AUTOMATIC ROBOT PATH PLANNING WITH CONSTRAINTS

by

David Adrian Sanders

20 August, 1990

A thesis submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy of the Council for National Academic Awards, following research conducted by the author at the School of Systems Engineering, Portsmouth Polytechnic.

The author has not been a registered candidate for another award of the Council for Academic Awards, or of a University during the research programme.
ABSTRACT

AUTOMATIC ROBOT PATH PLANNING WITH CONSTRAINTS

by D.A. Sanders

In a complex and flexible manufacturing environment, tasks may be dynamically reconfigured. In this situation a robot needs to plan paths automatically to avoid obstacles and rendezvous with changing target points. A novel path planning system is presented which takes into account both kinematic and dynamic constraints. The main part of the system comprises a robot "Path Planner" and "Path Adapter", both using a dynamic "World Model" updated by a vision system. The Path Planner contains a geometric model of the static environment and the robot. Given a task, the Path Planner calculates an efficient collision free path. This is passed to the control computer where a trajectory is generated.

Pre-determination of optimum paths using established techniques frequently involve unacceptably high time penalties. To overcome this problem the automatic path refinement techniques employed avoid the necessity for optimality before beginning a movement. Repeated improvements to the sub optimal paths initially generated by the Path Planner are made until the robot is ready to begin the new path. Algorithms are presented which give a rapid solution for simplified obstacle models. The algorithms are robust and are especially suitable for repetitive robot tasks.

Within the Path Planner, the robot structure is modelled as connected cylinders and spheres and the range of robot motion is quantised. The robot path, calculated initially only takes account of geometric, kinematic and obstacle constraints. Although this path is sub optimal, the calculation time is short. The path avoids obstacles and seeks the "shortest" path in terms of total actuator movement. Several of the new path planning methods presented employ a local method, taking a "best guess" at a path through a 2-D space for two joints and then calculating a path for the third joint such that obstacles are avoided. A different approach is global and depends on searching a 3-D graph of quantised joint space.

The Path Planner works in real time. If there is enough time available a "Path Adapter" modifies the planned path in an effort to improve the path subject to selected criteria. The Path Adapter considers dynamic constraints. The first robot path improvement method depends on detecting the joint motor currents in order to minimise changes in joint direction, the other is based on a set of adaptive rules based on simplified dynamic software models of the robot stored within the planning computer. The adapted path is passed to the control computer.

The static model of the robot work-cell is held in computer memory as several solid polyhedra. With the aid of a vision system, this model is updated as new obstacles enter or leave the work-place. Overlapping spheres and 2-D slices in joint space are used to model obstacles. In this form the vision system can be updated quickly and the obstacle data can be accessed efficiently by the path planning and path improvement algorithms.
ACKNOWLEDGEMENTS

The author wishes to express his gratitude towards the Control and Robotics Research Group at Portsmouth Polytechnic. Specifically to:-

Professor John Billingsley for his guidance and inspiration throughout the work presented in this dissertation and for introducing me to the joys of wind surfing.

Mr Bing Luk for ALWAYS being available with helpful advice, an optimistic view and to assist in eating any food.

Mr and Mrs Ali Husseinmardi for help and advice on the more practical aspects of the early work and for introducing me to behtarin mehmannavazie Irani.

Mr Dave Robinson for his constructive advice and for introducing me to the extra curricular items available at research conferences abroad.

Dr David Harrison for helping to persuade me to begin and for reading most of this.

Mr Ron Dadd for his advice and naval anecdotes during the project.

Also to Dr Fazel Naghdy, Mr Barry Haynes, Mr Paul Strickland, Mr Ian Stott, Mr Abraham Mazharsolook, Mr Peter Harris, Dr Jim Hollis, Mr Kevin West, Mr Arthur Collie, Mr Tom Cronk, Mr Antony Moore, for creating a good working environment and for putting up with me!

and to Ms Karen Covell for proof reading all this.

The work was partially supported by a grant from the Science and Engineering Research Council.

The research was conducted at Portsmouth Polytechnic, UK over the period January 1985 to December 1986 and from July 1987 to August 1990.
Definition of an Industrial Robot

There are several definitions of the term industrial robot. In this dissertation the definition of the British Robot Association is used:

*The industrial robot is a reprogrammable device designed to both manipulate and transport parts, tools or specialised manufacturing implements through variable programmed motions for the performance of specific manufacturing tasks.*

This is stated more briefly by the Department of Trade and Industry as:

*A robot is a reprogrammable mechanical manipulator.*

Other definitions are:

The Robotics Institute of America, (now the Robotics Industries Association).

*A robot is a reprogrammable multi-function manipulator designed to move materials, parts, tools, or specialised devices, through variable programmed motions for the performance of a variety of tasks.*

The marketing division of the Sirius Cybernetics Corporation. *Adams (1979)*

*Your Plastic Pal who's Fun to be with!*
CONTENTS

Title Page. i
Abstract. ii
Acknowledgements. iii
Definition of an Industrial Robot. iv
Contents. v

1. Introduction.
   1.1 Automatic Robot Path Planning: The Requirement. 1
   1.2 Automatic Robot Path Planning: A Description. 4
   1.3 Automatic Robot Path Planning: The Work Completed. 9
   1.4 The Robot Path: Obstacle Detection. 12
   1.5 The Robot Path: Planning in Joint or Cartesian space. 17
   1.6 The Robot Path: Criteria for Improvement. 21

2. Literature Survey and Background to the Research.
   2.1 Introduction. 23
   2.2 Modelling: Obstacles and the Static Environment. 25
   2.3 Modelling: The Robot. 28
   2.4 Path Planning. 30
   2.5 Robot Path Optimisation. 40

   3.1 Introduction. 48
   3.2 The Development of the Apparatus. 50
   3.3 The Development of the Systems. 54
   3.4 The Final Apparatus: Communications. 61
   3.5 The Final Apparatus: The G64 Bus. 63
   3.6 The Final Apparatus: The Mitsubishi RM.501 Robot. 66
   3.7 The Final Apparatus: The Robot Controller. 74
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Modelling of the robot and obstacles.</td>
<td>4.1</td>
<td>Introduction.</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>4.2</td>
<td>The Static Environment.</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>4.3</td>
<td>The robot.</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>4.4</td>
<td>Dynamic Obstacles.</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>4.5</td>
<td>The Transformation into Joint Space: Introduction.</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>4.6</td>
<td>The Transformation into Joint Space: Spheres.</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>4.7</td>
<td>The Transformation into Joint Space: 2-D Slices.</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>4.8</td>
<td>The Transformation into Joint Space: Other Models.</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>4.9</td>
<td>Results.</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>4.10</td>
<td>Discussion and Conclusions.</td>
<td>108</td>
</tr>
<tr>
<td>5. Image Data Processing and The Vision System.</td>
<td>5.1</td>
<td>Introduction.</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
<td>Overview of the Final Apparatus (Including the Vision System).</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
<td>Obstacle Detection: The Configuration of the Apparatus.</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>5.4</td>
<td>Obstacle Detection: Low Level Vision techniques.</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>Obstacle Detection: High Level Vision techniques.</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>5.6</td>
<td>Results.</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>5.7</td>
<td>Discussion and Conclusions.</td>
<td>146</td>
</tr>
<tr>
<td>6. Robot Path Planning.</td>
<td>6.1</td>
<td>Introduction.</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>6.2</td>
<td>Path Planning for a Single Link Manipulator.</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>6.3</td>
<td>Extension to 3-SPACE Local Heuristic Methods.</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>6.4</td>
<td>Extension to a 3-SPACE Global Method.</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td>6.5</td>
<td>Trajectory Generation.</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
<td>Results.</td>
<td>172</td>
</tr>
<tr>
<td></td>
<td>6.7</td>
<td>Discussion and Conclusions.</td>
<td>178</td>
</tr>
</tbody>
</table>
7. Robot Path Improvement to Minimise Peaks in Joint Motor Currents.

7.1 Introduction. 180
7.2 Monitoring of the Motor Drive Currents. 183
7.3 Path Improvement: To reduce changes in joint direction. 184
7.4 Path Improvement: To Include Force Sensing. 193
7.5 Results. 198
7.6 Discussion and Conclusions. 200

8. Robot Path Improvement Considering the Manipulator Dynamics.

8.1 Introduction. 202
8.2 The Dynamic Model: The Lagrangian Formulation. 205
8.3 The Dynamic Model: The Formulation of the Experiment. 213
8.4 The Dynamic Model: The Experimental Method. 216
8.5 The Dynamic Model: Results. 219
8.6 Application of the Model: The Improvement Method. 229
8.7 Application of the Model: Results. 233
8.8 Discussion and Conclusions. 235

9. Discussion Conclusions and Future Work.

9.1 Introduction. 237
9.2 The Systems and the Apparatus. 238
9.3 Modelling of the robot and obstacles. 239
9.4 Image Data Processing and The Vision System. 240
9.5 Automatic Robot Path Planning. 242
9.6 Path Improvement to Minimise Peaks in Joint Motor Currents. 244
9.7 Path Improvement Considering Manipulator Dynamics Equations. 245

A list of Relevant publications and presentations by the author. 247

Appendices  A: The Mixer and the DC Servo Amplifier. 249
B: The Detail of the Transformation Programs. 258
C: Edge Following and Line Fitting. 262
D: The Mechanical Detail of the Manipulators. 264

References. 274

Specifications of the Mitsubishi RM 501 Robot. 281
Chapter One

INTRODUCTION

1.1 Automatic Robot Path Planning: The Requirement.

Industrial Robots are generally more costly than dedicated handling machinery for pick and place and assembly operations. Their claim to superiority is through their ability to be reprogrammed to carry out a variety of tasks, but when robots are used, they tend to only be programmed for a finite repertoire of tasks. They have little autonomy. Although Robot technology is suitable for many areas of industry, including Flexible Manufacturing Systems (FMS), at present the costs associated with robot installation and their perceived programming complexity exclude them from many applications [Policy Studies Institute(1986)].

This dissertation describes improvements to robot programming and control methods that will allow robots to deal with unexpected situations in unstructured environments. The methods will decrease the complexity of installation and reprogramming while reducing the associated set-up and running costs. These improvements will help to justify their use in smaller factories and for new and wider applications including small batch manufacturing.

The applications suitable for robots are becoming increasingly more complex and industrial pressures are to use the technology efficiently and to reduce process down time. At present path planning is carried out by human operators or by human programmers on "Off Line" systems who construct programs to carry out a task in one of three ways:
(a) **Lead through:** The robot is directed along the path by a human operator and a computer records the joint coordinates at discrete intervals or at specific configurations.

(b) **Teach Pendant:** The robot is directed along the path by a human operator using manual controls, usually switches. The coordinates are recorded at discrete intervals.

(c) **Off-line programming:** The path is defined from a computer simulation of the robot and the work-place. The trajectory locus is then down-loaded to the robot controller.

Using these methods to develop new robot programs can be expensive and tedious. Programming and reprogramming forms a large part of the total cost associated with establishing a robot cell. This must be less than the cost of completing the task by other means. When robots are used for repetitive jobs, the costs are spread over many operations and this has contributed to limiting the use of robots to mainly repetitive tasks.

In all three methods of programming, including off line programming, (even using CAD packages with interactive graphics), it is the responsibility of the operator to choose the via points so that a robot path both avoids collisions and is efficient.

In the future these programming methods will not be satisfactory. Complex robot systems as part of FMS will have to adapt to new tasks in the work-place on line and without human intervention. Other applications make it
difficult for a human programmer to intervene, for example work in undersea, nuclear and space environments.

In his book, Craig (1989) states that most robot applications still involve repetitive tasks. Udupa (1977) introduced the infrequent initialisation hypothesis which says that in general, a large number of robot trajectories will tend to be planned in any given environment before the environment changes. This suggests robot paths are especially suited to on line reprogramming and the automatic and adaptive path planning procedures described in this dissertation.

As part of his research, Kumar (1988) completed a short study of robot programming requirements and described the following desirable characteristics for a robot system. Robots must:

(a) be capable of successful and precise execution of a specified task.
(b) be versatile and able to adapt to different tasks, as well as to a changing environment.
(c) execute the tasks in the most efficient manner, where the definition of efficiency could be flexible.

The work presented in this dissertation mainly concerns the improvement of characteristics (b) and (c), including new methods of automatically programming robots. The methods have the advantages of eliminating programming cost for new paths, reducing down-time and set-up time and allowing robots to be used for changing tasks in changing environments. Paths are automatically reprogrammed between tasks and whenever the environment changes. These paths can then be adapted to improve the
path. The result can be expected to be safer and more efficient in comparison with the other programming methods discussed.

The ability automatically to plan collision-free motions for a robot manipulator is one of the capabilities required to achieve task-level robot programming. Task-level programming is one of the principal goals of robotics research. It is the ability to specify the robot motions required to achieve a task in terms of task-level commands, such as:-

"Move box to Table"

rather than robot-level commands, such as:-

"GOTO 2.5, 6.3, 41.7, 36, 42, 90".

1.2 Automatic Robot Path Planning: A Description.

The path-planning problem, in its simplest form, is to find a path from a specified START configuration to a specified GOAL configuration that avoids collisions with obstacles. This problem is more complicated than the collision detection problem where a known robot configuration is tested for an impact. Automatic path planning is also dissimilar from on-line obstacle avoidance which entails revising a known robot path so as to circumvent unforeseen obstacles. Both these problems have now been solved for a number of particular robots and systems are in use in industry. The automatic path planning problem is still at the research stage.
Research in the USA and Europe into the subjects of "Path Planning" and "Trajectory Generation and Tracking" has developed along two parallel lines in isolation from each other. This dissertation presents work which attempts to cross this divide.

Generally a robot path planner performs the high-level functions, breaking the task into a sequence of smaller movements based on its knowledge of the environment and the capabilities of the manipulator. The inputs to the planner are a description of the state of the manipulator and the environment. Based on these inputs, it plans the task. The controller then moves the robot through the planned configurations.

Since processor speeds are limited, the planners have not as yet been expected to output a path at the speed which the controller can follow. A more practical arrangement has been for the "Path Planner" to work off-line and refuse new information during execution. All relevant information must be provided at the beginning of the planning program and the Planner plans the path for the whole task. In the work presented in this dissertation, the Path Planner works in near real time. The path is improved using information on the robot dynamics that has usually only been used in the past for the optimisation of robot trajectories.

The trend in robotics research has been to achieve the "Task" by partitioning the problem into three stages, the input to each stage being the output of its predecessor. The three stages may be considered as shown over the page:
Path Planning
Given a Task for a robot and a geometric description of its environment, plan a path that avoids collision with obstacles. The path is a function of space.

Trajectory Planning
Given a path to be followed by the end-effector, the actuator constraints and a dynamic description of the robot, find the positions and velocities of the joints to achieve the path. The trajectory locus describes trajectory curve in joint space and the trajectory specifies the robot configuration as a function of time and space.

Controller Trajectory Tracking
Given a trajectory and the dynamics of the robot manipulator, track the given trajectory by servoing the movement of physical actuators.

This suggests that in the future robot cells within fully automated manufacturing systems might be as shown in figure 1.1. These work cells will require automatic robot programming and reprogramming systems.

Figure 1.1: Robot Cell within a Fully Automated Manufacturing System.
Many papers on path planning have appeared in the literature during the 1980s. While much work has been done, progress towards efficient general techniques has been slow. This lack of progress appears to be due to the complexity of the problem. In computer complexity theory Hopcroft et al(1984(b)), Shwartz & Sharir(1984) and Shwartz & Yap(1987) have stated that the problem is exponential in the number of degrees of freedom.

The path planning problem has generally been regarded as a purely geometric problem and has tended to be considered by computer scientists. Path planning has involved building a geometric model of the world and a free moving object. These models are then used by procedures for determining spatial paths across the world model. Any system dynamics are ignored by this approach and few researchers have considered complex multi-link robot arms.

The trajectory planning problem has been considered by control engineers and assumes the path has already been planned. Trajectory planning is concerned with the manipulator dynamics, not with geometric world models.

Considerable advances have been made in Path Planning and Trajectory Planning, yet there are very few cases where any attempt has been made to combine the two. In recent years the need for combining them has been recognised, [Schwartz & Yap(1987), BRADY et al(1989)] but processor speeds and economic factors still dictate that the planning stage be isolated from the tracking stage, [BRADY et al(1982), Kumar(1988)].

Reducing the path planning problem to pure geometry allows a very precise problem statement and solution, but important non-geometric constraints are
not considered. Solving the geometric problem is computationally fast but in simple static environments, path optimisation may be more important than planning time. In a complex FMS solution speed is more important than path optimality.

It is accepted in this research that economic factors and processing speeds still remove the path planning problem from the controller and "On Line" optimal planning has not been possible. Instead, this dissertation presents automatic and adaptive reprogramming methods which consider dynamic constraints as well as geometric and obstacle constraints, but which do not attempt to optimise the path before beginning any movement. A path is found which is only improved if time is available.

In most published work dealing with off-line path planning techniques, the calculation time is not critical. This dissertation describes an automatic system which will work in real-time and allow the advantages of a truly flexible manufacturing system to be realised. In a highly flexible system, any changes require fast re-calculation of robot paths. The extent to which this can be achieved may determine the re-scheduling capabilities of an entire FMS.

In order to achieve real-time operation a compromise is initially made between the efficiency of the calculated path and the calculation time. For any path planning problem there is an optimum solution based on a chosen cost function. In a changing situation operational constraints make a faster sub-optimal solution more acceptable. In the method adopted for this work, initially a fast sub-optimal path is produced which is only improved if time allows. The sub-optimal solutions ensure that the calculated path is collision
free and tends to the shortest path in terms of total joint movement.

1.3 **Automatic Robot Path Planning: The work completed.**

Some research has been devoted to automatic path planning over the last five years, but few of the methods are simple enough and powerful enough to be practical. Algorithms are particularly scarce for robots with revolute joints, the most popular type of industrial robot.

With the exception of Khatib(1986) the robot path planning work completed in the past has required computation time that makes the robot wait before carrying out the planned trajectories. The methods presented in this dissertation allow the robot to continue working and new paths are automatically planned and improved as necessary.

Several methods of on line automatic robot path planning are described in this dissertation to provide a comparison of the two main classes of path finding algorithm, local, heuristic methods and global methods. The problems experienced by Khatib using the artificial repulsion approach are overcome and the initial work is similar to work presented by Balding & Preece(1986) and Balding(1987) in that specific configurations are represented by nodes within the local methods and the configuration space consists of a lattice of points, that is the space is discretised, within the global method. In this work each node in the lattice represents a small neighbourhood in configuration space and the total space will be called a **Configuration Space Graph** (CSG).

The local and heuristic path planning procedures produce real-time solutions
for a range of problems. They require little time to pre-process data to generate a 2-SPACE graph before searching for a path. The methods can be employed in circumstances where the environment changes frequently.

The global path planning method is established on a more rigorous mathematical treatment of the path finding problem. The method provides solutions which only consider the constraints of the obstacles and the restrictions of the world model. It requires time to pre-process data to generate the world model as a 3-SPACE graph in joint space but then furnishes real-time answers to a range of path planning problems.

Once a sub optimal path has been planned quickly by one of the path planning methods, the path is improved by considering some other constraints of the robot manipulator. Other constraints considered were the maximum joint velocities and accelerations and the robot dynamics.

The control computer initially receives input from the fast Path Planner and then from a slower but more efficient Path Adapter. The whole system accounts for both obstacles and dynamic constraints, and produces control signals for the robot actuators.

The remainder of this introduction discusses the preliminaries to the three major areas of automatic path planning considered in this dissertation to aid in understanding the background and literature survey presented in chapter two. The preliminaries are Obstacle Detection methods, the choice of space for Path Planning and criteria for Path Optimisation. Chapter two is an exceptionally extensive literature survey of the relevant previous work and
provides the background for the work described in the rest of the dissertation.

In Chapter three the development of the hardware and the systems is considered from the initial test rig to the final apparatus. As part of the work a novel parallel hierarchy control structure was developed and this is described. In chapter four the decisions on the types of model to use for the robot, obstacles and the environment is dealt with. In this chapter the new concept of using diverse models for different parts of the workplace is introduced. The models for the static environment were complex and time consuming while the changing environment was modelled in a fast, simple and novel way. In chapter five the vision system and methods of 3-D visual data processing and image acquisition are described, using the models selected from chapter four. Techniques are developed for incorporating the obstacle detection data into the decision making process of the "Path Planner".

Chapter six presents novel methods of automatic path planning. Several multi-degree of freedom Path Planning algorithms are described. These are developed from initial work considering two dimensional graphs of joint space. Several 3-SPACE methods presented are local and heuristic methods and one is the global method mentioned earlier in the dissertation.

In Chapter seven new methods of path improvement are considered which minimise peaks in the joint motor currents. The methods depend on detecting the joint motor currents in order to minimise changes in joint direction. Later work in this chapter considers the differences between transients caused by collision and those caused by changes in direction.
Identification of the parameters of the manipulator dynamics is studied in chapter eight. These parameters are used to influence the strategy of the Path Adapter. The identification procedure is demonstrated experimentally on a Mitsubishi RM-501 robot. This method depends on simplified dynamics models of the robot which are used to develop simple adaption rules. These rules are stored in the main computer and used to adapt the path in order to reduce the time taken to achieve the task. In the work described in the literature the dynamics of a manipulator have only been used to adapt a trajectory produced from some planned path. In the work described in this dissertation the dynamics are used at a higher level, in the Path Planner.

In Chapter nine the work is discussed and conclusions are presented, along with suggested future work. The algorithms presented have a number of advantages: they are simple to implement, are fast for robots with few degrees of freedom, can deal with robots having many degrees of freedom (including redundancy), and they can deal with confused and changing conditions.

Finally, it is demonstrated that the whole system can be employed for real-time control.

1.4 The Robot Path: Obstacle Detection.

The work-space of the manipulator includes all possible physical elements swept by the robot links as the robot joint angles vary from their minimum to maximum values. This work-space can include static and dynamic obstacles. Static work-space environments do not change with time and may be modelled by complex and accurate methods. Dynamic obstacles change with time and
even simple fixed sequence robots may require sophisticated obstacle detection and avoidance techniques to deal with them.

In automatic path planning, a prerequisite to circumventing any obstacle is to detect it. Various methods for detecting obstacles have been proposed by different authors including Udupa(1977) and Doty & Govindaraj(1982). These methods are now in use and are described generally in several text books, including:- Fu, Gonzalez & Lee(1987), Klafter, Chmielewski & Negin(1989) and Galbiati(1990).

Several detection methods were considered in this research, these were detection by:-

(a) Human Operator.
(b) Ranging.
(c) Force feedback.
(d) Vision Systems

These detection methods are considered in this section followed by a brief discussion and conclusions.

(a) **Detection By a Human Operator.** In this case the operator determines the obstacles in the work-space that may interfere with the manipulator. If the system is under direct operator control, this information is recognised by the operator and revised commands are sent to the robot.

If the trajectories are computer generated and computer controlled, then the
information about the obstacles has to be manually loaded into the computer to create a "World Model". During the planning of trajectories the computer checks against this stored information for possible collisions. This method is suitable only for environments which rarely change. Whenever the work-space environment changes, the system depends on the operator to update the information about the new work-space environment. The accuracy of the "World Model" depends solely on the operator and involves tedious and complicated surveying and entry of information.

(b) **Detection by Ranging.** Range detectors involve various technologies: light, acoustic, infra-red, etc. All employ a transceiver to update the world model. Some methods are described in Klafter, Chmielewski & Negin (1989)

Obstacle detection ranging systems take evasive action if the robot comes within the minimum safe distance from an obstacle. This method can be used for monitoring both static and dynamic environments.

Although the positional accuracy of ranging devices can be excellent, directivity of the transceivers creates 'blind' spots and limits the detection volume.

(c) **Detection by Force Feedback.** If unchecked the robot will try to overcome any obstacles in its path. In doing so, the various torques - both in the joints of the manipulator and the actuators of the various joints - will increase rapidly. These forces can be detected by using force detectors such as strain gauges placed at the joints [Raibert & Craig (1981)] or by monitoring the
actuator torques [Doty & Govindaraj(1982), Sanders(1987(a)), Luk et al(1988)]. On detection of abnormal changes, corrective action can be taken.

If the obstacle is movable, it will be forced to move and the torques in the manipulator momentarily increase to overcome the friction and inertia of the object. If the obstacle cannot be moved, the actuators will be torqued more and more until either the obstacle or the manipulator is damaged. Both these results are undesirable. There will be a momentary increase in the torques of the links that are moving at that instant. These torques exceed the torques that will be encountered during normal full load working conditions and evasive action can be taken.

(d) Detection By Vision. Vision systems comprise of one or more cameras, a controller and, often, special lighting equipment. Cameras are usually placed above the manipulator work-space. Typically the manipulator work-space is brightly lit and often back lit. The cameras constantly scan the work-space and pass the information to a computer where the information from other sensors may also be correlated. A snapshot of the work space is formed at discrete time intervals and this is described by Fairhurst(1988) and Galbiati(1990).

Since a vision system sees both the manipulator and the obstacles it is possible to use a feedback loop from the cameras to control the trajectory of the manipulator. Whenever a robot manipulator comes within the minimum clearance distance of an obstacle, corrective action can be taken. This needs a large amount of fast processing power, but there is no need to store information about the obstacles in large databases.
The main disadvantages of vision systems are their cost and complexity.

**Discussion:** Considering each of the four methods:-

Intervention by human operator after obstacle detection, (a), will be inefficient for complex, modern and future robot applications working in dynamic environments. However the initial data for the accurate static "World Model" can be entered by a human operator after detailed surveying and/or measurement.

The accuracy of detection by ranging, (b), depends on the number, configuration and accuracy of the transducers. To cover a whole work place a great many ranging devices would be required.

Detection by Force Sensing, (c), is cheap but is only suitable for low cost and very slow moving manipulators or as a "last resort" back-up protection mechanism. There is little application to the path planning problem, but the similar torques experienced as actuators change direction may be used within a path improvement algorithm.

Vision systems, (d), tend to be largely independent of the operator and there are several advantages to this method. Vision systems can monitor the manipulator and obstacles in a common universal frame. The system constantly keeps track of the work space environment of the manipulator and hence it can be used to monitor dynamic or frequently varying environments. The information can be used directly for collision detection or for updating the
Conclusions: In conclusion, a human operator is the most accurate source of data about the detailed static environment. This method was selected to enter the accurate data for the static work place where time constraints are less important. A vision system was selected to update the dynamic obstacles in the "World Model" as this detection method is global, covering the whole robot work-space area, while still being fast enough to work in real time. The disadvantages of methods (b) and (c) excluded their use for global obstacle detection, but the information from currents to the actuator joints is considered during the design of one of the new path optimisation methods discussed later in the dissertation.

1.5 The Robot Path: Planning in Joint or Cartesian Space.

Path planning problems are difficult partly because robots and obstacles are best described in different spaces. Path planning can take place in either space. Obstacles tend to be described in 3-D Cartesian space but robots tend to be time-consuming to describe in this space.

A typical robot is composed of a number of joints, and the movement from one configuration to another is accomplished by moving each joint. The state of a robot may be generally be defined as a vector specifying the various joint angles, \( \theta \). This representation is called a configuration and is a representation in joint space. In the case of the robot used in this research;

\[
\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T.
\]
If configurations are specified as a path in Cartesian Space, that is $[X,Y,Z,\theta,\epsilon,\phi]$, then the controller must compute the required robot joint configurations through inverse kinematics. If the path planner provides the reference path to the controller in real time, the controller must calculate the velocity transformations from cartesian space to joint space using complicated algorithms, usually using the inverse Jacobian. The computation of the inverse Jacobian is a non-trivial problem, particularly if it must be done at the frequency at which the controller must operate.

There are several instances where the path is specified in the Cartesian coordinates $[X,Y,Z,\theta,\epsilon,\phi]$, such as welding or parts handling. In such cases, the path must be transformed to joint coordinates $[\theta_1,\theta_2,\theta_3,\theta_4,\theta_5]^T$. The Jacobian is only a point transformation and there is no known functional transformation to map the entire path from task space to joint space. This means that the transformation must be accomplished at a certain number of discrete points. Taylor(1976) investigated this problem at Stanford.

A few controllers have been designed to operate at the task level, but these were slow and controlling at the joint level is simpler and tends to be faster. The actuator constraints and description of the robot dynamics is in terms of joint coordinates in joint space.

Several authors have made a once only transformation of the manipulator and its surroundings into some abstract space. Udupa(1977) enlarged obstacles by the width of the manipulator links to produce a 'primary map'. A transformation was applied which permitted the upper arm to be viewed as a point. The transformed space was called a primary chart and was a map of all
the positions of the end of the upper arm for which the upper arm was collision free. The secondary map was produced by enlarging the obstacles by the radius of the forearm.

The advantages of these transformations were that the path planning of a point or single line segment was much easier in these transformed spaces.

Lozano-Pérez & Wesly(1979) and Lozano-Pérez(1983) developed a method for the calculation of paths for polyhedral objects moving through a space littered with other polyhedral objects. The method involved transforming obstacles into an abstract space which he called Cspace. An example of how this method is used may be found in Red(1984). In that work the configuration space for a PUMA robot was calculated by a VAX mini computer. The configuration space was displayed graphically and the operator could plan a path for a point through this space. The path was then converted back into robot coordinates for execution of the task. The method worked off-line.

The configuration of a three-dimensional object may be specified by a six dimensional vector. The six dimensional space of configurations for an object Obj_A is denoted by Cspace_A. This contains all the information necessary to solve the find-path problem for Obj_A.

Lozano-Pérez reported that when an object was a three dimensional solid which was allowed to rotate, then a simple object Obj_A in real space became a complicated curved object in six dimensional Cspace. So he did not calculate such objects, instead he approximated objects by a series of two dimensional slices containing polyhedral shapes.
Brooks (1983(b)) represented two-dimensional (2-D) free space as a union of possibly overlapping generalised cones. The algorithm translated a polygonal moving body along the axes or spines of the generalised cones and rotated it at the intersections of the generalised cones. The algorithm was fast and generated paths that gave good clearance from obstacles. Brooks, together with Kuan, later improved the quality of the paths found by representing the 2-D free space as a union of generalised cones and convex polygons.

Brooks (1983(a)) transformed the space between the obstacles into freeways for the upper arm and payload of a robot. The two freeway spaces were searched concurrently with the constraint that the upper arm and payload were a fixed distance apart, due to the forearm. Brooks reduced the degree of freedom of the payload in order to simplify the problem.

The algorithm generated prisms of free space between obstacles. The obstacles were effectively only two and a half dimensional in that they had a two dimensional shape and a height. Thus a cube could be represented accurately but a tetrahedron could not.

Conclusion: The purpose of creating a different space through a transformation is to reduce the complexity of the path finding problem. Even with reduced complexity, none of the systems mentioned in this section achieved real time operation.

In joint space a robot configuration is represented by a point and the problem is reduced to finding a path for a set of connected single points through a set
of obstacles. The limits of the joints form the boundaries of the joint space. Path planning in this space is reduced to finding a collision free path for a point and this is a relatively simple problem.

There is little advantage to planning the robot path in Cartesian space, and many advantages to planning paths in joint space. This work uses joint space to plan the robot paths and transforms obstacles to this space. The methods are described in chapters five and six. Other work is described in Lozano-pérez(1983) and Faverjon(1984).

1.6 The Robot Path: Criteria for Improvement.

Given a task, the objective is to complete it efficiently. For path planning problems, this has usually been interpreted to be the shortest path. This has been dealt with in a few papers [Lozano-Pérez(1981), Sharir & Schorr(1984), Papadimitriou(1985), Bajaj & Moh(1988)], but these researchers tended to be concerned with free moving objects and not connected chains.

Industrial robots move to a desired goal by moving each joint individually. Any concept of shortest distance which would be relevant if the robot was a free moving disc or a sphere, is meaningless. For a robot, any definition of shortest distance may not be the best path in terms of safety or in terms of some dynamic performance such as minimum time or energy. It is also possible that any shortest path may not be possible.

The torque from the actuators must be considered for path improvement. In classical optimal control theory, the controller is designed in feedback form and typically, the dynamics of the robot can be expressed in the form
\[ \tau = D(d^2\theta/dt^2)(\theta) + h(d\theta/dt, \theta) + c(\theta) \]

where

\[ \begin{align*}
\tau &= \text{Vector of actuator torques.} \\
\theta &= \text{Vector of joint positions.} \\
D(d^2\theta/dt^2) &= \text{Inertia, acceleration-related symmetric matrix.} \\
h(d\theta/dt, \theta) &= \text{Nonlinear Coriolis and Centrifugal force vector.} \\
c(\theta) &= \text{Gravity loading force vector.}
\end{align*} \]

This expresses the nonlinear and coupled nature of the differential equations that describe the system. There are also other position, velocity and acceleration dependent constraints imposed on the system. This complexity means the "Path Adapter" cannot operate on-line at the speed of the robot control computer.

The actuator torques can also be read directly from currents from the servo amplifiers connected to the actuators. [Sanders et al(1987(c)).]

**Discussion:** In this work the Path Adapter will operate on a planned path to improve the robot performance in terms of some dynamic criteria, for example time.

No attempt will be made to solve all the dynamic equations in real time. Instead simplified dynamic models of the robot will be used to establish simple rules for the "Path Adapter". Direct measurement of the motor currents will also be used for path adaption. This work is presented in chapters seven and eight.
Chapter Two

LITERATURE SURVEY
AND BACKGROUND TO THE RESEARCH

2.1 Introduction

Automatic robot path planning algorithms must coordinate the essential aspects of the problem:

(a) Detecting obstacles.
(b) Appropriate representation of the robot and obstacles.
(c) Derivation of a suitable trajectory for the robot.

Detecting obstacles was considered in chapter one and the vision system is described in detail in chapter five. This chapter will consider (b) and (c).

Pieper(1969) investigated automatic programming for robots in the United States. Later work was completed by Udupa(1977) at CalTech and Widdoes in unpublished work at Stanford. This early research aimed to design a robot programmer for use in planetary exploration. Since then the major contributions in the field of path planning have come from Lozano-Pérez(1981 and 1985) at MIT and then IBM and Brooks(1983(a-d)). These both used polyhedral models to represent obstacles. This work has been extended more recently by Donald(1984 and 1987), Gouzenes(1984), Canny(1985 & 1987), Tseng(1987) and Hwang(1988). An alternative approach was reported by de Pennington et al(1983) where the mover was modelled by a series of interconnected spheres. Since then other authors have added to this research.
The main parts of path planning research systems have been:

(i) **The world model.**

(ii) **The path planning algorithms.**

(iii) **The output.**

The few research systems considering path planning algorithms for robots have used three inputs:

(a) **A geometric and kinematic description of a robot.**

(b) **A geometric description of the robot environment.**

(c) **The task description.**

The type of world model chosen to describe the robot environment has a considerable effect on the path planning algorithms and different types of models are discussed in Sections 2.2, (Modelling of the Obstacles and Environment) and 2.3, (The Robot Model).

Several different path planning methods have been proposed and these are discussed in Section 2.4, (Previous work in Path Planning). Finally, previous work in Path Optimisation is discussed in section 2.5.

Sections 2.2 and 2.3 provide the background to the original work presented in chapters four and five of this dissertation and sections 2.4 and 2.5 provide the background to the original work presented in chapters six, seven and eight.
2.2 Modelling: Obstacles and the Static Environment.

Many computer models are possible. In the fields of Computer Science, Artificial Intelligence and Robotics research the most popular method of representing objects has been by using "polyhedra". [Ahuja(1980), Lozano-Pérez(1981), Brooks(1983(a-d)), Schwartz & Sharir(1983(b)), Donald(1984), Sharir & Schorr(1984), Akman(1985), Leven(1985), and Dupont(1988)]

A polyhedron is a three dimensional solid figure with many planar faces. The edges where faces meet are linear. Most objects may be closely approximated by polyhedra and examples of programs which model moving objects and their environments by polyhedra is work by Lozano-Pérez(1983), Schwartz & Sharir(1983(a)), Hwang(1988) and GRASP described by Bonney(1985).

A polyhedron may be represented by a tree structure of edges, faces and vertices. An edge may be defined by its end points and a face may be defined by specifying its edges. The more complex the polyhedron the more edges, vertices and faces it has and hence the more data required to define it.

Having detected obstacles in the robot environment, a prerequisite to robot path planning is interference detection. Some work on interference detection among polyhedral solids was presented by Boyse(1979). To determine whether a polyhedron Poly_A intersected a polyhedron Poly_B, all the edges of Poly_A were tested to see if they intersected any of the sides of Poly_B. If Poly_A and Poly_B were simple cubes then each of the twelve edges of Poly_A had to be tested with each of the six faces of Poly_B. This gives a total of seventy two edge face tests. A test also had to be done to see if Poly_A was enclosed by Poly_B or vice versa.
"Solid modelling" has been used to represent the robot work-space. De Pennington(1983) used Constructive Solid Geometry (CSG). CSG models use simple shapes, called primitives, to produce complex and accurate representations of a robots' surroundings. The primitives fulfil particular mathematical properties, so that operations such as volume calculations and intersection checking can be carried out easily. An example is shown in chapter four.

Spatial occupancy enumeration (SOE), is another subset of solid modelling. Space is divided into a matrix of spatial cells. Each cell is defined either as containing an obstacle or free space. Ahuja(1980) and Dupont(1988) have shown that this method can be used to represent the path planning problem. In these works a tree structure was used to represent three dimensional space. Space was represented as a solid cubic block. This was subdivided into eight blocks. Each block was tested and given a "Colour Flag". A block was designated black if it was completely within an object, white if it was free space and grey if it contained object and space. Each grey block was then subdivided into another eight blocks. Recursive subdivision continued until a minimum sized block was reached. At this point any minimum sized grey blocks were designated as black.

To solve the collision detection problem using SOE the obstacle sets are calculated for the moving object and its surroundings. To detect collisions the two obstacle sets are compared, searching for two or more equivalent cells in the path to be black. The representation by a matrix of spatial cells has the advantage that it is convenient for computer storage, but the computing time required to generate the representations of a moving robot tends to be large.
Gilbert & Johnson (1985) and Khatib (1986) applied the method of representing obstacles by distance functions, in the case of Khatib, motivated by the electrostatic repulsion between like charges. For example the mover and obstacles could be represented by positive charges. This artificial potential repulsion approach was aimed at the local, short-term avoidance of obstacles in real time rather than automatic planning of robot paths. Although the algorithm does not quite solve the find-path problem, the use of repulsion force made this algorithm original and the system worked in near real time. The function tended to infinity as the point approached the surface and was zero beyond a certain distance from the obstacle. This representation had the advantage that the task of calculating the distance between the robot and the obstacle was replaced by the task of evaluating the simpler function. Compared to solid geometry or polyhedral models, these calculations were relatively fast.

The repulsion force was generated by a fictitious potential field around each obstacle due to a potential assigned to it. When any link of the robot arm approached an obstacle, a repulsive force pushed the link away from the obstacle. If \( P \) was the potential function used, and \( D \) was a function of the minimum distance between the link and the obstacle, then \( P \) became large as \( D \) became smaller, and became zero beyond a preset distance from the obstacle. The force on the mover because of any obstacle was calculated from the equation

\[
F = - \frac{dP}{dD} \frac{dD}{dx}
\]

where \( \mathbf{x} \) is the position vector of the mover.

A higher-level planner was assumed to generate the initial path needed.
Appropriate robot joint torques were calculated to follow the nominal trajectory and the force from the artificial potential field was incorporated to generate the final forces at the joints. This allowed each link of the robot to follow the nominal trajectory closely while avoiding any obstacles. The role of the artificial potential field was not to plan the path or the trajectory, but to bend it around obstacles. Khatib’s algorithm was notable because the local obstacle avoidance problem was realised at the lower control levels for real-time execution, instead of being included in the "Path Planner".

The main disadvantage was that only a limited number of obstacle shapes were available. Khatib stated "this potential is difficult to use for asymmetric obstacles where the separation between an obstacle’s surface and equipotential surfaces can vary widely".

Pieper(1969) used a world model consisting of simple solid primitives (cylinders and spheres). Cylinders could be joined to form composite obstacles and spheres were assumed to be supported by planes. These models were approximate but simple. Balding(1986) suggested that if the world were modelled just by spheres, intersection calculations could be greatly simplified and this is discussed in chapter four.

2.3 Modelling: The Robot.

Any robot consisting of a series of links and revolute joints may be represented by a general schematic model. If \( n \) is the number of robot joints, there are \( n \) coordinate frames which specify the robot configuration and variables which define joint positions in relation to the next.

Udupa(1977) simplified the geometric model of a modified Stanford Arm robot to connected lines, then one line and then a point, by using obstacle transformations. Before any obstacle transformations were carried out the basic robot model was defined as two connected cylinders. The advantages of this representation were that path planning for a line or cylinder was much easier than the more complicated shape of the real robot. The method of modelling by connected cylinders was also used by Balding(1986 & 1987) for a revolute robot.

Other methods used polyhedral representations of the mover [Brooks(1983(a)), Bonney et al(1985), Canny(1986), Hwang(1988)]. This is a very accurate method of representing the robot but the computational effort for calculating collisions and path planning is so large that as yet it is impossible to use this depiction for real-time calculations.

An efficient geometric method using connected spheres was proposed by de Pennington(1983). De Pennington was interested in collision avoidance rather than path planning. The method used a CSG solid model of the surroundings. The robot’s path was simulated and the swept volume of the robot-sphere model calculated. The robot swept volume and the obstacle volumes were compared and where intersections between volumes occurred, collisions were indicated. The spheres produced swept volumes of regularized cylinders or 'tori' under the restricted robot trajectories considered.
This method was unsuitable for automatic path planning and optimisation as the sweeping of the spheres was restricted to translational or rotational sweeping only. This excluded the movement of more than one joint at any one time and thus restricted the possible robot paths.

2.4 Path Planning.

The aim of robot path planning is to find a "trajectory locus" for a robot which will take it safely from one specified configuration to the next. The dichotomy of the problem is that the path produced should be as efficient as possible, but computer calculation time should be as short as possible.

Previous work in path planning has been governed by the obstacle and robot representations used. Usually, each path planning method may only be used with its own particular world model and robot model.

Most path planning research has not considered the case of open kinematic chains where all links in the chain must avoid collision. This situation is much more complicated than a free mover such as a mobile robot or unconstrained free moving shape. Often in the literature, the motion of the various simple movers considered has been restricted to pure translation, or to some mutually exclusive interleaving of translation and rotation. This dissertation solves the automatic path planning problem for the case of an open kinematic chain.

Donald(1984), Gouzanes(1984) and Balding(1987) divided path planning methods into two categories, local methods and global methods. All methods do not fit strictly into these categories but it is a useful categorisation and will
be used in this section. A table showing the evolution of Path Planning techniques is shown in figure 2.1.

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Method</th>
<th>Obstacle Representation</th>
<th>Mover Considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pieper</td>
<td>1969</td>
<td>Local Planning</td>
<td>Cylinders &amp; Spheres</td>
<td>Kinematic Chain</td>
</tr>
<tr>
<td>Udupa</td>
<td>1977</td>
<td>Local Planning</td>
<td>Polyhedral</td>
<td>JPLR (Modified Stanford arm)</td>
</tr>
<tr>
<td>Ahuja</td>
<td>1980</td>
<td>Local Planning</td>
<td>Polyhedral</td>
<td>3-D Octree</td>
</tr>
<tr>
<td>Lozano-Pérez</td>
<td>1981</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>Polyhedra with restricted motion</td>
</tr>
<tr>
<td>Brooks</td>
<td>1983</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>Polyhedra with restricted motion</td>
</tr>
<tr>
<td>Schwartz &amp; Sharir</td>
<td>1983</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>2-D Circles with restricted motion</td>
</tr>
<tr>
<td>De Pennington</td>
<td>1983</td>
<td>Collision Avoidance</td>
<td>CSG</td>
<td>Connected Spheres</td>
</tr>
<tr>
<td>Donald</td>
<td>1985</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>Cartesian Robots</td>
</tr>
<tr>
<td>Canny</td>
<td>1986</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>Polyhedra with restricted motion</td>
</tr>
<tr>
<td>Khatib</td>
<td>1986</td>
<td>Local Planning</td>
<td>Mathematical Functions</td>
<td>Kinematic Chain</td>
</tr>
<tr>
<td>Balding</td>
<td>1986</td>
<td>Local Planning</td>
<td>Spheres</td>
<td>Kinematic Chain</td>
</tr>
<tr>
<td>Tseng</td>
<td>1987</td>
<td>Global Planning</td>
<td>Polyhedra on floor surface</td>
<td>Kinematic Chain</td>
</tr>
<tr>
<td>Hwang</td>
<td>1988</td>
<td>Global Planning</td>
<td>Polyhedral</td>
<td>Polyhedra</td>
</tr>
<tr>
<td>Dupont</td>
<td>1988</td>
<td>Local Planning</td>
<td>Polyhedral</td>
<td>Kinematic Chain in a static cell</td>
</tr>
</tbody>
</table>

Figure 2.1: Past Work in Robot Path Planning
Local Path Planning Methods: Local methods use algorithms that find a path by repetitively considering configurations that are closer to the goal. When obstacles are encountered alternative strategies are tried, such as "Reverse and Move Left" or "Move Below". For local methods the problem is that of finding a series of intermediate positions connecting the Start and Goal configurations. The definition of the problem suggests that planning would be sensor based in any real time system. The advantage of these methods is that planning can take place when it is not possible to have a global world model. Local methods are often used in research for mobile robots and robots operating in unknown environments.

Pieper(1969) and later Balding(1986) utilised various heuristic procedures to move around a detected obstacle. For example the arm folded to move in front of an obstacle and extended to move over an obstacle. If more than one obstacle existed, an ostensibly productive move which avoided one obstacle may cause a collision with another. This sometimes caused the manipulator to oscillate between obstacles. The avoidance routines sometimes generated non-productive moves so it was necessary to continually check that progress was being made towards the Goal. If no headway was being made, the path finding strategy was changed. In general the algorithms failed if paths led between obstacles and in many cases there was no guarantee of finding a solution even if one or more existed.

Udupa(1977) planned trajectories for the upper arm and forearm of the Stanford Manipulator Arm separately. Firstly a trajectory was hypothesised for the upper arm directly between Start and Goal configurations. Where collisions were detected, sub-goals were introduced which were intended to
direct the path around the obstacles. For example, if a path between Pos_A and Pos_B was tested and a collision occurred then a sub-goal Pos_C between Pos_A and Pos_B was proposed. The paths between Pos_A and Pos_C and between Pos_C and Pos_B were then tested and so on, until either a clear path was found, or a calculation time limit was reached.

Having found the upper arm path, the forearm was planned for positions where the forearm could collide with obstacles.

Nguyen(1984) developed a fast heuristic algorithm for planning collision-free paths of a mobile robot in a cluttered planar work-space. The free space was described as a network of linked cones. Feasible positions and orientations of the mobile robot within the cone were computed. Feasible path segments were derived by local experts which used adjacency information of linked cones to generate local paths. Five local experts were used, namely, traversing a free convex region, sliding along an edge, circumventing a corner and going through a star-shaped region.

Khatib(1985 & 1986) used the local method described in previous sections. A manipulator moved in a field of forces. The obstacles were represented as repulsive surfaces and the goal as an attractive pole. The path planning method was to allow the summing of forces at each configuration to guide the robot to the goal.

This simple but effective method allowed obstacle avoidance to be carried out in near real-time using two PDP 11 computers and is the only other method the author has found in the literature that worked in near real time. The
The method could become trapped in a local point of minimum force if the robot was drawn between two obstacles where either no possible path existed or the robot had to pass close to the obstacles. This restricted the method to very simple environments.

Lumelsky developed an algorithm for planar 2-link manipulators which acquired the obstacle information from touch sensors located throughout their surface [Lumelsky(1987)]. The movement of the manipulator was restricted to those corresponding to linear changes in the joint variables, or those keeping parts of the manipulator in contact with the obstacles. When the manipulator hit an obstacle while travelling in the free space, it slid around the obstacle maintaining contact with the obstacle.

Tseng(1987) used an Archimedes spiral to define a path for the lower joints of a T3-776 robot. A path for the upper joints was then planned. All obstacles in the work-place were assumed to be resting on the floor of the work-cell and the upper arm passed over the top of the obstacles. The algorithms could not deal with obstacles which required the robot to pass under an obstacle.

**Global Path Planning Methods.** Global methods are usually applied after the problem has been reduced to finding a path for a point through space. This dissertation will use the term **Configuration Space** for the 3-D quantised joint space. Since this is the natural space for the robot, different paths can be easily compared. The actual path planning takes place in the subset of configuration space through which the point may pass, Gouzanes(1984) called this 'empty space' and it has also been termed 'free space' by Brooks(1983(a).
and (b)). This dissertation will use the term Free space.

There are two global approaches for finding free space.

(a) Calculate the space occupied by obstacles and subtract this from the configuration space; examples of this are Udupa(1977), Lozano-Pérez(1979) and (1981), Canny(1985) and (1986), Balding(1986), Tseng(1987) and Hwang(1988).

(b) Calculate the empty space directly; examples of this are Brooks(1983(a and b)), Chien(1984) and Gouzanes(1984).

The choice of approach depends on the type of representation used, and whether space is expected to be cluttered with obstacles or sparse. The fewer the obstacles, the more efficient is method (a) and the smaller the free space, the more efficient is method (b).

Lozano-Pérez & Wesley(1979), Sharir & Sheffi(1984), Donald(1984) and Maddila(1986) assumed a fixed orientation of a body, and a planar case. Lozano-Pérez, Wesley and Donald represented the surfaces of obstacles in configuration space. Except for very simple cases this is not easy to do. The simple cases used were 2-D and 3-D translation of solid objects. This amounts to a simple growing operation when the obstacles are transformed into a space they called C-Space. When a rotation is allowed the transformation is no longer obvious. Mobile robots are the most practical example of this problem, usually possessing two translational degrees of freedom and a rotational degree of freedom. The problem is reduced to the
motion-planning of a point amidst 'grown' obstacles.

In subsequent work [Lozano-Pérez(1981) and (1983), and Donald(1985)] the space occupied by obstacles was calculated using a slice projection technique. Projections of the obstacles onto horizontal planes were calculated for a range of Z axis values in cartesian space. These obstacles were then transformed into Configuration Space by considering the size and range of orientations of the moving object. Position and orientation of an object was represented by a six dimensional vector or a point in configuration space.

The algorithm worked for cartesian manipulators only. Obstacles were polyhedral prisms whose axes were perpendicular to the horizontal plane.

Objects were modelled as trees of convex polyhedra and space was represented by a tree of full, mixed or empty cells. A graph containing the empty cells was defined by considering the connectivity of the cells. The cells made up a graph they called the visibility graph in which each node was the verticee of a polyhedral obstacle. The path planning problem was solved by the graph searching method of Hart et al(1968).

Schwartz & Sharir (1984) extended the work of Lozano-Pérez to path planning for several disjoint discs. They considered the task of moving several circles among polygonal obstacles in the plane, a collection of several line segments joined at a common point and finally a rigid rod moving in three dimensions. The work was not extended to a robot manipulator or kinematic chain.
Brooks (1983) implemented a planar Path Planner by modelling the free space between obstacles as Generalised Cones. Brooks modelled empty space as 'freeways' along which the manipulator could move. He used a PUMA robot and separated the planning of the upper arm and forearm. The upper arm planning was done in joint space. Similar freeways were defined for the work-piece in real space. A path was found by considering the path for the upper arm and then seeing which work-piece freeways could be used with the upper arm path. Finally those paths which would cause a collision for the forearm were rejected.

This type of path planning produced paths which tended to have good clearances from obstacles. The method greatly restricted possible solutions because the constraints were applied to the movements whilst concurrently planning the upper arm and the work-piece in different spaces.

Chien et al. (1984) used the concept of a rotation mapping graph (RMG) to plan paths for a rod, and then they extended the idea to cover the Stanford Arm. Empty space was modelled as regions of collision free motion for the forearm of the Stanford manipulator. These regions were limited to those which implied collision free motion for the upper arm. These regions were then converted into a graph for searching by using a connectivity algorithm. Chien did not comment on the implementation of the algorithm or whether any practical work was completed.

Luh & Lin (1981) calculated the configuration space for the upper arm of the Stanford Arm. The path planning for the upper arm consisted of planning a path for a point among a polyhedral representation of the configuration
space. The shortest path for a point through this space was in straight lines between the edges of these obstacles. In later work, Luh presented an algorithm which, given an ordered set of edges, produced the minimum distance path. However, how to find which set of edges to use for the best path was not discussed.

O'Dunlaing et al (1984(a)) and (1984(b)) studied planar motion problems using retraction to Generalised Voroni diagrams. This work was further developed by Canny (1985). A Voroni diagram of a set of obstacles represents the locus of points that are equidistant from at least two of the obstacle surfaces, that is the locus of maximally distant points. Searches of the Voroni diagram tend to give the safest path solution in terms of distance from obstacles. An example is shown in figure 2.2.

![Voronoi Diagram](image)

**Figure 2.2:** An example of a Voroni Diagram.
Lozano-Pérez and Wesley (1979) used a similar map of free space called the visibility graph or V-Graph. A V Graph connects those vertices of obstacles that can be connected by a straight line that does not penetrate any obstacle. This is referred to as being able to 'see' another obstacle. This is shown in figure 2.3.

![Visibility Graph Example](image)

**Figure 2.3: An example of a Visibility Graph.**

Moving a ladder or a line segment among rectangular obstacles was considered by Maddila (1986). The global problem of moving a ladder was decomposed into several local motion planning problems. The free space was divided into corridors and junctions. Corridors are the 'hallways' between rectangular obstacles, and junctions are the areas where corridors meet. The movement of the ladder was either horizontal or vertical, and rotations took place at L-shaped junctions. A weighted graph called a motion graph was constructed from the solutions of the local sub-problems. The weights
represented the longest ladder that could be moved between the nodes of the motion graph so the algorithm was also capable of finding the longest length of the ladder that could be moved between two positions in the free space.

2.5 Robot Path Optimisation.

A few Path Planning algorithms have considered robot kinematics but hardly any have also considered dynamics. Research work considering system dynamics has not yet achieved on line automatic programming of robots.

Path optimisation algorithms used in the past have attempted to minimise some Cost Function of a robot path. Different criteria have been considered when estimating the cost. Five of the most important criteria are listed below.

(i) Distance travelled.
(ii) Time taken.
(iii) Energy used.
(iv) Component wear.
(v) Path safety.

The weighting given to each criteria in assessing the cost of a path must be decided before path optimisation can take place. Different applications require different emphases to be placed on the different criteria. Some criteria may oppose each other, for example, as the time taken for a robot path decreases so the energy consumed tends to increase. The "Optimum Path" is the compromise required between the various criteria.
The following sections discuss the factors affecting the five criteria and their
effect on each other.

(i) **Distance travelled.** The distance travelled by a robot may be defined,
either as the distance travelled by a point defined somewhere on the robot
(usually the gripper), or by the total amount of movement which the robot
has made.

The total amount of movement which a robot has made is the sum of the
movements of each robot axis. If there is a mixture of linear and rotary
movements then the rotary movements may be converted to linear ones by
defining their linear distances as,

\[ L_d = \theta \times L \]

where \( \theta \) = the rotary movement
\( L \) = the length of the link
\( L_d \) = the linear distance

Udupa(1977), Lozano-Pérez(1981) and Balding(1987) used the total distance
moved by a robot to calculate their 'minimum distance' paths. Gilbert &
Johnson(1985) used the distance travelled as a cost. Their solution technique
employed an interior penalty function, later used by Dubowsky et al(1985)
and (1986). When a robot is programmed to move between two positions
there are many different ways it may move, but most robots move in one or
more of three different ways: -

(a) Independent movement of axes
(b) Point to point linear interpolation
(c) Interpolation of robot axes.
It is important that any path optimised for shortest distance does not oppose the method used. The methods are described below:

(a) **Independent movement of axes.** The robot’s axes move independently from their starting positions to their finishing positions. This type of movement requires the minimum of computer control but it is difficult to model.

(b) **Point to point linear interpolation** A point is defined on the robot, and the robot moves such that the point travels in a straight line from the start point (START) to the goal point (GOAL).

(c) **Interpolation of robot axes.** The robot’s axes are interpolated such that they all have the same function of time. For example, if one of the robot’s axes has initial value $\theta_a$ and final value $\theta_b$ then,

$$\theta(t) = \theta_a + f(t)(\theta_b - \theta_a)$$

and $f(t)$ is the same for all other axes. For this type of interpolation all points on the robot arm describe complex curves in three dimensional space.

(ii) **Time taken.** The time criteria has tended to be applied to trajectories produced by path planners and not by the planners themselves.

In a study of minimum-time manipulator trajectory planning, Luh & Lin (1981) constrained Cartesian velocities and accelerations. Their
scheme required experimental identification of Cartesian velocities and acceleration bounds. Kim & Shin (1985) in a similar study, developed a method for minimum-time trajectory planning in joint space. In their study, an absolute path deviation was prescribed at each corner point, and local upper bounds on the joint accelerations derived from the arm dynamics.

Vukobratovic & Kircanski (1982), Shin & Mckay (1986) and Cesarone (1988) applied dynamic programming to the planning of trajectories where the path was specified, the control forces/torques bounded and the travel time given. Bobrow et al (1983) devised a specific technique to solve minimum time trajectory planning problems for a manipulator following a prescribed path under state dependent constraints on the torques/forces. Their algorithm cannot be extended to other performance criteria.

The time for a robot to move from one position to another depends on the following.

(a) Path Length.
(b) Path Complexity.
(c) Path Type.

(a) The path length. The greater the path length, the longer the minimum time taken for that path. The minimum possible time for a path is assumed when at least one robot joint is always changing at a maximum rate during the path. The speed of the robot's path is in turn affected by the complexity of the path.
(b) **The Path Complexity.** For a complex path, a larger amount of time is spent in accelerating and decelerating the robot arm so that average velocity is reduced.

(c) **The Path Type.** Paths which require large amounts of computing time to calculate, such as point to point linear interpolation, take a longer time to execute than paths calculated by, for instance, the interpolation of axes.

Sahar & Hollerbach (1986) recently described a general method for the planning of minimum time trajectories for robot arms. Sahar reports that 'optimal paths tend to be nearly straight lines in joint space'.

(iii) **Energy used.** Paul (1979) presented a technique which allowed the manipulator to transit smoothly from one straight-line segment to another with the motion being continuous in joint displacements, velocities and accelerations. Luh (1985) and Vukobratovic & Kircanski (1982) used the criterion of energy used by the robot motors to create a cost function for the robot path. They found that the factors affecting the energy used were the following.

(a) Distance Travelled.
(b) Time Taken.
(c) Path Shape.

(a) **The distance travelled.** Energy is dissipated in friction as the robot moves, so the further the robot moves the more energy
(b) **The time taken.** As transit time decreases for a given path, so accelerations and decelerations for the robot increase. This increases the energy used.

(c) **The Path Shape.** Smooth paths require least energy because accelerations and decelerations are reduced. Figure 2.4 shows the minimum distance path for a point moving around a rectangular obstacle. Figure 2.5 shows a path which would use less energy.

![Figure 2.4: Minimum distance path](image)
(iv) **Wear on the components.** Wear was considered by the author in Sanders *et al.* (1987(b)). As wear affects the mean time between failure and the servicing interval for a robot, reducing wear will increase productivity and reduce operating costs.

Wear on robots is affected by the same factors that affect the energy used, that is; Distance travelled, time taken and path shape.
Bonney (1985) described how the safety of a path may be viewed from three different standpoints.

(a) **The robot.** A robot may collide with obstacles if it is programmed to move too close to them. The path along which a robot is programmed to move may be different to that which it actually takes. One particular problem is the rounding off of corners. Most robots will follow straight paths which blend into other straight paths unless they are programmed to wait at via points.

To reduce the danger of a robot hitting obstacles, the nominal size of the obstacles may be increased by some safety margin. This ensures that if a robot does cut corners it will still miss obstacles.

(b) **The work-piece.** If the robot is moving quickly the forces on the work-piece will increase. This may cause the work-piece to move in the gripper or be dislodged from it.

(c) **Humans** As the speed of the robot increases so the danger to human operators is increased. This means that additional safety precautions may have to be taken.
3.1 Introduction.

To lend substance to the research work presented in this dissertation a robot system was constructed. This chapter describes the development of the equipment and the systems which ran on this apparatus.

Most modern industrial robots use sampled data control systems with hierarchical structures and several test rigs were constructed to develop a similar robot system for this work. The final apparatus is shown in figure 3.1. It consisted of a camera which provided an input to a computer vision system. This was connected to a path planning computer. A third computer was a dedicated robot controller with associated interfacing and DC servo-amplifiers, connected to a Mitsubishi Robot.

Figure 3.1: The Final Apparatus.
On the test rig the following processes were implemented:

(a) Path Planning and Path Adaption.
(b) Robot Control.
(c) Image data processing and vision data acquisition.

A simplified block diagram of the final systems is shown below in figure 3.2.

![Diagram of the final systems](image)

**Figure 3.2: A Simplified Block Diagram of the Final System**

The development of the sub-systems within the main computer and the controller are described in the following sections. The vision system is described in chapter five.
3.2 The Development of the Apparatus.

An early attempt was made to implement the system using a low cost prototype robot base as shown in figure 3.3. A CBM Series micro-computer was used as a controller. Later the micro-computer was upgraded to an Intel 8086 machine.

Figure 3.3(a): The Initial Test Rig.
(The prototype robot base)

Figure 3.3(b): The Initial Test Rig. (The Controller and robot base)
The robot base and the DC-Servo amplifier were designed and constructed as part of the project. The Servo Amplifier used to power the base is described briefly in appendix A and in detail by the author in Sanders(1988). The low level control loops are discussed briefly in section 3.7 and appendix A and in more detail in by the author in Sanders & Billingsley(1986).

When the system had been tested and proved to work the apparatus was expanded to include one Intel 8086 and one Intel 80286 based microcomputer. This was to allow the initial path planning and path adaption algorithms described in chapters six to eight to be implemented. Communication between these computers is discussed in section 3.4. Once this system had been tested with the prototype base joint, a robot was selected.

The robots made available to the author included a Unimation Puma, Syke Robotics 600-5 and Mitsubishi RM.501. The Unimation and Syke robots were complete systems, but the Mitsubishi was not. The robot only consisted of the mechanical structure, without a controller or servo-amplifiers. Early work on the project required access to the lowest levels of circuitry and machinery so the Mitsubishi RM.501 was selected. The robot motors were smaller than those used on the prototype base and they required correspondingly smaller servo amplifiers. The servo amplifier and mixer circuits were redesigned for use with +/−24 volts dc and to supply a smaller current to the motor. The Mitsubishi robot did not have integral tacho-generators, so a velocity signal was derived in the hardware from the back EMF across the motor. This transitional system is shown in figure 3.4.
On-line operation was an important aim of the work presented so the system was later changed to include two faster Intel 80286 microcomputers with co-processors. This final apparatus was later expanded to include the vision system described in chapter five.

The controller was equipped with a G64 bus system to allow expansion to control all joints of the robot. The final apparatus excluding the vision computer was as shown in figure 3.5.
1. Camera
2. RM-501 Robot
3. Obstacle
4. Path Planning Computer
5. Robot Control Computer
6. Power Supply Unit
7. G64 Rack
8. PC LabCard Connector Card

Figure 3.5: The Final Apparatus (Excluding the Vision Computer)
3.3 The Development of the Systems.

The system developed on the initial test rig had three main levels. The software for the higher levels was written in Basic and assembly language was used for the lowest level. The three levels were:

(a) The Supervisory Level.
(b) The Strategic Level.
(c) The Joint Servo Controller.

This is shown below in figure 3.6 and the different levels are described over the page.

![Diagram of the Initial Control System](image)

**Figure 3.6:** The Initial Control System for the Prototype Base.
(a) **Supervisory Level.** The supervisory level was the overall controller. This level handled interfacing with the human operator and the files containing the required movements of the robot base joint $\theta_1$ and for a simulated shoulder joint $\theta_{2sim}$.

(b) **Strategic Level.** The strategic level considered the demanded motion and assigned look up tables containing output voltage values to be used by the joint controller level. Demanded elementary movements were distributed to the joint control level. This was a more complicated task for the initial system as the shoulder joint $\theta_{2sim}$ was simulated at this level.

(c) **Joint Servo Controller.** This level was a dedicated joint controller for $\theta_1$ and realised the functional movements by controlling the joint angles using a position servo.

A Peak Detector was added to the system. This was a low level program which sampled the currents to the joint motors. In the initial system the Peak Detector fed information directly to the Supervisory level as shown in figure 3.7 over the page.

During simulated tasks, the current to the base motor from the DC-Servo Amplifier was recorded for a variety of velocities, loads and accelerations. The motor current was sampled via an A/D converter and initially was as shown in figure 3.8. This waveform was too noisy for interpretation, so smoothing methods were investigated. Suitable results were obtained using a simple low pass filter and a sample waveform is shown in figure 3.9.
 TASK

Supervisory Level

Peak Detector

Simulator

Strategic Level

Joint control Level

Position Feedback

Robot Movement

Figure 3.7: The addition of the Peak Detector to the Initial System.

Figure 3.8: Raw Current Data

Figure 3.9: Current Data passed through a Simple Low Pass Filter
The filtered wave-forms were studied and a new strategy for path adaption and force detection developed and details published in Sanders et al(1987(b)). This is discussed in detail in chapter seven.

A unique parallel hierarchical system structure evolved from this system. This is shown below in figure 3.10.

**Figure 3.10:** The Parallel Hierarchical Control System for the Transition Stage.

To speed up data processing and avoid redundancy, decision processes were performed in two computers. Slower control operations were established at the top of the structure and progressively faster operations were performed at lower levels. High level decisions and path adaption strategies were considered by the main computer while the second computer was controlling
the robot. In both computers, after obtaining information from a lower level, each level made decisions, {considering decisions from higher levels}, and forwarded commands to lower levels.

All levels and both computers were ruled by the Path Planner in the main computer. Target points could not be passed to the robot controller until a path had been planned. The robot controller had four levels which were similar to the hierarchical system described in the previous section. The levels are described below:-

- **Supervisory Level.** This level was no longer the overall controller as the Path Planner in the main computer ruled the system. The level no longer interfaced with a human operator, but instead received a trajectory locus from the Overseer level in the main computer.

- **Strategic Level.** This level worked as described previously.

- **Joint Control Level.** As before, this level executed the imposed motion of each degree of freedom. Similar software was associated with each joint of the robot except that different gains were used in the software loops for each joint.

The hierarchical structure of the main computer consisted of:-

- **Path Planner.** The Path Planner accepted a START configuration and a GOAL configuration from a human operator and produced a trajectory locus which was passed to the Supervisory level via the overseer. The
algorithms used during the transition stage were the initial 2-D local heuristic methods described later in chapter six.

- **Overseer.** The Overseer worked in parallel with the controller and oversaw the well-being of the robot. Specifically the Overseer accepted a path description as a trajectory locus from the Path Planner or new path information from the Path Adaption level and passed this information to the controller. During the work described in chapter seven the motor drive currents were monitored for collisions at this level.

- **Path Adaption.** This level considered path adaption possibilities from the information provided by the peak detector. Specifically improvements to reduce the current peaks in the motor drive current.

- **Peak Detector.** This was a low level program which sampled the currents to the joint motors and considered the relative amplitudes of successive readings. If the current rose above preset limits, an interrupt routine informed the Overseer.

Given a task in the form of a start and goal position, the system automatically planned a path with simulated obstacle constraints. In later work these obstacles were recognised using a vision system. For the initial work, the obstacles were introduced into the main computer using simulated data stored on a disk. This required the addition of a Data Processing level in the main computer. The Data Processor processed the obstacle data and passed a list of blocked joint positions to the Path Planner. This level was one of the two main software modules "TransformSphere.BAS" or
"TransformSlice.BAS" which are described in chapter five. The final structure was as shown in figure 3.11.

![Diagram of the final system structure.](image)

**Figure 3.11: The Final System (Excluding the Vision Sub-System)**

- **Software.** The main programs, communication routines and man/machine interfacing was initially written in 'C'. Time critical routines and routines accessing the BIOS at low levels, were written in assembly language. The assembly routines were linked by a common variable syntax. 'C' was selected for its portability, allowing development on off-line IBM and Apollo Domain computers away from the dedicated target system.

For the later work the high level programs were rewritten in Micro-Soft
Quick Basic. This allowed utilisation of the advanced editing facilities available in this software during development of the more complex 3-D path planning and path adaption programs. These programs are described in chapters six, seven and eight.

3.4 The Final Apparatus: Communications.

Communicating between the two computers used during the transition stage was relatively simple and was achieved using a single RS 232 serial link and later using parallel CIO plug in boards.

When the system was expanded to include the vision system, a more complex communications system was required. Each of the three computers needed to communicate with each of the other two and to interrupt at various levels. The vision system must interrupt the Path Planner and Controller if an obstacle appeared in the work place. The robot must interrupt the vision system to inform it if the robot was about to pass under the camera.

The system initially used spare ports on plug in PC LabCards but the routines to test for the arrival of information were complex and time consuming. The system was revised to use two serial RS-232 ports in each computer and this permitted the use of port interrupts with Quick Basic "On Event" instructions. The system was as shown in figure 3.12. Each computer was fitted with a 25 pin port as COM1 and a 9 pin port as COM2.
The main sets of data that were transferred were joint angles describing configurations of the robot. Blocked configurations were passed from the vision system to the Path Planner and the trajectory locus was passed from the Path Planner to the Controller. Other data handled by the communications sub-system included:

(a) A signal from the Vision System when a change in the environment was detected. This code was also passed to the Controller to warn that a new trajectory locus was being prepared. After receiving this code the Controller avoided moving the robot into the work area.

(b) An end of file code. If there were no blocked nodes then the Start of File code was immediately followed by this code.
A warning to the Vision System when the robot was about to enter the work cell. This stopped the Vision System from detecting the robot as an obstacle. When the robot was moving away from the work cell an "All Clear" code was sent to the Controller.

The "All Clear" signal depended on the number of dark pixels detected by the vision system. To test if the work-cell was empty, the number of black pixels in a frame were tested against a preset background noise level. If the frame was empty the all clear flag was set to TRUE and the Start of file, End of file codes were passed to the Path Planning computer. That is:-

IF (PicDetect% < BackgroundNoise%) AND (AllClear = false%) THEN
    PUT #1, 2, ef%; PUT #1, 2, ef% ; Send "All Clear" to the Path Planner.
    AllClear% = true%; ; Set the All Clear flag.
    ProcImage% = false%; ; A processed image is not stored.

3.5 **The Final Apparatus: The G64 Bus.**

The G64 bus is an industry standard and its inclusion allowed for expansion of the robot system in the future using the extensive range of G64 modules available. The G64 bus system consisted of the following:-

(a) A Transmitter card which connected to the robot control computer.

(b) A G64 rack system with an independent power supply.

(c) A ribbon cable and Receiver card which converted the 80286 machine signals to G64 specifications and connected to the G64 rack.
The G64 bus is a 16 bit bus with 17 address lines capable of addressing 256K. The system is shown in figure 3.13 with the servo-amplifiers mounted above. The computer interfaces to the dc servo-amplifiers were redesigned to plug into the G64 rack. Two plug in cards were used, one for the waist and shoulder joints and half of a card for the elbow joint. (A third card was built for the wrist joints but this is not considered in this dissertation).

The interface cards were all similar and each contained:-

(i) DACs to convert the digital output from the computer to analogue voltage for the servo amplifiers.

(ii) Decoder circuits for the optical encoders.

The table below shows the locations in memory of the G64 interface cards.

<table>
<thead>
<tr>
<th>G64 Memory Addresses</th>
<th>Optical Encoder Input Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>DAC Location</td>
</tr>
<tr>
<td>Base</td>
<td>&amp;AF404</td>
</tr>
<tr>
<td>Shoulder</td>
<td>&amp;AF406</td>
</tr>
<tr>
<td>Elbow</td>
<td>&amp;AF504</td>
</tr>
</tbody>
</table>

The LS2000 chips located on the G64 interface cards gave a two byte representation of each joint angle.
1. ±10V Power Line
2. DAC Input Connector
3. Optical Encoder Power Supply and Connector
4. DC Servo Cards
5. Motor Power O/P Cable
6. G64 Ribbon Cable
7. G64 Receiver Card
8. G64 Cards Containing DAC's and Optical Encoder Decoders
9. PC LabCard Input/Output Connector Card

Figure 3.13: The G-64 Bus System and Servo Amplifier rack.
3.6 The Final Apparatus: The Mitsubishi RM.501 Robot.

The Mitsubishi RM.501 is a five degree of freedom manipulator with three links as shown in the GRASP plot (figure 3.14). The main disadvantage in selecting this robot was the small working envelope and this is discussed further in chapter five.

The research considered path planning and adaption for the first three joints and the solution to the position problem for this robot is described in this section.

The robot link lengths were:

\[
\begin{align*}
\text{Upper-Arm} &= L_1 &= 220\text{mm}. \\
\text{ForeArm} &= L_2 &= 160\text{mm}. 
\end{align*}
\]

Three points on the robot were considered:

(a) The Origin.
(b) The Elbow.
(c) The Fore-Tip.

(a) The origin of cartesian coordinates was set at the centre of the shoulder joint \(\theta_2\) and the base joint \(\theta_1\). This point was referred to as "Origin".

\[
\begin{align*}
\text{Origin}_x &= 0 \\
\text{Origin}_y &= 0 \\
\text{Origin}_z &= 0 
\end{align*}
\]
Figure 3.14: GRASP Plot of the Mitsubishi RM.501 Robot.
(b) The "Elbow" was the intersection of the centre of the Upper-Arm and the elbow joint $\theta_3$.

$$\text{Elbow}_x = 220 \cdot \cos \theta_1 \cdot \cos \theta_2$$
$$\text{Elbow}_y = 220 \cdot \sin \theta_1 \cdot \cos \theta_2$$
$$\text{Elbow}_z = 220 \cdot \sin \theta_2$$

(c) The "Fore-Tip" was the centre of the end of the forearm and centre of joints four and five ($\theta_4$ and $\theta_5$).

$$\text{ForeTip}_x = \text{Elbow}_x + 160 \cdot \cos \theta_1 \cdot \cos (\theta_2 + \theta_3 \cdot \pi)$$
$$\text{ForeTip}_y = \text{Elbow}_y + 160 \cdot \sin \theta_1 \cdot \cos (\theta_2 + \theta_3 \cdot \pi)$$
$$\text{ForeTip}_z = \text{Elbow}_z + 160 \cdot \sin (\theta_2 + \theta_3 \cdot \pi)$$

These calculations were used for the work described in chapter five for the transformation of obstacles from real space to joint configuration space and for the robot experiments described later in this section. More general details of robot coordinate transforms may be found in Paul(1980) or Craig(1989).

A clockwise turn of the base joint was termed a positive change; as was a move of the shoulder or elbow upwards. The ranges of movement for the three main joints were:-

$$\theta_1 = -150 \text{ to } +150 \text{ degrees.}$$
$$\theta_2 = -30 \text{ to } +110 \text{ degrees.}$$
$$\theta_3 = +90 \text{ to } +180 \text{ degrees.}$$
A park configuration was defined for the robot with the joints at:

\[ \theta_1 = 0 \text{ degrees.} \]
\[ \theta_2 = 0 \text{ degrees.} \]
\[ \theta_3 = +180 \text{ degrees.} \]

In this position the arm was horizontal to the work surface as shown in figure 3.14. The joints were calibrated before use by driving each of the joints slowly to a set of limit switches. The limit switches were read via a PC LabCard. The calibration movement was carefully selected to withdraw the arm keeping the ForeTip at the same height. This prevented damage to the camera and robot in the event of failure in an unusual or dangerous position. Once the limit switch for the shoulder joint was detected the Elbow was raised and finally the base was driven clockwise to the end stop.

The RM-501 had several unusual design features. The two most notable were the mounting of the heavy motors within a tail some distance from the base and a spring mechanism to offset gravity loading on the shoulder joint. These features made the dynamic equations unusual. The robot arm tended to balance the weight of the motors with the arm above the park configuration. In the park configuration with no drive on the motors, the shoulder joint moved under the action of the spring mechanism to lift the robot arm. These features and their effects on the robot dynamics are discussed in more detail in chapter eight.

Many work-piece handling tasks successfully "robotised" to date are generally suited to first generation robots working in isolation from the rest.
of the production line. Many of these first generation manipulator operations can be completed by simple point to point movement with relatively uncoordinated joint control for the intermediate motions. These robot tasks will remain economic even as future generations and further developments become commercially available. Because this dissertation is concerned with Automatic Path Planning, the robot position throughout the path had to be more predictable. In most robot applications it is the repeatability which is the important property, but in this case the accuracy of the robot was more important. The robot controller is described in section 3.8 and some robot properties are discussed below.

Several initial experiments were undertaken to discover the properties of the robot required for the transformations made in chapter five and the path planning work described in chapter six.

Repeatability: Initially the robot was moved to a position so that a pin connected to the end effector made contact with a vertical wooden face. The wooden face was aligned with a line marked on the base surface as shown in figure 3.15.

The robot was programmed to move back and away from the start position to a constant position and then to return. The experiment was repeated with the robot moving to random positions and then returning. The position of the pin was recorded each time against the wooden face and the spread of positions recorded.

A plumb line was connected to the end effector in such a way that the
plumb line point just touched the base surface as shown in figure 3.16. The robot was programmed to move up and away from this position and similar experiments were conducted for horizontal repeatability.

**The Accuracy:** The plumb line was used to measure the accuracy of the position solution presented earlier in the X,Y plane. A rule was used to check the Z position.

**Results:** The repeatability experiments were repeated 25 times and the accuracy experiments were repeated for seven different random positions.

Although the repeatability quoted by the manufacturer was ±0.5 mm, the experimental repeatability was only within ±2 mm when the robot moved away to a constant position and only within ±3 mm when the robot moved away to random positions. The difference in these results could be due to the back-lash in the gear mechanisms. The accuracy was within ±3 mm.

From the results, the safety margin around the obstacles was set to 10 mm, just over three times the repeatability and accuracy. The safety margin is discussed further in chapter five.
Figure 3.15: The Horizontal Repeatability Experiments.
Figure 3.16: The Vertical Accuracy and Repeatability Experiments.
3.7 The Final Apparatus: The Joint Servo Controller.

An analogue and digital loop existed for each joint. The analogue loop was closed around the motor and the digital, sampled data loop was closed via the computer.

The analogue loop was to improve the motor time constant and response in the presence of disturbance and parameter variation. This loop used the back EMF from the motor. The digital, sampled data loop was closed via a micro computer containing the position and velocity control algorithms. The position feedback was derived from a 16 bit counter clocked by the optical encoders mounted on each joint. This count value had the following relationship with the joint angular positions:

\[
\begin{align*}
\text{Base} &= 80 \text{ counts per degree.} \\
\text{Shoulder} &= 83 \text{ counts per degree.} \\
\text{Elbow} &= 80 \text{ counts per degree.}
\end{align*}
\]

The situation of the controller is shown in figure 3.18 and the servo amplifier circuits are described in detail in SANDERS(1988).

The joint servo controller at the Joint control level used look up tables of output voltage values to the D/A converter depending on the position and velocity of each joint. The look up tables were stored in 2-D arrays so that output was a function of the error and the difference of the error with respect to the computer time constant.

\[
\text{Output} = f(e, \delta e/T).
\]
Maximum drive was output to the joint until a switching region was reached in this joint error space. Through this region the drive was changed to maximum reverse drive until a pure positional control region was reached close to the target position ($e < K$). This is as shown in figure 3.17.

**Figure 3.17: Controller Error Signal Phase Plane Description.**
Figure 3.18: A Diagram of the Device Connections
4.1. **Introduction.**

The environment of an industrial robot includes static and dynamic objects. The dynamic environment consists of the robot, objects to be manipulated and obstacles to be avoided. The free space left available to the robot depends on the accuracy of the models used for this changing environment.

As part of his dissertation Balding(1987) completed a study of modelling methods and considered the following as important requirements in representing the robot and the work place.

(a) Fast intersection calculations.
(b) Ease of use with path planning algorithms.
(c) Fast model generation.
(d) Low memory storage requirements.
(e) Efficiency (in terms of the work-place volume occupied at critical points).

In this chapter, the past work described in chapter two is examined and models are discussed with reference to the above requirements. The models considered can be divided into three categories: -

(a) The Static Environment.
(b) The Robot.
(c) Dynamic Obstacles.

These categories are discussed in the following sections.
4.2 The Static Environment.

Of the modelling methods discussed in chapter two, the following were investigated for the static environment:

(i) Polyhedral models.
(ii) Constructive solid geometry models.
(iii) Surface models.

Most published computer models of robot surroundings take the form of polyhedral obstacles with flat surfaces and straight edges as this geometry resembles the obstacles commonly found in robot work cells. These models are difficult to deal with in path-finding calculations and calculation is slow. If both the robot and obstacles are modelled by polyhedral shapes then the accuracy is high but computation time is extended. The GRASP plots shown in figures 3.15 and 3.16 of the last chapter are examples from a system using polyhedral models for all three categories. The system used was the 1990 version, but it could not perform calculations in near real time.

Constructive Solid Geometry represents conglomerations of objects as ordered binary trees. Figure 4.1 shows an example of a constructive solid geometry tree.

Terminal nodes are either primitive leaves which represent solid primitive shapes, or transformation leaves which contain the defining arguments of rigid motions. Non-terminal nodes represent operators such as rigid motions, intersection, difference or regularized union. In the example, non-terminal nodes are a union (U) and a translation. Two solid primitive shapes are shown in cross section in figure 4.1. These are combined using
three unions and two translations to form a more complex solid shape.

More information about CSG representations may be found in publications by Braid(1973), Braid(1975), and Requicha(1977).

![CSG Tree Diagram](image)

**Figure 4.1: An example of a CSG Tree.**

Surface modelling methods have been used to model complex surfaces in detail. An introduction to surface modelling is given by Ball(1983). Surface modellers use complex parametric functions such as Bezier equations to represent the detail of surfaces. These representations are difficult to use for intersection checking as only surfaces are represented. It is difficult to determine whether a point in space is inside an obstacle or not and consequently, to decide whether surfaces intersect is also difficult.
4.3 The Robot

The requirements for the robot model were similar to those for the dynamic obstacles described in section 4.4. For automatic path planning in real time, the most important factor was the speed of intersection calculation. A constraint was that the robot model selected needed to contain the entire volume of the robot.

A large number of robots have a similar design to that of the Mitsubishi RM 501 robot in that these robots have two major links, (the upper arm and the forearm) and three major joints (Base, Shoulder and Elbow). The simplest possible representation was two lines jointed at one end. Constant distances from the lines were then defined as enclosing the outer casing of the robot. This gave two connected cylinders with hemispherical ends. The advantages of this representation were that the cylinders modelled the robot links efficiently and the intersection calculations between the robot arm and obstacles were simple. The calculations simply consisted of:

(i) In the case of a sphere, finding the distance from the centre of a sphere to the closest point on the line. From this distance was subtracted the radius of the arm and the sphere, to give the distance between the arm surface and the sphere surface.

(ii) In the case of similar 2-D slices, the obstacle model was expanded by the radius of the arm and the calculation reduced to comparing the position of the centre line with the obstacle. This was similar to the 'growing' techniques of Udupa(1977).

The end effector was modelled as a sphere with a radius sufficient to enclose the gripper motors. The work-piece was assumed to be small and enclosed by this sphere. For future work, work-pieces could be modelled easily as additional spheres.
As the path planning algorithm was not designed for angular variation of the end effector, the orientation of the end effector was assumed to be fixed. The end effector position relative to the forearm was defined such that the gripper axis and the forearm were continuous.

4.4 **Dynamic Obstacles.**

Spheres are the simplest three dimensional shapes and Hopcroft et al (1983) described how to calculate intersections among spheres efficiently. These calculations were easily modified to deal with the intersection between lines and cylinders required for the robot model. The method of modelling dynamic obstacles by spheres was initially selected for use in this research.

Any shape may be modelled by spheres to any accuracy. The greater the accuracy required however, the larger is the number of spheres needed. Experimental work demonstrated that for larger numbers of spheres the computation time increased so that the accuracy of a model was limited by the computation time permitted for the path finding algorithm. This is described further in section 4.10

In general, it is difficult to decide on the best sizes and positions of spheres to model real obstacles. In practice the number of spheres used to model obstacles were 1, 2 and 4. This made the models simple and speeded up path calculation, requiring little computer storage, while still producing efficient robot paths. When multiple spheres were used for the global path planning method described in chapter six, there were complications in checking which joint configurations had been checked already for other
spheres. This meant there was an increase in processing time and when multiple spheres were used the sphere model was not the most efficient.

This initial work with sphere models was used to compare intersection calculation speeds for several other models of the dynamic obstacles. Two other models compared favourably:

(a) Similar 2-D slices in joint space.
(b) Six Sided Parallelepiped.

In all cases it was assumed that the 2-D cross section of the obstacle in the X-Y plane and the height (Z) of the obstacle was available. This data was entirely viewpoint dependant and could only provide knowledge concerning visible faces and explicit depth information. This was similar to the polyhedral models used by Brooks(1983(b)) in that the obstacles were effectively only two and a half dimensional. That is, they had a two dimensional shape and a height. The 3-D obstacle shapes considered during the work described in this dissertation were:

(i) A Cylinder.
(ii) A Cube.
(iii) A Simple six sided polyhedra.

Although the algorithms for sphere calculation were potentially simple, the parallelepiped or similar 2-D planar slices tended to model these 3-D shapes as accurately and in the case of the 2-D slices, more quickly than single or multiple spheres in discretised 3-D space. The method using 2-D slices is described. The models were calculated by considering two pairs of boundaries:
(a) The angles of the base joint, $\theta_1$, which bounded the obstacle ($\theta_1\text{min}$ and $\theta_1\text{max}$).

(b) The maximum distance $D_{\text{max}}$ and minimum distance $D_{\text{min}}$ from the origin (Maximum and minimum radii).

The obstacle was modelled as a series of similar 2-D planar slices. The reference slice was calculated within a boundary of a line from the Origin bounded by $D_{\text{max}}$ and $D_{\text{min}}$ and the limits of the Z axis. The BLOCKED configurations for the shoulder and elbow joints $\theta_2$ and $\theta_3$ were then calculated for this bounded plane and copied for all $\theta_1$ within the two bounding angles, $\theta_1\text{min}$ and $\theta_1\text{max}$.

For the global path planning method described in chapter six, this reduced the number of time consuming searches and tests for BLOCKED points that were required. The major part of the algorithm described in section 4.7 was reduced to copying values within a 3-D graph of configuration space. The obstacle was first modelled as a 2-D rectangle as this was the simplest model which could be derived from the row and column limits of the object under the camera. These limits were derived during the low level image processing described in chapter five.

The transformation of the 2-D slice models and sphere models into joint configuration space is described and compared in sections 4.6 and 4.7.
Once obstacles had been detected and modelled by the methods described in the earlier sections, the data was processed to transform the obstacles into joint configuration space. This was initially achieved for simulated obstacles in the Data Processing level of the main computer and later for real obstacles detected by the vision computer. The programs used are described in sections 4.6 and 4.7 and in appendix B. In all cases the robot upper arm and forearm were modelled as their minimum bounding cylinders, with hemispherical ends and the end effector was enclosed by a sphere.

For the global path planning methods it was necessary to transform the obstacles into joint configuration space. A point obstacle in cartesian space is not transformed into a point in joint space. If the point is within the robot work-space then it is transformed into one or more complex three dimensional shapes.

Complex shapes may be represented within a computer as geometric shapes, units of space or by approximating the shapes by mathematical curves. The global path planning method represented the obstacles as regions of joint space consisting of small units. The method was not restricted to any particular design of robot and may be used with any number of degrees of freedom. The program presented was based on the implementation for the three major axes of a Mitsubishi RM.501 robot.

For the global path planning method a graph was created which consisted of a three dimensional structure of unit regions. The 3-D graph had each
dimension corresponding to a principal degree of freedom of the robot arm, \( \theta_1, \theta_2 \) and \( \theta_3 \). The wrist configurations, \( \theta_4 \) and \( \theta_5 \) were not considered in the graph. Each unit was initially set to 'CLEAR' status and the positions (in joint space) at which the robot intersected obstacles were then calculated. Each unit represented a range of configurations for the robot, in terms of, \( (\theta_1_{\text{cent}}, \theta_2_{\text{cent}}, \theta_3_{\text{cent}}) \), plus a degree of movement away from these central joint values;

\[
\pm \delta \theta_1 \pm \delta \theta_2 \pm \delta \theta_3
\]

All units together represented the whole robot work-space and the number of units in the graph, Node_{Total}, was given by:

\[
\text{Node}_{\text{Total}} = \frac{(\theta_{1\text{max}} - \theta_{1\text{min}}) \times (\theta_{2\text{max}} - \theta_{2\text{min}}) \times (\theta_{3\text{max}} - \theta_{3\text{min}})}{2 \times \delta \theta_1 \times 2 \times \delta \theta_2 \times 2 \times \delta \theta_3}
\]

where

\[
\theta_{1\text{max}}, \theta_{1\text{min}} = \text{the upper and lower limits of } \theta_1.
\]
\[
\theta_{2\text{max}}, \theta_{2\text{min}} = \text{the upper and lower limits of } \theta_2.
\]
\[
\theta_{3\text{max}}, \theta_{3\text{min}} = \text{the upper and lower limits of } \theta_3.
\]

This will later be expressed as:

\[
\prod_{j=1}^{3} \frac{\theta_{j\text{max}} - \theta_{j\text{min}}}{2 \times \delta \theta_j}
\]

If at any configuration in a unit, the robot intersected an obstacle, then the unit was set to BLOCKED. If at all configurations in a unit the robot did not intersect an obstacle then the unit remained CLEAR. The path planning problem for the global approach described in chapter six was then reduced to finding a series of neighbouring units between the START and GOAL configurations that were still CLEAR.

The first method considered for transforming obstacles into joint configuration space was to check each unit of the graph for intersections with each obstacle. This method was slow, taking up to three minutes to
calculate for a point obstacle. The program running time was proportional to the number of obstacles and the number of nodes. If the free space was assumed to be larger than the blocked space, a faster method was to consider each obstacle and test for the nodes which could contain the transformed obstacle. This was the method adopted and the algorithm was as follows:

For a node in the graph where the robot could intersect the obstacle, recursively test all the neighbouring units to see if they are also within the reach of the robot.

The programs are described in the following sections:

4.6 The Transformation into Joint Space: Spheres.
The graph data structure described in section 4.5 was initialised. The limits of the graph corresponded to the angular limits for the robot's joints within the range of the work cell and obstacles outside this work-space were ignored. As the graph carried out intersection checks at a limited number of positions, only a limited number of trigonometric solutions were required and these were calculated at the start.

Before the obstacles were calculated all the units in the graph had a flag set to 'CLEAR' status. Four other flags were used with each node, these were:-

- 'New obstacle'
- 'Upper arm tested'
- 'Forearm tested'
- 'On list'

Each unit code was stored as one byte of computer memory in the array NodeStatus% and the flags used one bit each.

The obstacle data was received from a file or from the vision system and the first task for the program was to read this data.

The task was then split into two sub-tasks, firstly to calculate the upper arm and then to calculate the forearm blocked space on the graph. A
configuration was calculated at which the part of the arm under consideration was closest to the obstacle centre. If the forearm was being considered, then the configuration where the Foretip was at the centre of the sphere was calculated. For the upper arm, the configuration was calculated for which the centre line of the upper arm pointed at the sphere centre. If the obstacle was within the reach of the link being tested, then this configuration was the first unit for the transformed obstacle.

The base angle was calculated from the X,Y coordinates of the sphere as shown in figure 4.2. Firstly the modulus (L3) and the angle (Sphθ) from the robot to the centre of the sphere was calculated and a test was conducted to see if the sphere was out of range, in which case no further processing was necessary.

\[
\text{Waist Angle } \theta_1 = \text{InvTan} \left( \frac{Y}{X} \right)
\]
\[
\text{Modulus } XY = \sqrt{X^2 + Y^2}
\]
\[
\text{Sphθ} = \text{InvTan} \left( \frac{Z}{\text{Modulus } XY} \right)
\]
\[
L3 = \sqrt{X^2 + Y^2 + Z^2}
\]

The cosine rule was used to calculate the shoulder \( \theta_2 \) and elbow \( \theta_3 \) angles, as shown below.

\[
L1 = \text{Upper-Arm} = 220\text{mm}
\]
\[
L2 = \text{ForeArm} = 160\text{mm}
\]

\[
\theta_3 = \text{InvCos} \left[ \frac{L1^2 + L2^2 - L3^2}{2 \cdot L1 \cdot L2} \right]
\]
\[
\theta_2 = \text{InvCos} \left[ \frac{L1^2 + L3^2 - L2^2}{2 \cdot L1 \cdot L3} \right] + \text{Sphθ}
\]

87
If the sphere centre was too close to the robot then $\theta_3$ would exceed its lower limit ($\theta_3 < 90^\circ$). In this case $\theta_3$ was set to $90^\circ$ and $\theta_2$ was calculated using InvTan as the arm formed a right angled triangle as shown in the
following code:

\[
\text{If } \theta_1 < 90^\circ \text{ THEN} \\
\quad \theta_2 = 90^\circ \\
\quad \theta_3 = \text{InvTan} \left( \frac{L2}{L1} \right) + \text{Sphe} \\
\text{END}
\]

This gave a starting configuration close to the centre. When the lower limit of \( \theta_2 \) was exceeded, \( \theta_2 < -30^\circ \), the angle was set to minus 30° and the distance between the upper-arm and sphere centre was calculated (the modulus) using the subroutine FindModulus, from which the cosine could be used to find the new \( \theta_3 \). The pseudo code for this routine was:

\[
\text{If } -30^\circ < \theta_2 \text{ THEN} \\
\quad \theta_2 = -30^\circ \\
\quad \text{Calculate Modulus} \\
\quad \theta_3 = \text{InvCos} \left( \frac{L1^2 + L2^2 + \text{Modulus}^2}{2 \cdot L1 \cdot \text{Modulus}} \right) \\
\text{END}
\]

The first configuration was set to BLOCKED. Its neighbouring units were also tested and if they were set to BLOCKED then their neighbours were checked. The position problem was solved using the forward kinematic calculations developed in section 3.7 and the minimum distance between the obstacle and the robot arm was calculated, (provided that it had not completed the calculation before). The method continued recursively until the whole obstacle transformation was found.

All units were set to BLOCKED, which had any two opposite neighbouring units which were also BLOCKED. Any units which were on the edge of the now solid obstacle were recorded on a list. All the neighbours of the units on the list were tested, and the process repeated until the surface of the transformed sphere was completely defined.

Nodes which were BLOCKED were stored on a list of units to be expanded
later. When a unit was expanded it was retrieved from the list and new
BLOCKED points were added to the list. When all the nodes on the list
were exhausted the obstacle transformation was complete.

The operation of the lists, the testing routines, the expand routines and fill
in routines are described in appendix B.

The most important consideration was processing speed. Times for
calculating obstacles were recorded during the project and these are
presented in the results section of this chapter (Section 4.9).

4.7 The Transformation into Joint Space: 2-D Slices.

The program utilised the data structures described above. These were
initialised to form a 3-D graph of joint space and the required trigonometric
solutions were calculated at the start. All the units in the graph were set to
‘CLEAR’ status and similar flags were associated with each node. Obstacle
data was simulated or received from the vision system and the first task for
the program was to read this data. The two and a half dimensional model
was then created.

Firstly the limits in x were increased by the radius of the upper-arm:-

\[
\text{Start Row Clearance} = \text{Start Row} - \text{Upper Rad}\%
\]
\[
\text{End Row Clearance} = \text{End Row} + \text{Upper Rad}\%
\]

The modulus of the ends and centre point on the edge Start Col were
calculated with their angles as shown in figure 4.4. This is shown below for
the furthest end from the Origin.

\[
\text{Corner( Top Left, Angle\%)} = \text{InvTan} \left( \frac{\text{Start Col}}{\text{End Row Clearance}} \right)
\]
\[
\text{Corner( Top Left, Modulus\%)} = \sqrt{\text{End Row}^2 + \text{Start Col}^2}
\]
The following parameters of the model were found:

- The inside radius from the origin. \((D_{\text{min}})\)
- The outside radius from the origin. \((D_{\text{max}})\)
- The smallest base angle, \((\theta_{1\text{min}})\).
- The largest base angle, \((\theta_{1\text{max}})\).

If the obstacle was matched to a template then the height of the obstacle was extracted from the template, otherwise if the obstacle height was unknown, the height \((Z)\) was set to infinity. The segment was extrapolated to the Y axes so that calculation took place in the Y,Z plane. The modelled obstacle was expanded by the radius of the robot's upper-arm in
the Y and Z plane. \( \theta_1 \) was set to its new lower limit and the inverse kinematic solution was found for all the points within the obstacle, as shown in the following code:-

```plaintext
FOR Yaxis = (Radius%(min%) - UpperRad%) TO (Radius%(max%) + UpperRad%)
    FOR Zaxis = -255 TO (Radius%(Z%) + UpperRad%)
        CALL InvKinematics
    NEXT Zaxis
NEXT Yaxis
```

The coordinates in Y and Z were converted to robotic joint angles using the inverse kinematic solution in the subroutine InvKinematics. Firstly the distance from the origin to the cartesian point \((L3)\) and the angle to the point \((\text{Curv}\theta)\) were calculated.

\[
\text{Curv}\theta = \text{InvTan} \frac{Zaxis}{Yaxis}
\]

\[
\text{sqL3} = Yaxis^2 + Zaxis^2
\]

\[
L3 = \sqrt{\text{sqL3}}
\]

The upper-arm was checked against \(L3\) to see if a collision was possible. If within the reach of the upper-arm then \(\theta_2\) was set to \(\text{Curv}\theta\) and \(\theta_3\) was set to BLOCKED between its limits if \(\theta_2\) was within its limit.

If \(L3\) was less than the Forearm plus upper-arm then the Forearm collided with the point. \(\theta_2\) and \(\theta_3\) were calculated using the cosine rule and if \(\theta_2\) and \(\theta_3\) were within their limits the NodeStatus was set to BLOCKED. The subroutine SetupNodeStatus repeated the NodeStatus settings for \(\theta_1\) from \(\theta_1\),min to \(\theta_1\),max. This completed the model transformation.
4.8 The Transformation into Joint Space: Other Models.

Although the fastest global transformations were achieved using the solid sphere and 2-D slices, several other models were investigated in order to compare performance. These were:

(a) Semi Solid Spheres.
(b) Hollow Spheres.
(c) Simple Polyhedral Shapes.

(a) Semi Solid Spheres. The method was similar to that used for the solid sphere except that the expansion routine expanded two nodes at a time from the centre of the sphere. When a CLEAR node was reached, the node was placed onto a spare list (list 3) along with the node it had expanded from. Once all the expansions had revealed CLEAR nodes the subroutine ExpandIn tested the node pairs on list 3 to see if a collision occurred between them. When a collision occurred this was taken to be the edge of the sphere and the NodeStatus was set to collision, otherwise the node was assumed to be on the edge of the sphere. This is shown below.

***** Considering the Node Pairs on List3 *****

\[
diff1\% = (t1\% - list3\%(NoList3\%, RefX\%)))
\]
Repeated for t2\%, t3\%

\[
NoList3\% = NoList3\% - 1
inc\% = 1
\]

IF diff1\% <> 0 THEN ; Initially move the joint +5°.

IF diff1\% > 0 THEN
inc\% = -1

DO

\[
t1\% = t1\% + inc\%
\]

CALL Testpos ; Test the new position.
LOOP UNTIL (nodecode%(t1\%,t2\%,t3\%) AND 2) = 2
END IF

IF diff2\% <> 0 THEN ; Repeat the process for \(\theta_2\) and \(\theta_3\).

This method became complex when the centre node of the nodes being tested was CLEAR. As the inner node code was set to tested, that point was not retested and expanded. This meant the edge was not clearly
defined as the nodes around the inner node were not tested. This led to an attempt to use hollow spheres as described in the next section.

(b) **Hollow Spheres.** This program effectively followed the surface of the sphere, setting the surface nodes to BLOCKED so that the path planning program would be unable to enter the sphere. Instead of beginning the process at the sphere centre, the Z coordinate was set to the top of the sphere:

\[ \text{sphereEdge}(Z) = \text{SphereCentre}(Z) - (\text{Forrad} + \text{Radius}) \]

The program then expanded the nodes as described above, placing BLOCKED nodes onto the list. If more than six collisions occurred in a single expansion then the test node was inside the sphere and the collisions were removed from the list. Where less than six collisions had occurred the robot was following the edge. The CLEAR nodes recorded were assumed to represent the edge of the sphere when passing the data to the path planning computer. The changes made to the expandout routine are shown below:

```
t1% = t1% - 1 ; Move -5°.
IF t1% >. lowlim(1) THEN
  CALL Testpos ; Test the node.
  IF (nodecode%((t1, t2, t3)) AND 2) = Collision THEN
    colis% = colis% + 1
  ELSE
    limit% = limit% + 1
  END
ELSE
  limit% = limit% + 1
END

IF t1% <. highlim(1) THEN
  CALL Testpos
  IF (nodecode%(e1, e2, e3)) AND 2 THEN
    colis% = colis% + 1 ; Increment Collision store.
ELSE
  limit% = limit% + 1
```

This was repeated for \(\theta_2\) and \(\theta_3\). The collision store was checked and if equal to six, then the Foretip was inside the sphere and the collisions recorded from that expansion were removed.
This method was slow as every time the program entered the sphere, six nodes were tested and removed from the lists. The sphere models were complex in joint space and following the surface was a complex task, (especially when a single sphere could be transformed into two separate shapes in joint space).

(c) **Simple Polyhedral Shapes.** As discussed in chapter two, polyhedra are commonly used to model obstacles. The method modelled the obstacles as six sided parallelepipeds. The program established the position of the edges of the model in X and Y from simulated data or by calculating the limits of the rows and columns set in the vision program. The height of the object was retrieved from the associated template as described in chapter five.

In the subroutine TestPos the edge positions were expanded with the model radius of the part of the robot under test (ie upper-arm or forearm), as demonstrated below for an expansion of the forearm in X.

\[
\begin{align*}
\text{Expand}_X\text{Low}\% &= \text{EdgePosition}\%(\text{LowX}\%) - \text{ForRad} \\
\text{Expand}_X\text{High}\% &= \text{EdgePosition}\%(\text{HighX}\%) + \text{ForRad}
\end{align*}
\]

The cartesian coordinates of the arm were tested against the expanded polyhedral edge limits, as shown below.

\[
\begin{align*}
\text{IF} \ \text{Expand}_X\text{High} < [\text{foretip}(X\%)] > \text{Expand}_X\text{Low}\% \ \text{THEN} \\
\text{IF} \ \text{Expand}_Y\text{High} < [\text{foretip}(Y\%)] > \text{Expand}_Y\text{Low}\% \ \text{THEN} \\
\text{IF} \ \text{Expand}_Z\text{High} < [\text{foretip}(Z\%)] > \text{Expand}_Z\text{Low}\% \ \text{THEN} \\
\text{CALL} \ \text{PutonList}(t1\%, t2\%, 13\%) \ ; \ \text{Node added to list} \\
\text{nodecode}\%(11\%, 12\%, 13\%) = (\text{nodecode}\%(11\%, 12\%, 13\%) \ OR \ 2)
\end{align*}
\]

END

END
4.9 Results.

The most important consideration for the system was that it should be suitable for real time applications. Times for transforming obstacles were recorded during the project and as an example, the times for the different models to transform the vertical cylinder into joint space are shown below.

The times were recorded with the Z axis of the cylinder at $X = 0$ mm and $Y = 310$ mm with respect to the origin. The obstacle was simulated.

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (Seconds)</th>
<th>Number of BLOCKED nodes recorded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Sphere</td>
<td>9.8</td>
<td>2455</td>
</tr>
<tr>
<td>Two Spheres</td>
<td>15.6</td>
<td>2366</td>
</tr>
<tr>
<td>Hollow Sphere</td>
<td>25.8</td>
<td>2476</td>
</tr>
<tr>
<td>Simple Polyhedron</td>
<td>29.2</td>
<td>1995</td>
</tr>
<tr>
<td>2-D Slices</td>
<td>5.9</td>
<td>2504</td>
</tr>
</tbody>
</table>

**Figure 4.5: Table of Transformation times for an Upright Cylinder.**

The Sphere Model: An obstacle was modelled first as a single sphere of the smallest radius which would enclose the obstacle. Later, if time allowed it was modelled by two smaller spheres and then four spheres. Nodes set to BLOCKED associated with the first sphere tested usually also collided with other spheres. The forward kinematic solutions did not need to be recalculated for these nodes but the total calculation time increased with the number of spheres because the overhead of calculation for each sphere was greater than the saving in time achieved as the spheres became smaller. This meant the single sphere calculation was faster than the calculations for multiple spheres although the single sphere model was less accurate and
had a larger volume.

The problem when using more than one sphere was that the centre of several spheres would be set to BLOCKED (with some surrounding nodes) after the expansion of the first sphere. As these nodes were BLOCKED, later spheres were sometimes not retested so that many nodes were not added to the list.

To overcome this problem the centre was tested for an old collision. If TRUE then the node was placed on a new list, (list 3) where the node was expanded later. The nodes on the new list were then dealt with until the list was empty, meaning that for models using two spheres all the nodes outside the first sphere had been found. The routine is shown below:

```plaintext
IF nodestatus%(t1%, t2%, t3%) = 1 THEN ; Test for Old Collision.
   CALL PutonList3(t1%, t2%, t3%)
   DO
   FOR No% = 1 TO numonList3%
      CALL GetoffList3(t1%, t2%, t3%)
      CALL ExpandTest
   NEXT No%
   LOOP UNTIL numonList3% = 0
END
```

The subroutine ExpandTest tested each node after expansion to see if a collision had occurred with the first sphere. If it had then the node was added to list 3 to be expanded at a later date and a bit was set in the flag NodeStatus so that the node was not retested. If the node was CLEAR then the edge of the first sphere had been found and the node was tested for collision against the new sphere using a subroutine Testpos. This is shown over the page:-
t1% = t1% - 1
IF t1% > = lowlim%(1) THEN
    ***** Test for collision with 1st sphere if not tested before  *****
    IF (nodestatus%(t1%, t2%, t3%) AND 65) = 1
        nodestatus%(t1%, t2%, t3%) = 64 ; Set node to Tested
        CALL PutonList3(t1%, t2%, t3%)
    END
END

***** If no collision with other spheres test for new sphere *****
IF (nodestatus%(t1%, t2%, t3%) OR 1) = 0 THEN CALL TestPos
END

The 2-D Slice Model: The advantage of modelling the obstacle as a series of similar 2-D slices was that once the collision coordinates of $\theta_2$ and $\theta_3$ had been calculated for a particular $\theta_1$ then these collisions could be repeated for the limits of $\theta_1$ which collided with the obstacle. This reduced the main processing task to copying data rather than calculating forward or reverse kinematic solutions.

The representation of obstacles using similar 2-D slices was the fastest to transform into discrete 3-D joint configuration space. The graphical representation of the blocked angles for different obstacles with their different positions are shown in the following pages.

The results are for obstacles modelled as similar 2-D slices so that only the base angle limits are shown. As the BLOCKED nodes were copied between these bounding angles, the BLOCKED nodes are the same for each of the angles. The bounding angles are shown above each chart.
Shown below is the format for pages 101 to 106 which show a representation in joint configuration space for the different orientations of the six obstacles mentioned in section 4.4. The obstacles were separated for their configurations into the three with the largest $X,Y$ area and the three with a smaller $X,Y$ area.

![Diagram of obstacle configurations](image)

**Figure 4.6: The Position of the Obstacles shown in figures 4.7 to 4.39**

---

**PAGE FORMAT for the HORIZONTAL CYLINDER.** p.101

**HORIZONTAL POLYHEDRA.** p.102

**HORIZONTAL POLYHEDRA.** p.103

<table>
<thead>
<tr>
<th>1.1</th>
<th>1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>1.4</td>
</tr>
<tr>
<td>1.5</td>
<td>1.6</td>
</tr>
</tbody>
</table>

---

**PAGE FORMAT for the VERTICAL POLYHEDRA.** p.104

**VERTICAL CYLINDER.** p.105

**CUBE.** p.106

<table>
<thead>
<tr>
<th>2.1</th>
<th>2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Figure 4.7

Figure 4.8

Figure 4.9

Figure 4.10

Figure 4.11

Figure 4.12
Horizontal Polyhedra (Maximum Area)

### 50° to 135°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>50°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>55°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>60°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>65°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>70°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>75°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>80°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.13

### 60° to 120°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>60°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>65°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>70°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>75°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>80°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.14

### 70° to 110°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>70°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>75°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>80°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.15

### 40° to 80°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>40°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>45°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>50°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>55°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>60°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>65°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>70°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>75°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>80°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.16

### 60° to 100°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>60°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>65°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>70°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>75°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>80°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.17

### 85° to 120°

<table>
<thead>
<tr>
<th>Angle</th>
<th>Upper Joint</th>
<th>Waist Joint</th>
<th>Forehead Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>85°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>90°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>95°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>100°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>105°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>110°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>115°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>120°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>125°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>130°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
<tr>
<td>135°</td>
<td>10°</td>
<td>50°</td>
<td>10°</td>
</tr>
</tbody>
</table>

### Figure 4.18
Vertical Polyhedra.

<table>
<thead>
<tr>
<th>60° to 80°</th>
<th>Foramen Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>90°</td>
<td>115°</td>
</tr>
<tr>
<td>106°</td>
<td>96°</td>
</tr>
</tbody>
</table>

Upper Joint = 56°

75° to 95°

<table>
<thead>
<tr>
<th>40° to 85°</th>
<th>Foramen Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>90°</td>
<td>115°</td>
</tr>
<tr>
<td>106°</td>
<td>96°</td>
</tr>
</tbody>
</table>

Upper Joint = 56°

75° to 120°

<table>
<thead>
<tr>
<th>50° to 100°</th>
<th>Foramen Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>90°</td>
<td>115°</td>
</tr>
<tr>
<td>106°</td>
<td>96°</td>
</tr>
</tbody>
</table>

Upper Joint = 56°

Figure 4.25

65° to 95°

<table>
<thead>
<tr>
<th>50° to 100°</th>
<th>Foramen Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>90°</td>
<td>115°</td>
</tr>
<tr>
<td>106°</td>
<td>96°</td>
</tr>
</tbody>
</table>

Upper Joint = 56°

Figure 4.26

Figure 4.27

Figure 4.28

Figure 4.29
### Vertical Cylinder

#### 60° to 75°

<table>
<thead>
<tr>
<th>Joint</th>
<th>90°</th>
<th>115°</th>
<th>125°</th>
<th>135°</th>
<th>100°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
<tr>
<td>Joint</td>
<td>90°</td>
<td>115°</td>
<td>125°</td>
<td>135°</td>
<td>100°</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
</tbody>
</table>

**Figure 4.30**

### 65° to 115°

<table>
<thead>
<tr>
<th>Joint</th>
<th>90°</th>
<th>115°</th>
<th>125°</th>
<th>135°</th>
<th>100°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
<tr>
<td>Joint</td>
<td>90°</td>
<td>115°</td>
<td>125°</td>
<td>135°</td>
<td>100°</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
</tbody>
</table>

**Figure 4.31**

### 45° to 95°

<table>
<thead>
<tr>
<th>Joint</th>
<th>90°</th>
<th>115°</th>
<th>125°</th>
<th>135°</th>
<th>100°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
<tr>
<td>Joint</td>
<td>90°</td>
<td>115°</td>
<td>125°</td>
<td>135°</td>
<td>100°</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
</tbody>
</table>

**Figure 4.32**

### 75° to 130°

<table>
<thead>
<tr>
<th>Joint</th>
<th>90°</th>
<th>115°</th>
<th>125°</th>
<th>135°</th>
<th>100°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
<tr>
<td>Joint</td>
<td>90°</td>
<td>115°</td>
<td>125°</td>
<td>135°</td>
<td>100°</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>45°</td>
<td>55°</td>
<td>85°</td>
<td>115°</td>
<td>105°</td>
</tr>
</tbody>
</table>

**Figure 4.33**

**Figure 4.34**
<table>
<thead>
<tr>
<th>55° to 80°</th>
<th>80° to 105°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Joint 1</strong></td>
<td><strong>Joint 1</strong></td>
</tr>
<tr>
<td>90°</td>
<td>115°</td>
</tr>
<tr>
<td>100°</td>
<td>105°</td>
</tr>
<tr>
<td>25°</td>
<td>30°</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 4.35**

<table>
<thead>
<tr>
<th>60° to 115°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Joint 1</strong></td>
</tr>
<tr>
<td>90°</td>
</tr>
<tr>
<td>100°</td>
</tr>
<tr>
<td>25°</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 4.36**

<table>
<thead>
<tr>
<th>40° to 115°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Joint 1</strong></td>
</tr>
<tr>
<td>90°</td>
</tr>
<tr>
<td>100°</td>
</tr>
<tr>
<td>25°</td>
</tr>
<tr>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 4.37**

<table>
<thead>
<tr>
<th>70° to 140°</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Joint 1</strong></td>
</tr>
<tr>
<td>90°</td>
</tr>
<tr>
<td>100°</td>
</tr>
<tr>
<td>25°</td>
</tr>
<tr>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 4.38**

**Cube.**
UNKNOWN OBSTACLE

60° to 120°

Figure 4.40 Representation in 3-D Joint Space for an Unrecognised Obstacle.
The number of BLOCKED nodes for the different obstacles placed in the various positions was as shown below.

<table>
<thead>
<tr>
<th>Reference Position</th>
<th>Horizontal Cylinder</th>
<th>Horizontal Polyhedra (Minimum Area)</th>
<th>Horizontal Polyhedra (Maximum Area)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>2540</td>
<td>1425</td>
<td>2261</td>
</tr>
<tr>
<td>1.2</td>
<td>1890</td>
<td>975</td>
<td>1316</td>
</tr>
<tr>
<td>1.3</td>
<td>948</td>
<td>634</td>
<td>786</td>
</tr>
<tr>
<td>1.4</td>
<td>1236</td>
<td>616</td>
<td>1072</td>
</tr>
<tr>
<td>1.5</td>
<td>1224</td>
<td>608</td>
<td>1064</td>
</tr>
<tr>
<td>1.6</td>
<td>1206</td>
<td>628</td>
<td>1058</td>
</tr>
</tbody>
</table>

Figure 4.41: Table of Blocked Configurations.

<table>
<thead>
<tr>
<th>Reference Position</th>
<th>Vertical Polyhedra</th>
<th>Vertical Cylinder</th>
<th>Cube</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>978</td>
<td>854</td>
<td>1106</td>
</tr>
<tr>
<td>2.2</td>
<td>986</td>
<td>882</td>
<td>1050</td>
</tr>
<tr>
<td>2.3</td>
<td>1238</td>
<td>1364</td>
<td>1456</td>
</tr>
<tr>
<td>2.4</td>
<td>1914</td>
<td>1890</td>
<td>2050</td>
</tr>
<tr>
<td>2.5</td>
<td>1762</td>
<td>1762</td>
<td>2044</td>
</tr>
</tbody>
</table>

Figure 4.42: Table of Blocked Configurations.
4.10 Discussion and Conclusions

Once an obstacle increased above a certain size or was moved closer to the origin, part of the obstacle intersected both the Upper Arm and ForeArm joint space. Thus the work-space occupied by the obstacle suddenly increased and the calculation time increased. This can be seen in figures 4.25 - 4.27, 4.30 - 4.32 and 4.35 - 4.37.

For the transformation methods a graph of calculation time vs. discrete work-space volume can be expected to be linear, that is the calculation time for an obstacle was approximately proportional to the number of units tested, the total number of nodes being the work-space volume:

$$\prod_{j=1}^{3} \frac{\theta_{j,\text{max}} - \theta_{j,\text{min}}}{2 \times \delta \theta_{j}}$$

The computer time required for obstacle transformations was short. The initial conversion time to model the static environment was slow; up to three minutes of computer time depending on the complexity of the model, but the transformation was only performed when the system was powered up.

The accuracy of the models affected the performance of the Path Planner. High accuracy models required more computation time and therefore longer solution times. Low accuracy models required links or obstacles to be oversized to eliminate the chance of undetected collisions. Lowering the accuracy led to the rejection of valid solutions.

The world may be modelled accurately by Polyhedral, CSG or surface...
modelling methods but they are complex models requiring complex intersection calculations. The generation of these models was slow and they would have provided difficult problems for the heuristic algorithms described in chapter six. For global solutions the transformation of the static environment need only be made once, so that computation time is not a problem. The infrequent initialisation hypothesis of Udupa (1977) also suggested time need not be considered in transforming the static environment. An accurate model was therefore selected and Polyhedra were used to model the static environment. For later work using a simple static environment the model was reduced to a single polyhedral shape modelling the work surface.

For dynamic models, speed of calculation was important. The simplest possible intersection calculations for the local methods were made using the sphere model. Calculation was reduced to finding the distance from the robot to a point and subtracting the radius of the sphere to give the distance to the surface of the sphere. The initial work used this model.

Modelling with more than one sphere was considered. As the real environment for a robot becomes more complex so more spheres are needed for the model. It was considered how increasing the number of spheres might increase the accuracy of the model. The case of modelling a unit cube was investigated by Balding (1987). A cubic number of spheres was used, i.e. 1, 8, 27, 64 etc. The spheres formed a regular pattern and were equal in size. An infinite number of spheres was required to model the cube completely but modelling objects using the same sized spheres was inefficient. For example, in modelling a cube using sixty-four spheres of the
same size, eight of the spheres are totally enclosed and might easily be replaced by a single larger sphere without increasing the model volume. This is shown in cross section figures 4.43 and 4.44.

**Figure 4.43:** Cross Section of a cube modelled by 64 spheres.

**Figure 4.44:** Cross Section of a Cube modelled using different sized spheres.
To compare the modelling of obstacles using single and multiple spheres, as an example, a model of the cylinder using one and two spheres is compared. The volume of two spheres of radii 35 mm were compared to that of one sphere of 70 mm as shown below.

\[
\text{Volume of Two Spheres} \\
2 \times \frac{4}{3} \pi \times 35^3 = 359,188 \text{ mm}^3
\]

\[
\text{Volume of One Sphere} \\
\frac{4}{3} \pi \times 70^3 = 1,436,755 \text{ mm}^3
\]

The area of the two spheres would be much smaller except that the model of the robot must then be considered to find the union volume,

Robot ∪ Model

Upper-arm Model radius = 80 mm

Union Radius for a single sphere 70 + 80 = 150

Union Radius for two spheres 35 + 80 = 115

Union Volume of a single sphere \( \frac{4}{3} \pi \times 150^3 = 11,494,040 \text{ mm}^3 \)

Union Volume of Two Spheres \( 2 \times \frac{4}{3} \pi \times 115^3 = 12,741,211 \text{ mm}^3 \)

There was a similar number of collisions for both models. When points within the second sphere were not tested to see if they had collided during the calculations for a previous sphere, this partially explained the lack of improvement in processing time for the model using two spheres.

When the points within a sphere were tested to see if they collided with a
previous sphere a saving in processing time could have been expected but in the routine ExpandTest the nodes were continuously expanded until they reached the outer surface of a sphere where tests were conducted to see if the outer surface node collided with another sphere, before the next sphere had been filled. This wasted processing time.

Considering the simple six sided parallelepiped model, the volume of the model for the horizontal cylinder was less than that of both a single sphere or multiple spheres.

Parallelepiped Volume = \( (60 + 160)^2 \times (140 + 80) = 10,648,000 \text{ mm}^3 \)

(where 160 was included due to the upper-arm. This was added to allow for the expansion required in X and Y)

This potentially reduced the number of blocked nodes (and therefore the processing time), but the shape and therefore the calculations were more complex so that calculation time increased. This can be seen in figure 4.5.

Using the two dimensional slice model of the cylinder, \( \theta_2 \) and \( \theta_3 \) were only determined for a single slice. This reduced the processing time as this slice of BLOCKED nodes was copied for all \( \theta_1 \) within the bounding base joint angles. As shown in figure 4.5 on page 96, the number of BLOCKED nodes produced was similar to other models, so that the intersection volume was approximately the same as for the sphere and polyhedral models. This suggested an equivalent accuracy.

Considering the graphical representation of joint space for the various
obstacles shown on pages 100 to 105, for the obstacles with a larger surface area (Horizontal cylinder and six sided polyhedron), as the obstacle was placed further away from the robot, the range of waist angle collisions reduced, for example for the cylinder:-

Figure 4.7 range = 90°.
Figure 4.9 range = 50°.

The distance in Y had increased, reducing the total number of BLOCKED nodes shown in figure 4.41 on page 107 from 2540 to 948 and therefore reducing the processing time. This was illustrated when the obstacles were moved from position 1.1 to 1.3. The upper-arm collided with the obstacles at position 1.1 while at 1.3 the upper-arm was out of range. As the obstacles were moved from 1.4 to 1.6 the number of blocked nodes remained constant as did the processing time.

Considering the obstacles with a smaller top surface area, when the obstacles were near to the furthest edge of the work-cell (2.1 and 2.2) the waist joint range was small compared to positions 2.4 and 2.5. As part of the obstacle was out of range of the robot, less nodes were BLOCKED. As positions 2.1 and 2.2 were close, the robot upper-arm collided with the obstacle producing an increase in the number of BLOCKED nodes in figure 4.42 on page 107.

For an unknown obstacle the height was set to the reach of the robot. There were a large number of blocked nodes and the processing time increased but the system could deal with unknown obstacles. An example is
As can be seen from the graphical representation of joint space there were some gaps in the models. This occurred because the inverse kinematic solutions mapped cartesian coordinates to joint configuration space. As there is no simple continuous relationship between the two spaces a discrete increment of 10mm in X, Y and Z was used. This meant some nodes were missed in the middle of obstacles. Decreasing the increment value would solve this problem or using a dynamic increment value so that the value was large close to the centre of the obstacle and smaller once a surface was reached.

![Diagram showing mapping from Cartesian space to joint space](image)

**Figure 4.45:** Mapping from Cartesian Space to Joint Space for the model using similar 2-D slices.

The heuristic algorithms described in chapter six were made simpler by the use of spheres as the distance and direction of the robot to the nearest obstacle was easily calculated. Thus directions could be quickly modified heuristically to avoid collisions.
The detail of dynamic obstacles modeled by spheres was variable, for instance a cubic obstacle away from likely paths could be modeled as a single sphere. Although obstacles were initially modeled as a single sphere as this was fast, for critical items, and if time allowed, the obstacle could be modeled with greater numbers of spheres. The initial work-space volume would be reduced by this method, but the critical work-space for path finding may not be significantly affected.

Of the models considered in this dissertation, the method of modeling obstacles by similar 2-D slices (developed as part of the research work) had the fastest intersection calculation times. This model was adopted for the later work and will be used in future work. Using the sphere models and 2-D slices described, software models of the dynamic work-place were quickly passed to the main computer by the vision system. The vision system is described in the next chapter and the path planner is described in chapter six.

The calculation time for complex obstacles modeled as spheres was short as each sphere required only four data items, (three cartesian coordinates and the sphere radius). Similar 2-D slices were only slightly more complex, requiring the two bounding angles of the base joint $\theta_1$, the inner and outer radius and a height (five items of data).
Chapter Five

IMAGE DATA PROCESSING AND THE VISION SYSTEM

5.1 Introduction.

This chapter discusses the methods of retrieving information concerning the dynamic obstacles that were discussed in chapter two and chapter four. Chapter three described the evolution of the systems excluding the vision subsystem and this chapter describes the expansion of these systems to include a vision system.

The function of understanding a scene involves a complex sequence of computations. Processing must take place to raise the quality of the raw data to levels necessary to perform this analysis. The camera returned a voltage proportional to the light level of a range of points in a scene. This was visually represented by 256 shades of grey, Black through to White as displayed on a monochrome television set. These levels may not be a true representation as they are a product of many factors:-

- The reflectance function.
- The illumination.
- The orientation of the surface.
- Mutual illumination and shadowing.
- Electrical noise.
- Visual noise.

As discussed in chapter four, the sub-system was initialised with a description of the static environment modelled as polyhedra. Specifically the robot base, camera stand and the bench were modelled.

During the initial work described in this dissertation, simulated dynamic obstacles were loaded from a file. In the later work real obstacles were detected by the vision system. Several camera configurations and methods
were considered in order to generate a 3-D model:

(a) Using a Stereo Image from two cameras.
(b) Using a view with a single camera set at an angle to the work­
place.
(c) Using an overhead camera with pattern recognition to detect the
obstacle from a set of known obstacles.
(d) Using an overhead camera to establish the area in the X,Y plane
and setting the height to infinity for all obstacles.

These are discussed in section 5.3. The method selected was a single overhead
camera using pattern recognition techniques to identify the obstacles. The low
level image processing techniques are described in section 5.4 and the higher
level pattern recognition techniques are described in section 5.5.

5.2 Overview of the Final Apparatus (Including the Vision System).

The system hardware described in chapter three was expanded to include the
Vision Computer and was as shown in figure 3.1 on page 48.

At the same time the Data Processing module responsible for the
transformation of obstacles into joint configuration space was moved from the
main computer to the vision system computer. This allowed the three main
sub-systems described in chapter three to operate in parallel. That is: -

- The detection and modelling of the Obstacles.
- The Path Planner and Path Adapter.
- The Robot Controller.
The vision system hardware consisted of a dedicated 80286 micro-computer with a co processor and a plug in interface card for each Hitachi standard 625 line silicon vidicon camera. The initial system used a Digihurst MicroEye interface card to complete the A/D conversion from the analogue camera data. In its delivered state this was slow and took 20 seconds to capture an image of 256 x 160 pixels. By modifying the hardware this time was reduced to between 4 and 5 seconds.

Each interlaced frame from the camera took 20 ms. With a 625 line screen and an aspect ratio of 4:3, each line took 64µs and each picture element 0.14µs. This required the A/D to convert within 0.14µs (7.25 MHz); the A/D supplied with the converter board (ZN427) operated with a minimum conversion time of 10µs (0.1 MHz). This was below the rate necessary to operate a real time digital display and the system was restricted to retrieving one line scan of data during each frame. As the research presented in this
dissertation concentrated on Automatic Path Planning and not data acquisition, this was considered adequate. In order to demonstrate the system working in real time an Electric Studio Frame grabber was acquired towards the end of the work in 1990.

In both cases the interfacing with the camera A/D boards was written in 80286 Assembly Language.

- **Lighting:** The initial work used lighting from one or more angle poise lamps positioned over and around the work cell. A diffusing element composed of finely woven fibre was placed over the bulbs to evenly distribute the illumination but the method suffered from varying light conditions and noise. Attempts were made to eliminate this problem, mainly by flooding the object with illumination bright enough to submerge the shadows and noise, but the camera still required repeated calibration.

A calibration technique was used employing an initial scan which was free of obstacles. Throughout this scan, a tally was maintained on the highest and lowest gray levels in the scene. These ideally should have been 255 and 0, but due to lighting conditions they were usually between 110 and 0, less than 50% of the available range. The maximum and minimum values having been found, the contrast was stretched over the full dynamic range of 0 to 255. That is,

\[
I_{NEW} = \frac{(I_{OLD} - I_{min}) \times 255}{(I_{max} - I_{min})}
\]

where \( I = \) Pixel intensity.
The reference frame having been stored, the scene was continuously re-scanned and each element was subtracted from the reference frame to remove the effects of uneven illumination caused by reflections external to the scene. In practice noise points subtracted to give negative results and small differences gave misleading results. For the later work the lighting system was changed to a back lighting system and this reduced these problems.

5.3 Obstacle Detection: The Configuration of the Apparatus.

The four methods introduced in section 5.1 are discussed in this section.

(a) A Stereo Image from two cameras.
(b) A view with a single camera set at an angle to the work-place.
(c) An overhead camera with pattern recognition software to detect the obstacle from a set of known obstacles.
(d) An overhead camera but setting the height of obstacles to infinity.

(a) Stereo Imaging: Two cameras were positioned above the work place as shown in figure 5.2.

Two images were captured and stored in separate arrays. After edge detection as shown in figure 5.3, template matching methods were used to attempt to recognise the common features in both images. As the work was limited to black obstacles against a white background, this problem was more simple than the general template matching problem. This was similar to considering obstacles introduced into a known background where different images were compared.

The methods were similar to those presented by Lo(1990).
Figure 5.2: The Configuration for the Stereo Vision System.

Figure 5.3: The Stereo Image after Edge Detection.
Once obstacles had been recognised in both cameras the position and height of the object was calculated. The distance from the cameras to the obstacle was determined by comparing corner points of the obstacle in the two images. The obstacle height was calculated by subtracting this distance from the height of the cameras.

The two cameras were a known distance apart. The corner points of the obstacle appeared on both images. If these images were overlapped, the points did not coincide. The position in the image and the distance between the points could be used to determine the range of the obstacle and therefore the height of the obstacle. As obstacles were moved closer to the cameras (in practice by mounting the same obstacle on white bases of various height) the disparity became less. The system was calibrated to determine the height of the obstacles. The range could be approximated by:

\[ \text{Range} = r = d \{ \sqrt{(f^2 + x_L^2 + x_R^2)} \div (x_L - x_R) \} \]

The height of the object could then be found: \[ \text{Height} = h = h_c - r \]

where,

- \( r \) = Range from the left camera lens if the point was in the right side of the scene. or Range from the right camera lens if the point was in the left side of the scene.
- \( d \) = Distance between the camera lens centres.
- \( f \) = Focal length of the cameras.
- \( x_L \) = Distance of the image pixel from the left centre position.
- \( x_R \) = Distance of the image pixel from the right centre position.
- \( h \) = The height of the object.
The values of $x_L$ and $x_R$ could be positive or negative depending on their location relative to the centre of their respective images.

(b) **Single Camera viewing from an Oblique Angle:** This is shown in figure 5.4. A strong source of illumination was placed directly behind the camera and a diffused source placed at $90^\circ$ to the vision board. This configuration enabled faces of the object to be illuminated with different levels of incident light producing distinct gray regions separating the faces. Due to problems with camera movement no attempt was made to calibrate in this configuration. The calibration would have involved a non linear relationship as the size of the object varied with distance from the camera and some form of complex pattern recognition would have been required.
The configuration was considered and a program was written to produce a 3-D model of obstacles by fitting lines \((y = mx + c)\) to data produced by a trace routine. Each fitted line was stored in an array for later comparison as a gradient \((m)\) and a Y crossing point \((c)\). Code was also inserted to allow lines of infinite or zero gradient to be fitted. The method is discussed further in appendix C and is based on work presented by Oaten (1990). Due to the inaccuracies inherent in fitting data points and the fitting percentage of the line, several lines of similar gradients and crossing points were generated. A double sort routine was used to select similarly proportioned lines which were averaged to produce single lines corresponding to the straight edges of the object. Figure 5.5 illustrates the results of the line fitting procedure on a 3-D object (triangular prism).

![Figure 5.5: Line Fitting for a 3-D Triangular Object.](image)

From this mathematical representation, it was possible to generate a polyhedral model of an object. That is, the absolute positions of the corner
points were labelled and stored as a three dimensional wire frame representation including vertex labelling to indicate each edges' contribution to the complete object description. Three lines crossing at a point represented a visible corner, two crossing lines at a point represented corners where a third (or greater number) edge was hidden from the camera. Visible corners vertically above one another were assumed to be the height. Although 3-D spatial information was gleaned from a 2-D picture, edges hidden from view were unknown quantities. The programs developed demonstrated how much computation time and code length was necessary to analyze a relatively low resolution 3-D image.

(c) Single Camera above the work place: (Using Pattern Recognition): Since the camera was fixed, the X and Y coordinates were calibrated to refer the physical position to the array position. 1.55 mm per vertical pixel and 1.85 mm per horizontal pixel accurately positioned the object under the camera. The configuration was as shown in figure 3.1 on page 48.

(d) Single Camera above the work place: (Infinite Height): As described above, the camera being fixed enabled calibration of the X and Y coordinates to refer the physical position to the array position. In this case the height of the obstacles was not determined and the models of the obstacles were given an infinite height. This method was to be used if the template matching routines did not work in real time.
Discussion: Each of the methods above were considered and they are discussed individually:

(a) **Using a Stereo Image from two cameras.** Once the stereo system had been set in place the accuracy was dependent on the relative positions of the matched image points. (The Correspondence problem). The method attempted to match points using edge detection techniques and searching for corner points. This was achieved by creating a window around corners in one image and searching for similar areas in the other image. The method was unreliable and for the prototype system, the processing took up to 90 seconds while still not guaranteeing a result. The method would have required much more programming time and faster or parallel processors to produce a usable system. This is discussed further in chapter nine.

(b) **Using a view with a single camera set at an angle to the work-place.**

This method was very susceptible to changes in illumination of the scene. This required careful adjustment. Lighting aberrations and shadowing caused malfunctions in the software routines and often this resulted in the shadows being mistaken for regions of interest and being analyzed as objects.

When strong light was provided from directly behind the camera, the shadowing effects were minimised or moved to the rear of the object being viewed and thus partially hidden from the camera. This reduced but did not remove the possibility of shadow regions being analyzed as objects.

The system did not function correctly and the processing was far more complex than for any other configuration of the apparatus.
(c) **Using an overhead camera with pattern recognition techniques.**
This configuration of the apparatus gave the best results and allowed the simplest processing techniques to be used. This method was selected for the work described in this dissertation. The pattern recognition techniques are discussed in section 5.5.

(d) **Using an overhead camera, setting the model height to infinity.**
This method reduced the available free space for the robot as shown in figure 4.40 but was useful when the pattern recognition techniques failed to identify an object from the templates.

**Conclusions:** The configuration described in (c) and (d) was used. The use of templates in real time required the vertically mounted camera without the distortion problems associated with the obliquely mounted camera. In the latter case the size of the object varied with its distance from the camera. In both (a) and (b) the processing time was excessive.

Use of technique (d), setting the obstacle to infinite height, allowed the robot system to continue operating in the presence of unexpected obstacles.
5.4 Obstacle Detection: Low Level Vision Techniques.

Processing at a low level reduced the effects of noise and shadowing prior to reducing the size of the array and applying the techniques of edge detection and pattern recognition. The work described in this dissertation used standard techniques to enhance the raw image in order to enable accurate sizing and recognition of the object under the camera.

The methods described are based on spatial domain techniques, that is methods that operate directly on the pixels in an image. These can be expressed as:

\[
g(x,y) = T[f(x,y)]
\]

where
- \( f(x,y) \) is the original image.
- \( g(x,y) \) is the processed image.
- \( T \) is an operator over some neighbourhood of \( (x,y) \).

The following methods were considered and are discussed in this section:-

(a) Gray Level Weighting.

(b) Smoothing. (i) Neighbourhood Averaging. (ii) Weighted Neighbourhood Averaging. (iii) Median Filtering.

(c) Thresholding.

(d) Reduction of the Array Size.

(a) Gray Level Weighting: The data obtained from the camera provided explicit information regarding the gray level content of the scene but for object recognition it was more useful to enhance information regarding the object and to reduce the gray levels referring to the background. The implementation of the process involved an overall loss of information, although the loss of irrelevant data was offset by an increase in the relevant foreground gray levels.
By weighting the data, closely matched gray levels could be separated and this helped to organise the data in preparation for the thresholding procedure. The method worked but was time consuming. The process operated by calculating a histogram of the gray levels in a sampled image. Specifically it calculated the frequency of occurrence of each of the separate gray levels. A sample histogram of the work place with an obstacle present is shown in figure 5.6.

![Histogram of raw image data](image)

**Figure 5.6: A Sample Histogram of the Raw Image Data.**

The histogram shows a high incidence of 'white' (background) and a low incidence of dark gray (object). By flattening this histogram to a level at which the incidence of white was lowered and the incidence of gray was raised, an enhanced image was produced. The histogram flattening was performed by dividing the sum of the histogram values up to each gray level into a large number of increments for relatively frequent values and smaller values for rarer gray levels.

The program to perform gray weighting totalled the quantity of gray levels up to the maximum (usually ≈110) and divided this into sixteen equal cumulative frequency distributions. These individually aggregated frequencies having been set, the histogram was consulted and a new addition of frequencies begun.
When the quantities of frequencies equalled the individual frequency total, the gray level reached was marked and the process repeated from the marked gray level. The levels were more widely spaced in the regions of rarer frequencies. A histogram of this new adjusted image was taken, the result of which is shown below in figure 5.7.

![Histogram](image)

**Figure 5.7: A Histogram of the Adjusted Image Data.**

(b) **Smoothing:** Smoothing was a technique applied to the raw pixel data to remove noise. Smoothing algorithms tend to blur the image, especially at edges where there are abrupt changes in intensity. This can be related to the frequency plane where edges imply high frequency components which are smoothed using the low pass filter that many algorithms emulate. All smoothing algorithms in the spatial domain compare the value of a pixel with its neighbouring pixels and, using some form of interpolation, replaced the pixel in question with a smoothed value. This was particularly effective when applied to individual 'spot' noise. Three smoothing algorithms were considered:

(i) Neighbourhood Averaging.

(ii) Weighted neighbourhood averaging.
(iii) Median Filtering.

(i) **Neighbourhood Averaging:** Given an $N \times N$ image $f(x,y)$, the procedure generated a smoothed image $g(x,y)$ whose grey level at each point in the image $(x,y)$ was obtained by averaging the grey level values of the pixels contained in a predefined neighbourhood of $(x,y)$, say $(m,n)$. That is:

$$g(x,y) = \frac{1}{(m \times n)} \sum f(m,n)$$

That is for a 3x3 neighbourhood, the centre pixel in the window shown below was replaced by the average of the pixels in the 3x3 window.

![Figure 5.8: Neighbourhood Averaging.](image)

The centre pixel's new value became:

$$[\text{value}(x-1,y+1) + \text{value}(x,y+1) + \text{value}(x+1,y-1) + \text{value}(x-1,y) + \text{value}(x,y)$$

$$+ \text{value}(x+1,y) + \text{value}(x-1,y+1) + \text{value}(x,y+1) + \text{value}(x+1,y+1)] / 9$$

(ii) **Weighted neighbourhood averaging:** The technique employed was to weight only the centre pixel of the window and then to average as above. In
this work the best results were achieved with a weighting of eight.

\[
\begin{array}{ccc}
(x-1,y-1) & (x,y-1) & (x+1,y-1) \\
(x-1,y) & 8^* (x,y) & (x+1,y) \\
(x-1,y+1) & (x,y+1) & (x+1,y+1) \\
\end{array}
\]

**Figure 5.9: Weighted Neighbourhood Averaging.**

The centre pixel's new value was:

\[
\frac{\text{value}(x-1,y+1) + \text{value}(x,y+1) + \text{value}(x+1,y-1) + \text{value}(x-1,y) + \text{value}(8^* (x,y) + \text{value}(x+1,y) + \text{value}(x-1,y+1) + \text{value}(x,y+1) + \text{value}(x+1,y+1))}{16}
\]

(iii) **Median Filtering:** The same 3x3 window of pixels was used but the median of the values was selected as the new value for the centre pixel. As an example, the centre pixel and its 8 neighbours are shown in figure 5.10 below. 0 refers to black, whilst 15 refers to white.

\[
\begin{array}{ccc}
15 & 2 & 2 \\
3 & 9 & 10 \\
10 & 10 & 12 \\
\end{array}
\]

**Figure 5.10: Median Filtering.**

The values were then numerically ordered and the centre (median) value
selected. The median value from figure 5.10 is underlined:

\[ 2 \ 3 \ 3 \ 9 \ 10 \ 10 \ 10 \ 12 \ 15 \]

**Selection of the Smoothing method:** Using the averaging method many noisy 'spot' pixels were wrongly converted into small regions and the edges of obstacles became excessively blurred. This caused problems when accurate sizing of the object was required for object recognition.

Median filtering was computationally slower but produced better results in terms of less blurring and an almost total elimination of 'spot' noise. Any 'spot' noise was moved to the 'high' end of the list, and was not selected. The routine which performed median smoothing used a 'bubble sort' procedure to select the median value. Although a 'quick sort' procedure is usually faster for large unsorted arrays, for only nine elements the 'bubble sort' performed the operation faster.

(c) **Thresholding:** Later problems of analyzing abrupt changes in gray intensities would have been compounded by the smoothing operation which tended to soften sharp edges into ramp functions. Thresholding was used to segment an image into regions of similar gray levels. A threshold level was set and pixels were compared with this level. Pixels above the threshold were set to 1 and pixels below the threshold were set to 0. That is:-

\[
\begin{align*}
\text{If } f(x,y) \leq T & \text{ then } f(x,y) = 0 \\
\text{If } f(x,y) > T & \text{ then } f(x,y) = 1 \\
\text{where } T & = \text{Threshold Level.}
\end{align*}
\]
(d) **Reduction of the Array Size**: Processing time was a function of the size of the image array. The image array could be reduced by only considering the area containing an obstacle. The technique was to quickly scan the image data until an obstacle was detected. The routine then focused on the object to obtain the maximum amount of information concerning the image.

The method was to check the number of pixels below the threshold level after each column was scanned. When the number of pixels was greater than two and an obstacle had not been detected already, then the column scan steps were reduced to give a higher resolution and a flag "Object Detected" was set to true. The row of the image array was set and the "All Clear" flag used to signal the controller was set to false. This was achieved using the following code

```
IF NoPixels >= 2 AND ObjectDetected% = false% THEN
   ColumnSteps% = 2
   OffsetRow% = Row%
   ObjectDetected% = true%
   ObjectRow% = ObjectRow% + 1
   AllClear% = false%
   CALL ColumnScan ; load image data into array
END
```

While the image was being scanned the pixels were tested against the threshold level and once the scan left the obstacle the column scan steps were increased. The "ObjectDetected" flag was set to false so that data was no longer loaded into the new smaller array.

```
IF NoPixels% <= 1 AND ObjectDetected% = true% THEN
   ColumnSteps% = 4
   ObjectDetected% = false%
END
```

The array size was now the object row size x the number of columns. The limits of the columns which contained the object were then found in a similar manner.
5.5 **Obstacle Detection: High Level Vision Techniques.**

Several parameters were considered for the matching templates used for recognition, but before these were applied, edge detection techniques were used to locate the boundaries of separate regions. The process detected abrupt discontinuities within the image and used a local derivative operator to transform these discontinuities into marked edges. The gradient operator used to detect edges was defined as the two dimensional vector $G\nabla$ such that;

$$G\nabla = [G_x, G_y]^T$$

The gradient was defined as $|G_x| + |G_y|$ where

$$G_x = \frac{\delta f}{\delta x}$$
$$G_y = \frac{\delta f}{\delta y}$$

In digital form this was the difference in intensity of horizontal and vertical neighbours of the pixel under scrutiny using the first order difference:

$$G_x = f_x(i,j) = f(i+1,j) - f(i,j)$$
$$G_y = f_y(i,j) = f(i,j+1) - f(i,j)$$

Once the edges had been detected templates were produced to match the images against relevant details of obstacles held in computer memory. Several types of template matching algorithms are described in the literature [Groover et al.(1986), Fu et al.(1987), Fairhurst(1988), Galbiati(1990)]. Three were considered during the work presented in this dissertation:-
(a) Fixed size parameters - area, perimeter etc

(b) Rubber band parameters - internal angles, length of sides ratio etc

(c) Mathematical transformations (eg the HOUGH transform)

Fixed templates were used as a set of known obstacles were assumed and the parameters selected for the matching procedure were:-

(i) **Area**  
*Obtained by counting the pixels with a common property.*

(ii) **Perimeter**  
*Obtained by counting the connected edge pixels.*

(iii) **Diameter**  
*The maximum distance between edge points around an object.*

(iv) **Compactness**  
\[ \text{Compactness} = \frac{(\text{Perimeter})^2}{\text{Area}} \]

(v) **Thinness**  
\[ \text{Thinness} = \frac{\text{Diameter}}{\text{Area}} \]

These are described:-

(i) **Area**: During the thresholding process an Area count was inserted and incremented whenever a pixel was identified as part of an obstacle. That is:-

\[
\text{IF } f(X, Y) < T \text{ THEN} \\
\text{Area} = \text{Area} + 1 \\
\text{END}
\]

(ii) **Perimeter**: A similar process took place during Edge Detection. Whenever a pixel was identified as being part of an edge the perimeter count was incremented. That is:-

\[
\text{IF } (\text{Gradient X} + \text{Gradient Y}) > \text{GradientLevel} \text{ THEN} \\
\text{Perimeter} = \text{Perimeter} + 1 \\
\text{END}
\]
(iii) **Diameter**: During edge detection the positions of the edge pixels were stored in an array. On completion of the edge detection process each of these positions were compared to the other positions stored in the array. The largest distance between any two edge points was defined as the diameter. To calculate the distance between edge points the row value of the pixel under test was subtracted from the row value of the reference pixel and named "endx". A similar process was completed for the column values to produce "endy". The distance between the edge points was calculated and the largest distance was recorded as the diameter. This was the maximum distance between any two pixels in the obstacle shape. The algorithms used were:-

(If RefX and RefY identify the recorded edge pixel and X and Y identify the edge pixel under test), then

\[
\begin{align*}
\text{endx} &= (X - \text{RefX}) \\
\text{endy} &= (Y - \text{RefY}) \\
\text{TempDia} &= \sqrt{\text{endx}^2 + \text{endy}^2} \\
\text{IF TempDia} > \text{diameter THEN} \\
\text{diameter} &= \text{TempDia} \\
\text{END}
\end{align*}
\]

(iv) & (v) **Compactness and Thinness**: These were ratios derived from the Area, Perimeter and Diameter such that:-

\[
\begin{align*}
\text{Compactness} &= \frac{\text{Perimeter}^2}{\text{Area}} \\
\text{Thinness} &= \frac{\text{Diameter}}{\text{Area}}
\end{align*}
\]

Once the parameters had been found, two alternatives were considered for pattern recognition:-

(a) Probability.
(b) Average Error.
(a) **Probability:** To calculate the probability of an object being one of a set of known objects the parameters were taken as a percentage of error with respect to the template parameters. The error for each parameter for each template was calculated and summed and the error for each template parameter was divided by the summed error of that parameter for every template. This gave a probability value of error for that template for each parameter. When all the probability errors were added together a value of one was the result. The probability of an error occurring for each template was found by taking the mean probability of all the parameters for the particular template. The equations used are shown below:

\[
\text{Parameter Error} = \frac{(\text{Object Parameter} - \text{Template Parameter})}{\text{Template Parameter}}
\]

\[
\text{Probability Error} = \frac{\text{Parameter Error}}{\sum \text{Parameter Errors}}
\]

\[
\text{Average Prob of Template} = \frac{\sum \text{Probability Errors}}{\text{Number of Parameters}}
\]

For interpretation, it was assumed that if the computer could not recognise an object then the probability of an error for any template would be \(1/\text{Number of Templates}\). This value proved to be too high for interpretation so it was reduced to half the value, that is: -

\[
\text{Interpretation Value} = \frac{1}{(N \times 2)}
\]

Although the error for all the templates could be large for an unrecognisable object, one error value could be less than the others and the probability of an error occurring would be less for that particular template. The probability error could be low enough to be recognised as a known obstacle.

(b) **Average Errors:** The error for each parameter was found by comparing
the object and template parameters. The error was taken as a percentage with respect to the template parameter. The mean percentage of the parameter errors was then found to give the error for the template and from this the lowest error was selected to find the template which matched the object.

Parameter Error = \frac{(Object Parameter - Template Parameter)}{Template Parameter}

Average Error = \frac{\sum Parameter Errors}{Number of Parameters}

If the lowest percentage error of a template was less than twenty percent then it was assumed that none of the templates matched and the object was unknown. This method had a high success rate in interpretation and recognition once the templates had been established and this method was selected for the work described in this dissertation.

Position of the Obstacle: Once the obstacle had been recognised the centre of the obstacle in the x, y plane was found. The slope of the diameter was calculated and the X and Y coordinates were found for the point half way along the diameter, giving the centre position. The X and Y positions were added to (or subtracted from) the first edge position at the end of the diameter to give an approximate position of the centre of the obstacle. That is, if $endx$ and $endy$ were the difference in rows and columns between two edge points and:

\[
\begin{align*}
X_{pos} &= X \text{ Centre Coordinate.} \\
Y_{pos} &= Y \text{ Centre Coordinate.} \\
RefX \text{ and } RefY &= \text{The reference Edge Point for the diameter.} \\
X, Y &= \text{The opposite end of the diameter.}
\end{align*}
\]

then
\[ \Theta = \text{InvTan}(\text{endy} / \text{endx}) \]

\[ X_{\text{centre}} = \left( \frac{\text{diameter}}{2} \right) \times \cos \Theta \]

\[ Y_{\text{centre}} = \left( \frac{\text{diameter}}{2} \right) \times \sin \Theta \]

IF \(\text{RefX} > X\) THEN
\[ X_{\text{pos}} = X + X_{\text{centre}} \]
ELSE
\[ X_{\text{pos}} = X - X_{\text{centre}} \]
END

IF \(\text{RefY} > Y\) THEN
\[ Y_{\text{pos}} = Y + Y_{\text{centre}} \]
ELSE
\[ Y_{\text{pos}} = Y - Y_{\text{centre}} \]
END

Once the obstacle had been identified and positioned, the stored model was extracted. The variable \(\text{ShapeNo}\), was initialised to the number of the matched template and an array with the parameters of known obstacles was consulted to provide the \(Z\) coordinate/coordinates. The \(X\) and \(Y\) coordinates were known from the centre of the obstacle. If the obstacle was not recognised then the parameter 'Diameter' was converted into a radius and a 'dummy' obstacle inserted with a large \(Z\) axis value. This prevented the robot moving over an obstacle of unknown height, instead the robot could fold the elbow joint and move inside the object.

**Detection of Movement**: The row and column number in which the object was first detected (\(\text{RowOffset}\%\) and \(\text{StartCol}\%)\) was noted once the image had been captured. Limits were then set around the values to allow for changes due to lighting, shadows or noise. During subsequent scans \(\text{RowStart}\) and \(\text{ColStart}\) were tested against the limits set by the last scan. If they were outside the limits then it was assumed that the obstacle had been moved and reproprocessing took place to find the new position. This is shown below.

***** Testing the Obstacle Position against the Limits *****

IF \(\text{StartCol}\% > \text{Mov(ColPos}\%) \text{ OR StartCol}\% < \text{Mov(ColNeg}\%\) \text{ THEN}
\[ \text{Moved}\% = \text{true}\% \]
IF \(\text{RowOffset}\% > \text{Mov(RowPos}\%) \text{ OR RowOffset}\% < \text{Mov(RowNeg}\%\) \text{ THEN}
\[ \text{Moved}\% = \text{true}\% \]

***** Loading in new Limits before object reprocessed *****

\[ \text{Mov(Cpos}\%) = \text{Limits}(\%\text{Cend}\%) + 1 \]
\[ \text{Mov(Cneg}\%) = \text{Limits}(\%\text{Cend}\%) - 1 \]
The Detection of Multiple Obstacles: If two obstacles were detected, in order to store their image data separately, a variable RowImageNo was incremented each time an object was detected. The row and column limits of each image were stored in a 5 x 2 array called Limits%.

Obstacles could also appear above and below one another. An obstacle above and to the left of a lower object would cause the end column limit of the upper image to increase as the program took the second image to be part of the first. The End Column limit was stored in a temporary array and tested with the last end column scan limit. If the difference was greater than five it was assumed the first image had ended. The limits of the first image were then transferred into the second image location while the limits of the second image were transferred to the first image location. The information from the second image was inserted into first image location as shown in the following code:-

```plaintext
IF Temp%(ColEnd%, ColImages%) < column% THEN
    Temp%(ColEnd%, ColImages%) = column ; Up dating end column limit
    Difference = (Temp%(ColEnd%, 1) - Limits%(ColEnd%, 1))
    IF Difference > 5 THEN
        Transfer limits between array locations
    END
    Limits%(ColEnd%, ColImages) = Temp%(ColEnd%, ColImages%)
```

A similar routine dealt with obstacles above and to the right of a lower obstacle as shown in figure 5.12. In this case the start column limits were
tested. If the difference of the present limit compared to the limit of the last column scan was less than eight it was assumed a second object had been detected. As before, the limits obtained from the first image were then transferred to the second image location.

Figure 5.12: The obstacles in Alternative Positions.

As the image processing and pattern recognition routines were contained within a FOR NEXT loop, both obstacles were pattern matched and modelled.
5.6 Results.

A Screen Display: In the example shown below in figure 5.13, the camera was set 0.6m above the floor of the work cell (The top of the work bench) and the object was the polyhedron (A large matchbox).

![Image Number 1](image1.png)

<table>
<thead>
<tr>
<th>Filter</th>
<th>Thresholding</th>
<th>Edge Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="filter.png" alt="Filter" /></td>
<td><img src="thresholding.png" alt="Thresholding" /></td>
<td><img src="edge_detection.png" alt="Edge Detection" /></td>
</tr>
</tbody>
</table>

Area = 219
Perimeter = 60
Diameter = 24
Compactness = 16.43836
Thinness = .109589

Object Centre WRT robot X = -5.625 mm Y = 325.48 mm

---

**Figure 5.13: Identification of the Polyhedron.**

Figure 5.13 shows the identification of the object, the size of the object, the diameter and positional information. The changes in the image array after filtering, thresholding and edge detection can be seen with the matched template and percentage of error.

**Recognition Timings:** The recognition times for a horizontal cylinder using different sizes of array were recorded and are shown below. The processing included smoothing, thresholding, edge detection and template matching.

<table>
<thead>
<tr>
<th>Array Size</th>
<th>Rows x Columns</th>
<th>Processing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>440</td>
<td>22 20</td>
<td>1.4 seconds</td>
</tr>
<tr>
<td>2080</td>
<td>52 40</td>
<td>1.9 seconds</td>
</tr>
<tr>
<td>13,312</td>
<td>104 128</td>
<td>7.5 seconds</td>
</tr>
</tbody>
</table>

**Figure 5.14: Table of Recognition Timings for different Array Sizes.**
The following results were obtained using array reduction and are a comparison of the different timings for the different obstacles considered. The obstacles were placed with their longest length across the image array.

<table>
<thead>
<tr>
<th>Obstacle</th>
<th>Processing Time (milli seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Cylinder</td>
<td>950</td>
</tr>
<tr>
<td>Vertical Cylinder</td>
<td>450</td>
</tr>
<tr>
<td>Horizontal Polyhedra</td>
<td>920</td>
</tr>
<tr>
<td>(Maximum Area)</td>
<td></td>
</tr>
<tr>
<td>Horizontal Polyhedra</td>
<td>910</td>
</tr>
<tr>
<td>(Minimum Area)</td>
<td></td>
</tr>
<tr>
<td>Vertical Polyhedra</td>
<td>470</td>
</tr>
<tr>
<td>Cube</td>
<td>750</td>
</tr>
</tbody>
</table>

Figure 5.15: Table showing typical recognition times for various obstacles.

The processing times and the number of BLOCKED nodes detected were recorded for various obstacles in a variety of positions. The positions where the data was recorded is shown below and is reproduced from figure 4.6.

Figure 5.16: Reference Positions.
Processing Times: The times shown below in figure 5.17 included the recognition of the obstacle, transformation into discrete 3-D joint space and the transmission of the BLOCKED nodes to the path planning computer.

Times are in seconds.

<table>
<thead>
<tr>
<th>Reference Position</th>
<th>Horizontal Cylinder</th>
<th>Horizontal Polyhedra (Minimum Area)</th>
<th>Horizontal Polyhedra (Maximum Area)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>4.85</td>
<td>3.41</td>
<td>4.18</td>
</tr>
<tr>
<td>1.2</td>
<td>4.8</td>
<td>3.64</td>
<td>4.15</td>
</tr>
<tr>
<td>1.3</td>
<td>4.65</td>
<td>3.59</td>
<td>3.81</td>
</tr>
<tr>
<td>1.4</td>
<td>4.85</td>
<td>3.69</td>
<td>4.8</td>
</tr>
<tr>
<td>1.5</td>
<td>4.91</td>
<td>3.79</td>
<td>4.59</td>
</tr>
<tr>
<td>1.6</td>
<td>4.78</td>
<td>3.58</td>
<td>4.26</td>
</tr>
</tbody>
</table>

Figure 5.17(a): Processing Times for the Obstacles with larger areas in the X,Y plane

<table>
<thead>
<tr>
<th>Reference Position</th>
<th>Vertical Polyhedra</th>
<th>Vertical Cylinder</th>
<th>Cube</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>3.48</td>
<td>4.56</td>
<td>3.78</td>
</tr>
<tr>
<td>2.2</td>
<td>3.78</td>
<td>4.78</td>
<td>3.48</td>
</tr>
<tr>
<td>2.3</td>
<td>4.11</td>
<td>4.98</td>
<td>4.11</td>
</tr>
<tr>
<td>2.4</td>
<td>4.63</td>
<td>5.6</td>
<td>4.63</td>
</tr>
<tr>
<td>2.5</td>
<td>4.52</td>
<td>5.7</td>
<td>4.52</td>
</tr>
</tbody>
</table>

Figure 5.17(b): Processing Times for the Obstacles with smaller areas in the X,Y plane
5.7 Discussion and Conclusions.

The programs developed used various processing techniques to provide simple 2-D pattern recognition through the use of template matching. Data in a form capable of performing 3-D manipulation was then generated from the 2-D images. Problems were overcome by using mathematical image enhancement techniques applied to the image information from the camera.

Amongst the mathematical techniques to enhance the image, frequency analysis proved to be an important concept, but it could not be used in the real time system because of processing limitations.

The recognition of objects and their transformation into robot joint space was achieved. Data in a form capable of being analyzed by the Path Planner and Path Adapter was initiated. The program used the vertically mounted camera to analyze objects in the work space as binary images. The data was pre-processed and then used for template matching. If sphere models were being used then the object was initially modelled by the minimum bounding sphere, if time allowed, a more accurate spherical representation was presented. Later work used the 2-D slices discussed in detail in chapter four.

Parameters (iv) and (v), (Compactness and Thinness), were used to distinguish between obstacles of similar size but of a toroidal nature. They were also useful when shadow effects altered the absolute size of the obstacle as the ratios tended to be more repeatable than the other parameters.
An obstacle was "identified" when it satisfied the minimum error criterion set within the software. It was possible for an unknown object to satisfy the conditions set and to be incorrectly identified, although the program always correctly identified the maximum diameter and the minimum bounding cylinder necessary for obstacle avoidance without the need for the recognition processes.

The reduction in the array size reduced the recognition time from $>7$ seconds to $<2$ seconds. This is demonstrated for the horizontal cylinder in figure 5.14 using a forced array size. Using reduced array sizes selected by the system, the timings for different obstacles positioned in the same place in the $X,Y$ plane are shown in figure 5.15. The objects with a larger area tended to take a longer time, as the array sizes were larger.

The total processing times are shown in figure 5.17. The timings varied from 3.41 seconds for the polyhedron with minimum area showing to 5.7 seconds for the vertical cylinder. The large obstacles, for example the cylinder, tended to take longer to transfer into 3-D joint space.
Chapter Six

ROBOT PATH PLANNING

6.1 Introduction.

The different aspects of the work described in chapters three, four and five were combined to produce a robot work cell. Within this cell, a robot was able to move under a camera which viewed a section of its work space. The image produced by the camera was captured by a computer and analyzed. The analysis produced a list of configurations which were blocked to the robot.

This chapter describes the evolution of the Path Planner. The problem involved moving the robot from one place to another while avoiding collision with obstacles. Initially the 2-D Space problem was considered for the prototype robot base and a simulated joint and link. Later the work was extended to 3-D Space for use with a Mitsubishi RM 501 robot.

START and GOAL configurations were entered by a human operator. The Path Planning Computer used these configurations and the obstacle data to calculate a path for the robot to move safely through the field of vision of the camera. A third computer used this path to direct the movement of the robot around the work cell. Only a section of the robot work area was covered by the vision sub-system and other sub-systems would be required to cover the whole work area.

The position problem was solved for the START and GOAL positions using the forward kinematics solution presented in section 3.6. The result specified how much each joint had to be rotated to effect the desired movement and was an initial path. When no obstacles were detected by the vision system the direct
trajectory locus from START to GOAL was selected.

The control flow for the robot system was:-

\[
\begin{align*}
\text{Task} & \quad \text{Set by a human programmer and entered into the main computer.} \\
\text{Trajectory locus} & \quad \text{Calculated by the Path Planner in the main computer.} \\
\text{Robot co-ordinates} & \quad \text{Extracted from the trajectory locus at the Supervisory level of the Robot Controller.} \\
\text{Robot Trajectory & Robot Movements} & \quad \text{Generated by the Controller.}
\end{align*}
\]

In general, a TASK description contains information of the type shown below:

\[
\begin{align*}
\text{START} \\
\text{Wait (for) PART} \\
\text{Pick PART from BELT} \\
\text{Put PART into BOX} \\
\text{Goto START}
\end{align*}
\]

The research described in this dissertation was concerned with improvements to Path Planning methods which would allow automatic generation of the robot movements required to achieve such a TASK. Udupa(1977) divided path planning into three stages. This type of description has been used by many researchers to describe the problem and the path planning work described in this dissertation used similar stages. These were:-

(i) Path feasibility.
(ii) Approach planning.
(iii) Path planning.

These stages are described:-
(i) **Path Feasibility.** The task was a series of configurations through which the robot moved in order to carry out the task. The configuration of the robot at VIA points and the goal position were checked for feasibility. Positions which were out of the robot's work-space, or which would cause collisions with obstacles or the static environment, were not accepted from the human programmer.

(ii) **Approach Planning.** Approach paths were paths which moved from positions with clearance from obstacles to goal positions close to surfaces. In industrial applications approach paths tend to be short. They are related to machine geometry, and are calculated for specific machine configurations. The work described in this dissertation did not consider these paths in detail.

(iii) **Path planning.** The remainder of this chapter deals with the work completed concerning the Path Planning problem. To simplify the problem, path planning was initially completed for a two degree of freedom manipulator using the initial test rig described in chapter three, that is for the prototype base joint and a simulated shoulder joint and a simulated single link. This is described in section 6.2. The path planning methods were then extended to 3-D space and this is described in sections 6.3 and 6.4.

- **Configuration of the equipment:** As described in chapter five the camera was placed over a section of the workplace. The coordinates of the camera had to be referred to the joint coordinates of the robot. During the initial work with the prototype robot base an origin was defined as the centre of the base and the simulated shoulder joint and this was used as the origin for all coordinate systems. In all cases the cartesian coordinates were determined relative to the origin.
with X running from front to back of the work bench, Y running from left to right along the bench and with a vertical Z axis. The camera was positioned so that the camera base was at \(Y = 170\) mm and both were central on the X axis. The base originally used was 25 mm thick so that the surface was raised to \(Z = -225\) mm.

In Figure 6.1 the robot is displayed with the waist \((\theta_1)\) at 90°, the shoulder \((\theta_2)\) at 60° and the elbow \((\theta_3)\) at 120°.

**Figure 6.1: The Joint Angles.**

*Figure 6.2 shows a plan view of the system components when configured for use with the camera base and a front lighting system.*

**Figure 6.2: Plan View.**
6.2 Path Planning for a Single Link Manipulator.

Simple generalisation from the 2-D problem to the 3-D problem was not possible, but solving the 2-D problem was a useful introduction to the general Path Planning problem. The two degree of freedom manipulator used for this part of the work consisted of two joints co-located at the origin; the prototype base joint $\theta_1$ and a simulated shoulder joint $\theta_{sim}$ and a simulated single link $L_{sim}$.

Two obstacle models were considered during the work in 2-D space; a sphere and a simple parallelepiped (A solid bounded by parallelograms).

**Statement of the 2-D problem.** The robot was to move through a set of via points from a START to a GOAL configuration avoiding obstacles and without violating geometric constraints. Without considering orientation, the purely position problem, $p(t) = p(x_1,y_1 \ x_2,y_2 \ x_3,y_3 \ \text{etc})$ can be stated for a move from one position to the next as :-

From

$P_n = (X_n,Y_n)$, that is $P_n = (\theta_{i(n)},\theta_{2(n)})$

Move to

$P_{n+1} = (X_{n+1},Y_{n+1})$, that is $P_{n+1} = (\theta_{i(n+1)},\theta_{2(n+1)})$

Where

$P_n$ is the $n^{th}$ position in space.

$\theta_{ij}$ is the $j^{th}$ position of joint $i$ in a trajectory locus.

$X$ and $Y$ are cartesian coordinates.

Two solutions to this 2-D problem were considered:-

152
(a) A local and heuristic method: Having retrieved the data for the world model from a disc file containing simulated obstacles, the program had successive tasks provided. Each TASK was input by a human operator and consisted of the initial, intermediate and final coordinates of the ForeTip. The path trajectory locus was calculated from the TASK description and the model data. The trajectory locus consisted of robot coordinates and these were down loaded to the robot controller.

The approach path was defined as follows:

(i) Position the GOAL 10 mm above the final position.
(ii) Move down in a straight line at 1/3rd normal speed.
(iii) Simulate gripping the part.

When moving away from the final position a similar motion was used to simulate lifting a part 10 mm at 1/3rd normal speed. The approach paths were defined by a few lines of program code written by the human operator. If a new approach path was required it was simple to modify this code.

The START configuration was the first node on a graph. The path cost function was set to 0 for the START node and $FFFF_H$ for the other nodes. Paths around the obstacles were represented by nodes for both the sphere and parallelepiped models.
After the graph had been initialised for searching, the direct path from START to GOAL was tested. If this path was blocked then the algorithm selected new nodes until either the GOAL was reached, or all nodes had been tested. If all nodes were tested and no path was found then it was assumed that no path existed.

From the START node paths were considered to all the other nodes. Each of these paths were tested for collisions with obstacles. If a path was clear, the cost of the path was calculated. If the cost of the path to a new node was less than any previous path then the new cost and the previous node were stored for the new node. Once each node had been tested, the node was recorded on a list so that the node was not retested. This list formed the trajectory locus that was passed to the robot controller.

The method used for each of the models is described. The graph searching methods were based on that of Hart(1968). The simulated arm was modelled as a line segment fixed at the origin with a skin some constant distance from this line segment. The method for each of the models is described:

(i) **Parallelepiped:*** Each obstacle was represented as a parallelepiped by defining the corner points of the obstacle. The obstacles created an obstructed segment of the robot work space bounded by lines from an apex coincident with the origin. This segment was simplified to be bounded by four sides and then grown by the radius of the arm. The problem was reduced to a two degree of freedom line moving around these grown segments. This is shown in figure 6.3.
Figure 6.3: The Obstructed Volume due to a Parallelepiped.
From figure 6.3 it can be seen that using these models, the shortest path consisted of planes between these obstacles and the surfaces of particular obstacles. To determine the path a heuristic method of graph searching was used.

From the START configuration a graph was generated. Each simplified parallelepiped could be traversed via several edges and faces. Each edge had a node associated with it. In fact only two paths were possible around each obstacle and thus four paths were possible between obstacles. The routine to find the four bounding configurations for the base and simulated shoulder joint which corresponded to edges of the four bounding sides is shown.

\[
\begin{align*}
\text{ShoulderMax} &= -10; \quad \text{ShoulderMin} = 200 \\
\text{BaseMax} &= -200; \quad \text{BaseMin} = 200 \\
\text{FOR} \quad \text{Count} &= 1 \quad \text{TO PolyNo} \\
\quad \text{FOR} \quad \text{Corner} &= 1 \quad \text{TO NumberOfCorners}[\text{PolyNo}] \\
\quad & \quad \text{Base[Angle]} = ATN(\text{Corner}[x]/\text{Corner}[y]) \\
\quad & \quad \text{FindMod}(\text{Corner}[x],\text{Corner}[y],\text{Mod}) \\
\quad & \quad \text{Shoulder[Angle]} = ATN(\text{Corner}[z]/\text{Mod}) \\
\quad & \quad \text{IF} \quad \text{Base[Angle]} > \text{BaseMax} \quad \text{THEN} \\
\quad & \quad \quad \text{BaseMax} = \text{Base[Angle]} \\
\quad & \quad \text{ELSE IF} \quad \text{Base[Angle]} < \text{BaseMin} \quad \text{THEN} \\
\quad & \quad \quad \text{BaseMin} = \text{Base[Angle]} \\
\quad \text{END IF} \\
\quad & \quad \text{IF} \quad \text{Shoulder[Angle]} > \text{ShoulderMax} \quad \text{THEN} \\
\quad & \quad \quad \text{ShoulderMax} = \text{Shoulder[Angle]} \\
\quad & \quad \text{ELSE IF} \quad \text{Shoulder[Angle]} < \text{ShoulderMin} \quad \text{THEN} \\
\quad & \quad \quad \text{ShoulderMin} = \text{Shoulder[Angle]} \\
\quad \text{END IF} \\
\text{Next Corner} \\
\text{NEXT Count}
\end{align*}
\]

A node was assigned to each of these bounding configurations. From the START configuration each node was tested against a cost function, beginning with the configurations with the lowest base angle. The cost function was defined such that:

\[
\text{Cost} = d_{\text{Old-New}} + \Sigma d_{\text{GOAL-New}} - \Sigma d_{\text{GOAL-Old}}
\]
Where,

\[ d_{\text{Old-New}} = \text{Distance from old node to new node.} \]
\[ \Sigma d_{\text{GOAL-New}} = \text{Sum of the distances between nodes from the new node to the GOAL.} \]
\[ \Sigma d_{\text{GOAL-Old}} = \text{Sum of the distances between nodes from the old node to the GOAL.} \]

Distance was initially the total movement of both joints, and for later work it was assumed that both joints were capable of similar accelerations and velocities, so that distance was the largest difference of the two joints. Thus the cost between the old node and the new node was the extra distance that the robot was required to travel plus the new distance to the GOAL compared with the old distance.

(ii) **Spheres:** The obstacles were represented as spheres by defining the centre and radius of the sphere in cartesian coordinates. The radius was grown by the radius of the simulated arm and the problem reduced to a line segment with two degrees of freedom moving among cones. As the robot had two degrees of freedom the spheres formed blocked cones emanating from an apex coincident with the origin to form circles on the bounding sphere of the robot work area as shown in figure 6.4. From figure 6.4 it can be seen that in this case the shortest path from START to GOAL consisted of planes between the cones and arcs around the cones. From the START configuration a graph was generated. Each cone could be traversed in a clockwise or anticlockwise direction, so each obstacle had only two nodes associated with it. Thus two paths were possible (one to each node) and four paths were possible between cones. A routine to find the two nodes for a sphere obstacle is shown over the page:
Figure 6.4: The Cones Obstructed by a sphere Model.
The robot did not follow arcs around the circular segments but moved in planes as shown in figure 6.5. This movement was simpler. A similar cost function was defined for each node so that

\[
\text{Cost} = d_{\text{Old-New}} + \sum d_{\text{GOAL-New}} - \sum d_{\text{GOAL-Old}}
\]
Thus the cost between the old node and the new node was the extra distance that the robot was required to travel plus the new distance to the GOAL compared with the old distance. The node with the lower cost was selected in each case until the GOAL node was reached.

Figure 6.5: The Planar Movement Around Circular Segments (Viewed from the Origin).

(b) A Global Method: The working area was divided into a discrete graph of joint angles with nodes at increments of 5 degrees, so that the base angles were -180°, -175°, -170°… 170°, 175°, 180° and the simulated joint angles were 0°, 5°, 10°… 170°, 175°, 180°.

The method tested this discrete graph of 2-space from the START configuration to the GOAL configuration, checking each node for an obstruction. Data about each node was stored in a variable NodeStatus% as shown in the table in figure 6.6.
When the program started, NodeStatus% was defined as an array of the graph of joint angles. Each node within the array was set to clear. The BLOCKED nodes were loaded from a disc file 'ROBFILE2.DAT' and bit 6 of these nodes was cleared. With the BLOCKED nodes initialised the START and GOAL nodes were requested from the operator and the START node was placed onto a list.

The method was to test nodes around the graph of joint angles. From the START, each of the four nearest nodes was tested against a cost function to see if they were closer to the GOAL. Any nodes that were nearer were added to the list. The cost function was: \[ \text{Cost} = (\Sigma d_g)^2 + \Sigma d_s \]

where,
\[ \Sigma d_g = \text{Sum of the distances between nodes from the GOAL.} \]
\[ \Sigma d_s = \text{Sum of the distances between nodes from the START.} \]

The node stored at the top of the list was the one that had taken the least moves to arrive at its present location compared with other nodes which were equal distances from the GOAL configuration. This node then had its nearest three neighbours tested and so on until the test node was the GOAL node.

![Diagram of a Global Path in 2-D Space](image)

*Figure 6.7: A section of a Global Path in 2-D Space.*

Each time a node was added to the list, bits 1 to 3 of NodeStatus% were filled to record the direction moved to arrive at the node. When the test node had arrived at the GOAL the path was retraced by testing these bits. The list was displayed on the screen. A typical section of a path around a planar simulated complex obstacle is shown in figure 6.7.
6.3 Extension to 3-SPACE Local Heuristic Methods.

The methods described in section 6.2 were extended to plan paths for the three lower joints of a Mitsubishi robot. In 3-D space the simple obstacle models appeared as one or more complex shapes in joint space.

For the local and heuristic methods, either of the 2-D planning techniques described in the previous section were used to plan a path for the lower joints $\theta_1$ and $\theta_2$. The problem was then reduced to finding a path through a new transformed 2-D space for the joint $\theta_3$. This new problem could be solved by a local heuristic method for searching this new 2-D space.

From the 2-D path planning methods a series of configurations of the upper arm had been produced. Between these configurations the upper arm moved in planes. The forearm path planning algorithm had to avoid obstacles. An initial START configuration and a final GOAL configuration were known. In between these configurations there were configurations where the position of the upper arm was known but the forearm position was undefined.

Sphere models were easily extended to 3-D space but the parallelepiped was complex and required excessive processing. This was because the sphere models were effectively solid models and only one check was required to see how close the line representing the Forearm was to the centre of the sphere. The parallelepiped models were effectively wire frame models defined at their corners so that many calculations were required to see if the Forearm violated the obstacle space. For the remainder of this section sphere models are assumed.
Initially a trajectory locus in which the elbow joint $\theta_3$ moved directly to its final configuration was considered. This path was discretised, only allowing $\theta_3$ to move in multiples of 5 degrees between movements. The positions along this path were then checked for collisions. If the path for the forearm was obstructed (as shown in figure 6.8), then a new path was calculated.

For the range of configurations through which the forearm moved the sub-range where collision could occur was determined. The configurations at either end of this sub-range were noted. (A and B in figure 6.8). The range of movement of the base between these points was determined and points were proposed a similar distance above and below the configuration C, midway between A and B on the graph. If one of these configurations was CLEAR, (in this case node D), this was adopted as a node and therefore a via-point for the path, otherwise the distance from C was doubled and the new configurations checked. The new forearm path was then tested at 5 degree intervals and the process repeated if the path was obstructed.

![Figure 6.8: Local and Heuristic Forearm Path Planning](image)

**Figure 6.8: Local and Heuristic Forearm Path Planning.**
6.4 Extension to a 3-SPACE Global Method.

The 2-D global method described in section 6.2 for the prototype robot base and simulated shoulder joint was extended for use with the Mitsubishi robot. The working area was divided into a 3-D graph of joint angles with nodes at increments of 5 degrees, so that:

(i) The base angles were 30°, 35°, 40°..., 150°.
(ii) The shoulder angles were -30°, -25°, -20°..., 110°.
(iii) The elbow angles were 0°, 5°, 10°..., 90°.

Data about each node was stored in the variable NodeStatus% which was extended to include detail on the elbow movement as shown in figure 6.9.

<table>
<thead>
<tr>
<th>NodeStatus%</th>
<th>Bit Value</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128 80H</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>64 40H</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>32 20H</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>16 10H</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8 08H</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4 04H</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2 02H</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1 01H</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Level</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not On the List</td>
<td>On the List</td>
</tr>
<tr>
<td></td>
<td>BLOCKED</td>
<td>CLEAR</td>
</tr>
<tr>
<td></td>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td></td>
<td>Positive Direction</td>
<td>Negative Direction</td>
</tr>
<tr>
<td></td>
<td>BaseStill</td>
<td>Base Movement</td>
</tr>
<tr>
<td></td>
<td>Shoulder Still</td>
<td>Shoulder Movement</td>
</tr>
<tr>
<td></td>
<td>Elbow Still</td>
<td>Elbow Movement</td>
</tr>
</tbody>
</table>

**Figure 6.9:** Table of the detail of the Extended flag 'NodeStatus%'.

164
When the program started, NodeStatus% was defined as a 3-D array of the graph of joint angles. Each node within the array was initially set to CLEAR and the BLOCKED nodes were loaded from a data file 'ROBFILE3.DAT' or received from the vision system. With the graph initialised the START and GOAL nodes were requested from the operator and the START node was placed onto a list.

From the START configuration each of the six nearest nodes were tested to see if they were closer to the GOAL. The closest node was added to the list. This was repeated at each new node on the graph until the GOAL was reached. The cost function used for the 2-D case was extended for use in the 3-D graph and was

$$\text{Cost} = (\sum d_g)^2 + \sum d_s.$$  

Where,

$$\sum d_g = \text{Sum of the distances between nodes to the GOAL.}$$

$$\sum d_s = \text{Sum of the distances between nodes from the START.}$$

Each time a node was added to the list, bits 0 to 3 of NodeStatus% as shown in figure 6.9 were filled to record the direction moved to arrive at the node. When the test node had arrived at the GOAL the path was retraced using these bits. This list was displayed on the screen.

In the above method and the 2-D method described in section 6.2 only one joint was moved in each test. This needed to be improved to smooth the robot movement, decrease the number of nodes in the path and increase the speed of the movement. Two methods were considered:-
(i) Test the movement of more than one joint during path planning.

(ii) Find the diagonals within the 3-D graph once a path had been planned.

Before deciding which method to adopt, processing speed was considered. A diagonal on the global 3-D graph corresponded to more than one joint being in motion. If the diagonals on the graph were tested from a node then as the path was planned there would be a total of 26 tests for each node. The 26 tests were made up of: $6 + 12 + 8 = 26$, that is the 6 nodes tested by the previous method, the 12 nodes where two joints moved and the 8 nodes where all three joints moved. This gave a total of 26 nodes to test each time. These tests were considered for the three possible situations:

(i) If one joint moved in the same direction 3 times then $3 \times 26 = 78$ tests would have taken place. 60 more than before.

(ii) If one joint moved in the same direction twice and then another joint moved once, then a two joint diagonal would have been found in the graph so $52$ tests would have been computed. 34 more than before.

(iii) If each joint moved once, a three joint diagonal would have been found with only 26 tests, but this was still 8 more than before.

This method was unattractive and the method selected was the processing of the planned path data. A routine was written to scan the path data and modify it to include the diagonal movement of more than one joint. Each joint was tested in turn to find where the angle changed for a second time. In the first example shown over the page, this would be 4 nodes down from the start node, and in the second example it was only 3 nodes down. The node before would be moved up to the start node and the process restarted from that node. In this way the paths shown on the right were produced.
The following routine achieved the path modification:

' include diagonal movement
scanpos% = 0
DO
secchange% = 4  ' the offset of the second change in angle
FOR lp1% = 0 TO 2
  numchanges% = 0  ' the number of angle changes of the joint
  FOR Ip2% = 1 TO 3
    diff% = path%(scanpos% + Ip2%, lp1%) - path%(scanpos% + Ip2% - 1, lp1%)
    IF diff% <> 0 THEN
      numchanges% = numchanges% + 1
      IF numchanges% = 2 AND Ip2% < secchange% THEN secchange% = Ip2%
    END IF
  NEXT
NEXT
IF secchange% > 2 THEN  ' is there a diagonal
  path%(lp1%, 0) = path%(lp1% + secchange% - 2, 0)
  path%(lp1%, 1) = path%(lp1% + secchange% - 2, 1)
  path%(lp1%, 2) = path%(lp1% + secchange% - 2, 2)
END IF
scanpos% = scanpos% + 1
LOOP UNTIL scanpos% = pathpos%

A second improvement to the path was to remove the repeated nodes due to constant joint motion. This is demonstrated below.
This routine involved finding parts of the path where joint motion was constant and removing all the nodes in between the beginning and end of this movement.

\[
\begin{align*}
\text{remove constant change nodes} \\
\text{path2pos\%} = 1 \\
\text{FOR} \ Ip1\% = \text{pathpos\%} - 1 \ TO \ 1 \ \text{STEP} \ -1 \\
\text{find the change in angle of present and previous nodes} \\
\text{diff0u\%} = \text{path\%(lp1\% + 1, 0)} - \text{path\%(lp1\%, 0)} \\
\text{diff0d\%} = \text{path\%(lp1\%, 0)} - \text{path\%(lp1\% - 1, 0)} \\
\text{diff1u\%} = \text{path\%(lp1\% + 1, 1)} - \text{path\%(lp1\%, 1)} \\
\text{diff1d\%} = \text{path\%(lp1\%, 1)} - \text{path\%(lp1\% - 1, 1)} \\
\text{diff2u\%} = \text{path\%(lp1\% + 1, 2)} - \text{path\%(lp1\%, 2)} \\
\text{diff2d\%} = \text{path\%(lp1\%, 2)} - \text{path\%(lp1\% - 1, 2)} \\
\text{IF NOT (diff0u\% = diff0d\% AND diff1u\% = diff1d\% AND diff2u\% = diff2d\%)} \ THEN \\
\text{path2\%(path2pos\%, 0) = path\%(lp1\%, 0)} \\
\text{path2\%(path2pos\%, 1) = path\%(lp1\%, 1)} \\
\text{path2\%(path2pos\%, 2) = path\%(lp1\%, 2)} \\
\text{path2pos\% = path2pos\% + 1} \\
\text{END IF} \\
\text{NEXT}
\end{align*}
\]

Smoothness was then considered. Using a "string pulling" technique similar to that described by Dupont(1988) the path could be shortened. With the paths determined above, the movement of the robot was smoother as it left the last obstacle and headed for the GOAL than during the rest of the path. This was because the planner was drawn towards the GOAL and had to work around the obstacles but could move easily away from the last obstacle.

This is shown in figure 6.10. A new path was determined from the configuration midway between the START and the GOAL node configurations, back to the START.

The original path was stored in an array, the START was re-defined as the GOAL and a new path was determined. The two paths were combined by reversing the new path and adding the end of the original path from the furthest point onwards. This method produced a smoother path as demonstrated for the planar case in figure 6.11.
**Figure 6.10**: An Initial Path in the plane.

**Figure 6.11**: Revised Path in the plane after "String Pulling".
If the direct path from START to GOAL was blocked then the technique in 3-D space scanned two lines from the START and GOAL, travelling along the joint in which there was most difference. The code is shown below:

```plaintext
FUNCTION ScanLine1% (s1%, s2%, s3%, e1%, e2%, e3%)
    hit% = 0
    FOR lp% = s1% TO e1%
        IF NodeStatus%(lp%, CINT(s2% + (lp% - s1%) * (e2% - s2%) / (e1% - s1%)),
            CINT(s3% + (lp% - s1%) * (e3% - s3%) / (e1% - s1%))) AND 64 <> 0 THEN hit% = 1
    NEXT
    ScanLine1% = hit%
END FUNCTION
```

A revised technique produced via points to smooth the path and increase the speed of the trajectory produced. The joint with most difference was usually the base so this is used in the description; a line was tested from the START with the shoulder and elbow staying at the angles of the START configuration. Another line was scanned in the same manner but from the GOAL. In this way the range where a collision would occur was found. This was similar to points A and B in figure 6.8 on page 163. The node on the original path which was a maximum distance away from the START configuration was used to define the angles for joints $\theta_2$ and $\theta_3$ for two via-points. The base angles $\theta_{1v(i)}$ and $\theta_{1v(ii)}$ for these via-points were the base angles at the extremes of the range where a collision would have occurred.

A line was scanned from the START to GOAL of the obstacle at the shoulder and elbow angles of the furthest point, that is between the two proposed via-points. If this line was obstructed then the shoulder and elbow angles were moved out until a clear path was found. The whole new path including the two new via-points was then tested, and if clear, the path was passed to the Robot Controller.

For the simple obstacle models used, this path contained four nodes and the method worked in all the practical situations tested.
Once the path had been planned as a trajectory locus in joint space, the configurations were passed to the robot control computer. Although the robot path was discretised, the intermediate configurations were close to each other so that the trajectory resulted in the robot tracing a curve in joint space that was close to the planned trajectory locus.

The path was transferred initially using a floppy disc and later by an RS 232 link. The path was a set of joint angles in the order in which they had to be moved. The path movement routines in the controller moved each joint to within 80 encoder readings or 1° for the configurations specified at VIA POINTS and to within 4 encoder readings or 0.05° at the GOAL configuration.

To demonstrate the path planning routines using the Mitsubishi robot, the Robot Controller carried out a simulated task in the waist range +30° to -30° until a path was received from the main computer. Then the robot moved along the path to the destination and returned to the simulated task until a new path was received.

The robot trajectory locus consisted of a series of robot configurations in joint space. The robot control computer also operated in joint space but the controller did not always interpolate between coordinate positions in a predictable manner. To avoid this problem, intermediate configurations were generated by the path planner as robot coordinates. This meant the robot moved only small distances between defined configurations and hence the deviation from the path was negligible.
6.6 Results.

In chapters four and five, polyhedral models were selected for the static environment in the global path planner. The BLOCKED nodes due to the polyhedral models were calculated once and stored in the main planning computer. These nodes were loaded at the beginning of each planning session.

In later work the camera was raised so that it was out of reach of the robot and the static environment was simplified to a single polyhedron modelling the floor of the work place (The top of the work bench). The position of the ForeTip for the BLOCKED configurations are shown below in figure 6.12.

![Figure 6.12: The BLOCKED nodes stored for the Static Environment.](image)

The model of the static environment reduced the volume of discrete space left available to the path planner. The position of the ForeTip for the remaining CLEAR configurations are shown in figure 6.13.
The space available to the path planner was further reduced by the introduction of obstacles into the workplace. Figures 6.14 to 6.16 show the position of the ForeTip for the blocked configurations due to two types of obstacle, the cube and the cylinder (a beer can spray painted black). The 2-D slice model was used in all cases.
The phrase 'Real-time' has several interpretations and in this dissertation it is assumed to mean that the solution of the path planning problem takes less time than the robot takes to execute the path.
The total processing time for the system depended on:-

(a) The number of obstacles.
(b) The position of the obstacles.
(c) The size of the obstacles.

In practice solutions to the path planning problem were always found by both methods and the calculation time was within the limits for real time operation.

For a typical task, such as:

Simulate a task between base angles +30° and -30°, then "pick up" a part at one extreme of the area covered by the vision system, (base angle +30°) and move it to the other extreme (+150°), while avoiding an obstacle.

The robot trajectory tended to take >9 seconds and the total calculation time after inserting an obstacle was <9 seconds. The internal timer was interrogated during path processing. The local and heuristic 3-D space path planner tended to produce paths using the sphere obstacle representation within 3 seconds. The global path planner produced paths using 2-D slices in joint space within 1.4 seconds.

To show the paths planned by the robot within this dissertation, the robot was simulated using the forward kinematic solutions for the robot described in chapter three. The graphics facilities of Quick BASIC were used to draw the robot on the screen and this was captured using the GRAB feature of the Word Perfect word processor. The simulation methods were those described by Moore(1990). Figures 6.17 to 6.24 show an example of the robot arm moving along a planned path around a point object at X=0, Y=210, Z=50.
6.7 Discussion and Conclusions.

To achieve the simulation displayed in figures 6.17 to 6.28, the basic forward kinematics described in chapter three were adapted to the \((X, Y)\) coordinates of the screen. This was achieved using the two lines of code:

\[
\text{LINE (x,y) - (x-sf*SIN(\theta_1)*220*COS(\theta_2), y - sf*220*SIN(\theta_2))}
\]

\[
\text{LINE (x-sf*SIN(\theta_1)*220*COS(\theta_2), y - sf*220*SIN(\theta_2)) -}
\]
\[
\text{(x-sf*SIN(\theta_1)*(220*COS(\theta_2) + 160*COS(\theta_3)) + \theta_3 - \pi)),}
\]
\[
y - sf*(220*SIN(\theta_2) + 160*SIN(\theta_2 + \theta_3 - \pi))
\]

To this solution was added the skin of the robot. The path shown used the global path planning routines which produced paths that incremented joints by 5° per move.

The calculation time for both the planning methods was adequate, but the local and heuristic method tended to take twice as long compared to the global method. This was partly due to the calculations for the static environment being calculated every time for the local method. For the example, both methods took less than 9 seconds to plan a path. This compared with programming times of 5 to 20 minutes for programmers using the GRASP CAD off line robot programming package.

In practical environments the path planning computer always produced satisfactory paths in "real time" but the performance of the automatic programming system could be improved by:

(a) A cartesian robot which would simplify the algorithms.
(b) A parallel processing computer.
(c) Improvements in the software.
(d) Faster processing speeds.
The advantage of the methods described were that they used simple rules to solve problems which were difficult to analyze. The disadvantages of the local and heuristic methods were that they may not always find a path where one existed and the static environment was considered for every path. The fact that the forearm and upper arm were planned separately meant that many possible paths were not considered and hence the paths produced were unlikely to be the best path.

The local and heuristic programs were closely tied to the configuration of the robot. Some of the program code would need to be modified to accommodate the kinematic chain of a different robot. The 3-D global method was a more general solution to the problem and changing the robot would just require changing a module in the program.

For the global methods, the blocked nodes for the 3-D graph were loaded from disk or received from the vision system. The transformations were described in chapter five for the 2-D slice models and the sphere models. The 2-D slice models gave the best performance.

For the local heuristic methods the transformations took place during path processing and although the parallelepiped models performed better than the sphere models in 2-D space, in 3-D space, the sphere models were more efficient. This was due to the nature of the stored data concerning the obstacles, in that the sphere model was effectively a solid model while the parallelepiped was effectively a wire frame model.
Chapter Seven

ROBOT PATH ADAPTATION TO MINIMISE PEAKS IN JOINT MOTOR CURRENTS

7.1 Introduction.

Chapters three to six have described the main aim of the work presented in this dissertation:-

To create an automatic path planner in order to increase productivity.

Chapters seven and eight present a further aim of the work:-

To produce systems which would improve the performance of robots for which paths had already been planned by some means, automatic or otherwise.

Performance and speed can be achieved by specifying bigger or more efficient drivers, but this is not an efficient method. An increase in motor torque of 50% can only be expected to give a time reduction of up to 17%, (since time is proportional to the inverse of the square root of torque). To reduce time by a factor of two, the torque must be increased by a factor of four and heat dissipation by a factor of eight.

These solutions work by increasing the accelerations so that a stronger and more expensive robot structure is required. The methods presented in this chapter and chapter eight attempt to improve the robot path by removing
wasted energy.

The system initially employed two Apricot minicomputers and the Electro-Craft Corporation servo motor with velocity and position sensors mounted on the prototype robot base. The detail of the base is shown in Appendix D. The later work used the final apparatus described in chapter three.

- The Fundamental Concept for Chapters 7 and 8. Existing methods generate paths which may appear simple or obvious to the operator but which may not be efficient for the robot. Once a robot has been programmed to work within a complex system, possibly without the programmer ever seeing the work-place, it may be possible to improve the solution, thus providing the robot with a degree of autonomy. This chapter and chapter eight explore methods of adapting robot paths to produce faster and more efficient robot trajectories.

Recent commercial robotic CAD systems allow dynamic modelling of robots and machine tools within flexible assembly systems. Cell lay-out can be improved by testing various configurations and running different robot programs to optimise the cell design and product construction sequence. The programs produced can then be used through post-processors to directly program the robots on the factory floor.

Within computer design systems, complicated functions of space and time are decoupled from the operator and only simple descriptions of the desired motion are considered. The paths produced pass through "via-points" where
joint velocities may change abruptly.

Positions calculated by off line computer programming or CAD post-processors are represented in a coordinate frame (usually cartesian space) related to the joint variables by some homogeneous transform. In CAD systems, objective level programming tends to be used [Snyder(1975)], relating end effector position to a work-piece or object in the cell. This type of programming is easier for operators to visualise and model. Motions of the manipulator are described as motions of the tool frame relative to the world frame. Little consideration is given to the dynamics of the robot.

Once programmed with a set of space and time coordinates, a simple robot will carry out a sequence of motions with little sensing of the environment and with little correction once set in motion. The end effector paths produced by CADCAM (or off line programming) may be "too specific" and therefore the joint trajectories more complex than is required for these simple tasks. For first generation robot tasks, such precision is not always necessary. Other methods, such as teaching by following, produce a continuous path control that appears simple and ordinary to the human operator, but which may generate via-points which cause unnecessary current transients and torques in the electrical drives. Trajectories and paths may be further complicated by physical or safety restrictions for human teachers within the robots working volume.

In the work described in this chapter, improvement was achieved by having the software controller switch from the optimum look-up table to other
selected look-up tables, adapting the state space description of the controller. The robot trajectory and path were adapted accordingly.

The overseer described in chapter three received information from the Peak Detector and categorised the signals for use in the Path Adapter. Information on "Vital" and "Non-Vital" movements were entered by a human operator while entering the path at the keyboard. "Vital" movements were not changed by the adaption algorithms and represented sections of the robot path which passed close to obstacles or which were specific to some geometry in the workplace, for example; placing a part into a machine. The method is described in sections 7.3 and 7.4.

Before the path could be adapted to minimise peaks in joint motor currents the motor current had to be detected.

7.2 Monitoring of the Motor Drive Currents.

Monitoring the current in a D.C. motor for simple sensing is not in itself a novel idea. Usually it has been used for sensing large forces, although work at Portsmouth Polytechnic, presented by Naghdy et al (1985) had demonstrated its' use for sensing smaller forces.

Detailed analysis of actuator current was difficult and high levels of noise were present. This is shown in figure 3.8 on page 56. The current was sampled across a small resistance in series with the motor and the signal passed through a simple filter as described in Sanders et al (1987(b)) and in chapter three. Current transients were detected by considering the level.
and gradient of consecutive samples following a new destination signal from
the controller. New manipulator destinations were signalled as actuator
moves were generated by the controller.

7.3 Path Improvement: To Reduce Changes in Joint Direction.

Once the data had been analyzed, during the next repetition of the set of
movements the joint trajectories were adapted by changing the controller
look-up table. The simulated robot paths were thus modified to remove
some current peaks in the motor circuits. This was achieved by :-

(i) Running the actuator motor at low speed instead of stopping at non
vital points in trajectories which would normally mean the motor
stopped and restarted in the same direction (Irrelevant stops). This
was signalled by two consecutive current transients in opposite direct­
ions as the motor stopped and restarted.

(ii) Replacing the look-up table for an irregular stop with a table of a
low gain characteristic. This slowed the actuator so that non vital
destinations were never reached. Velocity was low so that current
transients were reduced when the joint stopped and then restarted in
the opposite direction. This was signalled by two consecutive current
transients in the same direction as the motor stopped and restarted.

The current waveforms for the two cases are shown in figures 7.1 and 7.2.
**Figure 7.1:** Current Waveform after an Irrelevant stop.

**Figure 7.2:** Current Waveform after an Irregular stop.
A new set of data for the servo motor current was recorded and successive sequences continued to minimise current transients by :-

(i) Increasing the velocity at zero error for irrelevant stops.
(ii) Reducing the gain characteristic for irregular stops.

In the case of a joint restarting in the same direction, (irrelevant stop), the joint controller changed to a look-up table which slowed the joint but which never reached a zero velocity. A small output was preset for zero position error. Arrival at a joint via-point was signalled just before it was reached and a new joint target and look-up table was selected as the simulated joint reached its' joint targets. The change to the current waveform is shown in figure 7.3.

In the case of a joint restarting in the opposite direction, the joint controller switched to a look-up table with a low gain characteristic. Arrival at a joint destination was signalled immediately and the joint moved at a lower velocity, never reaching the via point. A new joint target was selected when the other joints signalled their arrival. The change to the current waveform is shown in figure 7.4.
Figure 7.3: The difference in the elbow current waveforms after modifying the trajectory and the path.
Figure 7.4  The difference in the Elbow Current Waveforms after modifying the trajectory and the path.
**Software: The Controller.** Various non-linear control algorithms in sub-processes were loaded into the controller. Included was the optimum solution developed for the system by non-linearising the experimentally achieved, critically damped control algorithm. These were used to produce look-up tables in memory.

A repeated sequence of robot moves was entered by the human operator. During the first sequence the optimum solution was used for actuator control. As each joint angle target was passed, the controller signalled the main computer. A flag was associated with every move and the least significant bit stored the information concerning Vital moves:

\[1 = \text{vital} \quad 0 = \text{non-vital}.\]

In this work the flag was set by the human operator for each via-point while entering the sequence of moves. If consecutive actuator trajectories were "non vital", information on the type of non vital change was processed in the main computer and the program was modified in each sequence by passing control to the relevant look-up table for that type modification. The new look up table was then used between the via points.

**The Main Computer:** The main computer sampled the D.C current from the Servo Amplifier driving the actuator; either the prototype robot base or the Elbow joint of the Mitsubishi robot. The Peak Detector was a low level program module in the main computer which collected the information from the A/D board connected to the servo-amplifiers and passed the information to the Path Adaptor level. The Path Adapter accepted the
information from the peak detector and depending on the type of current peaks, advised the Overseer of possible changes to the joint trajectories. This information was passed to the supervisory level in the controller via the serial link. Only the transients associated with non-vital trajectories were considered. If two consecutive transients were non-vital, the relevant change of look up table for the type was selected. This data was transmitted via a serial link using the information from the flag associated with the move. The data included the move number as one byte and the type of non-vital transient as one bit in a second byte called the flag. The flag codes are described later in this section.

Once a move had been signalled, transients were identified in the path adapter by considering the relative level and gradient of four consecutive samples.

\[ i_n, i_{(n+1)}, i_{(n+2)}, i_{(n+3)} \]

Moves were signalled as the controller detected via-points being passed. Each via-point had a number, \( m \), associated with it and if the wave-form varied monotonically over three consecutive samples then a transient was detected. A forward gradient difference was calculated, so that:-

\[ \nabla i_n = i_{(n+1)} - i_n \]

where:-

\[ i = \text{instantaneous current.} \]
\[ n = \text{sample number.} \]
\[ \nabla i_n = \text{nth difference.} \]

Then,

\[ \nabla i_3 = i_{(n+3)} - i_{(n+2)} \]
\[ \nabla i_2 = i_{(n+2)} - i_{(n+1)} \]
\[ \nabla i_1 = i_{(n+1)} - i_n \]
If the wave-form was varying monotonically, then $V'_1$, $V'_2$ and $V'_3$ were of the same sign. When this was observed, the relative level and gradient was considered;

$$i_{(n+3)} - i_n = \sum_{n=0}^{n=2} V'_n$$

$$= \pm |\nabla_T|$$

where:-

- $i$ = instantaneous current.
- $|\nabla_T|$ = gradient over 4 samples.
- $n$ = sample number.

If $|\nabla_T|$ was greater than a constant $|k_g|$, a transient was detected and a transient marker was set in the main computer. The sign of $|\nabla_T|$ gave the direction of the actuator drive so that for each move, $m$:-

$$T_m = +1 \quad \text{or} \quad T_m = -1$$

If $|\nabla_T| > +k_g$ then $T_m := +1$

If $|\nabla_T| < -k_g$ then $T_m := -1$

If $|\nabla_T| < +k_g \& > -k_g$ then $T_m := 0$

where,

- $T_m$ = Transient marker.
- $|\nabla_T|$ = gradient over four samples.
- $k_g$ = gradient constant.
- $m$ = move number.

The code in the main computer for detecting a transient is shown over the page.
\[ n = n + 1 \]
\[ \text{Sample(Gradient())} \]
\[ \text{Gradient}(n) = \text{OldCurrent} - \text{Current} \]
\[ \text{OldCurrent} = \text{Current} \]
\[ \text{Newsign\%} = \text{SGN(Gradient}(n)) \]

IF NewSign\% = OldSign\% THEN

\[ \text{GradTotal} = \text{OldGradient} + \text{Gradient}(n) \]
\[ \text{Count\%} = \text{Count\%} + 1 \]

IF GradTotal > \text{PosTransConst} OR GradTotal < \text{NegTransConst} THEN

\[ \text{TransMarker\%(m)} = \text{NewSign\%} \]
\[ \text{Count\%} = 0 \]

END IF

END IF

ELSE

\[ \text{GradTotal} = 0 \]
\[ \text{Count\%} = 0 \]

END IF

\[ \text{OldSign\%} = \text{NewSign\%} \]

Considering two consecutive transients associated with NON-VITAL moves, the relevant change was signalled to the controller. If noise were introduced into the system and a reading could not be taken because the signs of the gradient changed during the sampling periods, a recalculation took place in the next pass.

Signals were categorised as "Irregular", "Unnecessary" or "No-change" depending on the peaks reported by the peak detector. The two lowest bits of the flag were used:-

\[
\begin{align*}
\text{xx00} & = \text{Non Vital Irrelevant Move.} \\
\text{xx10} & = \text{Non Vital Unnecessary Move.} \\
\text{xxx1} & = \text{Vital Move (Not to be changed).}
\end{align*}
\]
The signal to change the look up table was carried in the most significant bit so that an example would be \texttt{1xxx xx00}. This would instruct the controller to change to the look up table for irrelevant moves. The code is shown below:

\begin{verbatim}
IF Flag%(m) AND 1 = 0
    IF TransMarker%(m-1) = TransMarker%(m) THEN
        Flag%(m) = Flag%(m) + 1
    END IF
END IF
\end{verbatim}

7.4 \textbf{Path Improvement: To Include Force Sensing.}

Problems occurred with the adaptation method described in the previous section because similar transients were experienced when a motor was overloaded or when a joint met an obstruction and was forced to stop or slow down. In practice, in the later work the collisions occurred when the path was revised and the Forearm contacted with the work surface.

Methods of discrimination were investigated.

Forces exerted in cartesian space could be related to forces in the joint variables by a Jacobean matrix. The calculation of this matrix is described by Orin \& Schrader(1974).

\[ |F| = [T+Q] \cdot [J]^{-1} \]

where:-

\begin{align*}
F_x & = \text{vector of cartesian forces,} \\
T & = \text{vector of joint variables,} \\
Q & = \text{vector of external forces.}
\end{align*}
These joint forces could be used to detect collisions by monitoring the joint motor currents. Motor current waveforms during contact with hard obstructions were found to have a larger Amplitude than transients associated with changes in direction. This is shown in figure 7.4. These collisions were detectable. Contact with softer objects was more difficult to discriminate.

In addition to using the signals to the mixer from the tacho-generator, the system was modified to also calculate the velocity in the software from the changes in position and the change in time. This velocity was used to consider suspected collisions by comparison with the error demand value, $e_d$. Velocities were monitored and a large error with a low velocity suggested an overload or collision.

The controller informed the overseer in the main computer when a demand signal was generated via the serial interface. In this revised system any transients not associated with the generation of new demand signals were regarded as collisions by the overseer.

This work and work described by Nagdy & Wu(1987) and Sanders et al(1987(a)) and (1987(b)) has shown that it was possible to use software calculation with information from joint motor currents for force sensing. The motor current varied as the square of motor torque and manipulator forces were transmitted to the joints as the motors tried to overcome these forces. This generated transients and current peaks appeared on the current waveforms. The torques could be detected by monitoring the current.
Figure 7.5  The Difference between a transient due to a change in direction (a) and a transient due to a collision (b).
Collision Detection. A collision was notified from three levels.

(a) - In the Peak Detector within the Main Computer.
(b) - In the Strategic level of the Controller.
(c) - In the Overseer within the Main Computer.

(a) - The Peak Detector. The amplitude of the current was monitored and compared to preset limits. Transients exceeding these limits were regarded as collisions and an instruction to stop was passed to the controller.

(b) - The Strategic level. The error demand value was compared with changes in absolute joint position. A small change in position associated with a large error demand was assumed to be a collision.

(c) - The Overseer. Unexpected transients received by the peak detector not associated with a marker from the supervisor in the controller were regarded as collisions.

Current peaks were detected by considering the level and gradient of consecutive samples following a new destination marker signal. Actuator moves were signalled to the Overseer by the supervisory level of the controller as new manipulator destinations were generated. This allowed the detection of unusual current peaks. Transients not associated with a new destination signal were assumed as collisions.

The velocity of the joint was calculated in the controller from the monitored absolute positions. Any velocities approaching zero were compared to the
demand error signal.

The sampled data were analyzed and information on adaptable trajectories was considered in the main computer. The joint trajectories were adapted by changing the controller look-up table in the robot controller.

**Revised Software: The Controller.** The programs worked as described in section 7.3, except that

(i) The velocity was calculated and compared to the demand signal.

(ii) A sub routine was included to stop the robot if a collision was detected.

After each position reading was taken, the timer was interrogated and providing the timer had not changed in excess of a preset limit, the velocity was calculated. If three consecutive velocity readings were low and a large error demand existed, a collision was assumed and the manipulator was stopped. The code is shown below:

```plaintext
TempTime = TIMER - OldTime
IF TempTime < TimeLimit%
    Vel = (NewPos - OldPos) / TempTime
    IF Vel < VelLimit% AND Error > ErrLimit% THEN
        ZeroJoints()
    END IF
END IF
```

**Revised Software: The Main Computer:** The software was as described earlier except that if the current was consistently greater than a set level, $|V_{Level}|$, then a collision was detected. A collision counter was reset to zero following a reading less than $|V_{Level}|$ and was incremented as readings exceeded the level.
If, \( i_n > |\nabla_{\text{Level}}| \), then, \( \text{count}_n = \text{count}_n + 1 \).

If, \( \text{count}_n = R_{\text{totalcount}} \), then, Remove Power from the joints.

Where,

- \( i_n \) = Reading of the motor current for sample \( n \).
- \( \text{count}_n \) = Collision counter.
- \(|\nabla_{\text{Level}}| \) = Set level.
- \( R_{\text{totalcount}} \) = Set number of readings before detection was assumed.

The level was monitored and once exceeded, an interrupt signal was transmitted to the supervisory level of the controller via the serial interface and the robot was stopped. The code was as shown below.

```plaintext
IF Current > AmpsLevel% THEN
    CountLevel% = CountLevel% + 1
    IF CountLevel% = StopNo% THEN
        ZeroJoints()
    END IF
END IF
```

7.5 Results.

The path shown in figure 7.6 was input to the Main Computer and passed to the Controller. The time taken to complete the path was initially 7.2 seconds. This was reduced to 6.2 seconds, a saving of 15%. The method worked efficiently for this example path and for all other paths without obstacles and with obvious, unnecessary and irrelevant via-points.

An attempt was made to introduce dynamic obstacles into the path adaption algorithm. The processing had to interact with the path planning procedures and the software became complex and slow. The work was
conducted in 2-D on the initial test rig (the prototype robot base and a simulated joint). A working system was not achieved which could include dynamic obstacles. If more time had been available, it is unlikely that such a system could have worked in real time with the processing power available for this research.

![Table](image)

**Figure 7.6: An Example Robot Path.**

In order to test the collision detection work, collisions were simulated in the initial test rig using the prototype robot base. The base was forced to stop by jamming the large and very strong gear train shown in figure 3.3 during a move. This induced large torques in the motors and the force detection methods worked satisfactorily. Collisions were detected and differentiated from changes in direction. When the level detection algorithm was used with the method to compare velocity with demand signal, the level detection algorithm tended to detect a collision first, but occasionally collisions were detected when none occurred due to noise in the motor current waveform.
7.6 Discussion and Conclusions.

The system demonstrated that by processing information from the currents to a motor, robot trajectories and paths could be adapted during a repeated series of moves in order to minimise current and torque peaks and thereby reduce the accelerations in the system.

Unnecessary changes in direction of the robot joints in attempting to closely follow programmed paths produced by CADCAM or teaching pendant reduce the operating speed and efficiency of the robot and may excite resonances in the manipulator. Using information from the currents to the dc motors, these unnecessary changes in direction were removed from the path and the trajectory was adapted.

The joint actuator was an electro-mechanical unit and wear was important. Minimising the current and torque transients reduced the mechanical forces and stresses in the equipment.

The path adaption algorithms relied on information from the hardware to adapt the path. When the paths were complex and inefficient, improvement was realised. When the path was planned by the automatic path planning systems described in chapter six, no detectable improvement was achieved.

The method did not consider obstacles and when an attempt was made to introduce obstacles into the system, the processing became excessively complex. A working system was never achieved with the Mitsubishi robot and it is unlikely that a real time system could have been achieved with the processing power available. This is discussed further in chapter nine.
The work described in this chapter did consider discrimination between transients due to collision and change in direction. This work was successful, but the system occasionally detected collisions when none occurred. The method could be improved by only signalling a collision when both the current level has increased above a preset limit and there is a low velocity with a large error. The methods of discrimination were presented in Sanders et al(1987(b)).

The method adapted given trajectories for the prototype base and elbow joint of the Mitsubishi robot. A detailed description of the method and the initial results was presented by the author in Sanders et al(1987(a)). Although the paths tended to be faster, the adapted paths did not consider the obstacle constraints. Although the new path was not necessarily an improvement in terms of speed or distance travelled by an end effector, the revised robot path tended not to expend as much energy as accelerations were reduced or removed from the trajectory.

The method did encompass the idea that a robot could be automatically made to complete a task in a way more suited to itself rather than in a way which appears suitable to a human operator.
8.1 Introduction.

Chapters three to six have described the creation of an automatic path planner in order to increase productivity. Chapter seven presented a method to fulfil a further aim of the work:-

To produce systems which would improve the performance of robots for which paths had already been planned by some means, automatic or otherwise.

This chapter presents a second method to fulfil this aim.

A robot is a physical system and is subject to physical limitations. By considering these limitations the robot performance can be improved with reference to some criterion and refined paths calculated for the robot. As the robot task to be improved has been assumed as a repetitive series of movements, the reprogrammer can take some time in calculating the improved paths while the robot carries out its original program, only modifying the path when the set of destinations was repeated.

The method of path improvement presented in this chapter used a simple model of the robot dynamics to improve a given task.

Models of the dynamics for active mechanisms are complex and many...
procedures for generating models have been devised; some are described by Brady et al(1982) and a dynamics model for a manipulator carrying loads was derived by Izaguirre & Paul(1985).

Two major approaches in terms of the formulation of robot dynamics equations are the Newton-Euler method and the Lagrangian formulation. The Newton-Euler method solves the problem recursively to find joint torques one by one whereas the Lagrangian method solves the problem using closed-form differential equations.

An, Atkeson & Hollerbach(1986) employed the Newton-Euler formulation to determine the inertial parameters of robot links. These could then be used in the recursive dynamics computation described in Fu et al(1987). Neumann & Khosla(1985) adopted a hybrid procedure combining the Newton-Euler and Lagrange formulation of the dynamics to estimate the inertial parameters of the links. The Lagrangian formulation was first developed to compute closed-form manipulator dynamics by Uicker(1966) and later Kahn(1969).

Even though many of the theoretical problems in manipulator dynamics have been solved, the question of how to best apply the theories to robot manipulators is still being debated. In the work presented in this chapter, information on system dynamics was used to produce a set of simple rules for an automatic path improvement system.

The dynamics of the manipulator in closed form Lagrange equations were selected to represent the dynamics by a set of second-order coupled non-linear differential equations. The form of these equations was exploited in an attempt to establish a set of simple rules. An experimental procedure was applied to the Mitsubishi RM 501 robot described in chapter three. The measured quantities were the drive currents to the motors (which represented the torques) and the joint angular positions. This method was similar to the methods used by Kumar(1988) for a two link planar robot manipulator. The advantage of using this input-output form was that intermediate non-linearities (such as gear friction) and the motor characteristics were directly incorporated into the model. The results were unexpected and the model of the robot dynamics is discussed in section 8.6.

In the next section the Lagrange formulation for the Mitsubishi robot is outlined. In sections 8.3 and 8.4 the experimental identification procedure is described and in section 8.5 the results of this procedure are presented. Section 8.6 describes the simple rules developed from these results and section 8.7 presents the results of using these rules. The chapter concludes with discussion and conclusions in section 8.8.
8.2 The Dynamic Model: The Lagrangian Formulation for the Mitsubishi RM 501 Robot.

The formulation was based on the Lagrangian equation in terms of the Lagrangian coordinates \( q \) given by:

\[
\tau_i = \frac{d}{dt} \left( \frac{\partial L}{\partial (dq_i/dt)} \right) - \frac{\partial L}{\partial q_i}
\]

where,

\[ L = \text{The Lagrangian function.} \]
\[ q_i = \text{The coordinate of the ith element used to express the kinetic and potential energies.} \]
\[ \tau_i = \text{The torque.} \]

The relationships between the torques and the angular positions, velocities and accelerations of the links were obtained by considering the potential and kinetic energies. The Lagrangian \( L \) is defined as the difference between the kinetic and potential energy given by:

\[
L = K - P
\]

where:

\( K \) is the total kinetic energy.
\( P \) is the total potential energy.

In this chapter, using the expressions for \( K \) and \( P \) in terms of manipulator parameters, the equations for the dynamics of the three main links of the Mitsubishi robot were obtained in the form:

\[
\tau_i = \sum_{j=1}^{N} J_{ij} \frac{d^2 \theta_j}{dt^2} + \sum_{j=1}^{N} \sum_{k=1}^{N} H_{ijk} \left( \frac{d \theta_j}{dt} \right) \left( \frac{d \theta_k}{dt} \right) + G_i
\]
The Mitsubishi robot was assumed to consist of two main movable links; \( L_1 \) and \( L_2 \) of masses \( m_1 \) and \( m_2 \) which could be rotated through angles \( \theta_2 \) and \( \theta_3 \), as shown in figure 8.1. The robot base \( L_0 \), with mass \( m_0 \) could rotate through \( \theta_1 \). To determine the total kinetic and potential energy for the robot, each link was considered in turn.

![Diagram of the Mitsubishi robot](image)

**Figure 8.1:** The Model used for the Three Main Links and Masses of the Mitsubishi Robot.

The kinetic energy and potential energy equations of link \( L_0 \) were assumed to be:

\[
K_0 = I(d\theta_1/dt)^2
\]

\[
P_0 = 0
\]

where \( I \) is the moment of inertia of link \( L_0 \) about the Z axis.
Considering link $L_1$, expressing the cartesian coordinates of the assumed centre of mass shown in figure 8.1 in terms of the joint angles gave:

\[
\begin{align*}
X_1 &= L_1/2 \cos \theta_1 \cos \theta_2 \\
Y_1 &= L_1/2 \sin \theta_1 \cos \theta_2 \\
Z_1 &= L_0 + L_1/2 \sin \theta_2
\end{align*}
\]

Taking derivatives of the equations with respect to time gave:

\[
\begin{align*}
\frac{dX_1}{dt} &= -\frac{L_1}{2} \frac{d\theta_1}{dt} \sin \theta_1 \cos \theta_2 - \frac{L_1}{2} \frac{d\theta_2}{dt} \cos \theta_1 \sin \theta_2 \\
\frac{dY_1}{dt} &= \frac{L_1}{2} \frac{d\theta_1}{dt} \cos \theta_1 \cos \theta_2 - \frac{L_1}{2} \frac{d\theta_2}{dt} \sin \theta_1 \cos \theta_2 \\
\frac{dX_1}{dt} &= \frac{L_1}{2} \frac{d\theta_2}{dt} \cos \theta_2
\end{align*}
\]

Considering $V_1^2$ where

\[
V_1^2 = (\frac{dX_1}{dt})^2 + (\frac{dY_1}{dt})^2 (\frac{dZ_1}{dt})^2
\]

Using trigonometric identities to reduce the solution, the square of the velocity vector was:

\[
V_1^2 = (\frac{L_1}{2})^2 (\frac{d \theta_2}{dt})^2 + (\frac{L_1}{2})^2 (\frac{d \theta_1}{dt})^2 \cos^2 \theta_2
\]

The kinetic energy term and the potential energy term of link $L_1$ were thus assumed to be:

\[
\begin{align*}
K_1 &= \frac{1}{2} m_1 V_1^2 \\
&= \frac{1}{2} m_1 (L_1/2)^2 \left( (\frac{d \theta_2}{dt})^2 + (\frac{d \theta_1}{dt})^2 \cos^2 \theta_2 \right) \\
P_1 &= m_1 g L_0 + m_1 g (L_1/2) \sin \theta_2
\end{align*}
\]

where $g = \text{gravitational acceleration}$. 

207
The cartesian coordinates of the centre of mass of link $L_2$ were assumed to be:

$$X_2 = L_1 \cos \theta_1 \cos \theta_2 + \frac{L_2}{2} \cos \theta_1 \cos(\theta_2 + \theta_3 - \pi)$$
$$Y_2 = L_1 \sin \theta_1 \cos \theta_2 + \frac{L_2}{2} \sin \theta_1 \cos(\theta_2 + \theta_3 - \pi)$$
$$Z_2 = L_0 + L_1 \sin \theta_2 + \frac{L_2}{2} \sin(\theta_2 + \theta_3 - \pi)$$

Taking derivatives of the equations with respect to time gave:

$$\frac{dX_2}{dt} = -\frac{d\theta_1}{dt} \left( L_1 \sin \theta_1 \cos \theta_2 + \frac{L_2}{2} \sin \theta_1 \cos(\theta_2 + \theta_3 - \pi) \right)$$
$$- \frac{d\theta_2}{dt} \left( L_1 \cos \theta_1 \sin \theta_2 + \frac{L_2}{2} \cos \theta_1 \sin(\theta_2 + \theta_3 - \pi) \right)$$
$$- \frac{d\theta_3}{dt} \left( \frac{L_2}{2} \cos \theta_1 \sin(\theta_2 + \theta_3 - \pi) \right)$$

$$\frac{dY_2}{dt} = \frac{d\theta_1}{dt} \left( L_1 \cos \theta_1 \cos \theta_2 + \frac{L_2}{2} \cos \theta_1 \cos(\theta_2 + \theta_3 - \pi) \right)$$
$$- \frac{d\theta_2}{dt} \left( L_1 \sin \theta_1 \sin \theta_2 + \frac{L_2}{2} \sin \theta_1 \sin(\theta_2 + \theta_3 - \pi) \right)$$
$$- \frac{d\theta_3}{dt} \left( \frac{L_2}{2} \sin \theta_1 \sin(\theta_2 + \theta_3 - \pi) \right)$$

$$\frac{dX_2}{dt} = \frac{d\theta_2}{dt} \left( L_1 \cos \theta_2 + \frac{L_2}{2} \cos(\theta_2 + \theta_3 - \pi) \right)$$
$$+ \frac{d\theta_3}{dt} \left( \frac{L_2}{2} \cos(\theta_2 + \theta_3 - \pi) \right)$$

So that after reducing the solution using trigonometric identities, the expression for the square of the velocity vector was:

$$V_2^2 = (\frac{dX_2}{dt})^2 + (\frac{dY_2}{dt})^2 + (\frac{dZ_2}{dt})^2$$

$$= (L_1^2 + L_2^2/4)(\frac{d\theta_2}{dt})^2 + L_1 L_2 (\frac{d\theta_2}{dt})^2 \cos(\theta_2 + \theta_3 - \pi)$$
$$+ \frac{L_2^2}{4} (\frac{d\theta_3}{dt})^2 + \frac{L_2^2}{4} (\frac{d\theta_3}{dt})(\frac{d\theta_3}{dt}) \sin^2(\theta_2 + \theta_3 - \pi) +$$
$$\frac{d\theta_1}{dt}^2 \left( \frac{L_2^2}{4} \cos^2(\theta_2 + \theta_3 - \pi) \right) + L_1^2 \cos^2 \theta_2 + L_1 L_2 \cos \theta_2 \cos(\theta_2 + \theta_3 - \pi)$$

208
and the kinetic energy and potential energy terms are therefore given by:

\[
K_2 = \frac{m_2}{2}(L_1^2 + L_2^2/4)(d\theta_2/dt)^2 + \frac{m_2}{2}L_1L_2(d\theta_2/dt)^2\cos(\theta_2 + \theta_3 - \pi) \\
+ \frac{m_2L_2^2}{8}(d\theta_3/dt)^2 + (m_2L_2^2/8)(d\theta_2/dt)(d\theta_3/dt)\sin^2(\theta_2 + \theta_3 - \pi) \\
+ (d\theta_1/dt)^2(m_2/2)(L_2^2/4)\cos^2(\theta_2 + \theta_3 - \pi) + L_1^2\cos^2\theta_2 \\
+ L_1L_2\cos\theta_2\cos(\theta_2 + \theta_3 - \pi)
\]

\[
P_2 = m_2gL_0 + m_2gL_1\sin\theta_2 + m_2g(L_2/2)\cos(\theta_2 + \theta_3 - \pi)
\]

Having found the kinetic and potential energies for the three joints, the Lagrangian of the robot;

\[
L = K_0 + K_1 + K_2 - (P_0 + P_1 + P_2)
\]

was calculated so that:

\[
L = I(d\theta_1/dt)^2/2 + 1/2 m_1(L_1/2)^2 \{(d\theta_2/dt)^2 + (d\theta_1/dt)^2\cos^2\theta_2\} \\
+ (m_2/2)(L_1^2 + L_2^2/4)(d\theta_2/dt)^2 + (m_2/2)L_1L_2(d\theta_2/dt)^2\cos(\theta_2 + \theta_3 - \pi) \\
+ m_2L_2^2/8(d\theta_3/dt)^2 + (m_2L_2^2/8)(d\theta_2/dt)(d\theta_3/dt)\sin^2(\theta_2 + \theta_3 - \pi) \\
+ (d\theta_1/dt)^2m_2/2(L_2^2/4)\cos^2(\theta_2 + \theta_3 - \pi) + L_1^2\cos^2\theta_2 + L_1L_2\cos\theta_2\cos(\theta_2 + \theta_3 - \pi) \\
- m_1gL_0 - m_1g(L_1/2)\sin\theta_2 - m_2gL_0 - m_2gL_1\sin\theta_2 - m_2g(L_2/2)\cos(\theta_2 + \theta_3 - \pi)
\]

The following six derivatives were then found, \(\partial L/\partial \theta_1\), \(\partial L/\partial \theta_2\), \(\partial L/\partial \theta_3\), \(\partial L/\partial (d\theta_1/dt)\), \(\partial L/\partial (d\theta_2/dt)\) and \(\partial L/\partial (d\theta_3/dt)\) so that the Lagrangian equation in terms of the robot joints;

\[
\tau_i = \frac{d}{dt} \frac{\partial L}{\partial (d\theta_i/dt)} - \frac{\partial L}{\partial \theta_i}
\]
could be applied for each of the links $\theta_1$, $\theta_2$ and $\theta_3$ in turn.

The first dynamics equation was thus:

$$\tau_1 = (d^2\theta_1/dt^2)I + m_1(L_1/2)^2\sin^2\theta_2 + m_2(L_1/2)\sin\theta_2 + m_2(L_2/2)\sin\theta_3,$$

$$+ \frac{d\theta_1}{dt} \frac{d\theta_2}{dt} \cdot 2m_1(L_1/2)\cos\theta_2 - m_2L_1^2\cos\theta_2\sin\theta_2$$

$$+ m_2L_1(L_2/2)\cos\theta_2\cos\theta_3,$$

$$+ \frac{d\theta_1}{dt} \frac{d\theta_3}{dt} \cdot 2m_2L_2^2\cos\theta_2\cos\theta_3 + m_2L_1(L_2/2)\sin\theta_2\sin\theta_3,$$

This equation and the other torque equations had several components. They were:

- Effective inertias (and coupling inertias).
- Coriolis and centripetal coefficients.
- Gravity loadings.

so the equation for $\tau_1$ could be expressed in the form:

$$\tau_1 = D_{11} \frac{d^2\theta_1}{dt^2} + D_{12} \frac{d\theta_1}{dt} \frac{d\theta_2}{dt} + D_{13} \frac{d\theta_1}{dt} \frac{d\theta_3}{dt} + D_{1g}$$

where:

- $D_{11}$ = The effective moment of inertia about the Z1 axis
- $D_{12} \frac{d\theta_1}{dt} \frac{d\theta_2}{dt}$ = The coriolis torque acting at joint $\theta_1$ due to the velocities of the base $\theta_1$ and shoulder $\theta_2$.
- $D_{13} \frac{d\theta_1}{dt} \frac{d\theta_3}{dt}$ = The coriolis torque acting at joint $\theta_1$ due to the velocities of the base $\theta_1$ and the elbow $\theta_3$.
- $D_{1g}$ = The gravitational torque.
The second dynamic equation was:

\[ \tau_2 = \frac{d^2\theta_2}{dt^2} \left( m_1(L_1/2)^2 + m_2L_2^2 \right) + \frac{d^3\theta_3}{dt^2} \left( m_2L_2(L_2/2)\sin(\theta_2 + \theta_3 - \pi) \right) \\
+ \frac{d\theta_2}{dt} \frac{d\theta_3}{dt} 2m_2L_1(L_2/2)\cos(\theta_2 + \theta_3 - \pi) \\
- (\frac{d\theta_3}{dt})^2 2m_2L_1(L_2/2)\cos(\theta_2 + \theta_3 - \pi) \\
- (\frac{d\theta_1}{dt})^2 \left( m_1(L_1/2)^2 \cos\theta_2 \sin\theta_2 + m_2L_2^2 \cos\theta_2 \sin\theta_2 \right. \\
\left. + m_2L_2(L_2/2)\cos\theta_2 \cos\theta_3 \right) \\
- m_1g(L_1/2)\cos\theta_2 - m_2gL_1\cos\theta_2 - m_2g(L_2/2)\cos(\theta_2 + \theta_3) \]

where \( \tau_2 \) was the torque applied to \( \theta_2 \). This equation in coefficient form was:

\[ \tau_2 = D_{21} \frac{d^2\theta_2}{dt^2} + D_{22} \frac{d\theta_2}{dt} \frac{d\theta_3}{dt} + D_{2c1} \frac{d^2\theta_2}{dt^2} \\
+ D_{24} (\frac{d\theta_3}{dt})^2 + D_{25} (\frac{d\theta_1}{dt})^2 + D_{26} \]

where

- \( D_{21} \) = The effective moment of inertia about the \( Z_2 \) axis
- \( D_{22} \frac{d\theta_2}{dt} \frac{d\theta_3}{dt} \) = Coriolis torque due to velocities of the shoulder and elbow.
- \( D_{2c1} \) = Coupling inertia term between links \( L_1 \) and \( L_2 \).
- \( D_{24} (\frac{d\theta_3}{dt})^2 \) = Centripetal torque at \( \theta_2 \) due to the velocity of \( \theta_3 \).
- \( D_{25} (\frac{d\theta_1}{dt})^2 \) = Centripetal torque at \( \theta_2 \) due to the velocity of \( \theta_1 \).
- \( D_{2g} \) = The gravitational torque.

The third dynamics equation was:

\[ \tau_3 = \frac{d^2\theta_3}{dt^2} m_3(L_3/2)^2 + \frac{d^2\theta_2}{dt^2} [m_3L_1(L_2/2)\sin(\theta_2 + \theta_3 - \pi)] \\
+ (\frac{d\theta_1}{dt})^2 [m_4L_1(L_2/2)^2 \sin\theta_3 + m_4L_1(L_2/2)\sin\theta_2 \sin\theta_3] \\
+ (\frac{d\theta_2}{dt})^2 [m_2L_1(L_2/2)\cos(\theta_2 + \theta_3 - \pi)] - m_2g(L_2/2)\cos(\theta_2 + \theta_3 - \pi) \]
and in the coefficient form,

\[ \tau_3 = D_{31}d^2\theta_3/dt^2 + D_{3c1}d^2\theta_2/dt^2 + D_{33}(d\theta_1/dt)^2 + D_{34}(d\theta_2/dt)^2 + D_{3g} \]

where

- \( D_{31} \) = The effective inertia term at joint 3.
- \( D_{3c1} \) = The coupling inertia term between links \( L_1 \) and \( L_2 \).
- \( D_{33}(d\theta_1/dt)^2 \) = Centripetal torque acting at \( \theta_3 \) due to velocity \( d\theta_1/dt \).
- \( D_{34}(d\theta_2/dt)^2 \) = Centripetal torque acting at \( \theta_3 \) due to velocity \( d\theta_2/dt \).
- \( D_{3g} \) = The gravitational torque.

The expressions for the dynamics derived in this section consisted of variables, which were functions of sines and cosines of the joint positions and constants which depended on the manipulator link parameters such as link mass, centre of mass, and radii of gyration. Measurements could have been taken of the links to obtain the dimensions of centres of mass and radius of gyration for each link. The link masses could have been calculated from the measurements and the density of the materials and then the dynamics constants calculated.

Although values might have been calculated from measurements and drawings, the process would have been tedious. Measurement of parameters such as location of centre of masses and exact shapes would have been susceptible to errors. An alternative approach used in the work described in this dissertation was to obtain the constants by actually running the manipulator. The approach used direct input-output measurements during actual motion and then used the results presented in section 8.4 to produce simple rules for robot path improvement.

In the next section the experimental method is discussed.
8.3 The Dynamic Model: The Formulation of the Experiments.

Bejczy (1974) first noticed the disparity of the roles that different dynamics terms play in the dynamics equations and Paul (1981) and Paul et al (1983) extended the idea to the elimination of the insignificant dynamics terms and expressions within terms when using the equations for manipulator control. The importance of the velocity dependent terms has been controversial and Brady et al (1982) demonstrated that there are situations where centripetal and Coriolis forces dominate the inertial forces. The manipulator joints experience high velocities during gross motions when the controller accuracy is not critical. During fine motions when the control accuracy is important, joints move with high accelerations and low velocities so that the gravitational and inertial forces become dominant and velocity dependent forces are not so important.

As the work described in this dissertation was concerned with the gross motions associated with path planning and not the fine motions associated with approach paths, the inertial terms were assumed to be less significant.

The inertial and coupling inertia terms were excluded to give the following simplified equations:

\[
\begin{align*}
\tau_1 &= \frac{d\theta_1}{dt} \frac{d\theta_2}{dt} 2lm_1(L_1/2)\cos\theta_2 - m_2L_1^2\cos\theta_2\sin\theta_2 \\
&+ m_2L_1(L_2/2)\cos\theta_2\sin\theta_3 \\
&+ \frac{d\theta_1}{dt} \frac{d\theta_3}{dt} 2m_2(L_2/2)^2\cos\theta_2\cos\theta_3 + m_2L_1(L_2/2)\sin\theta_2\sin\theta_3 \\
\end{align*}
\]

\[
\begin{align*}
\tau_2 &= \frac{d\theta_2}{dt} \frac{d\theta_3}{dt} 2m_2L_1(L_2/2)\cos(\theta_2 + \theta_3) \\
&- (d\theta_3/dt)^2 2lm_1(L_2/2)\cos(\theta_2 + \theta_3 - \pi) \\
&- (d\theta_1/dt)^2 2m_1(L_1/2)^2\cos\theta_2\sin\theta_2 + m_2L_1^2\cos\theta_2\sin\theta_2 \\
&+ m_2L_2(L_2/2)\cos\theta_2\cos\theta_3 \\
&- m_1g(L_1/2)\cos\theta_2 - m_2gL_1\cos\theta_2 - m_2g(L_2/2)\cos(\theta_2 + \theta_3) \\
\end{align*}
\]

\[
\begin{align*}
\tau_3 &= (d\theta_1/dt)^2lm_3(L_2/2)^2\sin\theta_3 + m_3L_1(L_2/2)\sin\theta_2\sin\theta_3 \\
&+ (d\theta_2/dt)^2m_3L_1(L_2/2)\cos(\theta_2 + \theta_3 - \pi) - m_2g(L_2/2)\cos(\theta_2 + \theta_3 - \pi) \\
\end{align*}
\]
so that:—

\[ D_{12} = 2\left(m_1(L/2)\cos\theta_2 - m_2L_1^2\cos\theta_2\sin\theta_2 + m_2L_2(L/2)\cos\theta_2\cos\theta_3\right) \]

\[ D_{13} = 2\left(m_2L_2/2)^2\cos\theta_2\cos\theta_3 + m_2L_1(L/2)\sin\theta_2\sin\theta_3\right) \]

\[ D_{1g} = 0 \]

\[ D_{22} = 2m_2L_1(L/2)\cos(\theta_2 + \theta_3 - \pi) \]

\[ D_{24} = 2m_2L_1(L/2)\cos(\theta_2 + \theta_3 - \pi) \]

\[ D_{25} = m_1(L/2)^2\sin\theta_2 + m_2L_1^2\cos\theta_2\sin\theta_2 + m_2L_2(L/2)\cos\theta_2\cos\theta_3 \]

\[ D_{2g} = m_1L_1L_2\cos\theta_2 + m_2gL_1\cos\theta_2 + m_2g(L/2)\cos(\theta_2 + \theta_3) \]

\[ D_{33} = m_3(L/2)^2\sin\theta_3 + m_3L_1(L/2)\sin\theta_2\sin\theta_3 \]

\[ D_{34} = m_3L_1(L/2)\cos(\theta_2 + \theta_3 - \pi) \]

\[ D_{3g} = m_2gL_2(L/2)\cos(\theta_2 + \theta_3 - \pi) \]

To determine the dynamics constants experimentally, it was important to know the joint torques of all the joints at any time instant. This was achieved using the method described in section 7.2 to monitor the joint motor currents. As the manipulator joints were actuated by electric motors, joint motor currents provided a measurement of the torque being exerted by the joints. Figure 8.2 shows a typical relationship between a joint motor current and joint output torque and is reproduced from results presented by Hong(1986).

![Figure 8.2: A Sketch of Torque verses Motor Current.](image)
The output torque was approximately linear to the motor current except for an offset at the origin and a diverging curvature on both curves, which corresponded to the two directions of motion. The offset at the origin was caused by static friction that the joint must overcome before any motion at the joint could result. The diverging characteristic is explained by the load dependent nature of joint friction, which increases non linearly with an increase in load. In this work the functional relationship between joint torque and current was assumed to be a linear relationship so that the process of computing torque from current was a simple linear mapping and in practice the torque constants provided by the manufacturer were used in converting currents to torques.

**Summary:** The position and velocity were measured for various inputs. The joint torques necessary to generate motion were observed while the manipulator moved along trajectories with known motion parameters. Since the joint torque was directly related to the constants by the dynamics equations and the intermediate joint positions were known, a set of equations linear to the constants could be established from the readings of joint current and joint position and used to solve for the constants in the equations of the dynamics. This method took the non linearity of the manipulator into account and the method is described in the next section.
8.4 The Dynamic Model: The Experimental Method.

The procedures described in the previous section were initially applied to the prototype robot base shown in Appendix D and then to the base, shoulder and elbow joint of the Mitsubishi RM 501 robot with an end effector load of 2 lbs. The 80286 micro-computer controller provided torque commands to the motors through 8-bit D to A converters. The angular positions of the joints were fed back to the computer from optical encoders mounted on motor shafts. The encoder outputs were converted to a count representing position and were read by the computer via the G64 bus. Software for the system was developed in Desmet-C and then Quick-Basic. The motors were current controlled.

A series of three tests were conducted:-

(i) Static Tests.
(ii) Single Joint Motion Tests.
(iii) Multiple Joint Motion Tests.

(i) Static Tests: To obtain the gravitational constants from the knowledge of joint torques, the effects due to other dynamics terms were eliminated so that the joint torque became a function of gravity loading. Only the joint of interest was moved and the other joints were stationary. Under these test conditions, the velocity and acceleration dependent terms disappeared.

With the other joints locked in a particular configuration, the torque or force required to move each joint was measured. The gravitational torques were estimated by moving the manipulator to a desired configuration and then incrementing the output through the D/A converter 1 bit at a time until
motion was detected. The result of these measurements was a table of gravitational torques (D_{ig} for link i) for varying θ₁, θ₂ and θ₃.

If \( \tau_{pi} \) was the torque in one direction and \( \tau_{mi} \) in the other, and \( F_{is} \) represented static friction for joint i, the following equations were obtained:

\[
\tau_{pi} = D_{ig} + F_{is}
\]
\[
\tau_{mi} = -D_{ig} + F_{is}
\]
so that:

\[
D_{ig} = (\tau_{pi} + \tau_{mi})/2
\]

This procedure was repeated for each ten degree increment of each joint angle that occurred as a basis function for \( D_{ig} \). Two constants, \( A \) and \( B \) were to be determined to satisfy:

\[
A = m_2 g L_2 / 2
\]
\[
B = g L_1 (m_2 + m_1 / 2)
\]
so that:

\[
D_{3g} = A \cos (\theta_2 + \theta_3 - \pi) = -A \cos (\theta_2 + \theta_3)
\]
\[
D_{2g} = B \cos (\theta_2) - D_{3g}
\]
\[
D_{1g} = 0
\]

The results obtained were unexpected and are shown in figures 8.3 to 8.12 and figures 8.13 to 8.22. The results are discussed in section 8.6.a.
(ii) **single Joint Motion Tests**: This was achieved by driving the motors at a constant velocity. Practically, this was achieved by outputting a step velocity demand and running the joints through 10 degrees before taking any readings to avoid the inertial effects. Only one joint was moved at a time so that the governing equation was:

\[
\tau_i = b_i \left( \frac{d\theta_i}{dt} \right) + F_i + D_{ig}
\]

With gravitational compensation this could be reduced to:

\[
\tau_i = b_i \left( \frac{d\theta_i}{dt} \right) + F_i
\]

where

- \( F_i \) is the Coulomb friction
- \( b_i \) is the overall viscous damping coefficient.

so that the steady-state velocity was:

\[
\left( \frac{d\theta_i}{dt} \right)_{ss} = \frac{\tau_i - F_i}{b_i}
\]

The current required to maintain a constant velocity, and the velocity of the base joint for a constant demand output, were recorded for various configurations. Again the results were surprising and are discussed in section 8.6 and shown in figures 8.23 to 8.27.
(iii) **Multiple Joint Motion Tests:** To estimate the coupling terms in the dynamic equations, motions requiring joints to move simultaneously were applied. The same input was applied to joint i, first with joint j stationary and then with joint j also in motion. The response in the two cases with gravitational compensation was assumed as:

With coupling
\[
\tau_{ic} = H_{ij}(d\theta_i/c/dt)(d\theta_j/c/dt) + b_i(d\theta_i/c/dt) + F_i
\]

Without coupling
\[
\tau_i = b_i(d\theta_i/dt) + F_i
\]

so that
\[
H_{ij}\dot{\theta}_i\dot{\theta}_j = \tau_{ic} - \tau_i
\]

where the subscript c indicated the presence of coupling.

The measured motion responses together with previously computed values of \(b_i\) and \(F_i\) were to be used to evaluate the coupling coefficients in the above equations.

In the event, this evaluation was not necessary.

8.5 **The Dynamic Model: Results.**

The graphical results from the static and motion tests are presented in this section.
(i) **Static Tests:** The initial series of ten graphs show the shoulder current required to overcome gravity and the static friction of the shoulder joint for various configurations of the elbow joint.

![Shoulder Static Tests: Elbow = 90 deg](image)

**Figure 8.3:** Elbow Joint at 90 degrees.

![Shoulder Static Tests: Elbow = 100 deg](image)

**Figure 8.4:** Elbow Joint at 100 degrees.
Figure 8.5: Elbow Joint at 110 degrees.

Figure 8.6: Elbow Joint at 120 degrees.
Figure 8.7: Elbow Joint at 130 degrees.

Figure 8.8: Elbow Joint at 140 degrees.
Shoulder Static Tests: Elbow = 150 deg

Figure 8.9: Elbow Joint at 150 degrees.

Shoulder Static Tests: Elbow = 160 deg

Figure 8.10: Elbow Joint at 160 degrees.
Figure 8.11: Elbow Joint at 170 degrees.

Figure 8.12: Elbow Joint at 180 degrees.
Figures 8.3 to 8.12 show $\tau_{pi}$ (the torque in one direction) and $\tau_{mi}$ (the torque in the other direction). As discussed in section 8.4, $F_{is}$, the static friction for joint $i$ could be removed as: $\tau_{pi} = D_{ig} + F_{is}$
and $\tau_{mi} = -D_{ig} + F_{is}$ so that: $D_{ig} = (\tau_{pi} + \tau_{mi})/2$

The remaining $D_{2g}$ is shown in figures 8.13 to 8.22 with the Elbow angle marked underneath.

Figure 8.13: 90

Figure 8.14: 100

Figure 8.15: 110

Figure 8.16: 120
Gravity Effects: Elbow = 130 deg

Figure 8.17: 130

Gravity Effects: Elbow = 140 deg

Figure 8.18: 140

Gravity Effects: Elbow = 150 deg

Figure 8.19: 150

Gravity Effects: Elbow = 160 deg

Figure 8.20: 160

Gravity Effects: Elbow = 170 deg

Figure 8.21: 170

Gravity Effects: Elbow = 180 deg

Figure 8.22: 180
(ii) **Single Joint Motion Tests:** Figures 8.23, 8.24 and 8.25 show the current required to maintain a constant velocity for each joint for different configurations. Figure 8.25 contained unexpected results for the base joint and this is investigated further in figure 8.26 and 8.27.

**Figure 8.23:** The Current required to drive the Elbow at a constant velocity.

**Figure 8.24:** The Current required to drive the Shoulder at a constant velocity.

**Figure 8.25:** The Current required to drive the Base at a constant velocity.
Figures 8.26 and 8.27 show the Base joint velocity for different configurations of the Shoulder and Elbow.

**Figure 8.26:** The Base Joint Angular Velocity for varying Shoulder Configurations, with the Elbow static at 90 degrees.

**Figure 8.27:** The Base Joint Angular Velocity for varying Shoulder Configurations, with the Elbow static at 180 degrees.
(iii) **Multiple Joint Motion Tests:** The noise in the system was greater than any effects due to coupling between joints.

8.6 **Application of the Model: The Improvement Method.**

This section is in two parts:-

(a) A discussion of the results.

(b) The development of Simple Rules for Path Adaption

(a) **A discussion of the results.**

(i) **Static Tests:** The equations for the manipulator dynamics developed in section 8.4.i suggested that the maximum gravitational effect would be felt by joints $\theta_2$ and $\theta_3$ at $\theta_2 = 0^\circ$, $\theta_3 = 180^\circ$.

and the minimum effect at

$\theta_2 = 90^\circ$, $\theta_3 = 180^\circ$.

as the equations for the static case were expected to be

\[
\tau_2 = B \cos(\theta_2) + A \cos(\theta_2 + \theta_3) + F_{is}
\]

\[
\tau_3 = -A \cos(\theta_2 + \theta_3) - F_{is}
\]

The practical results in figures 8.3 to 8.22 show that the maximum effect was felt by the robot at

$\theta_2 = 40^\circ \& 55^\circ$, $\theta_3 = 180^\circ$.

and their were two minima, one of which was predicted at
\[ \theta_2 = 90^\circ, \quad \theta_3 = 180^\circ. \]

and a second at \( \theta_2 = 0^\circ, \quad \theta_3 = 110^\circ. \)

Detailed inspection of the robot revealed a spring included in the robot design as gravity compensation for the arm. From inspection of the static results, the spring effects could be roughly modelled by \( \cos \) of \( 2\theta_2 \) over the range \( 0^\circ \) to \( 45^\circ \), so that the equation for \( D_{2g} \) became approximately:

\[
D_{2g} = B \cos (\theta_2) + D_{3g} \cdot C \cos (2\theta_2) \quad \text{for} \quad \theta_2 < 45^\circ.
\]

\[
D_{2g} = B \cos (\theta_2) + D_{3g} \quad \text{for} \quad \theta_2 > 45^\circ.
\]

where \( C = B \).

(ii) **Single Joint Motion Tests:** Considering the equation from section 8.4.ii-

\[
(d\theta_i/dt)_{ss} = \frac{\tau_i - F_i}{b_i}
\]

joints \( \theta_2 \) and \( \theta_3 \) performed as expected as shown in figure 8.23 and 8.24, in that they were not affected by the configuration of the other joints. The base joint \( \theta_1 \) however, was affected by the configuration of \( \theta_2 \) and \( \theta_3 \). Figure 8.25 shows that the base joint had a steady state velocity which was dependent on joint angles \( \theta_2 \) and especially \( \theta_3 \).

It was expected that the velocity of \( \theta_3 \) would have been greater as the mass moved towards the Origin. The practical results show that this was not the case. In fact the opposite was true.
The inconsistency between the expected results and practical results for the base joint can be explained by considering the balancing of the robot arm and the large rear section of link $L_7$ which housed some of the motors. The large rear section can be seen in figure 8.28. This design meant that when the arm was extended horizontally the whole unit was balanced at the base joint, but with the arm vertical the rear section was pulled down by gravity causing increased friction within the base gearbox. This increase in friction resulted in a decrease in steady state velocity as shown in figures 8.26 and 8.27.

Figure 8.28: The Mitsubishi RM 501 Robot. (Showing the large rear section housing the motors).
(iii) **Multiple Joint Motion Tests**: Their were no measurable velocity effects due to coupling effects between the joints. Although results were not recorded their was an obvious inertia coupling between joints $\theta_2$ and $\theta_3$. This could be considered in future work.

(b) **The Development of Simple Rules for Path Improvement**.

Considering the results of the position and velocity tests, only three effects dominated the dynamics of the Mitsubishi robot. They were:

1. The varying effect of $\theta_2$ and $\theta_3$ on the friction of the base joint.
2. The balance spring connected to $L_1$.
3. The gravity effect of $\theta_3$ upon $\theta_2$.

These suggested two simple rules by which the robot path could be improved.

**RULE (i)** To reduce the base friction during movements of the base, the arm should attempt to balance the base mechanism by moving $\theta_2$ towards $0^\circ$ and $\theta_3$ towards $180^\circ$.

**RULE (ii)** To reduce the effects of gravity loading, the arm should move $\theta_3$ towards $90^\circ$ during motions of $\theta_2$.

Because rule (ii) has an effect on rule (i), rule (i) was given precedence over rule (ii).
8.7 Application of the Model: Results.

Once these rules had been established, motion tests were undertaken for various paths and the times for the revised paths were recorded. The tests were repeated with three different Mitsubishi RM 501 robots and typical results were:

(i) To test for the reduction in coulomb friction: The arm was initially moved from \([140^\circ, 0^\circ, 180^\circ]\) to \([-140^\circ, 0^\circ, 180^\circ]\) via \([0^\circ, 90^\circ, 180^\circ]\). The movement took an average of 4.44 seconds. When the test path was modified to use the same START and GOAL, but to move through a via-point at \([0^\circ, 0^\circ, 180^\circ]\) the robot took an average of 4.14 seconds. A saving of 0.3 seconds (=6.8%)

(ii) To test for the reduction in gravity loading: Similar tests were conducted for the shoulder and elbow, with the waist still (at \(0^\circ\)). The shoulder was moved from \(-10^\circ\) to \(90^\circ\) with the elbow at \(180^\circ\), this gave an average time of 1.94 seconds. When the path was modified so that the elbow moved in towards \(90^\circ\) until the shoulder reached \(50^\circ\) then moved out to \(180^\circ\), an average time of 1.74 seconds was recorded. A saving of 0.2 seconds (=10%)

The adaption rules were included in the automatic path planning and adaption system and the two sets of code are shown below:

```plaintext
ShoulderDiff = Shoulder(n%+1) - Shoulder(n%)
NewShoulder(n%) = Shoulder(n%) + ShoulderDiff/2

ElbowDiff = Elbow(n%+1) - Elbow(n%)

IF SGN(ElbowDiff) = 1 THEN
    NewElbow(n%) = Elbow(n%) - ShoulderDiff/6
ELSE
    NewElbow(n%) = Elbow(n%+1) - ShoulderDiff/6
END IF
```

233
BaseDiff = base(n%+1) - Base(n%)
NewBase(n%) = Base(n%) + BaseDiff/2
ShoulderDiff(n%) = Shoulder(n%+1) - Shoulder(n%)
ElbowDiff = ElbowDiff(n%) - ElbowDiff(n%+1)
IF BaseDiff < > 0 THEN
    IF (Shoulder(n%+1) > 0) AND SGN(ShoulderDiff) = 1 THEN
        NewShoulder(n%) = Shoulder(n%) - BaseDiff/2
        IF SGN(ElbowDiff) = 1 THEN
            NewElbow(n%) = Elbow(n%) + BaseDiff/4
        ELSE
            NewElbow(n%) = Elbow(n%+1) + BaseDiff/4
        END IF
    ELSE IF (Shoulder(n%+1) > 0) AND SGN(ShoulderDiff) = 0 THEN
        NewShoulder(n%) = Shoulder(n%+1) - BaseDiff/2
        IF SGN(ElbowDiff) = 1 THEN
            NewElbow(n%) = Elbow(n%) + BaseDiff/4
        ELSE
            NewElbow(n%) = Elbow(n%+1) + BaseDiff/4
        END IF
    ELSE IF (Shoulder(n%+1) < 0) AND SGN(ShoulderDiff) = 1 THEN
        NewShoulder(n%) = Shoulder(n%) + BaseDiff/2
        IF SGN(ElbowDiff) = 1 THEN
            NewElbow(n%) = Elbow(n%) + BaseDiff/4
        ELSE
            NewElbow(n%) = Elbow(n%+1) + BaseDiff/4
        END IF
    ELSE IF (Shoulder(n%+1) < 0) AND SGN(ShoulderDiff) = 0 THEN
        NewShoulder(n%) = Shoulder(n%) + BaseDiff/2
        IF SGN(ElbowDiff) = 1 THEN
            NewElbow(n%) = Elbow(n%) + BaseDiff/4
        ELSE
            NewElbow(n%) = Elbow(n%+1) + BaseDiff/4
        END IF
    END IF
END IF
IF NewElbow(n) > 180 THEN NewElbow(n) = 180
IF NewShoulder(n) < -30 THEN NewShoulder(n) = -30

An example of initial paths and their adapted paths after applying the rules developed in 8.6 is shown below in figure 8.29:-

```
70, 0, 100
150, -30, 100
90, 20, 150
30, 75, 125
```

**Figure 8.29:** Examples of Modified Paths.
In figure 8.29 two simple example paths are shown on the left and the result of applying the rules on page 232 are shown on the right. In both cases a via-point is generated which moves the shoulder and elbow through configurations which tend to reduce the friction on the robot base joint during motion.

8.8 Discussion and Conclusions.
A novel method of path improvement has been presented in this chapter. A method for calculating the manipulator dynamics model for a Mitsubishi RM 501 robot with rotary joints based on the Lagrange formulation was presented. The model was refined through a sequence of static tests, single joint and multiple joint motion tests. The model included the effects of gear transmission and friction.

From the simplified model, two simple rules for path improvement were developed. These rules were applied to adapt the paths of Mitsubishi robots. The method reprogrammed a path during the first sequence of a set of repeated paths by adding via-points which moved the robot through more profitable configurations.

The rules developed for the Mitsubishi robots were unexpected and in the case of the rule to reduce coulomb friction was the opposite of the expected result. The rules developed were specific to the Mitsubishi RM 501 robot but the new concept of using the manipulator dynamics to produce simple path reprogramming rules can be applied to any robot.
The results presented in section 8.7 suggested a maximum improvement of \( \approx 10\% \). In practice after considering 30 random paths, the average improvement was only 2.8\%. This is a satisfactory improvement but the adaption algorithms are coarse, and the selection of the via-points could be improved in future work. When the method was used with the path planning algorithms described in chapter six the software interfacing was clumsy and this could be improved in future work.

The software can be improved to interface more easily and quickly with the path planning algorithms and the addition of rules to include the inertias at the different joint angles would be a profitable next step.
Chapter Nine

DISCUSSION, CONCLUSIONS AND FUTURE WORK

9.1 Introduction.

This dissertation has presented solutions to the automatic robot path planning problem and demonstrated their implementation in real time. The algorithms can be expected to increase the autonomous ability of an industrial robot by automatically programming and reprogramming a controller in changing circumstances and environments. The research also explored methods of improving planned paths. Two new strategies for improvement were presented, one based on hardware monitoring of the servo amplifier currents and the second using simple rules developed from simplified robot dynamics equations. The research concentrated on methods of automatic path planning with constraints but during the work a novel parallel hierarchy control system evolved. The other original concepts presented in this dissertation included the following:-

The use of diverse models for different parts of the workplace. The models of the static environment were complex but accurate while the dynamic obstacles were modelled in a fast and simple way.

The use of simplified models of the robot dynamics to improve a robot path. The Engineering research work described in the literature has only used the dynamics at lower levels to adapt a robot trajectory. The work described in this dissertation has crossed the boundary between Computer Science and Engineering research into Robotics.

The use of monitored actuator torques to adapt a robot path. Although joint motor currents have been investigated in the literature, no attempt has ever been made in past work to use this information for path improvement.

The use of 2-D slices to enhance the speed of modelling obstacles in joint
The use of this simple and novel modelling method increased the processing speed of the path planner.

The remainder of this chapter will discuss the work described in chapters three to eight.

9.2 The Systems and the Apparatus.

The Apparatus: The apparatus was developed over a period of five years. During that time the state of the art of computer hardware has advanced. The author is continuing the work described in this dissertation as part of a Science and Engineering Research Council project in collaboration with Fast Filters (UK) and MIEKO Ltd (UK) using transputer arrays of parallel processors to replace the 80286 and 80287 processors.

The Systems: The sub systems worked together satisfactorily. The systems were designed to work in parallel in different computers and should move easily to the new parallel apparatus. The software is being rewritten in Occam from QuickBasic.

The Communications Sub-System: Investigation of the communication between sub systems revealed that communications speed was not a significant limiting factor compared with the time taken for the complex processing in each computer. For this reason and to use the interrupt facility, the two standard RS 232 ports available on each micro-computer were used. For future work, communications will be simpler as transputers and Occam were designed for fast communication between processors.
The G64 Bus: The G64 bus was adequate but proved to be a limiting factor as clock speeds through the bus were limited to 1 Mhz. It is the intention to expand the system to program three robots and to control a conveyor belt and other machinery. The G64 bus is not adequate for this purpose and will be replaced.

The Mitsubishi RM.501 Robot: The robot proved to be an interesting choice as the dynamics were unexpected and surprising. The robot had a limited reach and work area and for future work it is the intention to expand the system to use a Fanuc 600 series robot and either a Syke 600-5 or Unimation Puma robot.

The Robot Controller and Servo Amplifiers: The controller and servo amplifiers worked satisfactorily and they are now being redesigned for use with the Syke robot mentioned above.

9.3 Modelling of the robot and obstacles.

The Static Environment: For use with the Global Planning system, the static environment was modelled accurately as several polyhedra and was transformed into joint space before planning with dynamic obstacles. As this transformation took place once, at the beginning of the program, there was no time constraint. The use of different models for different parts of the work place is one of the novel concepts presented in this dissertation.

The robot geometric model: The robot geometric model consisted of two lines connected at the elbow joint surrounded by a skin a constant distance.
from this skeleton. This model was simple and proved to be fast.

**Dynamic Obstacles:** Several different models were considered for the dynamic obstacles and two were selected as they performed the transformation into joint space in the fastest times. The two models are discussed below:

(i) **Spheres:** The local path planner performed faster when using spheres compared with the other models. It should be noted that there must be a point at which increasing the number of spheres, in order to increase the accuracy of a model, becomes impractical and at this point the Data Processor could change to use a polyhedral model.

(ii) **2-D Slices:** Although spheres provided the fastest performance for the local path planning algorithms, 2-D slices proved to be faster to transform for the global path planner. This was due to a large amount of the complex processing being replaced by a simpler copying function.

**Other Models:** Of the other models considered, none performed favourably with the local path planner but the parallelepiped and the sphere provided favourable results with the global path planner.

**9.4 Image Data Processing and The Vision System.**

No claims for novelty are made for the vision system and many improvements could be made to this component.
The Configuration of the Apparatus: The configuration selected was a single camera placed above the work area. The camera placed at an angle and the use of two cameras were rejected because of the complexity of the processing required. Placing the camera directly above the workplace allowed a simple mapping in the X-Y plane and the later use of templates in real time required this simpler processing.

Initially the light source was placed behind the camera but later work used back lighting below the workplace. This part of the system could be improved and for future work using transputers and faster A/D boards, it is the intention to use a series of pictures for processing rather than single discrete pictures. This may allow the use of a light source above the work cell and the introduction of stereo vision techniques.

Low Level Vision techniques: The low level processing performed within the bounds of a small window (3x3) and had no knowledge of intensities outside of this window. All the methods performed satisfactorily, but Gray Level Weighting was excluded because of the time taken in processing. It is the intention to include this aspect in the future work using faster and dedicated parallel processors.

High Level Vision techniques: The higher level techniques aimed to interpret the data supplied in the form of edges and regions of some known object. This relied on some concept of 'intelligent' processing, that is the ability to extract pertinent information from a background of irrelevant detail. The edge detection method used was one of the simplest forms of 'intelligent' processing in that it extracted pertinent information regarding
the position and connection of edge points.

9.5 Automatic Robot Path Planning

2-SPACE Path Planning: Two methods were developed on the initial test rig (the prototype base), one a local and heuristic method and the other a global method. These methods worked satisfactorily in 2-SPACE when considering a simulated second joint and link, but it was not until the methods were extended to 3-SPACE using the Mitsubishi robot that physical results could confirm the expected results.

3-SPACE Local Heuristic Methods: The local heuristic method worked within the definition of real time used in this work but generally the planning took twice as long as the global method. As the advantage of a local planning method over a global planning method should have been a faster speed of operation, and this was not achieved, it is not intended to extend this method in the future.

The path planning process took less than 3 seconds and a large proportion of the planning time tended to be was taken up in considering the static environment. Any future work could consider methods of speeding up this part of the process. The method would be useful if a global model of the work area was not available, for example in undersea or space applications.

3-SPACE Global Method: The global method used a discrete range of values for each degree of freedom. In this work the range used for the robot was five degrees. If the range was extended to ten degrees, then the
number of units would be reduced by a factor of eight and the calculation
time could be reduced by a similar factor. The path planning process took
less than 1.4 seconds.

For simplicity, the range of values for each degree of freedom was set to the
same value. In practice particular degrees of freedom may be more
important than others. Smaller ranges of values could be used for the more
important robot axes, for example, the base angle $\theta_1$ in the case of the RM
501 robot.

For the future work, when manoeuvring a workpiece close to obstacles the
degrees of freedom of the gripper, (in this case $\theta_4$ and $\theta_5$) could be
considered. This would create a graph of more than three dimensions. The
disadvantage would be the size of graph, but it is intended to extend this
work to six degrees of freedom for two other robots.

A dynamic size of graph could be used. In large areas of either CLEAR or
BLOCKED nodes the unit ranges could be larger, but in the areas around the
surfaces of obstacles the graph could use smaller units. The processing to
achieve a dynamic graph may be complex, but it is the intention to
experiment with dynamic graphs on the new apparatus.

The performance of the system was encouraging in that the robot could
calculate and recalculate paths quickly ( < 9 seconds after introducing an
obstacle into the workplace). Performance for the path planning methods
was difficult to quantify as no other working systems existed to use as a
Bench Mark. Khatib(1986) presented work which described a collision
avoidance system which worked in near real time, and video film of the system working with a robot is held by the BBC. The work presented in this dissertation compared favourably with the system shown on the video.

**Trajectory Generation:** The trajectories generated were simple and it is the intention to use more complicated cubic (or higher order) splines for future work.

### 9.6 Path Improvement to Minimise Peaks in Joint Motor Currents

**Monitoring of the Motor Drive Currents:** Monitoring of motor drive currents was not in itself novel but the ways in which the information was used were original. Future work could consider other methods of measuring the joint torques.

**Adaption to reduce changes in joint direction:** This system successfully improved some robot paths and the method could be expected to extend the working life and service intervals of the servo motors and machinery, and in some cases increase the speed of operation. In the event of a collision, the system could be modified in future work to attempt to retrace the path and return to the previous set of trajectories by restoring the look-up table selection.

Scott(1984) reported that robot maintenance can be up to 10% of the original purchase price every year, and any reduction in maintenance costs or down time can have a substantial effect on the investment return or payback period. Minimising the current and torque transients reduced some
mechanical forces and stresses in the system. This should increase the up-time by extending the mean time between failure in a robot and maintenance may be less frequent. Minimising current transients resulted in energy conservation, allowing robots to run for longer periods from a given power source. This may be an important concept in any future mobile robots or for robots in inaccessible environments. Craig(1886) reported cases when this was important as time wasted recharging may be uneconomical and power pack replacement may be impractical.

The method of improvement was not successfully interfaced with a path planning system which included dynamic obstacles. It is not the intention to pursue this method further.

Collision detection: Limited collision detection was included. In any future work, some active compliance may also be achieved by considering the joint position errors and the joint forces. This active compliance may not be sufficient for difficult assembly tasks, but could aid specialised remote centre compliance devices or robots such as the I.B.M Selective Compliance Assembly Robot Arm (SCARA).

9.7 Path Improvement Considering the Robot Dynamic Equations.

Development of a Model of the Robot Dynamics: Specific rules needed to be produced for the unusual design of robot selected. This was achieved by carrying out tests on the robot to calculate the torques required to move the robot at various velocities and positions. The rules developed were specific to the Mitsubishi RM 501 robot but the methods and concepts could be
applied to any manipulator.

Minimum time paths are in general similar to minimum distance paths, but the shortest, most direct or most obvious path may not be the quickest path. The minimum time path can be expected to take unusual routes and the greater the degrees of freedom of a robot, the worse may be the link coordination from ad hoc motion planning. Conversely the greater the number of degrees of freedom, the more possibilities there are available for adaption and the greater the improvement possible.

**Adaption using the Dynamic Model:** The rules developed for the Mitsubishi robot were simple but had some effect, with on average a 2.8% improvement. The robot consisted of links which could be made to work together if kinetic energy and momentum were not wasted. The links exert reaction forces on one another that are generally harmful, but it may be possible to plan paths to minimise these effects, perhaps so that links can give helpful kicks to each other at the right times. \{parametric resonance\}. Future work will consider the inertia parameters for the Fanuc 600 series robot as a first step towards this improvement.

The robot was provided with a degree of autonomy and the result was similar to human workers adapting a repetitive task to make movements easier and less tiring. The methods crossed the boundary between Engineering and Computer Science research into robotics in that the manipulator dynamics were used at a level higher than that usually considered in Engineering research. Computer Science research has tended not to consider the dynamics of moving objects in path planning.
RELEVANT PUBLICATIONS AND PRESENTATIONS

BY THE AUTHOR


APPENDICES
Description of the Mixer stage: The mixer stage was a non-linear circuit based on a TL081 operational amplifier. The TL081 had proved itself as a robust and reliable op amp for control applications, directly replacing standard op amps such as the 741. The mixer had two feedback paths, one of which was non-linear and only took effect within predefined limits.

Unlike more conventional 'push pull' amplifiers, this design had four separate simple voltage supplies. Current supply to the d.c. motor in a forward or reverse direction is usually achieved with a single power supply. In this amplifier, current was applied by four separate high impedance output stages. The design was highly efficient and largely overcame the problems of crossover distortion and wasted power common in conventional servo motor power amplifiers.

The block diagram of the mixer circuit is shown in figure A.1. The feedback signal was split into two paths, both providing negative feedback. The outer loop was a simple linear feedback loop, but the inner loop was non-linear. The gain and range within which the non-linear circuit took effect was preset by the selection of suitable resistors and voltages. The outer loop was 'loose' (low gain), allowing high speed. The inner loop was high gain, providing 'tight' control within the limits.

The demand input to the mixer was an analogue voltage from the controller computer derived from a d/a circuit. The tacho signal was optional and was only used during the early work with the prototype robot base when a speed/voltage signal was available. The tacho input was mixed with the demand input and fed to the TL081 which was connected as a standard mixer using the negative input.

Description of the Mixer Circuit: The circuit diagram of the mixer is shown in Fig. A.2. R23 and R24 defined the gain of the demand input, in this case:

\[ \text{Demand} = \frac{R24}{R23} = \frac{43}{1.8} = 24 \]
The tacho signal was separated into two paths. One path was high gain, for low velocities close to the demanded position and consisted of the four diodes D3-D8 and resistors R25-R27. This inner loop only took effect when the tacho-feedback signal was within the two supplies at A and B, (in this case ± 10 volts).

A current flows from supply A to supply B. When the tacho input T was zero, half the current flows through D5 and D7, and half through D6 and D8. As the tacho input moves away from zero, say positive, the amount of current through D5 and R25 reduces and current through D7 and R27 increases. Thus, the voltage applied to the op-amp tends to increase in sympathy. The effect within the range was that the output to IC1 was via R26.

The high gain circuit only took effect within the limits of A and B, since when the tacho input was outside this range, say positive, no current can flow through D5, which was reverse biased and the voltage across was constant. So R26 and R24 defined the inner loop gain, in this case:

\[
\text{Loop gain} = \frac{42}{5.6} = 7.8
\]

A second path with a low gain for higher velocities where control was not so important, consisted of a simple resistor, R28. A capacitor, C1, was included to remove noise from the feedback signal. R28 and R24 defined the outer loop gain, in this case: Loop gain = \(\frac{43}{56} = 0.78\)

Resistor R29 and variable resistor VR1 adjusted the output of IC1 to zero for a zero input.

**Description of the Amplifier Design:** For rapid speed and fast responses, the amplifier had to be capable of delivering a substantial current. In this novel design, large demand voltage signals had current supplied from a higher voltage positive or negative supply (Fig. A.3). This would occur at high speed or for torques associated with large changes of force.

For smaller inputs, current was drawn from two lower voltage supplies. This was the case when the motor was at rest while sustaining a constant
When stationary the mechanical efficiency of the dc motor was near zero and the power associated with the current drawn from the power supply must be dissipated in the motor winding or control circuitry.

Because lower voltage supplies were used whenever possible, a power saving was achieved. Considering supplies ± 40 volts and ± 10 volts, the thermal dissipation of power in the controlling circuits was less than one eighth of that in a conventional circuit of typically ± 40 volts. The amplifier circuits of a conventional twin supply output stage are configured to have a low output impedance and are liable to excessive common current if both are simultaneously driven into conduction. This transition characteristic was crucial in traditional amplifier design. Basic design provides for a 'dead band' in which no conduction occurs. Crossover distortion was then present in the output waveforms.

The control circuits of the amplifier were configured to have a high output impedance, giving an output current which during the conducting phase of each circuit varied linearly with the applied demand signal. These outputs were safely connected. The lower voltage circuits were biased so that over a central range of input control signal, both circuits conducted. Outside the range, one or other circuit was cut off. (Fig. A.4). Within the range, the rate of change of output current with respect to the input signal was twice that outside the range and neither dead band nor discontinuity of current occurred.

**The Amplifier Circuit:** A block diagram of the circuit is shown in Figure A.3. The four circuits were configured so that the higher voltage amplifiers only conducted once the opposing low voltage amplifier had turned off. The amplifier circuit diagram is shown in Fig. A.5.

Two pairs of transistors control the four Darlington pair driver circuits. These are shown as TR1/TR2 and TR3/TR4 in the diagram. These pairs of transistors are configured so that neither pair can have both transistors conducting simultaneously. In the circuit shown, a change of input voltage at the emitters of about 1.4 volts was necessary to change from one state of conduction to the other.
Resistors R7 and R8 applied a bias voltage to the bases of TR1 and TR2. Similarly R9 and R10 bias TR3 and TR4. The resistors were set so that the low voltage drivers each conducted a moderate current, (eg 0.5 amps). The grounded base configuration of TR1, TR2 and TR3, TR4 caused the change in input voltage at the junction of R1 and R2 to result in a proportional change in the voltages across R11, R12, R13 and R14. If, as in this case, these resistors were equal in value, then the voltage changes were equal.

If the input voltage was steadily increased from zero then TR2 increased its collector output while driver TR3 decreased, cutting off at an input of 0.25 volts. Thus, TR3, the transistor controlling the + 10 volt supply, switched off completely before TR4, controlling the - 40 volt supply, began to conduct. TR1 and TR2 operated in a similar manner. As the input voltage continued to increase, only when the input was in excess of approximately 3 volts did TR4 and the - 40 volts supply begin to conduct.
Figure A.1: Block Diagram of the Mixer Circuit
Figure A.2: The Mixer Circuit Diagram
Figure A.3: Block Diagram Of the Servo Motor Amplifier
Figure A.4: Error/Output
Figure A.5: Circuit Diagram of the Servo Amplifier
Appendix B

The Detail of The Transformation Programs

This Appendix describes the detail of the two programs which transform the
dynamic obstacle models into joint space from cartesian space.

Setting of the lists: The first set of angles returned during the programs
TransformSphere.BAS and TransformSlice.BAS were for robot collision
with the centre of the sphere. The ForeFill flag was set for the "
Expandout" routines. The array NodeStatus was a status register which set
flags to give the status of a set of each node. This was set to BLOCKED
(ie bit 2 was set to 1). The angles were stored in an array called List1 as
shown below. The upper arm was tested to see if it would collide with the
sphere in any configuration and the nearest and furthest points of the
sphere were calculated.

NodeStatus(t1%, t2%, t3%) = 2
CALL PutonList(t1%, t2%, t3%)
NearestDistance% = L3% - Radius%
FurthestDistance% = L3% - Radius%

The upperarm was tested against the sphere model to see if it would collide
with the furthest point on the sphere. If it collided then the forearm was
not tested and the ForeFill flag was set to FALSE. The node was removed
from the list as shown below.

IF FurthestDistance% < UpperLength% THEN
ForeFill% = false%
CALL GetoffList(t1%, t2%, t3%)
END

The upperarm and sphere model were tested to see if a collision occurred
with the nearest point of the sphere. If a collision occurred, the ForeFill%
flag was set and the upperarm was set to point at the sphere's centre
(Sphθ). θ₃ was set to 180° and the angles were loaded onto the list. As
the upper-arm collided with the sphere, all possible θ₃ angles would also
collide. θ₃ was set to BLOCKED between its limits for the specified θ₁, θ₂

IF NearestDistance% < UpperLength% THEN
UpperFill% = true%
t2% = Sphθ%
CALL PutonList(t1%, t2%, t3%)
FOR Loop1% = LowLimit(t3%) TO HighLimit(t3%)
NodeStatus(t1%, t2%, Loop1%) = 2 ; Set to Collision
NEXT Loop1%
The flag register NodeStatus was tested to see if the particular node had already been tested by consulting bit 4 for the forearm test and bit 8 for the upperarm test.

\[
\begin{align*}
&\text{IF } (\text{NodeStatus}(t_1, t_2, t_3) \text{ AND } 4) = 4 \text{ THEN Foretested } = \text{true} \\
&\text{IF } (\text{NodeStatus}(t_1, t_2, t_3) \text{ AND } 8) = 8 \text{ THEN Upptested } = \text{true}
\end{align*}
\]

If the flags were not set and the NodeStatus was not set for an old obstacle, then the upperarm end point cartesian coordinates were calculated using the formula's for the forward kinematics solution described in section 3.7. If the forearm was to be tested then the Foretip position in cartesian coordinates was calculated and the NodeStatus flag was set to forearm tested.

The distance between the centre of the sphere and the end tip of robot was found and a test was conducted to see if the distance was less than the sphere radius plus the sphere model for the robot. If true, the node was placed onto list1 and set to BLOCKED. The same test took place for the upper-arm. If a collision occurred with the upper-arm then the procedure was repeated.

The subroutine Expandout tested all the nodes around the reference node using the subroutine TestPos. An example is shown below for the waist joint. The joint is set to $-5^\circ$, $+5^\circ$ and then returned to the reference node.

\[
\begin{align*}
&E1\% = E1\% - 1 \quad ; \text{setting to } -5^\circ \text{ of the ref node} \\
&\text{IF } E1\% > = \text{Low Limit} \text{ THEN CALL Testpos} \\
&E1\% = E1\% + 2 \quad ; +10^\circ \text{ now its } +5^\circ \text{ to the ref node} \\
&\text{IF } E1\% < = \text{High Limit} \text{ THEN CALL Testpos} \\
&E1\% = E1\% - 1 \quad ; \text{Resetting back to ref node} \\
&E2\% = E2\% - 1 \quad ; \text{As before except now its the upperarm and the process repeated}
\end{align*}
\]

The forearm (E3\%) was only tested for the forearm fill in by testing the flag TestType, which was passed from the subroutine FillIn. The subroutine expanded each node where a collision had occurred. Before this expansion the angles were removed from the list so that it was not expanded again. This was repeated until no further collisions occurred.

The first part of the subroutine checked whether there were two nodes on the list. If there were, the last one on the list would be the upper-arm node and this was transferred to a temporary array called list2. The nodes left on the list were the forearm nodes. The flag BothArm\% was set to true so
that after the forearm expansion the upper-arm node could be transferred back to list1.

The ForeFill flag was tested. If it was set to true then the FillIn for the Forearm was activated. The testtype flag was set to Foretest so that when calculating the forward kinematics in the subroutine TestPos, the routine knew that the Fortip needed to be calculated.

The node was removed from list1 and passed to the subroutine ExpandOut where the node was expanded and added to list1 if it collided with the obstacle. The routine continually removed nodes from list1, expanded them and tested for collisions, until no more collisions had occurred. (list1 became empty). This routine is shown below.

```plaintext
IF ForeFill% = true% THEN
    TestType% = ForeTest%
    DO
        FOR Loop1% = 1 TO NoNodesList1
            CALL GetoffList(t1%, t2%, t3%)
            CALL ExpandOut
        NEXT Loop1%
    LOOP UNTIL NoNodesList1 = 0
```

**Figure B.1:** The ExpandOut operation.
This routine was also used for the upper-arm with the flag TestType set to UpperTest.

The limits of $\theta_1$, $\theta_2$ and $\theta_3$ at which collisions occurred was found and this information was used when setting the NodeStatus collisions to old obstacle. This prevented the loops from repeating the limits of all three angles. This saved 1.2 seconds in interpreted Quick Basic. The NodeStatus were searched to find collisions (ie bit 2 set). These NodeStatus were then changed to old obstacle (ie all other bits were set to zero except bit one), otherwise the NodeStatus was reset to zero as shown below.

```
IF NodeStatus ( t1%, t2%, t3%) AND 2 = 2 THEN
    NodeStatus ( t1%, t2%, t3%) = 1
ELSE
    NodeStatus ( t1%, t2%, t3%) = 0
END
```
Appendix C

**Edge Following and Line Fitting**

The routines used for the limited work with stereo vision and the single camera at an angle, started at an arbitrary edge point and scanned around its immediate 8 neighbours in the plane to find a linking point. If no neighbouring point was found, the edge was said to be complete and another starting point was found. If more than one edge point was found (such as at a corner or meeting point) the routine followed one of the points whilst placing the other onto a list for future tracing. The program retained the gradient of the line it was fitting and searched its nearest neighbouring pixels for a point which continued this gradient. If this was not found, then an arbitrary pixel within the 8 was chosen whilst the pixels not selected were stored on a separate list which was expanded later. Each pixel checked was reset to a value which caused it to be undetectable to the program and thus not retraced. This technique provided an array of linked x and y edge points which was used to generate straight line information for a parallelepiped description of the object.

Line fitting used the data obtained from the edge trace routine to mathematically define vertices and their crossing points. The procedure used the 'least squares' process to match straight lines, fitting the \( y = mx + c \) formula from the edge descriptions generated from the local operator in the edge detection sequence.

\[
\begin{align*}
v &= y - y_1 = ax_1 + b - y_1 \\
v &= ax_1 + b - y_1 \\
v^2 &= (ax_1 + b - y_1)^2 \\
\Sigma v^2 &= \Sigma(ax + b - y)^2 = S
\end{align*}
\]

a and b were selected so that S was zero

\[
\begin{align*}
\delta S/\delta a &= \Sigma 2(ax + b - y) x \\
derivative &= 0 \text{ if } a \Sigma x^2 + b \Sigma x - \Sigma xy = 0
\end{align*}
\]

also

\[
\begin{align*}
\delta S/\delta b &= \Sigma 2(ax + b - y) \\
derivative &= 0 \text{ if } a \Sigma x + b n - \Sigma y = 0
\end{align*}
\]
where \( n \) is the number of points to be fitted.

\[
y = ax + b
\]

\[
Figure C.1: The Line Fitting Process.
\]

From these equations, \( a \) and \( b \) may be found:

\[
a = \frac{n \Sigma xy - \Sigma x \Sigma y}{n \Sigma x^2 - (\Sigma x)^2}
\]

and

\[
b = \frac{\Sigma y - x \Sigma x}{n}
\]

regression coefficient = 

\[
\frac{nxv - \Sigma x \Sigma y}{\sqrt{(n \Sigma x^2 - (\Sigma x)^2)(n \Sigma y^2 - (\Sigma y)^2)}}
\]
Appendix D

The Detail Of the Manipulators

This appendix includes design drawings of the two manipulators, the prototype robot base with simulated arm, and the Mitsubishi RM 501 robot.
Appendix D

The Detail Of the Manipulators

This appendix includes design drawings of the two manipulators used during the work described in this dissertation:

(i) The prototype robot base with simulated arm.
(ii) The Mitsubishi RM 501 robot.

The prototype robot base with simulated arm.

Figure D.1: Page 265  Design drawings of the Prototype Robot Base.
Figure D.2: Page 266  Design drawings of the Prototype Robot Base.
Figure D.3: Page 267  3 Projections of the Simulated link.
Figure D.4: Page 268  3-D View of the Prototype Robot Base, showing the positioning of the simulated links.
Figure D.5: Page 269  Side view of the Prototype Robot Base, showing the positioning of the simulated links.
Figure D.6: Page 270  3-D View of the Prototype Robot Base, showing the positioning of the simulated links.

The Mitsubishi RM 501 robot.

Figure D.7i: Page 271  The Outer Appearance of the Mitsubishi Robot.
Figure D.7ii: The interior of the Mitsubishi Robot.
Figure D.8: Page 272  The Outer Dimensions and Specifications of the Mitsubishi Robot.
Figure D.9: Page 273  The Range of movement of the Robot.
Figure D.3: 3 Projections of the Simulated link.
Figure D.4: 3-D View of the Prototype Robot Base, showing the positioning of the simulated links.
Figure D.5: Side view of the Prototype Robot Base, showing the positioning of the simulated link.
Figure D.6: 3-D View of the Prototype Robot Base, showing the positioning of the simulated links.
(1) Outer appearance

![Diagram of Mitsubishi Robot]

Outer appearance

(2) Interior of arm

![Diagram of Mitsubishi Robot Interior]

Interior of arm

Figure D.7i: The Outer Appearance of the Mitsubishi Robot.

Figure D.7ii: The interior of the Mitsubishi Robot.
### Outer Dimension Diagram

**Item** | **Specification**
--- | ---
Structure | Five degrees of freedom, Vertical multi-joint type
Range of movement |  
- Waist rotation: 300°
- Shoulder rotation: 130°
- Elbow rotation: 90°
- Wrist pitch: ±90°
- Wrist roll: ±180°
Permissible handling weight | max. 1.2 kg (includes weight of hand)
Maximum synthesis speed | 400 mm/sec (wrist tool surface)
Position repeat accuracy | ±0.5 mm (wrist tool surface)
Drive system | Electroservo drive by a DC servomotor
Main unit weight | about 27 kg

**Note:** The permissible handling weight (1.2 kg) is the value at a point 100 mm from the wrist tool surface.

---

**Figure D.8:** The Outer Dimensions and Specifications of the Mitsubishi Robot.
The range of movements when the hand is not attached is as follows.

Figure D.9: The Range of movements of the Mitsubishi Robot.
REFERENCES


AKMAN V 1985(PhD) "Shortest Paths Avoiding Polyhedral Obstacles in 3-D Euclidean Space", Rensselaer Polytechnic Institute, New York, USA.

AN, ATKENSON and HOLLERBACH 1986 "Estimation of Inertial Parameters of Rigid Body links of Manipulators", AI Memo 887, MIT, Feb 86.


Balding N.W. 1987(PhD) "Real Time Path Planning for Robot Arms", Durham (UK).


BRADY et al. 1979 "Planning and Execution of Straight Line Manipulator Trajectories". IBM Research Journal No 23.


BRAID I.C 1973(PhD) "Designing with Volumes", Cambridge University, UK.


BROOKS R.A. 1983(a) "Find Path for a Puma-Class Robot", Proc AAAI, Washington, USA, pp40-44.


BROOKS & LOZANO-PEREZ 1983(c) "A Subdivision Algorithm in Configuration Space for FindPath with rotation", Proc of Int Joint Conf AI, Karlsruhe (Germany), pp 799-806.

BROOKS R.A. 1983(d) "Planning Collision Free Motions for Pick and Place Operations",


CESARONE J. 1988(PhD) "Manipulator Collision Avoidance by Dynamic Programming", North-Western University.


DUBOWSKY, NORRIS AND SHILLER 1986 "Time optimal trajectory planning for robot manipulators with obstacle avoidance: A CAD Approach", IEEE Int Conf on Robotics and
Automation. Apr 7-10", paper no 276.

DU PONT P.E. 1988(PhD) "Planning Collision-Free Paths for kinematically redundant robots by selectively mapping configuration space",


FAVERJON B 1984 "Obstacle avoidance using an OCTREE in the configuration space of a manipulator", IEEE Conf on Robotics, Atlanta, pp 504-512.


HADDAD F.B 1985(PhD) "Design and Performance of a Position Controlled Manipulator", University of Sussex, UK.

HAJASE A J 1987(PhD) "Optimal Robot Trajectory Planning using Dynamic Models", Syracuse University, USA.


HASEGAWA T 1985 "Collision avoidance using characterized description of free space", Conf on Advanced Robotics, Tokyo, pp 69-76.

HONG Z 1986(PhD) "Design and implementation of a Robot Force and Motion Server", Purdue University, USA.


HWANG K.Y 1988(PhD) "Robot Path Planning using Potential Field Representation", University of Illinois, USA.


KAHN W.E 1969(PhD) "The near minimum time control of Open Loop Articulated Kinematic Chains" Comp Sci Dept, Stanford University, USA.


KUMAR S. 1988(PhD) "Motion Planning With Obstacles and Dynamic Constraints", Cornell University, USA.


MORAVEC h.p 1979 "Visual mapping by a robot rover", *Proc 6th Int Joint Conf on AI (Tokyo)* pp 598-600.


OLSON and BEKEY 1985 "Identification of Parameters in Models of Robots With Rotary Joints", *Proc of IEE Conf on Robotics and Automation, St Louis, pp 1045 - 1049*.

ORIN and SCRADER 1984 "Efficient Jacobian Determination for Robot Manipulators".
Robotics Research: The 1st International Symposium, MIT Press


PIEPER D.L 1969(PhD) "The Kinematics of Manipulators under Computer Control", Stanford University, (USA).


SCHWARTZ & SHARIR 1983(b) "On the Piano Movers Problem II - General Techniques for computing topological properties of real algebraic manifolds", Advances in Applied Mathematics, 4, pp 298-351.


SCHWARTZ & SHARIR 1984 "On the Piano Movers Problem V: The case of a rod moving in 3D space amidst polyhedral obstacles", Communications in Pure and Applied Maths, No 37, pp 815-848.


SHIN & MCKAY 1986 A dynamic programming approach to trajectory planning of robot manipulators


SYNDER WE. 1985 "Computer Interfacing and Control of Industrial Robots", Prentice Hall International


UDUPA S.M 1977(PhD) "Collision detection and avoidance in computer controlled manipulators",


**SPECIFICATIONS OF THE MOVE MASTER II RM-501 MODEL ROBOT**

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Five degree of freedom</td>
</tr>
<tr>
<td></td>
<td>Vertical multi-joint type</td>
</tr>
<tr>
<td><strong>Range of movement</strong></td>
<td></td>
</tr>
<tr>
<td>Waist rotation</td>
<td>300°</td>
</tr>
<tr>
<td>Shoulder rotation</td>
<td>130°</td>
</tr>
<tr>
<td>Elbow rotation</td>
<td>90°</td>
</tr>
<tr>
<td>Wrist pitch</td>
<td>+/- 90°</td>
</tr>
<tr>
<td>Wrist roll</td>
<td>+/- 180°</td>
</tr>
<tr>
<td><strong>Permissible handing weight</strong></td>
<td>maximum 1.2 kg</td>
</tr>
<tr>
<td></td>
<td>(includes hand weight)</td>
</tr>
<tr>
<td><strong>Maximum sythesis speed</strong></td>
<td>400 mm/sec</td>
</tr>
<tr>
<td></td>
<td>(wrist tool surface)</td>
</tr>
<tr>
<td><strong>Main unit weight</strong></td>
<td>about 27 kg</td>
</tr>
<tr>
<td><strong>Position detection</strong></td>
<td>optical transducer</td>
</tr>
<tr>
<td><strong>Actuator</strong></td>
<td>DC 12/24 V servomotor with brushes</td>
</tr>
</tbody>
</table>