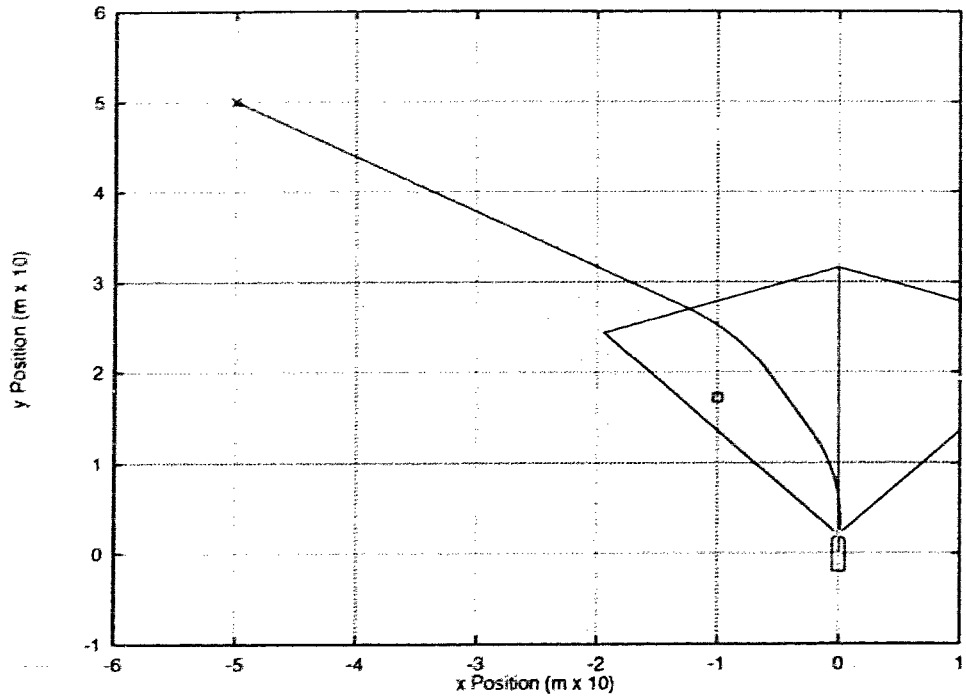


Figure 5.13: Phase III vehicle encountering a single object

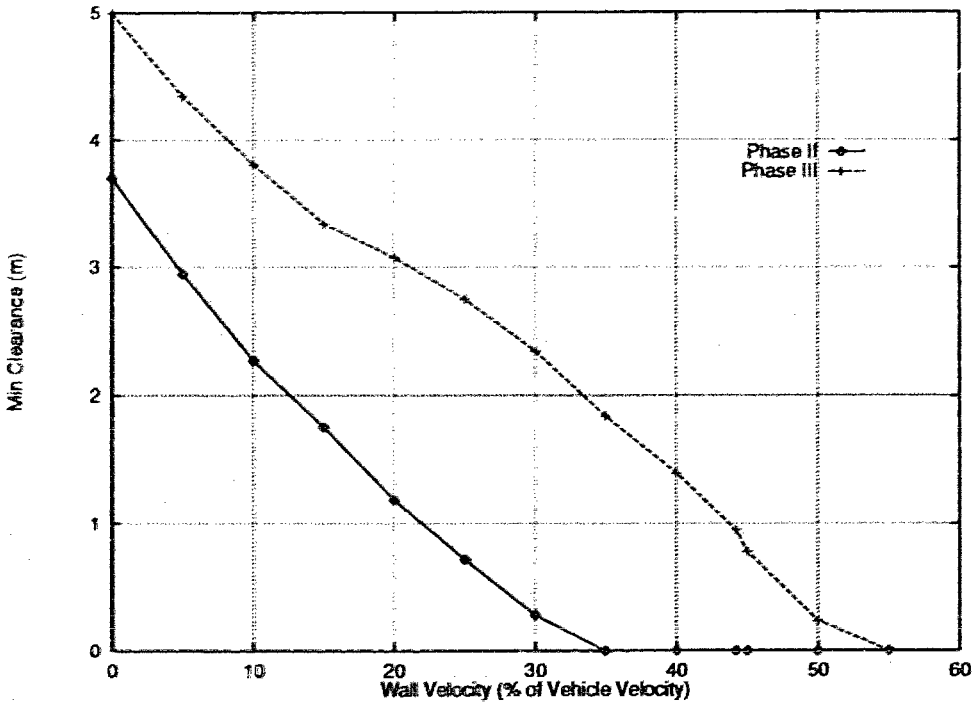


Figure 5.14: Robustness of the phase III vehicle to a dynamic environment and/or cumulative positional errors

to the problem. Consider using the phase III control mapping with the sonar beam patterns of phase I. That is, use the internal representations we have developed with improper beam patterns. In the flat wall scenario, the vehicle performs as expected, as shown in figure 5.15. The trajectory is very similar to that of the phase III vehicle

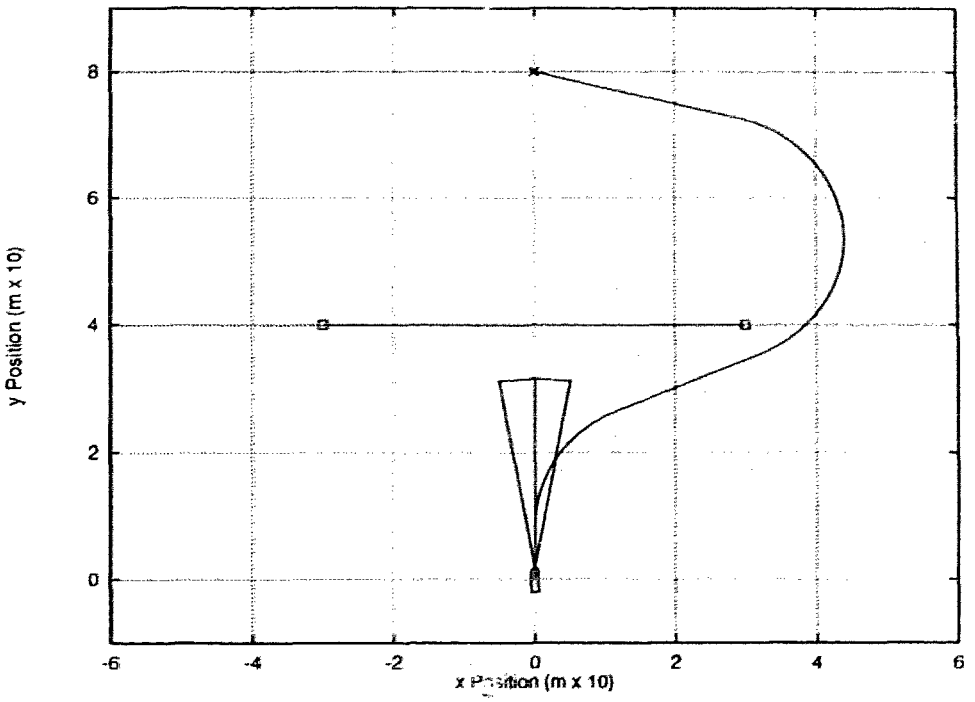


Figure 5.15: Brittle vehicle encountering a flat wall

and would be sufficient proof of a successful project for most designers. However, with the single object scenario, the vehicle passes the obstacle with a clearance of only $0.31m$ in simulation, which is equivalent to a collision. This scenario is shown in figure 5.16 and is similar to problems experienced by the ALV (Olin and Tseng 1991). That is, the vehicle collides with the obstacle because its sonar beam patterns never contact the obstacle. This results from a poorly designed vehicle in which the designer did not identify situations differentiated in sensor space. In short, for this example, sensor space is incomplete.

The robustness test provides more disappointing results. Figure 5.17 plots the results of this vehicle's robustness test relative to those of the phase II and phase III vehicles. As the size of the disturbance is increased, the performance of this system quickly degrades below that of the phase II system, which doesn't use any form of internal representation. Note that the response of this system to disturbances is very

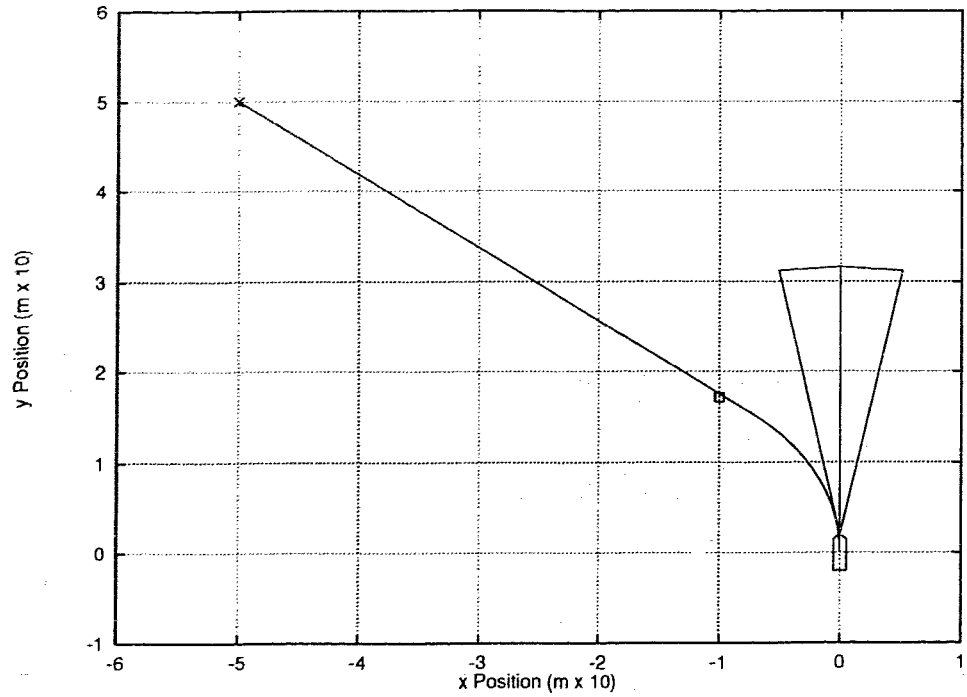


Figure 5.16: Brittle vehicle encountering a single object

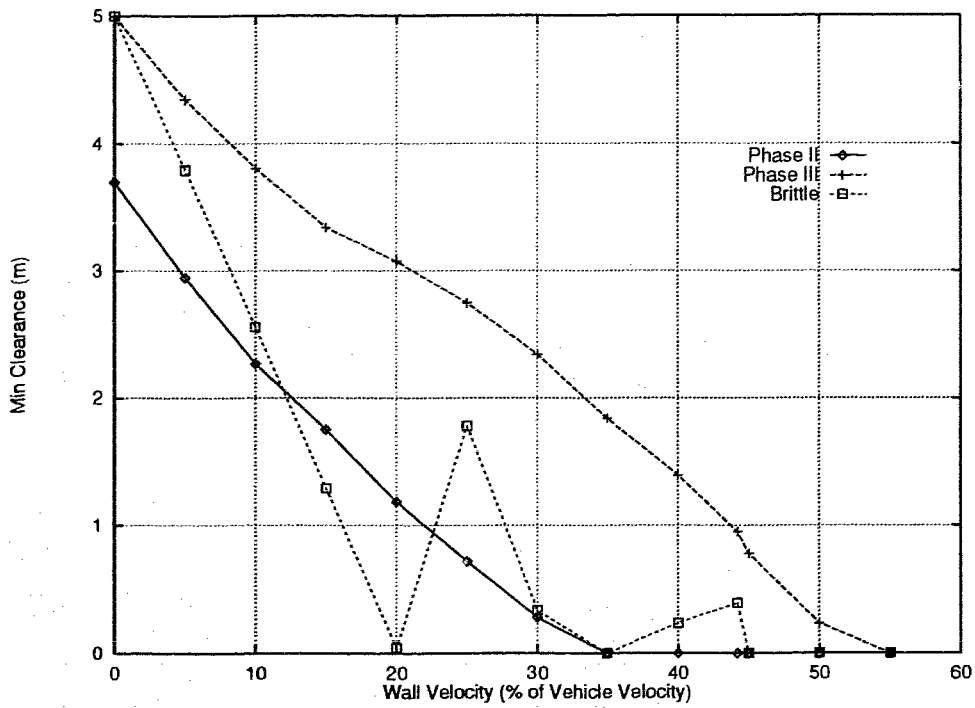


Figure 5.17: Comparison of the robustness of the brittle vehicle with the phase II and phase III vehicles

sporadic. For our system, the reason for the sporadic behaviour is that when the wall was moving at 25% of the vehicle's speed, the wall moved into the beam patterns an additional time, which caused the vehicle to turn again. Sporadic performance is typical of brittle systems since the way many brittle systems fail is often difficult to determine (Wallich 1991).

This system uses a relatively robust internal representation with an unacceptably incomplete sensor space. If less robust representations were used, the degradation of system performance under disturbance would be more drastic and more sporadic.

5.7 Summary

This chapter develops a design methodology for autonomous vehicles which develops systems that are robust to disturbances in their environment. The methodology is used to develop the sensor and control mappings of an AUV for the task of obstacle avoidance in an unknown obstacle field. We show that the vehicle is robust to disturbances by examining the clearance with which the vehicle passes a moving wall, a disturbance for which the vehicle was not designed. The limitations of our vehicle illustrate some of the limits of autonomous vehicles that are robust to disturbances in their environments. Our vehicle is limited to a wall following type of obstacle avoidance strategy.

In this chapter, we also show that the meanings associated with differentiable regions of sensor space are a function of the entire autonomous vehicle control cycle when the control environment is under-sensed. This fact means that autonomous vehicles operating in under-sensed control environments must be developed as a whole, and not as a collection of independent subcomponents. We also show that when vehicles are operating in under-sensed control environments, a priori determination of system brittleness is a difficult task without analysing the system in sensor space.

Through development of the vehicle in this chapter, we have found that sonar sensors are sufficient for the task of obstacle avoidance.

5.8 Discussion

In this chapter, we have seen how autonomous vehicles can be designed in a brittle manner through the inappropriate use of internal representations. The reason we were able to illustrate brittleness is that our system is very simple, and consequently, we were able to choose an appropriate scenario to illustrate the system's brittleness. Unfortunately, illustrating brittleness can be a difficult task when complex internal representations are used as part of autonomous vehicle control cycle. One reason for this fact is that the internal representations are often adapted, or patched, to solve problems encountered through experience with the vehicle. However, it is our opinion that by adding these patches, the designers construct systems that operate in their test environments, and they do not directly address the brittleness of their original design. It is our opinion that one reason many systems are brittle is that designers often make improper assumptions about the vehicle's environment that they might not even realize they are making. For example, designers might assume that tree's are permanent structures, because their removal is a rare occurrence. However, if the vehicle were to use the tree as a reference, the removal of the tree can make the system's brittleness manifest.

We have also seen the limited ability of a robust autonomous vehicle. To improve our vehicle's functionality requires sensors that are able to sense more of the control environment. For example, a sensor that sensed the entire obstacle avoidance environment of our vehicle would be ideal. Regardless of the solution to the sensing impediments, this chapter has outlined the need for better autonomous vehicle sensors.

Chapter 6

Conclusions

6.1 Summary

The lack of success in developing autonomous vehicles capable of performing many of the simple tasks that humans can perform led us to examine some of the fundamental assumptions underlying autonomous vehicle development. Traditionally autonomous vehicle development is viewed as an artificial intelligence problem, and consequently, one of the implicit assumptions made by most designers of autonomous vehicles is that the intelligence possessed by humans can be mechanically, or electrically, reproduced. In this thesis, we work from the assumption that humans are more than the sum of their components, and that machines are only the sum of their components. Consequently, autonomous vehicle development is viewed as a control problem in the sense that the vehicle is a machine operating in a control cycle.

The reason many autonomous vehicle designers do not view autonomous vehicle development as a control problem is that the control environment of autonomous vehicles is often under-sensed. We describe autonomous vehicle control in under-sensed control environments with a generalized control cycle that includes sensor space and actuator space. Analysis of control problems in sensor space is a natural extension to existing control theory and can also be used to describe more traditional, critically sensed control problems. One requirement of well-designed systems is that sensor space be complete.

Using Q-SAMs, which are lookup-table equivalents of computer-based control algorithms, we explored the potential of situation-based control and showed that Q-SAMs

can implement many forms of control laws. We filled a Q-SAM with a proportional control law to show that Q-SAMs can be filled with rule-based control laws. Through a process known as downloading, we recorded the control law of an expert controller whose control law was unknown. One requirement of the downloading process is that the sensors must differentiate the situations differentiated by the expert. We also explored adaptive situation-based control and found that adaptation during a single movement to the goal region is not, as yet, practical. One important implication of the existence of Q-SAMs is that the underlying principle of operation of all autonomous equipment is that they simply take a set of inputs and through some predefined law produce a set of outputs, which means that autonomous vehicles simply react to the environment in which they exist and, in our opinion, do not understand that environment.

In under-sensed control environments, situations differentiated in sensor space are a function of the entire autonomous vehicle as well as the environment in which the vehicle exists. Consequently, a good method of identifying differentiable situations is through physical experimentation that does not involve any form of internal representations. That is, experimentation that uses only recently sensed sensor values to determine actuator responses.

We used this idea as one of the corner stones of a new design methodology for autonomous systems. The other corner stone of the methodology is to use internal representations to improve performance in such a way that they do not affect actuator responses associated with critical environmental situations. This methodology develops systems that are robust to disturbances in their environment.

The methodology was used to determine the sensing requirements of an autonomous underwater vehicle using sonars for the task of obstacle avoidance in an unknown obstacle field. Through development of the vehicle, we showed the limitations of robust obstacle avoidance in an unknown obstacle fields using sonar sensors. Our vehicle is limited to a wall-following type of avoidance strategy. The vehicle development also illustrated that the manifestations of brittleness associated with internal representations are difficult to determine without appropriate sensor space analysis.

This thesis has laid the foundation for development of a control theory for under-sensed control environments, which will provide a mathematical basis for autonomous vehicle control systems.

6.2 Contributions

In this thesis, we have made several contributions to the autonomous vehicle community. Our most significant contribution is providing a different set of assumptions from which to base autonomous vehicle development. We assumed that computers and humans are fundamentally different, and that it is the designers, and not the computers or autonomous vehicles, that understand the environment in which the computer, or autonomous vehicle, exist. In accordance with this idea, we described autonomous vehicles in a control cycle where the control environment is often under-sensed. This process introduced the ideas of under-sensed and critically sensed control environments. It also introduced the concepts of sensor space, actuator space, situations, situation differentiation and situation identification. With these concepts, we showed that situations differentiated in sensor space are a function of the entire autonomous system when the control environment is under-sensed, whereas they are a function of only the sensor transformation when the control environment is critically sensed.

We have also introduced the autonomous vehicle community to Q-SAMs which are lookup-table equivalents of computer-based control systems. We have used Q-SAMs to illustrate the strengths of situation-based control.

The final contribution we made in this thesis is a design methodology for systems operating in under-sensed control environments which develops systems that are robust to disturbances in their environments.

6.3 Future Work

In this thesis, we laid the foundation of a new area of control theory: control in under-sensed environments. Consequently, one area of future work is the development of this control theory. Much of this work will revolve around digital systems, though it will have implications for continuous systems. This theory, when developed, will provide a mathematical framework for designing autonomous vehicles.

A second area of future work is sensor interpretation. A shortcoming of autonomous vehicles is their inability to construct reliable world maps using their own sensors. Overcoming this deficiency requires identification of unique environmental features that vehicle-based sensors can differentiate. These features will be the basis

of vehicle generated world maps. This research also requires a theoretical framework for describing systems which use sensors that accumulate errors, like on-board navigational units, to determine the maximum distance a vehicle can travel between features stored in the world map. It is our opinion that when autonomous vehicles are able to uniquely identify regions of the environment using their own sensors, the day of household autonomous vehicles will be upon us.

Appendix A

SHAKEY

In this appendix, we provide a more detailed description of the SHAKEY (Nilsson 1984)[1] project. In section A.1, we describe the robot and its control architecture. In section A.2, we explain how SHAKEY's reasoning system, STRIPS, develops plans for the robot. Finally, in section A.3, we describe how plans are decomposed into actions and provide a brief summary of this appendix.

A.1 SHAKEY and its Architecture

SHAKEY is a mobile cart that has a camera and two range finders which it can pan and tilt. SHAKEY determines its position and orientation from two shaft encoders connected to its two drive wheels.

SHAKEY's environment consists of a few rooms that are connected with doorways. The walls are light and their edges highlighted with thick dark lines. Inside the rooms are a few blocks and wedges, each painted a distinct colour for easy identification by the vision system under proper lighting, which is provided.

The general architecture of SHAKEY's higher-level control system is shown in figure A.1, which is a decomposition by function organization that is used by many autonomous systems. The sensing and sensor analysis subsystems update internal

[1]This is the classical SHAKEY reference but unfortunately we were unable to obtain it. Instead our information comes from personal knowledge and secondary sources like (Nilsson 1980), (Brooks 1991) and (Shapiro and Eckroth 1987).

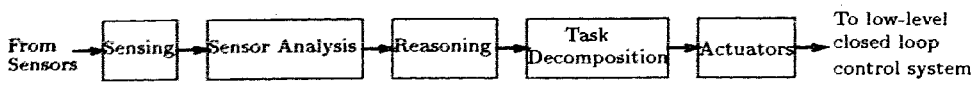


Figure A.1: Decomposition by function type of controller organization

representations that are stored in the reasoning subsystem. The reasoning subsystem uses the internal representations to develop a plan that accomplishes system goals, which are determined by an operator external to SHAKEY. A typical goal for SHAKEY is to organize the blocks in a certain fashion. The task decomposition subsystem decomposes the plan generated by the reasoning subsystem into physical actuator commands that are then executed by the actuator subsystem.

A.2 Reasoning in STRIPS

SHAKEY's reasoning engine is an implementation of STRIPS (STanford Research Institute Problem Solver), which is a non-commutative production system based on first order logic in the form of predicate calculus. We explain these ideas with an example from Nilsson (1980) because it is simple and clear. Everything in STRIPS is defined with well formed formulas (wffs), which are simply legitimate predicate calculus expressions. The state of the world shown in figure A.2 can be described with the following wffs:

$onfloor(A)$
$onfloor(B)$
$on(C, A)$
$clear(B)$
$clear(C)$
$handempty$

where $onfloor(x)$ represents the fact that object x is on the floor, $on(x, y)$ represents the fact that x is directly on top of y, $clear(x)$ represents the fact that there are no objects on top of x and *handempty* represents the fact that the robot is not holding any objects.

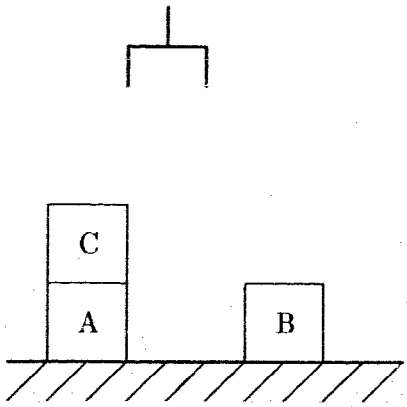


Figure A.2: Blocks World

SHAKEY reasons about its world by searching through different combinations of production rules to find a combination of production rules that will change the present state of the world to the goal state. Goals are also defined in the form of wffs like:

$$on(A, B) \wedge on(B, C) \tag{A.1}$$

which represents the goal of having a stack of blocks with A on top B in the middle and C on the bottom. Production rules represent physical actions that SHAKEY can perform in the world like

pickup(x)

preconditions:	$ontable(x), clear(x), handempty$
delete:	$ontable(x), clear(x), handempty$
add:	$holding(x)$

where the preconditions of the rule must exist in the current state of the world (ie. must be true) for the rule to be applied to the system. Application of a production rule removes the wffs in the delete list from the world state and adds the wffs in the add list to the world state, which changes the state of the world. Some of the production rules SHAKEY uses are:

$goto(x)$	moves the robot into the vicinity of door x
$push(dist, ob, tol)$	pushes object *ob dist* feet with a tolerance of *tol*
$roll(dist, tol)$	moves the robot forward *dist* feet with a tolerance of *tol*
$gothrudr(door, fromrm, torm)$	move through door *door* from room *fromrm* to room *torm*

The search through different combinations of production rules is done with the A^* algorithm, which produces the best possible plan according to some minimization function. The A^* algorithm is a search algorithm that used heuristics to increase the speed of the search process. The plan is represented as an ordered set of production rules like

$$\{goto(D1), gothrudr(D1, R1, R2), roll(5, 0.1), push(10, BL1, 0.5), \ldots\}.$$

A.3 Implementing the Plan

Once the plan is generated, each production rule of the plan is decomposed into specific actions by the task decomposition subsystem. For example, the production rule $goto(D1)$ is decomposed into a series of actions that will move SHAKEY to near door $D1$. Depending upon SHAKEY's position in the room, this move might require SHAKEY to perform a complex series of maneuvers. The route to the door is determined using a connected graph of the environment and the A^* algorithm to search through the graph for the best path. The connected graph is an internal representation of objects and routes in the world. It is used in addition to the blocks world internal representation of the world. Though not in the reasoning subsystem, path planning is another instance of reasoning performed by SHAKEY. Once the task is decomposed into physical actions the actuator subsystem physically implements each action in turn.

To summarize, SHAKEY maintains a list of facts about the world in the form of wffs. SHAKEY reasons about accomplishing goals using production rules and the A^* algorithm, which produces a plan that is an ordered list of production rules. Each element of the plan is decomposed into physical tasks that can be accomplished by the robot. These physical tasks are then executed by the physical robot. SHAKEY's intelligence is in the form of its ability to reason about accomplishing goals in the world.

Appendix B

Allen

In this appendix, we provide a more detailed description of Allen and the subsumption architecture. In section B.1, we describe the subsumption architecture and in section B.2, we describe Allen's operation in detail.

B.1 Subsumption Architecture

Most work involving behaviour based control stems from the ideas of Rodney Brooks and the subsumption architecture (Brooks 1986). Instead of decomposing a robot in the traditional sense of figure A.1, robots are decomposed into layers of task achieving behaviours as shown in figure B.1. Behaviours are defined by the external manifesta-

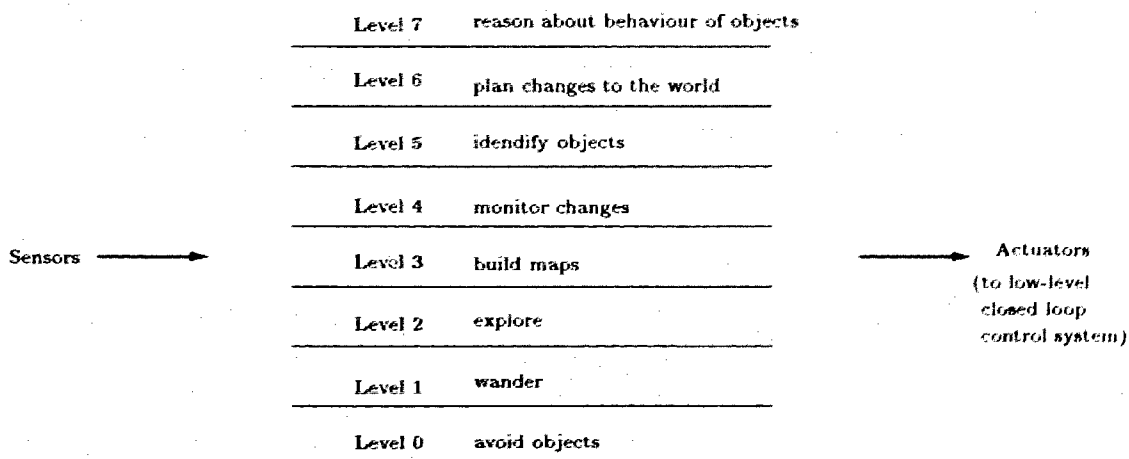


Figure B.1: Subsumption architecture

tions of activity with which they are associated, not by any internal representations they might possess. During development with the subsumption architecture, robots are constructed incrementally by building modules that accomplish each behaviour in turn. Higher layers (ie. larger numbers in figure B.1) are constructed in such a way that they subsume the control of all lower layers (ie. smaller numbers in figure B.1). This ensures that the functions of the lower, more survival oriented robot behaviours are always incorporated into control decisions. The first layer built is level 0.

Subsuming behaviours is analogous to how humans move their extremities. For example, our hand reflexively stops at, or moves away from, obstacles which it contacts, a level 0 behaviour. When our cognitive mind decides to move our hand across a table, it subsumes the level 0 behaviour so that we do not drive our hand through any obstacles, but instead our hand stops or moves around obstacles encountered enroute. The "cognitive mind" we refer to is represented by levels 6 and 7 of figure B.1. Note that Rodney Brooks' goal is to build artificial creatures capable of reasoning about their environment by adding increasingly sophisticated behaviours to his robots. It is Brooks' opinion that intelligence can result from a complex collection of behaviours.

The subsumption architecture has several advantages over traditional techniques. First, central representations that must satisfy all robotic needs, which can be difficult to develop, are not required. Instead, each behaviour uses sensor information and internal representations pertinent to its task, which may be completely independent of those used by other tasks. Secondly, the surprises encountered when subcomponents of figure A.1 are finally assembled together are avoided because there is a complete robotic system at all levels of subsumption development. By "surprises" we mean unexpected system failings. Building systems incrementally, as the subsumption architecture requires, prevents designers from over estimating the abilities of their systems because design deficiencies are discovered throughout the development process. In addition, the subsumption architecture explicitly handles the timing constraints discussed in (Albus 1981). Lower-level behaviours, which respond to the environment have shorter control periods while higher-level behaviours, which do more sophisticated work, have longer control periods. The subsumption architecture is a structure that can respond to the environment while it is thinking. Finally, the subsumption architecture degrades gracefully under failure. When a higher level behaviour fails

to provide a response in sufficient time, due to failure or computational load, control reverts to the next lower level, whose control is less adept, but still sufficient to at least guarantee the robot's safety. Because behaviours operate independently, systems built with the subsumption architecture do not suddenly fail, as is the case when many traditional designs are overloaded.

B.2 Allen

Allen (Brooks 1986) (Brooks 1990) is an implementation of the first three levels of the subsumption architecture. Allen physically moves with a drive unit that rotates in place by a specified number of degrees or moves forward a specified distance. For Allen, a general move is implemented by first turning and then moving forward. Allen is equipped with 12 sonar transducers evenly distributed around its circular frame, one of the transducers faces forward. Each transducer returns a "time of flight" value that is proportional to the distance to the nearest obstacle in front of the transducer.

Level 0 control is accomplished by using the transducers to move the vehicle away from obstacles. The inverses of the values returned by the sonar transducers are summed to determine a "force vector" whose direction indicates the centre of free space and whose magnitude is proportional to the distance the robot will move. The level 0 controller first sends a turn command to the motion controller which turns the vehicle. When the motion controller is done turning, it sends an acknowledgement to the level 0 controller which replies with the appropriate forward motion command. When the motion controller completes the forward motion, it sends an acknowledgement to the level 0 controller to indicate that it has completed the forward motion. If, at any time during forward motion, the sonar transducer facing forward indicates that an obstacle is too close to the vehicle, the level 0 controller sends a halt command to the motion controller, which stops the robot. This prevents the robot from colliding with objects that are moving or were not previously detected because of gaps in the sonar coverage. Providing the control period is sufficiently short, level 0 control moves Allen away from any objects moving towards it and stops Allen when it is about to collide with an object. In a static environment the level 0 controller moves Allen to the centre of free space and keeps it there.

The level 1 controller is designed to allow Allen to "wander" around the environment while avoiding obstacles. This task is accomplished by generating a random motion vector approximately every 10 seconds. The random motion vector is added to the force vector of level 0 to produce a new force vector which moves the vehicle towards the goal, while avoiding obstacles. The halt reflex of level 0 is still active. By adding a direction vector to the level 0 force vector, the level 1 controller subsumes the level 0 controller. A new direction vector is determined for the vehicle that incorporates the vehicle's obstacle avoidance behaviour.

The level 2 controller gives Allen the ability to move down corridors. This is done using the odometric and sonar sensors. When the vehicle stops for a short while, as determined by the odometric sensors, the level 2 controller determines the farthest location, relative to Allen, in sensed free space. The vector to this location replaces the randomly chosen direction vector of the level 1 control system. By replacing the level 1 direction vector, the level 2 controller subsumes the level 1 controller and also the level 0 controller.

With this control system, when the level 2 goal is unattainable, Allen simply choses another goal. For example, if the goal location is behind a short wall that was not initially detected, Allen will stop in front of the wall in such a way that the repulsive forces of the wall are balanced with the attractive force of the goal. For a short while Allen sits defeated. However once the level 2 controller notices the inactivity of the odometric sensors, a new goal is determined, based on the sonar values, and Allen heads toward that new goal. The net effect of this system is that Allen wanders aimlessly around its environment, tending to move down halls.

In summary, Allen has 3 behaviour layers, an obstacle avoidance behaviour, a vehicle behaviour, and an explore behaviour. The explore behaviour looks for the most distant location in sensed free space, and subsumes the wander behaviour, which in turn subsumes the obstacle avoidance behaviour to make Allen explore its environment. Allen's intelligence is exhibited when it moves around the environment without colliding with obstacles.

Appendix C

Autonomous Land Vehicle (ALV)

In this appendix, we provide a more indepth discussion of the Autonomous Land Vehicle (ALV) (Payton 1986), (Daily 1988), (Payton 1990), (Olin and Tseng 1991), (Thorpe 1991). In section C.1, we discuss the ALV and its control architecture. In section C.2, we describe the progress of the ALV project. Finally, in section C.3, we and describe future work on the ALV that was not fully described in the main body of the thesis.

C.1 ALV and its Control Architecture

The ALV is an 8-wheeled vehicle designed to navigate over mildly rough terrain. It uses an onboard navigation system to determine the vehicle's position, orientation, pitch and roll relative to the world. The ALV also has a range scanner that scans an 80° horizontal and 30° vertical swath in front of the vehicle. Experiments with the ALV were conducted in a grassy field that contained gullies and rocks.

The control architecture of the ALV is shown in figure C.1. The low-level closed loop control system of figure A.1 is incorporated into the Motion Controllers block of this architecture. For autonomous vehicles, the ALV's architecture is relatively standard in the sense that it has a sensing leg (the left side of the figure) and an actuation leg (the right side of the figure). The bottom portion of the architecture is connected to real sensors and real actuators. As we move vertically in the perception system, data is assimilated producing a more complete picture of the environment. As we move down the actuation leg, tasks are decomposed into increasingly smaller

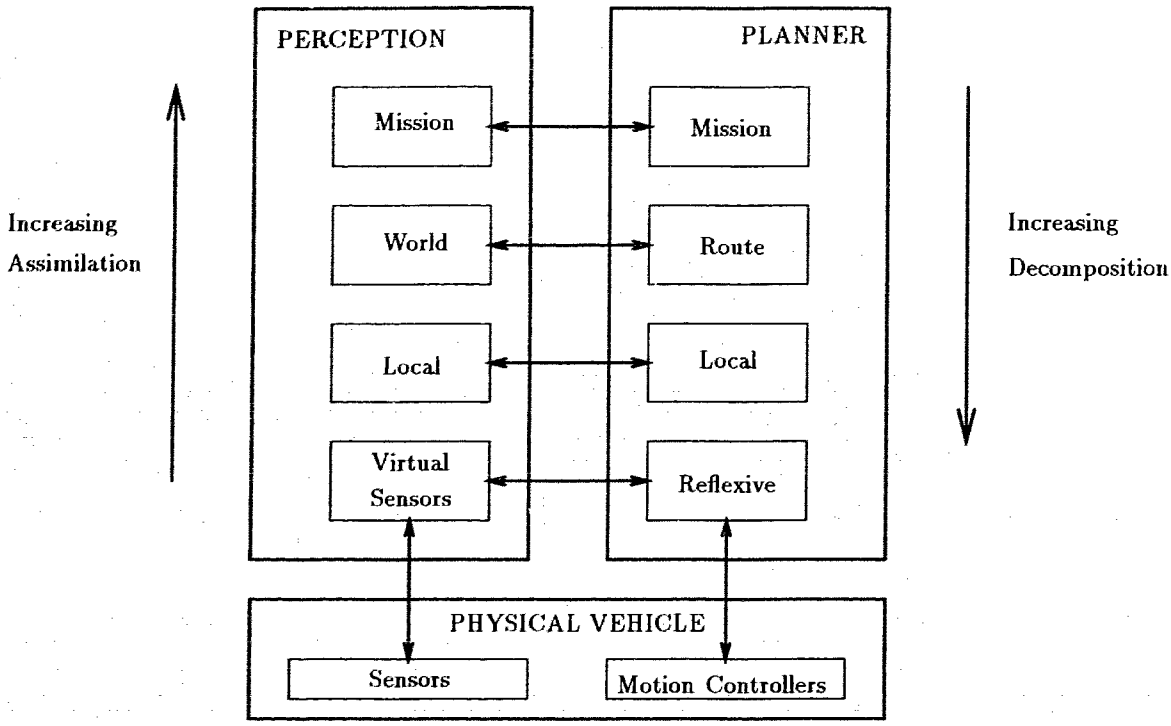


Figure C.1: ALV control architecture

subtasks to the point that they are actual motion commands. The different levels of the perception leg provide information which is pertinent to the tasks being decomposed in the respective levels of the actuation leg. The structure of figure C.1 is described in (Daily 1988) but the ideas supporting the architecture are described in (Payton 1986).

The mission planning module defines system goals and constraints, and instructs the mission sensing module to configure the sensors to look for specific landmarks in the environment. This level of the architecture is designed to interact extensively with human mission planners. The world perception module maintains a world map of the environment that includes a list of landmarks indicating which have and have not been sensed. The route planner module uses the world map and the constraints of the mission planner module to determine a satisfactory route through the environment. The local perception module performs sensor fusion. It identifies landmarks and passes this information to the world perception module, and it identifies obstacles and environmental conditions and passes that information to the local planning

module. The local planning module uses route information and environmental information to determine which reflexive behaviour will control the vehicle in the reflexive planning module. The virtual sensor module detects specific environmental features as requested by the local perception module or the reflexive behaviour module. The term "virtual sensor" is used because the sensors values might not correspond to a single sensor, but might result from the processing of several sensor values. The reflexive behaviour module implements the currently active behaviour as specified by the route planning module.

The implemented version of the ALV has two behaviours, called activities by (Daily 1988); one which is active when obstacles are sensed, "find-clearest-path-to-goal", and the other which controls the vehicle when obstacles are not sensed, "travel-toward-goal". Each activity consists of several "behaviours" which can issue vehicle commands like $speed = 3m/s$ and $turn = 10°/sec$ to a blackboard architecture that arbitrates requests according to a fixed arbitration scheme. Each "behaviour" requests specific information from the virtual sensor module which provides that information in a timely manner.

It is the opinion of some researchers that this structure is similar in operation to how humans perform tasks. At the lowest level are reactive elements that provide responses to the immediate environment. As we move up the architecture the activities become increasingly abstract and cognitive.

C.2 ALV Progress

Despite the well thought-out architecture, the implementation stage of the ALV has not progressed beyond the architecture's lowest levels (Daily 1988). The system implemented uses a predetermined world map without any landmarks requiring identification by the sensing systems. The world map contains information about trees, rocks, gullies and the slope of terrain in the environment. The world map is used to plan a route through the environment to a predefined endpoint by specifying waypoints along the route through which the vehicle must pass. The rangefinder is used in conjunction with the navigation system to develop a vehicle centred elevation map of the local environment. Virtual sensors calculate the distance the vehicle can safely

move in seven different forward directions along the elevation map. Only seven directions are analysed because of the processing constraints placed on the system by the real time environment. Along with the traversable distance, the virtual sensors return the reason the distance was chosen (ie. obstacle, lack of sensor data, etc.). When an obstacle is detected along one of the seven paths, the local planning module activates the "find-clearest-path-to-goal" activity which chooses the best of the seven evaluated routes with obstacle avoidance as the primary control emphasis. When no obstacles are present along the seven routes the "travel-toward-goal" activity controls vehicle motion. This activity chooses the route best suited to moving the vehicle to the next waypoint.

C.3 ALV Analysis and Future Work

The higher levels of the architecture were not implemented because the sensors used are insufficient to uniquely identify landmarks in the environment. This lack of sensing ability is also addressed in the implemented portions of the architecture because the system is designed to differentiate only the presence or absence of obstacles. An obstacle is defined as something through which the vehicle cannot traverse. The actual sensors are not used to identify rocks, trees or gullies. The information in the world map and the sensors is used to evaluate the feasibility of potential routes.

Though Dailey (1988) reported a successful experiment, much later work, (Payton 1990), (Olin and Tseng 1991) discusses some of the shortcomings of the original design and provides solutions to these shortcomings. The following discussion describes these solutions with the author's explanations as well as our own.

The first failing of the original design is that the system always moves to the next waypoint, regardless of the present situation. Figure C.2 illustrates this scenario when an unmapped obstacle prevents the vehicle from following its initial route. The vehicle circles the boulder to move to the next waypoint, which is clearly a bad move. The vehicle should have continued on to the third or fourth waypoint. Payton consider this to be an abstraction problem. When the vehicle turned away from the planned path, the waypoints abstracted from the route were no longer appropriate. Payton refer to the vehicle as "unopportunistic" because it did not move towards waypoint 3 when the opportunity presented itself. It is our opinion that the system was not

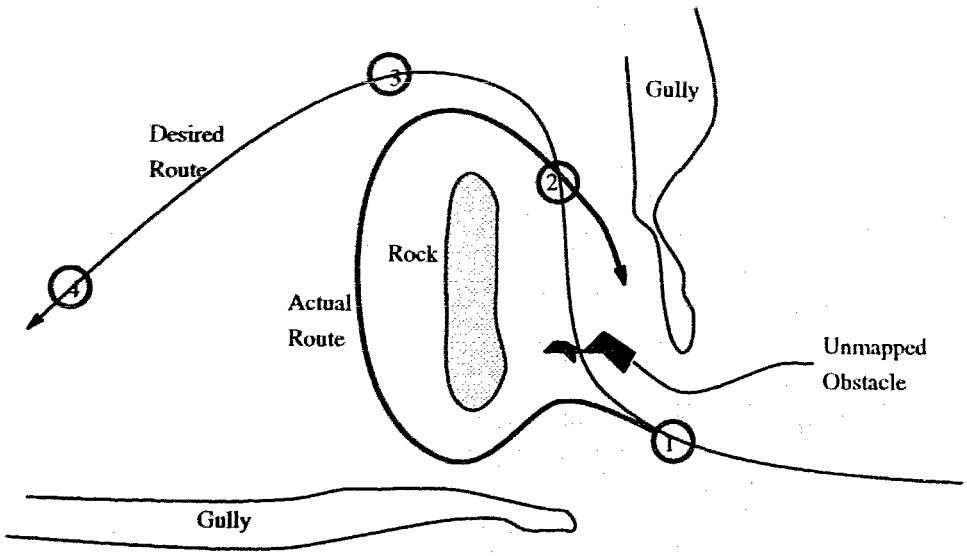


Figure C.2: ALV attempting to follow a blocked route

programmed properly. That is, the controller assigned an inappropriate response to the present environmental situation. We can consider the route determined by the planning system to be a very brittle form of internal representation. We use the term "very brittle" because the internal representation of the path is generated from the internal representation of the world map. Payton's solution is to generate a gradient field over the entire world map, as shown in figure C.3. Now, when the vehicle moves

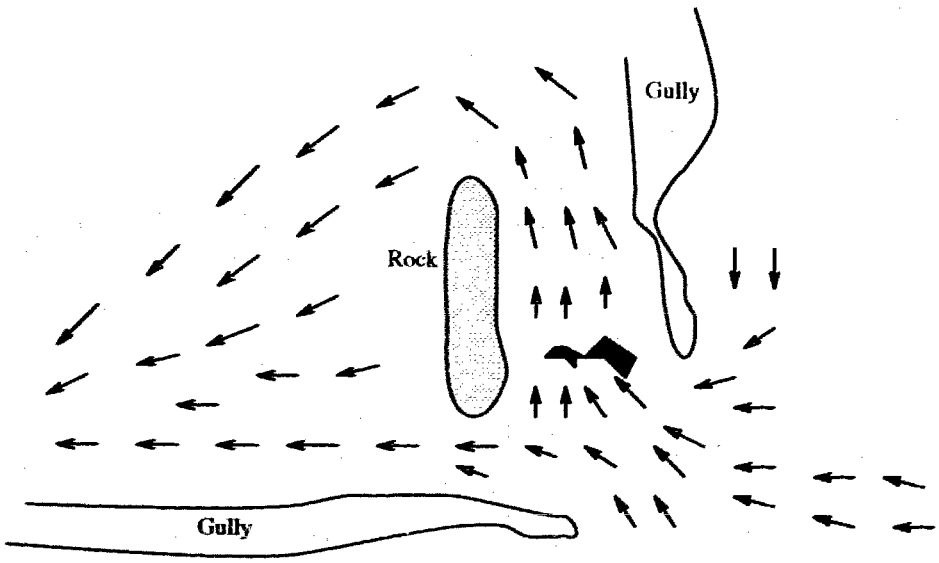


Figure C.3: ALV gradient field world map

off the planned path of figure C.2, it takes advantage of its new situation. It is our

opinion that Payton simply made better use of the situations in the environment that their vehicle's sensors are able to differentiate. This is one example of altering internal representations to address problems that are encountered during testing, instead of addressing the system brittleness directly.

Another deficiency of the original design, as noted in Olin and Tseng (1991), is that the vehicle has a tendency to move back towards objects that it has already cleared because the "travel-towards-goal" activity does not consider obstacles when determining the most appropriate route. Payton refers to this scenario as a "command arbitration problem" because it is his opinion that the problem lies in the fact that the vehicle's motion is determined exclusively by only one of the vehicle's two distinct activities. We refer to this scenario as behaviour fusion, described in section 2.3. Payton proposes to solve this problem with a connectionist approach where each activity states its preference for different actuator responses, and the most preferred response is chosen for actuation. This again is simply making better use of the situations in the environment that the vehicle is capable of sensing. Again, it is our opinion that Payton is making better use of the situations in the environment that his vehicle's sensors are able to differentiate.

Some of the problems discussed in section 2.3 were not addressed by Payton, though it is our opinion that they experienced them. For example, the brittleness associated with the cumulative errors of positioning systems was not discussed. From Olin and Tseng (1991), we know that 8 of 18 runs terminated successfully at the goal point defined by the land navigation system. However, the correspondence between the actual goal location and the sensed goal location was not mentioned. Also, the longest vehicle trip was only $735m$. Perhaps positioning system errors prevented longer runs.

Finally, Payton never addressed the brittleness associated with world maps and dynamic environments. Fortunately, the environment of the ALV was relatively static. That is, the hills, rocks and gullies never moved. However, if anything had changed after the world map was constructed, the ALV would never have known. For example, if the rock in figure C.3 were to roll down the page, the vehicle would not take advantage of the potential path through the rock's previous location.

To summarize, despite the elaborate architecture initially envisioned by the designers of the ALV, the implemented architecture basically differentiated situations

in the environment that the sensors were able to differentiate and assigned responses to those situations. Payton's recent work has made better use of situations in the environment which the sensing system is able to differentiate.

Appendix D

Situation-based Adaptive Control Surface

This appendix explains the shape of the control surface generated by the adaptive situation based control algorithm. The general shape of the control surface can be though of as an equilibrium between two pressures. One pressure, from the adaptation algorithm, increases the magnitude of the force stored in each Q-SAM quantum, and the other pressure, from the actuator response distribution function (ARDF), decreases the magnitude of the force stored in each quantum. In this appendix we describe these pressures and their equilibrium point.

The adaptation algorithm places a positive pressure on the magnitude of the force stored in each Q-SAM quantum because, on average, the force recorded to each quantum has a larger magnitude than the force stored in that quantum. To illustrate this fact, consider Q-SAM quantum i, which is associated with some positive error whose stored force, $F(i)$, is some value that is less than the equilibrium value. When the vehicle moves into sensor space quantum i, a force in the range $[-2.0N+F(i), 2.0N+F(i)]$ is applied to the system. Forces closer to $2.0N + F(i)$ are more likely to move the system towards the goal than forces closer to $-2.0N + F(i)$. Therefore, the expected force recorded to quantum i is greater than $F(i)$, which tends to increase the value stored in quantum i. The converse is true if quantum i is associated with some negative error. In fact, the pressure exerted on the force stored in quantum i can be mathematically represented as the expected increase in the force stored in quantum i

when a new force is recorded by the adaptation algorithm. That is,

$$\Delta F_{inc}(i) = (\hat{F}(i) - F(i))\, w_{ii} \tag{D.1}$$

where $\Delta F_{inc}(i)$ is the expected change in the value stored in quantum i, which is equivalent to a pressure exerted on that force, $\hat{F}(i)$ is the average force recorded in quantum i, $F(i)$ is the value presently stored in quantum i, and w_{ii} is the constant of proportionality, which is the diffusion factor from the ARDF (equation 4.2).

Counteracting the positive pressure exerted by the adaptation algorithm is the negative pressure exerted by the ARDF. We describe the pressure as negative because the ARDF usually reduces the magnitude of the value stored in each quantum by trying to make all the values equal to zero. This scenario arises because the ARDF, when repeatedly applied to a control surface without any interference from the adaptation routine, eventually transforms that control surface into a flat line. In our case, that flat line is the line $force = 0$ because the adaptation algorithm, on average, stores positive forces for positive errors and negative forces for negative errors, and the only value that is common to both positive and negative forces is zero. The pressure exerted by the ARDF can be represented as the expected change in the value stored in a quantum when a value is recorded to another quantum. That is,

$$\Delta F_{dec}(i,j) = (F(i) - \hat{F}(j))\, w_{ij} \tag{D.2}$$

where $\Delta F_{dec}(i,j)$ is the expected change in the magnitude of quantum i that results from recording a new force into quantum j and w_{ij} is the diffusion factor from the ARDF (equation 4.2).

On average, the control surface takes the shape that balances the two pressures described in equations D.1 and D.2 in each sensor space quantum. That is,

$$p(i)\Delta F_{inc}(i) = \sum_{j=1, i\neq j}^{N} p(j)\Delta F_{dec}(i,j) \tag{D.3}$$

where $p(i)$ is the probability of recording a value in quantum i at any time during adaptation, and N is the number of quantization regions in the Q-SAM. Equation D.3 must be satisfied for all sensor space quanta i. Combining equations D.1, D.2, and D.3, the equilibrium state is described as

$$0 = \sum_{j=1}^{N} p(j)(F(i) - \hat{F}(j))w_{ij} \;\; \forall i. \tag{D.4}$$

In equation D.4, $\hat{F}(j)$, the average force recorded to Q-SAM quantum j, and $p(j)$, the probability of being in a quantum j, are unknown. In the paragraphs that follow, we describe $\hat{F}(j)$ and $p(j)$ as functions of the vehicle's dynamical parameters, the Q-SAM parameters, the adaptation parameters, and the present state of the control surface.

First, we describe $\hat{F}(j)$ in terms of distributions of random variables associated with specific sensor space quanta. The adaptation algorithm records a force f into the Q-SAM if that force moves the system towards the goal region without moving it too much closer to the goal region. In chapter 4, we defined too much closer as 25% and 75% of the distance to the goal region. More specifically, $F(j) = f$ is recorded if

$$k(j) < x < t_c(j) \tag{D.5}$$

where $k(j)$ is the width of quantum j, which is $0.0625m$ in our case, $t_c(j)$ is the distance that is too close to the goal region for quantum j, and x is the distance from the edge of the quantized region farthest from the goal region to the position of the AUV after force f is applied for one control period. x is a function of the system dynamics specified in equation 3.1, which is repeated here as

$$x = af + bv + x_o \tag{D.6}$$

where

$$a = \frac{\Delta t}{d} - \frac{M}{d^2}(1 - e^{-\frac{d}{M}\Delta t}), \tag{D.7}$$

$$b = \frac{M}{d}(1 - e^{-\frac{d}{M}\Delta t}), \tag{D.8}$$

v is the velocity before application of the force f, and x_o is the actual location of the AUV in the quantum before application of the force f. These parameters are shown in figure D.1.

For any quantum, f, v, and x_o can be described as the random variables F, V, and X_o, respectively, which are constrained by the equation

$$X = aF + bV + X_o \tag{D.9}$$

where X is the random variable associated with x. From equations D.5 and D.6, a force f is recorded if

$$\frac{k(j) - bv - x_o}{a} < f < \frac{t_c(j) - bv - x_o}{b}. \tag{D.10}$$

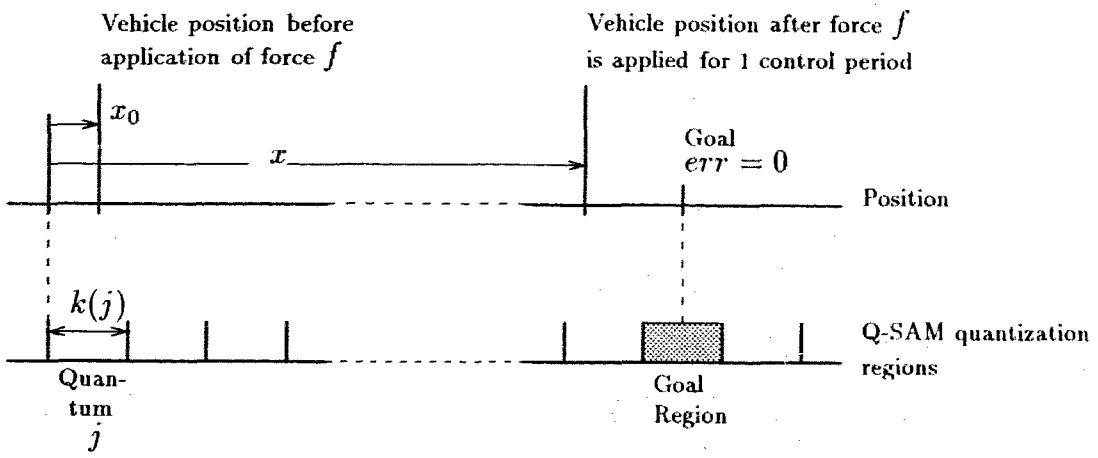


Figure D.1: Parameters described in equations D.5 and D.6

The average force recorded to a particular quantum is the average force satisfying equation D.10, which can be found by taking the expected value of the function

$$g(F,V,X_o) = \begin{cases} f & \frac{k(j)-bv-x_o}{a} < f < \frac{t_c(j)-bv-x_o}{a} \\ 0 & else \end{cases} \tag{D.11}$$

which is

$$\begin{aligned} \hat{F}(j) &= E_{FVX_o}\{g(F,V,X_o)\} && \text{(D.12)} \\ &= E_{VX_o}\{E_{F|VX_o}\{g(F,V,X_o)\}\} && \text{(D.13)} \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f_{VX_o}(v,x_o)\int_{\frac{k(j)-bv-x_o}{a}}^{\frac{t_c(j)-bv-x_o}{a}} f_{F|VX_o}(f)\,df\,dv\,dx_o. && \text{(D.14)} \end{aligned}$$

If we assume that F, V and X_o are independent for any quantum, then

$$\hat{F}(j) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f_V(v) f_{X_o}(x_o)\int_{\frac{k(j)-bv-x_o}{a}}^{\frac{t_c(j)-bv-x_o}{a}} f_F(f)\,df\,dv\,dx_o \tag{D.15}$$

where

$f_F(f)$ is the distribution of forces applied in the Q-SAM, which is uniformly distributed with a mean of $F(j)$ (ie. $U[F(j)-2.0, F(j)+2.0]$).

$f_V(v)$ is the distribution of velocities associated with forces that are recorded into that quantum j. This distribution is determined by the system parameters $M, d, \Delta t$, the range and quantization of the Q-SAM and the distributions of the random perturbations and new goal locations.

$f_{X_o}(x_o)$ is the distribution of the locations in the quantum associated with forces that are recorded into that quantum. This distribution is also a function of all the system parameters and is approximately uniform (ie.$U[0, k(j)]$).

We now describe $p(j)$ in a similar fashion as we described $\hat{F}(j)$. $p(j)$, the probability of recording a force into quantum j at any time, is

$$p(j) = p(\textit{being in quantum } j)\, p(\textit{recording} \mid \textit{you are in quantum } j). \tag{D.16}$$

$p(\textit{being in quantum } j)$ is a function of the random distribution of the goal locations, the control surface and the system dynamics. $p(\textit{recording} \mid \textit{you are in quantum } j)$ is the probability that the system leaves the quantum in one control period and does not move too much closer to the goal region. More specifically,

$$p(\textit{recording} \mid \textit{you are in quantum } j) = p(k(j) < x < t_c(j)) \tag{D.17}$$

where the probability density function of x is defined by equation D.9 and is a function of $f_F(f), f_V(v), f_{X_o}(x_o)$. That is,

$$f_X(x) = \frac{f_F(\frac{a}{x})}{|a|} * \frac{f_V(\frac{b}{x})}{|b|} * f_{X_o}(x) \tag{D.18}$$

where $*$ is convolution. Therefore equation D.17 becomes

$$p(\textit{recording} \mid \textit{you are in quantum } j) = \int_{k(j)}^{t_c(j)} f_X(x)\, dx \tag{D.19}$$

and

$$p(j) = p(\textit{being in quantum } j) \int_{k(j)}^{t_c(j)} f_X(x)\, dx \tag{D.20}$$

is the probability of recording in quantum j at any time.

To summarize, the control surface generated by the adaptation algorithm of chapter 4 is a function of the parameters associated with the system dynamics, the Q-SAM, and the adaptation algorithm. The average control surface is a balance between a pressure from the adaptation algorithm, which increases the magnitude of the force stored in each Q-SAM quanta, and a pressure exerted by the ARDF, which decreases the magnitude of the force stored in each Q-SAM quantum.

Appendix E

Sonar Beam Pattern Derivation

In this appendix, we derive the beam width α and beam length l of the sonar beam patterns of our vehicle in chapter 5 so that the vehicle can avoid all obstacles in an environment, where the most complex obstacle is a flat wall, using only recently sensed sensor data to determine actuator responses. We first determine the beam width and then the corresponding beam length. The discussion in this appendix focuses on the left sonar bam pattern with little mention of the right beam pattern because the derivation of both beam patterns are very similar.

The sonar beam patterns must be wide enough to continually sense all potential vehicle trajectories, with the vehicle's clearance. We refer to the trajectory of the vehicle's clearance as the clearance trajectory, as shown in figure E.1. The minimum beam width that satisfies this criteria is shown in figure E.1, where r is the vehicle's minimum turn radius, c is the vehicle's clearance, and d is the distance the sonars are in front of the centre of the vehicle. The trajectories shown in figure E.1 are those associated with the minimum turning radius, which require the widest beam width of all potential trajecties to sense. Note that in figure E.1, a portion of the vehicle trajectory, and a portion of the clearance trajectory, directly in front of the vehicle are not continually sensed. This region is called the critical region and the most forward point of the critical region is the release point R_l shown in figure 5.4, which is repeated here as figure E.2. The significance of the release point is that when the vehicle moves the sonar beam patterns away from obstacles after the release point (ie. between the vehicle and the release point), the obstacle is in the critical region, and obstacles in the critical region can violate the vehicle's clearance requirements

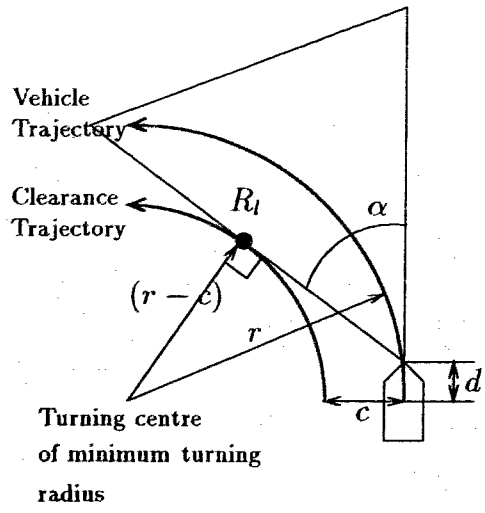


Figure E.1: Beam width derivation parameters

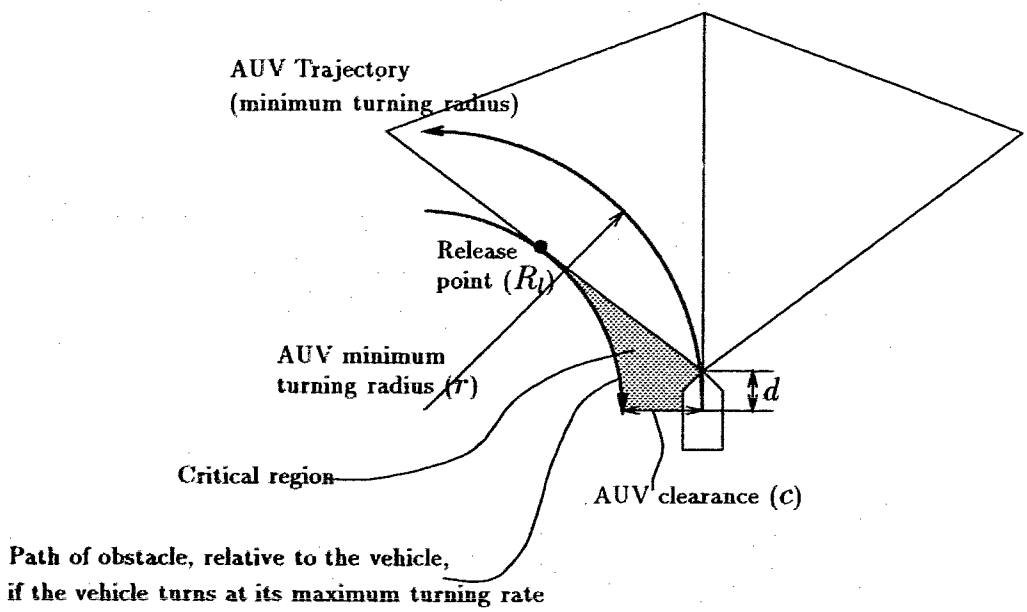


Figure E.2: Sensor critical region

without being sensed.

The geometric relationships associated with the beam width derivation are shown in figure E.3. At the release point R_l, the edge of the sonar beam pattern is tangent to

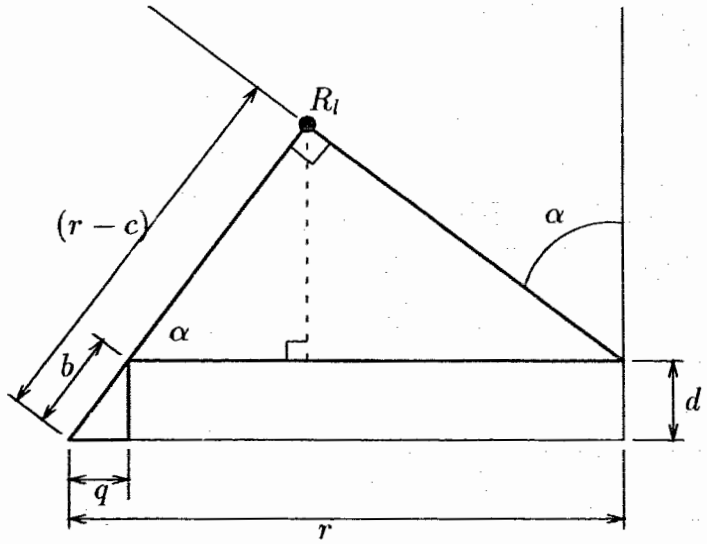


Figure E.3: Triangle in beam width derivation

the clearance trajectory and perpendicular to the line from the turning centre of the minimum turning radius to the relase point. This derivation begins with relationships obtained from the highlighted triangle in the left corner of figure E.3, which are

$$\sin\alpha = \frac{d}{b} \tag{E.1}$$

$$\cos\alpha = \frac{q}{b} \tag{E.2}$$

$$b^2 = d^2 + q^2 \tag{E.3}$$

where α is the desired beam width, and b and d are parameters used in this derivation. Another relationship is obtained from the upper highlighted triangle of figure E.3 and is

$$\cos\alpha = \frac{r-c-b}{r-q}. \tag{E.4}$$

Equating equations E.2 and E.4 results in the following relationship between q and b.

$$rq - q^2 = (r-c)b - b^2. \tag{E.5}$$

Replacing q in equation E.5 with q in equation E.3 and rearranging yields

$$r(b^2 - d^2)^{\frac{1}{2}} = (r-c)b - d^2. \tag{E.6}$$

Squaring both sides and rearranging results in the equation for b stated in the thesis as equation 5.2, which is

$$c(2r-c)b^2 + 2d^2(r-c)b - d^2(r^2+d^2) = 0. \tag{E.7}$$

The beam width α is then

$$\alpha = \arcsin\frac{d}{b} \tag{E.8}$$

where b is the appropriate solution to equation E.7.

From figure E.3, the left release point, in vehicle based coordinates, is

$$R_l = (-r + (r-c)\cos\alpha, d + (r-c)\sin\alpha). \tag{E.9}$$

Similarly, the right release point i.

$$R_r = (r - (r-c)\cos\alpha, d + (r-c)\sin\alpha). \tag{E.10}$$

Now, we address the beam length, which must be sufficiently long so that the vehicle can turn itself, and the release point, away from a flat wall. This scenario is depicted in figure E.4, which shows the vehicle and the release point when turning at

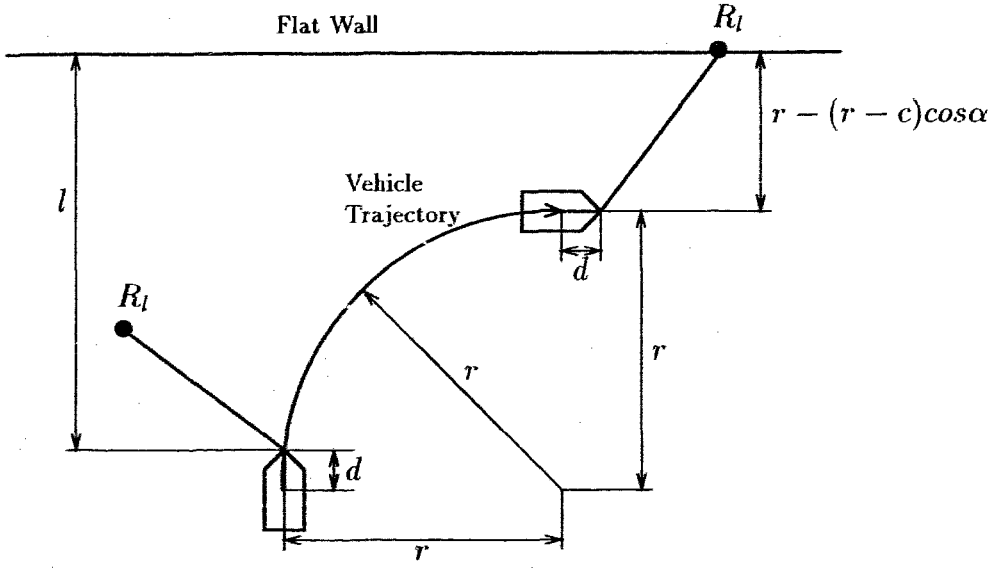


Figure E.4: Beam length derivation

the vehicle's maximum turning rate, which is the strategy of the our vehicle. From figure E.4, we obtain the relationship

$$l + d = r\,|R_{l_x}| \tag{E.11}$$

where $|R_{l_x}|$ is the magnitude of the x-coordinate of the left release point in vehicle based coordinates. Substituting the value of $|R_{l_x}|$ from equation E.9 results in the relationship

$$l + d = r + r - (r - c)\cos\alpha \tag{E.12}$$

because $(r - c)\cos\alpha < r$. Rearranging equation E.12 yields the beam length specified in the thesis as equation 5.5 which is

$$l = 2r - d - (r - c)\cos\alpha. \tag{E.13}$$

Appendix F

Path Planning Algorithm

In this appendix, we describe the calculations used by our path planning algorithm to determine the next vehicle waypoint, when no obstacles are present in the sonar beam patterns. First, we describe the algorithm and then we display the code that implements the algorithm. In this appendix, we focus on the scenario of an obstacle to the left of the vehicle because the calculations for obstacles to the right of the vehicle are very similar.

In this algorithm, the estimated object endpoint and the endpoint location are transformed from the world based coordinate system to the vehicle based coordinate system, shown in figure F.1. The shortest vehicle trajectory that passes to the right of the obstacle by a distance equal to the world map clearance r_{wm} is specified by the point (x_2, y_2) and the line $y = m_4 x$ in figure F.1. If the endpoint location (x_g, y_g) is to the left of the line $y = m_4 x$, the endpoint location is effectively behind the obstacle, so the point (x_2, y_2) is the next vehicle waypoint. If the endpoint is to the right of line $y = m_4 x$, the endpoint location (x_g, y_g) is not behind the obstacle, so the endpoint location (x_g, y_g) is the next vehicle waypoint. Finally, the next vehicle waypoint is transformed form vehicle based coordinates to world based coordinates for the low-level vehicle control system.

The algorithm that implements this code follows:

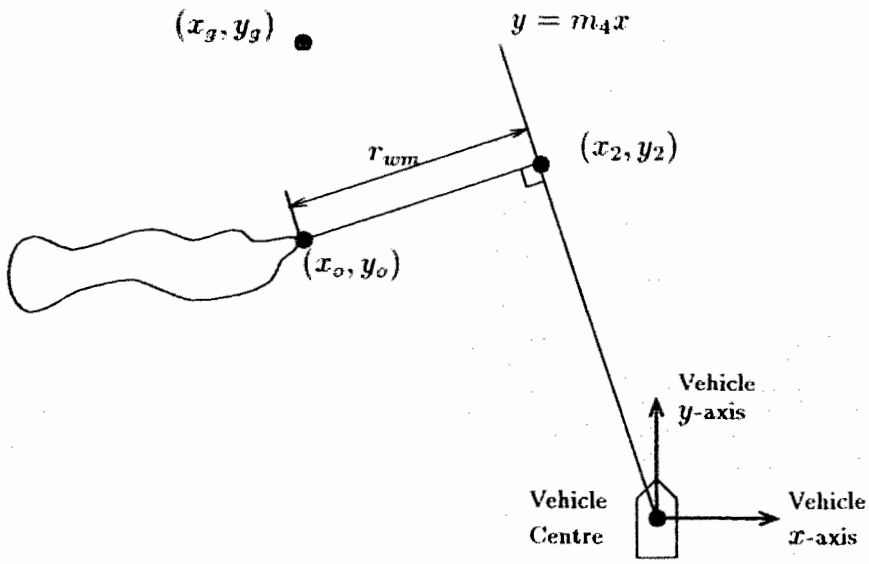


Figure F.1: Path planning algorithm

```
auv_wm_control()  /* the world map control algorithm */
{

float **a,**b;
Mat m1;
Point po;   /* object location, relative to vehicle */
Point pg;   /* the goal location, relative to the vehicle */
Point wp;   /* the calculated waypiont, relative to the vehicle */
Point p2;   /* the clearance point, defined in my notes */
float slope_m4;   /* from my math, sorry */

/* get object location in robot coordinates */
if (Obj_loc.x ≠ BIGREAL)
  {

    a = matrix(1,4,1,4);  /* Numerical Recipes form of transformation matrix */
    b = matrix(1,4,1,1);  /* A vector needed to do Gauss Jordan inverse */
```

```
b[1][1] = 1.0; b[2][1] = 2.0; b[3][1] = 3.0; b[4][1] = 2.0;/*arbirary numbers*/
Mat_to_nrmat(sub1.m,a);  /* convert sub matrix to numerical recipes mat */
gaussj(a,4,b,1);      /* invert matrix a, result into a */
nrmat_to_Mat(a,m1); /* convert inverse matrix back to us */
mat_point_mult(m1,&Obj_loc,&po);  /* determine the location of the object
                                     in AUV coordinate frame */
mat_point_mult(m1,&Old_goal,&pg);  /* determine the location of the goal
                                     in AUV coordinate frame */

/* determine new waypoint in robot coordinates  */

if  (po.y ≤ 0.0 )        /*  we have passed the object, so just go to the goal */
  {
    point_copy(&Old_goal,&goal.loc);
  }
else
  {   /* begin - calculating waypoints in AUV coordinates */

    if (beam == Left)
      {
        /* r2 = WM_clearance * WM_clearance;   clearance for ISE vehicle
                                              under world map control, squared */
        if ( po.x*po.x + po.y*po.y < WM_clearance * WM_clearance)
          {
            wp.x = 150.0; wp.y = 0.0; wp.z = 0.0; /* we are closer,
                        than the clearance, induce a hard right turn */
          }
        else
          {
            auv_get_line_and_clearance_point(po,&p2,&slope_m4);
            if (pg.x < (pg.y / slope_m4))
```

```
        {
          point_copy(&p2,&wp);  /* goal is behind object */
        }
      else
        {
          point_copy(&pg,&wp); /* goal is clear */
        }
    }
  }
else if (beam == Right)
  {
    /* r2 = WM_clearance * WM_clearance;   clearance for ISE vehicle
                                    under world map control, squared */
    if ( po.x*po.x + po.y*po.y < WM_clearance * WM_clearance)
      {
        wp.x = -150.0; wp.y = 0.0; wp.z = 0.0;/* we are  closer
                    than the clearance,  induce a hard left turn */
      }
    else
      {
        auv_get_line_and_clearance_point(po,&p2,&slope_m4);
        if (pg.x > (pg.y / slope_m4))
          {
            point_copy(&p2,&wp); /* goal is behind object */
          }
        else
          {
            point_copy(&pg,&wp);  /* goal is clear */
          }
      }
  }
```

```
        else
          {
           printf(" \n ERROR: beam was not left or right \n");
          }

        mat_point_mult(sub1.m,&wp,&goal.loc); /* convert the waypoint back
                                to world coordinates */
      } /* end calculating the waypoint in AUV coordinates */
  }
else  /* if no objects have been detected to date */
  {
   point_copy(&Old_goal,&goal.loc);
  }

auv_goal_seek_horizontal(); /* now, use the regular goal seeking control algorithm */

}
```

References

Agre, P. E. and D. Chapman (1990). What are plans for? In P. Maes (Ed.), *Designing Autonomous Agents*, pp. 17–34. Amsterdam, Netherlands: Elsevier Science Publishers B.V.

Albus, J. S. (1981). *Brains, Behaviour, and Robotics*, Chapter 5, pp. 113–120. Peterborough, New Hampshire: BYTE Books.

Arkin, R. C. (1990). Integrating behavioural, perceptual, and world knowledge in reactive navigation. In P. Maes (Ed.), *Designing Autonomous Agents*, pp. 105–122. Amsterdam, Netherlands: Elsevier Science Publishers B.V.

Badreddin, E. (1991, April). Tailoring the behaviour of a mobile robot - a recursive approach. In *Proceedings IEEE International Conference on Robotics and Automation*, Volume 3, pp. 2556–2561. Robotics and Automation Society of the IEEE.

Bellingham, J. G. (1990). Keeping layered control simple. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, pp. 3–8. Oceanic Engineering Society of the IEEE.

Brooks, R. A. (1986, March). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation 2*(1), 14–23.

Brooks, R. A. (1989). A robot that walks; emergent behaviours from a carefully evolved network. In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 692–696. IEEE Robotics and Automation Society.

Brooks, R. A. (1990). Elephants don't play chess. In P. Maes (Ed.), *Designing Autonomous Agents*, pp. 3–15. Amsterdam, Netherlands: Elsevier Science Publishers B.V.

Brooks, R. A. (1991, April). Intelligence without reason. Technical Report 1293, MIT Artificial Intelligence Laboratory.

Connell, J. H. (1989). A colony architecture for an artificial creature. Technical Report 1151, MIT Artificial Intelligence Laboratory.

Connell, J. H. (1992, May). SSS: A hybrid architecture applied to robot navigation. In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 2719–2724. IEEE Robotics and Autonomous Society.

Daily, M. (1988). Autonomous cross-country navigation with the ALV. In *Proceedings IEEE International Conference on Robotics and Automation*, Volume 2, pp. 718–725. IEEE Robotics and Autonomous Society.

Dreyfus, H. L. (1992). *What Computers Still Can't Do: A Critique of Artificial Reason.* Cambridge, Massechuesetts: MIT Press.

Jacquot, R. G. (1981). *Modern Digital Control Systems.* Marcel Dekker Inc.

Malcom, C. and T. Smithers (1990). Symbol grounding via a hybrid architecture in an autonomous assembly system. In P. Maes (Ed.), *Designing Autonomous Agents*, pp. 123–144. Amsterdam, Netherlands: Elsevier Science Publishers B.V.

Moravec, H. P. (1983, July). The Stanford Cart and the CMU Rover. *Proceedings of the IEEE 71*(7), 872-884.

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

Nilsson, N. J. (1980). *Principles of Artificial Intelligence.* Tioga.

Nilsson, N. J. (1984, July). SHAKEY the robot. Technical Report 323, SRI International, Artificial Intelligence Centre.

Olin, K. E. and D. Y. Tseng (1991, August). Autonomous cross-country navigation - an integrated perception and planning system. *IEEE Expert Magazine 6*(4), 16–30.

Payton, D. W. (1986). An architecture for reflexive autonomous vehicle control. In *Proceedings IEEE International Conference on Robotics and Automation*, Volume 3, pp. 1838–1845. IEEE Robotics and Autonomous Society.

Payton, D. W. (1990, November/December). Planned guided reaction. *IEEE Transactions on Systems, Man and Cybernetics 20*(6), 1370–1382.

Shapiro, S. C. and D. Eckroth (Eds.) (1987). *Encyclopedia of Artificial Intelligence*, Volume 2. John Wiley and Sons Inc.

Simmons, R. and E. Krotkov (1991). An integrated walking system for the ambler planetary rover. In *Proceedings IEEE International Conference on Robotics and Automation*, Volume 3, pp. 2086–2091. IEEE Robotics and Autonomous Society.

Smith, S. M. and D. J. Comer (1991, August). Automated calibration of a fuzzy logic controller using a cell state space algorithm. *IEEE Control Systems Magazine 11*(5), 18–28.

Thorpe, C. (1991, August). Toward autonomous driving: The cmu navlab. *IEEE Expert Magazine 6*(4), 31–42.

Ullman, S. (1984). Visual routines. In *Cognition*, Volume 18, pp. 97–159. Elsevier.

Van de Vegte, J. (1986). *Feedback Control Systems.* Prentice-Hall.

Wallich, P. (1991, December). Silicon babies. *Scientific American 256*(6), 124–134.