Shared Discrete Event Control of Human-Robot Systems

Peter H. Aigner
B.E.E.E. (Hons), R.M.I.T.

October 1998

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, except where explicitly stated, is original research the major portion of which has been done by the author. He worked under the supervision of the members of the advisory panel, namely Dr. Brenan McCarragher (chair and main supervisor), Dr. Jon Kieffer and Prof. Darrell Williamson. This work has not been submitted for a degree at any other university or institution.

Much of the research contained in this thesis has been published in or submitted to journals and conferences listed below.

Journal Papers:


Conference Papers:


Canberra, October 1998.

Peter Aigner
Department of Engineering
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200, AUSTRALIA.
Acknowledgements

Foremost I express my sincere thanks to my supervisor Brenan McCarragher for his continuous support during my time at the ANU. Brenan always encouraged me to see the big picture of my project. He was always able to motivate me when I saw no solutions. I would like to thank his endless efforts in commenting on my work and making suggestions where appropriate. He taught me a great deal which will serve me well in the future.

I would like to thank my parents. They are the ones who helped support me and provide the encouragement to pursue higher goals. A special thanks to my grandfather, Dr. W. Sprenkman, for the great conversations I have had with him when I visited. His stories provided me with the enthusiasm I needed to pursue my PhD. A thanks to my special friend Rebecca. Her presence towards the end of my studies was a welcome distraction as well as welcome driving force.

I wish to thank my friends of room 121 in the Engineering Department. Although the atmosphere in this office was not always conducive to work it certainly made the pursuit of my PhD very enjoyable. Initially there were Aidan Cahill and Geir Hovland. Aidan’s humor is still remembered on occasion as a valuable addition to room 121. Geir made an excellent and helpful colleague as well as good friend. His efforts to help me in the initial stages of my PhD are still drawn upon and appreciated. I want to thank David Austin, who joined Geir and myself in 1997. David who shared my interests in electronics was a very useful design partner. His knowledge and help in using unix and linux was always very welcome. I thank Jason Chen whose recent arrival has once again completed room 121 after Geir’s departure.

There are many other people to acknowledge so please forgive me if I forgot. Peter Dower’s “light” remarks were a great distraction from the sometimes frustrating bureaucracy. Werner Kraus’s soccer madness was enough to motivate for an indoor soccer game at any time of day.
I want to also thank the members of my thesis committee, Jon Kieffer and Darrell Williamson. I acknowledge the financial support from the Australian National University which allowed me to pursue my PhD at a great Australian university.

P.H.A
October 1998
"No one knows what it is that he can do till he tries."

Publilius Syrus
Abstract

The aim of this thesis is to expand and enhance discrete event control theory by integrating human decision making into the control of an otherwise autonomous system. Discrete event systems are an excellent tool for modelling and controlling complex systems. However, these systems, when employed to control a continuous time plant, typically suffer from difficulties in the conversion from the discrete to the continuous domain and vice versa. Process monitoring is an aspect of discrete event systems suffering from imperfections. The difficulty of sensing is compounded in unstructured and dynamic environments. Control synthesis, the conversion from the discrete into continuous time commands, suffers from a lack of practical methods.

Shared control on the other hand offers advantages such as human decision making capabilities and monitoring abilities. However, some tasks are difficult for humans and are better suited to autonomous control. Shared control combines the best of human abilities with the best of robotic abilities. Today shared control is applied to a wide variety of applications including health-care and service robotics. However, shared control lacks a consistent framework which can be applied to the wide variety of tasks. In this thesis the advantages of shared control are drawn upon to improve on the difficulties of discrete event control systems. Similarly, shortcomings of shared control are aided by discrete event systems. This combination is an important step towards more flexible automation systems.

The approach presented proposes a discrete event framework which implements a distributed control system with two controllers. The first of these controllers is an autonomous discrete event controller which controls the system without human interaction. The second controller is a discrete model of human interactions which, depending on human input, influences the commands given by or alters the state of the autonomous controller. The framework proposed makes use of features of discrete event systems to implement the command combination. The discrete modelling aspects are also of advantage in modelling human interactions, which can be quite complex.

A control synthesis method is also needed which allows the human to combine his control
commands with those of the autonomous controller. As mentioned, in discrete control systems there is a conversion from discrete to continuous (control synthesis) and vice versa (process monitoring). For shared control the human needs to be able to interact in both of these conversions. To facilitate this interaction two control techniques were studied, the first based on constraints and the second based on potential fields. These methods are suited to continuous interaction by the user. The user can also interact on a discrete level to affect the autonomous discrete event system directly.

The effectiveness of shared control and the operation of the framework is demonstrated by experiments. It is shown how the completion of a task can be greatly improved by shared control as opposed to autonomous or human control alone.

Additionally, the shared control framework is applied to the Robotic Cane, a robotic aid for the blind. The robotic cane requires human interaction to be a useful aid in a larger variety of tasks. Not only does the need for human interaction make the cane a useful testbed but it also demonstrates the operation of the shared control framework in a real-world example. It also shows some important interactions between the human user and the cane. Finally, the cane is used in experiments to show its value as a travelling aid for visually impaired users.
Contents

Declaration i

Acknowledgements iii

Abstract vi

Nomenclature xvii

1 Introduction 1

1.1 Discrete Event Control and Robotics 3
1.2 Shared Control in Robotics 5
1.3 Discrete Event and Shared Control 7
1.4 Contributions 9
1.5 Organisation of the Thesis 10

2 Modelling for Shared Control 12

2.1 The Discrete Event Formalism 12
2.2 The Shared Control Framework 13
2.3 The Plant 14
2.4 Autonomous Discrete Event Controller 15
2.5 Human Discrete Event Model 16
2.6 Interface 18
2.7 Hierarchical Modelling of the HDEM .................................................. 20
2.8 Conclusion .................................................................................................. 23

3 Control Synthesis and Human Interaction ................................................. 25

3.1 Position Constraints .................................................................................. 26
  3.1.1 Constraint Equations ........................................................................... 27
  3.1.2 Human Interaction .............................................................................. 29
  3.1.3 Constraints and the Physical World .................................................... 31
  3.1.4 Velocity Generation ............................................................................ 32

3.2 Experiments Using Constraints ................................................................. 33
  3.2.1 Without Human Interaction ................................................................. 37
  3.2.2 With Human Interaction .................................................................... 38

3.3 Potential Fields .......................................................................................... 44
  3.3.1 Modelling with Potential Fields .......................................................... 45
  3.3.2 Velocity Generation ............................................................................ 48
  3.3.3 Human Interaction ............................................................................. 48

3.4 Experiments Using Potential Fields .......................................................... 50
  3.4.1 Without User Interaction ................................................................... 50
  3.4.2 With User Interaction ....................................................................... 51

3.5 Evaluation of Constraints and Potential Fields ....................................... 58
  3.5.1 Human Interaction ............................................................................. 58
  3.5.2 Computational Comparison ............................................................... 59

3.6 Conclusion .................................................................................................. 60

4 Case Study: The Robotic Cane ................................................................. 62

4.1 The Robotic Cane ...................................................................................... 64
4.2 The Control System ................................................................................. 67
6.3.2 Five Supervisory Control Functions ........................................ 100
6.3.3 Analytical Research ................................................................. 101

Bibliography .......................................................... 101

A Experimental Setup .................................................. 111
   A.1 The Robot and the joystick ........................................... 111
   A.2 Computing Hardware .................................................. 112
   A.3 Client and Server Software .......................................... 113
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Block diagram of a hybrid dynamic system with human integration</td>
<td>13</td>
</tr>
<tr>
<td>2-2</td>
<td>The Human Discrete Event Model</td>
<td>17</td>
</tr>
<tr>
<td>2-3</td>
<td>Interface of combined hybrid dynamic system</td>
<td>19</td>
</tr>
<tr>
<td>2-4</td>
<td>Hierarchical model of the HDEM</td>
<td>21</td>
</tr>
<tr>
<td>3-1</td>
<td>Geometric definitions for constraints</td>
<td>27</td>
</tr>
<tr>
<td>3-2</td>
<td>Hierarchical HDEM model utilised for two types of continuous interaction</td>
<td>29</td>
</tr>
<tr>
<td>3-3</td>
<td>Example of a constraint conflict</td>
<td>32</td>
</tr>
<tr>
<td>3-4</td>
<td>Example of L-shape obstacle problem</td>
<td>32</td>
</tr>
<tr>
<td>3-5</td>
<td>Graphical representation of constraints</td>
<td>33</td>
</tr>
<tr>
<td>3-6</td>
<td>Experimental setup to test control using constraints in an obstacle avoidance</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>task using a 5-degree of freedom Eshed Scorbot VII</td>
<td></td>
</tr>
<tr>
<td>3-7</td>
<td>(a) Physical layout and ADEC states (b) ADEC automaton</td>
<td>37</td>
</tr>
<tr>
<td>3-8</td>
<td>Experimental results - autonomous operation (a) ADEC states and total velocities, (b) HDEM states and human velocities</td>
<td>38</td>
</tr>
<tr>
<td>3-9</td>
<td>Trajectory during autonomous operation</td>
<td>38</td>
</tr>
<tr>
<td>3-10</td>
<td>Experimental results - human interaction (a) ADEC states and total velocities, (b) HDEM states and human velocities</td>
<td>39</td>
</tr>
<tr>
<td>3-11</td>
<td>Trial stages 1, 2 and 3</td>
<td>40</td>
</tr>
<tr>
<td>3-12</td>
<td>Experimental results - constrained operation (a) ADEC states and total velocities, (b) HDEM states and human velocities</td>
<td>41</td>
</tr>
</tbody>
</table>
3-13 Trajectory during constrained operation ...................................................... 42
3-14 Operator moving the obstacle from position 1 to position 2 ......................... 43
3-15 Operator moving the obstacle into the workspace after robot commenced motion 43
3-16 Potential field of a conical attractive well .................................................. 45
3-17 Potential field of repulsive elliptic obstacle ................................................. 46
3-18 Potential field of the boundary ................................................................. 47
3-19 Cross section of boundary, effects of m. Solid line: m = 100; Dashed: m = 200 . 47
3-20 Potential field combining well, boundary and circular obstacle. The dot marks the well location. ................................................................. 48
3-21 Hierarchical HDEM model utilised for two types of continuous interaction . 49
3-22 Experimental setup for manoeuvring task using a 5-degree of freedom Eshed Scorbot VII .............................................................. 51
3-23 Experimental trajectory of robot moving inside composite potential field . 52
3-24 Experimental trajectory, human pushing against boundary (a) \( u_{MAX}^H = 0.05m/s \)  
(b) \( u_{MAX}^H = 0.04m/s \) ................................................................. 53
3-25 Experimental trajectory, human pushing into obstacle ................................. 54
3-26 Experimental trajectory, human assisting at local minimum ....................... 54
3-27 Experimental trajectory with human modifying obstacle position ............... 55
3-28 Experimental trajectory with human modifying obstacle size .................... 56
3-29 Experimental trajectory, human moving well ............................................. 56
3-30 Paths used for smoothness analysis .......................................................... 59
3-31 Smoothness figures for the constraint and potential field method for x and y velocities ............................................................... 60

4-1 The Robotic Cane ........................................................................................ 63
4-2 Schematic of the Robotic Cane .................................................................... 65
4-3 Sensing platform with three ultrasonic sensors, steering servos and microprocessor 66
4-4 Sensor arrangement and sensing angles (top view) .................................... 67
List of Tables

2.1 HDEM events ................................................. 17

3.1 States of the ADEC automaton ...................... 36

4.1 Control mapping, $\phi'(\nu(k), H)$, from ADEC and HDEC state to combined cane control command ........................................ 77

4.2 Control mapping, $\phi''$ ........................................ 78

4.3 Experiments conducted with combined velocity and sensor confidence constraints 84
Nomenclature

Acronyms

ADEC = Autonomous Discrete Event Controller
HDEM = Human Discrete Event Model
CTS = Continuous Time System
DEC = Discrete Event Controller
HDS = Hybrid Dynamic System

Symbols

ADEC (HDEM) transition function
ADEC (HDEM) output function
set of ADEC (HDEM) controller events
set of ADEC (HDEM) events
set of ADEC (HDEM) states
lower hierarchy HDEM transition function
lower hierarchy HDEM output function
set of lower hierarchy HDEM controller events
set of lower hierarchy HDEM events
set of lower hierarchy HDEM states
set of boundary constraints
ADEC state at time k
HDEM state at time n
HDEM event
HDEM event triggered by human input
HDEM event triggered by plant
lower hierarchy HDEM event
\[ \tau(k) = \text{ADEC event at time } k \]
\[ \tau^H(n) = \text{ADEC event triggered by HDEM} \]
\[ \tau^A(n) = \text{ADEC event triggered by plant} \]
\[ \eta(n) = \text{HDEM controller event} \]
\[ \nu(k) = \text{ADEC controller event} \]
\[ \chi^H() = \text{map to convert from human input to HDEM events} \]
\[ \phi^A = \text{discrete to continuous map of ADEC controller events} \]
\[ \psi^A = \text{continuous to discrete map of plant events affecting the ADEC} \]
\[ \psi^H = \text{continuous to discrete map of plant state to HDEM events} \]
\[ \Pi_{\phi hi} = \text{map to convert lower hierarchy HDEM events to upper level HDEM events} \]
\[ H = \text{human input} \]
\[ h(t) = \text{continuous component of human input} \]
\[ u(t) = \text{combined continuous input vector} \]
\[ u^A(t) = \text{continuous command from autonomous (human) system} \]
\[ \Omega() = \text{function to combine } u^A(t) \text{ and } u^H(t) \]
\[ f() = \text{description of continuous system} \]
\[ x(t) = \text{continuous state vector} \]
\[ q(t) = \text{position vector} \]
\[ q(t) = \text{velocity vector} \]
\[ \dot{h}(t) = \text{human input acceleration} \]
\[ a = \text{constraint vector} \]
\[ h_{\rho,\sigma} = \text{distance between a point and surface constraint} \]
\[ h_{\rho,\sigma} = \text{distance between a point and point constraint} \]
\[ d_{\rho,\sigma} = \text{position vector of point } \sigma \text{ or } (\rho) \]
\[ d_{(\rho)} = \text{position vector of a point (on surface } \sigma) \]
\[ n_{(\sigma)} = \text{normal to surface } (\sigma) \]
\[ d' = \text{starting position vector of a constraint} \]
\[ \Delta d = \text{change in the position vector of a constraint} \]
\[ J() = \text{function to solve a set of constraints} \]
\[ U_t(w,o,b) = \text{total potential field (well, object, boundary)} \]
\[ A_w(o,b) = \text{scaling constant of potential field of well (object, boundary)} \]
\[ (x',y') = \text{starting position of potential field} \]
$(\Delta x, \Delta y)$ = displacement of potential field

$v(t)$ = speed of robotic cane

$i, j$ = reference frame attached to the Robotic Cane

$(x(t), y(t))$ = position of robotic cane at time $t$

$\theta(t)$ = steering angle of robotic cane

$c(t)$ = sensor confidence

$a_{\text{max}}$ = maximum deceleration of Robotic Cane

$k_{1,2,3}$ = scaling constants

$s()$ = smoothness function
Chapter 1

Introduction

The aim of this thesis is to enhance and expand hybrid dynamic systems theory by integrating advanced human decision making into the control of an otherwise autonomous robotic system. With such an integration it will be possible to draw upon the advantages of the human operator and those of a robotic system. It is shown, by drawing upon the respective advantages, how shared control can improve the performance of a system compared to those which operates under autonomous or human control alone.

The terms “Hybrid Dynamic System (HDS)” and “Discrete Event Controller (DEC)” are used frequently throughout this thesis. It should be noted that a hybrid dynamic system incorporates a discrete event controller as well as a number of other sub-systems. The term hybrid comes from the joining of a discrete element, the DEC and a continuous system, such as a robot. The exact make-up of the HDS is explained later. The DEC usually models the process which is to be executed using the robot thus the phrase “discrete event modelling” refers to the design of the DEC.

Hybrid dynamic systems, incorporating a discrete event controller and a plant continuous in time (such as a robot), have been proven successful in a variety applications [14] [15] [40] [57] [62] [96]. The success of discrete event control of robotic systems has two fundamental reasons. The first reason comes about from the nature of some robotic tasks. Robotic assembly tasks, for example, inherently can be modelled as a sequence of asynchronous discrete events. In such a task an event would be the gain or loss of a contact between the two pieces to be assembled. A similar event model can be applied to assembly lines where the completion of a particular task can be recognised as an event. The second reason of why hybrid dynamic systems offer advantages over a purely continuous control system is that the nature of the discrete event controller allows complex systems to be broken up into smaller sub-systems [96]
CHAPTER 1. INTRODUCTION

which, individually, are easier to analyse and design. This breakdown is made possible because the discrete event controller provides a means to understand how complex sub-systems interact with each other to yield a complete control system.

From an implementation point of view, hybrid dynamic systems have many aspects which bring with them some added complexity. Firstly, the control system must recognise events (process monitoring). In the control of robotic systems these events are usually derived from a continuous state vector of the plant. This can often be difficult and error prone, particularly in complex environments [48]. Secondly, from the previous events and the desired goal an event trajectory must be planned. According to this trajectory the discrete controller issues a discrete command to reach the desired goal. Thirdly, a conversion from discrete control commands to continuous commands (control synthesis), which are suitable for the plant, must then take place. Finally, the stability of the hybrid dynamic system and the discrete event controller is often difficult to prove.

To develop a more powerful control theory that overcomes the problems in hybrid dynamic systems, I will integrate shared control into hybrid dynamic systems theory. Shared control is growing in popularity in many applications. On the topic of tele-robotics, "People are still very much involved. This is because many of the jobs to be done are non-repetitive and unpredictable, and therefore cannot be done by special-purpose machines that can be set up, preprogrammed, and then left to work by themselves. Or the jobs are one-of-a-kind, such that dedicated automatic devices to do them are too costly. So human perception, planning, and control are still required." [84]. Shared control allows a task to be shared between a machine and a human user. This sharing is critical in areas where humans inherently have to cooperate with machines such as the health-care, home-assistance and the service industries. The robot and the human user enhance each other's capabilities and support each other's weaknesses. It is for these reasons that shared control can be used in a wide variety of tasks including health-care [2] [17] [22] [80] [91], home-assistance [33] [60] [89], the service industries [3] [19] [61] [82] [92] and automobile driving [83]. All these applications by nature involve humans, either as operators, supervisors or users. By integrating an operator into a robotic control system a more intelligent, more adaptable and flexible machine can be realised. Tasks such as the disposal of hazardous materials [46], farming [45], monitoring systems [1] and tele-operation can be realised with one shared control system. These examples and many others demonstrate that the scope of shared control systems is expanding. Not only is the scope expanding, but the applications are becoming increasingly complex as indicated in [49] [74] [77]. To provide a consistent means to model potentially complex shared control systems, a model of a human operator should be integrated into the control of robotic systems using one consistent framework.
1.1 DISCRETE EVENT CONTROL AND ROBOTICS

Shared control offers the primary advantage of integrating human decision making and monitoring abilities into an otherwise autonomous control system. This fundamental ability assists in addressing some of the implementation difficulties of hybrid dynamic systems. In particular, this thesis addresses the conversion from the discrete to the continuous domain. This conversion can be influenced by the user in order to modify the robot's path or correct modelling inaccuracies. Difficulties associated with hybrid dynamic systems such as process monitoring and discrete trajectory planning are implicitly incorporated into some aspects of this thesis. Despite these additional aspects of hybrid dynamic control systems, the asynchronous nature of discrete event models makes them ideal when dealing with human operators. Humans interact with a control system as they see fit, which is likely to be at random intervals thus making the interactions asynchronous, a feature inherent in hybrid dynamic systems. Additionally the breakdown into sub-systems makes them attractive to applications where continuous control is too complex or inappropriate. For these reasons hybrid dynamic control theory is used in a variety of applications. However, the majority of these applications have been in fully autonomous systems. Unfortunately, in fully autonomous systems the disadvantages of hybrid control systems are amplified. As indicated, a human operator coupled with a hybrid dynamic system can easily improve on or even overcome some of the disadvantages. A paradigm of shared control is that the autonomous part and the human operator augment each other's weaknesses.

The expansion of hybrid dynamic systems theory to allow for shared control is the subject of this thesis. A shared hybrid dynamic control framework is developed which integrates a model of human interactions into the control system of an otherwise autonomous robot. This allows the user to control a robot together with a discrete event controller to improve on the performance of the autonomous or human system alone. The ultimate goal of this research is to reach a balance between the decisions a discrete event controller makes and those which the human operator makes. For example, the discrete event controller could make decisions based on system-wide performance and the human operator could make decisions based on local conditions that are not modelled by the discrete controller. In this case it is assumed that the human is capable of making quick intuitive decisions about localised model inconsistencies whereas the robot is focussed on the end goal of a task.

1.1 Discrete Event Control and Robotics

Shared control using hybrid dynamic systems has associated with it two fundamental components. The first of these is hybrid dynamic control and the second is shared control. Both of these must be examined individually before a discussion on the combination of the two can take
place. This section provides a background of hybrid dynamic systems and their use in robotics. Discussed are the modelling approaches used, their advantages and their shortcomings.

Hybrid dynamic systems have received a lot of attention in recent years, for examples see [16] [25] [26]. In some applications discrete event control simply lends itself as the “perfect” solution. In such application events can usually be easily defined and also recognised. In other systems the interactions between sub-systems often makes continuous control systems useless. Discrete event modelling seeks to simplify the modelling of complex systems by breaking them into smaller sub-systems which can be modelled more easily. Therefore discrete event systems are an ideal way of modelling and controlling large complex systems. If the resultant system is still too complex, tools exist to break down large discrete event models into smaller systems [98].

When a continuous plant is to be controlled using a discrete event controller, the hybrid dynamic system has three main sub-systems [42] [86]. The first of these sub-systems is an autonomous discrete event controller (ADEC). The second sub-system is the plant, continuous in time, which performs the physical task. Finally, an interface is employed to enable communication between the discrete and continuous domain. The operation of this set of sub-systems presents some challenges. Consider a process monitor which observes the plant. If a change (movement, pressure, illumination, loudness, etc.) in the plant occurs an event may have occurred. The process monitor must recognise that an event has occurred as well as determine the nature of the event. This process monitoring can be considered the conversion from the continuous time domain into the discrete event domain. Given that an event has occurred, the discrete event controller usually changes state - recognising that the plant has also changed. A new discrete control command is then issued by the discrete event controller which must then be converted back to the continuous domain for the plant to execute it. The plant then continues its task with the new control command until another event is recognised by the process monitor. This method of control has been proven successful in manufacturing systems [25], robotic assembly [62] [65] [66], mobile robot navigation [14] [56] [57] and many others [42] [86]. With this approach to control there exist problems in event recognition, discrete command generation and conversion of the discrete command into a continuous command which the plant can understand.

The first difficulty related to hybrid dynamic systems and the focus of some research is event recognition or process monitoring. Event recognition can performed using position sensing, force sensing, tactile sensing or vision systems. However, technologies in some of these areas, particularly in vision [87], are still immature. Vision sensing in particular has great processing needs which often rules it out as a sensing means. Hovland [48] improves on conventional methods of event recognition by utilising dynamic sensor selection. In this method additional sensors are used if the confidence of a single sensor detecting an event is too poor. To date
1.2. SHARED CONTROL IN ROBOTICS

there are no event recognition methods that work perfectly in all cases.

There exist a number of modelling techniques for the discrete event controller. These mainly stem from problems in the computer sciences and communication networks. Discrete event controllers can be modelled using automata or communicating sequential networks [47]. Another technique is based on Petri Nets [64] [88], where transitions between states are either enabled or disabled. This Petri Net technique lends itself to event trajectory planning as the enabled or disabled transitions can be used to define event paths [66]. These methods are well understood and do not present a difficulty when using discrete event control.

To control a robot (which is inherently a continuous time system) using a discrete event controller, a method of control synthesis needs to be devised to convert from the discrete into the continuous domain. The discrete event controller generates controller events. From these, the control synthesis is responsible for generating commands continuous in time, appropriate for the plant. The continuous plant then executes these commands.

A method presented in [62] [65] uses constraints to reduce, maintain or increase distances from surfaces in an assembly process. This constraint method of control synthesis works well in its application to assembly. However, this method has not been extended outside of the assembly application.

1.2 Shared Control in Robotics

This section contains a background of shared control as applied to robotic systems. It is the second component which contributes to the shared hybrid dynamic control framework presented in this thesis. This section provides motivation for shared control as well as presenting the shortcomings of current shared control methods.

In the broad field of shared control in robotics the human operator cooperates with the robot. The operator or user communicates goals, constraints and suggestions to a controller if these were not pre-defined or have changed. The user can also inquire about the status of the system. The otherwise autonomous controller is responsible for executing the task autonomously using its own sensors and actuators provided no additional instructions are given by the user. Such a shared control system draws on the advantages of both the human and the robot and can therefore extend the capabilities beyond what either the human or the robot could accomplish alone.

Robots are ideal for performing repetitive tasks quickly and accurately. The automation of the repetitive components of a task can relieve the human of some of the task-load. Robots can
move and manipulate large and heavy objects with ease, something a human cannot. Should either the human or the autonomous controller fail they can serve to back each other up. Such a control system can also execute a task automatically if there is no need for the user to intervene. Computers can record large amounts of information that can be incorporated into a controller, allowing for data storage and statistical decision making capabilities beyond those of humans.

On the other hand, in more complicated tasks, which may be trivial for humans, robots often don't perform well. Humans are excellent sensors in local environments. Humans are generally good decision makers in unpredictable environments. We can make accurate decisions quickly and sometimes without full knowledge of the entire system. These are all reasons as to why human cooperation with robots is essential to build flexible and adaptable machines. The level of cooperation between the operator and the robot can greatly depend on the task at hand, the relative abilities of the human and the robotic system to accomplish the task. Therefore a flexible control system, allowing for the variability in the human and robot, is needed.

The function of the user in a shared control system is to assist the machine. For example, in robotic assembly tasks such as a peg-in-hole task ([15] [63]) the pieces to be mated can suffer from misalignment. Human operators in such systems could "re-train" the robot by interacting with the control system. An important additional part of shared control is that the human operator and the machine must concurrently assist each other so that each other's weaknesses are augmented and each other's strengths are enhanced [68] [94]. Complex assembly processes such as car assembly lines present a system where this bidirectional assistance is necessary [28] [69].

A broad selection of shared control systems is discussed in [84]. This reference lists a number of generic supervisory functions which supervisory control schemes should allow the operator. These functions are planning, teaching, monitoring, intervening and learning. Levels of autonomy are also discussed and range from fully tele-operated to almost completely autonomous. Planning refers to the operator understanding the process and finding a way to reach the goal. The second function of teaching refers to the human teaching the computer new control strategies or task executions. Monitoring implies that the user must be able to obtain information about the system when required. The operator must be able to intervene in order to update information or assume direct control. Finally, learning implies that the user should be able to learn from the computer.

Currently most shared control systems have a clear delineation between tasks that the machine is to control and the tasks that the human is to control [84]. There is no real "sharing" of control. Cars are an example of this. The computer controls fuel injection and anti-locking brakes, but the human has control over the applying acceleration and braking. However, current
research in shared control supports the view that shared braking and steering are more effective when obstacles that threaten collision are detected [83]. In this research, both the human and the controller provide inputs to the car.

There are many examples in robotics which involve human interactions. These range from tele-robotic systems [44] [51] [95], where the operator must provide control input all the time, to systems such as health-care robots [60] [72], in which the user can issue a command and then leave the robot to execute the command. There are also cooperative robotic systems such as [3] [12] in which the user and the machine help each other in lifting an object. These systems control robots with varying degrees of autonomy. Unfortunately these systems model or integrate the human in a way that is very specific to the end application. There is a high degree of coupling between the human user (or a model of the user) and the robotic system. This makes the concepts behind these shared control systems difficult to apply to a larger variety of systems. For this reason a shared control framework, applicable to a range of systems, is needed. Such a framework will provide consistent methods for modelling and controlling shared control systems. It is the aim of this thesis to provide such a framework applicable to robotic systems utilising discrete event control theory.

1.3 Discrete Event and Shared Control

Discrete event controller design for robotic systems is reasonably straightforward in predictable environments. This design process is essentially part of the modelling and this is well understood. However, for most autonomous systems, the environment is only partially understood and there is a need for efficient and effective integration with human control. Due to some difficulties with hybrid dynamic systems and the advantages of shared control, shared control lends itself as a natural extension to hybrid dynamic systems. The combination improves on the process monitoring aspects of hybrid dynamic systems, especially in poorly modelled or dynamic environments. This monitoring is one of the five supervisory functions. The challenge of discrete event trajectory generation in changing environments can be improved by shared control as the human operator could redefine goals with ease. This falls under the category of planning, teaching or intervening with the control system.

The increasing use of hybrid dynamic systems and the existing use of these control systems in situations where human input is desirable presents an additional reason to extend hybrid dynamic systems for shared control.

One of the advantages of hybrid dynamic systems is the division into the three sub-systems. This advantage can be utilised when modelling for shared control. The human operator can
be modelled as an additional, fourth, sub-system and is easily integrated into an existing autonomous control system. The work described in [58] and [59] utilises Dual Petri Nets in a similar approach of modelling two separate controllers controlling one target. The uniqueness of the approach presented in this thesis and the work in [58] and [59] is the discrete event foundation on which it is based.

Proposed in this thesis is the definition of a fourth discrete sub-system, the Human Discrete Event Model (HDEM). This fourth sub-system is modelled according to information flow and human interaction rather than the dynamics of the human. Human factors including erroneous behaviour [75], decision making and mental workload and performance issues [24] [43] [68] [79] are important elements of shared control and human interface design. However, the proposed HDEM concentrates on information flow and therefore these factors are not modelled in the HDEM. It is therefore possible to concentrate the development of the HDEM on control system design as opposed to what type of user interface is best or how the long an operator can use a joystick.

Discrete event modelling allows potentially complex systems to be modelled in an effective and straightforward manner. Therefore, the discrete event modelling approach is used for modelling human interactions which can be quite complex. The separation of the human interaction model from the robotic control system is also important to this approach. The separation of the autonomous controller, the human interactions and the plant allows these sub-systems to be designed and analysed separately and their control commands combined as appropriate.

The combination of the three above sub-systems with an interface as the fourth component gives rise to the hybrid dynamic framework for shared control presented in this thesis. This structured framework allows for the robot to act autonomously without operator interaction. When the operator wants to interact with the control system he must be able to do so seamlessly without interrupting the "control flow".

In some shared control systems the user may need to be able to modify the path of a robot. This would be an interaction continuous in time as robot control commands are generally continuous in time. Similarly, in a hybrid dynamic system, process monitoring errors may occur i.e. the recognition of a false discrete event. For an operator to correct this a discrete interaction is needed. Therefore, the shared control framework must allow the user to interact on all levels of the system, continuous as well as discrete.

The control system must also be able to generate one continuous control command from the discrete controller command issued by the ADEC and the human input. This generation of a continuous control command is referred to as control synthesis. To assist the human operator
the control system may also need to limit or guide human input. If this guiding or limiting is incorporated into the control synthesis, the user could then follow particular trajectories quickly and accurately. It has been shown in [76] that hard virtual walls which constrain motion can improve performance in tele-operation tasks such as remote peg-in-hole insertion. This is because these virtual surfaces “guide” the operator to a goal. The limiting of the user input can aid the intervening and learning by the supervisor.

Two control synthesis methods are investigated in this thesis. The first method utilises constraints previously used for autonomous assembly [62] [65]. These constraints are also used in the guidance of human input. The second method of control synthesis discussed is based on potential fields. Potential fields provide a tool by which a robot can be guided autonomously towards a goal as well as allowing certain types of human inputs. Khatib [52] promoted potential fields for motion planning and obstacle avoidance. Since then potential fields have been applied to similar purposes for both mobile robot and multi-link robot path planning and obstacle avoidance [52] [90] [97]. I wish to exploit these motion planning capabilities of potential fields. However, potential fields have problems with creating spurious local minima other than the goal. This problem has been dealt with by harmonic potential functions, discussed in [30] [54]. These harmonic potential functions can be incorporated without much added complexity. Potential functions are also limited by the shapes that can be represented as well as how closely the potential field can envelope a real object. Superquadratic potential functions, discussed in [53], alleviate this to some degree. In shared control, the human operator can overcome these limitations of potential fields as well as performing the tasks outlined above.

This thesis presents an approach to shared control which utilises discrete event control theory. A hybrid dynamic systems framework is presented which integrates shared control commands through an additional sub-system. This sub-system models human interactions as an automaton. This method expands on the modelling capabilities of hybrid dynamic systems. Experiments and a case study utilising the methods discussed in this thesis demonstrate the applicability of this framework to shared control. The experiments also provide examples that show that shared control is more effective in controlling a particular task than autonomous control or human control alone.

1.4 Contributions

This thesis makes contributions to the advances in robotic control in the following areas:
1.5 ORGANISATION OF THE THESIS

i. A shared control framework is formulated using hybrid dynamic systems theory. Therefore advanced human decision making is integrated into an otherwise autonomous control system. The application of discrete event control theory in shared control is new contribution to the area of hybrid dynamic systems.

ii. Human interactions are modelled as an automaton. This automaton is integrated into the framework through the interface, a sub-system already present in hybrid dynamic systems. The discrete event formalism is especially useful as it makes modelling of complex systems such as human interactions straightforward.

iii. Two methods for the control synthesis are provided. Neither of these methods has been applied to shared control. The first of these methods is based on active and inactive constraints to calculate a velocity command. The second method utilises potential fields to determine a velocity command for the robot. An additional contribution is made because potential fields have not previously been employed in conjunction with discrete event control theory.

iv. The second method of integrating continuous commands from the user utilises potential fields to achieve the same result. Potential fields are new in the application to discrete event control systems.

v. Hierarchical discrete event control systems have been recognised to offer advantages over "single level" models. These advantages include greater ease of modelling more complex systems by allowing easier distinctions between events. The definition of sub-states also provided a means to more easily define control flows.

vi. A case study is provided to test the theories presented in this thesis.

1.5 Organisation of the Thesis

This thesis contains both theoretical results backed by experiments and a case study which further demonstrates the effectiveness of the proposed theories. The thesis comprises the following chapters:

Chapter 1, establishes the context of the research through a survey of literature available in the area of this thesis. The chapter contains background on both discrete event control as well as shared control. Some literature on other methods utilising discrete event control theory applied to shared control is also included.
1.5. ORGANISATION OF THE THESIS

Chapter 2 presents the hybrid dynamic control framework. In this chapter each of the four sub-systems of the framework are presented in detail. Also introduced in this chapter are details on the hierarchical model of the autonomous discrete event controller as well as the hierarchical model of the human interaction model.

Chapter 3 introduces two methods used to integrate control commands from the autonomous discrete event controller with commands issued by the human. These methods are based on constraints and potential fields respectively. This chapter includes experiments demonstrating the operation of the hybrid dynamic framework as well as the operation of the two methods of command integration. In this chapter the constraint method is also expanded to include constraints based on velocity and sensor confidence.

Chapter 4 presents the case study of the shared control system applied to the "The Robotic Cane", an assistive device for the visually impaired. The case study includes sections on the application of the framework, the control synthesis using constraints and experiments demonstrating the operation of the framework, the constraints and the cane as an assistive device for the blind.

Chapter 5 reflects performance issues associated with the shared control framework. The performance evaluation includes a comparison of shared control to autonomous control or human control alone as well as the operator's reaction to shared control.

Chapter 6 brings the conclusion to the thesis. Open problems and scope for further research is discussed.

Appendix A presents the experimental setup used in the experiments which demonstrate the operation of the framework, the constraints and the potential fields. An overview of the computational hardware and software is included.
Chapter 2

Modelling for Shared Control

2.1 The Discrete Event Formalism

The shared control framework presented in this thesis is based on hybrid dynamic systems which incorporate discrete event control theory. Ramadge and Wonham [73] define a discrete event system as a dynamic system that evolves in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of physical events. A discrete event controller models a physical system by a set of states. The controller can only have one active state at a given time and the state can change only at the occurrence of a discrete event. Part of the hybrid dynamic system which incorporates the discrete event controller must extract from a physical system the occurrence of an event. Based on the occurrence of that event the controller changes state and then issues a new control command.

Consider the control of a simple electric room heater. The discrete event model of this heater has two states, Not Warm Enough and Too Warm. Consider the initial state to be Not Warm Enough and the heater is therefore turned on. The temperature rises and this is sensed by a sensor. As soon as the temperature reaches a preset threshold (perhaps a thermostat set by the user) an event occurs. The event can be defined as It is now warm enough. The controller subsequently changes its state to Too Warm and issues a control command to turn the heating element off. The temperature will fall and another event will occur. This will cause the controller to return to the Not Warm Enough state and the heater will turn on again. This is a very simple example of discrete event control. It should be noted that the control command issued is of a discrete nature as it turns the heater either ON or OFF. In a more complex example, such as a heater with an adjustable power setting, the discrete control command issued by the discrete event controller may need to be converted into a continuous command.
for the more advanced heater to execute it. This is merely an extra level of complexity and does not change the principle of discrete event control.

2.2 The Shared Control Framework

Consider a conventional hybrid dynamic system which comprises of three sub-systems [42] [86]. The first of these sub-systems is an autonomous discrete event controller (ADEC). The second sub-system is a continuous plant, which performs the physical task. Finally, an interface is employed to enable communication between the discrete and continuous domain. The human operator can be modelled as an additional, fourth, sub-system and is easily integrated into an existing autonomous control system. Therefore the integration is achieved by combining two discrete event models, the first of human interactions and the second the ADEC.

The combined system is shown in Figure 2-1 and consists of four separate subsystems, namely the Human Discrete Event Model (HDEM), the Autonomous Discrete Event Controller (ADEC), an Interface and the Continuous Time System (CTS).

A similar approach which models two separate controllers, one for the human aspects and one for the machine, is presented in [58] and [59]. The controllers in these references utilise Dual Petri Nets. In this system however, there is no interface through which the sub-systems are combined. This makes the design and the analysis of the individual systems more complex.

The HDEM models the types of interactions a human can make using an automaton. Let us therefore consider the actions that a human user is likely to make. In a robotic system these interactions are likely to fall into one of four categories:

![Figure 2-1: Block diagram of a hybrid dynamic system with human integration](image-url)
1. The human may want to observe the robot, monitoring for errors or ensuring that the work space remains clear, therefore not interacting.

2. To interact with the system and affect operation, the supervisor has two options. The first option is to interact by issuing a command continuous in time. This can be a command to accelerate the robot or make it follow a specific path.

3. The second mode of interaction can be via a command discrete in time, such as an emergency stop or a command to turn left, etc.

4. In order to understand the system better the supervisor may require more information about the system (perhaps to review sensor data to aid in the monitoring of the robot).

Based on these four categories of possible interactions, the HDEM model is defined by an automaton with four states. The classes of interactions that are modelled are therefore:

1. No Interaction
2. Continuous Input
3. Discrete Input and
4. Information Request.

The ADEC controls the autonomous part of process and is also modelled by an automaton. By autonomous we refer to a process that would be operating in order to complete a desired goal or set of goals without human interaction. The ADEC generates control commands for the continuous system based on past and desired events. The CTS is the physical structure performing the task and its associated continuous time control system. Mathematically, the continuous plant will be defined by a set of differential equations describing the task. Lastly, the Interface provides a means of communication among the CTS, the HDEM and ADEC. Detailed descriptions of each of these sub-systems are given below.

### 2.3 The Plant

The plant is the structure which physically performs the task. In robotics this typically involves a several degree of freedom manipulator. The plant also governs the dynamics of controlled systems and hence a set of equations which describe its motion. Additionally, the plant also
includes a continuous (in time) control system. In robotics this continuous control system is often a type of position or velocity controller.

Consider a robotic system which is described by a state vector, \( \mathbf{x}(t) \), which describes the system in terms of position \( q(t) \), and velocity, \( \dot{q}(t) \). We can then write the state variable as

\[
\mathbf{x}(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}
\]  

(2.1)

The equation of motion of the robot in free space can then generally be written as

\[
\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), u(t))
\]  

(2.2)

It is from the continuous state variable, \( \mathbf{x}(t) \), that the occurrence of events must be recognised and the correct event determined. The detection of the correct event is important as the state of the ADEC depends on these transitions to accurately control the system.

In the example of the previously introduced electric heater, the continuous state variable is the temperature. The input variable would be the heat produced by the heating element.

### 2.4 Autonomous Discrete Event Controller

The continuous time plant is controlled by a task-level, discrete event controller. This controller is modelled as an automaton. The automaton has two or more discrete states which describe the state of the system. The states can represent any distinct condition of the robotic system. The states can include on and off, which room of a house a mobile robot is in or the status of a machine on an assembly line. As previously described the transitions between the states are termed events which are recognised as such by a process monitor. On every transition between states of the ADEC a new controller command is issued. This is a controller event.

This controller event is used to control the plant. The controller events can be as simple as “turn-heater-on” or more complex such as “go-to-room-'B'” in the case of a mobile robot.

Mathematically, the automaton is a quintuple, \( (S^A, E^A, C^A, \alpha^A, \beta^A) \), where \( S^A \) is the finite set of discrete states, \( E^A \) the set of events caused by the plant, \( C^A \) the set of controller events, \( \alpha^A : S^A \times E^A \to S^A \) is the state transition function, and \( \beta^A : S^A \to C^A \) is the output function. Superscript \( A \) denotes elements associated with the autonomous system. Each discrete state \( \gamma(k) \in S^A \) is defined to be a particular range of positions in the workspace of the robot. This includes areas occupied by obstacles, target areas and workspace boundaries. Each controller event \( \nu(k) \in C^A \) is generated by the discrete event controller. Controller events are discrete.
commands issued such that, based on the current state and previous events, the next desired event will occur. For example, to reach a particular goal (an event or state transition must occur) the controller event will select appropriate conditions so that the goal is reached. Plant events $\tau(k) \in E^A$ are generated by the plant but must be recognised by the process monitor.

One example of when a plant event occurs is when the robot enters a new area in the workspace. The index $k$ specifies the order of the discrete states or plant events. The dynamics of the discrete event controller are given by

$$\gamma(k+1) = \alpha^A(\gamma(k), \tau(k))$$

$$\nu(k) = \beta^A(\gamma(k))$$

Note that the state of the continuous system is $x$, whereas $\gamma$ is the discrete state variable of the discrete event controller and is dependent on $x$.

### 2.5 Human Discrete Event Model

Consider a robotic system designed to interact with a human. The user is assumed to have knowledge about the function of the system, the objective and the constraints of the system. It is now possible to define a variety of possible operator interactions. The different ways can be broken into the formerly mentioned classes of *No Interaction*, where the human allows the autonomous system to perform the desired task; *Continuous Interaction*, where the human can input continuous commands such as velocity and acceleration; *Discrete Interaction* in which state the human can issue discrete commands such as a key press to recalibrate, and finally; *Information Request* where the human can request more information about the system such as position data, force data, current state, etc.

Humans find it difficult to do more than one thing at one time reliably. This is particularly so if what they are doing relies on monitoring a dynamic system as is usually the case in robotics. Accident statistics relating to the use of mobile phones and driving [31]. Therefore we make the assumption that the operator should only perform one of the above actions at any given time. The exclusiveness of each of the interactions allows the modelling of the above classes as individual interaction states. A user interaction may occur at random times thus making the system asynchronous. It follows that a method of modelling asynchronous state based systems is desirable for the modelling of the human operator interactions. The event-driven, asynchronous nature of discrete event controllers [96] as well as the capability to model complex systems concisely makes discrete event theory desirable. The discrete event model of the HDEM with its four states, defined as $\delta(n)$, is shown in Figure 2-2. The descriptions of the HDEM
events, $\kappa$, are listed in Table 2.1. Note that there is no transition between the Information Request state and the two input states. The state space is therefore not fully connected. The lack of the connection comes about for the following reason. If an operator issues an information request in order to make a decision on what to do next there will be a finite time in which the operator evaluates the data. During this interval the operator is not interacting, merely monitoring the data and therefore the system will return to the No Interaction state. Once the data is evaluated he can then issue another information request or issue a command, either continuous or discrete. Therefore there is no need for direct connections between the Information Request state and the two input states. There is a need for a connection between the discrete and continuous input states. For example, a user may want to issue an emergency stop immediately after issuing a continuous command, i.e. a discrete command immediately after a continuous command.

![Diagram of HDEM](image)

**Figure 2-2: The Human Discrete Event Model**

<table>
<thead>
<tr>
<th>HDEM Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_1$</td>
<td>Information request made</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>Information request finished</td>
</tr>
<tr>
<td>$\kappa_3$</td>
<td>Continuous input made after no intervention</td>
</tr>
<tr>
<td>$\kappa_4$</td>
<td>Continuous input finished</td>
</tr>
<tr>
<td>$\kappa_5$</td>
<td>Discrete input made after no intervention</td>
</tr>
<tr>
<td>$\kappa_6$</td>
<td>Discrete input finished</td>
</tr>
<tr>
<td>$\kappa_7$</td>
<td>Discrete input made following continuous input</td>
</tr>
<tr>
<td>$\kappa_8$</td>
<td>Continuous input made following discrete input</td>
</tr>
</tbody>
</table>

**Table 2.1: HDEM events**

In a manner similar to the ADEC, the HDEM is modelled by an automaton which describes
the discrete states and the transitions, shown in Figure 2-2. The automaton is a quintuple \((S^H, E^H, C^H, \alpha^H, \beta^H)\), where \(S^H\) is the finite set of discrete human states, \(E^H\) the set of events caused by the human (and sometimes triggered by the plant), \(C^H\) the set of controller events, \(\alpha^H : S^H \times E^H \rightarrow S^H\) defines the state transition function and \(\beta^H : S^H \rightarrow C^H\) is the output function. The superscript \(H\) denotes elements associated with the human system. Each discrete state \(\delta(n) \in S^H\) is defined as one of the four possible human interactions. The index \(n\) specifies the order of the discrete states or events in the human system. Index \(n\) is similar to \(k\) in the ADEC. However, \(n\) marks times of events in the HDEM. Generally \(n\) and \(k\) are independent. However, at some time instances changes in both \(n\) and \(k\) may be synchronised due to the specific event which occurred. Each event \(\kappa(n) \in E^H\) is generated by a plant event \(\kappa^A(n)\) or an event caused by human input, \(\kappa^H(n)\) according to
\[
\kappa(n) = \kappa^A(n) + \kappa^H(n)
\]
where the event \(\kappa^H\) is derived from human input \(H\)
\[
\kappa^H(n) = \chi^H(H)
\]
where \(\chi^H\) is a function which maps human input to HDEM events. Note that \(H\) can be any human input, continuous, discrete or an information request. \(\kappa^A(k)\) in (2.5) is a HDEM event caused by changes in the plant and is derived by a mapping, \(\psi^H\), from plant events.
\[
\kappa^A(n) = \psi^H(x(t))
\]
Controller event \(\eta(n) \in C^H\) is generated by the HDEM. These controller events describe which HDEM events are enabled and which are disabled and can therefore restrict certain interactions. The dynamics of the human discrete event model are given by
\[
\delta(n + 1) = \alpha^H(\delta(n), \kappa(n))
\]
\[
\eta(n) = \beta^H(\delta(n))
\]
From human interaction through the HDEM, a continuous command, \(h(t)\), and the discrete command, \(r^H(n)\), are defined and “routed” to the Interface.

2.6 Interface

The fourth component, the Interface, provides a means of communication between the HDEM, ADEC and the plant. These cannot communicate directly because the plant is of a continuous nature and the other two systems are of a discrete nature. Due to the plant, ADEC and HDEM
communicating through the Interface, it is possible to interact with all aspects of the system via this one sub-module. This one point of interaction makes the integration of the HDEM straightforward.

To demonstrate the interactions within the Interface, the Interface of Figure 2-1 has been enlarged and is shown in Figure 2-3.

The Interface is bidirectional. In the ‘downward’ direction, the Interface combines the continuous command from the human with the command of the autonomous controller to generate a combined continuous command for the plant. In Figure 2-3 this combining function is indicated by $\Omega$. In the reverse direction, the Interface extracts from the plant state and human input, events for the ADEC and HDEM. This is shown on the figure by the maps $\psi^A$ and $\psi^H$ which act as a process monitor.

$\phi^A$, maps controller events of the ADEC to continuous plant inputs according to

$$u^A(t) = \phi^A(\nu(k)) \quad t_k \leq t \leq t_{k+1} \quad (2.10)$$

where $\nu(k)$ is the most recent controller event before time $t_k$. With respect to implementation it may be easier to combine equations (2.4) and (2.10) because the command is then issued based solely on the discrete state. Hence we obtain

$$u^A(t) = \phi^A(\beta^A(\gamma(k))) \quad t_k \leq t \leq t_{k+1} \quad (2.11)$$

It is important to note that the human can modify the properties of the $\phi^A$ mapping. This ability is important when changes need to be made to the continuous command generation.
Combining the continuous command from the ADEC (2.11) and the continuous input from the human \((h(t))\), a single control command can be issued.

\[
u(t) = \Omega(u^A(t), u^H(t)) \quad t_k \leq t \leq t_{k+1}
\]  

(2.12)

where \(u(t)\) is the combined continuous input vector to the plant and \(\Omega\) is the combining function.

The method by which this command is limited is described in Section 3.1.2.

The map \(\psi^A\) converts the continuous state space of the plant combined with discrete input from the human, \(\tau^H(n)\) into plant events, \(\tau(k)\).

\[
\tau(k) = \psi^A(x(t), \tau^H(n))
\]  

(2.13)

This equation forms the input to the state transition equation (2.3). Note that equation (2.13) does not imply that \(\tau(k)\) changes continuously as \(x(t)\) changes. The map \(\psi^A\) generates a new event when a new discrete state is entered or when a discrete human input initiates such an event.

### 2.7 Hierarchical Modelling of the HDEM

The structure of the HDEM with its four states has one major limitation. The HDEM does not resolve precisely enough how the controller should interact with the Interface and hence the ADEC. The four states of the HDEM serve well to define the classes of interactions. The four states also define the user interface well. The user can only interact in one way at one time. This makes the HDEM model a good tool for the design of multi-function control panels as user interfaces.

However, the current model lacks sufficient detail to model a wider range of interactions and the associated information flows available between the HDEM, the Interface and the ADEC. For example, if a system requires two types of discrete interactions such as a process monitoring correction and a goal redefinition then a single Discrete Input state is no longer adequate. The process monitoring correction requires interaction with mapping from the continuous domain into the discrete domain. The goal redefinition requires a change of the goal state in the ADEC. Although both interactions are discrete they affect different parts of the system and therefore require different control flows. Therefore an expansion of the HDEM is required.

Because the original structure of the HDEM is clear cut in its definition of interaction types, a hierarchical model is utilised to expand the HDEM. This hierarchical structure allows definitions of sub-states. These sub-states form subsets of the four previously defined HDEM states. Therefore the structure of the HDEM remains the same at the top level. This also implies that
2.7. HIERARCHICAL MODELLING OF THE HDEM

a user interface would potentially only have four separate areas each corresponding to a type of interaction (only three would actually be required as nothing is done in the No Interaction state). If a bigger model was to used at the top level then it is quite likely that the user interface would be larger or would not have its interactions as clearly defined.

Hierarchical discrete event control is discussed in [98]. An example of how hierarchical discrete event theory can be used to expand the HDEM into a system is shown in Figure 2-4. The figure shows a HDEM with three continuous interaction sub-states (δ2.1, δ2.2 and δ2.3) and two discrete interaction sub-states (δ3.1 and δ3.2). If the HDEM is in its No Interaction state, the sub-models are inactive. If either a continuous or discrete interaction is made, the respective sub-model becomes active indicated by an event (the thick line on the figure) and one of the sub-states is entered. With this structure it is possible to traverse the entire state space as defined in the top level HDEM. For example if a human input were to take the HDEM from state δ2.1 to state δ2.2 then the continuous state sub-system becomes de-activated as soon as state δ2 is left. This de-activation coincides with the activation of the Discrete Input sub-system and state δ3.2 is immediately entered. The activation and de-activation events of the sub-systems are indicated in the figure by the think lines.

Similar to the higher level HDEM, the low-level HDEM sub-models are modelled by automata. There potentially exists an automaton for each of the three interaction states, namely Continuous Interaction, Discrete Interaction and Information Request. Each of the three automata are
quintuples of the form \((S^H_{lo}, E^H_{lo}, C^H_{lo}, \alpha^H_{lo}, \beta^H_{lo})\), where \(S^H_{lo}\) are finite sets of the sub-states, \(E^H_{lo}\) the events, \(C^H_{lo}\) the set of controller events. \(\alpha^H_{lo}: S^H_{lo} \times E^H_{lo} \rightarrow S^H_{lo}\) defines the state transition function and \(\beta^H_{lo}: S^H_{lo} \rightarrow C^H_{lo}\) is the output function. The subscript \(lo\) denotes elements associated with the lower sub-systems and is defined as

- \(lo = CI\) if associated with the Continuous Interaction sub-system.
- \(lo = DI\) if associated with the Discrete Interaction sub-system.
- \(lo = IR\) if associated with the Information Request sub-system.

Each discrete state \(\delta(m)_{lo} \in S^H_{lo}\) is defined as one of the lower level states. The index \(m\) specifies the order of the discrete events in the sub-system. The events \(\kappa_{lo}(n) \in E^H_{lo}\) are generated by a the plant or by human input, as described in Section 2.5 for the HDEM.

\[
\kappa_{lo}(n) = \kappa^A(n) + \kappa^H(n)
\]

where \(\kappa^H(n)\) is derived from (2.6) and \(\kappa^A(n)\) is derived (2.7). Controller events \(\eta_{lo}(n) \in C^H_{lo}\) are generated by the HDEM sub-system.

The design of a sub-model is dependent on the required user interactions. If there were only one type of discrete interaction, one type of continuous interaction and one type of information request then there would be no requirement for sub-models. If however there is more than one type of each of the above categories, then a sub-model is defined according to the additional interactions required. For example, assume a small electric heater. It has an on/off switch. This is one discrete interaction type. The heater also has a continuous power control which determines the temperature of the heating element. This power control is one continuous interaction type. If it also has a continuous fan-speed control a second continuous interaction state is required. The heater also has a temperature readout button. This forms the information request. This heater would have the HDEM model with four states and a continuous state sub-model with an additional two states. The four states of the upper level HDEM are a Continuous Interaction State, a Discrete Interaction State, an Information Request State and a No Interaction State. The two states of the Continuous Interaction sub-model are Power Control and Speed Control.

It is important to note that the lower level component of the HDEM is a sub-model. This sub-model operates as a separate discrete event model (when activated) but is linked with the upper-level HDEM by particular events. Some events only trigger state changes in the sub-model; other events trigger state changes in both the lower level model and the high level HDEM. Consider the example of the heater. If the heater had been running by itself then
the top-level HDEM would be in the No Interaction state. If the fan speed was now adjusted, followed by an adjustment of the power control, the upper level HDEM would have changed state to Continuous Input (only one transition). The lower level HDEM would have become active and entered the Speed Control States followed by a transition (an event) to Power Control (two transitions). If the heater were now turned off, only one transition occurs in the upper HDEM (Continuous Interaction to Discrete Interaction). In the lower HDEM there are two transitions, the first de-activating the Continuous Interaction sub-model and the second activating the Discrete Interaction sub-model.

This connection between the lower and the upper model is defined by a mapping, $\Pi_{lohi}$, which is defined as

$$\Pi_{lohi} : \kappa_{lo}^H \rightarrow \kappa^H$$

This mapping enables communication between the lower and higher level and synchronises the two levels. This mapping is established by determining which low level events cause what changes in the high level system. For example, a transition between $\delta_{2.1}$ and $\delta_{2.2}$ would not cause an event in the upper level model, whereas a transition from $\delta_{2.1}$ to $\delta_{3.1}$ would.

Not only does the hierarchical control structure provide a consistent means for defining multiple user interactions of the same type but it extends this consistency to the user interface as well. Consider once again the heater. Its user interface could consist of an on/off switch, a Display Temperature button, one dial for both types of continuous input and perhaps a selector to select which type of input was to be made. If additional types of continuous input need to be added they could all be interfaced through the one dial. This is similar to the multiple functions of a computer mouse.

This hierarchical structure is implemented in the experiments of Section 3.4 and Section 4.2. In both these sections the system to be controlled requires more than one type of continuous or discrete interaction. In the first of these two sections two types of continuous input are utilised and in the second, four different types of discrete input are used.

### 2.8 Conclusion

A technique for integrating a human interaction model into discrete event control has been developed. The method proposed is based on hybrid dynamic systems theory that originally had three sub-systems. A model of human interactions was included to allow for shared control utilising discrete event theory. The system operates by combining discrete control commands from the human model with those of the autonomous controller, converting these to continuous
commands in the Interface and then issuing a combined command to the plant. In the feedback loop, events from the plant are extracted for both the HDEM and the ADEC.

This approach is applicable to a wide variety of control systems provided that the autonomous portion of the system can be modelled using discrete event theory. It must also be possible to derive the events which cause state changes in the HDEM.

Additionally, a hierarchical structure was presented which allows a large variety of human interactions under the categories of Continuous Input, Discrete Input, Information Request and No Interaction. The benefit of using this hierarchical structure is that the expansion to allow more interactions is still consistent and does not cause the HDEM to have a vast number of states. This can also be advantageous for user interface design.
Chapter 3

Control Synthesis and Human Interaction

In the previous chapter the control framework was presented. The autonomous discrete event controller and the plant were defined. For the ADEC to control the robotic system, the ADEC issues discrete control commands which are combined with human input to yield a combined controller command. The control commands issued by the ADEC are such that the system will eventually reach its final state. Generally, these discrete control commands must be converted into continuous commands for the robot to be able to execute them. This conversion from the discrete to the continuous together with human interaction is the subject of this chapter and is referred to as control synthesis.

Equations (2.10), (2.11) and (2.12) define the process of control synthesis. Essentially control synthesis is a mapping from the discrete commands, $\nu(k)$, to the continuous control variable, $u(t)$, which includes human input. The issue of control synthesis can be difficult to solve because the discrete command may be as simple as go to point 'B'. In the continuous domain this may involve avoiding obstacles or following a specific path or profile. The problem is compounded if the discrete event model of the process is very abstract or does not model all the details of the task to be performed.

Two methods for control synthesis are presented. Although both methods are appropriate for command generation, particularly in shared control, each of the methods has its advantages and disadvantages which warrant the discussion of both methods.

The first is based on active and inactive constraints to control the robot so that it progresses from one state to the next. The constraint method lends itself to discrete event systems because,
3.1. POSITION CONSTRAINTS

depending on what state is currently active, a particular set of constraints is also active or inactive. Solving the active constraints for a solution therefore becomes straightforward.

The constraint method has its origins in position control. Therefore, initially it is demonstrated how we can utilise position based constraints in controlling a pick and place type of operation. It is shown in this chapter how control commands issued by the user are integrated with this method. It is also demonstrated how the constraints can be used to limit the effect of human input as well as guide human input.

The second method is based on potential fields, a method that has proven itself in many mobile robot applications. Potential fields were chosen as they lend themselves to being easily modified and defined mathematically. The ease of modification is particularly important for a shared control system as this way modifications can be made by the user on-line. A straightforward mathematical definition is also important because for every new discrete controller command a new set of potential fields may need to be defined.

The control synthesis utilising potential fields is then demonstrated. Although potential fields are not new, it is the first instance in which these have been combined with discrete event control theory. The method is demonstrated through experiments in the same pick and place type task as above. It is also shown how potential fields can have a limiting effect on the human input.

Lastly, a comparison is made between the position constraint method and the potential field method. The two methods are first compared based on how easily they are modified by the user. Then, a comparison is made based on calculation time required and the smoothness of the path generated by the two methods.

3.1 Position Constraints

The first method used to define the map \( \phi^A \), which converts commands from the discrete event controller into continuous commands, is based on constraints. This method is well suited to discrete event control tasks as it allows each discrete state to be dealt with separately. Additionally, this method can be used to aid humans which is an important part of human-machine interaction [24]. It has been shown in [76] that hard virtual walls which constrain motion can improve performance in tele-operation tasks such as remote peg-in-hole insertion. This is because these virtual surfaces “guide” the operator to a goal. Additionally, humans are prone to make mistakes [68] [74] [75], so the constraints can also be used to limit human input action such that the robot remains within its operating limits.
3.1. POSITION CONSTRAINTS

3.1.1 Constraint Equations

Based on the current state and next desired state, a set of active constraints can be determined. Changes in the set of active constraints occur at ADEC events. A solution, yielding a velocity command, can be calculated from a set of active constraints [65]. These constraints are based on whether we desire to increase or decrease a distance. Where the distances are measured between a point on a robot (end-effector, centre of mobile robot, a point on an elbow, etc.) and a surface or point.

Let us assume an environment, shown in Figure 3-1, which can be modelled by a set of surfaces and points. For example, surfaces can describe boundaries, whereas points can define target locations or locations of obstacles. Ignoring the robot-arm and given that the end-effector is smaller than the smallest structure in the environment, the end-effector can be modelled as a point. We can now define two sets of constraints. The first set includes point-to-surface constraints and the second, point-to-point constraints.

![Figure 3-1: Geometric definitions for constraints](image)

Attached to the environment let there be a reference frame in which we define a set of coordinates which encompasses both the end-effector and the workspace in three dimensions. These coordinates are then expressed by a vector, where \( \theta_1 \theta_2 \theta_3 \) are the Euler angles

\[
q = [x \ y \ z \ \theta_1 \ \theta_2 \ \theta_3]^T
\]  

(3.1)

Figure 3-1 also shows geometric definitions required to specify the constraints. Let us denote the position vectors from the origin to two arbitrary points \( \rho \) and \( \sigma \) by \( \mathbf{d}_\rho \) and \( \mathbf{d}_\sigma \) respectively. Let us represent a surface \( \omega \) by its normal \( \mathbf{n}_\omega \) and let a vector to any point on the surface be \( \mathbf{d}_\omega \).
The distance $h_{p\omega}$ between point $p$ and surface $\omega$ is then given by

$$h_{p\omega} = (d_p - d_\omega)^T \cdot n_\omega$$  (3.2)

The distance between two points $p$ and $\sigma$ is given by

$$h_{p\sigma} = ||d_p - d_\sigma||$$  (3.3)

These distances can now be utilised to form constraints on the robot's motion. Distances between the robot and a surface or point may increase, decrease or remain constant. For example, the distance between the robot and a goal point has to decrease to reach the goal. At the same time, the distance between a robot and an obstacle has to increase or at least remain constant. These three conditions placed on the distances form constraints on the velocity command. It is mathematically possible, from the constraints, to determine a velocity command which will maintain, increase or decrease the distance from a surface or point.

Let us consider the maintaining condition in which we maintain a constant distance from a surface or point. To derive admissible velocities that satisfy the constraints (3.2)(3.3), we differentiate:

$$\frac{d}{dq}[(d_p - d_\omega)^T \cdot n_\omega] \frac{dq}{dt} = 0$$  (3.4)

$$\frac{d}{dq} ||d_p - d_\sigma|| \frac{dq}{dt} = 0$$  (3.5)

Equations (3.4) and (3.5) describe velocity constraints which are used to calculate a velocity that allows the robot to move without violating either constraint. Both (3.4) and (3.5) can be rewritten as

$$aq = 0$$  (3.6)

where $a$ is a 1x6 row vector and in the case of point to surface constraint is given by

$$a = \frac{d}{dq}[(d_p - d_\omega)^T \cdot n_\omega]$$  (3.7)

or in the case of a point to point constraint is given by

$$a = \frac{d}{dq} ||d_p - d_\sigma||$$  (3.8)

Equation (3.6) is the maintaining condition. The maintaining condition is particularly important for surface following as it keeps the distance between the robot and the surface constant. More importantly it keeps the distance constant at zero.

In order for the robot to reach a particular surface or point, we need the distance between a point and surface or two points to decrease. The constraint for this is now written as

$$aq < 0$$  (3.9)
3.1. POSITION CONSTRAINTS

This constraint is needed so that the robot can move towards a desired goal, for example, in a pick and place task. Similarly, if it is desired that the distance between a point and surface or between two points is to increase, the constraint is now

\[ a_q \geq 0 \quad (3.10) \]

This constraint to increase the distance can be utilised to make a robot avoid an obstacle.

3.1.2 Human Interaction

When using the constraint method for generating control commands, there are two ways in which a human operator can interact with the system on a continuous level (HDEM state \( \delta_2 \)). The first method is by adding a velocity command to a velocity command already generated by the constraint method. The second method of continuous interaction is by modifying the location of the constraints. There is also a discrete interaction possible. This discrete interaction allows the user to cancel any previous continuous command.

Because there are two methods of continuous interaction the hierarchical HDEM model can be used. The hierarchical model allows two separate continuous interaction states to be defined while preserving the HDEM structure defined in Section 2.5. Without a hierarchical model it would be difficult for the controller to distinguish between the two types of continuous interaction and therefore which set of human input commands to apply.

The hierarchical HDEM model is shown in Figure 3-2.

![Hierarchical HDEM model](image)

**Figure 3-2: Hierarchical HDEM model utilised for two types of continuous interaction**

According to the above model we can now define the continuous human input parameter \( h(t) \)
for each of the sub-states.

\[ h(t) = \begin{cases} 
  u^H(t) & \text{if } \delta(n) = \delta_{2.1} \\
  \Delta d & \text{if } \delta(n) = \delta_{2.2} 
\end{cases} \quad (3.11) \]

where \( u^H(t) \) is a velocity command input by the human and \( \Delta d \) represents the change in the position of the constraint.

In the first method the velocity input by the human, \( u(t) \), is added to the velocity generated by the autonomous system. Equation (2.12), is now defined as

\[ u(t) = u^A(t) + u^H(t) \quad (3.12) \]

In certain circumstances it may become necessary to guide or limit human input. Pre-defined trajectories not usually part of autonomous operation could be followed as guides or as evasive action to avoid errors. For example, the robot should autonomously steer towards a more favourable location if, due to the user input, the robot were to exit the workspace. This autonomous evasive action has the effect that the human input is restricted.

In certain states human input is to be limited. In these states an additional constraint is added. This additional constraint can depend on one or more conditions, for example the current state of the system, current robot position or current robot velocity.

Consider an example where a robot is to operate inside a restricted workspace and its continuous control input is a velocity command. Even with human input, the robot is to remain within the workspace. The workspace is modelled in such a way that the boundary of the workspace is a state of the discrete event model. In such a case, the additional constraint is added if the current state of the system is in the boundary state (i.e. current robot position is at the boundary). The additional constraint limits the control command, in this case velocity, to values which cause the robot to remain within the workspace. Assuming that the continuous human control input is a velocity command, \( u^H(t) \) and that \( u^A(t) \) is also a velocity command then the additional constraint is mathematically expressed as

\[ (u^A + u^H)^T \cdot n_\gamma \geq 0 \quad \gamma(k) \in S_B \quad (3.13) \]

where \( S_B \) is the set of states where human input is to be limited. This additional constraint forces the robot to remain inside the workspace.

The second way in which the human can interact is by modifying (changing position) any of the constraints. This interaction occurs through HDEM state \( \delta_{2.2} \). This has the effect of directly changing the generation of \( u^A(t) \) by changing parameters in the mapping \( \phi^A \), the mapping from discrete ADEC controller events to \( u^A(t) \). This is indicated on Figure 2-3 by the arrow through the \( \phi^A \) mapping.
This modification of constraints can be useful to change part of the environment model. The user input, $h(t)$ contains information on changing the position of the constraint, see Equation (3.11). The position is modified according to

$$d = d' + \Delta d$$  \hspace{1cm} (3.14)$$

where $d$ is the new position vector of the constraint and $d'$ represents the starting position vector of the constraint. By changing the position of the constraint, the velocity command for the continuous system is also modified.

### 3.1.3 Constraints and the Physical World

The constraints can be used to model environments in which the robot can operate. Surface constraints can describe boundaries and obstacles. Points can define target locations and locations of obstacles. In modelling real-world environments practical issues need to be addressed. First it must be decided what objects are to be modelled using the constraints. The basic shapes are either a cylinder or a flat surface. More complex obstacles which cannot be modelled by one constraint alone must be modelled by a set of constraints. The third practical issue associated with constraints is the activation of a new set of constraints when states change.

The constraint equations (3.2) (3.3) are based on distances between two points or a point and a surface. As such, objects are restricted to be modelled either by points or surfaces. The set of constraints considered is therefore not complete. Non-flat surfaces can be described with more complex constraint equations, allowing more complex shapes to be modelled. However, more complex obstacles can be modelled by superpositioning a set of surfaces, set of points or both, similar to the work in [52]. Therefore complex objects can be modelled by taking finer and finer representations of the environment. It is possible however to define too many constraints for a system and in such a case it may not be possible to solve the inequalities for a solution. In such instances it becomes necessary to carefully consider which constraints are active at a given time or reduce the number of constraints. An example of this type of constraint conflict is shown in Figure 3-3. As can be seen the goal point is behind the round obstacle, modelled by a point to point constraint which disallows the distance between the robot and the point to reduce. To approach the goal, there exists a constraint which tries to reduce the distance between the robot and the goal. The robot moves around the obstacle towards the goal. However, towards the end of the path, the distance towards the obstacle must reduce in order to reach the goal. This reduction is not allowed and therefore a constraint conflict exists.

Another problem exists in the following situation. Assume that a robot commences its motion in a concave portion of an obstacle (such as an L-shape) and the obstacle is modelled as a
3.1. POSITION CONSTRAINTS

Figure 3-3: Example of a constraint conflict

3.1.4 Velocity Generation

Following the issuing of the discrete controller command the continuous control command must be determined to control the robot. We consider velocity commands and hence assume that any continuous human input can be interpreted as a velocity command. To calculate the command velocity (2.12), the human input velocity, $u^H(t)$, must be determined as well as the autonomous velocity, $u^A(t)$. Let us assume that $u^H(t)$ is directly input by the human. We therefore need to solve for the autonomous velocity command.
3.2. EXPERIMENTS USING CONSTRAINTS

In any one state of the discrete event controller a set of constraints is active in order to reach the next state. This set is composed of any number of constraints of the form of (3.6) (3.9) (3.10). The problem is thus reduced to solving a set of inequalities for \( \dot{q} \). The result yields the autonomous velocity. We use the optimal velocity solution [65] method. A problem inherent to solving inequality constraints is that the solution may not be unique. A single solution is found by selecting a function which optimises reliability and robustness of the solution. The function maximises the minimum distance to each constraint and is defined by

\[
J = \max_{\dot{q}} \left[ \min_{j=1}^{m} a_j \dot{q} \right]
\]

(3.15)

where \( m \) is the number of constraints, \( a_j \) are the individual constraints which are currently active and \( a_j \dot{q} \) is the distance between the velocity command given by \( \dot{q} \) and constraint \( a_j \). Equation (3.15) is illustrated in Figure 3-5 in which two constraints are solved, \( a_1 \) and \( a_2 \). The solution is the point which is farthest from both of the inequality constraints. Note that all the constraint inequalities run through the origin and the solution is thus unbounded. As magnitude can be scaled at a later stage, an upper bound is implemented using

\[
\|\dot{q}\| = 1
\]

(3.16)

3.2 Experiments Using Constraints

To test the proposed control scheme which integrates a human into an autonomous system, a robotic manoeuvering task with interference from a fixed as well as a moving obstacle was im-
3.2. EXPERIMENTS USING CONSTRAINTS

implemented. The task involves moving the robot between two target areas, Target A and Target B while steering around the fixed obstacle. During the movement the workspace boundaries must not be crossed and the moving obstacle, a model train, must also be avoided. Because the dynamics of the train are unknown the train is not modelled using the constraints. Additionally, no form of obstacle sensing is employed. It is therefore the task of the human to recognise potential collisions between the robot and the moving obstacle and then to take action to avoid the collision.

This task was chosen because it contains elements representative of a variety tasks often encountered in robotics. The ability of a robot to reach a goal from a starting position is fundamental to a great deal of robotic tasks. These tasks include pick and place operations, mobile robot navigation and whole-arm manipulation tasks. In these tasks, while travelling to reach a goal, obstacles must often be avoided or approached. These obstacles can be walls, a second component (assembly) or other robots. Obstacle avoidance of fixed obstacles and boundaries as discussed in [52], [54], [55] and [71] presents problems not only in sensing but also in how to choose a path around the obstacles while still reaching the target. The example chosen for our experiments includes a fixed and an unpredictable moving obstacle as representations of these problems. Obstacle avoidance with unpredictable moving obstacles [34] [35] suffers from the lack of sensory information about the dynamics of the other obstacles. In our example we also have a lack of sensors as the moving obstacle cannot be detected by means other than the human operator.

Figure 3-6 shows the experimental setup. A 5-degree of freedom Eshed Scorbot VII performed the autonomous task. A cardboard cylinder provided a fixed obstacle which can be seen at the right-hand-side of the figure. A model train was utilised as the moving obstacle. It can be seen between the fixed obstacle and the robot end-effector. Further details of the setup can be found in Appendix A.3.

A description of the modelled workspace is shown in Figure 3-7(a). This figure also shows how the discrete states of the ADEC relate to the physical layout of the system. The ADEC automaton (nine discrete states) is shown in Figure 3-7(b). The descriptions of the ADEC states are listed in Table 3.1. The boundaries form surface constraints on the system.

Three sets of experiments were conducted to test the control theory utilising constraints. The first set tests the operation of the ADEC and constraints without any human interaction. The second set tests operation of the ADEC with human interaction. Finally, in the third set, constrained human interaction is tested.

In all three sets of experiments, the position of the robot is monitored (using the continuous
3.2. EXPERIMENTS USING CONSTRAINTS

Figure 3-6: Experimental setup to test control using constraints in an obstacle avoidance task using a 5-degree of freedom Eshed Scorbot VII

State vector $x(t)$ and is compared to a known environment model. If the robot is in an area contrary to the current state of the ADEC an event, $\tau(k)$, has occurred. Examples of events include the robot entering a target area, the robot entering free space or the robot reaching a workspace limit. On recognition of an event the new state is determined and a new autonomous velocity calculated using an algorithm implementing Equation (3.15).

If there was no continuous human input, the new continuous control only consists of the autonomous velocity command. In the case where there is human input, it is integrated into the control command according to Equations 3.11 and 3.12.

When the human input is a velocity command (HDEM state $\delta_{2,1}$), the velocity in (3.11) is calculated according to

$$u^H(t) = \int_{t(n)}^{t(n+1)} \tilde{h} \, dt$$

(3.17)

where $\tilde{h}$ is an acceleration which is commanded by the human. The limits of $t(n)$ and $t(n+1)$ on the integral represent the times of human input commencement (i.e. entering the Continuous Velocity Input state, $\delta_{2,1}$ of the HDEM) and cessation of human input.

An acceleration input was chosen as opposed to a velocity input as it reduces the amount of time the human controls the robot. The time reduction comes about since a velocity can be set (accelerated to) after which the system only needs to be observed until a velocity reduction is required. The acceleration input also allows the use of an "on-off" type joystick to achieve a range of input velocities which would otherwise require a proportional joystick.
The joystick allows for accelerations in the x, y and z directions to be input. The joystick is either on or off in each direction. When the joystick is pushed into the 'on' position an acceleration builds at a predefined rate to a desired velocity. Once returned into the 'off' position, the acceleration returns to zero at a much faster predefined rate. To obtain the velocity command, the acceleration is integrated. This integration time starts when the joystick is first 'on', and stops when the acceleration returns to zero. Also, an upper limit is imposed on the velocity in order to ensure safe operation. The limit is implemented by setting the acceleration input from the human to zero if the velocity reaches values higher than 0.1\(\text{m/s}\).

If the operator issues a velocity command which should be restricted then (3.13) must also be satisfied. Human input can become restricted if the ADEC state is one of the boundary states. The robot is not allowed to exit the workspace past the boundaries even with human interaction. If the robot's position is on one of the boundaries (ADEC states \(\gamma_4\) to \(\gamma_9\)) then human input must be limited. To this effect an additional constraint (3.13) must also be satisfied before a new continuous control command is issued. As soon as the robot is no longer in any of the above states (by leaving the boundaries and entering unrestricted workspace) the additional constraint no longer needs to be satisfied.

Discrete interactions are also studied in the experiments. The discrete interaction is modelled by state \(\delta_3\) in the HDEM model. The discrete interaction implemented is a command to cancel any previous continuous input from the human. This command sets the human input velocity to zero, \(u^H(t) = 0\). This cancellation command is very effective when further interaction is no longer desired and therefore the velocity command (resulting from input acceleration) is to be cancelled.
3.2. EXPERIMENTS USING CONSTRAINTS

3.2.1 Without Human Interaction

The data of a sample run demonstrating the autonomous operation of the robot moving from Target A to Target B and back again is presented in Figure 3-8. Figure 3-8(a) shows the states of the ADEC, $\gamma(k)$, and the measured velocities of the robot end-effector, $\dot{x}$, $\dot{y}$, and $\dot{z}$. Figure 3-8(b) shows the states of the HDEM, $\delta(n)$, and the human input velocity, $\dot{x}^H$, $\dot{y}^H$ and $\dot{z}^H$. It can be seen from Figure 3-8(b) that over the time interval shown the human did not interact (HDEM state $\delta_1$). The measured x-velocity shows that the robot was moving with a positive x-velocity until Target B was reached, which caused a change of state to $\gamma_2$. The y-velocity shows that the robot first moved in a positive direction, steering away from the obstacle. Then the y-velocity reversed to a negative value such that the robot reached Target B. After having reached Target B, the $\dot{z}$-direction reversed, and the robot again moved until Target A was reached (entered state $\gamma_1$) while the obstacle was avoided. The process then repeated itself. A plot of the robot trajectory is shown in Figure 3-9. This clearly shows that the fixed obstacle is avoided. While moving toward Target B the constraints that had to be satisfied were:

- a point to point constraint (3.9) to reduce the distance between the point target and the robot, and
3.2. EXPERIMENTS USING CONSTRAINTS

3.2.1 With Automatic Operation

A point to point constraint (3.10) to disallow decreasing of the distance between the fixed obstacle and the robot.

3.2.2 With Human Interaction

The human was allowed to observe the task by direct sight. Part of the track was screened off such that the human had limited time to react to the train’s movement. If the human observed that a collision between the train and the robot might occur, he could choose to manoeuvre the robot around the moving obstacle by using a joystick to input a continuous command. He then had to make a decision on how to steer the robot around the train. The train could be avoided by: (1) moving over the train (positive \( z^H \)), (2) moving behind the train (positive \( y^H \)), (3) moving around the front of the train (negative \( y^H \)), (4) speeding up to cross the track before the train reached the collision (positive \( x^H \)), (5) slowing down to let the train pass (negative \( x^H \)) or some combinations of the above.

Figure 3-8: Experimental results - autonomous operation (a) ADEC states and total velocities, (b) HDEM states and human velocities

Figure 3-9: Trajectory during autonomous operation
During the experiments the user remained in the Free Space state. This is the case in the first of the following experiments with human interaction. If the human were to approach the boundary of the workspace his input will be constrained. This is the case in the second example.

**Unconstrained Human Interaction**

Data demonstrating trials with various types of human interactions are shown in Figure 3-10(a) and (b). Figure 3-10(a) shows the states of the ADEC and the measured velocities of the robot end-effector. Figure 3-10(b) shows the states of the HDEM and the human input velocities. To aid discussion, the figure has been divided up into three stages. These will be examined further.

In **Stage 1**, the time interval from \( t=0 \) to \( t=14 \), a continuous input from the human was recorded. Figure 3-11 (Stage 1) shows the sequence of the demonstration taking place. At \( t\approx 2 \) the human recognised a potential collision and chose to move the joystick upwards in order to allow the train to pass under the robot. The state changed in the HDEM from \( \delta_1 \) to \( \delta_{2.1} \). When the human ceased to accelerate the robot (the human stopped interacting) the HDEM state returned to \( \delta_1 \). The human velocity however, remained at its non-zero level. The measured velocity \( \dot{z} \) shows how the human input affected the velocity of the robot. At \( t\approx 4 \) the human then pushed down on the joystick to reduce the speed of the robot in the upward \( z \) direction. This was necessary as \( u^H \) was larger than the maximum \( u^A \) and hence \( u^A \) could not overcome...
the human input. Without the downward $\dot{z}_H$ the robot would not have reached the target. The ADEC continued in state $\gamma_3$ until $t=7$, at which time the robot reached Target B (state $\gamma_3$). This event also caused the human input velocity to return to zero. After spending a small amount of time at Target B, the robot then moved back into free space.

In Stage 2, the time interval from $t=14$ to $t=27$, the human avoided the train by moving around the front of the train. Here the continuous velocity input was cancelled by a discrete input. Figure 3-11 (Stage 2) again shows the sequence of the demonstration. At $t=16$, the human issued a command to move the robot in the positive y-direction. The velocity change can be seen in Figure 3-10(b), $\dot{y}^H$. This interaction caused the robot to move away from the train and across the track in front of the train. While the human was accelerating the robot the HDEM was in state $\delta_{2,1}$. When the human stopped interacting the HDEM state returned to $\delta_1$. The velocity change was reflected by the change in $\dot{y}$. The human issued a discrete input at $t=18$ which caused the HDEM state to change to $\delta_3$. This event caused the continuous input to be cancelled as can be seen in $\dot{y}^H$. The HDEM only remains in state $\delta_3$ for a short instance. Once the discrete command was executed the state returns to $\delta_1$. At $t=22$, Target B was reached. The process then continued and the robot returned to Target A at $t=25$.

In Stage 3 of the demonstration, $t=26$ to $t=31$ seconds, a positive x-velocity was commanded by the human which caused the robot to accelerate quickly past the front of the train. The sequence is shown in Figure 3-11 (Stage 3). In this stage the velocity command was counteracted
3.2. EXPERIMENTS USING CONSTRAINTS

by a continuous human input in the negative x-direction. The human velocity was returned to zero when target B was reached. Note that because of the acceleration in the x-direction, it took less time to move from Target A to Target B.

**Constrained Human Interaction**

In some circumstance the human may have difficulty issuing precise continuous commands. The human input may also cause potentially hazardous situations. In these situations it is useful if the human input can be guided or constrained. In the case of our example the boundaries of the workspace may not be crossed. These boundaries are modelled as states of the system as shown in the ADEC automaton, Figure 3-7(b). If the current state of the ADEC is one of these boundary states, an additional constraint (3.13) is added. This keeps the robot within the workspace under any human input. If the human attempts to manoeuvre the robot outside the workspace, the robot moves along the boundary.

![Figure 3-12: Experimental results - constrained operation (a) ADEC states and total velocities, (b) HDEM states and human velocities](image)

The data of a sample run demonstrating the autonomous operation are presented in Figure 3-12. In the time interval from t=0 to t=15 the robot operated autonomously. Shortly after t≈16 the human input a velocity in the positive x-direction. At t≈16 the robot reached the upper y2 boundary, ADEC state 7. Figure 3-13 shows the trajectory of this trial. It can be seen from Figure 3-13 that the robot trajectory followed the constraint. The robot followed the boundary because the velocity solved for had to satisfy the equality constraint associated with
this state. The overshoot visible is caused by two factors.

New or additional constraints are activated when an event occurs. A constraint will only affect the control command once it has become activated. Therefore there will always be a short delay between the detection of an event (entering of a new constrained state) and the new control command taking effect. An additional delay comes about due to the time lag between actual (physical) events and recognised events. Therefore, when modelling real-world surfaces, the constraint which models the surface must be a distance, \( d_{\text{min}} \), from the actual surface such that \( d_{\text{min}} \geq v_{\text{max}} \Delta t \), where \( v_{\text{max}} \) is the maximum robot velocity and \( \Delta t \) is the sum of the above two described delays.

As \( t \approx 18 \) the robot reached the intersection between the two upper \( y \) boundaries, state \( \gamma_9 \). The robot then remained in this state until \( t \approx 23 \). The robot remained in this state (and position) because two boundary constraints met and therefore two equality constraints were solved for. The only possible solution of the autonomous command, \( u^A(t) \), was a velocity command of 0 m/s. Shortly after the human issued a new velocity command and thus moved the robot away from the boundary. The new human input velocity was issued at \( t \approx 22 \). This continuous input caused the HDEM state to change from \( \delta_1 \) to \( \delta_{2.1} \) and back to \( \delta_1 \). The state of the ADEC was in state \( \gamma_7 \) for a short period of time as the robot moved away from the boundary. At \( t \approx 24 \) free space is reached, state \( \gamma_2 \). At \( t \approx 26 \) the robot reached Target B. At this time the continuous velocity issued by the human was cancelled as Target B is a goal state of the system. The robot then continued the autonomous process back to Target A.

Constraint Position Modification

Two experiments were conducted in which the human could modify the position of the constraints. These results are shown in Figures 3-14 and 3-15. This modification of the robot's
path was performed by continuous human interaction in HDEM state $\delta_{2,2}$.

![Figure 3-14](image-url)

**Figure 3-14**: Operator moving the obstacle from position 1 to position 2

In the first experiment, shown in Figure 3-14 the obstacle was gradually moved by the human from an initial position, indicated by '1' on the figure, to a final location, indicated by '2'. This type of interaction is useful in dynamic environments in which obstacle positions can change and thus the human supervisor must modify the environment description. To model the obstacle, the experiment uses a "maintain or increase" distance constraint which can be moved by the user. Initially the robot moved to pass above the target. As the human moved the obstacle, the robot changes direction and then moves around below the obstacle.

![Figure 3-15](image-url)

**Figure 3-15**: Operator moving the obstacle into the workspace after robot commenced motion

In the last experiment, shown in Figure 3-15 the obstacle initially was not present in the workspace and was introduced shortly after the robot started on its path (this is indicated by the large arrow). It can be seen that at first the robot trajectory is directly towards the target. On introduction of the obstacle into the workspace, the trajectory is altered to avoid the obstacle.
3.3. POTENTIAL FIELDS

These trials show how both equality and inequality constraints were used to model an environment and how the velocity control commands were calculated from the same constraints. The method allowed the robot to reach the target while avoiding the obstacle. The constraining of human input was achieved effectively by adding only one additional constraint, (3.13).

The ability to constrain the human input is an advantage as the risk of the human making an error is reduced. The constraints were also utilised to guide the human in order to reach a particular goal or follow a precise path. In our experiments the intersection between two constraints could be considered a goal. Once the robot reached state \( \gamma_9 \) (the intersection) it stayed in this position until the human commanded otherwise.

The human was also able to move the constraints to change the robot's path. This ability is effective in dynamic environments where positions of obstacles change with time. In addition to the continuous commands, the user could issue a discrete command to cancel any previous continuous input. This was very effective in order to return the robot to autonomous operation. Other discrete commands such as 'go to alternate position' (if an alternative is previously defined) could also assist the user. To add such discrete commands, further states in the HDEM may need to be added. The hierarchical discrete event structure as discussed in Section 2.7 can be utilised for this.

3.3 Potential Fields

The second method of control synthesis utilises potential fields. This provides an alternative method for defining the map \( \phi^A \), i.e. the conversion of discrete controller events issued by the ADEC to continuous control commands. The potential field method also allows restricting or guiding human input which is important in the area of shared control.

Potential fields are used because they provide a straightforward method which can deal with difficult environments without a complex set of path planning rules. The potential field method also offers two main advantages over constraints. The first is the ease with which potential fields can be modified and the second is that complicated shapes are easier to describe in a more consistent manner. Potential fields suffer one major disadvantage; there is no direct control over the size of the velocity command issued. The velocity command solely depends on the gradient of the field. The constraint method allows the resulting velocity to be precisely defined using Equation (3.16).

For the purposes of user interaction and shared control, the fields can be modified by changing one or two variables. This easy way of making changes to the potential fields makes them
3.3. POTENTIAL FIELDS

attractive for online modification by a human supervisor. The ability to change the potential fields description quickly online is particularly useful for the user to perform path re-planning, alternate goal selection and correction of environment modelling errors.

3.3.1 Modelling with Potential Fields

Potential fields can be divided into two main groups, attractive and repulsive potentials. Attractive potentials can be represented by quadratic and conical wells [52] [53]. The potential $U_w$ of a conical well at any given point $(x, y)$ in the workspace can be derived by

$$U_w(x, y) = 2l[(x - x_w)^2 + (y - y_w)^2]^{\frac{3}{2}} - l$$  \hspace{1cm} (3.18)

where $x$ and $y$ are task space coordinates in the robot workspace, $l$ is a scaling constant and $(x_w, y_w)$ is the centre of the well. The scaling constant $l$ determines the depth of the well. However, it is important to note that the shape of the well is not affected by $l$ and thus $l$ is only useful as a scaling constant when used in conjunction with other types of fields.

This type of well is centrally attractive at any distance and is utilised in order for the robot to reach a target position, the centre of the well. Figure 3-16 shows a plot of the conical well. It should be noted that this well can be used in $n$ dimensions. However, any space with more than 2 dimensions becomes difficult to visualise.

Repulsive potentials are important in order to repel the manipulator from obstacles or repel the manipulator from a boundary which is not to be crossed. These repulsive potentials can also be used to constrain the involvement of the human. For example, a repulsive boundary can be used to disallow the human to manoeuvre the robot outside a given workspace.
3.3. POTENTIAL FIELDS

Objects which form obstacles to be avoided are often complex in shape. Mathematical models for potential fields can only model simple shapes such as cylinders, cubes, ellipsoids, trapezoids and the like. Hence complex obstacles may need to be modelled by a combination of several potential fields. A disadvantage of the addition of a group of potential fields, repulsive and attractive, is that often local minima are created other than the desired minima of the well. However, an advantage of having human involvement is that the human can manoeuvre the robot away from these undesired minima.

Two types of repulsive potential fields are utilised in this paper in order to show how the human can enhance and be limited by potential fields used for robot control. The first type of field is an ellipsoid (circle) to represent an obstacle in the workspace of the robot. The potential field $U_o$ of an ellipsoid can be represented by

$$U_o(C_o) = A_o e^{\frac{-C_o}{C_o}}$$  \hspace{1cm} (3.19)

where $A_o$ is a scaling constant and $C_o$ is given by

$$C_o = \left( \frac{x - x_o}{a_o} \right)^{2m} + \left( \frac{y - y_o}{b_o} \right)^{2m}$$  \hspace{1cm} (3.20)

where $x_o$ and $y_o$ define the centre of the ellipsoid and $m$ is an exponential parameter. Parameters $a_o$ and $b_o$ are the dimensions of the semi-major axis and the semi-minor axis of the ellipse respectively. Figure 3-17 shows a 3-D plot of the ellipsoid with equal minor and major axis (higher values of the potential field have been truncated).

![Figure 3-17: Potential field of repulsive elliptic obstacle](image)

The second type is a rectangle to represent a boundary to the workspace. The boundary is represented as an inverted rectangle. The potential field of the rectangle is calculated by

$$U_b(C_b) = A_b e^{\frac{C_b}{e - C_b}}$$  \hspace{1cm} (3.21)
where $A_b$ is a scaling constant and $C_b$ is determined by

$$C_b = \left[\left(\frac{x - x_b}{a_b}\right)^{2m} + \left(\frac{y - y_b}{b_b}\right)^{2m}\right]^{\frac{1}{m}} - 1$$  \hspace{1cm} (3.22)

where $x_b$ and $y_b$ specify the centre of the rectangle. The dimensions $a_b$ and $b_b$ are derived from the width $w$ and depth $d$ of the rectangle, calculated as follows

$$a_b = \frac{w}{2} \left(2 \frac{1}{m}\right) \hspace{1cm} b_b = \frac{d}{2} \left(2 \frac{1}{m}\right)$$  \hspace{1cm} (3.23)

Figure 3-18 shows a 3-D plot of the inverted rectangle (higher values of the potential field have been truncated).

The exponential parameter $m$, determines how fast the field grows. In effect it determines the radius of the curve of the field between low potential and high potential. This effect is shown on a cross section through the centre of the boundary in Figure 3-19. High values of $m$ result in fast transitions. With $m = 100$ (solid line) the transition between low and high potential is gradual, whereas the transition with $m = 200$ (dashed line) is very fast. There is a practical limit on $m$ as the parameter $C_0$ tends to grow very quickly and therefore can result in numerical overflows. Thus $m$ should be chosen to give a continuous curve which will not generate numerical overflows when the field is calculated.
3.3. POTENTIAL FIELDS

3.3.2 Velocity Generation

The continuous velocity for the robot manipulator is generated by calculating the derivative of the composite potential field [53].

\[ u^4 = \frac{\partial}{\partial x} [U_t] \hat{i} + \frac{\partial}{\partial y} [U_t] \hat{j} \]  

(3.24)

In our case of two dimensions, the composite potential field, \( U_t \), is found by adding the fields of the well (3.18), the obstacle (3.19) and the boundary (3.21) according to

\[ U_t = U_w(x,y) + U_o(C_o) + U_b(C_b) \]  

(3.25)

Figure 3-20 shows the addition of these three fields. Any number of individual potential fields can be added to give a composite field and the choice is not limited to only ellipsoids, rectangles and wells.

3.3.3 Human Interaction

As with constraints, potential fields offer two methods of interacting with the control system on a continuous level i.e. through HDEM state \( \delta_2 \). This is in addition to the discrete interaction method utilising HDEM state \( \delta_3 \).

The potential field method generates a velocity command for the plant. It follows that the first method of continuous interaction allows the human to add a velocity command to that generated from the potential field. To exploit the ease of modification related to potential fields, the second method allows the human to modify the potential field model of the environment.
Figure 3-21: Hierarchical HDEM model utilised for two types of continuous interaction

The hierarchical HDEM model described in Section 2.5 can once again be used. Because there are two continuous interaction methods the model shown in Figure 3-21 can be used.

Similar to equation (3.11) and in accordance with the hierarchical HDEM model we can now define the continuous human input parameter \( h(t) \) for each of the sub-states

\[
h(t) = \begin{cases} 
  u_H(t) & \text{if } \delta(n) = \delta_{2.1} \\
  [\Delta x, \Delta y, A_0] & \text{if } \delta(n) = \delta_{2.2}
\end{cases}
\]  

(3.26)

where \( u_H(t) \) is a velocity command input by the human, \( \Delta x \) and \( \Delta y \) represent the change in position of the field and \( A_0 \) is the potential field scaling constant which can be varied between 0 and a maximum value determined at the programming stage.

Each method of human interaction results in the robot trajectory being altered. The method of adding to the autonomous velocity has the benefit that if the human commands more speed in the x-direction, the robot will move faster in the x-direction. Thus this type of interaction is more "intuitive" to the human operator and therefore can be very useful in temporary situations which require fast reaction times. The second human interaction, modifying the potential field description, changes the position and/or size of the fields. This has the effect of altering the velocity calculated in (3.24). Therefore the robot will move according to the new field description. This second method is more useful in situation where the environment model has changed on a permanent basis. Note that the modification of the potential field description may lead to local minima being created (or removed).

In the first method the velocity input by the human, \( u(t) \), is added to the velocity generated
3.4. EXPERIMENTS USING POTENTIAL FIELDS

by the autonomous system (3.24). Equation (2.12), is now defined as

\[ u(t) = u^A(t) + u^H(t) \]  

(3.27)

The human velocity command is limited by a maximum value determined at the time of programming. The maximum can be regarded as the "strength" of the human. The maximum is imposed so that large velocities, exceeding the robot's safe operating limits, are not reached. The size of this maximum also affects the manoeuvring capabilities of the human inside the field. Higher maxima for this velocity allow the human to manoeuvre the robot closer to the repulsive potential fields because higher potential gradients are required to cancel the human velocity input.

The second way in which the human can interact, HDEM state \( \delta_{2,2} \), is by modifying (changing position and/or size) of any components of the composite potential field. This has the effect of directly changing the generation of \( u^A(t) \) by changing parameters in the mapping \( \phi^A \) (as indicated in Figure 2-3 by the arrow through the \( \phi^A \) mapping).

Moving the well, obstacle or boundary is useful if the environment model is incorrect. The fields can then be re-positioned to more accurately model the environment. In this type of interaction, \( h(t) \) contains information on changing the position and size of the potential field, see Equation (3.26). The position of the field is modified according to

\[ x_0 = x_0' + \Delta x \quad y_0 = y_0' + \Delta y \]  

(3.28)

where \( x_0' \) and \( y_0' \) represent the starting location of the centre of the well, obstacle or boundary. By changing the size and position of the potential field, the velocity command for the continuous system is again modified.

3.4 Experiments Using Potential Fields

3.4.1 Without User Interaction

The method of integrating human input with autonomous commands generated by potential fields is demonstrated by experiments. The experiments were implemented using a 5-degree of freedom Eshed Scorbot VII (Figure 3-22) which performed a simple manoeuvring task. The task is based on a type of pick and place operation, where the robot starts at a point A and then moves to a second point B. Human interaction may be required if an alternate goal is specified, for example if the picked up object is damaged and it is to be placed in a special bin. It can then be left up to the human to move the robot to the special bin. Preprogrammed guides
which assist the human to manoeuvre the robot to the bin may assist in this task. In dynamic environments, the human can manoeuvre the robot around moving obstacles or even modify the environment model to avoid potential collisions. Different ways of human interaction with

![Experimental setup for manoeuvring task using a 5-degree of freedom Eshed Scorbot VII](image)

the otherwise autonomous system are shown in each of the following experiments.

In the first experiment, the composite field of equation (3.25) was used to generate robot velocities and the robot was not interfered with by the human. Figure 3-23 shows the recorded robot trajectory. A contour of the potential fields (object and boundary) at field magnitude zero is indicated on the graphs by the dashed line. The well was located at position $x = 0.56$ and $y = 0.0$ and indicated by the word "Target". It can be seen that the robot was attracted to the well, but at the same time the velocity generated was such that the obstacle was avoided. The human did not interact in this example, thus the human interaction state remained in state $\delta_1$ (Figure 2-2).

### 3.4.2 With User Interaction

The experiments with human user interaction are divided into two sets. In the first set the user added velocity commands to the autonomous velocity derived from the potential fields. The state of the HDEM during user interaction was $\delta_{2,1}$.

In the second set the user modified the potential field itself and therefore the map $\phi^A$. Here the state of the HDEM was $\delta_{2,2}$. 
3.4. EXPERIMENTS USING POTENTIAL FIELDS

Velocity Command Input

The next four experiments show how the human can interact with the system by issuing a continuous velocity command. While the human is interacting with the system the HDEM state is $\delta_{2,1}$. Here Equations (3.26) and (3.27) are of importance as these show how the human input affects the robot. Issuing velocity commands allows for trajectory alterations on a one-off basis. If the robot were to move back to the origin, its trajectory would be once again only dictated by the environment model.

In the first experiment of velocity interaction which is shown in Figure 3-24, the human used a 4-degree of freedom joystick to input a velocity command. In this example the human manoeuvred the robot toward the boundary. This example shows how the boundary restricts the movement of the human. This restriction can be regarded as guiding the human input. Human input is restricted since the human cannot move over the top of the boundary. This is important as limits on human movement can be implemented in this way. Human guidance is achieved since the robot follows the boundary.

The point at which the human started to interact is indicated by the point marked "H". The human input was added to the ADEC velocity command (Equation (3.27)) such that the robot moved toward the boundary. As the robot approached the boundary, $u^A(t)$ increased in the $y$-direction and thus eventually counteracted the input by the human. In Figure 3-24(a) the maximum allowed human velocity, $u_{M}^{H}$, was 0.05m/s and it can be seen that it was possible for the human to manoeuvre the robot very close to the potential boundary. In Figure 3-24(b) the maximum allowed human velocity was 0.04m/s. In this case the human’s movement was more restricted and it was not possible for him to make use of all the allowable workspace. Thus the “strength” of the human plays an important role when dealing with potential fields as a restrictive medium. The curvature in the trajectory is due to the conical shape of the well.
and the closer proximity of the robot to the well.

The velocity in the x-direction was derived from the well and hence the robot moved along the boundary. At the point where the robot began to approach the well, the human cancelled the velocity input. If the human had not cancelled the input, the robot would have reached a human induced local minima in the corner of the workspace.

This example shows how the human input was constrained by the boundary potential field. Even though the human input was pushing against the boundary, the potential field did not allow the human to manoeuvre the robot out of the workspace. From the two experiments it is clear that the choice of the human's maximum velocity (or the choice of gradient of the field) affects the human manoeuvring capabilities dramatically.

The human input can also be restricted by the obstacle. This is shown by the following example. Here the human attempted to push the robot into the obstacle. Note that $u_{\text{MAX}}^H = 0.05m/s$. It can clearly be seen from Figure 3-25 that the autonomous velocity, in this case mainly due to the obstacle, counteracted the human velocity input due to the addition (3.27).

Finally, velocity interaction is useful in cases where the environment description has introduced
3.4. EXPERIMENTS USING POTENTIAL FIELDS

Figure 3-25: Experimental trajectory, human pushing into obstacle

local minima. This is often the case when environments are modelled by potential fields. In the following experiment the shape of the obstacle was modified so that a local minimum was created. The position of the local minimum is indicated on Figure 3-26 by the cross at (0.36, 0.01). It can be seen from the figure that the robot moves towards the target but then becomes 'trapped' behind the obstacle. To assist the robot to escape the local minimum, the human input a small positive y-velocity. This velocity helped to manoeuvre the robot closer to a 'potential ridge' which had developed between the boundary and the expanded obstacle. The attraction of the well was then great enough to 'pull' the robot past the obstacle and then once again towards the well.

Figure 3-26: Experimental trajectory, human assisting at local minimum

Potential Field Modification

In the remaining experiments the human modified the potential field description. This is also achieved via the continuous interaction state $\delta_{2,2}$. Here, the second definition of $H$ in (3.26) is being utilised to modify the size of the fields and the position. This type of interaction is important as it can be used to modify the environment description on a permanent basis.
The environment model may need to be modified if there were errors in the initial environment description. Modifications can also be carried out in environments where obstacles are not fixed but rather move in the robot workspace. After a modification in the potential field, the robot trajectory would be dictated by the modified field. However, the modification of the field is more difficult for the human as he cannot immediately tell how the modified field will change the velocity. To understand what effect the field modification will have, it is important that the human can visualise the potentials. Unfortunately potentials of more than 2 dimensions are difficult to visualise. The following field modifications are performed on-line after the robot has begun its path from the origin to the target.

In this experiment the human modified the position of the repulsive obstacle in order to update the model of a dynamic environment. It can be seen in Figure 3-27 that with the obstacle at location 1, the initial trajectory would have passed the obstacle on the positive y-side. The human then gradually moved the obstacle towards location 2. To achieve this movement, Equation (3.28) was utilised to alter the centre of the obstacle. Eventually the trajectory generated by the modified field is such that the robot passed the obstacle on the negative y-side. The ability of the human to modify the position of the obstacle field is significant in that the trajectory can be significantly altered.

![Figure 3-27: Experimental trajectory with human modifying obstacle position](image)

The human can also alter the size of the obstacle from non-existent to a maximum size. Figure 3-28 shows an experiment where the obstacle was first very small (marked “1” on the graph) and then was gradually increased to its maximum (marked “2”) by the human. The parameter $A_0$ in (3.26) was utilised for this purpose. It can be seen in Figure 3-28 that with the obstacle very small the robot was moving directly toward the location of the well unaffected by the small obstacle. As the obstacle was increased in size by the human the robot was forced to move away from the obstacle and eventually around it towards the well.

In a similar way to moving the obstacle, the human can also move the well and thus the human
3.4. EXPERIMENTS USING POTENTIAL FIELDS

Figure 3-28: Experimental trajectory with human modifying obstacle size

has the ability to change the goal of the task. Changing the goal of a task is important in systems which are designed to complete a variety of tasks and the switching between tasks occurs infrequently. Tasks in which tools are damaged often require alternate task goals so that the tool can be changed. In the following experiment the human moved the well, and thus altered the goal, from its original location to (0.55, -0.12) and it can be seen that the robot reaches this point.

Figure 3-29: Experimental trajectory, human moving well

It was shown that the potential field method provides a means to synthesise continuous control commands from discrete controller commands. The potential fields can also be used to restrict human input, particularly in the case of a velocity input. However, the biggest advantage of potential fields is that these can easily be modified and thus modelling errors can be corrected while the system is on-line. This provides great flexibility and the system can easily be adapted to dynamic environments. However, if changes to the field were made, these were not immediately obvious to the user. This is explained further below.

The potential field provided a means for a boundary which was used to guide as well as restrict the human input (Figure 3-24). This is useful in applications where the human uses the robot
as a guide to complete a task. However, with the type of field used for the boundary in these experiments, the transition between no effect from the field and the field being strong enough to stop the human is gradual (see Figure 3-19). Therefore it can be difficult to specify an exact boundary and variations in the parameters, such as the maximum human speed, affect the manoeuvring capabilities of the human to a great extent. The effect of human manoeuvring near the boundary with different maximum velocities is shown in Figures 3-24(a) and (b). The interaction method of adding a human velocity was easier for the human as the input was directly reflected by the robot's change in velocity. This means that if the human input a command to move left, the robot also moved left. This is more intuitive than perhaps moving a field left and the robot moving towards the right. The input of velocity also allowed the robot to overcome a modelling problem, a local minimum. The addition of a velocity command has the limitation that it is a temporary command and if the process was to be repeated the same velocity command would have to be issued to overcome the local minimum once again.

Modification of the potential fields was a second method to change the robot velocity. The creation of a new field of varying size and position was demonstrated by experiments in Figures 3-27 and 3-28. This is important for robots moving in unpredictable environments where obstacles may move in and out of the workspace. The ability to modify the goal position (Figure 3-29) is also of benefit as it allows alternate goals, which were not initially modelled for, to be included. If modelling errors of the workspace were made then these errors can be corrected by modifying the fields. Additionally, local minima which often occur in potential field models can be overcome by human velocity input (Figure 3-26). This method of changing the path of the robot is more permanent. Once the field is modified, it remains modified. However, this method is difficult to use to directly affect robot motion as the modification of the field does not necessarily move the robot in the desired direction. For example, if the robot is avoiding a cylindrical obstacle by circumventing it to the left and the human then modifies the potential field and moves it to the left - the robot would first back away and then move towards the right. In this case the robot motion did not reflect the user interaction directly. Thus the method of field modification is better suited to modify environment modelling. The modification of the position of a field has disadvantages. Unless the human has a clear picture of the location of the fields, it is possible for him to move an attractive field near a repulsive field and the robot would then never reach the attractive well. This also applies vice versa. It would be difficult to follow a particular trajectory (guiding) by modifying potential fields. The first method is more attractive for this purpose.

The addition of the hierarchical structure made it possible to have two different types of continuous interaction from the user. It is therefore vital in systems requiring several types of continuous interaction as well as discrete interaction to model the HDEM using hierarchical
3.5 Evaluation of Constraints and Potential Fields

The potential field and constraint method were evaluated and compared in two ways. The first method of comparison is by the type of human interaction, whether by velocity modification or model modification. The second method for comparison is computation which includes time taken and smoothness of path.

3.5.1 Human Interaction

The first method of human interaction is via HDEM state δ2,1, velocity interaction. Here the user input a velocity command which is added to the velocity command generated by either the constraint method or the potential field method. Both the constraint method and the potential field method performed equally well under an added velocity command. Even when the human input becomes restricted, both methods function quite well. This can be seen in Figures 3-13 and 3-24.

When the human chose to modify the robot’s path by modifying the model of the environment, both methods also performed well. Consider the case where the human moved the obstacle within the workspace. The constraint example is shown in Figure 3-14 and the potential field method is demonstrated in Figure 3-27. In both cases the path of the robot is such that it first moves above the obstacle but then reverses and moves below the obstacle. Potential fields in this case have an advantage over constraints because these can model more complex shapes and thus can move these shapes rather than moving only one constraint such as a point constraint as demonstrated.

In Figure 3-15 the human moves an obstacle from outside the workspace into the robot’s path. Again, the obstacle is successfully avoided and the robot reaches the goal. In Figure 3-28 the operator “grows” the obstacle by increasing the size of a potential field. The introduction of a new obstacle into the workspace is therefore available to both methods. However, the potential to increase or decrease obstacles in size is only possible with the potential field method. This size alteration is not possible with constraints as these are either present or not present.

A comparison based on the above results is difficult to make because both constraints and potential fields were successful in guiding the robot to its goal while avoiding obstacles as necessary. However, potential fields do provide more flexibility with regard to modifying the
model of the environment.

3.5.2 Computational Comparison

The first computation comparison is based on the smoothness of the resulting robot motion. The second is the processing time required to perform the conversion from the discrete command issued by the ADEC to the continuous command issued to the plant.

Path smoothness is directly related to the smoothness of the velocity of the robot. For example, a constant velocity is smooth, as is a steady ramp or a velocity which arcs from zero to a maximum and back to zero. Therefore we can define smoothness to be the standard deviation of the time derivative of the robot velocity according to

\[ s = \sigma \left( \frac{d}{dt} u(t) \right) \]  

(3.29)

where \( \sigma() \) is the standard deviation. A lower standard deviation indicates a smoother path whereas a higher value indicates more erratic movement.

Figure 3-30 shows a graph of a robot path recorded by performing a similar experiment as described in Section 3.2. The robot started on the left and moved towards the obstacle on the right. Note that the continuous input to the robot, \( u(t) \), is a velocity command. The velocities in the \( x \) and \( y \) directions were recorded and the smoothness figure calculated using Equation (3.29). The results are summarised in the bargraph shown in Figure 3-31. It can be seen on this bargraph that both the \( x \) and \( y \) velocities produced by the potential field method are smoother than those calculated using the constraint method.

The reason for smoother velocity commands from the potential field method is the use of the gradient of the potential field to calculate the velocity (Equation (3.24)) which is continuous at
3.6 Conclusion

Two methods of control synthesis were described in this chapter. These methods were supported by experimental results which demonstrate the operation of the synthesis methods. Both methods were shown to be suitable for autonomous robot control and for shared control. Both methods provided ways of limiting or guiding human input as well as providing autonomous control commands. Both methods avoided the obstacle and reached the target which makes them both suitable. However, the smoothness of the path sets the two methods apart, as well as their operating speed. Here the potential field method proved to be superior to that utilising the constraints. The constraint method has the advantage that it allows the velocity to be defined more accurately by Equation (3.16). In cases where simple obstacles or walls need to be defined, the constraint method can also be better as constraints are an abrupt
boundary. Potential fields on the other hand provide a "softer" boundary because their gradient changes continuously.

Two types of interaction methods were presented. The first method allowed direct modification of the robot's velocity and the second method was model modification. The input of a velocity command by the user is better suited for one-off manipulation of the robot's path. It is better as a one-off modification because once made, the human would not be able to interact with the system by velocity modification again without changing the command. The modification of the potential fields however is better suited to permanent modification of the environments description, as any changes that were made could remain even after the HDEM has returned to the No Interaction state.
This chapter presents a case study on the shared control framework applied to the robotic cane, an assistive device for the visually impaired. Experiments which demonstrate the operation of the shared hybrid dynamic control framework are presented. Additional experiments demonstrate the operation of different control strategies for the robotic cane.

Assistive devices for people who have impairments such as disability, old age or the loss of one or more senses can greatly improve their quality of life. A great variety of such assistive devices (such as hearing aids, artificial limbs and powered wheelchairs [29]) is already available to the general public. Many more such device are still being researched, including nursing robots [20], self-guiding wheelchairs [17] [60] and electronic travel aids (ETA) for the visually impaired [93]. The devices still under investigation set themselves apart from hearing aids and artificial limbs in that these require the control of the device to be shared. The human may be responsible for the high level control of the system, such as route planning, whereas the machine takes over more localised control, such as obstacle avoidance or wall following [18].

A shared control system combines the strengths of both robots and humans in order to augment each others’ weaknesses [68] [94]. In the case of someone who is visually impaired, an ETA could assist the person in walking through everyday environments. Such a device is the GuideCane [22] or the Robotic Cane shown in Figure 4-1.

Several electronic travel aids have been devised since the early 1970's. The Pathsounder [81], an ultrasonic device designed to be worn on the chest, informed the wearer about the distance to the nearest obstacle by an audible or tactile warning. The Mowat Sensor [70] is a hand-held sonar device which vibrates according to the distance from obstacles. The Nottingham Obstacle Detector [13] is a similar device with an audible warning with different pitches representing
distances. Optical sensors, such as the Laser Cane [38], had sensors incorporated into a cane. Other devices including the Sonicguide [50] and Trisensor [36] had more than one sensor to provide more information to the user and give better localisation of obstacles.

There are several drawbacks with all of these early ETAs. Perhaps the most fundamental drawback is the need for the user to scan the environment as he does with a normal cane. Note that the sonic guide and trisensor do not need scanning, but these do not detect low obstacles near the ground. The scanning is an activity which takes a continuous conscious effort by the user and is particularly difficult for people who have additional disabilities other than their visual impairment. Another drawback is audio feedback as this interferes with a visually impaired person's most important sense, hearing. Devices such as the GuideCane, the Robotic Cane and the Navbelt [85] overcome these drawbacks by sensing the environment in multiple directions and steering around obstacles autonomously.

Additionally, devices which are designed to aid people must allow the user easy interaction with the device. Therefore completely autonomous devices which simply avoid obstacles are not good enough. The ability to interact is important as the user may want to walk in a different direction to what the autonomous control system currently allows. This research allows the user to interact in such situations, for example, the approach of a closed door. This research also provides a consistent framework with which the human integration is made possible.

There are many obstacle avoidance methods for mobile robots. These include the vector field histogram method [21] applied to ultrasonic obstacle avoidance in the GuideCane and the Navchair. Although more sophisticated than other map-building techniques such as [23] [27] [32] the vector field histogram method still relies on building a localised map to determine where there is free space for the robot to steer. Potential fields [52] have been coupled with
4.1. THE ROBOTIC CANE

map building to provide an easy means to derive control commands from generated maps. However, these methods ignore the dynamics of the mobile platform - it takes longer to stop when travelling fast.

Methods which include the dynamics of a mobile robot as constraints are presented in [14] and [39]. The former of these two references utilises a discrete event controller to steer a mobile robot through a corridor while keeping track of the approach speed to walls. The discrete event model, however, models the entire environment as opposed to relative position to obstacles, which is the approach utilised here. The robot also knows the location of the final goal. [39] uses three sensor in a manner similar to this implementation. The turning of the robot is based on the distance sensor data while taking into account the dynamics of the robot which is an additional advantage of this research.

For the Robotic Cane the shared hybrid dynamic control framework is used. The control system makes use of the proximity of obstacles, cane velocity, sensor confidence, human interaction or a combination of the above to monitor the environment and hence issue steering commands. In this thesis three combinations of the above are implemented and tested. The first experiments use monitoring based only on velocity to determine if events have occurred. In this case obstacles are avoided earlier if the approach speed to an obstacle is higher. The second method uses sensor confidence for event recognition. Sensor confidence is based on sensor variability. If confidence is low then obstacles are avoided earlier. The last method tested is a combination of velocity and sensor confidence. This combination overcomes some of the disadvantages of the each of the other two methods used alone.

Additionally, the human can interact with the cane through a joystick mounted on the handle of the cane. With this joystick, discrete commands can be issued, such as “Go Straight” or “Turn Left”. These commands are important for the shared control aspects of the system and allow the user to interact when needed. The three control methods introduced above are each tested with and without human interaction.

4.1 The Robotic Cane

The Robotic Cane is a device that aids visually impaired people to travel safely in obstacle rich environments. The motivation for the cane presented here came from the “GuideCane” [22]. The Robotic Cane is a light, wheeled platform with a long handle so that it can be comfortably pushed in front of the user while walking. The cane steers around obstacles which are sensed with three ultrasonic distance sensors. Therefore, to avoid obstacles, the user simply has to follow the cane’s path. One difference from the GuideCane is that there are two independently
steerable wheels which allows the wheels to act as a brake. The Robotic Cane uses only three forward looking sensors, whereas the GuideCane utilises nine. The GuideCane also has three upwards facing sensors and one which senses downwards. Another difference is the control system. The Robotic Cane uses a discrete event controller to make steering decisions as opposed to building a map of the localised environment.

Figure 4-2 shows a schematic of the cane. It consists of two independently steerable, unpowered rollerblade wheels on top of which is mounted an array of three ultrasonic distance sensors. Rollerblade wheels offer good friction on most types of ground. A joystick is mounted on the handle so that the user can interact when needed. The cane weighs approximately 1.6kg without batteries. These elements and associated design decisions of the cane are discussed in more detail below.

With the current experimental setup the cane has an umbilical cord (carrying power and serial data) by which it is connected to a host computer. The host computer executes the discrete event control and determines the new control command. The on-board Motorola mc68332 micro-controller currently processes ultrasonic sensor information, encoder counts and generates control pulses for the servos. It is possible to control the device entirely from the on-board processor. However, as memory is limited, data recording on the cane itself is limited. Therefore, for development and data recording purposes an umbilical cord is attached. The need for on-board power (approx. 200mA) is also eliminated which is useful in an experimental setup.

The cane senses obstacles using three sonar sensors which can be seen in Figure 4-3. Each of the ultrasonic sensors is a time of flight distance sensor. The three sensors detect obstacles in a range from 0.2m to 5.0m in a time interval of 33ms each. The accuracy of each sensor depends on the distance to the obstacle as well shape, texture and size of the reflecting surface. The
4.1. THE ROBOTIC CANE

Figure 4-3: Sensing platform with three ultrasonic sensors, steering servos and microprocessor

sensors are arranged as shown in Figure 4-4 with Sensor 1 on the right and Sensor 3 on the left (as viewed from behind). For the cane to avoid obstacles at walking speed, the sensing must be most effective in front of the cane. The sensor arrangement must concentrate on obstacles in front of or just to the front left and front right of the cane. To accommodate the forward movement and after experimentation on sensor placement, the sensors were positioned such that their centre axes are at 0° and at ±40°. This arrangement gave good coverage in front of the cane at walking speed. Each of the sensors has a cone angle of approximately ±10° with a maximum range of approximately 5m. Therefore this sensor arrangement creates holes at ±(10° to 30°) and directly to the side of the cane. In these holes small obstacles cannot be detected. More sensors can be employed to overcome these holes.

As little as three sensors can be employed because the cane is primarily travelling forwards. Obstacles in the holes will pass the cane on either the left or the right and will either be detected by the available sensors or passed without detection or collision. Obstacles immediately to the left and right are of no consequence to the travel direction of the cane as it cannot turn at 90°. The use of three sensors also keeps the cost and complexity of the control system to a minimum.

The steering direction is based on the sensor data which is processed by the control system. Commands from the control system drive two independent actuators (RC servos) to set the turning angle of each wheel. With this method, turning can be implemented as well as a stopping or braking manoeuvre. The braking is achieved by turning wheels in opposite directions. The opposing wheels block any movement of the cane without sliding of the wheels. The resulting friction can be felt by the user thus alerting him to slow down.

Optical encoders fitted to each of the wheels allow the distance travelled and the speed of the
4.2 THE CONTROL SYSTEM

To avoid or approach obstacles based on sensor information and human input, the cane utilises the shared control framework introduced in this thesis. The control system, shown in Figure 4-5, is made up of four sub-systems. For the cane to avoid obstacles without human interaction the Autonomous Discrete Event Controller (ADEC) is used. The user can interact via the Human Discrete Event Model (HDEM). The third sub-system is the plant, which is the cane to be determined. The speed is used by the control system to determine the distance at which obstacles are to be avoided (far from the obstacle if the cane is travelling fast).

To allow for shared control, a small thumb activated joystick is mounted on the handle of the cane, near the user's hand. Using the thumb, the user can push the joystick in any of four directions, each of which results in a different command to the cane. These commands include "Go Straight", "Turn Left", "Turn Right" and "Approach". This user interaction is used when the human wants to go in different directions to what the autonomous control system has directed.

For example, consider a user wanting to walk next to a wall. Under autonomous operation the cane would steer away from the wall. By issuing the "Go Straight" command the user can remain on a path which does not steer away from the wall. If there are obstacles sensed by the cane and the human issues the "Go Straight" command, a command compromise is reached. This is explained further in Section 4.4. The Approach mode is used if the user wants to approach obstacles closer than possible under autonomous operation. In the approach mode the cane becomes "less sensitive" to obstacles. This reduction in sensitivity allows the cane to get closer to obstacles before taking the same avoidance measures as it would under autonomous operation.

4.2 The Control System

Figure 4-4: Sensor arrangement and sensing angles (top view)
4.2. THE CONTROL SYSTEM

itself with its sensors and actuators. Finally the interface enables communication between the above three sub-systems.

![Diagram](image)

Figure 4-5: Hybrid dynamic systems framework for shared control of the robotic cane. \((\theta(t))\) is the continuous servo angle input, \(x\) is the continuous state (state of the sensors, encoders and servos), \(\tau\) is an ADEC event, \(\kappa^A\) is a HDEM event caused by the ADEC, \(\nu\) is an ADEC controller event and \(H\) a human input), \(\psi\) is a map from discrete control commands to \(\theta(t)\) and \(\phi\) is a process monitor

4.2.1 The Plant

The plant is the physical structure performing the task and its associated control system. To successfully control and integrate a model of the cane into the control system, its equations of motion must be determined. Figure 4-6 shows a reference frame, \([i, j]\), attached to the cane.

![Diagram](image)

Figure 4-6: Equations of motion
Note that the cane always travels in the x direction. The kinematics of the cane relative to an obstacle can be expressed as

\[
\dot{q}_i(t) = \begin{bmatrix}
v(t) - y(t - \Delta t)\dot{\theta}(t) \\
(v(t) + x(t - \Delta t))\dot{\theta}(t)
\end{bmatrix}
\]

where \(\dot{q}_i(t)\) is change in the distance to an obstacle at time \(t\) (1 \(\leq i \leq 3\) refers to one of the three sensors), \(v(t)\) is the speed with which the human is pushing (always in the x direction and determined by wheel encoder readings), \(\dot{\theta}(t)\) is the change in the input command and therefore the new direction of travel, \(\Delta t\) the time between sensor samples and \((x(t), y(t))\) is the measured position of an obstacle relative to the cane at time \(t\). Note that the accuracy of \((x(t), y(t))\) is only as accurate as the sensors and that each sensor can sense only one obstacle at one time.

### 4.2.2 Autonomous Discrete Event Controller of the Cane

If the user chooses not to interact, only the ADEC controls the autonomous obstacle avoidance behaviour of the cane. The ADEC is modelled as a finite state machine which is event driven. This finite state machine models possible conditions associated with the three sensors. An inequality associated with each sensor is true if the cane is far enough from an obstacle and moving slow enough, otherwise the inequality is false. (This is further discussed in Section 4.3). Therefore each of the three inequalities effectively represents whether or not an obstacle is sensed in front of a particular sensor. With three inequalities which are either true or false the ADEC has eight states which are represented in Figure 4-7. The status of the three inequalities, associated with each of the sensors, are indicated in the figure as three boxes. An inequality which is true is indicated in white and by 1, false inequalities are black and marked with a 0. It should be noted that the state space pictured in Figure 4-7 is fully connected as any inequality can change at any time, independent of other sensors.
As shown in Figure 4-5 the input to the ADEC is the $k^{th}$ event, $\tau(k)$. An event is defined as a change in the one or more of the inequalities, i.e. an inequality is now true which was previously false and vice versa. An example of an event occurring is the cane approaching an obstacle on the cane’s right. Then inequality 1, the inequality associated with sensor 1, changes status and becomes false. This transition is an event described by $\tau(k)$.

More sensors could be used resulting in more inequalities and therefore more states. This can lead to state explosion, a phenomenon not uncommon in discrete event systems. This results in a tradeoff between the size of the ADEC state space and the additional “insight” given by more sensors. An additional tradeoff occurs between the added insight and the sensing cost. It is possible to control the cane effectively with three sensors and a control system with eight states. Therefore only three sensors are being used.

The goal of the cane is to avoid obstacles. In terms of the control system, avoiding obstacles means that the goal state is always the state in which the system is unrestricted, $\gamma_1$ (where all inequalities are true). Therefore, based on the current state of the ADEC, controller commands are issued to reach state $\gamma_1$ or to remain in this state. These commands take the form of controller events $\nu(k)$, issued by the ADEC. In the example above, to avoid the obstacle towards the right, the control command issued would be a light left turn (provided no human input occurred at this time). This controller event, $\nu(k)$, is issued immediately after a new state is entered and remains fixed until there is another state change. The controller event changes when a new ADEC state is entered. The new controller event is a discrete representation of the desired turning angle. Note that in this case there is a one-to-one mapping between the controller event $\nu(k)$ and the ADEC state, $\gamma$. This implies that every time a particular ADEC state is entered (no matter what the previous state was) the same controller event is issued.

4.2.3 Human Discrete Event Model

Human interactions are not trivial to model [49] [74] [77]. There exists a variety of interactions which can occur at any time. The event-driven, asynchronous nature of discrete event systems [96] as well as the capability to model complex systems concisely makes discrete event theory appropriate to model the human interactions.

In Section 2.5 a general model of the HDEM is developed. This model categorises different interactions which the user can make with the control system into several classes. These classes are represented as states of an automaton. For the Robotic Cane, two categories are of importance. These categories are No Interaction and Discrete Input. Discrete input are commands that do not vary with time. The discrete commands utilised for the cane are turning commands
such as “Turn Left”. Although not implemented as part of the HDEM for the cane, it is possible to allow for continuous interaction by the human. Continuous inputs to the system could be steering commands which do not cause the wheels of the cane to turn to discrete angles (as defined in Section 4.4) but cause the wheels to turn more continuously.

To keep the structure presented in [5] the cane control system utilises a hierarchical structure of the HDEM and the Discrete Input state has sub-states of Go Straight, Turn Left, Turn Right and Approach. These sub-states correspond to the inputs which the human can make using the joystick.

The HDEM automaton is shown in Figure 4-8.

![Figure 4-8: HDEM automaton](image)

The human can let the system operate autonomously. The HDEM would therefore be in the No Interaction state, \(\delta_1\). However, what if the user wants to approach an obstacle that would normally be avoided? To interact the human issues one of the discrete commands using the joystick and the HDEM changes state into Discrete Input, \(\delta_3\). The discrete commands allowed are Approach \(\delta_3.1\), Turn Left \(\delta_3.2\), Turn Right \(\delta_3.3\) and Go Straight \(\delta_3.4\). States \(\delta_2\) and \(\delta_4\) are not used for the robotic cane.

As shown in Figure 4-5, inputs to the HDEM from the interface are events \(\kappa^A(k)\). These events can trigger state changes in the HDEM from the rest of the system. For example, the human input may be cancelled and the HDEM would return to the No Interaction state if an event had occurred which returns the ADEC to state \(\gamma_1\) (no obstacles sensed). This can be the case if the autonomous command overrides the input of the human. Commands from the HDEM to the interface are represented by \(\mathcal{H}\).
4.3. PROCESS MONITORING

4.2.4 Interface and Control Mapping

The fourth component, the Interface, provides a means of communication between the HDEM, ADEC and the plant. These cannot communicate directly because the state vector of the plant is continuous in time whereas the other two systems are of a discrete nature. Shown on Figure 4-5 the interface is bidirectional. In the ‘downward’ direction, the interface combines the commands from the human, $H$, with commands of the autonomous controller, $\nu(k)$, and maps these via $\phi$ to a combined continuous command for the plant, $\theta(t)$. The mapping $\phi$ is defined as

$$\theta(t) = \phi(\nu(k), H)$$  \hspace{0.5cm} (4.2)

In the reverse direction a process monitor, $\psi$, extracts information from the state vector of the plant, $x(t)$. This information consists of events for the ADEC, $\tau(k)$, and events for the HDEM, $\kappa^A(k)$.

$$\begin{bmatrix} \tau(k) \\ \kappa^A(k) \end{bmatrix} = \psi(x(t))$$  \hspace{0.5cm} (4.3)

4.3 Process Monitoring

To control the cane we must know what state the autonomous controller is in and what the desired goal state of the controller is. In discrete event systems theory an initial state is defined and any state change is caused by an event. Therefore, we must be able to monitor the occurrence of events. Since the cane must avoid obstacles we use data from the ultrasonic distance sensors to determine if an event has occurred. As stated previously, a change in an inequality associated with one of the three sensors from true to false or vice versa indicates the occurrence of an event and therefore a state change in the ADES. The recognition of events is done by a process monitor, defined as $\psi$.

Three different types of process monitors are presented and tested in this thesis. The first is based on velocities relative to potential obstacles [14]. The inclusion of velocity of a mobile platform in determining when to steer has the advantage that braking distances are now incorporated. This is important when reaction time is a factor, which is the case for the cane because the human must follow a turn or slow down behind the cane. The second method utilises sensor confidence. Consider a robotic system moving in an environment solely sensed by distance sensors. If the sensors in a particular direction change often or rapidly (and the robot is not moving at comparable rate) then it can be said that the sensor confidence is low. On the other hand, if sensors are accurate they tend not to change rapidly and sensor confidence is high. By allowing motion in directions only where sensor confidence is high a useful method of obstacle
avoidance can be established. As the third process monitor, we combine speed measurements and sensor confidence to make the system more reliable than either method alone.

4.3.1 Velocity Dependent Monitoring

Consider an obstacle with which the robot is not allowed to collide. If the robot is heading toward the wall, action to avoid the wall must be taken sooner if the robot is travelling fast. This gives rise to a restriction based on the velocity perpendicular to the wall and the distance from the wall.

\[ |q(t)|^2 - 2a_{\text{max}} k_1 |q(t)| < 0 \]  

(4.4)

where \( a_{\text{max}} \) is the maximum deceleration of the system. This inequality gives rise to the velocity profile near a wall shown in Figure 4-9. The solid line represents the maximum allowable velocity perpendicular to the wall at a specific perpendicular distance from the wall. The shaded area represents free motion where the robot is not constrained. The parameter \( k_1 \) is a scaling constant set to 1.0 in most states of the system. By increasing \( k_1 \) the above test can be made more "lenient". This means that the inequality will remain true until higher speeds are reached or until we get closer to the obstacle. This ability to change \( k_1 \) allows the user to approach obstacles closer without an event being recognised and therefore turning away.

4.3.2 Sensor Confidence Dependent Monitoring

Sensor confidence can be used in determining when the system needs to react. In a system in which sensors are inaccurate or give conflicting readings sensor confidence can give a good measure of when the system needs to be more conservative. For example, if sensors readings
change by large amounts between samples, the confidence in the sensors may be low. In this situation the system must be constrained earlier because “trust” in sensors is low. Sensor confidence is defined as a measure of how often the sensors change given that the sensors did not move relative to the environment.

Let a confidence parameter, \( c(t) \), vary up and down depending on the recorded sensor data. This confidence parameter increases if the sensors give the same result with each sensor scan. The confidence decreases if the results of two scans differ. Not only does the sensor confidence help if poor sensors are employed, it has an additional effect in cluttered environments. For example, if the cane is moving in a cluttered environment the sensors change regularly as different obstacles are detected. Therefore the control system becomes more “cautious” and constraints are activated further from obstacles and at lower speeds. If the sensor readings are steady, in open environments for example, the cane becomes less “cautious” allowing the user to exercise more freedom in his path. The confidence parameter has limits of

\[
1.0 < c(t) < 100.0
\]

The level increases and decreases according to

\[
c(t) = c(t - \Delta t) + \begin{cases} +0.5\Delta t & \text{if } sensor(t) = sensor(t - \Delta t) \\ -1.0\Delta t & \text{if } sensor(t) \neq sensor(t - \Delta t) \end{cases}
\]

where \( \Delta t \) is the time interval between scans in seconds. From (4.6) it can be seen that the amount the confidence rises and falls depends on how often a new sensor scan is performed. The confidence level decreases twice as fast as increasing it. This difference in the increasing and decreasing rate was determined to be suitable from experimental results. It has the effect that, once the cane is not confident, it takes longer to regain confidence in the sensors.

![Figure 4-10: Confidence inequality vs. distance](image)

The inequality based on confidence is given by

\[
c(t)|q(t)| - k_2 \geq 0
\]
where \( k_2 \) is a constant. Equation (4.7) has the effect that if confidence, \( c(t) \), is low, the distance between the cane and an obstacle must be high without avoidance measures being taken. This effect is shown in Figure 4-10. In an obstacle avoidance situation this constraint has the effect that if the confidence is high, the distance to obstacles can now be small before the robot becomes constrained. If the confidence is low, the robot will become constrained further from potential obstacles.

### 4.3.3 Combined Velocity and Sensor Confidence

Sensor confidence alone can sometimes be misleading in providing a means of activating constraints. If sensor bandwidth is low but the environment is changing rapidly, the confidence can remain high as sensors are not changing quickly enough. In this case avoidance action may be taken too late thus causing a potentially dangerous situation. The combination of sensor confidence with another method is thus logical. In this case velocity monitoring is considered in conjunction with sensor confidence to improve the performance over either method on its own.

The sensor confidence is determined as in equations (4.5) and (4.6). The combination of the two methods sees the confidence inequality (4.7) integrated into equation (4.4). An obstacle is now assumed to be far enough away if

\[
|\dot{q}(t)|^2 - 2a_{\text{max}} c(t) k_3 |q(t)| \leq 0
\]

where \( k_3 \) is a scaling constant. Let us assume that \( c(t)k_3 = k_1 \) then inequality (4.8) is the same as the velocity constraint inequality, (4.4). This means that the behaviour of the cane is the same as with a velocity monitoring only. If the velocity is constant then the condition (4.8) has the same form as (4.7) and the behaviour is the same as with confidence monitoring only. Thus the two methods become integrated into one.

### 4.4 Control Synthesis

Each time there is state change in ADEC, i.e. one of the three sensor inequalities changes from true to false or vice versa, a new control command, \( \nu(k) \), is issued. Similarly a new \( \mathcal{H} \) is issued from the HDEM. These discrete control commands are mapped using the control mapping \( \phi \), Equation (4.2), to continuous control commands, \( \theta(t) \). A lookup table defines the control mapping for the cane. This table is broken into two parts for clarity according to

\[
\theta(t) = \phi''(\phi'(\nu(k), \mathcal{H}))
\]
For the cane, there is a one-to-one mapping between the ADEC states and the discrete control command $\nu(k)$. Similarly there is a one-to-one mapping from HDEM states to $H$. The first mapping, $\phi^\prime$, from $\nu(k)$ and $H$ (i.e. ADEC and HDEM states) to control commands is listed in Table 4.1. These commands are established according to what a natural course of action would be, except in cases of conflict which are further dealt with below. By "natural course of action" we refer to commands such as turn left if there is an obstacle on the right. The control commands which are listed in Table 4.1 are determined at the time of programming.

The control commands listed are then converted to commands suitable for the cane. This mapping, $\phi^\prime\prime$ is listed in Table 4.2. The table indicates which commands are used during autonomous operation (no human interaction), shared control or both. The turning angles are also determined and fixed at the time of programming. During a Stop the wheels are pointed in opposing directions. This not only physically inhibits the cane from rolling forwards but if the human continues to push the cane forwards, the friction between the cane wheels and the ground can easily be felt. This friction again alerts the user to stop walking. Similarly the command Brake & Left (or Right) turns the wheels either left or right but one wheel is turned less than the other so that friction is created therefore again alerting the user.

In table 4.1 conflicts are indicated. Conflicts arise if the human input is not identical to the command of the autonomous controller. An example of this is the user wanting to approach a closed door in order to open it. It is possible that the user commands the cane to go right even though an obstacle such as the door is sensed on the right. In such a case it must be determined whether to allow the user to override autonomous commands or to compromise. To facilitate the decision making, conflicts have been broken down into weak and strong conflicts.

Weak conflicts occur when the human control command is different, but not opposite to, the command issued by the ADEC. An example of a weak conflict is the case where the ADEC is in state $\gamma_7$ (an obstacle is sensed on the right) but the user wants to go straight. The autonomous command would normally be a light left turn. However the straight command from the human does not directly conflict with an obstacle on the left - a weak conflict.

In the case of a weak conflict a compromise would be reached since no immediate danger exists. The cane will not immediately hit an obstacle and therefore the cane is used to aid the user in his decision. In the given example, the robotic cane would then perform a brake & left instead of a light left turn which would occur under purely autonomous control.

A strong conflict exists when human input is opposite to the ADEC command. For example, if a wall was sensed on the left and the human wants to go left, then a strong conflict exists. In such a case a warning is sounded and the human is given ultimate control. It is left up to the
### 4.4. CONTROL SYNTHESIS

<table>
<thead>
<tr>
<th>$ADEC$ State $(\gamma(k))$</th>
<th>Controller Event $\nu(k)$</th>
<th>$HDEC$ State $(\delta(n))$</th>
<th>Combined Control Command</th>
<th>Conflict Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 $\gamma_1$ Straight</td>
<td>$\delta_1$ Approach</td>
<td>Straight</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Straight</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Left</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,4}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>1 1 0 $\gamma_2$ Light Left</td>
<td>$\delta_1$ Approach</td>
<td>Light Left</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Light Left</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Left</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>1 0 1 $\gamma_3$ Stop</td>
<td>$\delta_1$ Approach</td>
<td>Stop</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Stop</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Stop</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>1 0 0 $\gamma_4$ Left</td>
<td>$\delta_1$ Approach</td>
<td>Left</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Left</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Left</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>0 1 1 $\gamma_5$ Light Right</td>
<td>$\delta_1$ Approach</td>
<td>Light Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Light Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Light Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Weak</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,4}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>0 1 0 $\gamma_6$ Straight</td>
<td>$\delta_1$ Approach</td>
<td>Straight</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Straight</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>0 0 1 $\gamma_7$ Right</td>
<td>$\delta_1$ Approach</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Right</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>0 0 0 $\gamma_8$ Stop</td>
<td>$\delta_1$ Approach</td>
<td>Stop</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,1}$ Left</td>
<td>Stop</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{2,2}$ Right</td>
<td>Stop</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,3}$ Straight</td>
<td>Stop</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\delta_{3,4}$ Straight</td>
<td>Stop</td>
<td>Strong</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Control mapping, $\phi'(\nu(k), \mathcal{H})$, from $ADEC$ and $HDEC$ state to combined cane control command.
user to exercise caution. The cane surrenders control in the case of a strong conflict because it
is assumed that the user has some knowledge as to what he intends to do. It would therefore be
too disabling to the user if there was no mechanism for the cane to do as the user commands.
This allows the user to walk through doorways and approach obstacles if so desired.

### 4.5 Operation of the Robotic Cane and the Framework

Experiments were conducted to demonstrate the operation of the shared control framework. In
the first set of experimental data the cane was pushed through an environment without human
interaction. During the second experimental run the human inputs commands at various times,
sometimes conflicting with autonomous control commands from the ADEC.

Figure 4-11 shows the path which the cane travelled during autonomous operation. Figure 4-12
shows the sensor data and the state of the ADEC during the experiment. During the whole
experiment the state of the HDEM was No Interaction, $\delta_1$.

From time $t = 0$ to $t \approx 2.5$ no obstacles were sensed. The ADEC state remained in state $\gamma_1$.
The cane moved straight ahead in this interval. An obstacle was then detected by the right
sensor, sensor 0. The ADEC changed state to $\gamma_2$. The cane made a light left turn until the
cane sensed the obstacle in the corner at time $t \approx 6$. Sensor 0 did not receive the echo pulse at
time $t \approx 4$ and thus a glitch was recorded. For the next six seconds, until time $t \approx 12$ sensors
0 and 1 both detected either a wall or the obstacle. The ADEC state was $\gamma_4$ and a left turn
was executed. After $t \approx 12$ the wall was only “visible” to the right sensor (Sensor 0). A light
left turn was commanded. It can be seen from the path that the cane slowly turned away from
the wall until $t \approx 15$ at which time the cane was boxed in. All sensors were able to detect
an obstacle and ADEC state $\gamma_8$ was entered. The wheels were therefore turned in opposite directions and thus alerted the user to stop walking.

During this first experimental run the cane did not collide with any obstacles and it was easy for the blindfolded user to follow the cane as well as stop when required. Adverse steering commands from the short glitches had little effect and did not alter the direction of travel greatly; this can be seen from the path. The data from this experiment show that the cane discrete event control system operates effectively to avoid obstacles.

In the second experimental run, the user entered commands using the joystick at various times in order to approach obstacles. Figure 4-13 shows the recorded path of the cane. Figure 4-14 shows the ADEC states, sensor data and the discrete human input states, $\delta_1$, $\delta_{3.1} - \delta_{3.4}$.

From $t = 0$ to $t \approx 3$ no obstacles were sensed. The ADEC was in state $\gamma_1$, and the HDEM remained in the No Interaction state, $\delta_1$. From $t \approx 3$ to $t \approx 5$ the user pushed the joystick right and since no obstacles were sensed the cane turned right. The user then simply walked...
behind the cane, making a right turn this can be seen on the path plot. Note that during the time of user interaction, the ADEC state did not change whereas the HDEM state changed to Discrete Input - Right Turn, $\delta_{3,3}$. Shortly thereafter the wall was sensed by sensors 0 and 1, the ADEC state changed to $\gamma_4$ and the cane turned left. During this left turn a sensor glitch was also recorded. From $t \approx 10$ to $t \approx 20$ the user again issued a turn right command. Note that this is in strong conflict with the left turn command issued due to sensor data. An audio warning was sounded to alert the user of the conflict situation. It can be seen on the path of the cane that the cane turned around the corner.

At $t \approx 20$ the user no longer wanted to turn right thus the HDEM state changed to $\delta_1$. The walls remained visible to sensors 0 and 1 and the ADEC remained in state $\gamma_4$, turning left. At $t \approx 22$ the wall only remained visible to sensor 0, ADEC state $\gamma_2$, a light left turn command was therefore issued. The user then issued a Go Straight command, HDEM State $\delta_{3,4}$, to walk down the corridor at time $t \approx 24$. As this constitutes a weak conflict a left and brake command was issued. The turning of one wheel and slipping of the other alerted the user to the
weak conflict, and he followed the weak left turn. At time $t \approx 30$ the user ceased to issue the Go Straight command (HDEM state returned to $\delta_1$) and a light left turn resulted. Shortly thereafter the end of the corridor was sensed by all three sensors. The ADEC state changed to $\gamma_8$, and the stop command was issued. The human wanted to keep going in order to open a closed door. A Go Straight command was issued again causing an audio warning due to the strong conflict.

The experiments show how the control framework operates to avoid obstacles with or without human interaction. During the experiments the cane effectively helped the user to manoeuvre through an indoor environment. With a little practice the user could follow the direction in which the cane steered. This shows that the control mapping was defined effectively. The only command that was not so intuitive was the left and brake command used during a weak conflict. However, a different kinematic solution of the cane could achieve the turning and braking as desired. In times of strong conflict between the user input and the autonomous control command the human remained in control. The audio warning during these conflicts was effective in alerting the user.

## 4.6 Experiments using Velocity and Confidence

Several experiments were conducted using the robotic cane shown in Figure 4-1. These experiments were conducted to test the operation of methods utilising velocity and confidence as well as the combination of velocity and confidence for process monitoring.

### 4.6.1 Velocity Monitoring

The velocity monitoring implemented using equation (4.4) served to turn the cane and user away from an obstacle sooner if travelling at higher speeds. Shown in Figures 4-15 and 4-16 are two experiments with process monitoring based on velocity only. The user did not interact and thus the HDEM state remained in the No Interaction state, $\delta_1$ throughout the experiment. The $a_{max}$ parameter from Equation (4.4) was set to $2.0m/s^2$ and $k_1 = 1.0$.

Figure 4-15 shows operation with a fast approach speed. On the figure the length of each line represents the speed with which the cane was travelling. State transitions or events in the ADEC are indicated by the $\gamma_x \rightarrow \gamma_y$ labels, where $\gamma_x$ is the previous ADEC state and $\gamma_y$ the new state. It can be seen from the figure that an event occurred approx. 2m from the wall (inequality 3 associated with the left most sensor became false) with a speed of approx. 3m/s. This event caused the ADEC state to change from $\gamma_1$ to $\gamma_5$. The cane then executed a right
4.6. EXPERIMENTS USING VELOCITY AND CONFIDENCE

The state transitions from $\gamma_5$ to $\gamma_7$ and back are due to the middle sensor also receiving and echo from the wall.

Figure 4-15: Path of cane with a high speed approach

Figure 4-16 shows the above experiment repeated with a slow approach speed towards the wall. The speed here was approx. 0.8m/s. It can be seen that the cane did not turn away until 0.5m from the wall. In this case only inequality 3 (left sensor) became false ($ADEC$ state $\gamma_5$).

The experiment was successful in that the cane turned further from the wall at higher speeds. A disadvantage of this method comes about from reacting to obstacles further away and the sensors being less accurate at longer distances. This means that sensing errors increase and hence incorrect control commands become more frequent.
4.6. EXPERIMENTS USING VELOCITY AND CONFIDENCE

4.6.2 Sensor Confidence Monitoring

Experiments were conducted to test the monitoring based on sensor confidence. The approach to a wall with low and high confidence was tested. The constant, $k_2$, in inequality (4.7) was set to 30.0. This value causes the inequalities to be false at distances less than 0.3m even at maximum confidence. Therefore the cane will not get closer than than 0.3m to an obstacle without turning or stopping.

In the first experiment shown in Figure 4-17. The value of the confidence parameter is shown as the length of the line segments. The cane approached the wall in ADEC state $\gamma_1$. With a confidence of approximately 10.0% inequality 2 and 3 become false at approx. 2.0m from the wall. This causes an event and the ADEC enters state $\gamma_7$ and thus turns away from the wall.

In contrast Figure 4-18 shows the cane approaching the wall in state $\gamma_1$ with a confidence of 100.0% at the start. The confidence starts to drop as soon as the sensors pick up the changes in the distance from the wall. An event occurs when sensor 3 causes its inequality to become false at a distance of 1.2m from the wall with a confidence of approx. 20.0%. Soon after this event, inequality 2 becomes false, triggering another event so that the ADEC state changes from $\gamma_5$ to $\gamma_7$. The confidence continues to drop as various sensor echoes are received. The low confidence now ensures that inequalities already false remain false even at longer distance so that the cane remains turned away from the wall.

The sensor confidence operates well in uncluttered environments which allow the confidence to rise between obstacles. However, a disadvantage comes about in cluttered environments where the confidence tends to remain low. The cane will turn away from obstacles at long distances. In the experiment the cane turned away at a distance of 2.0m when the confidence was low. This reduces the space to move dramatically and becomes a means of control which is too
4.6. EXPERIMENTS USING VELOCITY AND CONFIDENCE

Restrictive in indoor environments where distances between obstacles (walls, doors, furniture and fittings) is rarely greater than 4m. This is one of the reasons why we combine this method with the velocity monitoring.

4.6.3 Combined Velocity and Sensor Confidence

The combination of the velocity and the sensor confidence methods for process monitoring were also tested. Four experiments were conducted, each showing the one of the four extremes possible with two variables. These four experiments and the outcomes are summarised in Table 4.3. During this set of experiments the scaling constant (4.8), $k_3$ was set such that

$$2.0 < a_{max} c(t) k_3 < 6.0$$

Figures 4-19 to 4-22 show the above four experiments. The line segments represent velocity only.

In the first experiment with high velocity and low confidence the expected outcome is that the cane is extra “cautious”. Therefore we expect the cane to turn away from the wall early. This experiment is shown in Figure 4-19. The wall is approached and it can be seen that an event
occurs at the point at which recording started ("Start" on the graph). This event caused the cane to turn away. First the leftmost sensor sensed an obstacle such that inequality 3 became false. Shortly thereafter the middle sensor also triggered an event and the wall was turned away from early.

The experiment with high velocity and high confidence is shown in Figure 4-20. However the turn away from the wall was initiated at 2m from the wall, closer than the previous experiment. This is because the higher confidence (as opposed to low confidence previously) allows the cane to get closer to obstacles. When the turn was initiated, both inequalities 2 and 3 became false. Inequality 2 became true shortly thereafter. At the end of the travelled path, the ADEC state changed back to $\gamma_1$. Here the cane was parallel to the wall and no obstacles were detected.

The third experiment had low velocity and low confidence. The low velocity normally allows the cane to get closer to walls. However the low confidence causes the cane to turn earlier.
than it would with a high confidence. This experiment is shown in Figure 4-21. Here the cane turned at a distance of approximately 1.0m. It can be seen from the figure that there were several transitions between ADEC states $\gamma_7$ and $\gamma_5$. These are caused by the middle sensor first sensing the wall parallel with the y-axis, then not sensing an obstacle and then sensing the wall parallel with the x-axis.

![Figure 4-21: Combined constraints: low velocity low confidence](image)

In the final experiment with the combined method, the cane turned away at a distance of 0.7m. This distance is the lowest distance. It is achieved with a high confidence and a low velocity. Both the low velocity and high confidence allow the cane to get close to an obstacle. It can be seen from Figure 4-22 that throughout the turn the cane was in state $\gamma_7$. This indicates that both the middle and left sensors caused their respective inequalities to be false. This state remains because as the wall is approached the confidence drops (due to sensor readings changing). The lowering of the confidence has the effect that the cane becomes more "sensitive", therefore ensuring that inequalities already false remain false longer, even if the distance from the obstacle increases slightly. This effect is essentially a form of hysteresis.

![Figure 4-22: Combined constraints: low velocity high confidence](image)
The results indicate that the method of combining velocity and sensor confidence is a valid way of performing process monitoring. The combined method also overcomes the disadvantage of sensor error of velocity sensing at longer distances as the confidence method is better at longer distances. At shorter distances where the confidence alone perform poorly, the velocity method helps. Therefore the combination adds a further degree of robustness to the control of the cane.

### 4.6.4 User Control Interaction

In the last set of experiments the human user interacted in order to reach alternate goals. Presented are two ways in which the user can interact. Both ways of user interaction are discrete interaction and thus in both cases the HDEM state changed from \( \delta_1 \) to \( \delta_3 \).

The first user interaction demonstrates that the user can interact with the system to make the cane go where the user wants or needs to go. Shown in Figure 4-23 is the trajectory which first operates autonomously (HDEM state \( \delta_1 \)). The ADEC state changes from \( \gamma_1 \) to \( \gamma_2 \) as the cane approaches the wall. The human then interacts and issues a discrete left turn command (\( \delta_1 \) to \( \delta_{3,2} \)). As this left turn command is not in conflict with any of the sensors the cane turns left. Shortly after the turn commenced an event occurs (the false inequality 1 becomes true) and the ADEC state returns to \( \gamma_1 \).

![Figure 4-23: Path of cane, with human interaction](image)

In the second example the human interacts by issuing the discrete approach command, HDEM state \( \delta_{3,1} \). This command causes the \( k_1 \) parameter in equation (4.4) to be set to 3.0. This has the effect that the control system becomes less conservative. This in turn allows the user to approach obstacles closer so that he may pass through closed doors. The path shown in Figure 4-24 depicts this experiment.
At first the cane is operating autonomously (HDEM state $\delta_1$). As the wall is approached the cane turns away (ADEC state change from $\gamma_1$ to $\gamma_7$). The cane continues its turn until the human issues the discrete approach command. He issues the command in order to approach the door which is approx 1.2m away. Here the HDEM state changes from $\delta_1$ to $\delta_{3,1}$. This makes the monitoring less "sensitive" because the $k_1$ parameter changes from 1 to 3. The reduced sensitivity causes an event which causes the ADEC state to change to $\gamma_1$. The user then pushes the cane straight to within 0.5m of the closed door. Note that the HDEM state is still $\delta_{3,1}$. At this point inequalities 2 and 3 became true again and ADEC state $\gamma_7$ is once again entered. The approach mode has come as close as possible and the cane now turned right, away from the door. However the user, wanting to travel through the door, approached it further by issuing another discrete command, turn left ($\delta_{3,2}$) to go through the doorway - which he is about to open. This left turn command is visible on the figure at the end of the path, just in front of the door.

The results showed that monitoring based on velocity works quite well on its own. Figures 4-15 and 4-16 show how the cane avoided the approaching wall earlier and faster, respectively. However, the velocity method presents a design tradeoff at high speeds because the larger, less accurate, distance measurements become more significant.

The confidence method alone also worked in avoiding obstacles. However, in environments which are changing regularly or are cluttered, the confidence remains low. For this reason the cane will often steer away earlier than is often necessary, thus reducing the available space. Figures 4-17 and 4-18 show the cane avoiding the wall earlier or later depending on the sensor confidence.

The cane performed well, operating at high velocity with high confidence parameters. The advantage of this method is that it overcame the difficulties associated with the velocity or
The first set of results demonstrate how a shared control system using the shared hybrid dynamic control framework presented in Chapter 2 was implemented. It was demonstrated how the framework was adapted to work with the robotic cane, a system in which shared control is vital in order for the cane to be an effective aid for a visually impaired person. The results also show how the framework operated and how the ADEC issued control commands according to the state it was in. The hierarchical structure of the HDEM was provided a useful means to distinguish between the four different types of discrete interactions the user could make. The design of the ADEC, HDEM and the plant was straightforward because the framework allowed these sub-systems to be designed individually. Only the interface design required knowledge of all three other sub-systems. The sharing of control allowed the user to negotiate closed doors which he would have been unable to do without a shared control system in place.

The second set of results showed how three different sensing strategies are used as a control scheme for the robotic cane. The results showed how the status of the inequalities are used to determine when events have occurred and thus which state the ADEC changes to. The constraints utilised the distance to obstacles, velocity of the cane and sensor confidence as a means for determining what the current ADEC state was.
Chapter 5

Performance Evaluation

A qualitative performance analysis of the shared control framework is presented in this chapter. There are two performance issues to be considered. The first factor is the fulfilment of the goal that shared control provides advantages over autonomous or human control alone. The second performance issue is the user’s impression on whether the shared control system is useful. The user’s impression is very subjective since factors such as motivation, alertness, fatigue, activity level and learning affect a person’s performance and therefore judgement at any given time [37].

5.1 The Shared Control System

The performance of a shared control system can be determined by implementing a task and then executing this task under three control mechanisms:

- under autonomous control
- under human control, and
- under shared control.

There are two measures that can be applied under each of the above three control mechanisms. The first measure is either the success or failure to complete the task. The second measure, provided the task is completed, is the amount of time it took to complete. These measures were chosen as these can be applied to the all three control mechanisms. For example a measure such as human workload does not apply in the autonomous control test. Similarly human workload would be 100% in the human control case. Other measures such as accuracy in following a path are also not particularly useful as these require an “ideal” path to be defined. Such an
"ideal" path cannot be defined as it is left entirely up to the human operator how he wishes to interact in the tasks listed below. Therefore measures such as time and success rate are useful in gauging the performance under each of the control mechanisms in each of the following three tasks.

The experiments implemented in this thesis were tested under each of the three control mechanisms which are summarised below:

- **Task 1** Is the experiment presented in Section 3.2 in which a moving obstacle (a model train) crosses the path of the robot. The robot must not collide with the obstacle; collision is considered a failure. The robot and train are unsynchronised and therefore collision danger exists. The robot must also move between two targets, A and B. In this case any autonomous control commands are achieved by the constraint method. Enforced is the limitation that both target areas are to remain fixed. The human can input a velocity command only.

- **Task 2** Is the same experiment as above only to be completed using the potential field method. Here a limitation enforced is that the user cannot create a new potential field such that the robot always moves over the train under autonomous control (although this is theoretically possible).

- **Task 3** This task utilises the Robotic Cane. A blindfolded user has to manoeuvre through an open, 1m wide door. Starting at 4m from the doorway the user is positioned so that if he were allowed to see, he would be able to see the door without turning his head. Success of the task is the collision free passage through the door. It is assumed that the user knows that he is more or less facing the open doorway.

The results of the above three experiments are presented in Figures 5-1 to 5-3. The figures show the average time taken to complete the task as well as the fastest and slowest times. Also shown are graphs of the success rate of the three tasks under autonomous control, human control and shared control.

It can be seen from Figures 5-1 and 5-2 that for Tasks 1 and 2, the autonomous process was relatively fast with an average time of 7.2 and 7.1 seconds for the constraint method and the potential field method respectively. However, the autonomous process only has a success rate of 87% and 86%. In these 13% or 14% of failures, the robot crashed into the train or vice versa. These crashes occur because there is no sensing of the train. Under human control the robot almost never crashes into the train. This is simply a factor of monitoring the system more effectively and knowing when to take avoidance measures. However, under human velocity...
5.1. THE SHARED CONTROL SYSTEM

Figure 5-1: Time taken and success rate for Task 1 under autonomous, human and shared control

Figure 5-2: Time taken and success rate for Task 2 under autonomous, human and shared control

input, the time taken to reach the target is larger by more than a factor of two compared to autonomous control. This large increase in time taken is associated with the difficulty for the human to localise the target exactly and slow the robot down sufficiently not to overshoot. This is a disadvantage easily overcome by the shared control. The robot operates autonomously for most of the time (85%-86%) and the remainder of cases the human operator monitors for errors and corrects them. Once the train has passed, the human simply surrenders control to the autonomous system by cancelling any continuous input (using a discrete command). The robot then takes over and accurately reaches the target in only slightly more time than under autonomous operation. The shorter minimum times are achieved by accelerating the robot quickly in front of the train, thus resulting in a time advantage. Similarly the times slower than those achieved under autonomous operation are due to the user slowing the robot down to let the train pass. From these two examples it can be seen that shared control is an effective means to reduce erroneous behaviour (more robust system) at only a slight increase in the time taken.
5.1. THE SHARED CONTROL SYSTEM

The third task utilised the Robotic Cane. To complete the task it was necessary to pass through an open door under the three control mechanisms (autonomous control, human control and shared control). The results are shown in Figure 5-3. Under autonomous control, the cane operated with a success rate of 70%. For the remaining 30% the cane had a tendency to steer away from the door and then subsequently away from the wall, therefore effectively doing a U-turn. The user had difficulty in negotiating the door without help. As mentioned, although blindfolded, the user does know approximately where the door is located and the 50% success rate was achieved only by feeling the way using his free hand to assist and the joystick to steer the cane where desired. When successful, the user took about 10 seconds to negotiate the door. Finally as with Task 1 and Task 2, the shared control system performed best. Provided the user has some sense of direction as to the door’s location then the shared control system works in 90% of cases. If the user steers the cane towards the general direction of the door and then leaves the cane to do the finer motion, the cane successfully steers itself and the user through the door. The 10% failure figure in the shared control system comes from the user trying to do too much too close to the door and usually the cane collided with the door frame.

It is apparent from the three figures that the success rate is improved under shared control. In Task 3 the success rate was improved to 90% as opposed to 70% and 50% in the autonomous and human control cases respectively. These examples either require the human to sense for the robot or require the robot to sense for the human. A perfect example of shared control occurs where the human and the robot enhance or replace each other’s capabilities. The accumulation of strengths in one sub-system helps to minimise the weaknesses of the other sub-system. In terms of time required for task completion in Tasks 1 and 2, the shortest time recorded under shared control was less than both the shortest autonomous time and human control time. The
average time required to complete the task was slightly higher than in the cases for autonomous control alone. This can be attributed to the user sometimes interacting longer than necessary. The slightly longer average is also due to the user sometimes waiting for the train to pass before allowing the robot to cross. In Task 3 the shared control time is 50% longer than autonomous control alone. This is due to the fact that when autonomous control achieves its goal it reaches it directly. The difficulty is that autonomous control does not reach its goal reliably. The extra time taken under human control is due to the cases where autonomous control would normally fail and the human has to “save” the cane. Overall, for a small increase in the time required to complete the task, a worthwhile increase in the success rate is achieved. This is why shared control is an important component of modern control systems.

5.2 Human Factors

An important part in measuring the performance of a shared control system is the operator’s view. To qualitatively assess the operation of the shared control system as seen through the operator, a variety of users were asked several questions after having used the shared control systems implemented in Tasks 1, 2 and 3 as described above. The people using the systems had no training and generally only used the system for short periods of time (one minute or less).

The questions asked were:

1. To complete the task, do you prefer
   a) autonomous control
   b) human control
   c) shared control?

2. Does the robot (autonomous control system) help in finding the goal?
   yes/sometimes/no

3. During command conflicts, do you prefer to
   a) let the robot operate autonomously
   b) have control yourself
   c) share the control (compromise)?

4. How did you find the user interface?
   intuitive/OK/difficult to use

5. Comments?
5.2. HUMAN FACTORS  

Let us first consider the results for Task 1 and Task 2, the manoeuvring experiment in which the robot must avoid a moving obstacle and a stationary obstacle. A summary of the responses given is presented below. Most users agreed that for the task to be successful shared control is needed. However, comments such as why not install another sensor to detect the train were also received. Although it is possible to install an additional sensors, the installation would remove the need for shared control thus ruining the testbed. The users predominantly answered “sometimes” to whether the robot helped in finding the goal. The reason for this becomes apparent from answers to question four, that the user interface was difficult to use. However, some comments such as “the ability to cancel the command given is great” indicate that the discrete command to cancel the continuous input is very useful. The discrete command is useful, because even if the human made an error in the continuous input, the discrete command returns autonomous operation thus giving full control to the autonomous system. 57% of the users preferred to let the robot operate autonomously in times of conflict, 11% preferred to have purely human control and 32% liked to share the control with the robot. These results indicate that the operator trusts the robot to complete the task successfully. However, the 32% figure indicates that the operator wishes to retain some control even during conflicts. It should be noted here that a good shared control system should make provisions to allow the robot to operate autonomously, let the human have control and compromise depending on the type of conflict. As indicated earlier, the answers to the quality of the user interface were predominantly “difficult to use” indicating that the velocity input is not very intuitive and could be improved.

With respect to Task 3, the Robotic Cane, the responses were also recorded and are summarised below. All users indicated that shared control is necessary. These responses all indicate that shared control is necessary in a device such as the Robotic Cane. The answers to question two were mostly “sometimes” and the remainder of the answers were “yes”. This can suggest either that the cane was not effective in avoiding obstacles or the cane does not know what the overall goal is. The latter is a design issue and it was not intended for the cane to have an ultimate goal but rather avoid obstacles in a localised sense. The cane sometimes collides with obstacles due to a mixture of sensing problems and the human not working with the cane. The user is expected to work with the cane and therefore turning his body the same direction the cane has turned, without this cooperation the cane is less effective. During control conflicts the survey indicated that people either wanted to share the control or have complete control over the device. This is in contrast to the answers for Task 1 and 2. It is assumed that because the human operator is directly involved (connected with the system) he therefore wishes to remain in control, at least partially. The user interface was generally perceived to be “OK”. Comments received suggested that an audible indication be given when the cane turns either left or right so that the user can make a better effort to turn when the cane is turning.
5.3 Conclusion

The general performance of the hybrid dynamic shared control system was examined together with the views of the users. The overall evaluation of the shared control system is good. Shared control draws on the advantages of the human and autonomous components and therefore provides a means to do things which either the human or the robot would not be able to accomplish alone.

The users in general agree that shared control is a useful control paradigm. The results indicate that most people are quite happy to let the machine do the work if the human is not directly coupled with the task. This of course depends on how safety critical the task is. If for example tasks 1 and 2 were to be performed in a nuclear power plant then answers to question three may have been different. In both cases the users agreed that the user interface is a critical component and that its design is a deciding factor in how easily a control system is interacted with.
Chapter 6

Conclusions and Further Research

6.1 Introduction

This chapter reviews the major results and contributions of the work presented in this thesis. First, the contributions of the shared control framework are summarised. The contributions of the unique human interaction model are presented. Then the methods for control synthesis utilised in this thesis are discussed. Finally, the practical aspects and conclusions reached from the Robotic Cane are presented with regard to the aims of the shared control framework.

Additionally, the work presented in this thesis opens areas for further research and discussion. The topics presented in this section are: 1) conflict resolution between the human and the autonomous controller; 2) performance analysis of both the efficiency of the model and human interaction processing; and 3) evaluation of the control system with regard to the five supervisory control functions should also be undertaken, in particular the function of learning.

6.2 Major Conclusions and Results

The main contributions of this thesis are divided into four areas. The first and most significant contribution is the extension of discrete event systems to allow for shared/supervisory control in a consistent framework. The shared discrete event control framework comprises of four sub-systems. This division allows the sub-systems to be designed and analysed separately.
The framework offers three main advantages. 1) It allows for more reliable control in poorly modelled or dynamic environments. 2) The use of discrete event systems provides a means to integrate shared control into systems too complex to model with conventional techniques. 3) The user can interact with the system on a continuous as well as a discrete level using one consistent framework.

By using this discrete event framework in the experiments provided, it was possible for the human to make intelligent decisions, such as selecting how to avoid obstacles, and then effectively communicating these decisions to the autonomous system. We looked at some fundamental interactions between the human, the human discrete event system, the autonomous discrete event controller and the continuous system. Successful information transfer was extremely important for the framework to operate.

The ability of human interaction allows machines with less external sensing to be used in a larger variety of tasks. Examples include, but are not limited to, manned or remote vehicle control, assembly line inspection, cleaning and maintenance. Many of these areas were previously thought of as too difficult. However, because the human input removes the need for a multitude of sensors to interact in unpredictable environments more areas are being opened up.

Although the user was now integrated into the system, a degree of separation was maintained in the form of subsystems which simplified the integration. The individual subsystems, the ADEC, HDEM and the continuous plant, were all treated and analysed as separate systems. Aiding the separation are sub-models which in themselves are discrete event or continuous systems and are only combined by an interface which does not change the structure of the other systems. Therefore existing control theory, discrete and continuous, could be applied to analyse the sub-systems. A further advantage is that when the human was interacting with the system, the continuous commands from the ADEC were still valid. Therefore, the autonomous process always continued whether the human had interfered or not.

The second major contribution of this thesis is the development of the human interaction model, the HDEM. The HDEM is defined according to information flow and human interactions, rather than the dynamics and abilities of the human. The human interaction model is modelled as an automaton with four primary states. These states define the limited interaction available to the user. These states are defined as No Interaction, Discrete Input, Continuous Input and Information Request. The advantage of this model is that existing discrete event control theory applies and it can therefore be easily integrated into the remaining control structure. This model has been extended to allow for a hierarchical structure. This expanded structure brings with it finer division of possible interactions thus better defining the control flow.
6.3. FURTHER RESEARCH

The thesis presents two methods for command synthesis. The first method, based on constraints, was adopted from existing control theory. The method was extended to offer shared control between the HDEM commands and the ADEC commands. The method was also utilised to restrict the human input to safe levels or to guide human input along a predefined path. This method is attractive to discrete event control as the definition of the constraints often depends on the state definition of the system. A disadvantage of this method is the potential for conflicting constraints which can result in an invalid control solution. The second method is based on potential field theory, a control strategy typically used for navigation and already well established. This method has been coupled to discrete event systems for the first time in this thesis. This method also provides for limiting or guiding human input. However, the design of the potential fields to control the event trajectory is more involved than the definition of the constraints.

The method was demonstrated by two sets of physical implementation. The first set of experiments, performed with a five degree of freedom manipulator, demonstrated the applicability to and the capability of the shared control framework in a simple manipulation task. The experiments with the manipulator were also used to demonstrate the operation of both control synthesis methods with or without continuous human interaction. Additional support and extension of some of the theory presented was provided by a case study, the Robotic Cane. The cane demonstrated the successful operation of the shared control framework. A control method utilising thresholds based on velocity and confidence constraints ensured that the cane avoided obstacles successfully. Discrete interactions provided a convenient means for the user to interact in this case.

To conclude, the issue of human interaction in otherwise autonomous control systems deserves attention. It is an issue which cannot be ignored, not only because academia is interested, but also because there is a need. The need arises from the drive toward building machines that can do more than one thing, adaptable machines. Shared control should be considered from both the continuous domain as well as the discrete. The discrete domain however offers advantages such as modelling capabilities that surpass the conventional continuous methods.

6.3 Further Research

6.3.1 Conflict Resolution

The discrete shared control framework has two sources of control commands. The sources are the ADEC and the HDEM. In effect this system is a distributed control system with two controllers.
It is therefore possible that the control command issued by the ADEC conflicts with that of the HDEM. Note that the command from the HDEM is based on human interactions. The issues related to control conflicts are (1) to identify a conflict; and (2) investigate resolution techniques. Ultimately the conflict resolution problem has to resolve whether to trust the machine or the human operator.

Are there different classes of conflict? These could include conflicts in modelling, monitoring and control. The class of control conflicts includes three areas of potential conflict. The first is a difference in task goals. The task goal is represented as a discrete state and either the human or the ADEC must determine this goal. The operator and the ADEC could also negotiate a new goal, but finally there must be only one goal state. The second control conflict area is the sequence of discrete events (or states) that should be traversed in order to reach the goal. Often there are two ways to complete a task. The operator and the ADEC must again negotiate on which path is best. The third area of conflict is the mapping of the discrete control commands to continuous commands to get from one discrete state to the next.

Should we give the human ultimate control or should there be a command compromise? The problem of how much the human can be trusted is a psychological problem and is probably very dependent on the end application of the control system.

If the decision of the controller is to compromise, then it becomes an issue of what weights should be assigned to each of the commands. The weights may be different depending on the current state of the ADEC or HDEM. In the case of the Robotic Cane, the different classes of conflict resulted in different weights being assigned to the commands from the ADEC and the HDEM.

A sound theoretical conflict model needs to be developed here.

6.3.2 Five Supervisory Control Functions

The five supervisory control functions are planning, teaching, monitoring, intervening and learning. During the demonstration of the shared control framework by the experiments, planning is addressed. The planning function relies on the user being able to set intermittent goals or change the goal of the task permanently. Teaching is also addressed to some extent. Monitoring of the system is made possible by the Information Request state. Intervention has been proven through discrete and continuous inputs. It is the learning function that requires further attention. Although the framework is capable of supporting learning by human supervisors, this has not been implemented or demonstrated. Learning by the human supervisor implies that the person be taught what actions are allowed and what actions are not. Some type of feedback
is necessary. The ability of the constraints or the potential fields to limit user interaction is an important advance in providing feedback to the user.

6.3.3 Analytical Research

In this thesis an important advance in discrete event systems is made. A shared control framework is presented. However, the research is application driven and certain theoretical details have not been developed fully for a general description. The results demonstrate that the framework is operational and that it effectively integrates human decision making with discrete event systems. The operation of the framework is not demonstrated in an analytical sense. A complete analysis would be a challenging research objective.
Bibliography


[68] Christine M. Mitchell and Gunilla A. Sundström. Editorial, human interaction with com-
plex systems: Design issues and research approaches. IEEE Transactions on Systems,

based on nonholonomic elements. In Proceedings of the IEEE International Conference on


In Proceedings of the IEEE International Conference on Robotics and Automation, pages
802–807, Atlanta, Georgia, April 1993.

[72] Catherine Rafflin and PR. Alain Fournier. Learning with a friendly interactive robot for

[73] Peter J. Ramadge and W. Murray Wonham. The control of discrete event systems. Pro-


[76] L. B. Rosenberg. Virtual Fixtures: Perceptual overlays enhance operator performance in
telepresence tasks. PhD thesis, Department of Mechanical Engineering, Stanford Univer-


of mental workload and human performance in complex systems. IEEE Transactions on

[80] Alberto Rovetta, Remo Sala, Xia Wen, and Arianna Togno. Remote control in telerobotic
July 1996.


Appendix A

Experimental Setup

This appendix presents the equipment and software used to implement experiments performed, other than the robotic cane. The experiments were conducted using an Eshed Scrobot VII and a joystick utilising a force sensor for human input.

The computing hardware required to control the robot and take readings from the joystick are two Motorola VME162 boards with a 68040 processors. These VME boards both run the VxWorks operating system and control the robots. A Sun workstation is used as an overall controller. Communication between the modules is performed via an Ethernet local area network.

Described is an overview of the control software running on the Sun workstation as well as the VME boards. The communication process between the workstation and the VME boards was performed using a client/server approach.

An overview of the system is shown in Figure A-1. The Ethernet connects the workstation and the VME boards. The VME boards are an interface between network and the robot as well as the global robot controller. The VME boards communicate with the joint controllers which interface to the motors and sensors on the robot.

A.1 The Robot and the joystick

Figure A-2 shows the 5 degree of freedom Eshed Scrobot VII robot. Its joints are controlled by individual joint controllers running PID controllers. These joint controllers also keep track of the position of the joint. These controller communicate via a serial RS422 link with the VME board.
A.2 Computing Hardware

The processing is done on three platforms. These are the Sun workstation and two VME boards. These two different units are described further below:

- The Sun workstation runs Solaris 5.2, a unix-based operating system. One workstation is used to co-ordinate and control the experiment. The workstation runs a client which communicates with the two VME boards. The client also maintains the ADEC and HDEM. It also performs data logging and trajectory calculation (Chapter 3.1.4).

- Each of VME boards runs the VxWorks real-time operating system. The CPU is a Motorola mc 68040-40MHz. The VxWorks environment allows easy multitasking facilities which are important to implement the server functionality required in our system. Each of the VME boards runs a server with which the client communicates via a message passing system. The first VME board controls the robot according to messages received. It also
Figure A-2: The Eshed Scorbot

runs a PID controller to achieve position and velocity control of the robot. The required joint positions are sent to the individual joint controllers which set the PWM of the supply voltages to control the robot. This VME board also returns messages containing robot position and discrete events (Chapter 2) when they have occurred which are used further by the client. The second VME board processes data from the joystick. This VME board also processes messages from the client and sends the force data to the client, in form of a message, if asked.

Communication between these individual processors occurs via an ethernet local area network.

A.3 Client and Server Software

An overview of the software running on the hardware components outlined previously is shown in Figure A-4. The communication protocol between the client and the servers, a message system, is an integral part of the servers and the client. Additional to this message system, the server and the client perform additional functions as described below.

The force sensor server provides the functionality to execute commands received via the message exchange system. The main functions provided are: read forces, setting/resetting of offsets and coordinate transformations. These functions are all implemented on one of the VME boards.
The robot server provides the user with a straightforward interface to make the robot perform the following functions: move with velocity $v$, goto point $x$ and initialise. The server can also keep records of the current robot position, velocity and acceleration in joint and task space. The gains in the PID controller to perform the above functions can also be altered through the server with the appropriate message. The server can also recognise discrete events when they occur. These are relayed back to the client.

The client is represented by a flowchart shown in Figure A-5. It communicates with the servers and orchestrates the overall control system. The client implements both discrete event control systems, the ADEC and the HDEM.
Figure A-4: Software modules running on the hardware components
A.3. CLIENT AND SERVER SOFTWARE

Initialise robot and force sensor

Get position and force data from servers

Did a HDES event occur?

Yes

Determine HDES event

No

Did an ADES event occur?

Yes

Determine ADES event

No

Does it affect HDES?

Yes

No

Calculate velocity command

Send motion request to robot server

Figure A-5: Flowchart of the client software