THESES SIS/LIBRARY
R.G. MENZIES LIBRARY BUILDING NO:2
THE AUSTRALIAN NATIONAL UNIVERSITY
CANBERRA ACT 0200 AUSTRALIA

TELEPHONE: +61 2 6125 4631
FACSIMILE: +61 2 6125 4063
EMAIL: library.theses@anu.edu.au

USE OF THESES

This copy is supplied for purposes
of private study and research only.
Passages from the thesis may not be
copied or closely paraphrased without the
written consent of the author.

Control of Sensory Perception for Discrete Event Systems

Geir Edvin Hovland

Siv.Ing. (NTNU Norway)

August 1997

*A thesis submitted for the degree of Doctor of Philosophy
of The Australian National University*

Department of Engineering
Faculty of Engineering and Information Technology
The Australian National University

Declaration

The work contained in this thesis, unless explicitly stated, is original research whose major portion was done by the author. The work has not been submitted for a degree at any other university or institution. The research has been carried out in collaboration with Dr. Brennan J. McCarragher. The results presented in the thesis have been published or submitted to journals and conferences as listed below.

Journal Papers:

- [J1] G.E. Hovland and B.J. McCarragher, Hidden Markov Models as a Process Monitor in Robotic Assembly, *The International Journal of Robotics Research*, Vol. 17, No. 1, To Appear Feb. 1998.
- [J2] G.E. Hovland and B.J. McCarragher, Control of Sensory Perception in Discrete Event Systems Using Stochastic Dynamic Programming, *ASME Journal of Dynamic Systems, Measurement and Control*, Submitted July 1997.
- [J3] G.E. Hovland and B.J. McCarragher, Control of Sensory Perception in a Mobile Navigation Problem, *The International Journal of Robotics Research*, Submitted August 1997.

Conference Papers:

- [C1] G.E. Hovland and B.J. McCarragher, Controlling Sensory Perception for Indoor Navigation, *The 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, Submitted August 1997.
- [C2] G.E. Hovland and B.J. McCarragher, Sensitivity Analysis of a Sensory Perception Controller, *Proceedings of Control 97*, Sydney, 20-22 October 1997.

- [C3] G.E. Hovland and B.J. McCarragher, Combining Force and Position Measurements for the Monitoring of Robotic Assembly, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, 8-13 September 1997.
- [C4] G.E. Hovland and B.J. McCarragher, Dynamic Sensor Selection for Robotic Systems, *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, 20-25 April 1997, pp. 272-277.
- [C5] G.E. Hovland and B.J. McCarragher, Control of Sensory Perception Using Stochastic Dynamic Programming, *Proceedings of the 1st Australian Data Fusion Symposium*, Adelaide, 21-23 November 1996, pp. 196-201.
- [C6] G.E. Hovland and B.J. McCarragher, Frequency-Domain Force Measurements for Discrete Event Contact Recognition, *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, 22-28 April 1996, pp. 1166-1171.
- [C7] G.E. Hovland and B.J. McCarragher, State Transition Recognition in Robotic Assembly Using Hidden Markov Models, *Proceedings of the 1995 National Conference of the Australian Robot Association*, Melbourne, 5-7 July 1995, pp.75-86.
- [C8] G.E. Hovland and B.J. McCarragher, A Hidden Markov Approach to the Monitoring of Robotic Assembly, *Proceedings of the 6th Irish DSP and Control Colloquium*, Queen's University Belfast, Belfast, 19-20 June 1995 (F. Gaston and G. Dodds eds).

In addition to the above papers, the following papers whose contents do not directly relate to the material in the thesis were published as joint work with other members of the Automated Systems Laboratory Group in the Department of Engineering, ANU.

- [E1] B.J. McCarragher, G. Hovland, P. Sikka, P. Aigner and D. Austin, Hybrid Dynamic Modelling and Control of Constrained Manipulation Systems, *IEEE Robotics and Automation Magazine*, June 1997 Special Issue on Discrete Event Systems.
- [E2] G.E. Hovland, P. Sikka and B.J. McCarragher, Skill Acquisition from Human Demonstration Using a Hidden Markov Model, *Proceedings of the 1996 IEEE*

International Conference on Robotics and Automation, Minneapolis, 22-28 April
1996, pp. 2706-2711.

Canberra, August 1997.

Geir Edvin Hovland

Department of Engineering

Faculty of Engineering and Information Technology

The Australian National University

Canberra ACT 0200, AUSTRALIA.

Acknowledgements

First I give my sincere thanks to my supervisor Brennan McCarragher for his continuous support during my time at the ANU. He always made sure I kept the bigger picture with me, even when I was buried in details. His guidance, encouragement, comments and suggestions were eye-openers and his influence will follow me in the future.

The atmosphere of room 121 in the Engineering Department was always special. My office-mates Peter Aigner and David Austin were endless sources of humour and enthusiasm. In the early stages of the project I was lucky to work with Pavan Sikka and his friendship helped me to adapt to a new country and working environment. I had the best possible sounding-board for my research and source of ideas from former and current members of the Automated Systems Laboratory, in particular Peter Aigner, Pudji Astuti, David Austin, Aidan Cahill, Werner Kraus, Paul Logothetis and Pavan Sikka. Other people who made my stay in the department interesting and enjoyable were Paul Compston, Peter Dower, Deshan Lu and Rajmohan Madhavan. A big thank you to you all.

I want to thank the members of my thesis committee, Jon Kieffer and Iven Mareels. Also, I acknowledge the financial support from the Research Council of Norway which allowed me to spend these rewarding years in Australia.

G.E.H

August 1997

Abstract

The problem of controlling sensory perception for use in discrete event feedback control systems is addressed in this thesis. The sensory perception controller (SPC) is formulated as a sequential Markov decision problem. The SPC has two main objectives; 1) to collect perceptual information to identify discrete events with high levels of confidence and 2) to keep the sensing costs low. Several event recognition techniques are available where each of the event recognisers produces confidence levels of recognised events. For a discrete event control system running in normal operation, the confidence levels are typically large and only a few event recognisers are needed. Then, as the event recognition becomes harder, the confidence levels will decrease and additional event recognisers are utilised by the SPC. The final product is an intelligent architecture with the ability to actively control the use of sensory input and perception to achieve high performance discrete event recognition.

The discrete event control framework is chosen for several reasons. First, the theory of discrete event systems is applicable to a wide range of systems. In particular, manufacturing, robotics, communication networks, transportation systems and logistic systems all fall within the class of discrete event systems. Second, the dynamics of the sensing signals used by the event recognisers are often strong and contain a large amount of information at the occurrence of discrete events. Third, because of the discrete nature of events, feedback information is not required continuously. Hence, valuable processing time is available between events. Fourth, the discrete events are a natural common representational format for the sensors. A common sensor format aids the decision process when dealing with different sensor types. Fifth, the sensing aspect of discrete event systems has often been neglected in the literature. In this thesis we present a

unique approach to on-line discrete event identification.

The thesis contains both theoretical results and demonstrated real-world applications. The main theoretical contributions of the thesis are 1) the development of a sensory perception controller for the dynamic real-time selection of event recognisers. The proposed solution solves the Markov decision process using stochastic dynamic programming (SDP). SDP guarantees cost-efficiency of the real-time SPC by solving a sequential constrained optimisation problem. 2) A sensitivity analysis method for the sensory perception controller has been developed by exploring the relationship between Markov decision theory and linear programming. The sensitivity analysis aids in the robust tuning of the SPC by finding low sensitivity areas for the controller parameters.

Two real-world applications are presented. First, several event recognition techniques have been developed for a robotic assembly task. Robotic assembly fits particularly well in the discrete event framework, where discrete events correspond to changes in contact states between the workpiece and the environment. Force measurements in particular contain a significant amount of information when the contact state changes. Second, the sensory perception control theory and the sensitivity analysis have been demonstrated for a mobile navigation problem. The cost-efficient use of sensory perception reduces the need for mobile robots to carry heavy computational resources.

Contents

Declaration	i
Acknowledgements	iv
Abstract	v
Nomenclature	xi
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Organisation of the Thesis	5
2 Formulation of the Sensory Perception Control Problem	7
2.1 Discrete Event Formalism	7
2.2 Literature Survey	10
2.2.1 Open-Loop Sensor Planning Solutions	10
2.2.2 On-Line Dynamic Sensor Selection	12
2.3 Discounted Markov Decision Process	15
3 Stochastic Dynamic Programming Solution for Real-Time Implementation	18
3.1 Introduction	18
3.2 Sensory Perception Controller	20

3.2.1	Stochastic Dynamic Programming Model	22
3.2.2	Stochastic Dynamic Programming Algorithm	26
3.2.3	Limitation on Total Cost	29
3.2.4	Data Fusion	29
3.2.5	Properties of the Sensory Perception Controller	32
3.3	Sensory Perception Control Example	34
3.4	Evaluation	39
3.4.1	Advantages	40
3.4.2	Limitations	41
3.5	Conclusion	42
4	Linear Programming Solution to Sensitivity Analysis	43
4.1	Introduction	43
4.2	Linear Programming Solution	44
4.2.1	Linear Programming Equations	45
4.2.2	Linear Program in Standard Form	46
4.3	Sensitivity Analysis	47
4.4	Sensitivity Analysis Example	50
4.4.1	Sensitivity of the Discount Factor α	51
4.4.2	Sensitivity of the State Reward Constant K	54
4.4.3	Sensitivity of the State Rewards $r(s_i, a_1)$	56
4.4.4	Sensitivity of the Monitor Rewards l_i	57
4.5	Discussion and Conclusion	58
5	Case Study I: Control of Sensory Perception in Robotic Assembly	60
5.1	Introduction	60
5.2	Literature Survey	61

5.3	Discrete Event Model of Robotic Assembly	63
5.4	Discrete Event Control Architecture	66
5.5	Process Monitors	68
5.5.1	Geometric Modelling	68
5.5.2	Event Detection	69
5.6	Experiments	70
5.7	Discussion and Conclusions	77
6	Case Study II: Control of Sensory Perception in Mobile Navigation	79
6.1	Introduction	79
6.2	Literature Survey	81
6.3	Discrete Event Model of Navigation Problem	82
6.4	Process Monitoring Techniques	85
6.5	Experiments on Sensory Perception Control	87
6.5.1	Experiments with Closed Doors	89
6.5.2	Experiments with Open Doors	91
6.6	Experiments on Sensitivity Analysis	95
6.6.1	Sensitivity of the Discount Factor α	95
6.6.2	Sensitivity of the State Reward Constant K	97
6.7	Conclusions	98
7	Conclusions and Further Research	99
7.1	Introduction	99
7.2	Major Conclusions and Results	100
7.3	Further Research	101
7.3.1	Imperfect Event Recognition	101
7.3.2	Extension to More Complex Tasks	102

7.3.3	Continuous Control of Sensory Perception	103
7.3.4	Analytical Research	104
7.3.5	Parallel Decision Making	104
References		105
A Process Monitoring Using Hidden Markov Models		112
A.1	Introduction	112
A.2	Observation Symbols	114
A.3	Length of Observation Window	116
A.4	Training	117
A.5	Event Recognition	121
A.6	Evaluation	122
B Process Monitoring Using Distance Functions		123
B.1	Distance Functions	123
B.2	Classification of Contact States	124
B.3	Evaluation	126
C Process Monitoring Using a Multilayer Perceptron		128
C.1	Multilayer Perceptron Network	128
C.2	Training Algorithm	129
C.3	Real-Time Operation	130
C.4	Evaluation	131
D Process Monitoring Using Qualitative Reasoning		133
D.1	Qualitative Reasoning	133
D.2	Confidence Levels	134
D.3	Evaluation	135

Nomenclature

Acronyms

DEC	: Discrete Event Controller
DES	: Discrete Event System
DF	: Distance Functions
HMM	: Hidden Markov Model
LP	: Linear Programming
MLP	: Multilayer Perceptron
QR	: Qualitative Reasoning
SDP	: Stochastic Dynamic Programming
SPC	: Sensory Perception Controller

Symbols

A	: LP constraint matrix in standard form
$a_k \in \mathcal{A}$	: Markov action space
$a_n(z, s_i)$	: Optimal action function
b	: LP constraint vector in standard form
c	: LP cost vector
C	: Confidence level
e_i	: Discrete event
$e^*(n)$	: Recognised event by the SPC after the n th action
(F_x, F_y, M_z)	: Planar force/torque measurements
$(\Delta F_x, \Delta F_y)$	: Change of planar force measurements
K	: SDP state reward constant
k_i	: Individual monitor costs
l_i	: Individual monitor rewards

$\mathbf{N}$	: Submatrix of $\mathbf{A}$ corresponding to non-basic LP variables
N_m	: Number of available process monitors
$P_{n,ij}(z, a)$	: SDP state transition probability
(P_x, P_y, θ)	: Planar position measurements
$R_n(z, s_i, a_k)$	: Markov reward function
$r(s_i, a_1)$	: Markov state rewards
$s_i \in \mathcal{S}$	: Markov state space
$V_n(z, s_i)$	: Optimal value function
$\mathbf{x}$	: LP state vector in standard form
$\mathbf{x}^*$	: Optimal basic feasible LP solution
$\mathbf{x}_B$	: LP basic variable vector
$\mathbf{x}_N$	: LP non-basic variable vector
z	: Permutation of process monitors
z^*	: Optimal permutation of process monitors
α	: Discount factor
γ_i	: State describing DES
γ^*	: Final recognised DES state
κ	: Total accumulated SDP cost
$\lambda(s_i, a_k)$	: LP state variables
μ	: Confidence level discretisation function
$\sigma(n)$	: State of SPC after the n th action
τ	: Map from discrete events to DES state
ϕ	: Edges of geometric model
Φ	: Sensitivity analysis parameter
ψ	: Surfaces of geometric model

Chapter 1

Introduction

1.1 Motivation

Perceptual capabilities are often the main bottleneck for successful operation of discrete event autonomous systems. As the degree of system uncertainty and ambiguity increases, one sensor alone may not provide sufficient information. Research in the area of multi-sensor fusion has increased the robustness of systems operating in uncertain environments. However, these solutions often require high computational power by utilising all the sensors continuously. As the tasks become more complex and more numerous, the number of relevant features of the environment quickly exceeds the sensing and processing resources that are feasible to supply to real-world control systems. Hence, there is an increasing need for controlling sensory perception. An intelligent sensory perception system is able to perform time-constrained on-line analysis of autonomous systems. Typically, in normal operation only a few sensors are needed. Then, as the quality of the sensing decreases, more sensors are utilised. The design of a sensory perception controller is an important contribution to the development of discrete event autonomous systems operating in uncertain and unpredictable environments.

The control of sensory perception fits particularly well in the discrete event framework. The traditional approach to the control of sensory perception in feedback control systems requires heavy continuous computations during execution. The continuous computations often outweigh the actual sensing costs and the practical value of such

solutions is limited. In the discrete event formalism, the control of sensory perception is only needed at the occurrence of the events. Hence, valuable processing time is available between the occurrence of events for other components of the discrete event control architecture.

A wide range of different sensor types have been used in autonomous control systems. The sensory perception models and sensor selection strategies often depend on the specifics of the sensor types used in each application. Hence, transferring these methods to other systems with different sensor types is often difficult. For example, active vision research has focussed on planning using vision sensors, such as cameras, range finders and illuminators. A host of other sensors such as tactile sensors, three-dimensional range sensors, force-torque sensors and acoustic sensors are currently being used in, for example, robotics and manufacturing applications. Tarabanis, Allen and Tsai (1995) surveyed the area of vision-based systems and write: *Further work needs to be done to properly integrate these sensors and their unique constraints into the overall planning system.*

One advantage of the discrete event framework is the common representational format provided for different sensor types. The natural common format is the discrete events occurring during execution. The general sensory perception control theory presented in this thesis is applicable to a wide range of sensor types in different discrete event control systems. In particular, the discrete event framework has proven successful in manufacturing problems, robotic assembly, mobile robot navigation, communication networks, transportation systems and logistic systems.

The control of discrete event systems has received a significant amount of attention in recent years, see for example Baccelli, et.al. (1992), Cassandras (1993), Cassandras, Lafortune and Olsder (1995), Rubinstein and Shapiro (1993). Traditionally, perfect sensing of discrete events has been assumed. In other work, for example Kumar and Garg (1995), Özveren and Willsky (1990), the requirement of perfect event recognition is reduced to a sub-set of all possible discrete events. State ambiguities are allowed to develop, but these must be resolvable after a bounded interval of events. For most practical systems the assumption of perfect event sensing is unrealistic. To increase applicability of discrete event theory, there is a need for dealing with the sensing aspects

of discrete events. Discrete event identification is one of the main difficulties with interfacing continuous-time systems with discrete event controllers.

In this thesis we present a new and unique approach to discrete event identification. A sensory perception controller actively selects different event monitors to identify discrete events as they occur with high levels of confidence and low sensing costs. The method improves the applicability of discrete event theory by increasing the event recognition rates compared to single sensor systems. The total sensing costs are low compared to multi-sensor fusion techniques where all the sensors are used for every event occurrence. Hence, the proposed method is well suited to event identification in real-time feedback control systems operating in uncertain environments.

1.2 Contributions

The thesis contributes in the following areas:

- i. The sensory perception controller (SPC) is formulated as a sequential Markov decision problem in a discrete event feedback control system. The Markov formulation allows for fast-real time solutions and provides a facility for sensitivity analysis. The discrete event formalism offers advantages in providing a common representational format for different sensor types, highlighting sensing at the occurrence of discrete events where the signal dynamics are often strong and reducing the sensing costs by avoiding continuous control of sensory perception.
- ii. A fast real-time SPC algorithm has been developed solving the Markov decision process using stochastic dynamic programming (SDP). SDP is used to determine the optimal sequence of sensing strategies. The SDP algorithm provides a cost-efficient solution to the sensory perception control problem. A low-cost SPC is required to improve the performance compared to existing multi-sensor fusion algorithms.
- iii. A thorough sensitivity analysis of the controller has been developed. The sensitivity analysis aids in the robust tuning of the SPC by finding low sensitivity areas for the model parameters. The sensitivity analysis is performed by exploring the

relationship between discounted Markov decision problems and linear programming. Linear programming allows for a thorough sensitivity analysis without having to solve the problem from scratch for each new model parameter.

- iv. The last decade has seen little effort in applying Markov decision theory to practical systems. In this thesis the sensory perception controller has been successfully implemented in two applications; robotic assembly and mobile navigation. The cost of sensing is reduced compared to multi-sensor systems where all the sensors are used all the time. The experiments also demonstrate that the sensory perception controller achieves higher event recognition rates than any individual sensor. Moreover, the experiments show how the sensing system is able to recover from undetected discrete events.
- v. The applications required the development and implementation of several process monitoring techniques. The appendices present new and unique techniques for discrete event recognition based on force/torque and position measurements. Hidden Markov Models (HMMs) have proven to be a powerful tool for discrete event recognition. The method 1) allows for dynamic force/torque measurements, 2) accounts for sensor noise and friction and 3) exploits the fact that the amount of force information is large at the occurrence of the events.

A monitor using position measurements has been developed. The position based monitor is sensitive to geometrical model uncertainties, but still adds useful information to the sensory perception controller. The main advantage of the method is its computational efficiency.

A multilayer perceptron (MLP) network using both force/torque and position measurements for event recognition has been implemented. One advantage of this method compared to the other solutions presented is the fact that it models both dynamic and static behaviour. The MLP is able to achieve relatively large successful event recognition rates compared to other methods.

1.3 Organisation of the Thesis

The thesis contains both theoretical results and two demonstrated real-world applications. It is comprised of the following chapters.

Chapter 2 formulates the problem of sensory perception control. The discrete event framework is used and the sensory perception controller is formulated as an active element of a feedback control structure. The chapter contains a literature survey of research relevant to the sensory perception problem and concludes with a formulation of the SPC as a sequential Markov decision process.

Chapter 3 presents the stochastic dynamic programming solution to the Markov decision process. The solution provides a fast algorithm for use in real-time feedback control systems. Several different data fusion methods are incorporated and a sensory perception control example is given. The chapter ends with a discussion of some advantages and limitations of the proposed method.

Chapter 4 examines the sensitivity analysis of the controller developed in Chapter 3. Due to the iterative nature of dynamic programming, a direct sensitivity approach is difficult. The sensitivity analysis is performed by exploring the relationship between discounted Markov decision problems and linear programming.

Chapter 5 demonstrates the control of sensory perception for a planar robotic assembly task. The assembly task is modelled as a constrained motion system and the discrete events correspond to changes in the motion constraints. The control of sensory perception is demonstrated using four different event recognition techniques.

Chapter 6 demonstrates the control of sensory perception in mobile navigation. The navigation problem is modelled as a discrete event system, where the discrete events correspond to changes in the mobile unit motion constraints. The sensory perception problem is demonstrated for a planar model containing three rooms, fixed walls and open or closed doors.

Chapter 7 brings the conclusions of the thesis. Open problems and areas for further research are discussed.

The appendices present several process monitors which allow us to demonstrate the control of sensory perception for robotic assembly and mobile navigation in Chapter 5 and 6.

Appendix A presents a process monitor based on Hidden Markov Models (HMMs).

Each discrete event is modelled by a HMM which represents a stochastic, knowledge-based system. The HMMs are trained off-line on planar force/torque measurements. In real-time operation all HMMs corresponding to possible events are evaluated. The event with the highest model score is chosen and the associated confidence level is also calculated from the HMM model scores.

Appendix B presents a process monitor based on position measurements and distance functions. The monitor is based on a geometrical model of the world and calculates the nearest distances between all relevant surfaces and edges. The monitor recognises the world states and calculates confidence levels based on the distance functions.

Appendix C describes a multilayer perceptron process monitor. The network is trained off-line on force/torque and position measurements. Each network output corresponds to a discrete event. In real-time operation the measured forces/torques and positions are used as the network inputs. The event with the highest corresponding network output is chosen and the associated confidence level is calculated from the network outputs.

Appendix D describes a process monitor based on qualitative reasoning. The control of sensory perception requires each monitor to produce confidence level outputs of the recognised discrete events. The monitor based on qualitative reasoning was developed by McCarragher and Asada (1993). The original contribution in this appendix is the incorporation of confidence levels to the process monitor.

Chapter 2

Formulation of the Sensory Perception Control Problem

2.1 Discrete Event Formalism

The sensory perception control problem addressed in this thesis is formulated in the discrete event control framework. Ramadge and Wonham (1989) defined a discrete event system (DES) as a *dynamic system that evolves in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of physical events.*

Figure 2.1: Sample path of a discrete event system.

Figure 2.1 shows a sample path of a discrete event system. The DES state variable, denoted $\gamma(t)$, is piece-wise constant and can only change value at the occurrence of a discrete event. The sensory perception controller (SPC) has two main objectives; 1) to collect perceptual information to identify discrete events with high levels of confidence and 2) to keep the sensing costs low. The general sensory perception control problem for discrete event systems is formulated as follows.

Problem Statement: *Given the occurrence of a discrete event, consult event monitors such as to minimise the cost related to the event recognition error plus the costs of obtaining the monitor outputs, ie. find the optimal V*

$$V = \min f(e^* - e) + \sum_{i \in \mathcal{I}} g(k_i) \quad (2.1)$$

The term $\sum g(k_i)$ represents the cost of obtaining the monitor outputs, where k_i are the individual monitor costs. The individual monitor costs are usually fixed and determined off-line. Only the event monitors actually consulted by the SPC are contained in the set $\mathcal{I}$. The term $f(e^* - e)$ represents the cost related to the event recognition error, where e^* is the recognised event from the SPC. In general, the correct event e is unknown. Hence, the event recognition error $e^* - e$ is also unknown. In this thesis, each individual event monitor produces a confidence level $C \in [0, 1]$ which is used to estimate the event recognition error. A large confidence level indicates a low recognition error. A trade-off has to be made between the event recognition errors and the monitor costs. Low monitor costs often result in large average event recognition errors, while large monitor costs often result in low average event recognition errors.

Figure 2.2 shows the block diagram of the discrete event control structure. The perceptual capabilities of the discrete event system consist of several process monitors. Process monitor i recognises a discrete event $e^i(t_k)$ occurring at time t_k . Due to noisy measurements, model uncertainties or world unpredictability, the recognised event $e^i(t_k)$ may not correspond to the actual physical event $e(t_k)$ that occurred. A very important feature of a process monitor is its ability to indicate the confidence level of the recognised event. A good process monitor produces low confidence levels for events recognised incorrectly and large confidence levels for events recognised correctly.

Figure 2.2: *Sensory perception control system set in a discrete event framework.* $\gamma^*(t_k)$ is the final recognised discrete state from the sensory perception controller, $C^i(t_k)$ and $e^i(t_k)$ are confidence levels and recognised discrete events from process monitor i , $\mathbf{v}(t_k)$ is the discrete event controller command vector and $\mathbf{z}(t)$ is the continuous process plant measurement vector.

When a discrete event occurs, the sensory perception controller (SPC) has the option of consulting any of the process monitors. The SPC has two main objectives. First, the SPC must use the recognised events $e^i(t_k)$ and the corresponding confidence levels $C^i(t_k)$ efficiently to correctly recognise the actual discrete events. Even when some of the $e^i(t_k)$ are incorrect, a robust SPC is able to recognise the correct event. Second, the SPC must keep the perceptual costs low. In real-world applications computational resources are limited and have to be shared between the perceptual system and the other parts of the control system.

The SPC maps the final recognised event $e^*(t_k)$ to a final recognised DES state $\gamma^*(t_k)$ which is sent to the discrete event controller. The discrete event controller changes its command vector $\mathbf{v}(t_k)$ depending on $\gamma^*(t_k)$. The command vector depends on the process plant, but typically contains reference commands and parameters for low-level process plant controllers. The command vector $\mathbf{v}(t_k)$ will drive the process plant until the next discrete event occurs. The sensory perception control system is then activated again.

A strict requirement of any SPC is low real-time overhead cost. The control of sensory perception becomes meaningless if the overhead cost outweighs the actual sensing costs. The control of sensory perception fits particularly well in the discrete event control framework and offers several advantages. Because of the discrete nature of events, the control of sensory perception is not required continuously, see Figure 2.1. Hence, valuable processing time is available for other parts of the control system between the occurrence of events. Moreover, at the occurrence of discrete events the dynamics of the sensing signals used by the event recognisers are often strong and contain a large amount of information. Further, the discrete events are a natural common representational format for the sensors. A common sensor format aids the decision process when dealing with different sensor types.

2.2 Literature Survey

Research related to the control of sensory perception has been addressed by several different research communities. In this section we discuss some of the solutions available in the literature. Section 2.2.1 presents open-loop sensor planning solutions. Section 2.2.2 presents dynamic sensor selection strategies where the sensor strategies depend on sensor information obtained during execution. None of the solutions presented in the literature deal with sensory perception in discrete event feedback control problems. The issue of discrete event sensing has often been ignored in discrete event systems research. The development of a closed-loop sensory perception controller for use in the DES framework is the main contribution of this thesis compared to existing solutions in the literature. Moreover, the presented sensory perception controller shares one advantage with the surveyed papers in on-line dynamic sensor selection. The sensor selection strategies depend on both *apriori* expected behaviour and measurements obtained during execution.

2.2.1 Open-Loop Sensor Planning Solutions

Traditionally, the sensor planning and execution parts of intelligent systems have been separated. The sensing is only based on the *apriori* expected behaviour of the sensors.

Such planners are referred to as open-loop planners and examples have been presented by Dean, Basye and Lejter (1991), Dean, Camus and Kirman (1990). The sequential decision problem is modelled as a Bayes network and planning is defined as the process of generating and evaluating different actions in terms of their off-line predicted consequences so as to choose among them.

Chrisman and Simmons (1991) outline an approach to the construction of plans containing explicit sensing operations with the objective of finding cost efficient plans with respect to both action and sensing. Markov decision theory is used to find an optimal sense-act plan given a particular static sensing policy. The sensing operation in each world state is fixed. However, the sequence of the world states is unpredictable and, as a result, the sequence of sensing operations is not fixed. The advantage of a fixed sensing policy is a reduction of the planning complexity. One disadvantage with pure static sensing is that potential sensing behaviour is restricted.

An early work in the area of open-loop sensor selection for use in feedback control was presented by Meier, Peschon and Dressler (1967). The control of a measurement subsystem in a stochastic system using a dynamic programming approach was considered. The method is developed for process plants with a known non-linear, stochastic model. Optimal control vectors to the plant and the measurement subsystem are calculated. The problem is shown to be a generalisation of the combined optimisation problem. The method can be applied to fixed process plants with known models. The problem solution is calculated off-line.

A survey of sensor planning in computer vision was presented by Tarabanis, Allen and Tsai (1995). Most of the surveyed papers are based only on *a priori* information about the environment, the sensors and the task. Sensor parameters are calculated off-line and then used on-line for the execution of the tasks. Three distinct areas of research are identified; object feature detection, model-based object recognition and localisation, and scene reconstruction. Strategies that automatically determine sensor parameter values that achieve these tasks with a certain degree of satisfaction are developed.

For all these methods the sensing is only based on *apriori* information and the sensory information obtained during execution has no influence on the real-time decisions. Hence, in autonomous control systems such fixed sensing schemes often fail to achieve

the desired goals.

2.2.2 On-Line Dynamic Sensor Selection

Research in on-line dynamic sensor selection has been undertaken in several different communities. Here we present solutions from closed-loop planning, feedback control theory, decision fusion and active vision.

AI Closed-Loop Planning

To overcome the limitations of traditional open-loop planners, closed-loop planners and sensory perception controllers have been proposed by several authors. Chrisman (1992) presented a closed-loop planner based on Markov decision process theory. The closed-loop plans contain 1) action choices that depend on sensory information obtained during execution and 2) sequences of actions which serve to gather information upon which to base future decisions. The closed-loop planner is able to anticipate and plan for a large set of possible outcomes; even outcomes considered by traditional planners as failures. One limitation of closed-loop planning is the large computational complexities involved.

Zilberstein (1996) presented a different solution where both the sensing and the planning were implemented as anytime algorithms. An anytime algorithm is able to produce answers at any time, but quality of the answers increase with time. Run-time monitoring is used to determine the optimal time allocation to both sensing and planning. One advantage of the method is the capability to operate in real-time with limited computational resources. One limitation is the requirement of anytime sensing algorithms.

Feedback Control Systems

Oshman (1994) presented a sensory perception controller which can be implemented on-line in time-varying linear systems. At each measurement epoch one of the available measurement subsystems is utilised. The algorithm continuously provides the spectral factors of an estimation error covariance matrix, which are used as the basis for the

real-time selection of the measurement subsystems. The method is limited to linear systems and is not applicable to discrete event identification. Moreover, the SPC runs continuously and performs matrix calculations such as singular value decomposition. Hence, the SPC significantly reduces the bandwidth of the overall control loop.

A closed-loop control system with a countable set of predictors was presented by Kulkarni and Ramadge (1996). Each predictor p produces an output estimate $\hat{y}_p(t)$ of a fixed but unknown system. Performance measures are derived from the resultant prediction errors $\hat{y}_p(t) - y(t)$. The authors have developed real-time switching laws for the dynamic selection of predictors. The method applies to non-linear continuous control systems, but is not applicable to discrete event systems. Again, the continuous model formulation causes the SPC to reduce the bandwidth of the overall control loop.

A different approach to the control of sensory perception was presented by Aidarous, Gevers and Installé (1975), where optimal sensor locations were obtained for a class of linear systems. The method uses a state error covariance matrix which is updated during execution to calculate the optimal locations of the movable sensors. The approach requires a Ricatti equation to be solved during execution.

The traditional approach to the control of sensory perception in feedback control systems requires continuous computations during execution. In this thesis we present a new approach using the discrete event formalism. The SPC is only activated at the occurrence of discrete events. Between the occurrence of events, the discrete event control system runs without any interference from the SPC. Moreover, the costs incurred by continuous operations such as solving Ricatti equations and matrix decompositions often outweigh the actual sensing costs. Hence, the practical value of such solutions is limited. In this thesis the SPC is based on both *a priori* expected behaviour and real-time measurements. The cost of the SPC is reduced by performing the heavy computations off-line.

Decision Fusion

The research area of decision fusion deals with decision making in uncertain environments with several identical or different sensor types. An extensive survey containing

reprints of 25 selected papers in the area was presented by Dasarathy (1994). Most of the work presented deals with detection problems with distributed sensors, where the dominant theoretical framework is Bayesian hypothesis testing. However, most of the work does not consider the cost of collecting and processing sensory information. Hence, these methods can significantly reduce the bandwidth when used in real-time control systems.

Among the first studies of communication costs and consulting decision makers was the paper by Papastavrou and Athans (1986), where a version of the second-opinion problem is analysed. A primary decision maker has the option of soliciting, at a cost, the opinion of a consulting decision maker when faced with uncertain measurements. The solution is deterministic and given by a likelihood ratio test.

Another example was presented by Sakaguchi (1994) where the dynamic sensor selection was demonstrated for two haptic sensing systems. Sensors such as a thermo-sensor, a pressure sensor, a vibration sensor and a piezoelectric sensor were used in the experiments. One example given is the classification of the type of material, for example aluminium, wood or rubber using these sensors. Each sensor uses a pre-processor to classify the material type and the author uses sequential Bayesian decision theory to actively select sensors and fuse the individual decisions. The author considers the entropy of the model space. Observations are repeated until the entropy becomes smaller than a threshold value. At any time in the decision process, the sensor which is expected to most reduce the entropy is selected. The method can be categorised as a dynamic sensor selection method since the posterior probability distributions and the entropy used in the decision process are continuously updated using the sensor measurements.

Active Vision

Most applications utilising control of sensory perception are found in active vision systems. Active vision research often deals with only one sensor, the camera, but parameters such as the viewpoint, the focus and the zoom are actively controlled. The book by Aloimonos (1993) describes various topics including navigation, obstacle avoidance and hand-eye coordination. The book contains a large amount of references to control of sensory perception in active vision.

Rimey and Brown (1994) presented a selective vision system which sequentially collects evidence to answer a specific question with a desired level of confidence using Bayes nets. Only selected areas of the images are processed to answer the specific questions and thus reducing the overall sensing costs required.

Hager (1990) presented a sensor fusion and planning strategy based on Bayesian decision theory. System states are represented by probability distributions and the decision theory accounts for the on-line sensor measurements by using Bayes rule to update the probability distributions. The optimal decisions are found by trading-off computational sensing costs and value of information. Experiments in active camera positioning and tracking are presented.

In the paper by Bajcsy (1988) the control strategies are formulated as a search of a sequence of steps that would minimise a loss function while maximising the information collection using sequential Bayes decision theory. The author discusses several examples of active perception, such as obtaining depth maps and 2-D image segmentation.

A method for determining camera viewpoint optimality was presented by Trucco et.al. (1997). Viewpoint optimality is defined as a function of feature visibility and measurement reliability. Optimal sensor positioning plans are generated on-line and a complete inspection session is demonstrated involving 3-D object positioning, planning optimal position inspection and feature measurement from the optimal viewpoint.

A different approach for determining camera viewpoints based on Dempster-Shafer theory was presented by Hutchinson and Kak (1989). The approach is to automatically propose a sensing operation, and then to determine the maximum ambiguity that would remain in the world description if that sensing operation were applied. The system then applies the sensing operation which minimises this ambiguity.

2.3 Discounted Markov Decision Process

The past decade has seen a notable development in both applied and theoretical results on Markov decision processes, Puterman (1994) and White (1993). However, in that period relatively little effort was put in applying the quite useful Markov decision

model to practical problems, Tijms (1994). In this thesis we formulate the sensory perception control problem as a discounted Markov decision problem. The theory has been successfully applied to both robotic assembly and mobile navigation.

Markov decision process theory has two main advantages compared to other sequential decision theories. First, the optimal actions at any decision epoch are independent of the actions at previous decision epochs. This property allows for efficient decision making algorithms, such as stochastic dynamic programming. Second, a thorough sensitivity analysis is possible by exploring the relationship between Markov decision processes and linear programming.

The general sequential decision problem and the underlying Markov decision process are illustrated in Figures 2.3 and 2.4.

Figure 2.3: *Sequential Decision Problem.*

Figure 2.4: *Markov Decision Process.*

At decision epoch n the decision process is in state $\sigma(n) \in \{s_1, s_2, \dots, s_{N_s}\}$. The SPC selects one action $a(n) \in \mathcal{A}$ and a reward $R(\sigma(n), a(n))$ is given, depending on the state $\sigma(n)$ and the action $a(n)$. The outcome of the action $a(n)$ is, in general, unpredictable

and determines the next Markov state $\sigma(n + 1)$. The unpredictable outcomes are modelled by the state transition probabilities $P_{ij}(a)$ in the Markov model. In this thesis the decision maker seeks to optimise the following criterion for each Markov state s_i .

$$V_N(s_i) = \max E_{\pi} \left[\sum_{n=1}^N R(\sigma(n), a(n)) \alpha^{n-1} \mid \sigma(0) = s_i \right] \quad (2.2)$$

where E_{π} is the expectation operator and π is a policy for choosing the actions $a(n)$ at decision epoch n . N is the total number of decision epochs. The factor $\alpha \in [0, 1]$ discounts future expected rewards. Due to the unpredictable outcomes of the actions $a(n)$, future rewards are worth less than immediate rewards. Equation (2.2) is a general criterion defined by Ross (1983). The criterion is equivalent to equation (2.1) where the rewards are equivalent to negative costs.

Another common optimality criterion in Markov decision theory is the long-run average reward per unit time. The average reward function is generally written on the form

$$g(s_i) = \lim_{n \rightarrow \infty} \frac{1}{n} V_n(s_i) \quad (2.3)$$

where V_n has the form in equation (2.2) without discounting ($\alpha = 1$). The long-run average reward criterion is best suited to problems with long or infinite decision horizons. For the sensory perception control problem the decision horizon is usually small and given by the number of available event monitors. Hence, in this thesis we confine ourselves to the discounted reward criterion in equation (2.2).

In the following chapters the general Markov decision theory is applied to the sensory perception control problem. In Chapter 3 a real-time SPC is presented based on stochastic dynamic programming (SDP). SDP provides a computationally efficient solution to Markov decision processes. In Chapter 4 we present a sensitivity analysis method of the SPC by solving the Markov decision process using linear programming techniques. The sensitivity analysis aids in the robust tuning of the SPC parameters.

Chapter 3

Stochastic Dynamic Programming Solution for Real-Time Implementation

3.1 Introduction

In this chapter we present a sensory perception controller (SPC) solving the Markov decision process using the theory of stochastic dynamic programming (SDP). SDP provides cost-efficient solutions to sequential multi-stage decision problems. Hager (1990) and Sakaguchi (1994) describe brute-force and tree search methods used on-line to determine optimal sensor sequences using Bayesian decision theory. The improved performance of dynamic programming compared to these methods comes from solving multi-stage problems by analysing a sequence of simpler inductively defined single-stage problems. To be of any practical value, the extra cost by the SPC can not outweigh the actual sensing costs. Otherwise, a more cost-efficient solution would be to use all available sensors and multi-sensor fusion techniques. Hence, computationally efficient methods such as dynamic programming are required.

Many feedback control systems have bandwidth requirements. We show how an upper limit of available CPU time is incorporated into the SDP algorithm. Given the upper limit of CPU time, the SDP algorithm solves the constrained optimisation problem.

The final solution is optimal with regard to the given monitor rewards, confidence rewards and the CPU time limit.

As an example of the use of stochastic dynamic programming for predicting and evaluating different actions see Brown and Whitney (1994), where planning of robot grinding tasks is studied. Measurements of the volume of weld material and time remaining in the grinding task are discretised, and a state in a discrete state space which most nearly approximates the measurements is chosen. From this state all possible actions are evaluated using stochastic dynamic programming, and the recommended action from the first pass is executed. Due to the stochastic nature of grinding, this state may or may not correspond to the expected result from the first control action. Regardless, one again looks up the state most nearly corresponding to the measurement and again reads off the action and executes it. Dynamic programming provides an efficient tool for evaluating future expected rewards of different actions. In our work the SDP algorithm evaluates future expected rewards for all possible sequential consultations of event monitors.

Another example of the use of dynamic programming in a control system is found in Shin and McKay (1986). A geometric desired path in the manipulator's joint space is specified a priori. The DP algorithm is used off-line for trajectory planning and provides solutions for the manipulator's joint positions, velocities and accelerations. These variables are then used on-line as references for the real-time control problem. A cost function is defined which is minimised subject to the manipulator's dynamics and limits on the joint torque derivatives. The method is able to handle general cost functions and constraints. For the sensory perception problem the dynamic programming is applied in a similar fashion. Optimal value functions and optimal actions are calculated off-line. The dynamic programming is combined with on-line measurements by the real-time controller to make sensing decisions.

A problem similar to the sensory perception problem, the unit commitment problem, has been studied by power systems researchers. Examples of dynamic programming based methods for the unit commitment problem are found in Pang and Chen (1976), Pang, Sheble and Albuyeh (1981). The main effort in this area is to develop algorithms for reducing the number of combinations of units to be considered by the dynamic

programming algorithm. The number of combinations is equal to $2^N - 1$ where the number of units studied is as high as $N = 96$. The results of the research show that significant potential savings can be achieved through the use of dynamic programming based methods for unit commitment problems, though there is no assurance that they will yield optimal schedules due to the reduced number of combinations considered. For the sensory perception problem the number of units (or process monitors) is usually much less, and restricting the search space is not as important as for the unit commitment problem. Hence, finding the optimal solution is usually less costly in the sensory perception problem.

3.2 Sensory Perception Controller

The DES is divided into a discrete event control (DEC) part and a sensing part. The focus in this thesis is on the discrete event sensing part which consists of the sensory perception controller and a set of process monitors, see Figure 2.2. A process monitor recognises discrete events as they occur and the monitor produces confidence levels of its decisions. Examples of such process monitors are presented in the Appendices A,B,C,D.

Each process monitor is consulted at most once for every event occurrence. The motivation for consulting each monitor at most once is the fact that discrete events are short-lived. The dynamics of the sensing signals are strongest immediately after the occurrence of the events. The most efficient event monitoring is achieved by analysing these early dynamic signals. The extra event information gained by analysing later measurements is usually insignificant. Moreover, no extra information is gained by consulting a monitor twice on the early dynamic sensor signals.

An example of how the SPC works when three process monitors are available is given in Figure 3.1. The monitors produce confidence levels $C^i \in [0, 1]$. First, an event called e_1 occurs. The SPC decides to consult process monitor 3 first. Monitor 3 incorrectly recognises a different event, $e^3 = e_2$, and produces a confidence level of $C^3 = 0.50$. Given this relatively low confidence level, the SPC decides to continue and consults process monitor 1 next. This monitor correctly recognises event $e^1 = e_1$ and produces

Figure 3.1: *Example of SPC actions and process monitoring usage. The SPC sequentially consults monitors and receives a confidence level C^i and a recognised event e^i for each consultation.*

a confidence level of $C^1 = 0.60$. Again, given the relatively low confidence level the SPC decides to continue and finally consults monitor 2. Monitor 2 correctly recognises the event $e^2 = e_1$ with a high confidence, $C^2 = 1.00$. The SPC now terminates the discrete event sensing. The recognised event $e^* = e_1$ is mapped to a recognised discrete state γ^* which is sent to the DEC.

The example given above illustrates several important aspects of the sensory perception controller. First, the process monitors are consulted sequentially until a high confidence level is achieved or all monitors are consulted. Second, the sensor selection is based on the real-time confidence levels from the monitors. The sequential order of the process monitors is not fixed. The dynamic programming algorithm to be presented generates optimal actions for all sequences of monitors and the results are stored in a lookup table. The measured confidence levels are then used to find the optimal sequential order of monitors in the lookup table. Third, in addition to the measured confidence levels, the SPC also uses a priori information such as the cost and the estimated recognition rate of each monitor when making its decisions.

Figure 3.2 shows the relationship between sensors and process monitors. Each event monitor i uses one or more of the available sensors and a cost k_i is associated with each monitor reflecting the CPU time spent in real-time operation. Each monitor outputs a recognised discrete event e^i with an associated confidence level C^i . The sensory perception controller uses stochastic dynamic programming to produce a final recognised event e^* with confidence $C(e^*)$. The recognised event e^* is mapped to a state $\gamma^* = \tau(e^*)$ describing the current state of the discrete event system. The goal is to produce outputs with large confidences $C(\gamma^*) = C(e^*)$ while keeping the total cost

low. In normal operation only a few of the monitors are needed. Then as the event recognition becomes harder, additional monitors are utilised sequentially by the SPC.

Figure 3.2: *Relationship between sensors and process monitors. Each process monitor uses measurements from one or more sensors. The monitors classify discrete events and produce confidence levels of their decisions. The individual monitors are not always consulted by the SPC, as indicated by the dashed lines.*

In section 3.2.1 we formulate the sensory perception process as a stochastic dynamic programming problem and in section 3.2.2 we describe the stochastic dynamic programming algorithm. To allow for a limit on the total acceptable cost, a minor modification is required to the SDP algorithm as described in section 3.2.3. In section 3.2.4 we discuss the modifications to the model to allow for different data fusion algorithms. Finally, in section 3.2.5 we present some useful properties of the SPC.

3.2.1 Stochastic Dynamic Programming Model

The goal of the dynamic programming is to evaluate all possible permutations of process monitor sequences and choose actions based on the evaluations. Figure 3.3 shows the possible actions for one particular permutation of process monitors, ie. $z = 2 - 1 - 3$. The SDP algorithm is an iterative solution. The algorithm starts from the last monitor in the sequence and works itself backwards. The term *permutation* is defined as follows.

Definition 1 (Permutation) *A permutation $z \in \mathcal{Z}_n$ is an ordered sequence of process monitors of length n , where each process monitor occurs only once. The number of permutations N_p in $\mathcal{Z}_n$ is given by $N_p = n!$* $\square$

$z_1 = 1 - 2 - 3$ and $z_2 = 3 - 2 - 1$ are both valid permutations in $\mathcal{Z}_3$. Dynamic programming is an effective algorithm for evaluating all possible permutations.

Figure 3.3: Example of fixed process monitor structure $z = 2-1-3$ and possible actions. The dynamic programming starts from the end (iteration 1) and calculates the optimal solution backwards, where $a_1 = \text{Terminate}$ and $a_2 = \text{Continue}$.

The first step in SDP analysis is to define a state space. Measurements of the process monitor confidence levels are discretised into a discrete state space. In this chapter we define the SDP state space as follows.

Definition 2 (SDP State Space) Let $\mathcal{S} = \{s_1, \dots, s_{N_S}, s_\infty\}$ be a finite discrete state space with N_S elements $s_i \in [0, 1]$ representing the confidence level output from a process monitor. The state s_∞ represents the terminating state. $\square$

The dynamic programming model is formulated as an optimal stopping problem and a terminating state s_∞ is defined, see Ross (1983). For stopping problems, no future reward can be earned once the program enters the terminating state. One advantage of the stopping problem formulation is the fact that the optimal solution can be found with a finite number of iterations.

The next step in defining the model, is to define the action space.

Definition 3 (Action Space) The set of possible actions at any given iteration in the dynamic programming algorithm is defined as

$$\mathcal{A} = \left\{ \begin{array}{ll} a_1 & \text{Terminate} \\ a_2 & \text{Continue} \end{array} \right\} \quad (3.1)$$

$\square$

When the dynamic programming is terminated (action a_1) a discrete event is identified by the sensory perception controller and the information is sent to the discrete event controller, see Figure 2.2.

In dynamic programming a reward function is associated with every discrete state and action. The reward function is defined as follows.

Definition 4 (Reward Function) *The reward is a function of discrete state and action, ie.*

$$R_n(z, s_i, a_k) = r(s_i, a_k) + L_n(z, a_k) \quad s_i \neq s_\infty, a_k \in \mathcal{A}, z \in \mathcal{Z}_{N_m} \quad (3.2)$$

$$R_n(z, s_\infty, a_k) = 0 \quad a_k \in \mathcal{A}, z \in \mathcal{Z}_{N_m} \quad (3.3)$$

where $r(s_i, a_k) \geq 0$ is the state reward given action a_k . N_m is the number of available process monitors. $L_n(z, a_k) \geq 0$ is the monitor reward of action a_k at iteration n in permutation z , reflecting factors such as time spent and computational burden invoked by the process monitor. $\square$

A large confidence level of the final decision is desired. We choose the termination state rewards as $r(s_i, a_1) = Kr(s_i)$. The rewards are increasing with increasing states $s_i \in [0, 1]$, where the state reward constant $K > 0$ is a scaling factor. The continuation state rewards are zero, ie. $r(s_i, a_2) = 0$. The continuation monitor reward is defined as follows.

$$L_n(z, a_2) = l_{m_n(z)} \quad (3.4)$$

where $l_{m_n(z)} > 0$ is the reward of using the process monitor at the n -th position in structure z . For example, in Figure 3.3 we have $m_1(z) = 3$, $m_2(z) = 1$ and $m_3(z) = 2$. The monitor reward for the termination action is zero, ie. $L_n(z, a_1) = 0$. The total reward (state and monitor reward) is zero for the terminating state s_∞ , which is typical for optimal stopping problems.

The last step in the stochastic model is to define discrete state transition probabilities. These probabilities will depend on the action taken. We define the transition probabilities as follows.

Definition 5 (Transition Probabilities) *The probability of being in SDP state s_i at iteration n and SDP state s_j at iteration $n + 1$ is defined as follows.*

$$P_{n,ij}(z,a) = \begin{cases} 0 & \text{if } a = a_1 \text{ and } s_i, s_j \neq s_\infty \\ 1 & \text{if } a = a_1, s_i \in \mathcal{S} \text{ and } s_j = s_\infty \\ P_{m_n(z)}(s_j | s_i) & \text{if } a = a_2 \text{ and } s_i, s_j \neq s_\infty \\ 0 & \text{if } s_i = s_\infty \text{ and } s_j \neq s_\infty \\ 1 & \text{if } s_i = s_j = s_\infty \end{cases} \quad (3.5)$$

□

Figure 3.4: Sensory perception problem formulated as a Markov decision process.

The underlying Markov model is shown in Figure 3.4. Initially, the decision process starts in state $\sigma(0) = s_1$ indicating a low confidence, ie. no process information is currently available. The termination reward is typically low in s_1 and the optimal action is likely to consult the first monitor. The resulting confidence level is recorded and the decision process enters a new state s_i . Then, if the termination reward in this new state is larger than the continuation reward, the decision process will terminate. Otherwise, the next monitor is consulted, a new state s_i is recorded and the decision problem repeats until the optimal action is to terminate or all monitors are consulted.

Note that action a_1 forces the system to enter the stopping state s_∞ . When the system has entered the stopping state, it can never leave this state. Since, the reward in the stopping state is zero, equation (3.3), the exact infinite horizon SDP solution can be found in a finite number of iterations. The maximum number of iterations required is equal to the number of process monitors, which ensures the computational efficiency of

the SDP algorithm.

We define a discretisation function as follows.

$$\mu(C) = \operatorname{argmin}_{s_i} |s_i - C| \quad (3.6)$$

μ is a mapping from continuous monitor outputs C to the state space $\mathcal{S}$. Let $\mathcal{C} = \{C : s_j = \mu(C)\}$. $\mathcal{C}$ is simply the set of all continuous confidence levels which are mapped to SDP state s_j . We then have

$$P_{m_n(z)}(s_j | s_i) = \int_{x \in \mathcal{C}} p_{m_n(z)}(x) dx \quad (3.7)$$

where $p_{m_n(z)}(x)$ is the probability density function of the confidence levels from monitor $m_n(z)$.

The state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $R_n(z, s_i, a_k)$ and the state transition probabilities $P_{n,ij}(z, a)$ completely describe the stochastic dynamic programming model.

3.2.2 Stochastic Dynamic Programming Algorithm

The objective of the SDP algorithm is to maximise the expected rewards, ie. with N iterations we want to find

$$V_N(z, s_i) = \max E_\pi \left[\sum_{n=1}^N R_n(z, \sigma(n), a_n) \alpha^{n-1} \mid \sigma(0) = s_i \right] \quad (3.8)$$

where E_π is the expectation operator and π is a policy for choosing the actions $a_n \in \{a_1, a_2\}$ at iteration n . The SDP state at iteration n is denoted $\sigma(n)$ and $0 \leq \alpha \leq 1$ is a discount factor, motivated by the fact that rewards to be earned in the future are worth less than immediate rewards. Equation (3.8) is a performance measure taking reliability and complexity factors into account. These factors are commonly used as a performance measure for intelligent machines, see for example Lima and Saridis (1994).

Since there are no rewards to be earned after the program enters the terminating state s_∞ , the SDP algorithm can be written as follows. For any $z \in \mathcal{Z}_{N_m}$, let $V_0(z, s_i) = r(s_i, a_1)$ and for any iteration $n > 0$

$$V_n(z, s_i) = \max \left[r(s_i, a_1), R_n(z, s_i, a_2) + \alpha \sum_{j=1}^{N_S} P_{n,ij}(z, a_2) V_{n-1}(z, s_j) \right] \quad (3.9)$$

$$a_n(z, s_i) = \begin{cases} a_1 & \text{if } V_n(z, s_i) = r(s_i, a_1) \\ a_2 & \text{otherwise} \end{cases} \quad (3.10)$$

$V_n(z, s_i)$ is the optimal value function and $a_n(z, s_i)$ is the optimal action at iteration n given permutation z and discrete state s_i . Since there are no rewards to be gained after the termination action, the reward for action a_1 is simply $r(s_i, a_1)$. The continuation reward depends on the reward $R_n(s_i, a_2)$ and rewards to be gained in the future, which are discounted by the factor $0 \leq \alpha \leq 1$. $V_n(z, s_i)$ is the maximal expected return of an n -stage problem that confers a terminal reward $V_0(z, s_j)$ if the process ends in state s_j , Ross (1983). For example, for the process monitor sequence $z = 1 - 2 - 3$, we have the following interpretation. $V_1(z, s_i)$ is the maximum expected reward when using monitor 3 and then stopping. $V_2(z, s_i)$ is the maximum expected reward with monitor sequence 2 - 3 and $V_3(z, s_i)$ is the maximum expected reward with monitor sequence 1 - 2 - 3.

Figure 3.5: SDP algorithm for finding optimal value function and optimal actions.

A flow diagram of the SDP algorithm is shown in Figure 3.5. This algorithm is performed only once and off-line. The optimal value functions $V_n(z, s_i)$ and optimal actions $a_n(z, s_i)$ are stored in a lookup table for each state $s_i \in \mathcal{S}$ and permutation $z \in \mathcal{Z}_{N_m}$.

Figure 3.6: Real-time SPC algorithm for selecting process monitors.

The SPC algorithm is shown in Figure 3.6. This algorithm uses the lookup table generated by the SDP algorithm and is performed on-line in the control system for every event occurrence. Given the current SDP state $\sigma(n)$ before SPC decision number

$n+1$, the SPC algorithm finds the optimal ordering z^* of enabled monitors, consults the monitor at position $N_m - n$ in z^* , receives the measurements $C(n)$ and $e(n)$ from this monitor and updates the new SDP state $\sigma(n+1)$ and the recognised event $e^*(n+1)$. The decision process is repeated until the optimal action is to terminate or all monitors have been consulted. The recognised event e^* is finally sent to the DEC.

3.2.3 Limitation on Total Cost

A limitation on the total cost is easily incorporated into the SDP algorithm. The SPC algorithm in Figure 3.6 remains unchanged, while the changes to the SDP algorithm in Figure 3.5 are described as follows.

1. Specify the total acceptable cost M .
2. For each permutation z generate a total accumulated cost sequence $\kappa(z, n) = \sum_{i=1}^{N_m+1-n} k_i$, where N_m is the number of monitors. For example, if the permutation is $z = 1 - 2 - 3$ and the monitor costs are $k_1 = 1$, $k_2 = 2$ and $k_3 = 3$, then the cost sequence is the accumulated sum $\kappa(z, 1) = 6$, $\kappa(z, 2) = 3$ and $\kappa(z, 3) = 1$, caused by the backward-induction form of the SDP algorithm.
3. In the SDP algorithm the termination action is chosen if the total accumulated cost is larger than M , ie.

$$\left. \begin{aligned} a_n(z, s_i) &= a_1 \\ V_n(z, s_i) &= r(s_i, a_1) \end{aligned} \right\} \quad \text{if } \kappa(z, n) > M \quad (3.11)$$

By incorporating equation (3.11) in the off-line SDP algorithm, the cost-efficiency of the SPC in real-time operation is guaranteed.

3.2.4 Data Fusion

The SPC presented in Figure 3.6 uses two functions, $f(\cdot)$ and $g(\cdot)$, to calculate the new SDP state $\sigma(n)$ and the recognised discrete event e^* . The updated variables depend on the outputs C^i and e^i obtained from the monitors during execution. Equation (3.7) uses only data from the last monitor consulted. The discrete event recognised by the final

monitor is mapped to γ^* which is sent to the DEC. Using only the final monitor means that existing information from previous monitors is ignored. To make better use of the process monitoring information, we discuss in this section the required modifications to the SDP model to allow for different types of data fusion. We consider two additional data fusion functions; 1) selecting the discrete event from the monitor with the largest confidence and 2) a fusion method based on consensus among monitors. Dynamic programming is based on the principle of optimality, Bellmann (1957). In practice the principle requires the rewards and the transition probabilities to be functions only of the last SDP state and the subsequent action. Hence, when incorporating different data fusion algorithms in the SDP algorithm, the new model can not violate these requirements.

First, we consider the fusion algorithm where the output from the monitor with the largest confidence is chosen. The action set $\mathcal{A}$ and the reward function $R_n(z, s_i, a_k)$ remain unchanged, while the state transition probabilities require modifications. The state set $\mathcal{S}$ remains unchanged, but the interpretation of the SDP state $\sigma(n)$ is new. Let $C(n) = C^{m_{N_m-n}}(z^*)$ be the continuous confidence output from the monitor consulted at decision n in the optimal sequence z^* . Let $\sigma(n-1)$ be the *maximum confidence* of all monitors used at decision 1 to $n-1$. Then, the new SDP state after decision n is given by

$$\sigma(n+1) = \max [\sigma(n), \mu(C(n))] \quad (3.12)$$

where $\sigma(n+1)$ is the maximum confidence of the monitor consulted by decision n and all previous monitors. Note that $\sigma(n+1)$ is a function of the measurement $C(n)$ and the previous state $\sigma(n)$ only. Hence, the principle of optimality still holds. The modified state transition probability $P'_{n,ij}(z, a)$ is the same as equation (3.5) except that

$$P'_{m_n(z)}(s_j | s_i) = \begin{cases} 0 & \text{if } s_j < s_i \\ \sum_{k=1}^i P_{m_n(z)}(s_k | s_i) & \text{if } s_j = s_i \\ P_{m_n(z)}(s_j | s_i) & \text{if } s_j > s_i \end{cases} \quad (3.13)$$

where $P'_{m_n(z)}(s_j | s_i)$ is the modified transition probability when the action taken is a_2 and $s_i, s_j \neq s_\infty$.

Second, we consider a data fusion algorithm based on event consensus among the monitors. In contrast to the other two fusion methods presented, the consensus based

method considers both the confidence levels C and the recognised events e when calculating the new Markov decision state. Let $e(n-1)$ and $e(n)$ be the recognised discrete events from the monitors consulted at decision $n-1$ and n , respectively. A consensus among these monitors is given by $e(n-1) = e(n)$. The SDP state $\sigma(n+1)$ after decision n when using consensus is then given by

$$\sigma(n+1) = \begin{cases} \mu(C(n)) & \text{if } e(n) \neq e(n-1) \\ \mu(1 - (1 - \sigma(n))(1 - C(n))) & \text{if } e(n) = e(n-1) \end{cases} \quad (3.14)$$

If the monitors consulted at decision n and $n-1$ disagree, the data from the last monitor is used. Because of the principle of optimality, the SDP model only allows for consensus among monitors used consecutively. Hence, when disagreements occur, data from the last monitor is used to form a consensus with future decisions. If there is an agreement, the new confidence is equal to 1 minus the probability of both monitors choosing the wrong event. Note that $\sigma(n+1)$ is only a function of the measurements $C(n)$, $e(n)$, $e(n-1)$ and the previous Markov state $\sigma(n)$. Hence, the requirements for using dynamic programming are still satisfied. Let $p = Pr(e(n) = e(n-1))$ be the probability of a consensus among the monitors consulted at decision n and $n-1$. For a consensus we have

$$\sigma(n+1) = \mu(1 - (1 - \sigma(n))(1 - C(n))) \quad (3.15)$$

Let $\bar{x}$ and $\underline{x}$ be the upper and lower boundaries of $C(n)$ which result in $\sigma(n) = s_i$ to $\sigma(n+1) = s_j$, ie.

$$\begin{aligned} \bar{x} &= \operatorname{argmax}_{C(n)} [s_j = \mu(1 - (1 - s_i)(1 - C(n)))] \\ \underline{x} &= \operatorname{argmin}_{C(n)} [s_j = \mu(1 - (1 - s_i)(1 - C(n)))] \end{aligned}$$

The modified state transition probability $P''_{n,ij}(z, a)$ is then the same as equation (3.5) except that

$$P''_{m_n(z)}(s_j | s_i) = (1 - p)P_{m_n(z)}(s_j | s_i) + p \int_{\underline{x}}^{\bar{x}} p_{m_n(z)}(x) dx \quad (3.16)$$

Note that fusion based on consensus requires the additional parameter p to be estimated, where p is the probability of consensus among monitor $m_n(z)$ and $m_{n+1}(z)$.

To summarise the three different data fusion methods, we have

$$\begin{bmatrix} \sigma(n+1) \\ e^*(n+1) \end{bmatrix} = \begin{bmatrix} f(\sigma(n), \mu(C(n))) = \mu(C(n)) \\ g(e(n), e^*(n)) = e(n) \end{bmatrix} \quad (3.17)$$

$$\begin{bmatrix} \sigma(n+1) \\ e^*(n+1) \end{bmatrix} = \begin{bmatrix} \max [\sigma(n), \mu(C(n))] \\ \begin{cases} e(n) & \text{if } \sigma(n+1) = \mu(C(n)) \\ e^*(n) & \text{otherwise} \end{cases} \end{bmatrix} \quad (3.18)$$

$$\begin{bmatrix} \sigma(n+1) \\ e^*(n+1) \end{bmatrix} = \begin{bmatrix} \begin{cases} \mu(1 - (1 - \sigma(n))(1 - C(n))) & \text{if } e(n) = e(n-1) \\ \mu(C(n)) & \text{otherwise} \end{cases} \\ e(n) \end{bmatrix} \quad (3.19)$$

Equation (3.17) uses data from the last monitor consulted only, equation (3.18) selects the event with the maximum confidence and equation (3.19) is based on consensus among monitors. Only one of the equations (3.17)–(3.19) is used by the real-time SPC algorithm in Figure 3.6 to find the new SDP state $\sigma(n+1)$ and the recognised discrete event $e^*(n+1)$ whenever a monitor has been consulted.

We have shown how different data fusion algorithms may be incorporated without major changes to the SDP model. Moreover, the principle of optimality, which is one of the assumptions required by dynamic programming, still holds with the more advanced data fusion algorithms presented in this section. As will be seen in the sensory perception control example in section 3.3, the data fusion algorithms presented improve both the reliability and the cost of the sensory perception.

3.2.5 Properties of the Sensory Perception Controller

The sensitivity and stability analyses of SDP solutions are often difficult due to the iterative nature of these solutions. In this section we present some useful properties of the dynamic programming solution for our particular choice of model structure. The effects of a varying state reward K and monitor rewards L_n on the optimal process monitor structure z^* are studied. To simplify the notation, we drop the parameter z .

First, we show that if the state reward K and the monitor rewards L_n are multiplied by the same factor $0 < \epsilon < \infty$, the solution remains unchanged.

Proposition 1 *Let $\bar{V}_n(s_i)$ be the optimal value function when K and L_n in the original problem are replaced by $\bar{K} = \epsilon K$ and $\bar{L}_n = \epsilon L_n$, where $0 < \epsilon < \infty$. The optimal monitor structure in the new problem $\bar{z}^*$ is then equal to z^* in the original problem and the optimal actions remain unchanged.*

Proof: The result is proven by induction. We have $\bar{V}_0(s_i) = \epsilon K r(s_i) = \epsilon V_0(s_i)$. We assume that for $n - 1$ we have $\bar{V}_{n-1}(s_i) = \epsilon V_{n-1}(s_i)$. Then, for n , we have

$$\begin{aligned}\bar{V}_n(s_i) &= \max \left[\epsilon r(s_i, a_1), \epsilon R_n(s_i, a_2) + \alpha \sum_{j=1}^{N_S} P_{n,ij}(a_2) \bar{V}_{n-1}(s_j) \right] \\ &= \epsilon \max \left[r(s_i, a_1), R_n(s_i, a_2) + \alpha \sum_{j=1}^{N_S} P_{n,ij}(a_2) V_{n-1}(s_j) \right] \\ &= \epsilon V_n(s_i)\end{aligned}\tag{3.20}$$

The value function is simply multiplied by ϵ . Hence, the optimal actions remain unchanged and the structure with the largest value function in the original problem remains the optimal structure in the modified problem. ■

Second, we show that a multiplication by $0 < \epsilon < \infty$ on the state reward constant K has the same effect on the solution as a division by ϵ on the monitor rewards.

Proposition 2 *Let $\bar{V}_n(s_i)$ and $\tilde{V}_n(s_i)$ be the optimal value functions when K and L_n in the original problem are replaced by $\bar{K} = \epsilon K$ and $\tilde{L}_n = \frac{L_n}{\epsilon}$, respectively. Then, the resulting optimal monitor structures $\bar{z}^*$ and $\tilde{z}^*$ are identical.*

Proof: We have

$$\tilde{V}_n(s_i) = \max \left[r(s_i, a_1), \frac{R_n(s_i, a_2)}{\epsilon} + \alpha \sum_{j=1}^{N_S} P_{n,ij}(a_2) \tilde{V}_{n-1}(s_j) \right]\tag{3.21}$$

$$\begin{aligned}\bar{V}_n(s_i) &= \max \left[\epsilon r(s_i, a_1), R_n(s_i, a_2) + \alpha \sum_{j=1}^{N_S} P_{n,ij}(a_2) \bar{V}_{n-1}(s_j) \right] \\ &= \max \left[\epsilon r(s_i, a_1), \epsilon \frac{R_n(s_i, a_2)}{\epsilon} + \alpha \sum_{j=1}^{N_S} P_{n,ij}(a_2) \bar{V}_{n-1}(s_j) \right]\end{aligned}\tag{3.22}$$

By Proposition 1 the value functions in equations (3.21) and (3.22) lead to the same optimal monitor structure, ie. $\bar{z}^* = \tilde{z}^*$ and the optimal actions remain unchanged. ■

Proposition 2 shows that only the relative difference between the monitor rewards and the state rewards matter. Dividing the monitor rewards and multiplying the state rewards by the same factor ϵ leaves the solution unchanged. Propositions 1 and 2 are intuitive and desirable results for the SDP solution. For example, when the process monitors are run on a slower or faster processor board, the monitor rewards (measure of CPU time) will change. By changing the state reward constant K accordingly, we can guarantee that the SDP solution remains unchanged.

3.3 Sensory Perception Control Example

To demonstrate the proposed method we study a sensory perception controller with $N_m = 6$ available process monitoring techniques. The monitors are numbered from 1 to 6 and have characteristics as shown in Figure 3.7.

Figure 3.7: Six different classes of monitoring techniques.

The probability density functions for the low, medium and high confidence monitors are shown in Figure 3.8. Two different cases of monitor costs are studied. The costs are shown in Table 3.1, where the monitor rewards are calculated by $l_i = k_{\max} - k_i$.

	k_1	k_2	k_3	k_4	k_5	k_6
Case 1	4	4	4	1	1	1
Case 2	4	4	4	2	2	2

Table 3.1: Monitor costs for two different cases.

The discretised state space $\mathcal{S}$ for the confidence levels and the state rewards $\mathcal{R}$ were

Figure 3.8: *Probability density functions for the monitor confidence levels.*

chosen as follows.

$$\mathcal{S} = \{0.0, 0.5, 0.75, 0.875, 0.9375, 0.96875, 0.984375, 0.9921875, s_{\infty}\} \quad (3.23)$$

$$\mathcal{R} = K * \{0, 0, 1, 2, 3, 4, 5, 6, 0\} \quad (3.24)$$

The first state in the SDP state set $\mathcal{S}$ is $s_1 = 0.0$. The next state, s_2 , is half the distance from the previous state s_1 to 1, ie. $s_2 = 0.5$. In general, the state s_j is half the distance from s_{j-1} to 1, $j \in \{2, \dots, N_S\}$. The motivation for the selection of $\mathcal{S}$ and $\mathcal{R}$ above is that increasing a low confidence is relatively easy, while increasing a large confidence is difficult. By the choice above, the same increase in reward is given for a change from s_2 to s_3 as for a change from s_7 to s_8 , even if s_2 to s_3 increases the confidence by 0.25 while for s_7 to s_8 the increase in confidence is only 0.0078125. The scaling factor is chosen to be $K = k_{\max}$. By choosing $K = k_{\max}$, the reward for choosing the fastest monitor is equal to the smallest non-zero state reward. Hence, the state rewards dominate the monitor rewards, which ensures that the dynamic programming will terminate when a relatively large confidence level is reached.

With six process monitors available, the generation of the lookup table by the off-line SDP algorithm requires approximately 2 CPU seconds on a SPARC Ultra. The real-time SPC algorithm in Figure 3.6 requires approximately 1ms of CPU time on a Motorola 68040 based vxWorks board. Hence, the SPC is well suited to fast real-time operation.

A sample set of 1000 discrete events was generated with confidence levels obtained from the density functions in Figure 3.8. The average recognition rates of the low, medium and high confidence monitors in the sample set were 48.6%, 69.2% and 87.4%, respectively. Figure 3.9 shows the average usage of each of the monitors for the two cases of monitoring costs, where $\alpha = 0.7$ and the consensus fusion method was used. The scaling factor was chosen $K = k_{\max} = 4$. Note that the major change from case 1 to 2 is a large reduction of the usage of monitor 5. Because of the extra cost and the low confidence levels, monitor 5 is no longer attractive to the sensory perception controller. Monitor 6 is the most attractive one, even for case 2 where the cost is twice as high as for case 1. Despite its large cost, monitor 3 is used 49.5% and 69.9% of the time for the two cases. The reasonably large usage is caused by the large confidence levels from monitor 3. Figure 3.9 shows how the sensory perception is a compromise between cost and confidence.

Figure 3.9: Average usage of the process monitors for two different sets of monitor costs. The consensus fusion method is used with $\alpha = 0.7$.

Figure 3.10 shows the average costs and recognition rates of the different fusion methods as a function of α on the sample set of 1000 events. The recognition rate is defined as the ratio of the number of correct events e^* versus the total number of events in the sample set. The monitor costs for case 1 are used. Note that the average costs have large values for large α , which can be seen from equation (3.9). The value of the continuation action a_2 is increasing with an increasing α . Hence, action a_2 occurs more

Figure 3.10: Average costs and average recognition rates of the different fusion methods as a function of the discount factor α with monitor costs $k_1 = k_2 = k_3 = 4$ and $k_4 = k_5 = k_6 = 1$.

often. When a_2 occurs more often, the average total cost has a tendency to increase, since the monitors are consulted more frequently. However, an increase in α does not guarantee an increased monitor usage, which can be seen from Figure 3.10. When the cost increases, more data is collected which normally improves the recognition rate. An example where an increased monitor usage does not increase the recognition rate is given in Figure 3.10 for the data fusion method based on maximum confidence. When α changes from 0.43 to 0.44 the average cost increases from 1.00 to 1.51 while the average recognition rate decreases from 87.5% to 85.1%. For both choices of α monitor 6 is used all the time. When $\alpha = 0.44$, monitor 4 and 5 are used after monitor 6 25.3% and 25.4% of the time, respectively. The relatively low confidence levels from monitor 4 and 5 cause the decrease in recognition rate. Note that for the consensus based method there is no major decrease in recognition rate when α increases, which indicates a better use of the collected information. However, even for the consensus based fusion method there is no guarantee that the recognition rate will increase with increasing α , see for example Figure 3.10 when α changes from 0.11 to 0.12. For this particular example the recognition rate for monitor 6 alone ($\alpha = 0.11$) is higher than when consensus is being used among several monitors ($\alpha = 0.12$ and usage 2.5%, 3.0%, 25.4% and 100.0% for the monitors 3, 4, 5 and 6, respectively).

The best overall performance is achieved with the consensus based sensory perception. For large values of α , the consensus method has the highest recognition rates and reasonably low costs. A good result is achieved for $\alpha = 0.65$ with an average recognition rate of 95.7% and average cost of 3.31. As a comparison, the best individual monitor alone has an average recognition rate of 87.4% with a cost of 1. The cost of using all six monitors all the time is 15 with a recognition rate of 98.2% when using consensus based data fusion.

Figure 3.11: Set of process monitors where the cost increases with the reliability.

Figure 3.12: Average costs and average recognition rates of the different fusion methods as a function of the discount factor α with monitor costs $k_1 = k_4 = 1$, $k_2 = k_5 = 2$ and $k_3 = k_6 = 3$.

A more realistic choice of monitor costs and recognition rates is considered next. Normally the cost of a process monitor increases with the reliability. We study an example

with six monitors and characteristics as shown in Figure 3.11. The low, medium and high confidence classes are again given by Figure 3.8 and the monitor costs are given by $k_1 = k_4 = 1$, $k_2 = k_5 = 2$ and $k_3 = k_6 = 3$.

Figure 3.13: Average usage of the process monitors for $\alpha = 0.61$. The consensus fusion method is used and the monitor costs are $k_1 = k_4 = 1$, $k_2 = k_5 = 2$ and $k_3 = k_6 = 3$.

The average recognition rates and costs as functions of α are shown in Figure 3.12. Again, the best results are obtained from the consensus based fusion method. A good result is achieved for $\alpha = 0.61$ with a recognition rate of 95.2% and a total cost of 5.24. The monitor usage for this case is shown in Figure 3.13. The most reliable individual monitor (3 or 6) used by itself has a recognition rate of 87.4 with a total cost of 3. The cost of using all the monitors all the time is 12. As expected, the ratio of the SPC cost compared to the cost of using all the monitors is larger for the more realistic case than for the cases studied earlier in this section.

3.4 Evaluation

In this section we discuss the advantages of the proposed method compared to other approaches such as using a single monitoring technique or using all the event monitors simultaneously. We also discuss some of the limitations of the dynamic programming approach to the control of sensory perception.

3.4.1 Advantages

The main advantage of the sensory perception control structure is an improved event recognition rate compared to individual event monitors while the total cost is kept low. For cases where it is too expensive to use all the event monitors simultaneously, the maximum total cost can be limited and hence the proposed solution is well suited for use in real-time control systems with bandwidth requirements.

As mentioned in the previous section, the proposed method is well suited to fast real-time operation by combining the lookup table generated off-line and measured confidence levels in the decision making. For the case study with six process monitors, the generation of the lookup table required approximately 2 CPU seconds on a SPARC Ultra. For systems where the sensing costs are much larger than this number, one might be willing to accept a larger overhead cost and use the dynamic programming during execution. The extra benefits of on-line dynamic programming, is the ability to actively adapt SPC parameters such as the probability distributions shown in Figure 3.8. Dynamic programming is a cost-efficient solution compared to brute-force methods and hence is a good alternative in on-line operation.

Another advantage of the proposed method is the ease with which new process monitoring techniques can be added to the system. The discrete state space $\mathcal{S}$, the action space $\mathcal{A}$, the structure of the state transition probabilities and the reward function all remain unchanged when new monitors are added to the system. Hence, the proposed dynamic programming model can be transferred without major changes to sensory perception problems with different numbers of available monitoring techniques. The only changes when a new process monitor is added to the system are an additional iteration by the SDP algorithm and a larger search space for the sensory perception controller.

The proposed method requires no knowledge of the process plant. The only requirement is that each process monitor produces a recognised discrete event with an associated confidence level. When such monitoring techniques are available, the proposed method can be applied to a wide range of discrete event control problems. Two of the fusion methods, last monitor and maximum confidence fusion, require only the confidence levels for making decisions. The consensus based fusion method requires both the

recognised discrete events and the confidence levels from the process monitors.

3.4.2 Limitations

As can be seen from Figures 3.10 and 3.12 the recognition rates and the average costs are highly non-linear functions of the discount factor α . The discontinuous jumps in the recognition rates and costs are caused by the max function in equation (3.9). The nonlinearities and the iterative form of the SDP equation make direct analytical sensitivity and stability analyses of the SDP equations difficult.

The dynamic programming algorithm requires a relative large amount of memory. In the case study six process monitoring techniques were available. Hence, the number of permutations is 720. The size of the lookup table is $N_m N_p (N_S + 1) = 38880$ in the case study. Each element in the lookup table consists of a floating point number for the optimal value function V_n and a binary number for the optimal action a_n . For systems with a larger number of process monitors, one has to reduce the number of monitor sequences to consider. One solution is to introduce priority lists and sub-optimality to reduce the number of permutations. This problem has been addressed by power systems researchers. Pang and Chen (1976) presented a system involving sequencing of up to 96 units using priority lists.

As mentioned in the previous section the dynamic programming is performed off-line and no adaptation on model parameters is performed. However, if the monitor parameters, such as costs and probability densities, change values only when the DES state γ changes, the adaptation problem can be avoided. In this scenario a different lookup table can be generated off-line by the SDP algorithm for each DES state γ . When a discrete event occurs, the next state γ is unknown. However, the state $\gamma(t_k - 1)$ has previously been recognised by the SPC with high reliability. Hence, depending on the previous DES state $\gamma(t_k - 1)$, the SPC selects the correct lookup table and executes the optimal actions as before. The more accurate description of the model parameters can improve the SPC recognition rates and costs. The trade-off is a larger computational complexity in the off-line SDP algorithm.

3.5 Conclusion

A new technique for the control of sensory perception in a real-time discrete event control system has been developed. The proposed method is based on stochastic dynamic programming, which provides an efficient solution to the multi-stage sequential decision problem. The method is well suited for use in real-time feedback control systems with bandwidth requirements. A maximum total cost (CPU time) can be specified, and the SDP algorithm solves the constrained optimisation problem. The final optimal solution takes monitor rewards, confidence level rewards and the upper CPU time limit into account.

The SDP approach solves the sequential decision problem by using a foresight of future process monitor characteristics, such as confidence level probability distributions and monitor costs. Hence, the sensory perception controller generates quite complex decisions taking current and future rewards into account.

The method allows for incorporation of a wide range of sensing techniques. The only requirement of the process monitors is that they recognise discrete events as they occur with an associated confidence level.

Three different data fusion techniques have been incorporated into the SDP model; 1) using only the last monitor consulted, 2) using the monitor with maximum confidence and 3) the decisions are made based on consensus among monitors. The best results were achieved when using the consensus method. The performance on a test case was studied and the average recognition rates achieved by the sensory perception controller were found to be larger than any individual monitor while the average total processing costs were kept low. The control of sensory perception is an important step towards designing more reliable and robust control systems for operation in uncertain real-time environments.

Chapter 4

Linear Programming Solution to Sensitivity Analysis

4.1 Introduction

To gain insight and allow for robust tuning of any control methodology a sensitivity analysis is required. One limitation of the work presented in Chapter 3 is the ad-hoc tuning of the sensory perception controller (SPC) parameters, such as the discount factor α and the state reward constant K . In this chapter we overcome this limitation by presenting a thorough sensitivity analysis of the SPC. The SPC developed in Chapter 3 solves the sensor selection process by using stochastic dynamic programming (SDP). In this chapter we show that linear programming (LP) is an alternative solution to the sensory perception control problem. The LP solution is not as efficient as the SDP solution, but sensitivity analysis of linear programs is well-known and a large number of theoretical results is available in the literature.

Stochastic dynamic programming provides an efficient solution to Markov decision problems. However, one limitation of SDP is the fact that a direct sensitivity analysis is difficult. Fiacco (1983) presented a direct approach to sensitivity analysis of nonlinear programming. The theory applies to static deterministic problems. By contrast, few results are available in the literature for the direct sensitivity analysis of stochastic dynamic programming. Under these circumstances, linear programming provides a

powerful alternative for the sensitivity analysis of Markov decision processes. LP has a mature theory for sensitivity analysis, see for example Chvátal (1983), Rothenberg (1979), Solow (1984), Thie (1988).

The LP solution is not as efficient the stochastic dynamic programming solution in Chapter 3 for solving Markov decision problems. In Puterman (1994, p. 230) a comparison was made between linear programming and dynamic programming. It was found that for a Markov decision model with 2000 states and 10 actions in each, the LP solution was 12 times slower than the SDP solution. In a practical setting, the SDP solution is the preferred method for solving the Markov decision problems, while linear programming is used only for the sensitivity analysis.

The sensitivity analysis allows for the robust tuning of the SPC model parameters. With the results presented in this chapter, we are able to answer questions of the following type: "If a parameter α changes from α_1 to α_2 , will this cause the SPC to change the sequential order in which monitors are consulted?". The ability to study the effects of uncertain confidence measurements is another important feature of the sensitivity analysis. For example, an event monitor may produce high confidence levels while incorrectly classifying events. By comparing the optimal actions for different Markov states, we are able to analyse the effects of uncertain confidence measurements. The measured confidence levels determine the current Markov state $\sigma(n)$. Hence, uncertainties in the measurements lead to uncertainties in the Markov state. In contrast to open-loop sensor planning strategies, we are able to analyse the behaviour of the real-time sensor selection strategies.

In section 4.2 the sensory perception control problem is solved using linear programming. Section 4.3 presents the sensitivity analysis using linear programming theory and section 4.4 demonstrates the application of the sensitivity analysis.

4.2 Linear Programming Solution

The LP solution to Markov decision processes is presented in section 4.2.1 and the LP is presented in standard form in section 4.2.2. The standard form allows for the use of standard theory for the sensitivity analysis of linear programs.

4.2.1 Linear Programming Equations

LP solutions to general Markov decision processes have been presented by Kallenberg (1983), Mine and Osaki (1970), Puterman (1994). The LP objective function is given in equation (4.1) under the constraints (4.2) and $\lambda_n(s_i, a_k) \geq 0$ for all $n = 0, \dots, N_m$, for all states s_i and all actions a_k . N_m is the number of available process monitors.

$$\max \sum_{n=0}^{N_m} \alpha^n \sum_{s_i \in S} \sum_{a_k \in A} R_n(s_i, a_k) \lambda_n(s_i, a_k) \quad (4.1)$$

$$\sum_{a_k \in A} \lambda_n(s_j, a_k) = \begin{cases} h_j & \text{for } n = 0 \\ \sum_{s_i \in S} \sum_{a_k \in A} P_{ij}(a_k, n-1) \lambda_{n-1}(s_i, a_k) & \text{for } n = 1, \dots, N_m \end{cases} \quad (4.2)$$

Let $\sigma(0)$ be the initial Markov state. Then $h_j = 1$ if $\sigma(0) = s_j$ and $h_j = 0$ otherwise. Solving (4.1) under the constraints (4.2) yields an *identical* solution to the dynamic programming equation (3.9), see Mine and Osaki (1970) for the proof. Solving a linear program gives a set of basic and non-basic variables. One characteristic of the LP formulation above, is the fact that for each state s_i , one and only one of the variables $\lambda_n(s_i, a_1)$ or $\lambda_n(s_i, a_2)$ is a basic variable. Hence, the action a_k for which $\lambda_n(s_i, a_k)$ is a basic variable uniquely determines the optimal action for each state and iteration number n . These optimal actions are identical to the SDP actions in equation (3.10). The standard Simplex algorithm is used to compute the basic variables and hence the optimal actions.

The variable n is the iteration number in the SDP formulation. The LP approach is not an iterative solution. All the actions for the entire decision horizon are solved simultaneously. However, in both the LP and the SDP formulations the variable n is used to distinguish actions at different discrete times in the Markovian decision process. Note that for the LP problem state variables $\lambda_n(s_i, a_k)$ are needed for each value of n . Hence, with a large number of available monitors, the number of variables and constraints in the LP algorithm becomes large. With two actions (a_1 and a_2) and N_m available monitors, the LP state space is $2N_m$ times larger than the SDP state space. The compact state space formulation is one of the reasons for the computational efficiency of the SDP algorithm.

All possible sequential permutations of process monitors are considered. Hence, as for the SDP algorithm in Chapter 3, the LP algorithm must be solved for each permutation. Given a permutation, the variables $\lambda_0(s_i, a_k)$ determine the optimal actions for the first monitor, while $\lambda_1(s_i, a_k), \dots, \lambda_{N_m-1}(s_i, a_k)$ determine the optimal actions for monitors $1, \dots, N_m$ in the chosen permutation.

4.2.2 Linear Program in Standard Form

To be able to use the sensitivity analysis results available in the literature, the LP in equations (4.1) and (4.2) must first be written on the following standard form

$$\begin{aligned} &\text{Maximise} && \mathbf{c}\mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\ &&& \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{4.3}$$

By selecting the variables $\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ properly, the LP in (4.3) is identical to the solution in equations (4.1) and (4.2).

First, let $s_{N_s+1} = s_\infty$. We then choose the $\mathbf{x}$ vector as follows.

$$\begin{aligned} \mathbf{x} = & [\lambda_0(s_1, a_1), \dots, \lambda_0(s_{N_s+1}, a_1), \lambda_0(s_1, a_2), \dots, \lambda_0(s_{N_s+1}, a_2), \\ & \lambda_1(s_1, a_1), \dots, \lambda_1(s_{N_s+1}, a_1), \lambda_1(s_1, a_2), \dots, \lambda_1(s_{N_s+1}, a_2), \\ & \vdots \\ & \lambda_{N_m}(s_1, a_1), \dots, \lambda_{N_m}(s_{N_s+1}, a_1), \lambda_{N_m}(s_1, a_2), \dots, \lambda_{N_m}(s_{N_s+1}, a_2)]^T \end{aligned} \tag{4.4}$$

The constraints $\lambda_n(s_i, a_k) \geq 0$ ensure that $\mathbf{x} \geq \mathbf{0}$.

Let $\sigma(0)$ be the initial state in the Markov decision process. Let $h_j = 1$ if $\sigma(0) = s_j$ and $h_j = 0$ if $\sigma(0) \neq s_j$. Then, we choose the $\mathbf{b}$ vector as follows.

$$\mathbf{b} = [h_0, \dots, h_{N_s+1}, 0, \dots, 0]^T \tag{4.5}$$

The choices for $\mathbf{x}$ and $\mathbf{b}$ above combined with the constraints in equation (4.2) uniquely define the $\mathbf{A}$ matrix such that $\mathbf{A}\mathbf{x} = \mathbf{b}$.

The selection of $\mathbf{x}$ and the LP objective criterion (4.1) uniquely define the $\mathbf{c}$ vector, ie.

$$\begin{aligned}
\mathbf{c} = & [R_0(s_1, a_1), \dots, R_0(s_{N_s+1}, a_1), R_0(s_1, a_2), \dots, R_0(s_{N_s+1}, a_2), \\
& \alpha R_1(s_1, a_1), \dots, \alpha R_1(s_{N_s+1}, a_1), \alpha R_1(s_1, a_2), \dots, \alpha R_1(s_{N_s+1}, a_2), \\
& \vdots \\
& \alpha^n R_{N_m}(s_1, a_1), \dots, \alpha^n R_{N_m}(s_{N_s+1}, a_1), \alpha^n R_{N_m}(s_1, a_2), \dots, \alpha^n R_{N_m}(s_{N_s+1}, a_2)]
\end{aligned}
\tag{4.6}$$

The $\mathbf{c}$ vector contains all the discounted rewards, and hence $\mathbf{c}$ will be referred to as the LP reward vector.

When the LP is in standard form, the standard Simplex algorithm, see for example Press, et.al. (1992), is used to find the optimal basic variables.

4.3 Sensitivity Analysis

Once the linear program is in standard form (4.3), we can apply well known sensitivity analysis techniques for LP. The LP $\mathbf{c}$ coefficients allow for sensitivity analysis of the rewards. As mentioned earlier in this chapter, the rewards contain the two main objectives; 1) Large confidence levels and 2) Fast monitoring. Hence, the reward vector $\mathbf{c}$ is the most interesting variable for the sensitivity analysis. The reward vector contains several interesting model parameters, such as the discount factor, monitor rewards and state rewards.

Let $\mathbf{x}^* = (\mathbf{x}_B^*, \mathbf{x}_N^*)$ be the optimal basic feasible solution (bfs), where $\mathbf{x}_B^*$ are the optimal basic variables and $\mathbf{x}_N^*$ are the optimal non-basic variables. Let N_B and N_N be the number of basic and non-basic variables, respectively. Let $\mathbf{B}$ be an $N_B \times N_B$ sub-matrix of $\mathbf{A}$ corresponding to the basic variables and let $\mathbf{N}$ be an $N_B \times N_N$ sub-matrix of $\mathbf{A}$ corresponding to the non-basic variables. Let $\mathbf{c}_B$ and $\mathbf{c}_N$ be the reward coefficients of the basic and non-basic variables, respectively. Then the LP optimality criterion can be formulated as follows.

$$\mathbf{c}_N - \mathbf{c}_B \mathbf{B}^{-1} \mathbf{N} \leq \mathbf{0} \tag{4.7}$$

See for example Solow (1984) for the proof. The matrix $\mathbf{B}^{-1}$ can be found from the output matrix of the Simplex algorithm by selecting the columns corresponding to the

basic variables. Equation (4.7) can be used to determine the range of values over which the elements of $\mathbf{c}$ can be varied without changing the optimal bfs, $\mathbf{x}^*$. Let Φ be the sensitivity parameter, for example the discount factor α or individual rewards $R(s_i, a_k)$ in equation (4.6). The method in Table 4.1 calculates the Φ values for each optimal bfs $\mathbf{x}^*$.

1. Let $\Phi = \Phi_{\max}$

2. Use equation (4.6) to find $\mathbf{c}(\Phi)$ and solve LP, ie. find $\mathbf{x}_B^*(\Phi)$.

3. Find $0 \leq \epsilon \leq 1$ such that

$$\bar{\mathbf{c}}_N - \bar{\mathbf{c}}_B \mathbf{B}^{-1} \mathbf{N} \leq 0$$

where $\bar{\mathbf{c}} = (\bar{\mathbf{c}}_B, \bar{\mathbf{c}}_N) = \mathbf{c} \cdot \mathbf{f}(\epsilon, \Phi)$

4. $\Phi = \epsilon \Phi$. Save Φ value and corresponding $\mathbf{x}_B^*(\Phi)$, $\mathbf{c}_B(\Phi)$.

While $\Phi > \Phi_{\min}$ go to 2.

Table 4.1: Method for finding the intervals of Φ where the bfs $\mathbf{x}^*$ stays constant.

The function $\mathbf{f}$ is vector valued and of same dimension as $\mathbf{c}$, where $\cdot$ denotes array multiplication. The exact form of the vector valued function $\mathbf{f}$ depends on which parameter, for example α or $R(s_i, a_k)$ is being analysed. The equation $\bar{\mathbf{c}} = \mathbf{c} \cdot \mathbf{f}(\epsilon, \Phi)$ simply modifies the reward vector $\mathbf{c}$ when the parameter Φ is modified to $\epsilon \Phi$. For the discount factor α , for example, we have $\mathbf{f}_i(\epsilon, \alpha) = \epsilon^n$ where n is the iteration number of the i th element of equation (4.6).

The LP reward vector $\mathbf{c}$ contains all the model parameters α , K and $r(s_i, a_k)$. Hence, the algorithm in Table 4.1 can be used for the sensitivity analysis of all these parameters. The only difference from one parameter to another is the form of the vector valued function $\mathbf{f}$.

The method above will calculate a discrete set $\mathcal{E}$ of values for Φ . Between any two neighbour values in $\mathcal{E}$ the optimal bfs $\mathbf{x}^*$ is constant. Hence, the optimal action, a_1 or a_2 , is also constant between any two neighbour values of Φ in $\mathcal{E}$. When changing a model parameter, it is of course possible to formulate the new problem and to solve it from scratch. However, this approach is both wasteful and unnecessary. By using the

method in Table 4.1, the LP normally has to be solved only for a small set of Φ values. One example in the experiments section shows that a complete sensitivity analysis of a continuous parameter using LP is 4.3 times faster than solving the problem from scratch for each new parameter value using a brute-force SDP approach.

Until now we have described how to analyse the sensitivities of some parameters for a fixed permutation of monitors. As mentioned earlier in this chapter, what we are really interested in, is the sensitivity of these parameters on the decisions made by the sensory perception controller. For example, if the initial SDP state is $\sigma(0) = 0.5$ and the discount factor changes from 0.9 to 0.7, will the SPC consult monitor 2 rather than monitor 1 first? To answer these kinds of questions, the method in Table 4.1 must be executed on all possible permutations.

Once the discrete set $\mathcal{E}$ and the corresponding $\mathbf{x}_B^*(\Phi)$, $\mathbf{c}_B(\Phi)$ are found for all permutations, the calculation of z^* is straight-forward. First, the value functions are calculated as follows. Let $\Phi_{\min} \leq \delta \leq \Phi_{\max}$. For example, if the sensitivity of α is analysed, then $0 \leq \delta \leq 1$. For all values of δ , let $\mathbf{x}_B^*(\delta)$ be the basic components of the optimal bfs and $\bar{\mathbf{c}}_B(\delta)$ be the basic components of the modified reward vector. Then, the optimal reward of the LP in (4.3) given the reward vector $\bar{\mathbf{c}}(z, \delta)$ is given by

$$V(z, \delta) = \bar{\mathbf{c}}_B(z, \delta)^T \mathbf{x}_B^*(z, \delta) \quad (4.8)$$

The modified reward vector $\bar{\mathbf{c}}_B(z, \delta)$ is calculated for each permutation z from the stored values of $\mathbf{c}_B$ in Table 4.1 and the function $\mathbf{f}$, ie. $\bar{\mathbf{c}}_B(z, \delta) = \mathbf{c}_B(z, \Phi) \cdot \mathbf{f}(\frac{\delta}{\Phi}, \Phi)$. The method in Table 4.1 uses the optimality criterion in equation (4.7). Hence, for any parameter value δ there exists an optimal stored solution $\mathbf{x}_B^*$ and Φ is the stored parameter value for this solution.

The optimal permutation z^* is finally given by

$$z^*(\delta) = \underset{z}{\operatorname{argmax}} V(z, \delta) \quad (4.9)$$

By using equations (4.8) and (4.9) the Markovian decision problem can be solved for any parameter value δ by using the stored parameter values Φ and the stored solutions $\mathbf{x}_B^*$. The optimal solution $\mathbf{x}_B^*$ usually remains constant for a large value of parameter values δ , which ensures the efficiency of the LP sensitivity analysis.

Figure 4.1: *Example of sensitivity analysis of SPC decisions with four different permutations of event monitors. The optimal permutation z^* and actions depend on the parameter α .*

Consider the example in Figure 4.1. The figure shows z^* and the actions for one Markov state as functions of $\Phi = \alpha$. Only four different permutations of event monitors are considered in the example. The optimal actions (a_1 or a_2) are found by solving the LP for each permutation z and the optimal permutation z^* is found from the largest optimal reward, equation (4.9). When $\alpha \leq 0.3$, the optimal permutation is $z^* = 2 - 1 - 3$, but since the optimal action is to terminate, no monitor would be consulted for this value of α . When $0.3 < \alpha \leq 0.7$, we have $z^* = 1 - 2 - 3$ and since the optimal action is to continue, monitor 1 will be consulted first. When $0.7 < \alpha \leq 1$, we have $z^* = 1 - 3 - 2$ and again monitor 1 will be consulted first. The example shows that the decision made by the SPC does not necessarily change when the optimal permutation z^* changes. In general, a complete sensitivity analysis of the SPC requires knowledge of z^* and the actions for all values of Φ and for all Markov states.

4.4 Sensitivity Analysis Example

In this section we study the sensitivity analysis of the example given in section 3.3. Six monitors are available to the SPC and the fusion method in equation (3.17) is used.

The different fusion methods result in different Markov state transition probabilities P_{ij} . Hence, the different data fusion methods will only give different constraints in equation (4.2) and as a result different A matrices in equation (4.3). The monitors are numbered from 1 to 6 and have the characteristics shown in Figures 3.7 and 3.8. The monitor costs are chosen $k_1 = k_2 = k_3 = 2$ and $k_4 = k_5 = k_6 = 1$. The key SPC parameters to tune are the discount factor α and the state reward constant K . In sections 4.4.1 and 4.4.2 the sensitivities of these parameters are studied. Proposition 1 in Chapter 3 shows that multiplying *all* the state rewards by a factor ϵ has the same effect as dividing *all* the monitor rewards by the same factor ϵ . However, the proposition can not be used for individual state and monitor rewards. Sections 4.4.3 and 4.4.4 demonstrate the sensitivity analysis of individual rewards using the LP approach.

In all the following sections the sensitivity of the first monitor decision is demonstrated. Similar results can easily be achieved for decisions 2–6 by disabling all the monitors that have already been consulted. The first decision, however, is the most costly to analyse. Since all monitors are enabled, the sensitivity analysis has to evaluate the maximum number of monitor permutations.

4.4.1 Sensitivity of the Discount Factor α

Figures 4.2 and 4.3 show the sensitivity of the decisions made by the SPC as a function of the discount factor α . The method in Table 4.1 was used to find the optimal bfs $\mathbf{x}^*$ for all the permutations and equations (4.8) and (4.9) were used to find the optimal permutation z^* . All the process monitors are available to the SPC, and the figures show which process monitor is consulted first when the initial states are $\sigma(0) = s_2$ and $\sigma(0) = s_3$.

The results from the LP sensitivity analysis in Figure 4.2 and 4.3 were verified using the original SDP algorithm developed in Chapter 3. Note that one of the low cost monitors (4,5,6) is always consulted first. The behaviour of the SPC as α varies is explained as follows. When α is large, the continuation rewards are large and the low cost monitors dominate. When α is small, the continuation rewards are small and the reliable monitors dominate. Note the behaviour of the SPC when $\alpha \in [0.86, 0.90]$ in the figures. The optimal permutations for these α values are shown in Table 4.2. When

Figure 4.2: *Monitor consulted first by the SPC as a function of the discount factor α with all 6 monitors available. The initial state is $\sigma(0) = s_2 = 0.50$. Note that the optimal action is always to consult a monitor (4,5 or 6), except for $\alpha = 0$.*

Figure 4.3: *Monitor consulted first by the SPC as a function of the discount factor α with all 6 monitors available. The initial state is $\sigma(0) = s_3 = 0.75$. The optimal action is to consult a monitor (4,5 or 6), except for $\alpha < 0.20$.*

$\alpha \leq 0.86$, monitor 6 is always consulted first. Monitor 6 is the best individual monitor and is the obvious first choice when the continuation rewards are low. When $\alpha \geq 0.90$, the continuation rewards are large. The optimal permutation is $z^* = 4-5-6-3-2-1$ to pick up the extra continuation rewards of monitor 4 and 5. When $0.86 < \alpha < 0.90$, we have trade-offs between picking up extra monitor rewards and the terminating state rewards. For example, when $\alpha = 0.87$, the optimal permutation is $z^* = 4-6-5-3-2-1$. Note that monitor 5 and 6 have changed positions in the optimal permutation

0.86	$z^* = 6 - 5 - 4 - 3 - 2 - 1$
0.87	$z^* = 4 - 6 - 5 - 3 - 2 - 1$
0.88	$z^* = 4 - 6 - 5 - 3 - 2 - 1$
0.89	$z^* = 5 - 4 - 6 - 3 - 2 - 1$
0.90	$z^* = 4 - 5 - 6 - 3 - 2 - 1$

Table 4.2: *Optimal permutations of process monitors when $\alpha \in [0.86, 0.90]$.*

compared to the case when $\alpha \geq 0.90$. Reliable monitors are moved forwards in the optimal permutation when α decreases. When α changes from 0.90 to 0.89, we have the same effect with monitors 4 and 5. Monitor 5 is more reliable than monitor 4 and is moved forwards when α decreases.

The main difference between Figure 4.2 and 4.3 is the α value for which terminate becomes the optimal action. The discount factor only alters the continuation rewards. If the optimal action is terminate, the value function V is independent of α , which can be seen from Figure 4.3 where V is constant when $\alpha < 0.20$. When α increases, the continuation reward will sooner or later become larger than the termination reward and hence continue will be the optimal action for large α values. In state s_2 the termination reward is zero, and hence in Figure 4.2 the optimal action is always continue when $\alpha > 0$.

The discount factor value influences the average SPC cost and event recognition rate. In Chapter 3 good results were achieved for α values between 0.6 and 0.7. Except for the behaviour when $\alpha \in [0.86, 0.90]$, we see from Figure 4.2 and 4.3 that the local sensitivities of the SPC decisions are low. There are large intervals of α where the optimal actions are constant.

The analysis described above considers the sensitivity of the decisions generated off-line by the Markovian decision process. Another important feature of the sensitivity analysis is the ability to study the SPC decisions when faced with uncertain measurements. The SPC decisions react on-line in response to actual confidence measures that occur in operation. The confidence levels determine the state variable $\sigma(n)$. Hence, uncertainties in the confidence levels can result in incorrect $\sigma(n)$ state values. To determine the effects

on confidence level uncertainty, several α -plots such as Figures 4.2 and 4.3 must be compared. Note that Figures 4.2 and 4.3 are identical when $\alpha \geq 0.20$. Hence, when $\alpha \geq 0.20$, the first monitor consulted by the SPC is the same for the states $\sigma(0) = s_2$ and $\sigma(0) = s_3$. For this example, the sensitivity to uncertainties in the confidence measures is low when $\alpha \geq 0.20$. For other states $\sigma(0)$ in the state space (3.23), the α -plots will still be identical, but for different α values. For example, the α -plots for $\sigma(0) = s_3$ and $\sigma(0) = s_4$ are identical when $\alpha \geq 0.57$.

4.4.2 Sensitivity of the State Reward Constant K

Figure 4.4 shows the sensitivity of the decisions made by the SPC as a function of the state reward constant K . The discount factor value was chosen to be $\alpha = 0.75$. Again the method in Table 4.1 was used to find the optimal bfs $\mathbf{x}^*$ and the optimal permutation z^* .

As in the previous section all the process monitors are available to the SPC, and the figure shows which process monitor is consulted first when the initial states are $\sigma(0) = s_2$, $\sigma(0) = s_3$ and $\sigma(0) = s_4$. Note that one of the low cost monitors is always consulted first. When K is large, the state rewards dominate and the best individual monitor, monitor 6, is consulted first. As K decreases, the monitor rewards become relatively larger and monitor 4 and 5 are used first to pick up the extra monitor reward.

In Figure 4.4 the parameter K takes continuous values in the range $[0.5, 6]$. For all the permutations the linear program was solved on average 10.6 times (step 2 in Table 4.1) when $K \in [0.5, 6]$. An example of stored K values are given in Table 4.3 for one permutation. The example shows all the K values for which the LP is solved and the corresponding ϵ values (step 3 in Table 4.1). The stored K values and the corresponding

K	6.00	1.89	1.65	1.29	0.90	0.77	0.63	0.59	0.56	0.45
ϵ	0.31	0.87	0.78	0.70	0.86	0.82	0.94	0.95	0.80	0.00

Table 4.3: Example of stored K values for permutation $z = 6 - 5 - 4 - 3 - 2 - 1$.

$\mathbf{x}^*(K)$ and $\mathbf{c}_B(K)$ (step 4 in Table 4.1) describe the complete sensitivity analysis for any value of K . The algorithm in Table 4.1 is based on the optimality criterion in

Figure 4.4: *Monitor consulted first by the SPC as a function of the state reward constant K with all 6 monitors available. Identical results were obtained for $\sigma(0) = s_2 = 0.50$, $\sigma(0) = s_3 = 0.75$ and $\sigma(0) = s_4 = 0.875$.*

equation (4.7). Hence, the results in Table 4.3 show, for example, that the optimal solution for $K = 6.00$ and $z = 6 - 5 - 4 - 3 - 2 - 1$ is also optimal in the *entire* continuous range $K \in [1.89, 6.00]$.

Once the algorithm in Table 4.1 is completed for all permutations, no further calculations are required to solve the Markovian decision problem for *any* value of $K \in [0.5, 6]$. The stochastic dynamic programming approach, on the other hand, has to be solved from scratch for each new value of K . For example, if the discretisation step of K is chosen to be 0.01, a brute-force sensitivity analysis of $K \in [0.5, 6]$ requires the SDP to be solved 550 times for each permutation. Puterman (1994, p. 230) estimated the SDP algorithm to be 12 times faster than the *LP* algorithm. For this example, the LP approach to sensitivity analysis is $\frac{550}{12 \times 10.6} = 4.3$ times faster than a brute-force SDP approach. The benefit of using the LP approach to sensitivity analysis compared to brute-force methods depends on the discretisation step of the model parameters. As the discretisation step of the parameters, in this case K , decreases, the computational benefit of using the LP approach increases.

4.4.3 Sensitivity of the State Rewards $r(s_i, a_1)$

A change in the state reward constant K changes all the state rewards $r(s_i, a_1)$. In this section the sensitivity analysis of the SPC decisions as only one of the state rewards changes is demonstrated. The discount factor and the state reward constant are fixed at $\alpha = 0.75$ and $K = 2$. To compute $\bar{c}$ in Table 4.1, all occurrences of $r(s_i, a_1)$ in the LP reward vector c , equation (4.6), are changed to $\epsilon r(s_i, a_1)$.

Figure 4.5 shows the decisions made by the SPC as $r(s_3, a_1)$ takes values in $[0, 6]$. All the process monitors are available to the SPC, and the figure shows which process monitor is consulted first when the initial state is $\sigma(0) = s_3$. A change in a state reward directly alters the termination rewards. Hence, when the initial state is s_3 and $r(s_3, a_1)$ becomes sufficiently large, the optimal action will be a_1 -terminate. Since the termination reward is $Kr(s_3, a_1)$, the value function V is a linear function of the state reward when the optimal action is terminate, see Figure 4.5.

Figure 4.5: Monitor consulted first by the SPC as a function of the state reward $r(s_3, a_1)$ with all 6 monitors available. The initial state was chosen $\sigma(0) = s_3 = 0.75$. When monitor=0, the optimal action is terminate.

In Figure 4.5 the parameter $r(s_3, a_1)$ takes continuous values in the range $[0, 6]$. For all the permutations the linear program was solved on average 20.4 times (step 2 in Table 4.1) when $r(s_3, a_1) \in [0, 6]$. If the discretisation step of $r(s_3, a_1) \in [0, 6]$ is chosen to be 0.01, then the brute-force SDP approach must be solved 600 times. For the sensitivity analysis of the parameter $r(s_3, a_1)$, the LP approach is $\frac{600}{12 \cdot 20.4} = 2.5$ times faster than

a brute-force SDP approach.

4.4.4 Sensitivity of the Monitor Rewards l_i

A multiplication of the state rewards by a factor $\epsilon > 0$ is equivalent to a division of the monitor rewards by the same factor, see Proposition 1. Hence, the sensitivity analysis for the state reward constant can also be used to study the effects of changes on all the monitor rewards simultaneously. However, the sensitivity analysis of a single monitor reward l_i is different and is demonstrated in this section. Again, we chose $\alpha = 0.75$ and $K = 2$.

Figure 4.6: *Monitor consulted first by the SPC as a function of the monitor reward l_6 with all 6 monitors available. The initial state was chosen $\sigma(0) = s_2 = 0.50$.*

As in section 3.3 the monitor rewards are found from $l_i = k_{\max} - k_i$. With $k_1 = k_2 = k_3 = 2$ and $k_4 = k_5 = k_6 = 1$ the nominal value for l_6 is 1 and the continuation monitor reward is found from equation (3.4). Figure 4.6 shows an example of the decisions made by the SPC as the monitor reward l_6 takes values in $[0, 0.4]$. When $l_6 < 0.04$, monitor 6 is no longer the optimal. Monitor 5 which has a large monitor reward and medium confidence takes over as the optimal choice.

In Figure 4.6 the parameter l_6 takes continuous values in the range $[0, 0.4]$. For all the permutations the linear program was solved on average 1.9 times (step 2 in Table 4.1) when $l_6 \in [0, 0.4]$. The stored l_6 values are given in Table 4.4 for one example

l_6	0.40	0.28	0.22
ϵ	0.70	0.79	0.00

Table 4.4: *Example of stored l_6 values for permutation $z = 1 - 2 - 3 - 4 - 5 - 6$.*

permutation. The example shows all the l_6 values for which the LP is solved and the corresponding ϵ values (step 3 in Table 4.1). If the discretisation step of $l_6 \in [0, 0.4]$ is chosen to be 0.01, then the brute-force SDP approach must be solved 40 times. For the sensitivity analysis of the parameter l_6 , the LP approach is $\frac{40}{12 \cdot 1.9} = 1.8$ times faster than a brute-force SDP approach.

4.5 Discussion and Conclusion

The LP approach to sensitivity analysis demonstrated in this chapter provides a computationally efficient algorithm when the number of parameter values to be considered is large. The sensitivity analysis was demonstrated for several examples where the LP approach was up to 4.3 times faster than a brute-force approach using the SDP algorithm for each new model parameter. Solving the new problems from scratch is both wasteful and unnecessary. The LP optimality criterion in equation (4.7) tells when a solution is no longer optimal. Hence, new problems only have to be solved again from scratch when the optimality test fails.

The example in section 4.4 demonstrates the sensitivity analysis of measurements obtained during execution. The ability to study the effects of uncertain confidence measurements is an important feature of the analysis. In contrast to open-loop sensor planning, we are able to analyse the behaviour of the real-time sensor selection strategies. Analysis of real-time behaviour is important in noisy and uncertain autonomous systems, where open-loop sensor planning solutions often fail to achieve the desired goals. The sensitivity analysis is able to find SPC parameter intervals where noisy measurements has little or no effect on the optimal SPC actions. For example, Figures 4.2 and 4.3 show identical optimal actions when the parameter $\alpha \geq 0.20$ for the Markov states s_2 and s_3 . Hence, the sensitivity analysis is important in the robust tuning of

the SPC.

The key SPC model parameters to tune are the discount factor α and the state reward constant K . Figures 4.2, 4.3 and 4.4 show that the sensitivities to local changes are small. The figures contain large areas where the optimal decisions remain constant. For the tuning of the SPC, it is desirable to choose the parameters from these low sensitivity areas. However, low sensitivity to changes in the model parameters is not the only tuning criterion. In practice, the tuning of the parameters require consideration of both performance (such as high recognition rates and low sensing costs) and the local sensitivities.

The main contribution of this chapter is the development and demonstration of the sensitivity analysis of the sensory perception controller. The analysis aids in the robust tuning of the SPC by finding low sensitivity areas for the model parameters. The analysis is performed by exploring the relationship between discounted Markovian decision theory and linear programming. The sensitivity analysis is a fundamental part of the solution methodology by providing insight into both the off-line sensor planning and the on-line behaviour during execution. For example, Figure 3.10 shows the average recognition rate and cost as functions of the discount factor α . By using results from the sensitivity analysis, we can find intervals of α where the optimal decisions remain constant. Hence, the recognition rates and monitor costs also remain unchanged in such intervals. By using the sensitivity analysis presented in this chapter, we are able to guarantee large recognition rates and low costs for a large number of SPC parameters. To successfully control discrete event autonomous systems, large event recognition rates and low cost SPC solutions are required.

Chapter 5

Case Study I: Control of Sensory Perception in Robotic Assembly

5.1 Introduction

Inaccurate sensors and geometrical uncertainties frequently cause the failure of real-time robotic assembly systems. Single sensor systems provide high bandwidth solutions, but often lack the required robustness in noisy environments. Multiple sensor systems often have increased robustness, but may fail to meet response time requirements. Control of sensory perception aims to keep the sensing costs low while increasing robustness by actively selecting different sensing techniques in real-time. In normal operation only a few sensing techniques are needed. Then, as the quality of the sensory perception decreases, additional sensors are utilised.

The discrete event framework has been successfully applied to the modelling and control of robotic assembly tasks, McCarragher and Asada (1995a,b). They define a discrete event in an assembly task as a change in the state of contact. Changes in the dynamic equations of motion describing the process are significant at the occurrence of the events. McCarragher and Asada write:

The abstraction to discrete event modelling highlights the necessary transitions for successful assembly. Moreover, the abstraction allows for planning on a task level rather than the cumbersome process of exact trajectory planning. Lastly, a

discrete event approach provides the means to recognise and recover from unwanted situations.

However, the difficulties involved in accurate discrete event recognition still limits the discrete event approach to assembly. In this chapter we present the new SPC approach to event recognition. Four event monitors are available and the SPC actively selects different event monitors. We present experiments demonstrating the benefits of the sensory perception control architecture. The dynamic sensing strategy achieves an event recognition rate as high as 98% with less than half the sensing costs compared to a fixed multi-sensor system using all the monitors for every event occurrence.

5.2 Literature Survey

In this section we present related work in the area of contact recognition in robotic assembly. Approaches using force/torque and position/velocity measurements are discussed. Several of the approaches are suitable process monitor candidates for use in a real-time sensory perception control architecture of robotic assembly.

Eberman and Salisbury (1994) describe a method for detecting state changes in a dynamic system. Four states are considered, impacts, sliding across a surface, no contact and grasping contacts. For each of these states, a statistical force-signal model is built. The no contact model, for example, is chosen to be the Gaussian distribution $N(\mu_0, V_0)$. The validity of the models is checked against spectrograms of empirical data. Hypothesis testing is used to recognise different states, where there is one stochastic representation of the force measurements for each state. Since the number of possible states is only four, the information given by the method is limited. For example, in robotic assembly the state "sliding across a surface" covers a whole range of situations, since there may be many surfaces involved.

In the work by Hirai (1989) the highlight is on the recognition of the contact formation between the workpiece and the environment, not the change in contact. Assuming quasi-static motions of the workpiece, negligible friction and noiseless sensing, they use the theory of polyhedral convex cones to find the set of statically admissible contact forces and moments between the workpiece and the environment. The possible contact

formations are then the formations where the measured forces/moments are contained in this set. This method only applies to systems with quasi-static motions, negligible friction and noiseless sensing.

Trinkle and Zeng (1995) describe a motion prediction method for a system with several bodies and multiple points of contact. Assuming quasi-static, planar rigid-body motions, they have developed a model to predict the motion of a single rigid workpiece contacted at multiple points by a general planar manipulator. The method allows for sliding and rolling contacts. Trinkle and Zeng plan to extend the formulation to three dimensional motions and multiple workpieces. This method applies only to systems with quasi-static motions.

Dutr, Bruyninckx and De Schutter (1996) present a different approach based on the energy of the manipulated object. The energy is the sum of the kinetic and potential energies. A total energy error function is defined, which depends on the geometrical uncertainty parameters v . The values of v for which the total energy error function reaches a minimum are found and these values describe the contact situation. The contact model is validated by observing only one signal, the same total energy error function. Two advantages of the method is that it does not require collection of training examples and the contact models are validated using only one signal. One limitation is the assumption of quasi-static motions.

Dutr, et.al. (1996), Katupitiya, et.al. (1996) present a method using an extended Kalman filter for the identification of the grasping and geometrical contact uncertainties for a modelled contact topology. The approach is based on virtual contact manipulators, where different manipulators are used for grasping, environment and contact uncertainties. The monitoring of contact state changes is solved by carrying out a statistical test on the sum of normalised and squared innovations of the Kalman filter. One advantage of the method is the incorporation of both force and velocity measurements for successful monitoring. One limitation of the results is off-line identification.

In the work by Badano, et.al. (1991) a different approach to robotic assembly is used. They show experiments for a planar peg-in-the-hole assembly where the position error between the peg and the hole is compensated for by slight random movements. One limitation of this approach is that the random movements might change the contact

state. Also, for a more complex assembly task, there is no guarantee that random movements will successfully insert the workpiece in the environment.

The traditional approach to monitoring and control of robotic assembly tasks involves accurate position measurements and special-purpose fixtures. The traditional approach is costly due to long load time and effort required to design and manufacture special-purpose fixtures, manual set-up and change of fixtures, and overhead cost associated with storing and retrieving a multiplicity of fixtures. Thus, the traditional approach is not suited to the flexible manufacturing environment where production volume is low and product variation is high, Shirinzadeh (1993). Moreover, the use of highly accurate fixtures for workpiece positioning requires equally accurate position measurements of the manipulator's end-effector, which adds to the cost of the overall manufacturing system.

To overcome some of the problems with position based assembly, several solutions have been developed. One example is presented by Shirinzadeh (1993) where the design of reconfigurable fixtures are described. By using reconfigurable fixtures for several different assembly tasks, the costs associated with single-purpose fixtures are reduced. However, accurate position measurements of the manipulator's end-effector are still needed.

In the paper by Gürocak and de Sam Lazaro (1995) they use strain gauge readings to account for inaccuracies in the position measurements for a cylindrical peg-in-the-hole assembly. The strain gauges are located in a compliant wrist and used to calculate the magnitude of the position error and its direction. The manipulator is then moved in the opposite direction of the error. This method is sensitive to the force measurements and the noise in the measurements reduce the effectiveness of the method.

5.3 Discrete Event Model of Robotic Assembly

The discrete event model of the planar assembly task is based on the motion constraints of the workpiece in the manipulator's gripper. The equation of motion of the workpiece

in free-space is generally written as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (5.1)$$

where $\mathbf{x}(t)$ is the continuous time space vector, and $\mathbf{u}(t)$ is the input vector. As the workpiece interacts with the inertially fixed environment, the free-space dynamics of equation (5.1) become constrained. For each edge-surface contact, this constraint may be written as

$$g_j(\mathbf{x}(t)) = 0 \quad (5.2)$$

where g_j is the constraint equation for the j th edge-surface contact.

Figure 5.1: Model of contact states for a planar assembly task.

Definition 6 (Discrete State) A discrete state is defined as a contact formation consisting of zero, one or more edge-surface contacts between the workpiece and the environment. The set of discrete states is then the set of all possible contact formations, McCarragher and Asada (1995a). $\square$

We define a set of contact states as shown in Figure 5.1 for a planar assembly task. Each contact state has a unique set of zero, one or more constraints on the form given by equation (5.2).

Figure 5.2: *Contact state network for the planar assembly task.*

The discrete events occurring during the assembly task are now defined using the definition of the contact states.

Definition 7 (Discrete Event) *A discrete event in robotic assembly is a change of contact state. The events are discrete in time and describe a gain or a loss of a single edge-surface contact between the workpiece and the environment.* $\square$

We use the notation from Astuti (1995) where γ_i is used for contact states, $M = \{\gamma_1, \gamma_2, \dots, \gamma_m\}$ is a finite set of discrete states and $E \subset M \times M$ is a finite set of discrete events. Any discrete event e_k is then given by an ordered pair $e_k = (\gamma_i, \gamma_j) \in E$, describing a change from the discrete state γ_i to γ_j . Given a contact state γ_i , only events from a subset of E are possible. We define this subset of admissible events as $E_{\gamma_i} \subset E$. The subset E_{γ_i} is determined from the contact state network of the assembly task, see Figure 5.2. For the contact state network shown in Figure 5.2 there are 25 possible discrete events as defined in Table 5.1.

$e_1 : \gamma_1 \rightarrow \gamma_2$	$e_6 : \gamma_3 \rightarrow \gamma_1$	$e_{11} : \gamma_4 \rightarrow \gamma_6$	$e_{16} : \gamma_6 \rightarrow \gamma_4$	$e_{21} : \gamma_9 \rightarrow \gamma_7$
$e_2 : \gamma_1 \rightarrow \gamma_3$	$e_7 : \gamma_3 \rightarrow \gamma_2$	$e_{12} : \gamma_5 \rightarrow \gamma_3$	$e_{17} : \gamma_7 \rightarrow \gamma_5$	$e_{22} : \gamma_9 \rightarrow \gamma_{10}$
$e_3 : \gamma_1 \rightarrow \gamma_6$	$e_8 : \gamma_3 \rightarrow \gamma_4$	$e_{13} : \gamma_5 \rightarrow \gamma_4$	$e_{18} : \gamma_7 \rightarrow \gamma_8$	$e_{23} : \gamma_{10} \rightarrow \gamma_9$
$e_4 : \gamma_2 \rightarrow \gamma_1$	$e_9 : \gamma_4 \rightarrow \gamma_3$	$e_{14} : \gamma_5 \rightarrow \gamma_7$	$e_{19} : \gamma_7 \rightarrow \gamma_9$	$e_{24} : \gamma_{10} \rightarrow \gamma_{11}$
$e_5 : \gamma_2 \rightarrow \gamma_3$	$e_{10} : \gamma_4 \rightarrow \gamma_5$	$e_{15} : \gamma_6 \rightarrow \gamma_1$	$e_{20} : \gamma_8 \rightarrow \gamma_7$	$e_{25} : \gamma_{11} \rightarrow \gamma_{10}$

Table 5.1: *Possible events for an assembly task with 11 contact states.*

5.4 Discrete Event Control Architecture

The discrete event control architecture is shown in Figure 5.3. The control structure is divided into a continuous and a discrete part. The continuous part controls the velocity of the workpiece in Cartesian space, where the velocity reference is set by the DEC. The process plant, consisting of the manipulator, the workpiece and the environment, is nonlinear and time-variant. Hence, different continuous controllers and velocity references are needed for different situations. For example, the DEC might vary the robot PI gains for each individual joint or change the velocity commands depending on the discrete contact state γ . The purpose of the sensory perception control architecture is to detect and classify the time-varying situations, represented by the occurrence of discrete events.

Figure 5.3: Block diagram of discrete event system with control of sensory perception for robotic assembly.

The control system is described briefly as follows. γ_d is the desired contact state, $\mathbf{F}$ is the force/torque measurements from the interaction between the robot and the environment, $\mathbf{v}_d$ is the Cartesian velocity reference from the DEC, $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the position and velocity measurements from the robot and $\boldsymbol{\tau}$ is the commanded motor torques from the continuous controllers. The function $\mathbf{J}^\#(\mathbf{q})$ is a transformation from Cartesian space

velocities to joint space velocities, ie. $\dot{\mathbf{q}}_d = \mathbf{J}^\#(\mathbf{q})\mathbf{v}_d$. In the experiments we use the closed-loop tracking solution presented in the paper by Chiacchio, et.al. (1991, page 413). The closed-loop solution avoids the long-term drifts which occur with the simple inverse Jacobian approach. The manipulator joints are controlled individually by PI velocity controllers.

The discrete event model and controller are described by McCarragher and Asada (1995a,b). First, a discrete event model of the assembly process is developed, such as the model in Figure 5.1 and Table 5.1. Based on this model, the DEC directs the assembly process through a series of events to the desired end state in the discrete event model. The DEC uses a discrete event trajectory which specifies the discrete controls. The discrete control vector is then mapped into continuous velocity commands to be sent to the robot arm. The execution of these velocity commands will ideally cause the system to pass through the desired events. Since the workpiece is in contact with the environment, the dynamic equations of motion depend on the discrete contact state. Astuti (1995) describes how to find the constraint equations for each contact state by using distance functions. The desired Cartesian velocity commands $\mathbf{v}_d$ are then found by solving an optimisation problem with linear constraints in the commanded velocities.

It is extremely difficult to guarantee that the velocity commands will guide the system through the desired events. To overcome this problem, the SPC is used to give the DEC feedback, ie. to tell which events occur. The individual process monitors use both the manipulator joint angles and the force/torque measurements resulting from interaction between the robot arm and the assembly environment. The outputs of the process monitors are recognised discrete events and associated confidence levels. The SPC sequentially selects one or more of the process monitors to produce one final recognised state γ^* as described in Chapter 3. The final SPC output γ^* is compared with the desired contact formation γ_d . If there is a mismatch between these two, an error has occurred and proper control actions must be given to correct the error. Ideally, the commanded velocities from the DEC will guide the system through the desired event trajectory. However, since it is difficult to achieve this, process monitoring techniques play a very important role in discrete event control. By using active control of sensory perception the system is able to detect errors and unwanted situations, which is an

important step towards an intelligent control structure with the ability to recover from errors.

5.5 Process Monitors

The sensory perception control architecture in Figure 5.3 uses four process monitoring techniques. The Hidden Markov Models (HMM) process monitor is described in detail in Appendix A. The distance functions (DF) monitor is described in Appendix B. The multilayer perceptron (MLP) monitor is described in Appendix C and the qualitative reasoning (QR) monitor is described in Appendix D.

5.5.1 Geometric Modelling

The DF process monitor requires a geometrical model of the assembly task. The distance functions are defined by Astuti (1995). The distance between a surface ψ and an edge ϕ , denoted by $h_{\psi\phi}$, is defined as the nearest distance between ϕ and coordinates on the surface ψ . The different surfaces and edges for the L-shaped assembly considered in this chapter are shown in Figure 5.4.

Figure 5.4: Left: Labels of all the surfaces and the edges of the planar assembly task. The position vectors for the surfaces and the edges in the inertial frame are given in Table 5.2. Right: Position and force/torque coordinate frames.

From the manipulator joint positions and the forward kinematics we obtain the 3 DOF position measurements (P_x, P_y, θ) tied to the workpiece. The inertial coordinate frame (x_0, y_0) is located at the base of the manipulator. Using l_ϕ as the length of the surfaces and l_h as the length of the hole, we can describe all the edges and the surfaces by the measurements (P_x, P_y, θ) , see Table 5.2. $(P_{xd}, P_{yd}, 0)$ is the 3 DOF position of the edge d and describe the *a priori* estimated position of the environment. Figure 5.4 also shows the coordinate frame for the force/torque measurements F_x , F_y and M_z used by the HMM, the MLP and the QR process monitors.

$\mathbf{d}_b = (P_x, P_y)$	
$\mathbf{d}_c = \mathbf{d}_b + (-l_4 \cos \theta, -l_4 \sin \theta)$	
$\mathbf{d}_a = \mathbf{d}_b + (l_3 \sin \theta, -l_3 \cos \theta)$	
$\mathbf{d}_{11} = \mathbf{d}_b + (-l_2 \cos \theta + (l_1 + l_3) \sin \theta, -(l_1 + l_3) \cos \theta - l_2 \sin \theta)$	
$\mathbf{d}_{12} = \mathbf{d}_{11} + (-l_1 \sin \theta, l_1 \cos \theta)$	
$\mathbf{d}_{52} = \mathbf{d}_c + (l_5 \sin \theta, -l_5 \cos \theta)$	
$\mathbf{d}_d = (P_{xd}, P_{yd}) = (235.5\text{mm}, 81.1\text{mm})$	$\mathbf{d}_{51} = \mathbf{d}_c$
$\mathbf{d}_e = \mathbf{d}_d + (l_h, -l_9)$	$\mathbf{d}_{61} = \mathbf{d}_d + (-l_6, 0)$
$\mathbf{d}_f = \mathbf{d}_d + (l_h, 0)$	$\mathbf{d}_{62} = \mathbf{d}_d$
$\mathbf{d}_{21} = \mathbf{d}_{12}$	$\mathbf{d}_{71} = \mathbf{d}_d$
$\mathbf{d}_{22} = \mathbf{d}_a$	$\mathbf{d}_{72} = \mathbf{d}_d + (0, l_7)$
$\mathbf{d}_{31} = \mathbf{d}_a$	$\mathbf{d}_{81} = \mathbf{d}_e + (l_8, 0)$
$\mathbf{d}_{32} = \mathbf{d}_b$	$\mathbf{d}_{82} = \mathbf{d}_e$
$\mathbf{d}_{41} = \mathbf{d}_b$	$\mathbf{d}_{91} = \mathbf{d}_e$
$\mathbf{d}_{42} = \mathbf{d}_c$	$\mathbf{d}_{92} = \mathbf{d}_f$

Table 5.2: Positions of all edges and surfaces obtained from the 3-DOF measurements P_x , P_y and θ .

5.5.2 Event Detection

The amount of force information available when there is a change in the constraint equations is large. This is for example illustrated in Figure A.3 which is a similar result to Figure 1 in Eberman and Salisbury (1994). To be able to extract this large amount of information, an event detector is needed. A change in the constraint equations is characterised by a sudden change in at least one of the force measurements. The changes in the force measurements are found from the time derivatives of F_x and F_y .

$$H_x(t) = \frac{1}{\Delta T} (F_x(t) - F_x(t - \Delta T))$$

$$H_y(t) = \frac{1}{\Delta T}(F_y(t) - F_y(t - \Delta T))$$

ΔT is considered a filter design parameter. The frequency response of $H_x(t)$ is given by

$$\left| \frac{H_x(j\omega)}{F_x(j\omega)} \right| = |1 - e^{-j\omega\Delta T}| \quad (5.3)$$

We have that $\max |1 - e^{-j\omega\Delta T}| = 2$ when $\omega\Delta T = \pi + n2\pi$, $n = 1, 2, \dots$. The filter cut-off frequency is then given by $\omega_c = \frac{\pi}{\Delta T}$. A large value of ΔT will result in differentiation of the force signals for low frequencies only. The simplicity of the filters in the time-domain, $H_x(t)$ and $H_y(t)$, is an advantage in fast on-line event detection.

The event detector used here is described by the following threshold test.

$$H_x(t)^2 + H_y(t)^2 \geq (\Delta F)^2 \quad (5.4)$$

As in the paper by McCarragher and Asada (1993), we determine the threshold by looking at the standard deviation of the measurements. We do not want the effects from the friction forces to trigger the threshold test, so the workpiece was slid across a surface and $(\Delta F)^2$ was chosen to be a 3σ deviation from the mean value of $H_x(t)^2 + H_y(t)^2$. Note that the choice of ΔT is dependent on several factors such as the dynamics of the manipulator, the dynamics of the force sensor and the surface properties. In the experiments in section 5.6 we used $\Delta F = 9.5N$ and $\Delta T = 1/3sec$ which gives a filter cut-off frequency $\omega_c = 3\pi rad/s = 1.5Hz$.

5.6 Experiments

The experimental setup consists of an Eshed Scorbot five degree-of-freedom manipulator and a JR3 force/torque sensor, see Figure 5.5. The dimensions of the experimental setup are given in Figure 5.6.

Once a discrete event is detected by the method in section 5.5.2, any of the four process monitors illustrated in Figure 5.3 can be consulted by the SPC.

Figures 5.7 and 5.8 show the force/torque and the position measurements *immediately* after an event has been detected by the algorithm in section 5.5.2. If the DF process monitor is consulted, it uses the static position measurements P_x , P_y and θ to calculate

Figure 5.5: *Eshed Scorbot 5-DOF manipulator and JR3 force/torque sensor used in the experiments. The workpiece is attached directly to the force/torque sensor.*

Figure 5.6: *Dimensions of the experimental setup.*

the set of relevant distance functions. The forward kinematics of the manipulator and the workpiece directly translate the joint angles to the 3-DOF position measurements corresponding to the workpiece edge d_b . All the other edges and surfaces are found from d_b using the geometric model of assembly, see Table 5.2. If the HMM monitor or the QR monitor are consulted, they use the dynamic force/torque measurement F_x , F_y , M_z . If the MLP monitor is consulted, it uses both position and force/torque

Figure 5.7: Example of force/torque and position measurements for the assembly event sequence $e_1 \rightarrow e_5 \rightarrow e_8 \rightarrow e_{10}$

Figure 5.8: Example of force/torque and position measurements for the assembly event sequence $e_{14} \rightarrow e_{19} \rightarrow e_{22}$

measurements for the contact state monitoring of assembly. The goal of the SPC is to sequentially select one or more of these different monitoring techniques to successfully recognise the contact states as they occur during the assembly operation. Note from Figures 5.7 and 5.8 the dynamic behaviour of the force measurements. Both the HMM and the MLP monitors use frequency-domain measurements and windowing techniques to extract as much information as possible from these measurements. However, the thorough analysis of the forces incurs a large cost. The CPU times of the four process

Process Monitor	CPU Sec.	Rec. Rate %
1 – Distance Functions	0.10	79.0
2 – Multilayer Perceptron	0.59	87.0
3 – Qualitative Reasoning	0.08	84.0
4 – Hidden Markov Models	0.87	86.0

Table 5.3: *Required CPU times (CPU seconds) and average recognition rates of the process monitors in real-time operation.*

monitors are given in Table 5.3. The DF monitor uses only the measurements P_x , P_y and θ and the QR monitor uses only the change of force measurements ΔF_x and ΔF_y . As a result, both these methods have low computational complexities.

The SPC and the four different process monitors were tested on a sample set of 100 discrete events. The number of events successfully recognised by the monitors are shown in Table 5.3. The thorough force analysis by the HMM and MLP monitors result in reasonably large recognition rates. Note, however, that even larger recognition rates have been reported for the HMMs (97%) by Hovland and McCarragher (1998) and for the MLP (94%) by Hovland and McCarragher (1997a). In this chapter we are demonstrating the efficiency of the sensory perception controller. Hence, the average recognition rates of the HMM and the MLP monitors are reduced on purpose. The reduction in recognition rates is achieved by reducing the number of force training examples for these two methods.

i	1	2	3	4	5	6	7	8	∞
s_i	0.0	0.5	0.75	0.875	0.9375	0.96875	0.984375	0.9921875	∞
$R(s_i, a_1)$	0	0	K	$2K$	$3K$	$4K$	$5K$	$6K$	0

Table 5.4: *Markov state space and state rewards $R(s_i, a_1)$.*

The Markov decision state space and the state rewards are defined in Table 5.4. The discount factor was chosen $\alpha = 0.85$. The monitor rewards are defined by the costs of the process monitors given by Table 5.3. Let k_i be the cost of monitor i given in Table

5.3 by the CPU times. In the experiments we used the following monitor rewards.

$$l_i = 2 \left(\max_i k_i \right) - k_i \quad \text{where } i \in \{1, 2, 3, 4\} \quad (5.5)$$

The state reward factor was chosen $K = 2(\max_i k_i) = 1.74$. The motivation for the parameter selection above is that increasing a low confidence is relatively easy, while increasing a large confidence is difficult. By the choice above, the same increase in reward is given for a change from s_2 to s_3 as for a change from s_7 to s_8 , even if s_2 to s_3 increases the confidence by 0.25 while for s_7 to s_8 the increase in confidence is only 0.0078125. Equation (5.5) makes all the monitor rewards positive. Hence, the decision process will always consult a monitor when the confidence levels are low. By choosing $K = 2(\max_i k_i)$, the reward for choosing the fastest monitor is smaller than the smallest non-zero state reward. Hence, the state rewards dominate the monitor rewards, which ensures that the decision process will terminate when a relatively large confidence level is reached.

Figure 5.9: Monitor sequence $z = 1 - 3 - 2$ showing measurements, recognised discrete events $e(n)$, Markov decision states $\sigma(n)$ and total CPU time.

Figure 5.9 shows an example of how the SPC sequentially consults the monitors for an occurrence of the event $e_{11} : \gamma_4 \rightarrow \gamma_6$, where consensus based decision fusion is used. The SPC first consults monitor 1. Monitor 1 measures the Cartesian positions P_x , P_y and θ for the event recognition. Event $e(1) = e_9$ is incorrectly classified by monitor 1 with a confidence of 0.500. The nearest Markov decision state is $\sigma(1) = s_2$, see Table 5.4. Because of the low confidence, the SPC decides to consult monitor 3 next. Monitor 3 is based on qualitative reasoning and measures the force changes ΔF_x and ΔF_y . Event e_{11} is correctly classified with a confidence of 0.538. There is no

consensus among monitor 1 and 3, hence the SPC uses the data from the last monitor, see equation (3.19). The new recognised event becomes $e(2) = e_{11}$ and the nearest Markov state is again $\sigma(2) = s_2$. Again, the low confidence causes the SPC to consult monitor 2 next. Monitor 2 is the multilayer perceptron classifier which uses dynamic force measurements F_x , F_y and M_z plus the Cartesian position measurements. Monitor 2 correctly classifies event e_{11} with a confidence of 0.586. There is now a consensus among monitor 2 and 3. They both recognise event e_{11} . Hence, the new confidence level is calculated from equation (3.19), ie. $C = 1 - (1 - s_2)(1 - 0.586) = 0.793$. The resulting nearest Markov state is $\sigma(3) = s_3$. The SPC now terminates the decision process. The event $e^* = e(3) = e_{11}$ is mapped to the final recognised contact state $\gamma^* = \gamma_6$ which is sent to the discrete event controller. The total CPU time used by the SPC for this example is 0.67. This example is typical for the operation of the SPC. The fastest techniques are normally consulted first. The slower but more reliable monitors are then consulted only when the confidence levels are low. Hence, the SPC is able to achieve event recognition with large confidence levels while keeping the sensing costs low.

Figures 5.10 and 5.11 show the average costs and recognition rates for the four individual monitors, for all the monitors combined and for the three different SPC fusion methods. The performances were tested on a sample set of 100 discrete events. In the figures DF is the distance functions process monitor, MLP is the multilayer perceptron, QR is the qualitative template matching method and HMM is the Hidden Markov Model method. SPC1 is the sensory perception method using the last monitor, SPC2 is sensory perception based on maximum confidence and SPC3 is sensory perception based on consensus. ALL is a consensus-based method on all the monitors described as follows.

$$\begin{aligned} e^* &= \operatorname{argmax}_{e_i} \sum_{j=1}^4 S_j(e_i) \\ \gamma^* &= \tau(e^*) \end{aligned} \quad (5.6)$$

where $S_j(e_i)$ is the score for event e_i from process monitor j as described in the appendices. As expected, the best recognition result (99%) was obtained using all the monitors for every event occurrence. The best SPC recognition rate was achieved by using the consensus based fusion method (98%). However, the average SPC sensing costs are less than half the cost of using all the monitors. Moreover, the SPC recogni-

Figure 5.10: Average costs of the process monitors and the sensory perception controller.

Figure 5.11: Event recognition rates of the process monitors and the sensory perception controller.

tion rates are higher than any individual event monitor. Similar results were presented by Hovland and McCarragher (1997b) when using three process monitors for robotic assembly. The experiments demonstrate significant cost savings of the SPC compared to multi-sensor fusion techniques without drastically reducing the event recognition rates.

5.7 Discussion and Conclusions

The new SPC approach to event identification increases both robustness and performance of robotic assembly tasks. In an industrial setting, the sensing cost savings compared to multi-sensor fusion techniques increase the number of assembly operations per time unit. The increased recognition rates compared to single-sensor systems increase the robustness and reduce the need for human intervention.

One advantage of the discrete event framework is the ability to model complex assemblies. McCarragher (1996) presented the discrete event control of a 3D assembly task. Naturally, the number of contact states and discrete events becomes large. However, given any contact state, only a limited number of discrete events are physically possible. An assembly task with a large number of contact states usually has a small number of admissible events for each contact state. Hence, for the process monitoring techniques only the admissible events have to be evaluated and a high performance classification is still possible.

In general, there is no guarantee that the information in any measurement signals is sufficient for successful contact monitoring. For complex assembly tasks it is desirable to make use of a wide range of sensory information, including force/torque, position/velocity and vision. Moreover, it is desirable to combine the benefits from static, quasi-static and dynamic methods. The sensory perception controller for assembly demonstrated in this chapter provides a framework for the efficient incorporation of all these methods. The only requirement of the monitors is the generation of confidence levels of the recognised contact events.

As demonstrated in the experiments, the average recognition rate of the SPC is better than any individual monitor. Moreover, the sensing costs are less than half the cost of a fixed multi-sensor system using all the event monitors for every event occurrence. In particular, for a sample set of 100 discrete events occurring during assembly operations, the SPC achieved an average recognition rate of 98% using consensus based decision fusion.

In this chapter we have presented a new and unique approach to contact state identification of robotic assembly. The problem formulation highlights sensing at occurrences of discrete events. The events correspond to motion constraints changes. As a result, control of sensory perception is not required continuously which reduces the need for heavy computational power of robotic assembly cells. The SPC is formulated as a sequential Markov decision process and solved using stochastic dynamic programming. Both the selective use of event monitors and the computational efficiency of the real-time SPC ensure robust and high performance control of robotic assembly.

Chapter 6

Case Study II: Control of Sensory Perception in Mobile Navigation

6.1 Introduction

Sensory perception is one of the main bottlenecks in mobile navigation. The most common approach to sensory perception in mobile navigation is to continuously use all available sensors. Different sources of information are combined into one representational format. This operation is often referred to as multi-sensor fusion, see Kam, Zhu and Kalata (1997) and Luo and Kay (1989) for extensive surveys. As a robot's tasks become more complex and more numerous, the number of relevant features of the environment quickly exceeds the sensing and processing resources that are feasible to supply to a robot, Simmons (1990). Hence, the control of sensory perception is an important step towards designing autonomous mobile robots in complex and uncertain environments.

Leonard and Durrant-Whyte (1991) summarised the general problem of mobile robot navigation by three questions: "Where am I?", "Where am I going?" and "How should I get there?". The discrete event model provides an elegant framework for addressing these questions. The first question, which is the focus of this chapter, is addressed by the real-time identification of discrete events. The other two questions are addressed by discrete event trajectory planning, see for example McCarragher and Asada (1995b).

The discrete events become a natural common representational format for the sensors. A host of sensors such as tactile sensors, three-dimensional range sensors, force-torque sensors and acoustic sensors are currently being used in, for example, robotics and manufacturing applications and are well suited to discrete event identification. Hence, the sensory perception controller presented in this chapter is not limited to mobile robot navigation problems. In particular, manufacturing, robotic assembly, communication networks, transportation systems and logistic systems all fall within the class of discrete event systems and have to deal with real-time discrete event identification.

A new and unique discrete event model of mobile navigation in structured environments is presented. When the mobile unit is in contact with a surface, the free space motion becomes constrained. The discrete events are defined as changes in the mobile unit constraint equations. The advantage of such a model is a complete independency of given tasks. The successful completion of any given task becomes issues for the discrete event controller and the event trajectory planner. Hence, the proposed modelling of discrete events can be used in a wide range of applications.

The idea of using the discrete event framework in mobile navigation is not new. Košecká and Bogoni (1994), Košecká and Bajcsy (1994) and Bogoni and Bajcsy (1994) presented several discrete event solutions for navigation, obstacle avoidance, piercing and picking. The work presented demonstrates the discrete event framework's ability to describe complex interactions between different components in a systematic fashion. However, the discrete event models presented are highly task dependent. Examples of events are *obstacle detected* and *avoid obstacle*. Several problems such as navigation and obstacle avoidance are presented, but a new discrete event model is required for each different task.

Moreover, the sensing aspects are often neglected in discrete event systems research. The detection and identification of discrete events are assumed to be perfect. In this chapter we present a new and unique approach to on-line discrete event identification in mobile navigation by using an active sensing architecture.

6.2 Literature Survey

The area of *active mapping* has similarities to the sensory perception control problem addressed in this thesis. Typical sensors used in indoor environments do not provide perfect information about the environment. The information obtained from the sensors is imprecise, often ambiguous and may also be incorrect. For any dynamic map-building algorithm, correct feature identification can only be achieved through accumulation of observations over time. One example of active mapping was presented by Maksarov and Durrant-Whyte, (1995). An extended Kalman filter was used for sensor-based map building and navigation of an autonomous mobile vehicle. A number of different hypotheses about the underlying structure of the robot's environment are maintained and the hypotheses are gradually re-fined through re-observation. The geometric features are treated as targets and the mobile vehicle as an observer. The problem of map building and maintenance is considered as a problem of multiple target tracking.

Another example of active mapping was presented by Elfes (1991). The occupancy grid framework was used which uses estimation-theoretic and Markov random field models to compose information from multiple sensors and multiple points of view. Given a sensor range reading and a stochastic sensor model, the recursive estimation of the occupancy grid is done using Bayes' theorem. The author demonstrated sequential sensor scans from multiple locations as the robot moves around in the environment. Compared to fixed sensing locations, the uncertainty decay of the occupancy grid is faster using moving sensors.

Arkin and MacKenzie (1994) presented a framework for integrating multiple perceptual algorithms within a reactive robotic control system. A model using finite state receptors was developed as a means of expressing sequential perceptual processing over space and time in the context of a particular motor behaviour. Examples were presented in the context of mobile robot docking.

In this chapter we demonstrate Hidden Markov Models (HMMs) and a multilayer perceptron for the location problem in mobile navigation. The application of these tools to mobile navigation is not new. Zhu (1991) used a HMM to describe and predict the motions of obstacles in a dynamic environment. Initially an image of the scene is acquired

by a visual sensing device. This image is processed and any potential obstacles in the environment are detected. The HMM is then used to make predictions of the obstacle motions and a collision-free path is searched for and finally manoeuvre commands are generated. As the mobile robot proceeds, this cycle repeats.

One example of perception using a multilayer perceptron for mobile robot guidance is the extensive work by Pomerleau (1993). The multilayer perceptron architecture is similar to the architecture used by the MLP process monitor described in Appendix C. The input layer receives input from the vehicle's video camera. Each input unit is fully connected to a layer of 5 hidden units which are in turn fully connected to a layer of 30 output units. The output layer is a linear representation of the direction the vehicle should travel in order to keep the vehicle on the road. One application of the network is a driver monitoring and fatigue warning system.

6.3 Discrete Event Model of Navigation Problem

The problem of mobile navigation is immense. Different strategies are required for static, dynamic, partially known and unknown environments. We focus our attention to the localisation problem in a planar static environment as shown in Figure 6.1. The proposed method handles noisy and imperfect sensing, but requires all surfaces in the world to be modelled. For simplicity, all surfaces in the model in Figure 6.1 are straight lines. More complex surfaces, for example circles and semi-circles, can easily be incorporated into the model. However, such surfaces complicate the event identification algorithms.

We have chosen an environment consisting of three rooms, several walls and doors. The doors can be open or closed. There is no direct method for sensing whether a door is open or closed. This scenario allows us to demonstrate the ability of the SPC to recover from undetected events.

Each wall and door is modelled as a straight line. The mobile unit is modelled as a point (p_x, p_y) with a direction θ , see Figure 6.2. The point (p_x, p_y) of the mobile unit can be in any of the free spaces or contact any of the walls and doors. The equation of

Figure 6.1: *Contact states for the navigation problem. The dotted lines describe doors. Both sides of the doors are modelled.*

motion of the mobile unit in free-space is generally written as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (6.1)$$

where $\mathbf{x}(t)$ is the continuous time space vector, and $\mathbf{u}(t)$ is the input vector. As the mobile unit interacts with an inertially fixed environment, the free-space dynamics of equation (6.1) become constrained. For each point-surface contact, this constraint may be written as

$$g_j(\mathbf{x}(t)) = 0 \quad (6.2)$$

where g_j is the constraint equation for the j th point-surface contact. The distance function from the point to the surface is an ideal candidate for g_j . The distance functions can be constructed such that all $g_j(\mathbf{x})$ are positive inside the room and negative outside. Then, free-space motion in a room is constrained by the walls, ie.

$$\begin{aligned} i &= \operatorname{argmin}_j |g_j(\mathbf{x}(t))| \quad \text{for all } j \in \mathcal{J} \\ g_i(\mathbf{x}(t)) &\geq 0 \end{aligned} \quad (6.3)$$

Figure 6.2: Model of mobile unit consisting of a planar position and direction.

where $\mathcal{J}$ is the set of all constraints corresponding to a surface in the room. Equation (6.3) simply says that the mobile unit has to be on the room side of the closest surface. Since the mobile unit is modelled as a single point, only one constraint is active except for the corner points. Point-point contacts are not modelled. For many applications, point-point contacts occurring at corner points are short-lived and can be extremely difficult to detect and identify. A *discrete state* is defined as follows.

Definition 8 (Discrete State) A discrete contact state γ_i is given by a unique constraint equation. For a point-surface contact (including walls and doors) the constraint is given by equation (6.2). For a no contact state (rooms) the free-space motion is given by equation (6.3). $\square$

The contact state network is shown in Figure 6.3. Note that the individual rooms are only connected through the door states. Hence, the addition of more rooms to the model is trivial. Also note that both sides of the doors are modelled as discrete states. The constraints for the two sides of a door are different caused by the requirement of $g_j(\mathbf{x}) \geq 0$ on the room side of a surface.

A *discrete event* is now defined as follows.

Definition 9 (Discrete Event) The mobile unit can be in any free space state or contact doors and walls. A discrete event is defined as a physical change of state $\gamma_i \rightarrow \gamma_j$ between the mobile unit and the planar environment. $\square$

Figure 6.3: *Contact state network. The non-directional lines indicate bi-directional transitions between states. The directional lines indicate a transition from a door state to free space.*

6.4 Process Monitoring Techniques

The process monitors used by the sensory perception controller are described in the appendices. Appendix A describes the Hidden Markov Models (HMMs) process monitor, which uses the planar frequency-domain force/torque measurements for event recognition. Appendix B describes the process monitor based on position measurements and distance functions. Appendix C describes a multilayer perceptron used as an event classifier and finally Appendix D describes a qualitative reasoning process monitor. The algorithm described in section 5.5.2 is used to detect when discrete events occur. When a discrete event has occurred, each of the four process monitors is able to classify the event in real-time and produce an associated confidence level.

The monitor based on the distance functions requires a geometric model of the naviga-

tion task. A surface in Figure 6.1 is given by the position vectors of the two endpoints. For example, the two endpoints of the surface γ_4 are denoted $\mathbf{p}_{\gamma_4,1}$ and $\mathbf{p}_{\gamma_4,2}$ and are shown in Figure 6.4.

Figure 6.4: Endpoints of surface given by γ_4 .

For each room i we have

$$\begin{aligned} \mathbf{p}_{\gamma_{k_0^i},1} &= \mathbf{p}_i \\ \mathbf{p}_{\gamma_k,1} &= \mathbf{p}_{\gamma_{k-1},2} \\ \mathbf{p}_{\gamma_k,2} &= \mathbf{p}_{\gamma_k,1} + \mathbf{l}(\gamma_k) \end{aligned} \quad (6.4)$$

where $\mathbf{p}_i$ is the initial reference point for room i . We have $\mathbf{p}_1 = (0, 0)^T$, $\mathbf{p}_2 = \mathbf{p}_{\gamma_{16},1}$ and $\mathbf{p}_3 = \mathbf{p}_{\gamma_{28},1}$. The constants k_0^i are the surface numbers corresponding to $\mathbf{p}_i$, ie. $k_0^1 = 4$, $k_0^2 = 18$ and $k_0^3 = 38$. The vectors $\mathbf{l}(\gamma_k)$ are given by Table 6.1.

$\mathbf{l}(\gamma_4) = (l_4, 0)^T$	$\mathbf{l}(\gamma_{17}) = (0, l_{17})^T$	$\mathbf{l}(\gamma_{30}) = (0, l_{30})^T$
$\mathbf{l}(\gamma_5) = (0, -l_5)^T$	$\mathbf{l}(\gamma_{18}) = (0, l_{18})^T$	$\mathbf{l}(\gamma_{31}) = (-l_{31}, 0)^T$
$\mathbf{l}(\gamma_6) = (-l_6, 0)^T$	$\mathbf{l}(\gamma_{19}) = (l_{19}, 0)^T$	$\mathbf{l}(\gamma_{32}) = (0, -l_{32})^T$
$\mathbf{l}(\gamma_7) = (-l_7, 0)^T$	$\mathbf{l}(\gamma_{20}) = (0, -l_{20})^T$	$\mathbf{l}(\gamma_{33}) = (l_{33}, 0)^T$
$\mathbf{l}(\gamma_8) = (0, -l_8)^T$	$\mathbf{l}(\gamma_{21}) = (l_{21}, 0)^T$	$\mathbf{l}(\gamma_{34}) = (0, -l_{34})^T$
$\mathbf{l}(\gamma_9) = (-l_9, 0)^T$	$\mathbf{l}(\gamma_{22}) = (0, l_{22})^T$	$\mathbf{l}(\gamma_{35}) = (-l_{35}, 0)^T$
$\mathbf{l}(\gamma_{10}) = (0, l_{10})^T$	$\mathbf{l}(\gamma_{23}) = (l_{23}, 0)^T$	$\mathbf{l}(\gamma_{36}) = (l_{36x}, l_{36y})^T$
$\mathbf{l}(\gamma_{11}) = (-l_{11}, 0)^T$	$\mathbf{l}(\gamma_{24}) = (0, -l_{24})^T$	$\mathbf{l}(\gamma_{37}) = (0, l_{37})^T$
$\mathbf{l}(\gamma_{12}) = (0, -l_{12})^T$	$\mathbf{l}(\gamma_{25}) = (l_{25}, 0)^T$	$\mathbf{l}(\gamma_{38}) = (-l_{38}, 0)^T$
$\mathbf{l}(\gamma_{13}) = (-l_{13}, 0)^T$	$\mathbf{l}(\gamma_{26}) = (l_{26x}, l_{26y})^T$	$\mathbf{l}(\gamma_{39}) = (0, l_{39})^T$
$\mathbf{l}(\gamma_{14}) = (0, -l_{14})^T$	$\mathbf{l}(\gamma_{27}) = (l_{27}, 0)^T$	$\mathbf{l}(\gamma_{40}) = (l_{40}, 0)^T$
$\mathbf{l}(\gamma_{15}) = (0, -l_{15})^T$	$\mathbf{l}(\gamma_{28}) = (0, -l_{28})^T$	$\mathbf{l}(\gamma_{41}) = (0, -l_{41})^T$
$\mathbf{l}(\gamma_{16}) = (-l_{16}, 0)^T$	$\mathbf{l}(\gamma_{29}) = (-l_{29}, 0)^T$	

Table 6.1: Vector descriptions of each surface γ_k .

The distance functions are described in Appendix B. The distance functions are positive on the room side of the surface and negative otherwise. The distance between a surface $\psi \in \{\gamma_4, \dots, \gamma_{41}\}$ and the point ϕ , denoted by $h_{\psi\phi}$, is defined as the nearest distance between ϕ and coordinates on the surface ψ .

6.5 Experiments on Sensory Perception Control

The experimental setup is shown in Figure 6.5. A Polhemus position sensor and stylus were used to obtain the planar position measurements, while a JR3 force/torque sensor was used to measure the planar forces and the torque. The Polhemus and the JR3 sensors were connected to a Motorola 68040 based VxWorks board. All the process monitors and the sensory perception controller were implemented on the VxWorks board for real-time control. Figures 6.6 and 6.7 show examples of the measured positions and forces/torque for a typical discrete state trajectory. The discrete states are defined by Figure 6.1 and the measurement coordinate systems are defined by Figure 6.5.

Figure 6.5: *Experimental setup consisting of maze, Polhemus position sensor and JR3 force/torque sensor (located under the maze). The outer dimensions of the rectangular maze is 25.6cm by 25.6cm.*

The arrows at the bottom of Figure 6.7 show the occurrence of the discrete events. The goal of the sensory perception controller is to utilise the different process monitors to recognise discrete events as they occur while keeping the total sensing cost low.

The required CPU times and the average recognition rates of the monitors in real-time

Figure 6.6: *Example of planar position track showing the orientation of the stylus.*

Figure 6.7: *Planar force/torque measurements for a particular mobile navigation state trajectory. The arrows at the bottom of the figure show the occurrence of discrete events. Between the occurrence of the events, the current contact state is indicated.*

operation were found from a sample set of 300 discrete events and are given in Table 6.2. Note the relatively low recognition rates by monitor 3 and 4. These two monitors use the force/torque measurements only. Note from Figure 6.5 that the force sensor is located in the centre of the maze and that there are several parallel surfaces. This setup

Process Monitor	CPU Sec.	Rec. Rate %
1 – Distance Functions	0.11	72.8
2 – Multilayer Perceptron	0.61	76.0
3 – Qualitative Reasoning	0.09	66.0
4 – Hidden Markov Models	0.91	68.4

Table 6.2: *Required CPU times (CPU seconds) and average recognition rates of the process monitors in real-time operation.*

causes the measured planar forces and torque to be extremely similar for several of the discrete events. Hence, process monitoring using force only is difficult. We added Gaussian noise $\mathcal{N}(0,1)$ [cm] to the position measurements to reduce the recognition rates of the position based monitors. Further, the maze was moved 2cm (8% of maze width) in the P_x direction to make the model in Figure 6.1 inaccurate. In the following sections we show that even with these limitations in the modelling and the sensors, the sensory perception controller is able to achieve high performance event monitoring.

6.5.1 Experiments with Closed Doors

The sensory perception controller was tested on a sample set of 300 discrete events; 100 discrete events occurring in each of the rooms defined by the free spaces γ_1 , γ_2 and γ_3 . As for the robotic assembly case study in Chapter 5, the Markov state space and the state rewards were chosen as described in Table 5.4. The monitor rewards are defined by the costs of the process monitors given by Table 6.2. Let k_i be the cost of monitor i . In the experiments we used the same monitor rewards as equation (5.5). The state reward factor was chosen $K = 2(\max_i k_i) = 1.82$. As for the robotic assembly case study, the reward for choosing the fastest monitor is smaller than the smallest non-zero state reward. Hence, the state rewards dominate the monitor rewards, which ensures that the decision process will terminate when a relatively large confidence level is reached.

Figure 6.8 shows the average costs and recognition rates of the three different decision fusion methods as a function of the Markov discount factor α . The best results are

Figure 6.8: Average costs and average recognition rates of the different fusion methods as a function of the discount factor α .

clearly obtained when using the consensus based method. The best results for all the fusion methods were obtained with $\alpha = 0.85$. With large discount factors α , future rewards are large. The consultation action a_2 dominates and the monitor costs increase. With an increased monitor usage the recognition rates also normally increase, depending on the effectiveness of the decision fusion methods. With small α values, the future rewards are small. The termination action a_1 dominates and the monitor usage is low. When few monitors are used, the overall recognition rate is naturally low.

Figure 6.9 shows the average costs and recognition rates as a function of the state reward factor K for the three fusion methods. The discount factor was fixed at $\alpha = 0.85$. When K is large, the state rewards dominate the monitor rewards. Hence, the monitor usage is relatively low. When K is small, the monitor rewards dominate and the average costs are large. Note that the recognition rates have their highest values when K is between 1.46 and 1.82. For large K values the average costs decrease and as a result the recognition rates decrease naturally. For small K values the monitor rewards dominate the state rewards and the average costs are large. However, the SPC will consult new monitors even if a large confidence level is reached. Hence, in this case the large costs do not result in large recognition rates.

Figure 6.9: Average costs and average recognition rates of the different fusion methods as a function of the state reward factor K .

To summarise, the Figures 6.10 and 6.11 show the average recognition rates and costs of 1) each individual process monitor, 2) all the monitors used all the time with consensus-based decision making and 3) the performance of the sensory perception controller using three types of decision fusion. When all the monitors are used for every event occurrence, the consensus-based decision making is performed as in equation (5.6). As expected, the highest recognition rate (98%) was achieved when using all the monitors for every event occurrence. The best SPC result (96.4%) was achieved with consensus-based decision making. However, the average costs of the SPC methods are approximately one third of the cost of using all the monitors. Note also that all the sensory perception methods have higher recognition rates than any individual process monitor.

6.5.2 Experiments with Open Doors

In Figures 6.6 and 6.7 the mobile unit passes through an open door $\gamma_1 \rightarrow \gamma_{15} \rightarrow \gamma_2$. The force event detection algorithm described in section 5.5.2 is not able to detect these events. In this section we show how to recover from undetected events and the influence of undetected events on the performance of the sensory perception controller.

Process monitoring in the discrete event framework only evaluates events in the set of

Figure 6.10: Average costs of the process monitors and the sensory perception controller. *DF* is the distance functions process monitor, *MLP* is the multilayer perceptron, *QR* is the qualitative template matching method and *HMM* is the Hidden Markov Model method. *ALL* is a consensus-based method on all the monitors, *SPC1* is the sensory perception method using the last monitor, *SPC2* is sensory perception based on maximum confidence and *SPC3* is sensory perception based on consensus.

Figure 6.11: Event recognition rates of the process monitors and the sensory perception controller.

admissible discrete events. To be able to recover from undetected events, the set of admissible events must take into consideration possible future contact states that are not directly connected to the current contact state in the contact state network, Figure 6.3. When passing through an open door, two events are undetected, for example $\gamma_1 \rightarrow \gamma_{15}$ followed by $\gamma_{15} \rightarrow \gamma_2$. Hence, a look-ahead of three events is required to correctly classify new events after passing through the door. Let $\Gamma_{ij} = 1$ if $\gamma_i \rightarrow \gamma_j$ is a physical possible event in the contact state network in Figure 6.3. Then, with a look-ahead of three events, the set of admissible discrete events in any contact state γ_i

is given as follows.

$$\mathcal{E}_{\gamma_i} = \{\gamma_i \rightarrow \gamma_j, \gamma_j \rightarrow \gamma_k, \gamma_k \rightarrow \gamma_l\} \text{ for all } j, k, l \text{ such that } \Gamma_{ij} = \Gamma_{jk} = \Gamma_{kl} = 1 \quad (6.5)$$

Process Monitor	CPU Sec.	Rec. Rate %
1 – Distance Functions	0.14	67.6
2 – Multilayer Perceptron	0.81	65.7
3 – Qualitative Reasoning	0.12	56.2
4 – Hidden Markov Models	1.16	60.1

Table 6.3: *Required CPU times (CPU seconds) and average recognition rates of the process monitors with a look-ahead of three events.*

The increased number of events in the set $\mathcal{E}_{\gamma_i}$ increases the costs and decreases the average recognition rates of the monitors as described in Table 6.3. One reason for the decrease in recognition rates of the force based monitors is the increased number of parallel surfaces to be considered. As mentioned earlier, the force/torque measurements are very similar for parallel surfaces. The reason for the lower recognition rate of the position based monitor is the fact that a larger number of events are possible for similar measurements. For example, the events $\gamma_1 \rightarrow \gamma_4$ and $\gamma_4 \rightarrow \gamma_1$ might have identical position measurements.

The sensory perception controller was tested on a sample set of 200 discrete events; 100 events occurring after passing through an open door and 100 events occurring with no doors involved. A look-ahead of three events was used by all four process monitors for all the events in the sample set.

Figures 6.12 and 6.13 show the average recognition rates and costs of 1) each individual process monitor, 2) all the monitors used for every event occurrence with consensus-based decision making and 3) the performance of the sensory perception controller using three types of decision fusion. The consensus-based decision making with all monitors is the same as in equation (5.6). Again, the highest recognition rate (96.2%) is achieved when using all the monitors for every event occurrence. However, the cost of using all the monitors all the time is more than double the cost of the slowest sensory perception controller. The highest recognition rate of the SPC (91.4%) was achieved

Figure 6.12: Average costs of the process monitors and the sensory perception controller with a look-ahead of three events.

Figure 6.13: Event recognition rates of the process monitors and the sensory perception controller with a look-ahead of three events.

with consensus-based fusion. Note the relatively low recognition rates of the SPC when using fusion based on the last monitor and maximum confidence, (73.8%) and (80.4%). These two methods are only based on the confidence levels from the monitors. Because of the increased number of events in the set of admissible events, the confidences are in general lower than for the experiments in section 6.5.1. Still, the recognition rates of the SPC are higher than any individual monitor for all three fusion methods. We have demonstrated how the SPC is able to recover from undetected events and achieve high recognition rates by increasing the set of admissible events.

For the particular contact state network in Figure 6.3 any contact state can be reached in maximum four events from any starting point. Even if the additional look-ahead required to cover the entire search space is only one, the increase in the size of the set of admissible events is large. For example, in γ_1 the set of admissible discrete events

is almost doubled when the look-ahead is increased from three to the entire search space. In general, a trade-off has to be made between sensing costs and the additional robustness gained by a large look-ahead horizon.

6.6 Experiments on Sensitivity Analysis

In this section we demonstrate the sensitivity analysis of the SPC used for the mobile navigation problem using the linear programming approach in Chapter 4. The highest performance was achieved by using consensus-based decision fusion method, see Figures 6.10 and 6.11. We demonstrate the sensitivity analysis on the discount factor α and the state reward constant K using consensus-based decision fusion, equation (3.19). The monitor costs are fixed and are given by the CPU times in Table 6.2.

6.6.1 Sensitivity of the Discount Factor α

Figure 6.14 and 6.15 show the sensitivity of the decisions made by the SPC as a function of the discount factor α . The method in Table 4.1 was used to find the optimal LP solution $\mathbf{x}^*$ for all the permutations and equations (4.8) and (4.9) were used to find the optimal permutation z^* . All the four process monitors are available to the SPC, and the figures show which process monitor is consulted first when the initial states are $\sigma(0) = s_1$ and $\sigma(0) = s_2$. In section 6.5 high recognition rates and low sensing costs were achieved for $\alpha = 0.85$. The sensitivity on the first optimal decision made by the SPC is low for $\alpha = 0.85$, as seen from Figures 6.14 and 6.15.

Note from the figures that monitor 1 or 3 is always consulted first by the SPC, which is caused by the low CPU costs of these two monitors, see Table 6.2. The SPC consults the fast monitors (distance functions and qualitative reasoning) first to gain the large monitor rewards. Then, depending on the confidence levels after the fast monitors have been consulted, the slower monitors may or may not be consulted. This behaviour also explains the low cost performance of the SPC in Figure 6.10 compared to using all the monitors for every event occurrence.

The SPC decisions in Figures 6.14 and 6.15 are explained as follows. When α is low,

Figure 6.14: *Monitor consulted first by the SPC as a function of the discount factor α when the initial SPC state is $\sigma(0) = s_1$.*

Figure 6.15: *Monitor consulted first by the SPC as a function of the discount factor α when the initial SPC state is $\sigma(0) = s_2$.*

future rewards are discounted by a large value and, as a result, the immediate rewards dominate. When the immediate rewards dominate, the optimal sequential order of monitors for both initial states $\sigma(0) = s_1$ and $\sigma(0) = s_2$ is $z^* = 1 - 2 - 3 - 4$. Then, as α increases, monitor 3 is moved forwards in the optimal sequence. For large values of α , the optimal sequence is $z^* = 3 - 1 - 2 - 4$. Monitor 3 is consulted first to pick up the large monitor reward. Since future rewards are not discounted by a large value, the dynamic programming allows the remaining monitors, (1, 2, 4), to increase the confidence levels. As the initial state $\sigma(0)$ increases from s_1 to s_2 , we see from the figures that monitor 1 is the optimal first choice for larger values of α . Similarly, when $\sigma(0) = s_3$, monitor 1 is optimal for $0 \leq \alpha \leq 0.67$. Monitor 3 is fast, but has relatively low confidence levels. When $\sigma(0)$ increases, consulting monitor 3 may cause a reduction of the overall confidence. Hence, monitor 3 is less attractive for large initial states $\sigma(0)$.

Note that the Figures 6.14 and 6.15 are very similar. For larger values of $\sigma(0)$ the α -plots have the same structure, but the change of optimal first choice from monitor 1 to 3 will occur for larger values of α . The Markovian decision state $\sigma(0)$ is determined from the real-time confidence level measures. Hence, the sensitivity of the SPC decisions to noisy confidence measurements is relatively low.

6.6.2 Sensitivity of the State Reward Constant K

The algorithm in Table 4.1 is used for the sensitivity analysis of the state reward constant K . The discount factor value is chosen $\alpha = 0.85$. The method in Table 4.1 is used to find the optimal basic feasible LP solution $\mathbf{x}^*$ and the optimal permutation z^* .

All the four process monitors are initially enabled. When the state reward constant takes continuous values $K \in [0.5, 6]$, the optimal first decision is *always* to consult monitor 3 for the initial states $\sigma(0) = s_1$, $\sigma(0) = s_2$ and $\sigma(0) = s_3$. For the other states $s_4, \dots, s_8$, the optimal decision is terminate. Hence, the choice of K has no influence on the first SPC decision when $\alpha = 0.85$ and the other model parameters are chosen as described earlier in this chapter.

For all the 24 permutations the linear program was solved on average 8.1 times (step 2 in Table 4.1) when $K \in [0.5, 6]$. An example of stored K values are given in Table 6.4 for one permutation. The example shows all the K values for which the LP is solved and the corresponding ϵ values (step 3 in Table 4.1). The stored K values and the

K	6.00	5.53	2.57	2.02	0.85	0.84	0.82	0.51
ϵ	0.92	0.46	0.79	0.42	0.99	0.98	0.58	0.00

Table 6.4: *Example of stored K values for permutation $z = 1 - 2 - 3 - 4$.*

corresponding $\mathbf{x}^*(K)$ and $c_B(K)$ (step 4 in Table 4.1) describe the complete sensitivity analysis for any value of K . The algorithm in Table 4.1 is based on the optimality criterion in equation (4.7). Hence, the results in Table 6.4 show, for example, that the optimal actions for $K = 5.53$ and $z = 1 - 2 - 3 - 4$ are also optimal in the *entire* continuous range $K \in [2.57, 5.53]$.

Once the algorithm in Table 4.1 is completed for all permutations, no further calculations are required to solve the Markovian decision problem for *any* value of $K \in [0.5, 6]$. The stochastic dynamic programming approach, on the other hand, has to be solved from scratch for each new value of K . For example, if the discretisation step of K is chosen to be 0.01, a brute-force sensitivity analysis of $K \in [0.5, 6]$ requires the SDP to be solved 550 times for each permutation. Puterman (1994, p. 230) estimated the

SDP algorithm to be 12 times faster than the LP algorithm. For this example, the LP approach to sensitivity analysis of the parameter K is $\frac{600}{12 \times 8.1} = 6.2$ times faster than a brute-force SDP approach.

6.7 Conclusions

We have presented a new approach for controlling sensory perception in mobile navigation. The navigation problem is modelled as a discrete event system. The cost-efficient use of sensory perception reduces the need for mobile robots to carry powerful computational power. The cost of sensing is reduced compared to multi-sensor systems where all the sensors are used for every event occurrence. The experiments also demonstrate that the sensory perception controller achieves higher event recognition rates than any individual sensor.

The sensitivity analysis developed in Chapter 4 was applied to the mobile navigation problem. In the experiments it was found that the LP approach to sensitivity analysis was up to 6 times faster than a brute-force SDP approach. The experiments show that the decision sensitivity to changes in the SPC parameters α and K is relatively low.

The experiments in this chapter demonstrate an important feature of the discrete event formalism. By increasing the look-ahead horizon of the event monitors, the perceptual system is able to detect and recover from errors in the event trajectories. In the experiments the event of passing through an open door is undetected. When the event monitors use a look-ahead of three events, the SPC is able to identify the correct event after passing through the open door. However, the sensing costs are larger when undetected events are considered. In general, a trade-off has to be made between sensing costs and the additional robustness gained by a large look-ahead horizon.

Chapter 7

Conclusions and Further Research

7.1 Introduction

This chapter reviews the major results and contributions of the work presented in the thesis. First, the original aspects of the problem formulation are summarised. Second, the contributions of the two solution methods, SDP and LP, are described. Third, the application of the SPC to the two case studies, robotic assembly and mobile navigation, are reviewed.

The work presented in the thesis motivates and raises several topics for further research, extensions and improvement of the sensory perception controller, the discrete event formalism and the sequential decision theory. The topics discussed in this chapter are:

- 1) Dealing with imperfect event recognition.
- 2) The extension of the methods (both individual monitors and the SPC) to more complex tasks, such as 3D robotic assembly or navigation.
- 3) The extension of the SPC to continuous feedback control systems.
- 4) A further theoretical analysis of the SPC.
- 5) The incorporation of both parallel and sequential decision making.

7.2 Major Conclusions and Results

The main contributions of the thesis are divided into four areas. The first contribution is the formulation of the problem as a sequential Markov decision process in the discrete event framework. Markov decision process theory has two main advantages compared to other sequential decision theories. 1) The optimal actions at any decision epoch are independent of the actions at previous decision epochs. This property allows for efficient decision making algorithms. 2) The theory provides facilities for sensitivity analysis. The discrete event formalism also offers several advantages. 1) Because of the discrete nature of events, feedback information is not required continuously. Hence, valuable processing time is available between the occurrence of events. 2) at the occurrence of discrete events the dynamics of the sensing signals used by the event recognisers are often strong and contain a large amount of information. 3) the discrete events are a natural common representational format for the sensors. A common sensor format aids the decision process when dealing with different sensor types.

The second main contribution of the thesis is the development of a fast real-time SPC. The proposed technique solves the Markov decision process using stochastic dynamic programming. The dynamic programming is performed off-line and the results are stored in a lookup table. The lookup table is used by the real-time sensory perception controller (SPC) for sequentially selecting one or more process monitors for successful discrete event recognition. To be of any practical value, the extra incurred cost by the SPC can not outweigh the actual sensing costs. Otherwise, a more cost-efficient solution would be to use all available sensors and multi-sensor fusion techniques. Stochastic dynamic programming provides a computationally efficient solution to the control of sensory perception.

The third main contribution is the development of a method for the sensitivity analysis of the SPC. The analysis aids in the robust tuning of the SPC by finding low sensitivity areas for the model parameters. The analysis is performed by exploring the relationship between discounted Markovian decision theory and linear programming. The sensitivity analysis is a fundamental part of the solution methodology by gaining insight into both the off-line sensor planning and the on-line behaviour during execution.

The fourth main contribution is the application of the SPC to two practical control systems. The first application consists of a planar assembly task modelled as a constrained motion system. Four different event monitors using force/torque and position measurements have been developed. The experimental results presented show that the SPC achieves higher event recognition rates than any individual monitor with a sensing cost of less than half the cost of using all the monitors for every event occurrence. The second application of the SPC has been demonstrated for a mobile navigation problem. As a robot's tasks become more complex and more numerous, the number of relevant features of the environment quickly exceeds the sensing and processing resources that are feasible to supply to a robot. The SPC reduces the overall sensing costs and, as a result, reduces the required processing resources for mobile robots. A new discrete event model of mobile navigation is presented and the experiments demonstrate high performance event recognition.

7.3 Further Research

7.3.1 Imperfect Event Recognition

The experiments in Chapter 6 demonstrated a look-ahead horizon of several events. The approach allows for recovery from undetected and incorrectly classified events. However, the larger sets of admissible discrete events reduce the individual monitoring recognition rates and increase the sensing costs. A different approach is to keep track of several event trajectories. Each event trajectory would have an associated confidence level. Whenever a new discrete event occurs, the confidence levels of several physical possible event trajectories are updated based on the measured confidence levels from the consulted event monitors. Hence, when an event is incorrectly classified, the correct event trajectory would still be evaluated for future event occurrences. For example, consider the event trajectories in Figure 7.1. Now consider the following scenario. Event e_1 occurs but the monitoring system *incorrectly* classifies e_2 as the correct event. The two events are assigned confidence levels $C(e_1) = 0.4$ and $C(e_2) = 0.6$. Next, event e_3 occurs and the monitoring system assigns the confidences $C(e_3) = 0.5$, $C(e_4) = 0.3$ and $C(e_5) = 0.2$. By multiplying the confidence levels to find the trajectory confidences,

Figure 7.1: *Keeping track of several event trajectories.*

we get $C(e_1 \rightarrow e_3) = 0.4 * 0.5 = 0.20$, $C(e_2 \rightarrow e_4) = 0.6 * 0.3 = 0.18$ and $C(e_2 \rightarrow e_5) = 0.6 * 0.2 = 0.12$. Hence, the event trajectory $e_1 \rightarrow e_3$ has the highest confidence and is correctly classified despite the fact that e_1 was initially classified incorrectly. This method allows for error detection and recovery in the event trajectories. The increased robustness, however, increases the sensing costs incurred by the consideration of larger sets of admissible events.

Another approach is the incorporation of sensing considerations in the discrete event trajectory planning. Discrete events which are difficult to monitor should be avoided in the desired event trajectories. The real-time or apriori expected confidence levels from the process monitors are natural variables to consider in both real-time or off-line discrete event trajectory planners.

7.3.2 Extension to More Complex Tasks

As the complexity of the tasks increases, more process monitors must be added to the sensory perception architecture for successful event recognition. The number of permutations of monitors considered by the SPC is equal to $N_m!$, where N_m is the number of available monitors. When the number of monitors becomes large (> 10), the memory requirements quickly exceed the memory provided by real-time systems. Hence, the SPC can only achieve sub-optimality by considering a sub-set of all possible permutations. Pang and Chen (1976), Pang, Sheble and Albuyeh (1981) presented priority lists for finding sub-optimal sequential decisions for the unit commitment problem. By

using priority lists for the sensory perception problem, a natural choice is to let the low cost monitors always be consulted before the high cost monitors. Restricting the sequences in this fashion, reduces the number of permutations to consider and allows for the inclusion of a large number of monitors. Pang, Sheble and Albuyeh (1981) successfully demonstrated the dynamic programming approach for the sub-optimal sequential selection of 96 units.

For more complex tasks the number of discrete events to be considered by the process monitors will naturally increase. However, one property of the discrete event formalism prevents a dramatic increase of complexity for the monitoring problem; the sets of admissible events are usually small even for complex tasks. McCarragher (1996) studied the discrete event control of a 3D assembly task. Naturally, the number of contact states and discrete events becomes large. However, given any contact state, only a limited number of discrete events are physically possible. An assembly task with a large number of contact states usually has a small number of admissible events for each contact state. Since the process monitors only consider the admissible events, high performance classification is still possible.

7.3.3 Continuous Control of Sensory Perception

The SPC presented in this thesis is formulated in the discrete event framework, which has been successfully applied to a wide range of autonomous control systems. However, the majority of modern control problems are naturally formulated as continuous feedback systems. Some SPC solutions exist for continuous feedback systems, for example Oshman (1994) and Kulkarni and Ramadge (1996). However, these SPC solutions are computationally expensive and often outweigh the sensor cost savings. Hence, for continuous control systems, there is a need for further research in developing fast-real time algorithms. We believe the off-line SDP algorithm utilising real-time measurements for controlling sensory perception is an attractive approach in continuous feedback control systems as well as discrete event systems.

7.3.4 Analytical Research

In this thesis a new approach to event monitoring has been presented. Chapters 3 and 4 present a real-time controller algorithm and a sensitivity analysis based on stochastic dynamic programming and linear programming, respectively. The research was application driven and few simplifying assumptions were made. However, as a result, some of the results in the thesis are only empirical. For example, the Figures 6.8 to 6.13 were found from experimentation. A challenging research direction is a further theoretical analysis to gain a deeper insight into the behaviour of the SPC, such as recognition rates and sensing costs as functions of the SPC parameters. For example, if certain probability distributions of the recognised events and confidence levels are assumed, the SPC problem is made application independent and analytical results are possible. However, the theoretical analysis of stochastic programming solutions, such as SDP, are recognised as difficult research problems, Fiacco (1983).

7.3.5 Parallel Decision Making

The objectives behind the control of sensory perception is to reduce computational requirements such as multiple processors while at the same time keeping the event recognition rates high. However, for certain applications the extra costs incurred by parallel computing may be acceptable. The added option of parallel consultation of event monitors opens up new and challenging research problems. For example, a system consisting of two parallel processors and more than two event monitors must take both parallel and sequential decision making into account. In a system with two processor boards and three event monitors, the initial decision will be to consult a single monitor 1, 2, 3 or two monitors in parallel, 1 – 2, 1 – 3 or 2 – 3. For such systems, careful considerations are required to prevent the combinatorial explosion of the decision models.

References

- Aidarous, S.E., M.R. Gevers and M.J. Installé (1975), "Optimal Sensors' Allocation Strategies for a Class of Stochastic Distributed Systems", *International Journal of Control*, Vol. 22, No. 2, pp. 197-213.
- Aloimonos, Y. (Ed.) (1993), *Active Perception*, Lawrence Erlbaum Associates, Inc., New Jersey.
- Arkin, R.C. and D. MacKenzie (1994), "Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation", *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 3, pp. 276-286.
- Astuti, P. (1995), *The Convergence and Control of a Class of Hybrid Dynamic Systems*, PhD Thesis, The Australian National University, Department of Engineering.
- Baccelli, F., G. Cohen, G.J. Olsder and J.P. Quadrat (1992), *Synchronization and Linearity: An Algebra for Discrete Event Systems*, John Wiley & Sons, Ltd.
- Badano, F., M. Betemps, T. Redarce and A. Jutard (1991), "Robotic Assembly by Slight Random Movements", *Robotica*, Vol. 9, pp. 23-29.
- Bajcsy, R. (1988), "Active Perception", *Proceedings of the IEEE*, Vol. 76, No. 8, pp. 996-1005.
- Bellmann, R. (1957), *Dynamic Programming*, Princeton University Press, Princeton, New Jersey.
- Bogoni, L. and R. Bajcsy (1994), "Functionality Investigation Using a Discrete Event System Approach", *Robotics and Autonomous Systems*, Vol. 13, No. 3, pp. 173-196.
- Brown, M.L. and D.E. Whitney (1994), "Stochastic Dynamic Programming Applied to Planning of Robot Grinding Tasks," *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 5, pp. 594-604.
- Cassandras, C.G. (1993), *Discrete Event Systems: Modeling and Performance Analysis*, Richard D. Irwin, Inc., and Aksen Associates, Inc.
- Cassandras, C.G., S. Lafortune and G.J. Olsder (1995), "Introduction to the Modelling, Control and Optimization of Discrete Event Systems", *Trends in Control:*

- A European Perspective*, Springer Verlag, pp. 217-291.
- Chiacchio, P., S. Chiaverini, L. Sciavicco and B. Siciliano (1991), "Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy", *The International Journal of Robotics Research*, Vol. 10, No. 4, pp. 410-425.
- Chrisman, L. and R. Simmons (1991), "Sensible Planning: Focusing Perceptual Attention", *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI)*, AAAI Press, Menlo Park, Cal., pp. 756-761.
- Chrisman, L. (1992), "Planning for Closed-Loop Execution Using Partially Observable Markovian Decision Processes", *In AAAI Spring Symposium Series: Control of Selective Perception*, Stanford University.
- Chvátal, V. (1983), *Linear Programming*, W.H. Freeman and Company, New York.
- Dasarathy, B.V. (1994), *Decision Fusion*, IEEE Computer Society Press.
- Dean, T., K. Basye and M. Lejter (1991), "Planning and Active Perception", *Autonomous Mobile Robots: Control, Planning, and Architecture*, IEEE Computer Society Press, Los Alamitos, CA, pp. 197-202.
- Dean, T., T. Camus and J. Kirman (1990), "Sequential Decision Making for Active Perception", *Proceedings of the DARPA Image Understanding Workshop*, pp. 889-894.
- Dutr, S., H. Bruyninckx and J. De Schutter (1996), "Contact Identification and Monitoring Based on Energy", *Proceedings of the 1996 International Conference on Robotics and Automation*, Minneapolis, Minnesota, pp. 1333-1338.
- Dutr, S., H. Bruyninckx, S. Demey and J. De Schutter (1996), "Solving Redundant Contact and Grasp Uncertainties", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, 8-13 September 1997.
- Eberman, B. and J.K. Salisbury (1994), "Application of Change Detection to Dynamic Contact Sensing", *The International Journal of Robotics Research*, Vol. 13, No. 5, pp. 369-394.
- Elfes, A. (1991), "Dynamic Control of Robot Perception Using Stochastic Spatial Models", *Autonomous Mobile Robots: Control, Planning, and Architecture*, IEEE Computer Society Press, Los Alamitos, CA, pp. 203-218.
- Fiacco, A.V. (1983), *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Mathematics in Science and Engineering, Vol. 165, Academic Press, New York.
- Groca, H.B. and A. de Sam Lazaro (1995), "Fuzzy Logic and Position Sensing for Precision Assembly", *Journal of Robotic Systems*, Vol. 12, No. 2, pp. 135-146.

- Hager, G.D. (1990) *Task-Directed Sensor Fusion and Planning: A Computational Approach*, Kluwer Academic Publishers.
- Hannaford, B. and P. Lee (1991), "Hidden Markov Model Analysis of Force/Torque Information in Telem Manipulation", *The International Journal of Robotics Research*, Vol. 10, No. 5, pp. 528-539.
- Hirai, S. (1989), "Analysis and Planning of Manipulation Using the Theory of Polyhedral Convex Cones", PhD Thesis, Kyoto University, Department of Applied Mathematics and Physics.
- Hovland, G.E. and B.J. McCarragher (1996), "Frequency-Domain Force Measurements for Discrete Event Contact Recognition", *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, 22-28 April, pp. 1166-1171.
- Hovland, G.E. and B.J. McCarragher (1997a), Combining Force and Position Measurements for the Monitoring of Robotic Assembly, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, 8-13 September 1997.
- Hovland, G.E. and B.J. McCarragher (1997b), "Dynamic Sensor Selection for Robotic Systems", *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, 20-25 April, pp. 272-277.
- Hovland, G.E. and B.J. McCarragher (1988), "Hidden Markov Models as a Process Monitor in Robotic Assembly," *The International Journal of Robotics Research*, Vol.17, No.1, To Appear.
- Huang, X.D., Y. Ariki, M.A. Jack (1990), *Hidden Markov Models for Speech Recognition*. Edinburgh, Edinburgh University Press.
- Hutchinson, S.A. and A.C. Kak (1989), "Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities", *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 6, pp. 765-783.
- Johansson, R.S. (1978), "Tactile Sensibility in the Human Hand: Receptive Field Characteristics of Mechanoreceptive Units in the Glabrous Skin Area", *Journal of Physiology*, Vol. 281, pp. 101-123.
- Kallenberg, L.C.M. (1983), *Linear Programming and Finite Markovian Control Problems*, Mathematisch Centrum, Tract No. 148, Amsterdam.
- Kam, M., X. Zhu and P. Kalata (1997), "Sensor Fusion for Mobile Robot Navigation", *The Proceedings of the IEEE*, Vol. 85, No. 1, pp. 108-119.
- Katupitiya, J., S. Dutré, S. Demey, J. De Geeter, H. Bruyninckx and J. De Schutter (1996), "Estimating Contact and Grasping Uncertainties Using Kalman Filters

- in Force Controlled Assembly", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'96)*, Osaka, Japan, pp. 696-703.
- Košecká, J. and L. Bogoni (1994), "Application of Discrete Event Systems for Modeling and Controlling Robotic Agents", *Proceedings of the International Conference on Robotics and Automation*, San Diego, pp. 2557-2562.
- Košecká, J. and R. Bajcsy (1994), "Discrete Event Systems for Autonomous Mobile Agents", *Robotics and Autonomous Systems*, Vol. 12, No. 3/4, pp. 187-198.
- Kulkarni, S.R. and P.J. Ramadge (1996), "Model and Controller Selection Policies Based on Output Prediction Errors," *IEEE Transactions on Automatic Control*, Vol. 41, No. 11, pp. 1594-1617.
- Kumar, R. and V.K. Garg (1995), "Optimal Supervisory Control of Discrete Event Dynamical Systems", *SIAM Journal on Control and Optimization*, Vol. 33, No. 2, pp. 419-439.
- Leonard, J. and H.F. Durrant-Whyte (1991), "Application of Multi-Target Tracking to Sonar-Based Mobile Robot Navigation", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 376-382.
- Lima, P. and G. Saridis (1994), "A Performance Measure for Intelligent Machines Based on Complexity and Reliability," *Symposium on Robot Control '94*, Sep. 19-21, Capri, Italy, pp. 885-890.
- Lippmann, R.P. (1987) *An Introduction to Computing with Neural Nets*. IEEE ASSP Magazine, Apr. 1987, pp. 4-22.
- Luo, R.C. and M.G. Kay (1989), "Multisensor Integration and Fusion in Intelligent Systems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, No. 5, pp. 901-931.
- Maksarov, D. and H. Durrant-Whyte (1995), "Mobile Vehicle Navigation in Unknown Environments: A Multiple Hypothesis Approach", *IEE Proceedings. Control Theory and Applications*, Vol. 142, No. 4, pp. 385-400.
- McCarragher, B.J. and H. Asada (1993), "Qualitative Template Matching Using Dynamic Process Models for State Transition Recognition of Robotic Assembly," *ASME Journal of Dynamic Systems, Measurements and Control*, vol. 115, no. 2A, pp. 261-269.
- McCarragher, B.J. and H. Asada (1995a), "The Discrete Event Control of Robotic Assembly Tasks", *ASME Journal of Dynamic Systems, Measurements and Control*, Vol. 117, No. 3, pp. 384-393.
- McCarragher, B.J. and H. Asada (1995b), "The Discrete Event Modelling and Trajectory Planning of Robotic Assembly Tasks", *ASME Journal of Dynamic*

- Systems, Measurements and Control*, Vol. 117, No. 3, pp. 394-400.
- McCarragher, B.J. (1996), "Task Primitives for the Discrete Event Modeling and Control of 6-DOF Assembly Tasks", *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 2, pp. 280-289.
- Meier, L., J. Peschon and R.M. Dressler (1967), "Optimal Control of Measurement Subsystems," *IEEE Transactions on Automatic Control*, vol. AC-12, no. 5, pp. 528-536.
- Ogasawara, G.H. (1991), "Control of Sensing and Planning Using Distributed Processing and Decision Analysis", In *AAAI Fall Symposium Series: Sensory Aspects of Robotic Intelligence*, Monterey, CA.
- Mine, H. and S. Osaki (1970), *Markovian Decision Processes*. American Elsevier Publishing Company, Inc.
- Oshman, Y. (1994), "Optimal Sensor Selection Strategy for Discrete-Time State Estimators," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 307-314.
- Özveren, C.M. and A.S. Willsky (1990), "Observability of Discrete Event Dynamic Systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 797-806.
- Pang, C.K. and H.C. Chen (1976), "Optimal Short-Term Thermal Unit Commitment," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-95, no. 4, pp. 1336-1346.
- Pang, C.K., G.B. Sheble and F. Albuyeh (1981), "Evaluation of Dynamic Programming Based Methods and Multiple Representation for Thermal Unit Commitments," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 3, pp. 1212-1218.
- Papastavrou, J.D. and M. Athans (1986), "A Distributed Hypothesis-Testing Team Decision Problem with Communication Cost", *Proceedings of 25th Conference on Decision and Control*, Athens, Greece, pp. 219-225.
- Picone, J. (1990), "Continuous Speech Recognition Using Hidden Markov Models", *IEEE ASSP Magazine*, Vol. 7, No. 3, pp. 26-41.
- Pomerlau, D.A. (1993), *Neural Network Perception for Mobile Robot Navigation*, Kluwer Academic Publishers.
- Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery (1992), *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press.
- Puterman, M.L. (1994), *Markov Decision Processes*. John Wiley & Sons, Inc.

- Rabiner, L.R. and B.H. Juang (1986), "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, Vol.3, No. 1, pp. 4-16.
- Ramadge, P.J. and W.M. Wonham (1989), "The Control of Discrete Event Systems", *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81-98.
- Rimey, R.D. and C.M. Brown (1994), "Control of Selective Perception Using Bayes Nets and Decision Theory", *International Journal of Computer Vision*, Vol 12:2/3, pp. 173-207.
- Ross, S.M. (1983), *An Introduction to Stochastic Dynamic Programming*. New York, Academic Press.
- Rothenberg, R.I. (1979), *Linear Programming*, Elsevier North-Holland, Inc., New York.
- Rubinstein, R.Y. and A. Shapiro (1993), *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, John Wiley & Sons, Ltd.
- Sakaguchi, Y. (1994), "Haptic Sensing System with Active Perception", *Advanced Robotics: The international journal of the Robotics Society of Japan*, Vol. 8, No. 3, pp. 263-283.
- Shin, K.G. and N.D. McKay (1986), "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," *IEEE Transactions on Automatic Control*, vol. AC-31, no. 6, pp. 491-500.
- Shirinzadeh, B. (1993), "Issues in the Design of the Reconfigurable Fixture Modules for Robotic Assembly", *Journal of Manufacturing Systems*, Vol. 12, No. 1, pp. 1-14.
- Simmons, R. (1990), "Robust Behaviour with Limited Resources", *AAAI Stanford Spring Symposium*.
- Solow, D. (1984), *Linear Programming: An Introduction to Finite Improvement Algorithms*. Elsevier Science Publishing Co., Inc.
- Tarabanis, K.A., P.K. Allen and R.Y. Tsai (1995), "A Survey of Sensor Planning in Computer Vision", *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 1, pp. 86-104.
- Thie, P.R. (1988), *An Introduction to Linear Programming and Game Theory*, John Wiley & Sons, Inc.
- Tijms, H.C. (1994), *Stochastic Models: An Algorithmic Approach*, John Wiley & Sons Ltd, Chichester, England.
- Trinkle, J.C. and D.C. Zeng (1995), "Prediction of the Quasistatic Planar Motion of a Contacted Rigid Body", *IEEE Transactions on Robotics and Automation*, Vol.

11, No. 2, pp. 229-246.

Trucco, E., M. Umasuthan, A.M. Wallace and V. Roberto (1997), "Model-Based Planning of Optimal Sensor Placements for Inspection", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, pp. 182-193.

White, D.J. (1993), *Markov Decision Processes*, John Wiley & Sons, Ltd., Chichester, England.

Zilberstein, S. (1996), "Resource-Bounded Sensing and Planning in Autonomous Systems", *Autonomous Robots*, Kluwer Academic Publishers, Vol. 3, No. 1, pp. 31-48.

Zhu, Q. (1991), "Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 390-397.

Appendix A

Process Monitoring Using Hidden Markov Models

A.1 Introduction

The process monitor developed in this appendix is based on Hidden Markov Models (HMMs) and has been presented by Hovland and McCarragher (1998). An HMM is defined as a doubly stochastic process. The underlying stochastic process is *not* observable (it is hidden), but can be observed through another set of stochastic processes that produce the sequence of observed symbols, Rabiner and Juang (1986). Stochastic force/torque measurements are used to obtain information about the discrete events, which is illustrated in Figure A.1. Each state in the hidden process has a stochastic description of the force measurements. The observation symbols are compared to these stochastic descriptions (for example Gaussian distributions) to find the probability of each hidden state.

In addition to the force/torque measurements being stochastic, the underlying process must also be stochastic. In our work each hidden model represents a discrete event. The underlying stochastic process corresponds to the internal behaviour of the events. The internal behaviour is influenced by several factors. In a robotic assembly task, for example, the internal stochastic behaviour is caused by the position of the workpiece in the gripper, friction between the workpiece and the environment or tolerancing errors

Figure A.1: *Illustration of a HMM. The underlying stochastic process can only be observed through another stochastic process.*

in the geometries of the workpiece and the environment.

The stochastic behaviour of the underlying process is very important since the whole theory of Hidden Markov Models is based on the fact that both the underlying process and the observed process are stochastic. In the paper by Hannaford and Lee (1991) the underlying process is not the internal behaviour of the events, but rather the sequence of discrete events. In that case the assumption has to be made that the sequence of discrete events is stochastic. However, when the hidden process is the sequence of discrete events, the HMM probabilities will be highly dependent on the event trajectories chosen in the training set. In the method presented here, the HMMs are independent of the actual event trajectory and hence modifying the trajectory on-line to recover from errors does not influence the recognition rate of the HMMs.

There are many possible HMM models for the events. One type of model which has been successfully applied to speech recognition and also works well for dynamic systems using force measurements, is the absorbing parallel left-to-right model illustrated in Figure A.2. (See Figure 9 in Picone (1990) for other possible left-to-right models.) For this model an event always starts in the state q_1 and ends in q_6 or q_7 , but there are several possible paths through the states. The observed symbols proceed forward in time which is the main reason for using a left-to-right model. The left part of the

Figure A.2: HMM λ^k for each event. q_i are the states and a_{ij} are the hidden stochastic process state change probabilities.

model is associated with the early measurements while the right part of the model is associated with measurements at some time later. Since these measurements might be very different, a left-to-right model is ideal for extracting this information. Left-to-right models are a good tool for describing how signals evolve in time. This is an important property of the HMMs. With this property we are able to describe dynamic motions with dynamic force/torque measurements of the workpiece, as opposed to quasi-static motions.

A.2 Observation Symbols

One key element in successful process monitoring is the observation symbols. For the HMM monitor the frequency components of the force/torque signals are used as observation symbols. There are several reasons for using frequency components. First, when an event occurs in a dynamic system where there is a sudden change in the constraint equations, the frequency band is broad and occurs within a short time scale of the event, see Figure A.3. Therefore, a lot of information can be extracted by using frequency components as observation symbols. Second, by normalising the total energy in the measurements, we make the observation symbols independent of the signal magnitudes, ie. a gentle contact (event) will have similar observation symbols as a brutal contact. It is also worth noting that research on human tactile perception has shown that our skin force sensors are sensitive to force frequencies, Johansson (1978). By using frequency

Figure A.3: *Spectrogram of the F_y force measurements for two events. The first event starts at $t = 2.0\text{sec}$ followed by another event at $t = 4.2\text{sec}$. The measured force was sampled at 500 Hz. Notice that the frequency band is broad and occurs within a short time scale of the event.*

components the training will estimate the mean and the covariance of the energy in different frequency components of the force/torque measurements. Hannaford and Lee (1991) use the mean and covariance of the force magnitude measurements directly in a robotic system where a HMM is used as a process monitor.

For the HMM monitor presented here the continuous observation symbols are represented by the continuous vector $\mathbf{x}_t$. The observation probability $b_j(\mathbf{x}_t)$ is then given by

$$b_j(\mathbf{x}_t) = \mathcal{N}[\mathbf{x}_t, \mu_j, \Sigma_j] \quad (\text{A.1})$$

where $\mathcal{N}$ is the normal density, μ_j and Σ_j are the mean vector and covariance matrix associated with the internal state j in the HMM. It is possible to use discrete observation symbols, but better performance has been reported for continuous HMMs, which is caused mainly by the vector quantisation error in discrete observation symbols, Huang, Ariki and Jack (1990).

We now define the transform

$$C_j(\omega_i) = \left| \log \frac{\mathcal{F}(\omega_i)_j}{\mathcal{F}(0)_j} \right| \quad (\text{A.2})$$

where $j \in \{F_x, F_y, M_z\}$ are the three degree of freedom force/torque measurements for a planar task from time t to $t + W$. W is a constant design parameter representing the length of the observation window. $\mathcal{F}$ is the Fourier transform and the ω_i 's are the observation frequencies. Each frequency component is divided by $\mathcal{F}(0)_j$ to make them independent of signal magnitudes. The logarithm function is used to make the covariance of the observation symbols smaller. Small covariances will make the observation probabilities larger relative to the hidden process state probabilities, which is desirable in the training and the operation of the HMMs. The reason we want the observation probabilities large compared to the hidden process state probabilities can be seen from Figure A.1. The hidden process is observed via the observation probabilities.

With n observation frequencies, the continuous vector $\mathbf{x}_t$ is then given by

$$\mathbf{x}_t = [C_{F_x}(\omega_1), \dots, C_{F_x}(\omega_n), C_{F_y}(\omega_1), \dots, C_{F_y}(\omega_n), C_{M_z}(\omega_1), \dots, C_{M_z}(\omega_n)]^T \quad (\text{A.3})$$

A.3 Length of Observation Window

Once the observation symbols are defined, we have to choose the length of the observation window, W . This choice is a compromise between the amount of information and the time spent waiting for the information. The additional information from time $t - 1$ to time t can be measured by looking at the average change in the observation vector $\mathbf{x}_t$ for all discrete events. We use a threshold test on the change of the observation vector, since the amount of information from an event will decrease with time. The threshold test is given by

$$\frac{\sum_{e_k \in E} \sum_{\mathbf{L}_t^k \in \mathbf{L}^k} \left| \mathbf{x}_t^{\mathbf{L}_t^k} - \mathbf{x}_{t-1}^{\mathbf{L}_t^k} \right| / \left| \mathbf{x}_{t-1}^{\mathbf{L}_t^k} \right|}{n_e n_l} \leq \Delta x \quad (\text{A.4})$$

where $|\cdot|$ is the vector norm, n_e is the number of events $e_k \in E$ and n_l is the size of the training sets $\mathbf{L}^k$. When the change in the observation symbol vector from time $t - 1$ to t is less than a chosen threshold percentage Δx , we stop making observations. The data used is then $\mathbf{x}_0, \dots, \mathbf{x}_t$. The threshold test in equation (A.4) is performed

once and off-line on the entire training set and the same observation length is used for all events. The same observation length is used in the on-line recognition process. By using this threshold test we assure that a certain amount of the information will be contained in the observation symbols $\mathbf{x}_0, \dots, \mathbf{x}_t$.

A.4 Training

The first step of the HMM recognition method is to build a HMM for each event $e_k \in E$. The most difficult problem in HMMs is how to adjust the model parameters to maximise the probability of the observed data matching the model. There is no known way to solve this analytically for a maximum likelihood model Huang, Ariki and Jack (1990), so using a numerical method is desirable. The method we use is a standard estimation method called the Baum-Welch re-estimation formula, see for example Rabiner and Juang (1986).

The following notation is used for a HMM based on continuous observations:

$$\begin{aligned}
 T &= \text{length of observation sequence} \\
 N &= \text{number of hidden process states in the HMM} \\
 Q &= \{q_1, q_2, \dots, q_N\} \text{ hidden process states} \\
 A &= \{a_{ij}\} \text{ HMM hidden process probability matrix} \\
 \pi &= \text{initial state probability} \\
 \delta &= \text{absorbing state probability} \\
 \mathbf{x}_t &= \text{continuous observation vector at time } t \\
 B &= \{b_j(\mathbf{x}_t)\} \text{ HMM observable process probability vector} \quad (\text{A.5})
 \end{aligned}$$

The compact notation $\lambda = (A, B, \pi)$ represents a HMM. For the model in Figure A.2 we have

$$\begin{aligned}
 N &= 7 \\
 Q &= \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\} \\
 \pi &= [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T \\
 \delta &= [0 \ 0 \ 0 \ 0 \ 0 \ \delta_1 \ 1 - \delta_1]^T
 \end{aligned}$$

where $\delta_1 \in [0, 1]$ will be changed during the training. δ_1 is the probability of ending in hidden state q_6 , while $1 - \delta_1$ is the probability of ending in hidden state q_7 .

The Baum-Welch method starts with initial values of all the HMM parameters. After applying the method, the probability of the observed data matching the HMM is increased. Applying the Baum-Welch formula once is called an iteration. For the probability to reach a (local) maximum, several training iterations are required.

The Baum-Welch re-estimation formula is based heavily on two variables, the *forward* and *backward* variables. They are defined as follows:

Definition 10 (Forward variable α)

$$\alpha_t(j) = Pr(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_t, j = q_j \mid \lambda^k) \quad (\text{A.6})$$

where $\alpha_t(j)$ is the probability of the partial observation sequence (until time t) and hidden process state q_j at time t , matching the model λ^k . $\square$

The forward variable α is calculated for every hidden process state in the HMM λ^k at every discrete time step $t = 1, \dots, T$ as follows:

1. $\alpha_1(i) = \pi_i b_i(\mathbf{x}_0) \quad 1 \leq i \leq N$.
2. Calculate $\alpha(\cdot)$ along the time axis, for $t = 2, \dots, T$, and $1 \leq j \leq N$.

$$\alpha_t(j) = \left[\sum_{i=1}^N \alpha_{t-1}(i) a_{ij} \right] b_j(\mathbf{x}_t) \quad (\text{A.7})$$

Definition 11 (Backward variable β)

$$\beta_t(i) = Pr(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \dots, \mathbf{x}_T \mid j = q_i, \lambda^k) \quad (\text{A.8})$$

where $\beta_t(i)$ is the probability of the partial observation sequence from $t+1$ to the end, given the hidden process state q_i at time t , matching the model λ^k . $\square$

The backward variable β is also calculated for every hidden process state in the HMM λ^k at every discrete time step $t = 1, \dots, T$.

1. $\beta_T = \sigma$, absorbing state probability.

2. Calculate $\beta(\cdot)$ along the time axis, for $t = T - 1, \dots, 1$, and $1 \leq i \leq N$, ie.

$$\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j) a_{ij} b_j(\mathbf{x}_{t+1}) \quad (\text{A.9})$$

Definition 12 (HMM probability) *The HMM probability is defined as the probability of the entire observation sequence $\mathbf{x}_0, \dots, \mathbf{x}_T$ matching the model λ^k . Finding this probability for a model λ^k is called scoring the model. This is done by using the forward variable α as follows.*

$$Pr(\mathbf{x}_0, \dots, \mathbf{x}_T | \lambda^k) = \sum_{j=1}^N \alpha_T(j) \quad (\text{A.10})$$

□

In a dynamic system the HMM probability for λ^k is the probability of the observed force/torque measurements matching the model λ^k for event $e_k \in E$.

From the forward and backward variables we now define two new variables describing the HMM state probabilities. The difference between HMM state probabilities compared to hidden process state probabilities a_{ij} is the consideration of *both* the hidden process and the observable process in the HMM state probabilities. These new variables will be used in the final form of the Baum-Welch re-estimation formula.

Definition 13 (HMM state probability $\sigma(j)$)

$$\sigma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{Pr(\mathbf{x}_0, \dots, \mathbf{x}_T | \lambda^k)} \quad (\text{A.11})$$

where $\sigma_t(j)$ is the probability of being in the hidden process state q_j at time t , given the observation sequence $\mathbf{x}_0, \dots, \mathbf{x}_T$ and the model λ^k . □

Definition 14 (HMM state transition probability $\xi(i, j)$)

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)}{Pr(\mathbf{x}_0, \dots, \mathbf{x}_T | \lambda^k)} \quad (\text{A.12})$$

where $\xi_t(i, j)$ is the probability of a path being in state q_i at time t and making a transition to state q_j at time $t + 1$, given the observation sequence and the model. □

We now use the definitions for $\sigma(j)$ and $\xi(i, j)$ to express the Baum-Welch re-estimation formula. For each event $e_k = (\gamma_i, \gamma_j) \in E$ we have a training set $\mathbf{L}^k = \{\mathbf{L}_1^k, \mathbf{L}_2^k, \dots, \mathbf{L}_n^k\}$

where the individual $\mathbf{L}^k$'s are matrices containing force/torque samples describing event e_k . In the planar case we have that $\mathbf{L}_l^k = \{\mathbf{l}_{Fxl}^k, \mathbf{l}_{Fyl}^k, \mathbf{l}_{Mzl}^k\}^T$ where $\mathbf{l}_{Fxl}^k$, $\mathbf{l}_{Fyl}^k$ and $\mathbf{l}_{Mzl}^k$ are vectors containing the force measurements in the x- and y-direction and the torque measurements in the z-direction, respectively, describing event e_k . These measurements are transformed into observation symbols $\mathbf{x}_t$ as described in section A.2. Given the model λ^k with initial parameters, the re-estimation formula for the internal state transition parameters a_{ij}^k is then given by

$$\bar{a}_{ij}^k = \frac{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^{T-1} \xi_t^{\mathbf{L}_i^k}(i, j)}{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^{T-1} \sum_{j=1}^N \xi_t^{\mathbf{L}_i^k}(i, j)} \quad (\text{A.13})$$

We see that the estimate $\bar{a}_{ij}^k$ is the sum of all transition probabilities from the state q_i to the state q_j divided by the probability of being in state q_i for model λ^k . The new estimates $\bar{a}_{ij}^k$ are based on the HMM state transition probabilities $\xi(i, j)$, so we see now how the hidden process is estimated through the observable process. Since the $\mathbf{x}_t = \mathbf{x}_t^{\mathbf{L}_i^k}$'s are different for each training sequence, we also have that the ξ_t 's are different for each training sequence $\mathbf{L}_i^k$ and hence are written $\xi_t^{\mathbf{L}_i^k}$.

By using multivariate Gaussian distributions, the estimates for μ_j^k and Σ_j^k for the model λ^k in equation (A.1) are given by

$$\bar{\mu}_j^k = \frac{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^T \sigma_t^{\mathbf{L}_i^k}(j) \mathbf{x}_t}{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^T \sigma_t^{\mathbf{L}_i^k}(j)} \quad (\text{A.14})$$

$$\bar{\Sigma}_j^k = \frac{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^T \sigma_t^{\mathbf{L}_i^k}(j) (\mathbf{x}_t - \bar{\mu}_j^k)(\mathbf{x}_t - \bar{\mu}_j^k)^t}{\sum_{\mathbf{L}_i^k \in \mathbf{L}} \sum_{t=1}^T \sigma_t^{\mathbf{L}_i^k}(j)} \quad (\text{A.15})$$

Theorem 1 (Maximum likelihood estimates) *The Baum-Welch re-estimation formula given in the equations (A.13) – (A.15) is an iterative solution for a (local) maximum likelihood HMM.*

Proof: See Huang, Ariki and Jack (1990). ■

Define the scalar $S^k = \sum_{\mathbf{L}^k \in \mathbf{L}} Pr(\mathbf{x}_0^k, \dots, \mathbf{x}_T^k \mid \lambda^k) / n_l$ to be the training set score for λ^k based on the entire training set, where n_l is the size of the training sets $\mathbf{L}^k$. The training procedure is then described as follows.

1. Choose an initial model λ^k for each event $e_k \in E$.
2. Get the new model parameters from equation (A.13) to (A.15). These new model parameters are based on the entire training set $\mathbf{L}^k$ for that event. This is called *supervised* training since we know that the training set $\mathbf{L}^k$ belongs to the event e_k .
3. If the change in the training set score S^k is below a chosen threshold value, the training is completed. Otherwise go to step 2.

A.5 Event Recognition

The event recognition method is described as follows.

1-Training. First we build a HMM for each event in the event set. We use the observations from the training set $\mathbf{L}$ to estimate the optimum parameters for each event e_k , represented by λ^k , for the k^{th} event in the event set. We use well known estimation techniques like the Baum-Welch re-estimation formulas.

2-Operation. From on-line measurements we have a single test sequence, $\mathbf{Y} = \{\mathbf{y}_{Fx}, \mathbf{y}_{Fy}, \mathbf{y}_{Mz}\}^T$ containing the 3-DOF force/torque measurement vectors in the x-, y- and z-direction, respectively. For each event model, we calculate the model score for λ^k , $S(e_k) = S(\lambda^k) = Pr(\mathbf{Y}|\lambda^k)$. The model score is found using Definition 12.

3-Evaluation. We choose the event whose model score is highest, ie.

$$k^* = \arg \max_{e_k \in E_{\gamma_i}} [S(\lambda^k)]$$

where E_{γ_i} is the set of admissible events for the given contact state γ_i .

By comparing the individual HMM scores, a confidence level of the recognised event is calculated. A confidence level of the final decision can be defined as follows.

$$C(e^*) = \frac{P_{k^*}}{\sum_k P_k} \in (0, 1] \quad (\text{A.16})$$

where P_k is the individual model score for HMM Λ_k and P_{k^*} is the largest HMM model score.

A.6 Evaluation

The HMM monitor allows for dynamic force/torque measurements. There are no restrictions on the system accelerations as there are for quasi-static motions. By allowing dynamic motions, the dynamic system is no longer restricted to certain kinds of trajectories. We are now able to design continuous controllers with no restrictions on the accelerations. It should be noted, however, that the quasi-static approaches do not need training and hence the implementation is less time-consuming.

The HMM method is model based where the models are trained on real measurements from the process plant. Hence, no assumptions on the measured forces/torques are made. Training the models on empirical data helps to ensure the validity of the models.

Since the method is trained on empirical data, the effects from friction measurement noise are accounted for. In fact, any linear and non-linear phenomena in the dynamic measurements are accounted for as long as the training data contain these phenomena. The HMM approach is based on a stochastic description of both the observable and the hidden process. Hence, the stochastic nature of phenomena such as friction is also taken into account. However, the method is not able to describe non-repeatable phenomena. For example, wear and tear of manipulators and workpieces may cause the original training data to be an inaccurate description of the contact forces.

The frequency band of the force measurements is large when there is a sudden change in contact formation. By using the frequency components of the measured data, we are able to extract this information. Frequency information is also able to eliminate the difference between hard and gentle contacts.

One limitation of the HMM approach is the amount of training data required. Hovland and McCarragher (1998) demonstrated a HMM process monitor for a planar robotic assembly task. A training set size of 20 training examples for each of 25 discrete events was used, which gives a total of 500 examples. Similarly, in Chapter 6 a training set size of 1580 was used for a planar mobile navigation problem. However, the average recognition rate using HMMs was demonstrated by Hovland and McCarragher (1998) to be as high as 97%. If lower performances are acceptable, smaller training sets can be used.

Appendix B

Process Monitoring Using Distance Functions

B.1 Distance Functions

In this appendix we describe the process monitor based on distance functions for classifying contact states. Each surface is modelled as a straight line and is given by the position vectors of the two endpoints. The distance functions to be presented are based on the definition by Astuti (1995). For the process monitor presented here we have added the requirement of the distance function being positive on the normal side of the surface and negative otherwise. The process monitor calculates the distances between relevant surfaces and edges of the geometric model. The distance between a surface ψ and the edge ϕ , denoted by $h_{\psi\phi}$, is defined as the nearest distance between ϕ and coordinates on the surface ψ .

Let ψ_1 and ψ_2 define the endpoints of the surface ψ . For the edge ϕ , we denote $\mathbf{d}_\phi = (p_x, p_y)^T$ as the vector coordinate of ϕ w.r.t. the inertia frame. The distance between the surface ψ and the edge ϕ is then defined as

$$h_{\psi\phi} = \begin{cases} \text{sgn}[(\mathbf{d}_\phi - \mathbf{d}_{\psi_1})^T \mathbf{n}_\psi] |\mathbf{d}_\phi - \mathbf{d}_{\psi_1}| & \text{if } (\mathbf{d}_\phi - \mathbf{d}_{\psi_1})^T (\mathbf{d}_{\psi_2} - \mathbf{d}_{\psi_1}) < 0 \\ \text{sgn}[(\mathbf{d}_\phi - \mathbf{d}_{\psi_2})^T \mathbf{n}_\psi] |\mathbf{d}_\phi - \mathbf{d}_{\psi_2}| & \text{if } (\mathbf{d}_\phi - \mathbf{d}_{\psi_2})^T (\mathbf{d}_{\psi_1} - \mathbf{d}_{\psi_2}) < 0 \\ (\mathbf{d}_\phi - \mathbf{d}_{\psi_1})^T \mathbf{n}_\psi & \text{otherwise} \end{cases} \quad (\text{B.1})$$

where $\text{sgn}(k) = 1$ if $k \geq 0$ and $\text{sgn}(k) = -1$ otherwise. The values $|\mathbf{d}_\phi - \mathbf{d}_{\psi_1}|$ and

$|\mathbf{d}_\phi - \mathbf{d}_{\psi_2}|$ describe the distances when the point ϕ is not directly above the surface ψ , see Figure B.1. The value $(\mathbf{d}_\phi - \mathbf{d}_{\psi_1})^T \mathbf{n}_\psi$ is the distance from the surface ψ along the surface normal $\mathbf{n}_\psi$ to the point ϕ .

Figure B.1: Distance functions between a surface ϕ and three points ψ_1 , ψ_2 and ψ_3 . Note that the distance between the surface ϕ and the point ψ_1 is negative caused by the direction of the surface normal $\mathbf{n}_\phi$.

B.2 Classification of Contact States

In presence of model uncertainty and noisy position measurements, the exact positions of the surfaces and the edge ϕ are not known. In this case, we can never guarantee correct contact state classification by using position measurements alone. However, the distance functions still contain useful information, since a surface and an edge with a small distance are more likely to be in contact than a surface and an edge with a large distance function.

We let the model uncertainty be described by a distance ϵ and define the function f as follows.

$$f(x) = \begin{cases} 1 - \frac{x}{\epsilon} & \text{if } 0 \leq x \leq \epsilon \\ 1 + \frac{x}{\epsilon} & \text{if } -\epsilon \leq x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.2})$$

Let $\mathcal{H}$ be the set of distance functions. Let $\mathcal{H}_i \in \mathcal{H}$ be the set of distance functions

required to be small for contact state γ_i . For example, Figure B.2 shows the surfaces

Figure B.2: *Example of geometric model of planar peg-in-hole assembly task.*

ψ and the edges ϕ for the geometric model of a planar peg-in-hole assembly task. For the contact state γ_1 shown, the set of distance functions required to be small consists of two elements, ie. $\mathcal{H}_1 = \{h_{1e}, h_{7b}\}$.

Using the sets $\mathcal{H}_i$, each contact state γ_i is scored as follows.

Definition 15 (Contact State Score) *For all states except the free spaces the contact state score is a function valued between 0 and 1 and defined as follows.*

$$S(\gamma_i) = \frac{1}{|\mathcal{H}_i|} \sum_{h \in \mathcal{H}_i} f(h) \in [0, 1] \quad (\text{B.3})$$

where $|\mathcal{H}_i|$ is the cardinal number of $\mathcal{H}_i$. For an edge ψ in a free space state no distance functions are required to be small. For a free space state γ_i , let $h_i \in \mathcal{H}$ be the distance function of the nearest surface. Then, we define the free space state score as

$$S(\gamma_i) = 1 - f(h_i) \in [0, 1] \quad (\text{B.4})$$

□

Given the contact state scores, the recognised contact state from the process monitor is easily found as follows.

Definition 16 (Recognised Contact State) *The contact state with the largest score,*

γ^* , is chosen as the recognised contact state as follows.

$$\gamma^* = \underset{\gamma_i}{\operatorname{argmax}} S(\gamma_i) \quad (\text{B.5})$$

□

Definition 17 (Confidence Level) *Given the contact state network and the current contact state, only a limited number of new contact states can be reached. We denote this subset Γ_i . The confidence level of the recognised contact state is then defined as*

$$C(\gamma^*) = \frac{S(\gamma^*)}{\sum_{\gamma \in \Gamma_i} S(\gamma)} \in [0, 1] \quad (\text{B.6})$$

□

If, for example, the current contact state is γ_3 for the contact state model in Figure 6.3, only the contact states γ_{38} , γ_{39} , γ_{40} and γ_{41} are physically reachable. Hence $\Gamma_3 = \{\gamma_{38}, \gamma_{39}, \gamma_{40}, \gamma_{41}\}$. If, for example, the contact state scores are $S(\gamma_{38}) = 0.9$, $S(\gamma_{39}) = 0.1$, $S(\gamma_{40}) = 0.2$ and $S(\gamma_{41}) = 0.3$, then the recognised contact state is γ_{38} with confidence level $C = \frac{0.9}{0.9+0.1+0.2+0.3} = 60\%$.

The sensory perception controller requires every monitor to recognise the discrete events. For the distance functions monitor, the recognised contact states are mapped to discrete events, ie.

$$\begin{aligned} e^* &= \gamma(k-1) \rightarrow \gamma^* \\ C(e^*) &= C(\gamma^*) \end{aligned} \quad (\text{B.7})$$

The distance functions monitor requires the apriori knowledge of the DES state $\gamma(k-1)$ before the event occurred.

B.3 Evaluation

The process monitor based on position measurements and distance functions does not require extensive training. The only requirement is a geometric description of the different contact states. Hence, the monitor is easy to implement. Another advantage

is the low computational resources required. The low cost makes the monitor an ideal component of a sensory perception control architecture.

The main limitation of the method is the requirement of a geometric description of each surface and edge in the world model. In practice, the geometric model can be difficult to obtain or even non-stationary. Moreover, the classified contact states and confidence levels are sensitive to uncertainties in the position measurements. Process monitoring using the distance functions is most efficient in structured and stationary environments.

Appendix C

Process Monitoring Using a Multilayer Perceptron

C.1 Multilayer Perceptron Network

The process monitor described in this appendix is based on a multilayer perceptron network (MLP) and has been presented by Hovland and McCarragher (1997a). An MLP is well suited to discrete event classification when a number of different measurements are available. A network with three layers of nodes is used. No more than three layers are required in perceptron-like feed-forward nets because a three-layer net can generate arbitrarily complex decision regions, Lippmann (1987). The network has one output node for each discrete event. Hence, when using the network as a classifier, the discrete event with the largest output value is chosen.

The output of any node in layer n is given by the outputs from the previous layer. The network is fully connected, hence the output of node i in layer $n \geq 1$ is given by

$$x_{n,i} = f \left(\sum_{j=1}^{N_{n-1}} w_{n-1,ij} x_{n-1,j} \right) \quad (\text{C.1})$$

where N_{n-1} is the total number of nodes for layer $n - 1$ and w_{ij} is the network weight between node i in layer n and node j in layer $n - 1$. The inputs to the nodes in the input layer are given by the force and position measurements, ie.

$$x_{0,j} \in \mathcal{X}_0 = \{\mathbf{p}_0, \mathbf{F}_0(t), \dots, \mathbf{F}_0(t + KT)\} \quad (\text{C.2})$$

Figure C.1: *MLP neural network with one hidden layer. Each node in the input layer has $K + 2$ input vectors. The first input vector, $\mathbf{p}_0$, contains the distance functions. The following inputs, $\mathbf{F}_0$, contain the frequency-domain force symbols. E is the number of discrete events.*

The force and position input vectors are described in section A.2 and B.1. Frequency-domain force/torque measurements, equation (A.3), are used to generate the force input vectors $\mathbf{F}_0(t)$ and the position input vector $\mathbf{p}_0$ consists of the set of distance functions, equation (B.1).

The particular training algorithm to be presented in section C.2 assumes the sigmoid form of f , ie.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (\text{C.3})$$

The sigmoid function guarantees that the node outputs are between 0 and 1.

C.2 Training Algorithm

The training uses the back-propagation algorithm. For a detailed description of the method, see for example Lippmann (1987). The algorithm may be summarised by:

$$\Delta w_{n-1,ij} = \eta \delta_{n,i} x_{n-1,j} \quad (\text{C.4})$$

where $\eta > 0$ is the update parameter. When the weights lead into the output layer, $\delta_{n,i}$ is given by:

$$\delta_{3,i} = (d_i - x_{3,i})(x_{3,i}(1 - x_{3,i})) \quad (\text{C.5})$$

d_i is the desired output for node i in layer $n = 3$. The desired outputs are either 0 or 1. For weights leading into nodes preceding the output layer, $\delta_{n,i}$ is given by:

$$\delta_{n,i} = \left(\sum_{j=1}^{N_{n+1}} \delta_{n+1,j} w_{n+1,ij} \right) (x_{n,i}(1 - x_{n,i})) \quad (\text{C.6})$$

where $w_{n+1,ij}$ is the weight between node i in layer n and node j in layer $n + 1$. N_{n+1} is the number of nodes in layer $n + 1$.

The training algorithm is now described by the following procedure:

1. The network weights $w_{n,ij}$ are initially set to small random numbers.
2. Forward pass: For a given input, calculate all outputs for all layers by using equations (C.1) – (C.3).
3. Backward pass: By using the node outputs from step 1, calculate $\Delta w_{n-1,ij}$ by using equations (C.4) – (C.6). d_i is set to 1 for the output (discrete event) corresponding to the current training example and 0 for the other outputs.
4. Step 2 and 3 are repeated for each training sequence in the training set before the weights $w_{n,ij}$ are updated. This is called a *sweep*.
5. Step 2–4 are repeated until the network error

$$\epsilon = \frac{1}{\epsilon_0} \sum_{i=1}^E \sum_{j=1}^{N_s} \frac{1}{2} (d_i - x_{3,ij})^2 \quad (\text{C.7})$$

is below a desired value. ϵ_0 is the error after the first sweep (iteration) and E is the number of network outputs. $x_{3,ij}$ is the output value for node i in the output layer for training example j for event e_i . N_s is the number of training samples.

C.3 Real-Time Operation

Once the MLP network has been trained, it can be used efficiently in real-time operation. First, a discrete event detection algorithm is needed. For assembly, a change in

the constraint equations is characterised by a sudden change in at least one of the force measurements. The event detector is the same as in section 5.5.2 and is described by the following threshold test.

$$[F_x(t) - F_x(t - \Delta T)]^2 + [F_y(t) - F_y(t - \Delta T)]^2 \geq \Delta F \quad (\text{C.8})$$

where ΔT and ΔF are design parameters. If the threshold test is able to detect gentle contacts, it will also detect brutal contacts. Hence, the design parameters are found by looking at the force measurements for gentle contacts.

Once a discrete event is detected, it is classified by using the MLP network and the forward pass described in the previous section. Given the real-time force and position measurements, the forward pass calculates the outputs of all the network outputs $x_{3,i}$. Given the current contact state γ , only discrete events in the set of admissible events $\mathcal{E}_\gamma$ can occur. The scores for the admissible events are given by $S(e_k) = x_{3,k}$. The discrete event $e^* \in \mathcal{E}_\gamma$ with the largest output value is chosen, ie.

$$e^* = \arg \max_{e_k \in \mathcal{E}_\gamma} S(e_k) \quad (\text{C.9})$$

The different values of the MLP outputs corresponding to the admissible events can be used to calculate a confidence level $C(e^*)$ of the recognised event e^* , for example

$$C(e^*) = \frac{S(e^*)}{\sum_{e_k \in \mathcal{E}_\gamma} S(e_k)} \in [0, 1] \quad (\text{C.10})$$

C.4 Evaluation

One advantage of the proposed method compared to other existing solutions for recognition of contact state transitions is the modelling of both dynamic and static behaviour. The dynamic force measurements depend on a number of system parameters, such as workpiece and environment stiffnesses, sensor noise and dynamics manipulator dynamics. These factors may vary from one contact state to another and hence help to improve the performance of the discrete event recognition. The position measurements, on the other hand, contain mostly static information during the short time from an event occurs until it is recognised by the monitor. The MLP network successfully incorporates both static and dynamic measurements.

One limitation of the proposed method is the relatively long time required to collect event training examples and the large number of training iterations required by the MLP. The training of the MLP weights is computationally very costly. For the assembly task presented in Chapter 5, a set of 20 training examples for each of the 25 discrete events was generated, which gives a total of 500 training examples. The training of the MLP network required several CPU hours on a SPARC Ultra.

The incorporation of several different data types is straight-forward when using MLPs. In this appendix an MLP event classifier using force and position measurements is described. The addition of other different sensor information, such as real-time images, requires small modifications to the method. However, the number of input symbols will increase which increases the number of weights and training times of the MLP.

The main advantage of the MLP in a sensory perception control architecture is its high discrete event recognition rates. However, the real-time costs are relatively large. Ideally, the MLP classifier will only be used when faster monitoring techniques fail to recognise discrete events correctly.

Appendix D

Process Monitoring Using Qualitative Reasoning

D.1 Qualitative Reasoning

McCarragher and Asada (1993) developed a process monitoring technique based on qualitative reasoning. Dynamic analysis was used to model a robotic assembly task. Constrained and unconstrained rigid body dynamics and an impact analysis were included to determine expected force signals for assembly processes.

The process monitor based on qualitative reasoning looks at changes in the planar force measurements, ΔF_x and ΔF_y . McCarragher and Asada (1993) found the qualitative templates for a gain and loss of contact from the general constrained equations of motion. The qualitative value of a signal, S , is denoted $[S]$. A qualitative value can be represented using a quantity space representation of $\{-, -, 0, +, ++\}$. McCarragher and Asada (1993) used this representation for the derivative of the force signals. For a loss of contact it was found that either $[\Delta F_x] \in \{-, +\}$ or $[\Delta F_y] \in \{-, +\}$. Similarly, for a gain of contact it was found that either $[\Delta F_x] \in \{-, ++\}$ or $[\Delta F_y] \in \{-, ++\}$. Let F_{--} , F_- , F_+ and F_{++} be the force thresholds for the quality values $[-, -, +, ++]$. McCarragher and Asada (1993) found the force thresholds F_- , F_+ from a 3σ deviation from the mean force values in free space. The force thresholds F_{--} , F_{++} were found from a 3σ deviation from the mean forces occurring during an impact. Each discrete

event is described by at least one quality value, $q_x(e_k)$ for ΔF_x or $q_y(e_k)$ for ΔF_y .

Figure D.1: *Two example events for a planar peg-in-hole assembly task.*

Figure D.1 shows two example discrete events for a planar peg-in-hole assembly task. The event e_1 illustrates a gain of contact, while e_2 illustrates a loss of contact. The quality values for these two events are given by $q_x(e_1) = 0$, $q_y(e_1) = ++$ and $q_x(e_2) = 0$, $q_y(e_2) = -$.

D.2 Confidence Levels

McCarragher and Asada (1993) did not discuss how to assign confidence levels to the recognition of discrete events based on the qualitative analysis. In this appendix we define the confidence levels using the membership functions illustrated in Figure D.2. In the current contact state γ only a subset of all possible events are physically admissible given by the contact state network. Let $\mathcal{E}_\gamma$ be the set of admissible discrete events in contact state γ . Let $q_x(e_k)$ and $q_y(e_k)$ be the quality values describing event $e_k \in \mathcal{E}_\gamma$. We now define membership functions f_q as functions of force as illustrated in Figure D.2 where $q \in \{--, -, 0, ++\}$.

Using these definitions, based on the quantitative force measurements ΔF_x and ΔF_y each event e_k is scored as follows.

$$S(e_k) = \frac{1}{2} \left[f_{q_x(e_k)}(\Delta F_x) + f_{q_y(e_k)}(\Delta F_y) \right] \quad (\text{D.1})$$

Figure D.2: *Membership functions used to find the confidence levels of recognised contact states from process monitor based on qualitative analysis.*

If an event is characterised by only one template $i \in \{x, y\}$, then the score is

$$S(e_k) = f_{q_i(e_k)}(\Delta F_i) \quad (D.2)$$

Given the set of physically admissible events $\mathcal{E}_\gamma$ the confidence level of each event $e_k \in \mathcal{E}_\gamma$ is defined as

$$C(e_k) = \frac{S(e_k)}{\sum_{e_k \in \mathcal{E}_\gamma} S(e_k)} \in [0, 1] \quad (D.3)$$

The discrete event with the highest confidence level is chosen as the recognised discrete event, ie.

$$e^* = \operatorname{argmax}_{e_k} C(e_k) \quad \text{where } e_k \in \mathcal{E}_\gamma \quad (D.4)$$

D.3 Evaluation

Since the monitoring is only based on qualitative reasoning and force measurements, the monitor has problems classifying contacts in systems with several parallel surfaces. For example, gain of contacts with parallel surfaces have identical force templates.

McCarragher and Asada (1993) listed three advantages of the qualitative monitor. 1) The qualitative process monitor allows for a fast determination of the change of state; one does not have to wait for the force transients to die out. The time for identification and verification is low. 2) There are often times in dynamic systems when two contact states are separated by a short-lived dynamic state. The fast qualitative monitor is able to detect such state changes. 3) the monitor is able to make a dynamic distinction between the transitions that result in a loss of contact from those that result in a gain of contact. Moreover, the method does not require the collection of large force/torque training sets.

The main advantage of the monitor from the sensory perception control perspective is the low computational effort required. Moreover, for tasks such as assembly, the force measurements allow for a high successful event recognition rate. The monitor is a valuable component of sensory perception control systems where a low sensing cost is one of the main criteria.